Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Problem M

Seminar
Algorithmen für Programmierwettberwerbe

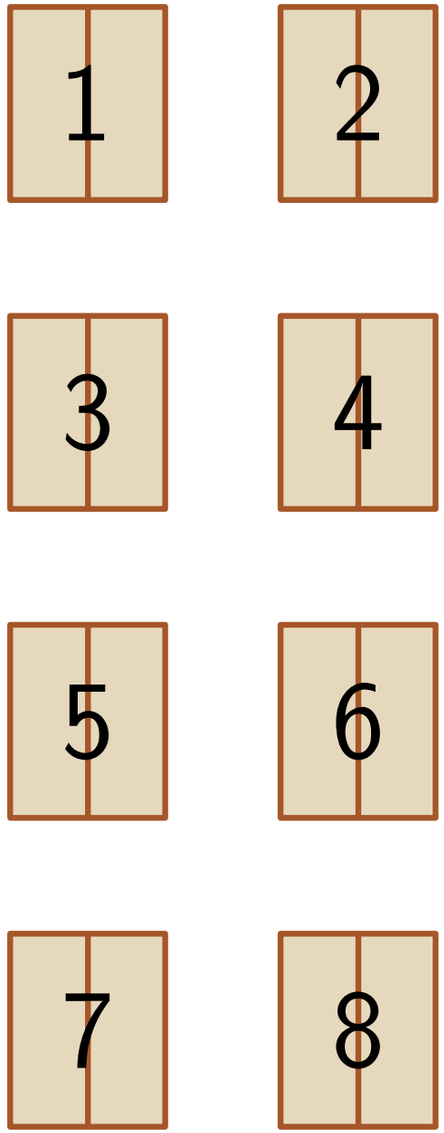
Sommersemster 2021

Jonas Zeier

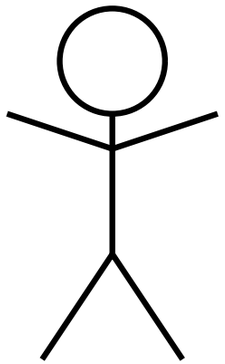
Martin Herold

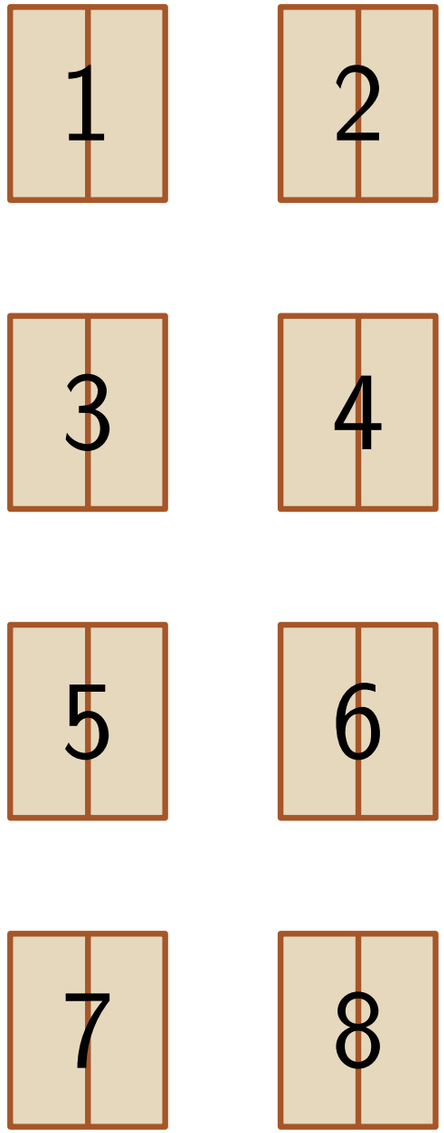
Prince of Python

Prince of Python

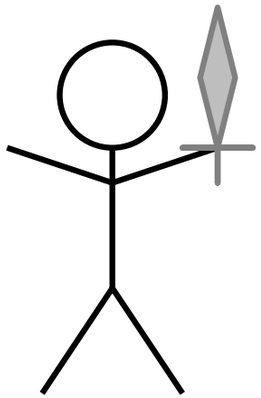


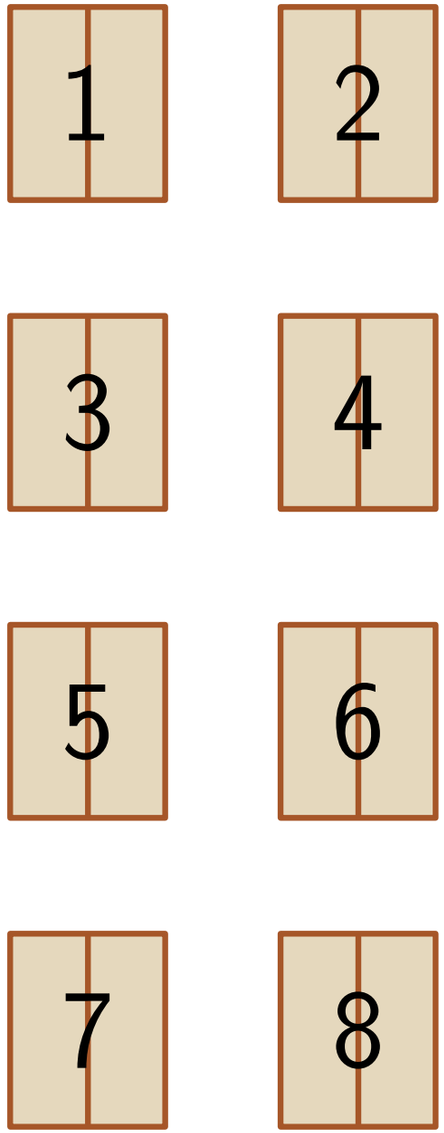
Prince of Python



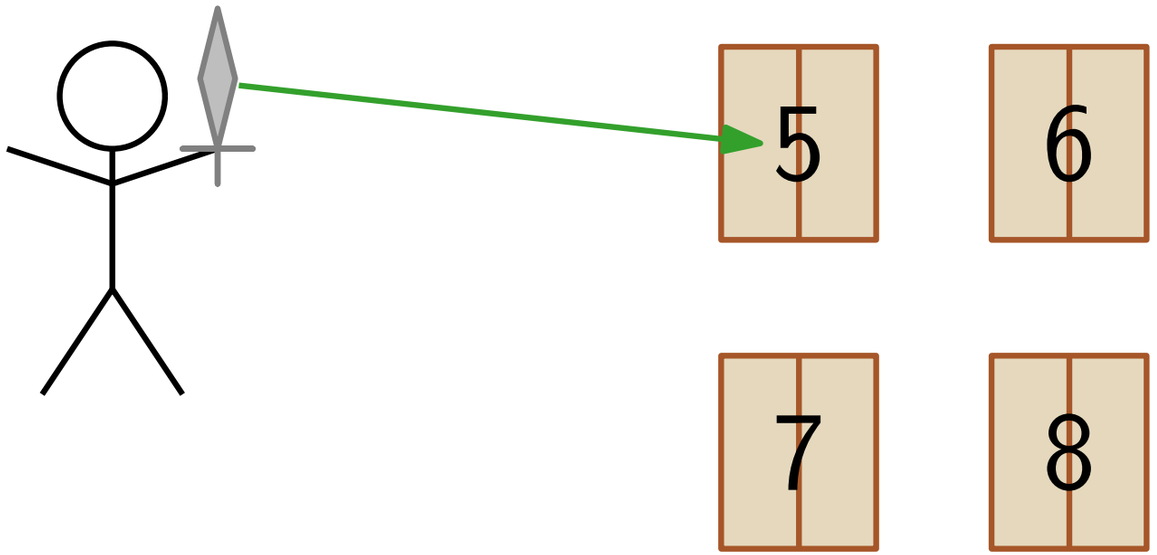


Prince of Python

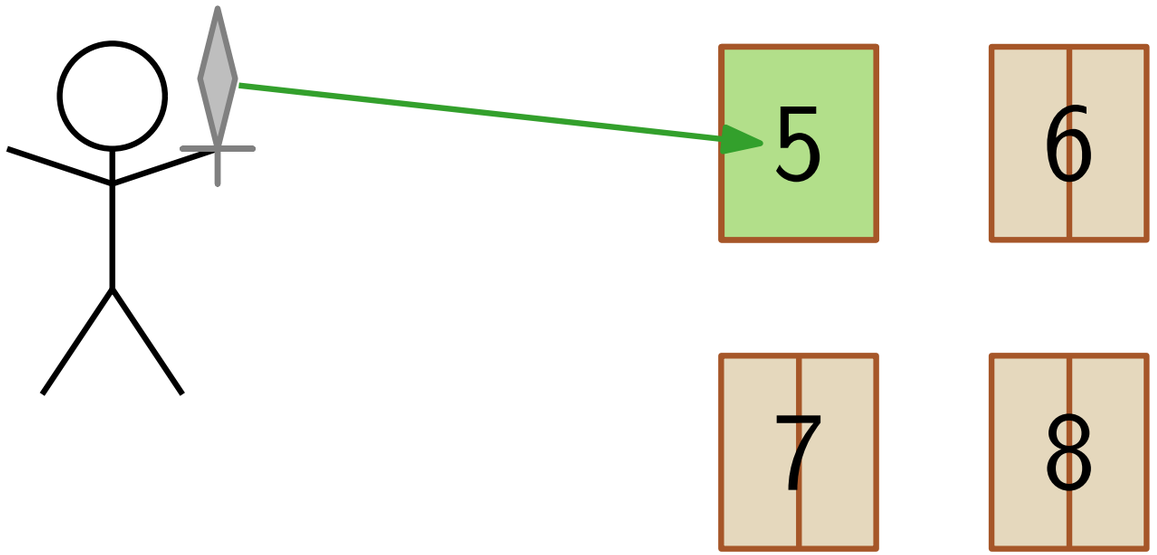




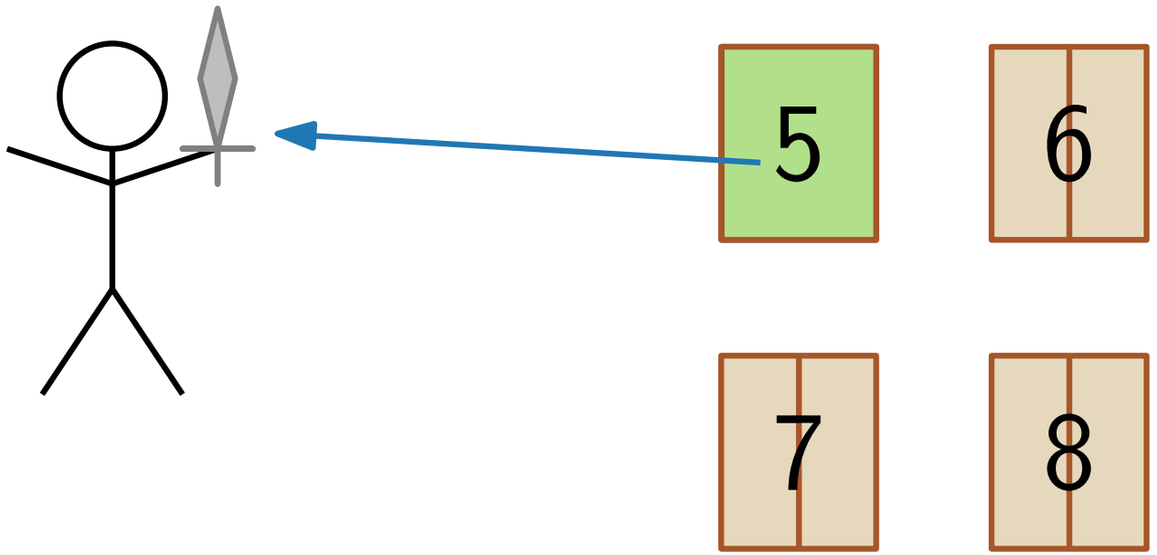
Prince of Python



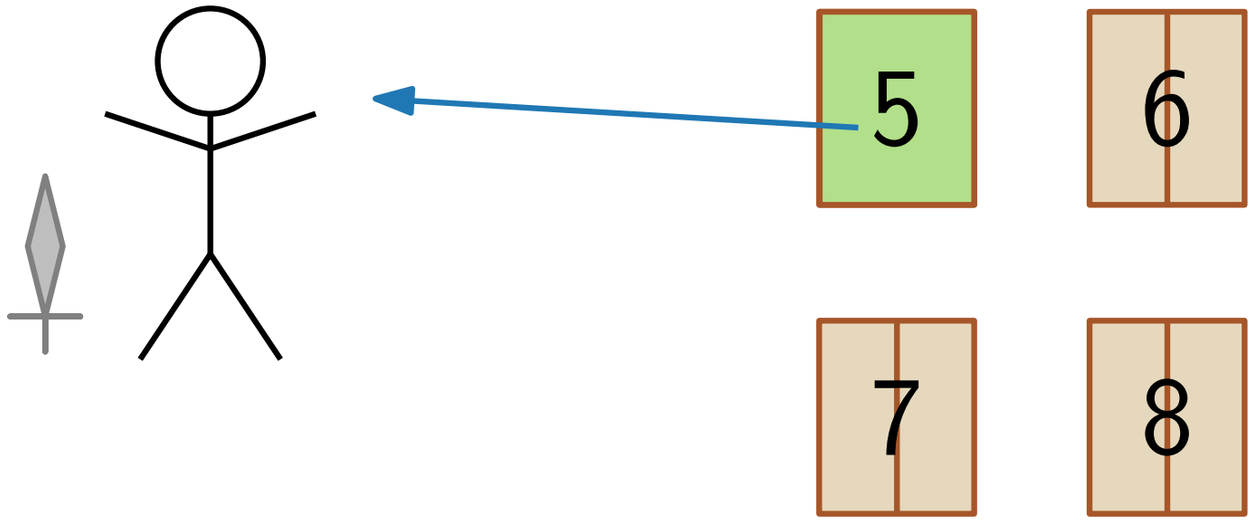
Prince of Python



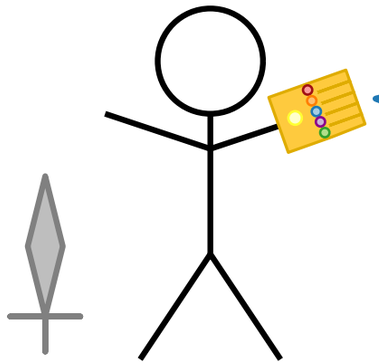
Prince of Python



Prince of Python



Prince of Python



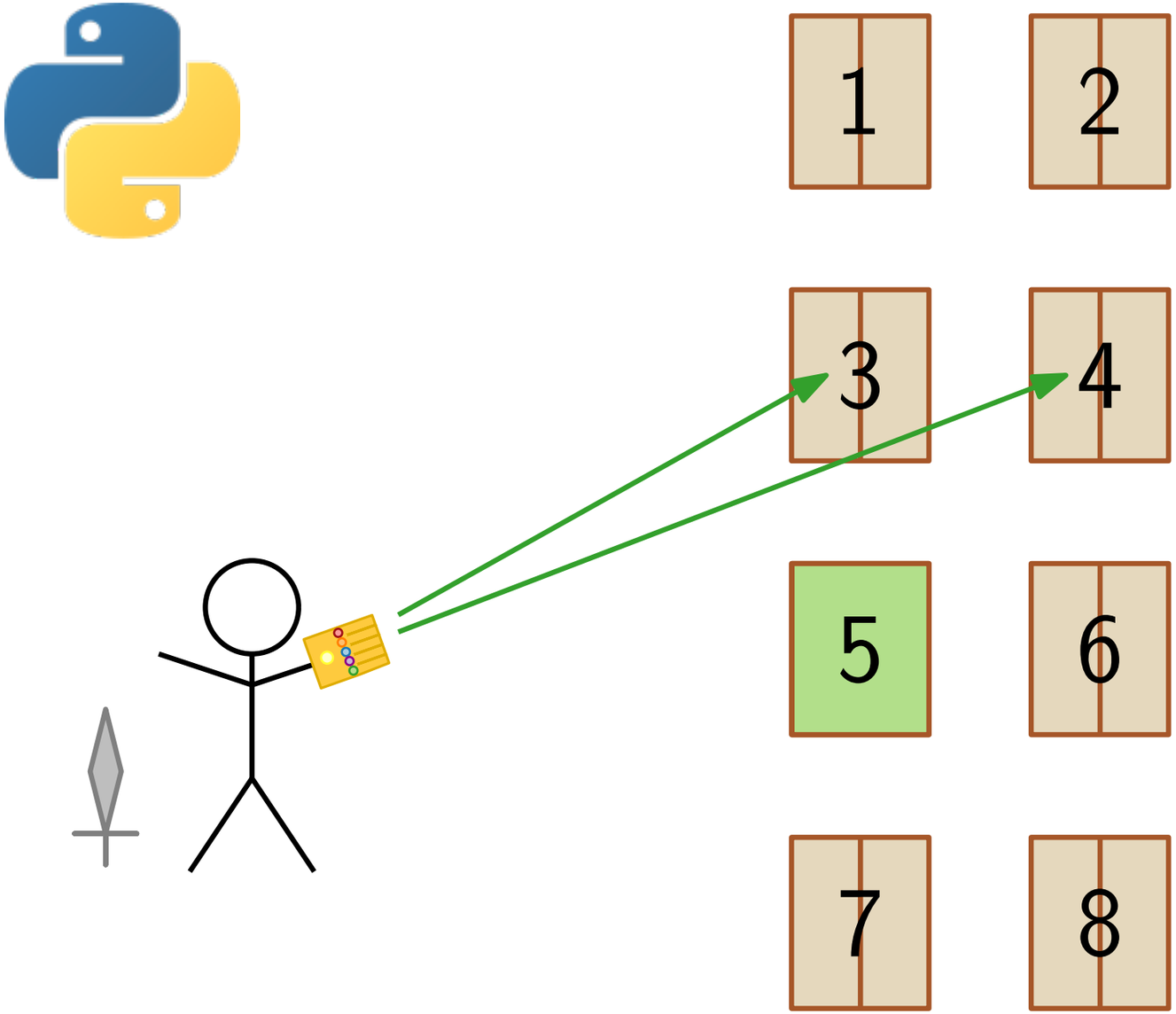
1 2

3 4

5 6

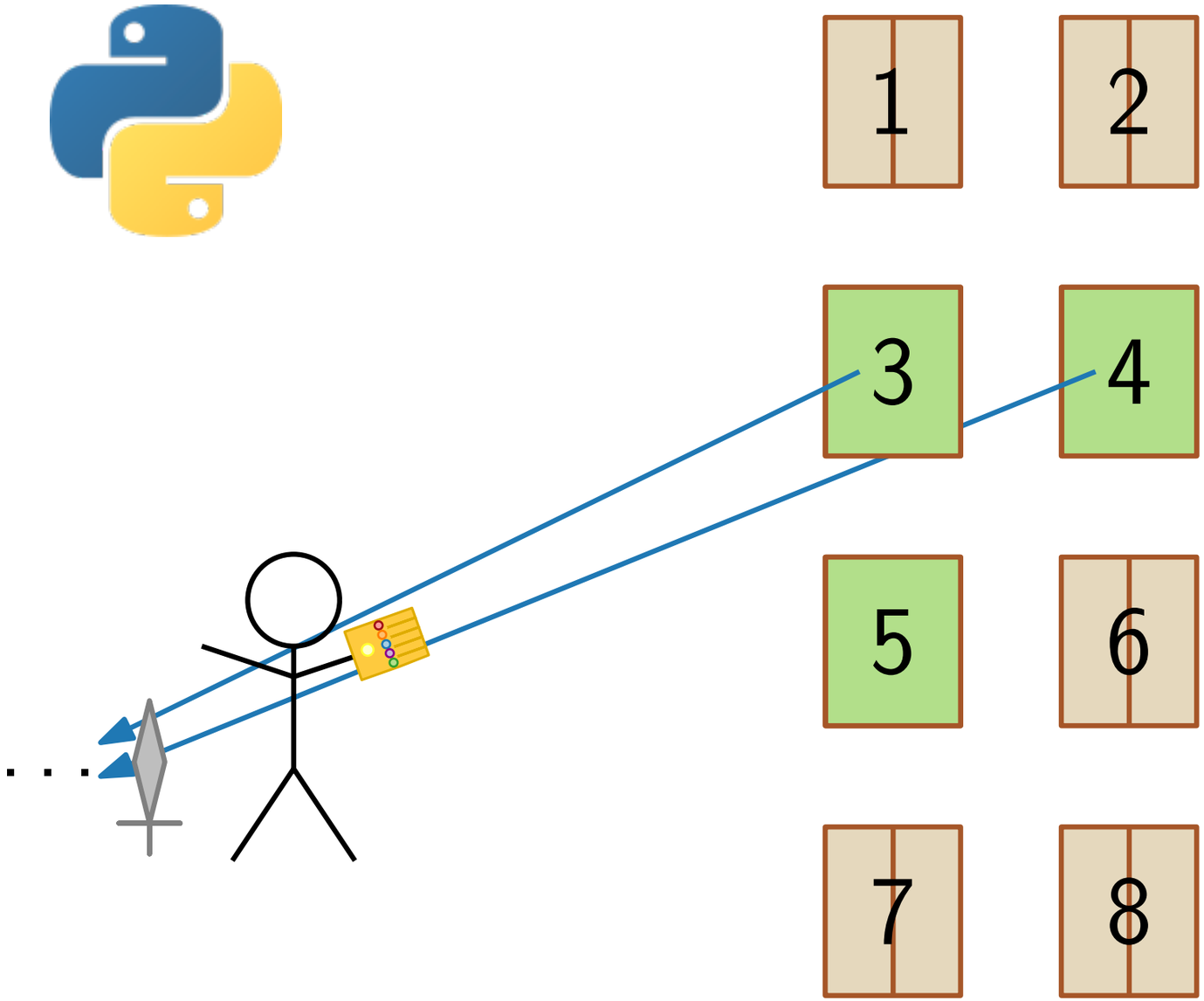
7 8

Prince of Python

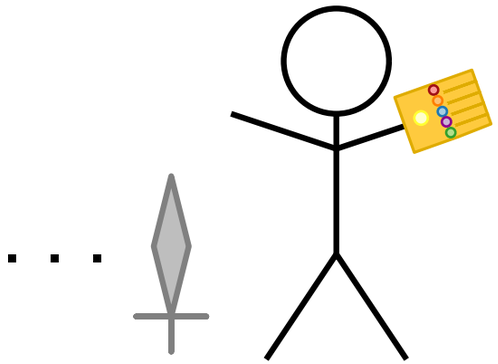


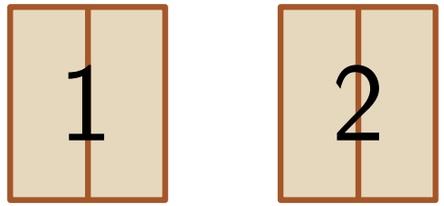
Prince of Python

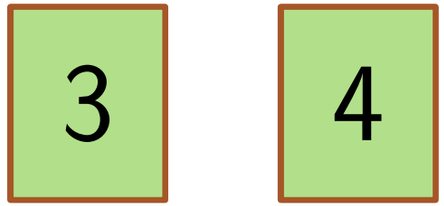
Prince of Python

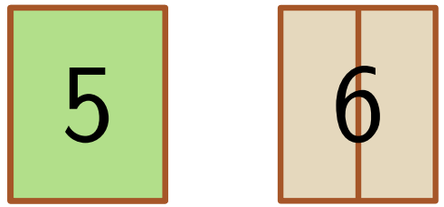


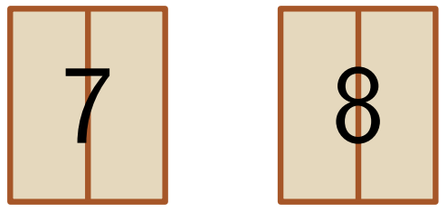
Prince of Python



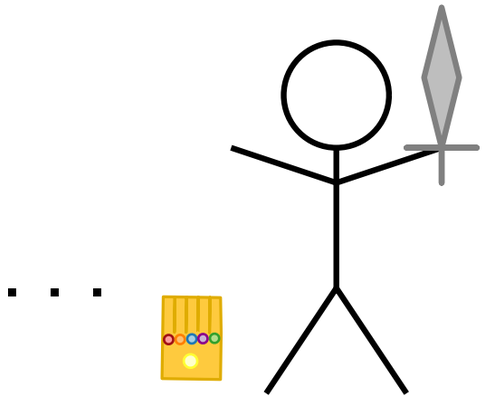


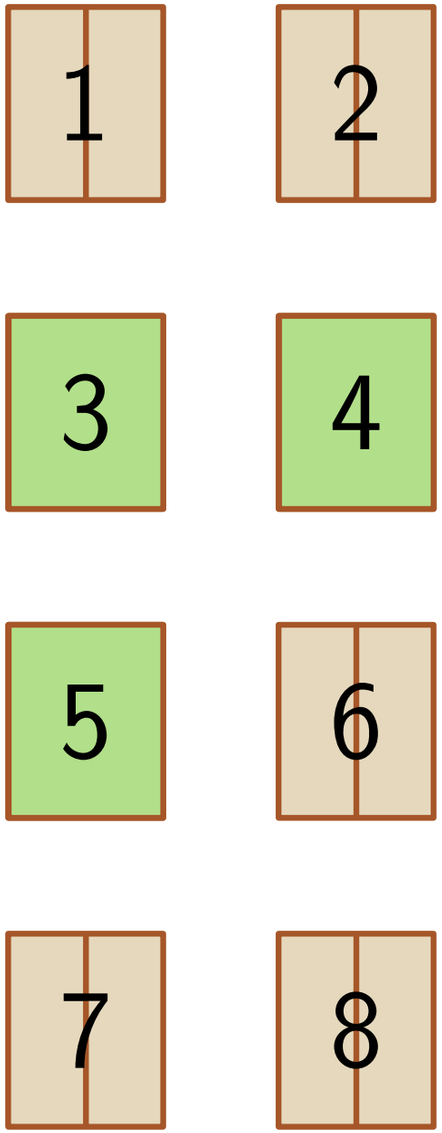




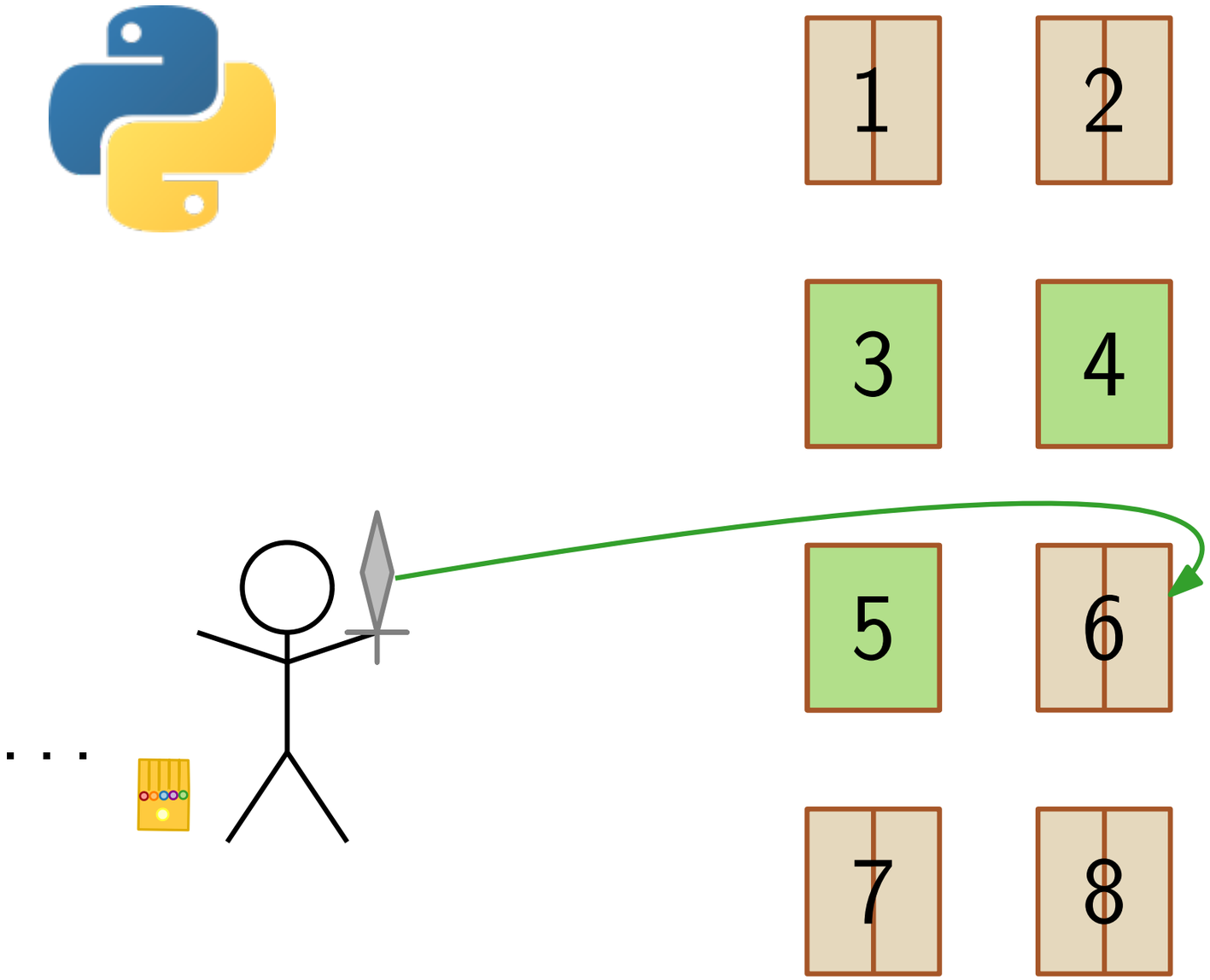


Prince of Python

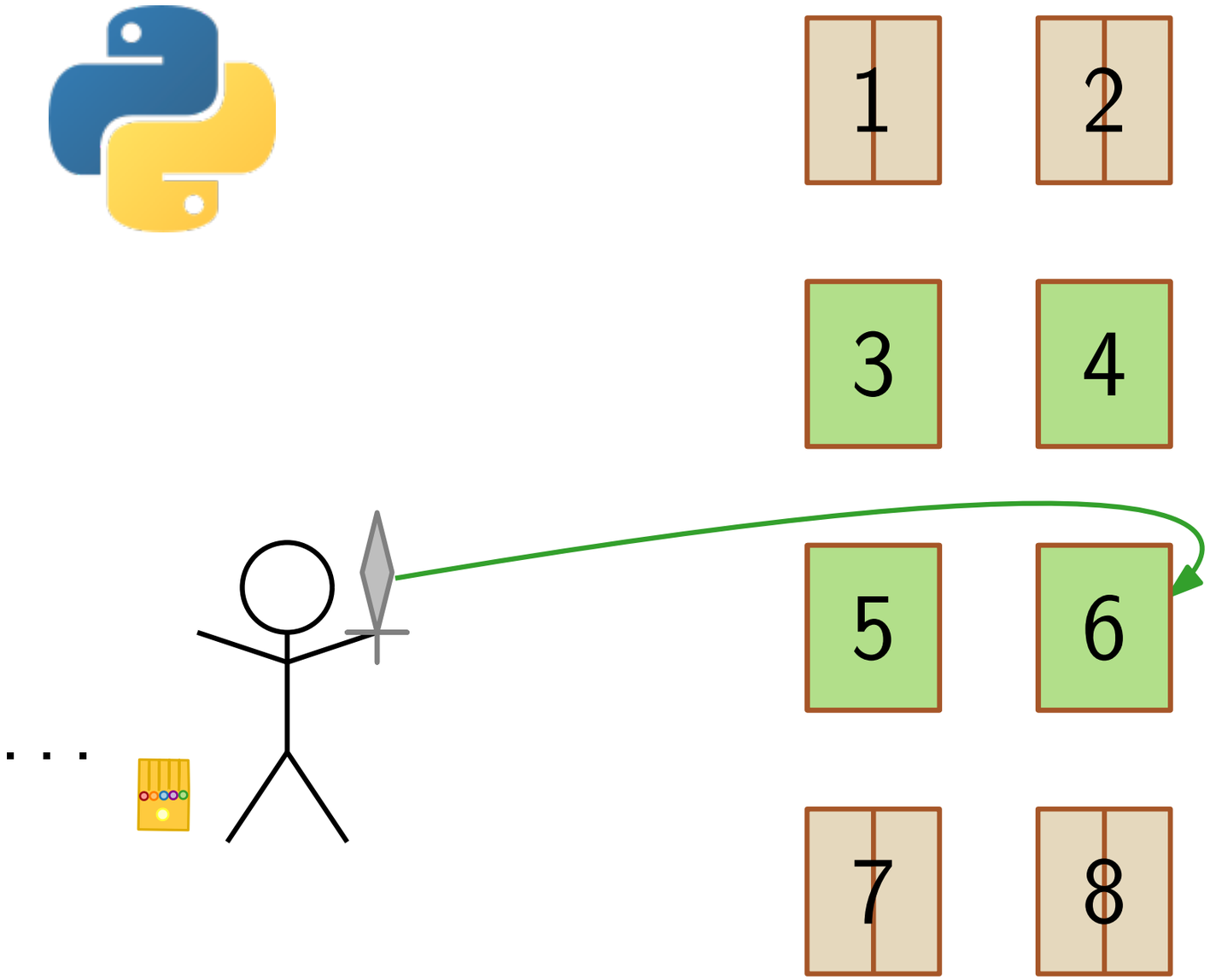




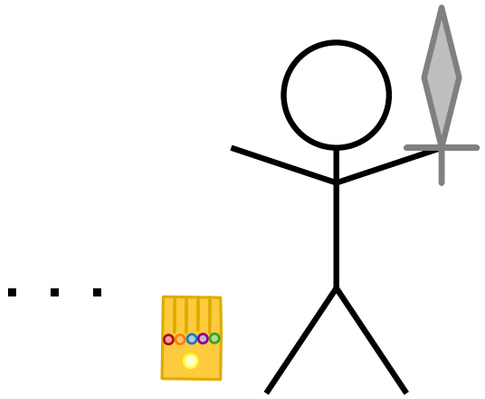
Prince of Python

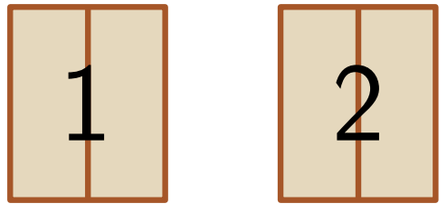


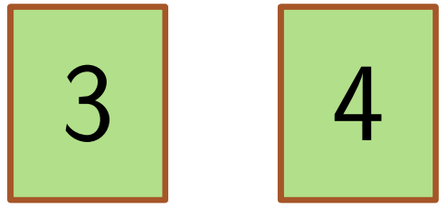
Prince of Python

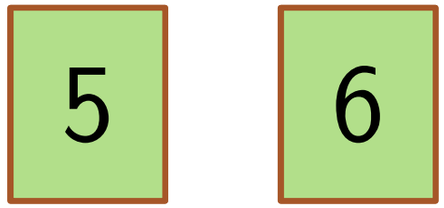


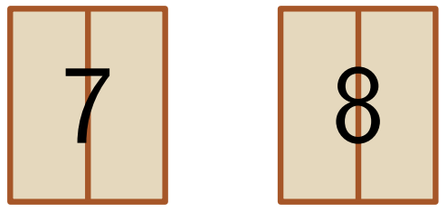
Prince of Python





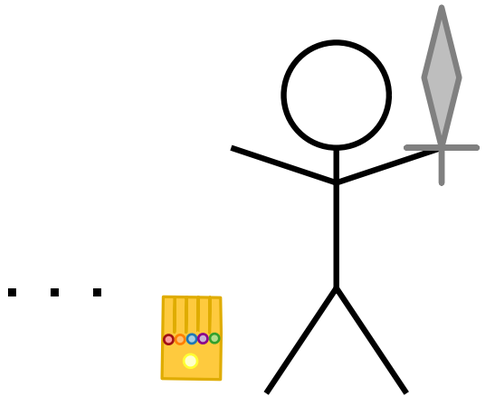






1.	
2.	
3.	
4.	

Prince of Python



1 2

3 4

5 6

7 8

1.	
2.	
3.	
4.	

Problemstellung

Problemstellung

Eingabe:

Problemstellung

Eingabe:

- Erste Zeile: Anzahl Level n

Problemstellung

Eingabe:

- Erste Zeile: Anzahl Level n 3

Problemstellung

Eingabe:

- Erste Zeile: Anzahl Level n

n

3

Problemstellung

Eingabe:

- Erste Zeile: Anzahl Level n
- n weitere Zeilen:

n

3

Problemstellung

Eingabe:

- Erste Zeile: Anzahl Level n
- n weitere Zeilen:
  - Abkürzungsitem x_i

n
3
1

Problemstellung

Eingabe:

- Erste Zeile: Anzahl Level n
- n weitere Zeilen:
  - Abkürzungsitem x_i
  - Abkürzungszeit s_i

n
3
1

Problemstellung

Eingabe:

- Erste Zeile: Anzahl Level n
- n weitere Zeilen:
  - Abkürzungsitem x_i
  - Abkürzungszeit s_i

n
3
1 1

Problemstellung

Eingabe:

- Erste Zeile: Anzahl Level n
- n weitere Zeilen:
  - Abkürzungsitem x_i
  - Abkürzungszeit s_i
  - $n+1$ weitere Zahlen: $a_{i,0}, a_{i,1}, \ldots, a_{i,n}$
    Reguläre Zeit für Level i mit Item j: $a_{i,j}$.

n
3
1 1

Problemstellung

Eingabe:

- Erste Zeile: Anzahl Level n
- n weitere Zeilen:
  - Abkürzungsitem x_i
  - Abkürzungszeit s_i
  - $n+1$ weitere Zahlen: $a_{i,0}, a_{i,1}, \ldots, a_{i,n}$
    Reguläre Zeit für Level i mit Item j: $a_{i,j}$.

n
3
1 1 40 30 20 10

Problemstellung

Eingabe:

- Erste Zeile: Anzahl Level n
- n weitere Zeilen:
  - Abkürzungsitem x_i
  - Abkürzungszeit s_i
  - $n+1$ weitere Zahlen: $a_{i,0}, a_{i,1}, \ldots, a_{i,n}$
    Reguläre Zeit für Level i mit Item j: $a_{i,j}$.

n

3

1 1 40 30 20 10

3 1 95 95 95 10

2 1 95 50 30 20

Problemstellung

Eingabe:

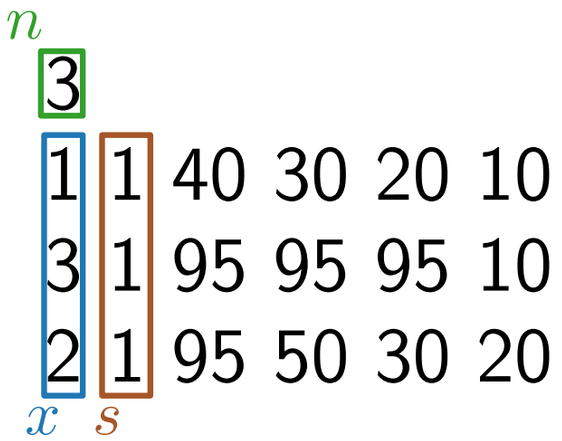
- Erste Zeile: Anzahl Level n
- n weitere Zeilen:
  - Abkürzungsitem x_i
  - Abkürzungszeit s_i
  - $n+1$ weitere Zahlen: $a_{i,0}, a_{i,1}, \ldots, a_{i,n}$
    Reguläre Zeit für Level i mit Item j: $a_{i,j}$.

n
3
1 1 40 30 20 10
3 1 95 95 95 10
2 1 95 50 30 20
x

Problemstellung

Eingabe:

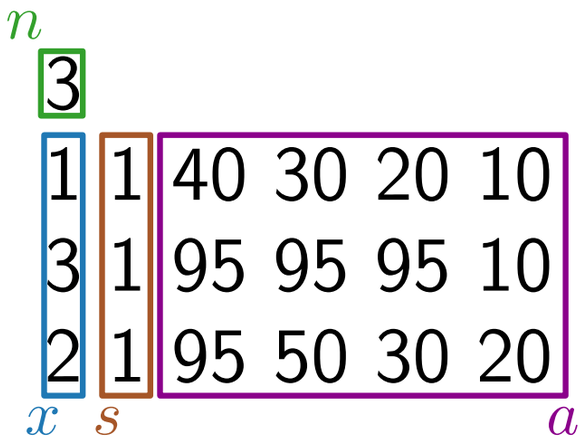
- Erste Zeile: Anzahl Level n
- n weitere Zeilen:
  - Abkürzungsitem x_i
  - Abkürzungszeit s_i
  - $n + 1$ weitere Zahlen: $a_{i,0}, a_{i,1}, \ldots, a_{i,n}$
    Reguläre Zeit für Level i mit Item j: $a_{i,j}$.



Problemstellung

Eingabe:

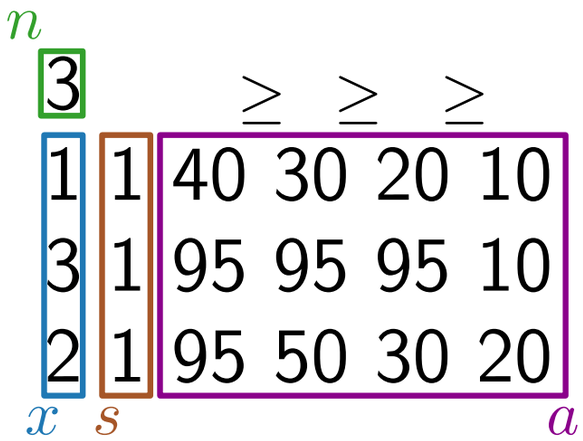
- Erste Zeile: Anzahl Level n
- n weitere Zeilen:
  - Abkürzungsitem x_i
  - Abkürzungszeit s_i
  - $n + 1$ weitere Zahlen: $a_{i,0}, a_{i,1}, \ldots, a_{i,n}$
    Reguläre Zeit für Level i mit Item j: $a_{i,j}$.



Problemstellung

Eingabe:

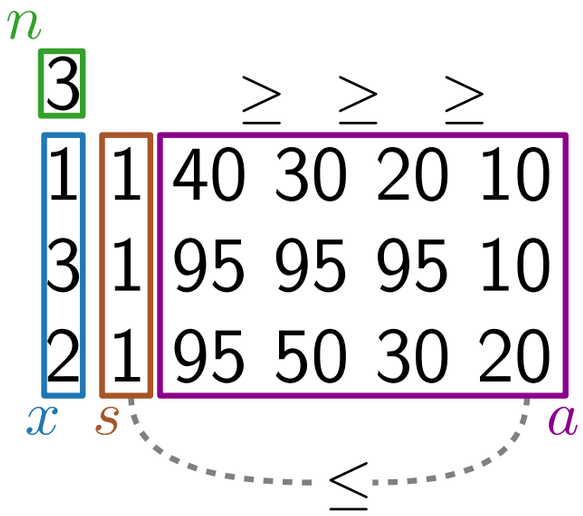
- Erste Zeile: Anzahl Level n
- n weitere Zeilen:
  - Abkürzungsitem x_i
  - Abkürzungszeit s_i
  - $n+1$ weitere Zahlen: $a_{i,0}, a_{i,1}, \ldots, a_{i,n}$
    Reguläre Zeit für Level i mit Item j: $a_{i,j}$.



Problemstellung

Eingabe:

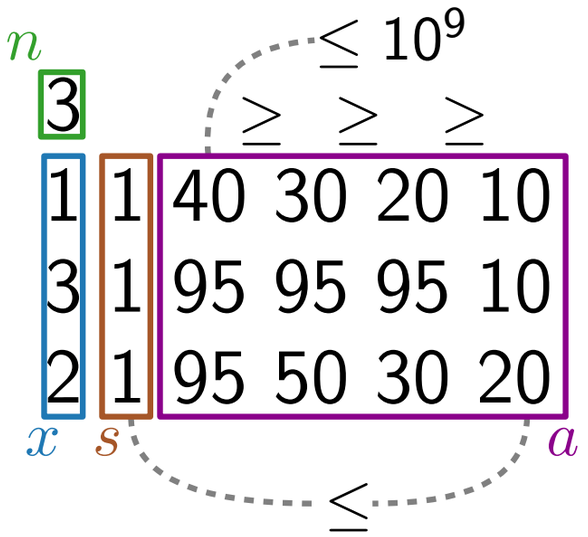
- Erste Zeile: Anzahl Level n
- n weitere Zeilen:
  - Abkürzungsitem x_i
  - Abkürzungszeit s_i
  - $n+1$ weitere Zahlen: $a_{i,0}, a_{i,1}, \ldots, a_{i,n}$
    Reguläre Zeit für Level i mit Item j: $a_{i,j}$.



Problemstellung

Eingabe:

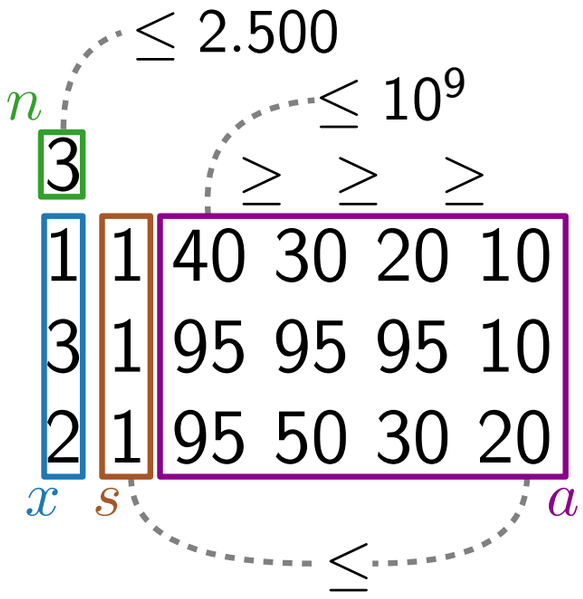
- Erste Zeile: Anzahl Level n
- n weitere Zeilen:
  - Abkürzungsitem x_i
  - Abkürzungszeit s_i
  - $n+1$ weitere Zahlen: $a_{i,0}, a_{i,1}, \ldots, a_{i,n}$
    Reguläre Zeit für Level i mit Item j: $a_{i,j}$.



Problemstellung

Eingabe:

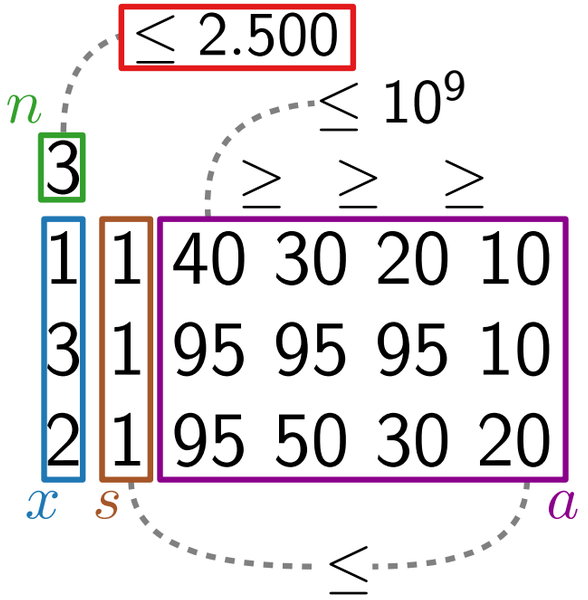
- Erste Zeile: Anzahl Level n
- n weitere Zeilen:
  - Abkürzungsitem x_i
  - Abkürzungszeit s_i
  - $n+1$ weitere Zahlen: $a_{i,0}, a_{i,1}, \ldots, a_{i,n}$
    Reguläre Zeit für Level i mit Item j: $a_{i,j}$.



Problemstellung

Eingabe:

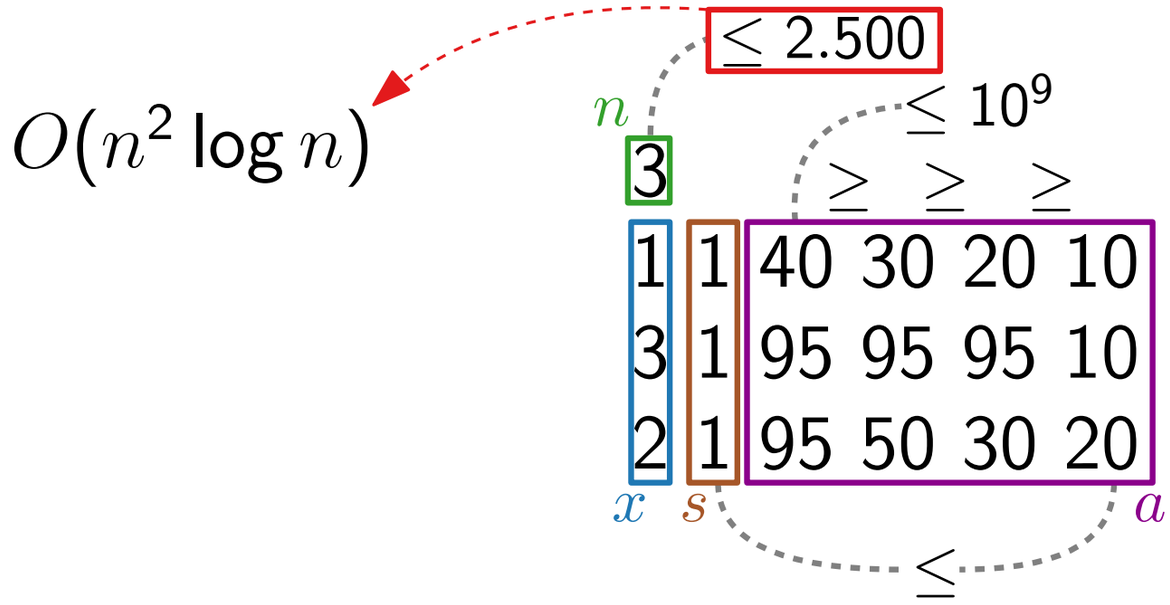
- Erste Zeile: Anzahl Level n
- n weitere Zeilen:
  - Abkürzungsitem x_i
  - Abkürzungszeit s_i
  - $n+1$ weitere Zahlen: $a_{i,0}, a_{i,1}, \ldots, a_{i,n}$
    Reguläre Zeit für Level i mit Item j: $a_{i,j}$.



Problemstellung

Eingabe:

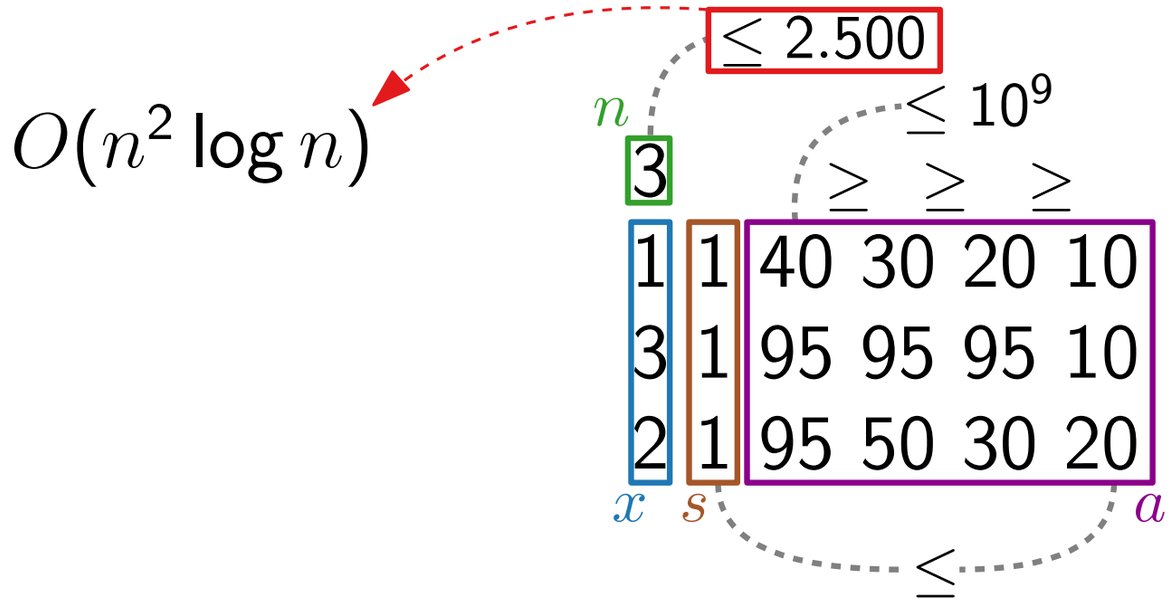
- Erste Zeile: Anzahl Level n
- n weitere Zeilen:
  - Abkürzungsitem x_i
  - Abkürzungszeit s_i
  - $n+1$ weitere Zahlen: $a_{i,0}, a_{i,1}, \ldots, a_{i,n}$
    Reguläre Zeit für Level i mit Item j: $a_{i,j}$.



Problemstellung

Eingabe:

- Erste Zeile: Anzahl Level n
- n weitere Zeilen:
  - Abkürzungsitem x_i
  - Abkürzungszeit s_i
  - $n+1$ weitere Zahlen: $a_{i,0}, a_{i,1}, \ldots, a_{i,n}$
    Reguläre Zeit für Level i mit Item j: $a_{i,j}$.

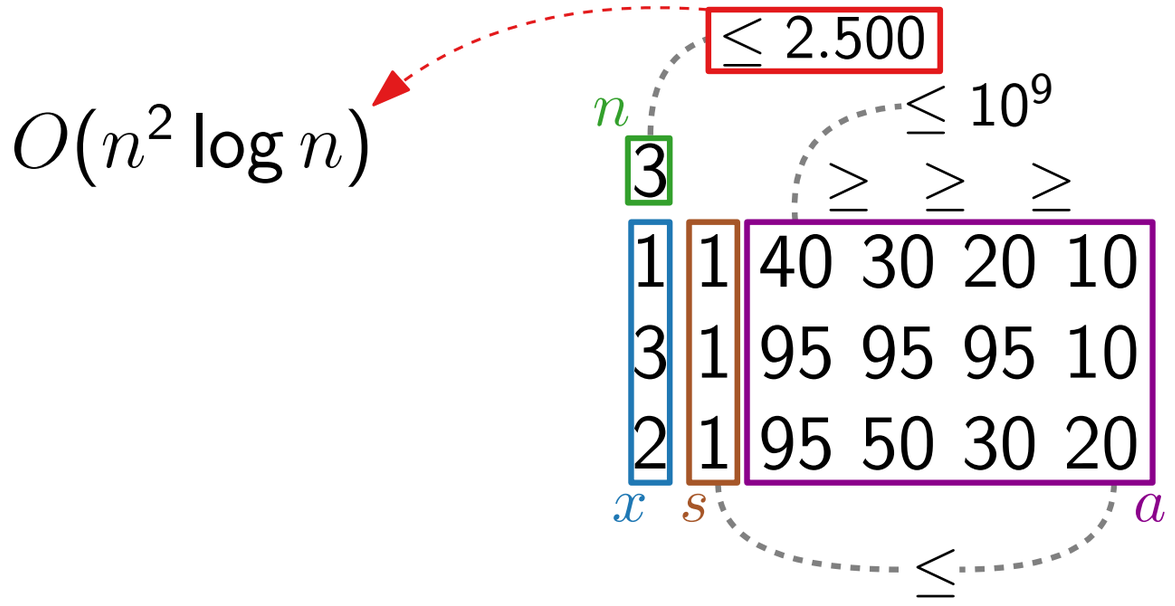


Ausgabe:

Problemstellung

Eingabe:

- Erste Zeile: Anzahl Level n
- n weitere Zeilen:
  - Abkürzungsitem x_i
  - Abkürzungszeit s_i
  - $n+1$ weitere Zahlen: $a_{i,0}, a_{i,1}, \ldots, a_{i,n}$
    Reguläre Zeit für Level i mit Item j: $a_{i,j}$.



Ausgabe:

Minimale Zeit, in der man alle Level in beliebiger Reihenfolge absolviert.

Gesucht

Gesucht

Levelreihenfolge:

Gesucht

Levelreihenfolge:

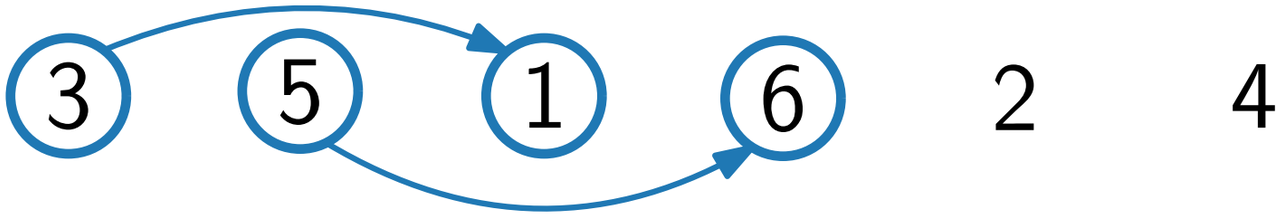
3 5 1 6 2 4

Gesucht

Levelreihenfolge:

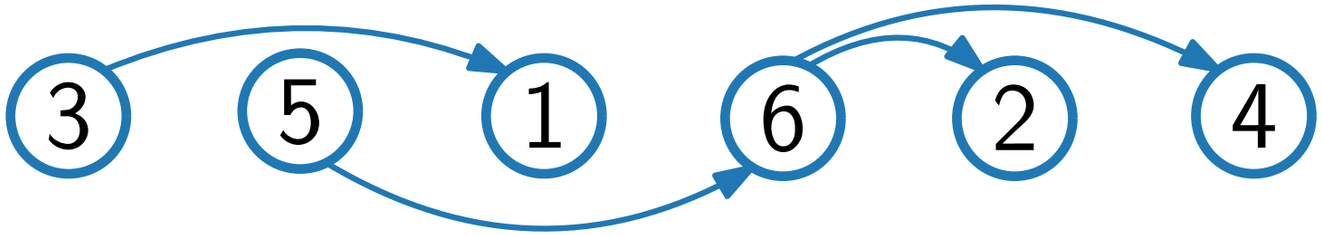
Gesucht

Levelreihenfolge:



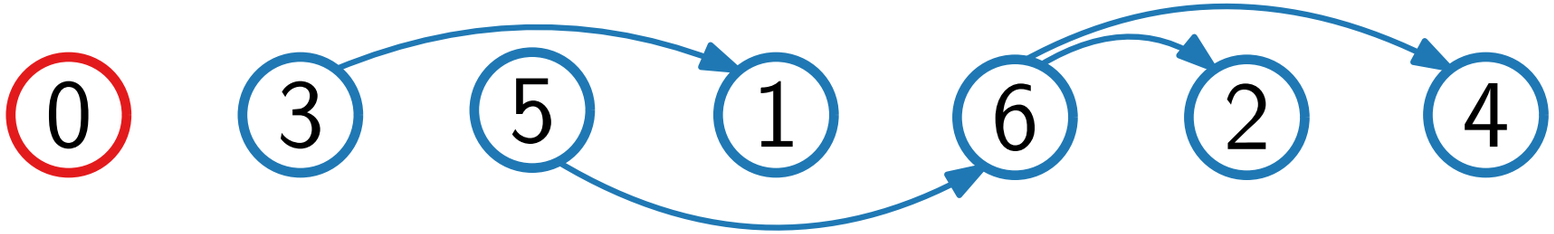
Gesucht

Levelreihenfolge:



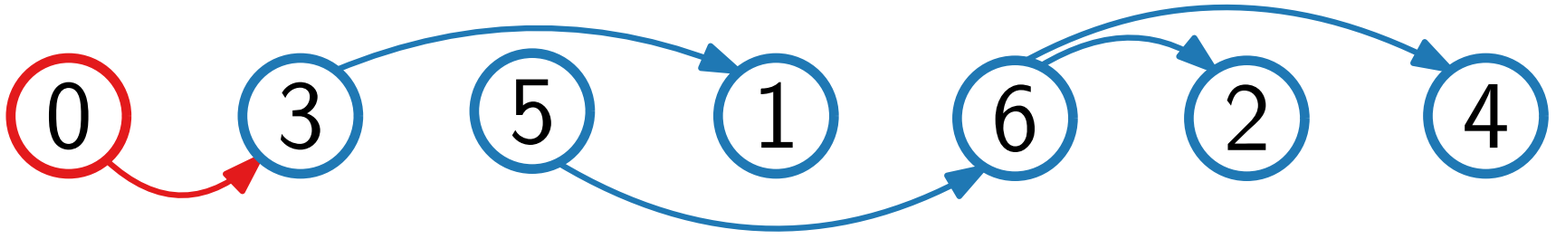
Gesucht

Levelreihenfolge:



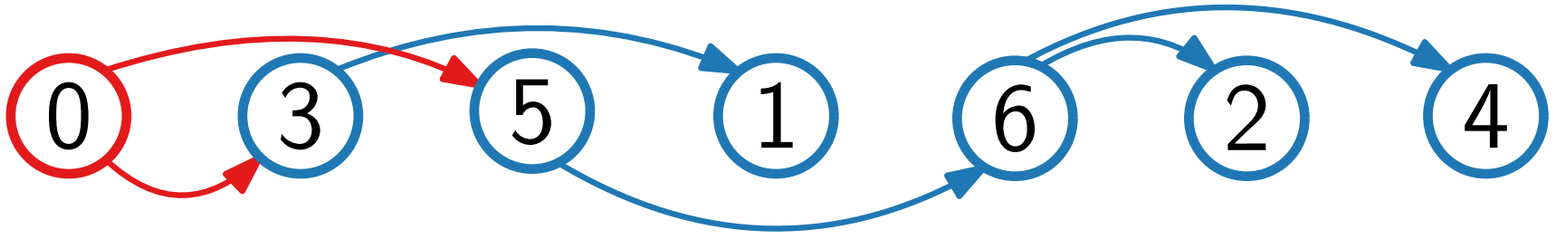
Gesucht

Levelreihenfolge:



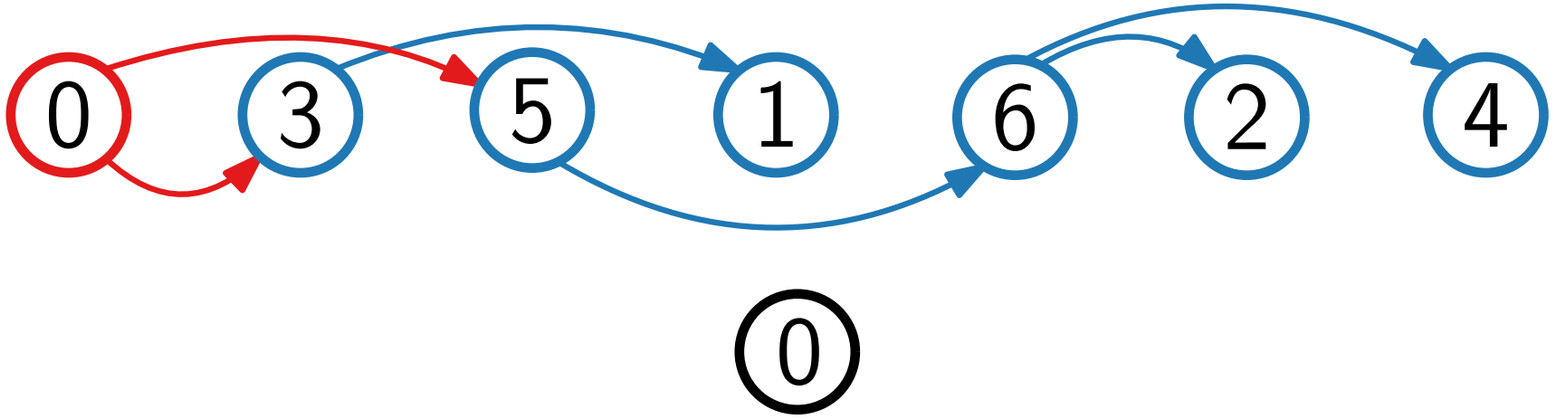
Gesucht

Levelreihenfolge:



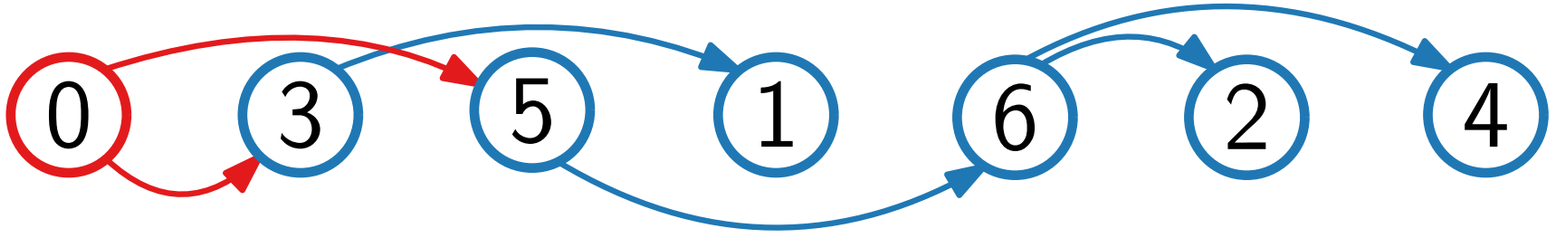
Gesucht

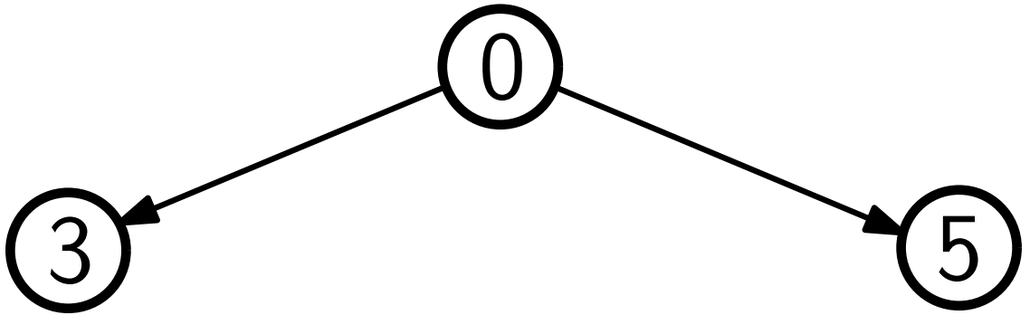
Levelreihenfolge:



Gesucht

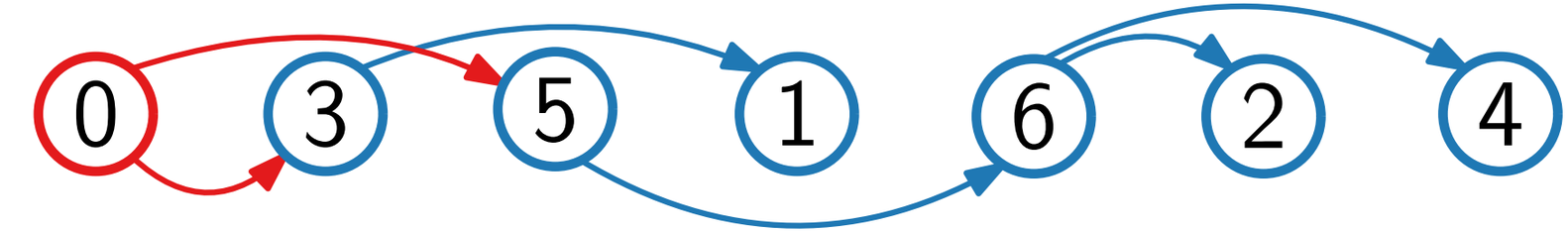
Levelreihenfolge:

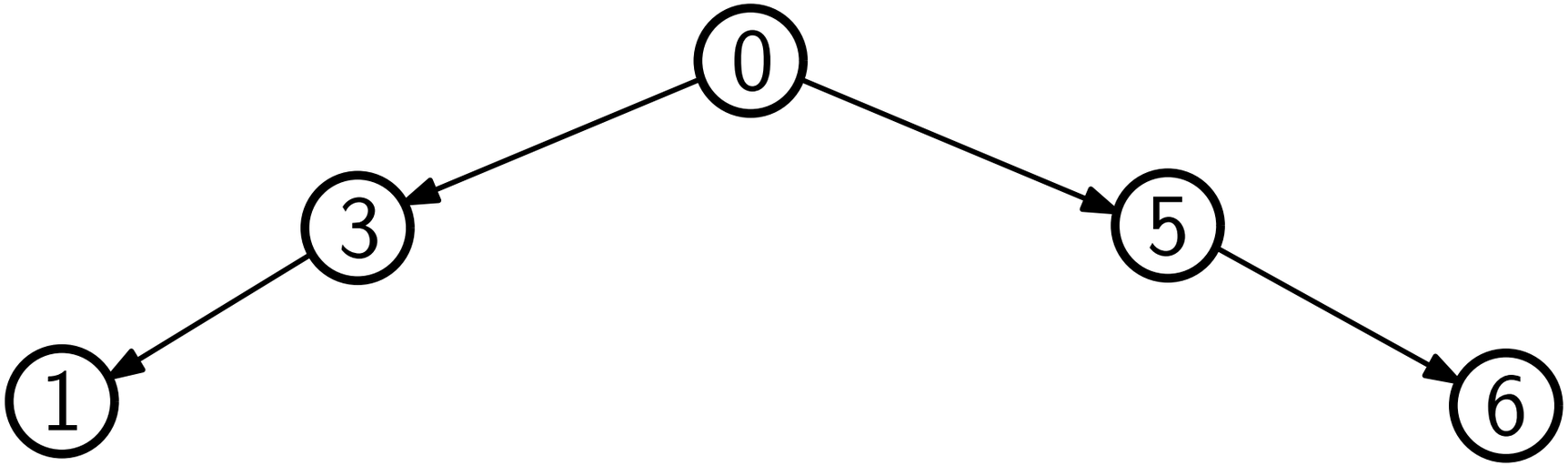




Gesucht

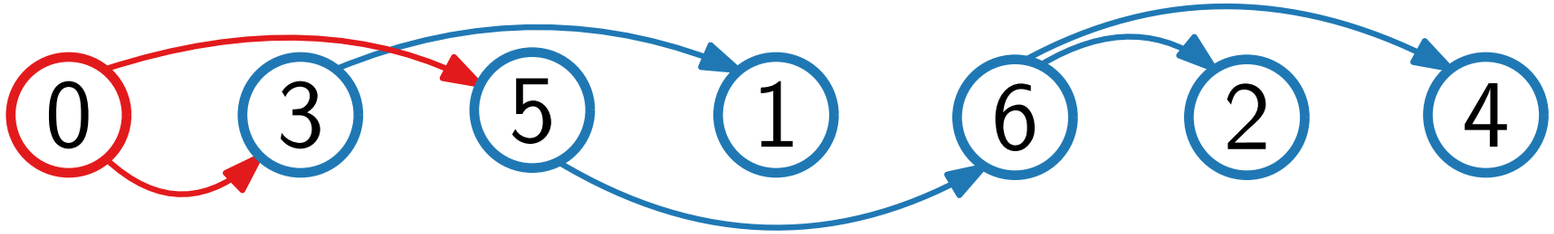
Levelreihenfolge:

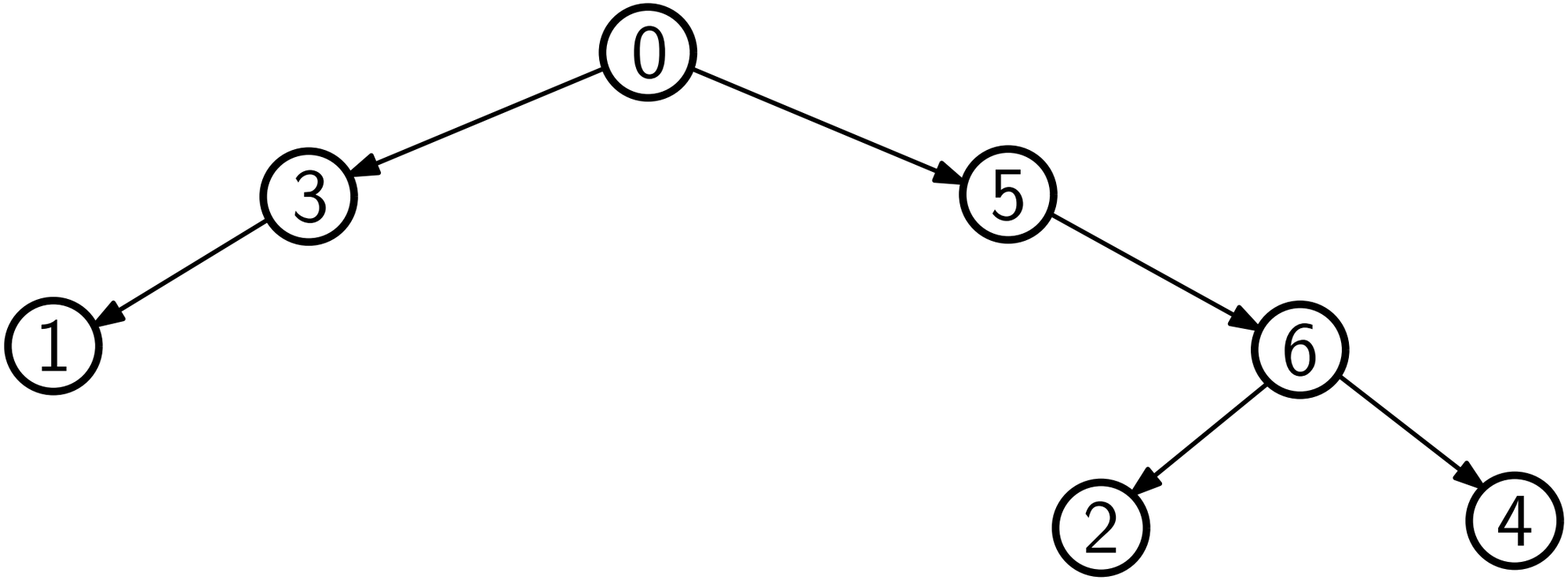




Gesucht

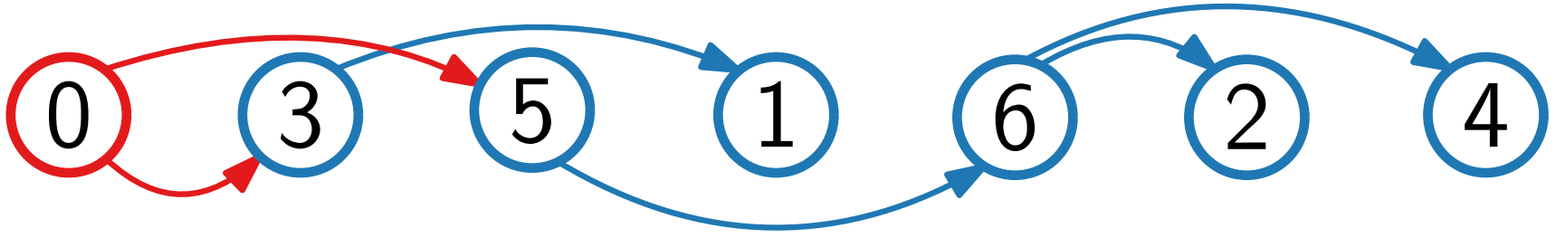
Levelreihenfolge:

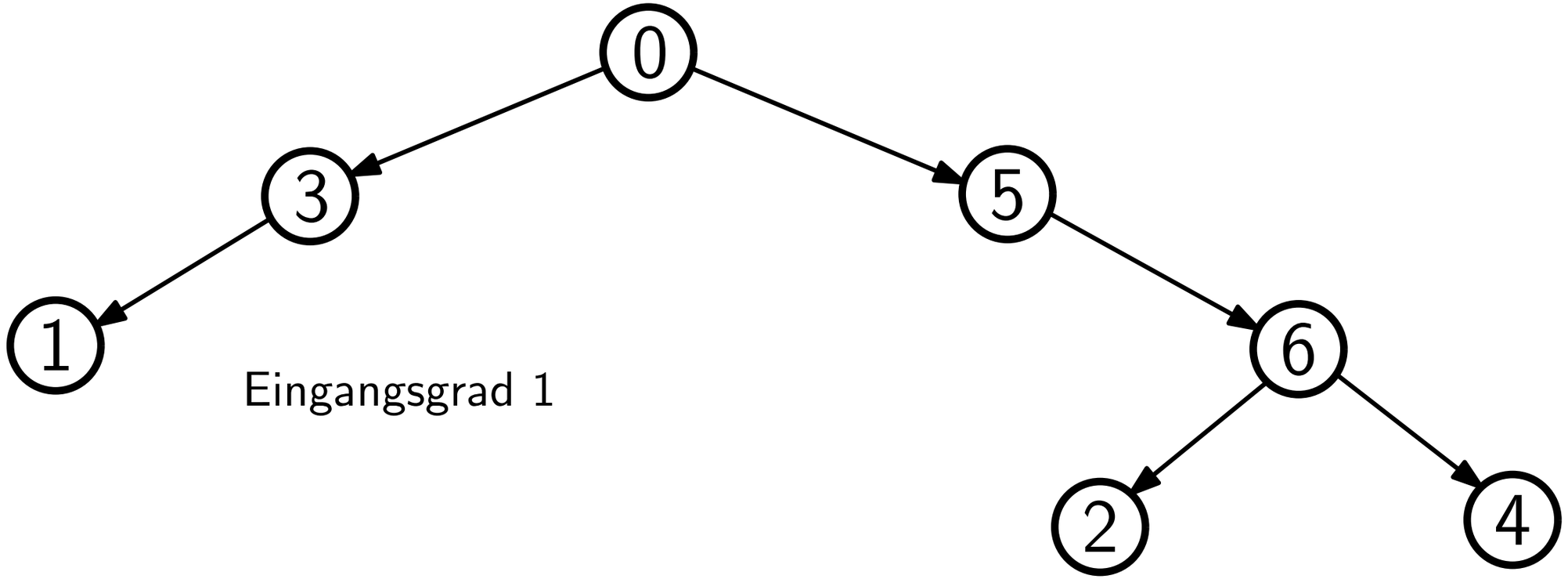




Gesucht

Levelreihenfolge:

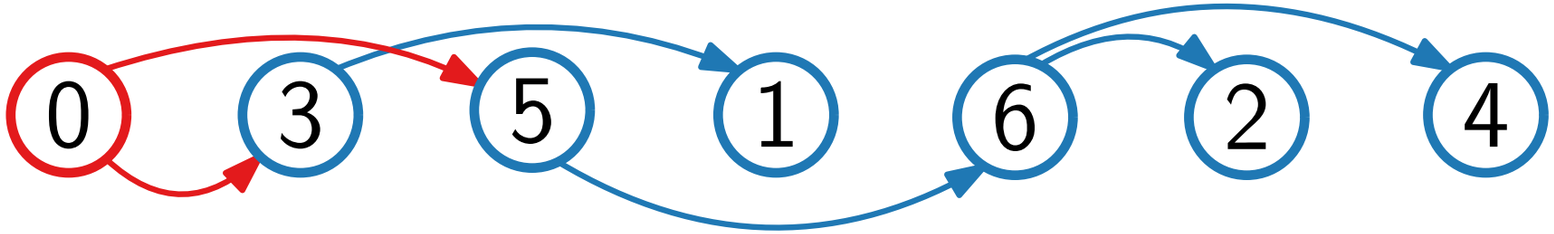


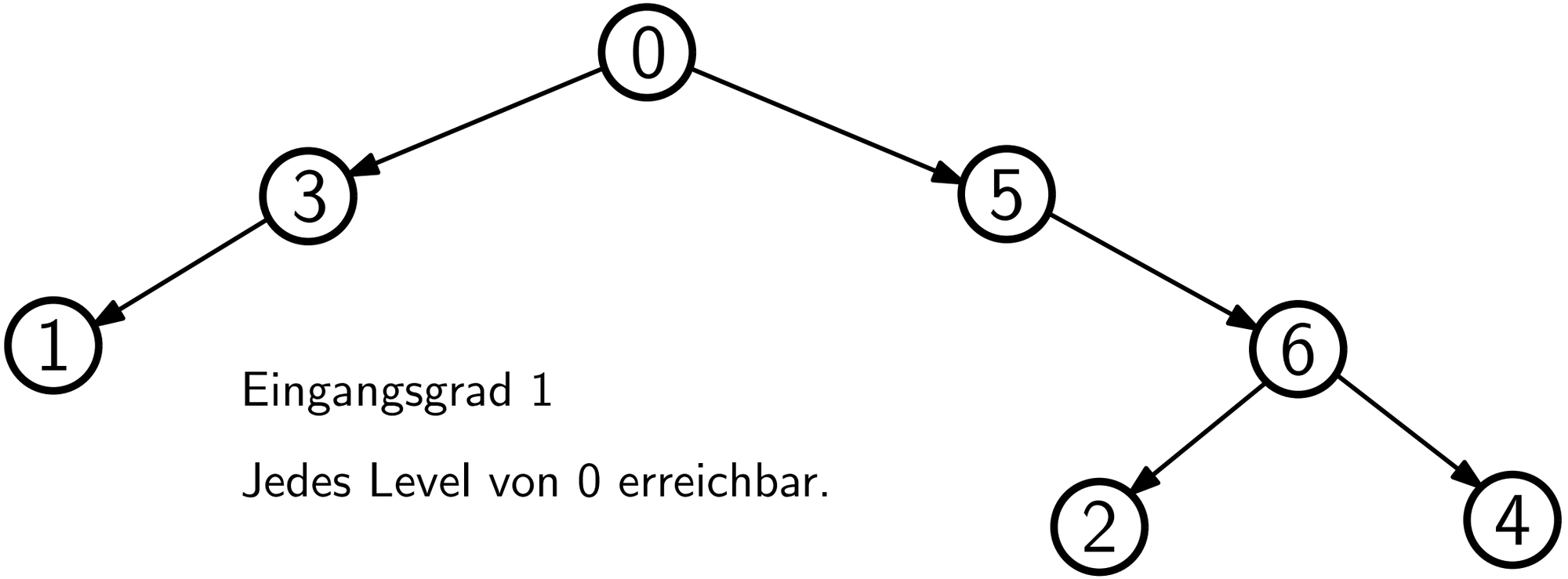


Eingangsgrad 1

Gesucht

Levelreihenfolge:



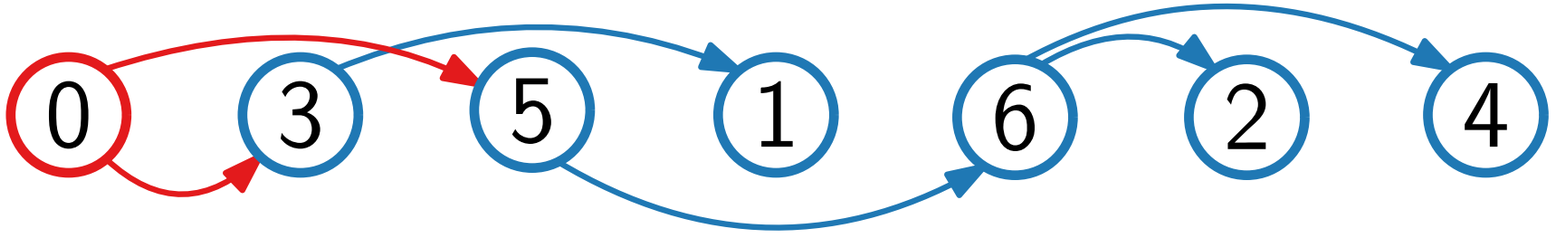


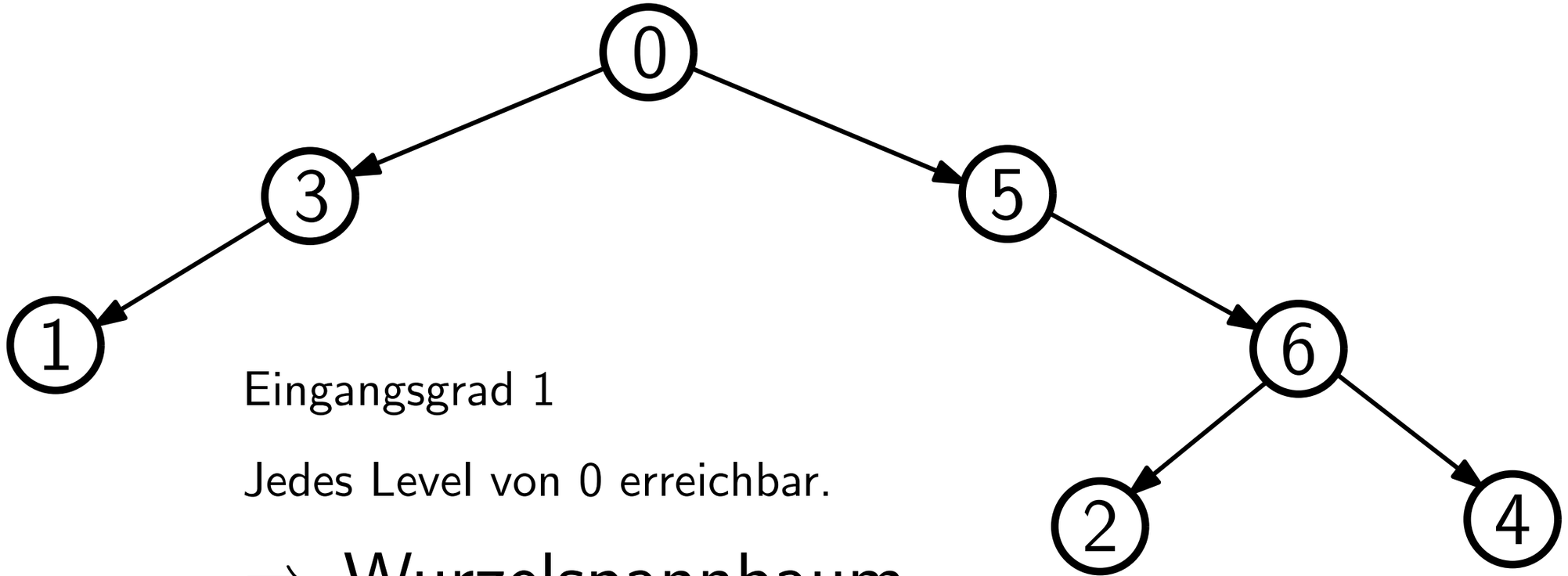
Eingangsgrad 1

Jedes Level von 0 erreichbar.

Gesucht

Levelreihenfolge:





Eingangsgrad 1

Jedes Level von 0 erreichbar.

$\Rightarrow$ Wurzelspannbaum

Minimale Wurzelspannbäume

Minimale Wurzelspannbäume

Satz. Edmonds' Algorithmus berechnet einen minimalen s-Wurzelspannbaum.

Minimale Wurzelspannbäume

Satz. Angewandt auf einen gewichteten (Multi-) Graphen $G = (V, E)$, läuft Edmonds' Algorithmus in $O(VE)$ Zeit.

Minimale Wurzelspannbäume

Satz. Angewandt auf einen gewichteten (Multi-) Graphen $G = (V, E)$, läuft Edmonds' Algorithmus in $O(V|E|)$ Zeit.

Vollständiger Graph: $|E| \in \Theta(|V|^2) \Rightarrow O(|V|^3)$

Minimale Wurzelspannbäume

Satz. Angewandt auf einen gewichteten (Multi-) Graphen $G = (V, E)$, läuft Edmonds' Algorithmus in $O(V|E|)$ Zeit.

Vollständiger Graph: $|E| \in \Theta(|V|^2) \Rightarrow O(|V|^3)$

Tarjan (1977):

Minimale Wurzelspannbäume

Satz. Angewandt auf einen gewichteten (Multi-) Graphen $G = (V, E)$, läuft Edmonds' Algorithmus in $O(VE)$ Zeit.

Vollständiger Graph: $|E| \in \Theta(|V|^2) \Rightarrow O(|V|^3)$

Tarjan (1977):

$O(|V|^2)$ für dichte Graphen.

Minimale Wurzelspannbäume

Satz. Angewandt auf einen gewichteten (Multi-) Graphen $G = (V, E)$, läuft Edmonds' Algorithmus in $O(VE)$ Zeit.

Vollständiger Graph: $|E| \in \Theta(|V|^2) \Rightarrow O(|V|^3)$

Tarjan (1977):

$O(|V|^2)$ für dichte Graphen.

Sehr kompliziert

Minimale Wurzelspannbäume

Satz. Angewandt auf einen gewichteten (Multi-) Graphen $G = (V, E)$, läuft Edmonds' Algorithmus in $O(VE)$ Zeit.

Vollständiger Graph: $|E| \in \Theta(|V|^2) \Rightarrow O(|V|^3)$

Tarjan (1977):

$O(|V|^2)$ für dichte Graphen.

Sehr kompliziert ☹

Minimale Wurzelspannbäume

Satz. Angewandt auf einen gewichteten (Multi-) Graphen $G = (V, E)$, läuft Edmonds' Algorithmus in $O(VE)$ Zeit.

Vollständiger Graph: $|E| \in \Theta(|V|^2) \Rightarrow O(|V|^3)$

Tarjan (1977):

$O(|V|^2)$ für dichte Graphen.

Sehr kompliziert ☹

Alternativ:

Minimale Wurzelspannbäume

Satz. Angewandt auf einen gewichteten (Multi-) Graphen $G = (V, E)$, läuft Edmonds' Algorithmus in $O(VE)$ Zeit.

Vollständiger Graph: $|E| \in \Theta(|V|^2) \Rightarrow O(|V|^3)$

Tarjan (1977):

$O(|V|^2)$ für dichte Graphen.

Sehr kompliziert ☹

Alternativ:

Gegebener Graph ist nicht beliebig
$(j > j' \Rightarrow a_{i,j} \leq a_{i,j'})$

Minimale Wurzelspannbäume

Satz. Angewandt auf einen gewichteten (Multi-) Graphen $G = (V, E)$, läuft Edmonds' Algorithmus in $O(V|E|)$ Zeit.

Vollständiger Graph: $|E| \in \Theta(|V|^2) \Rightarrow O(|V|^3)$

Tarjan (1977):

$O(|V|^2)$ für dichte Graphen.

Sehr kompliziert ☹

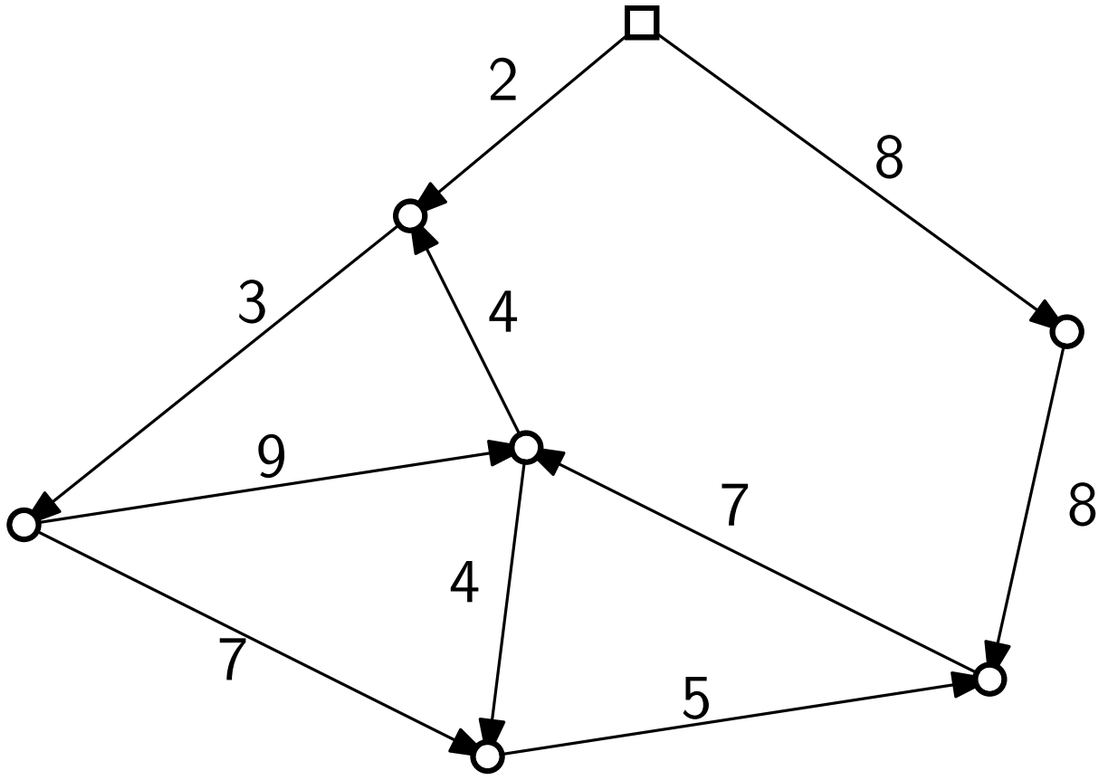
Alternativ:

Gegebener Graph ist nicht beliebig $(j > j' \Rightarrow a_{i,j} \leq a_{i,j'})$

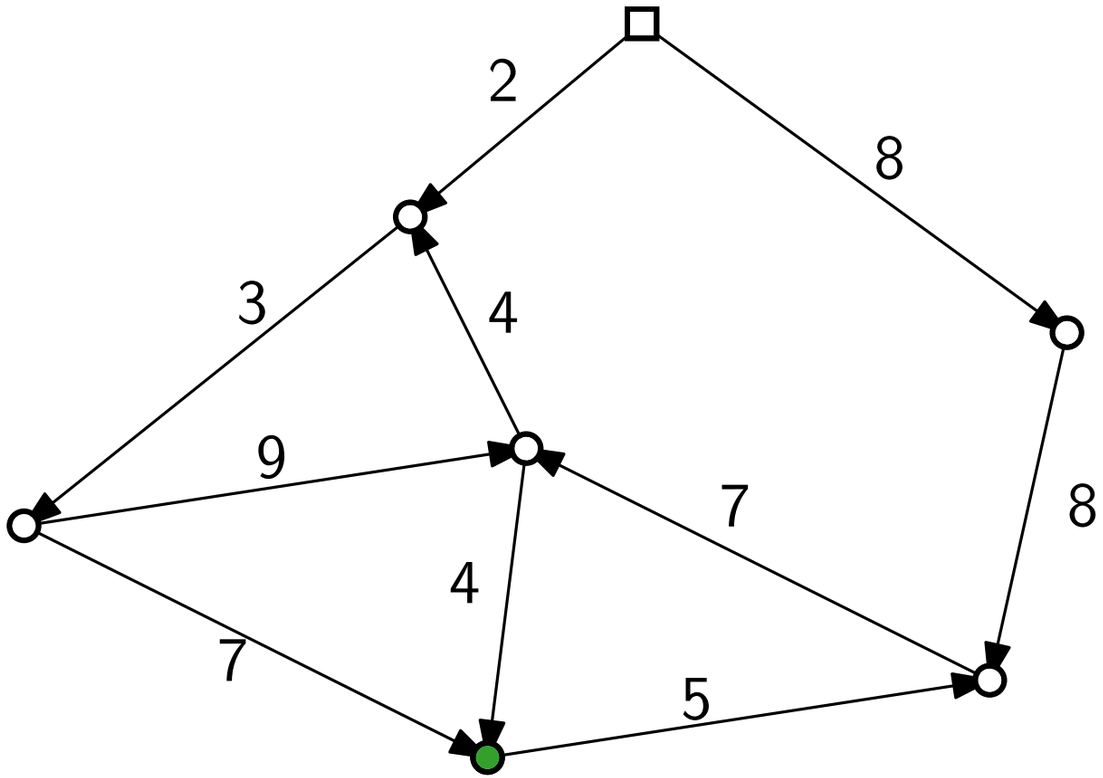
Nutze diese Eigenschaft aus

Edmonds Algorithmus – Illustration

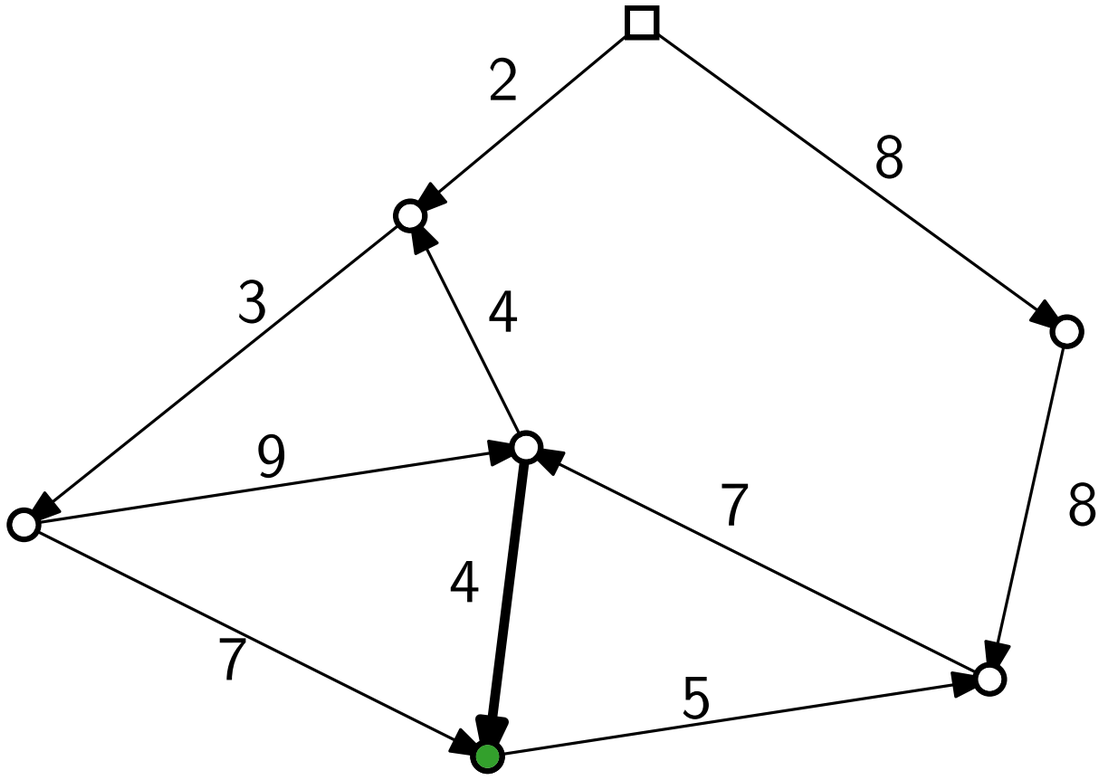
Edmonds Algorithmus – Illustration



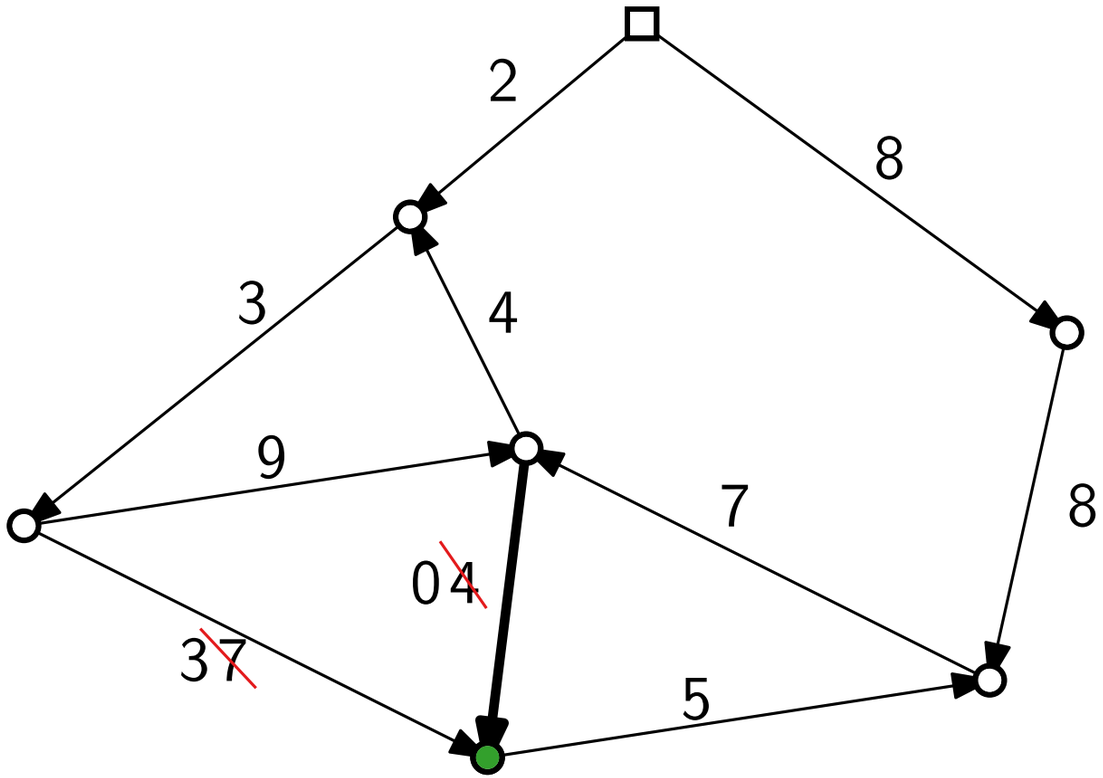
Edmonds Algorithmus – Illustration



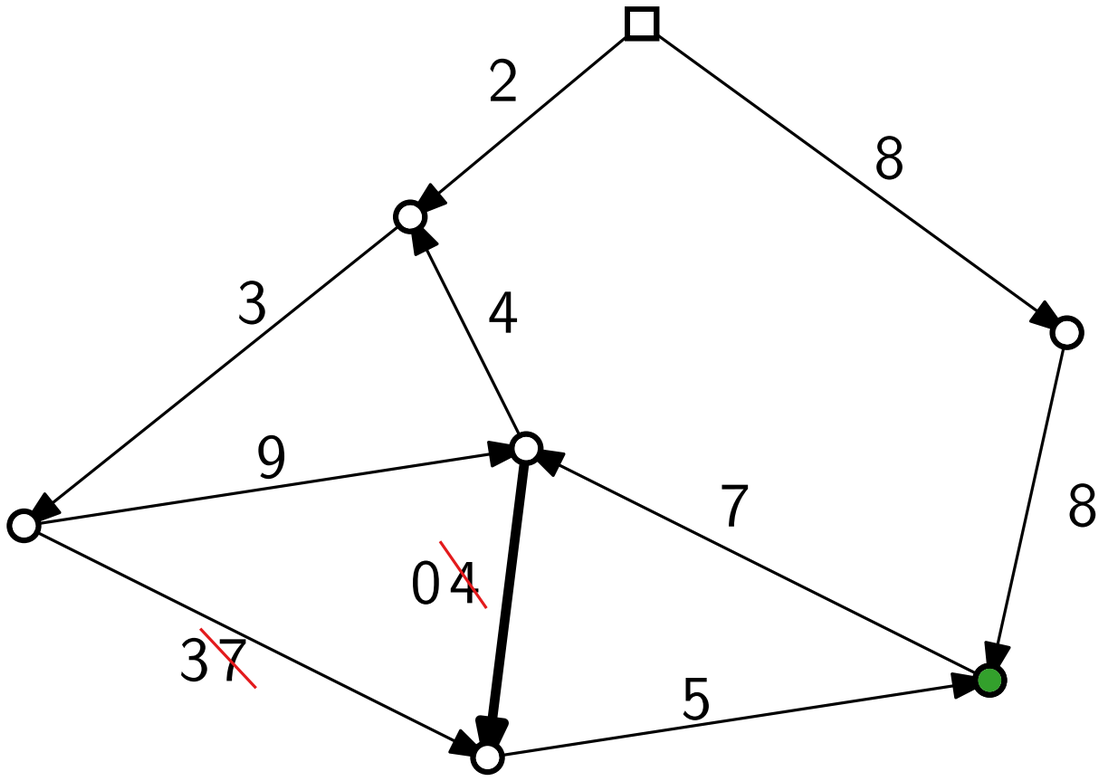
Edmonds Algorithmus – Illustration



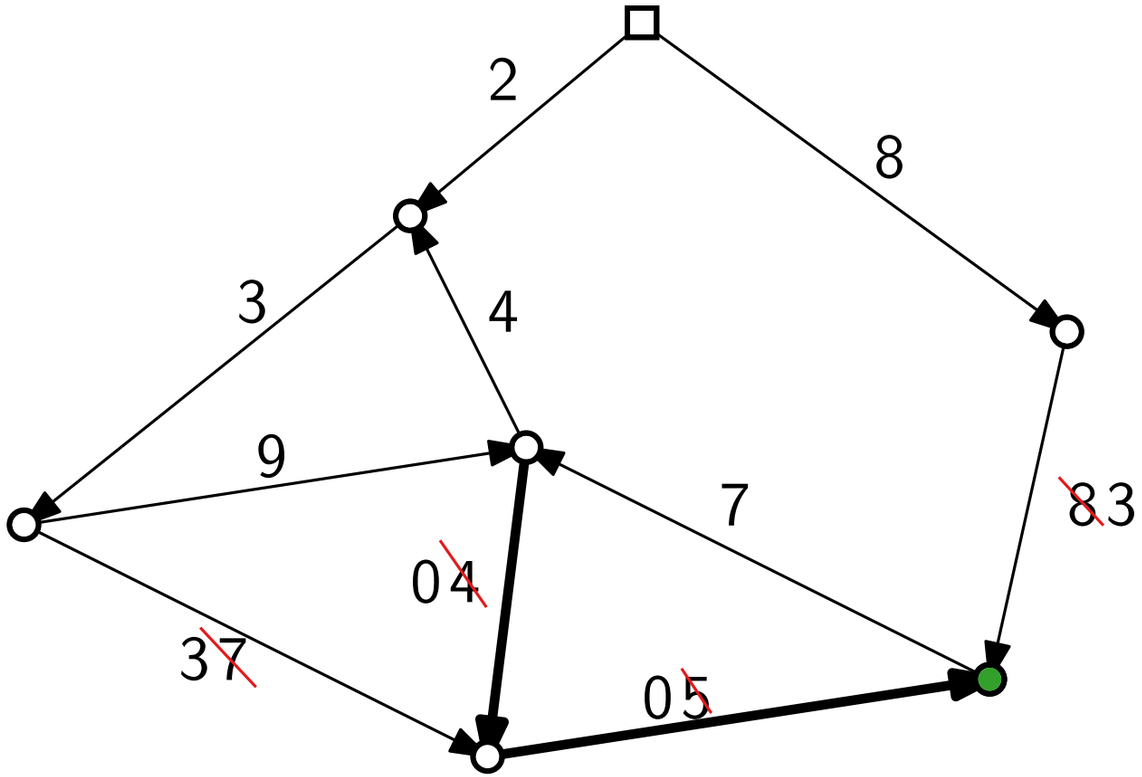
Edmonds Algorithmus – Illustration



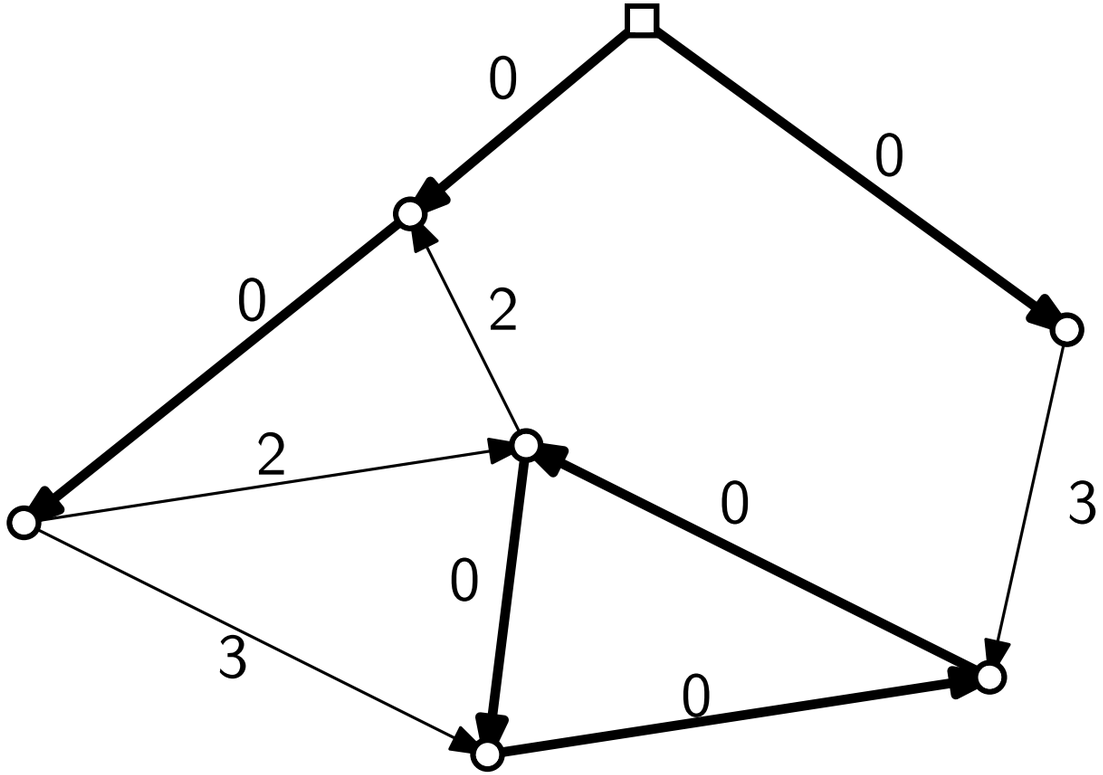
Edmonds Algorithmus – Illustration



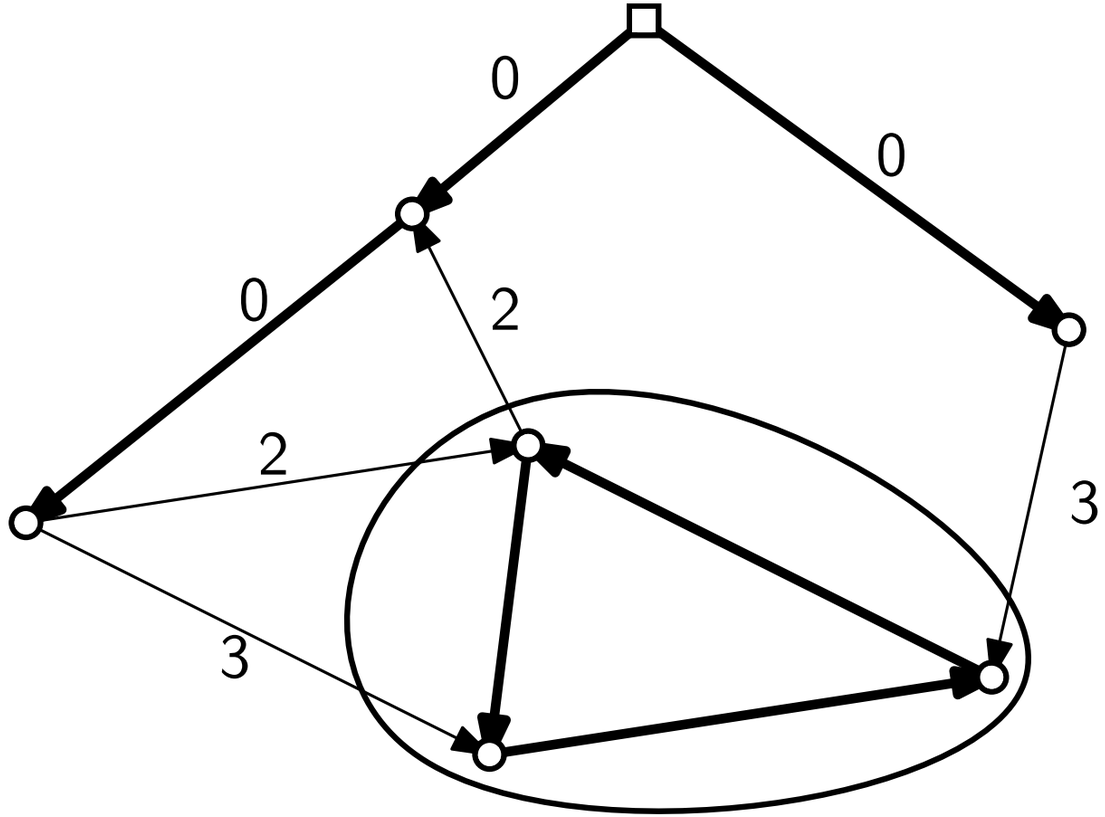
Edmonds Algorithmus – Illustration



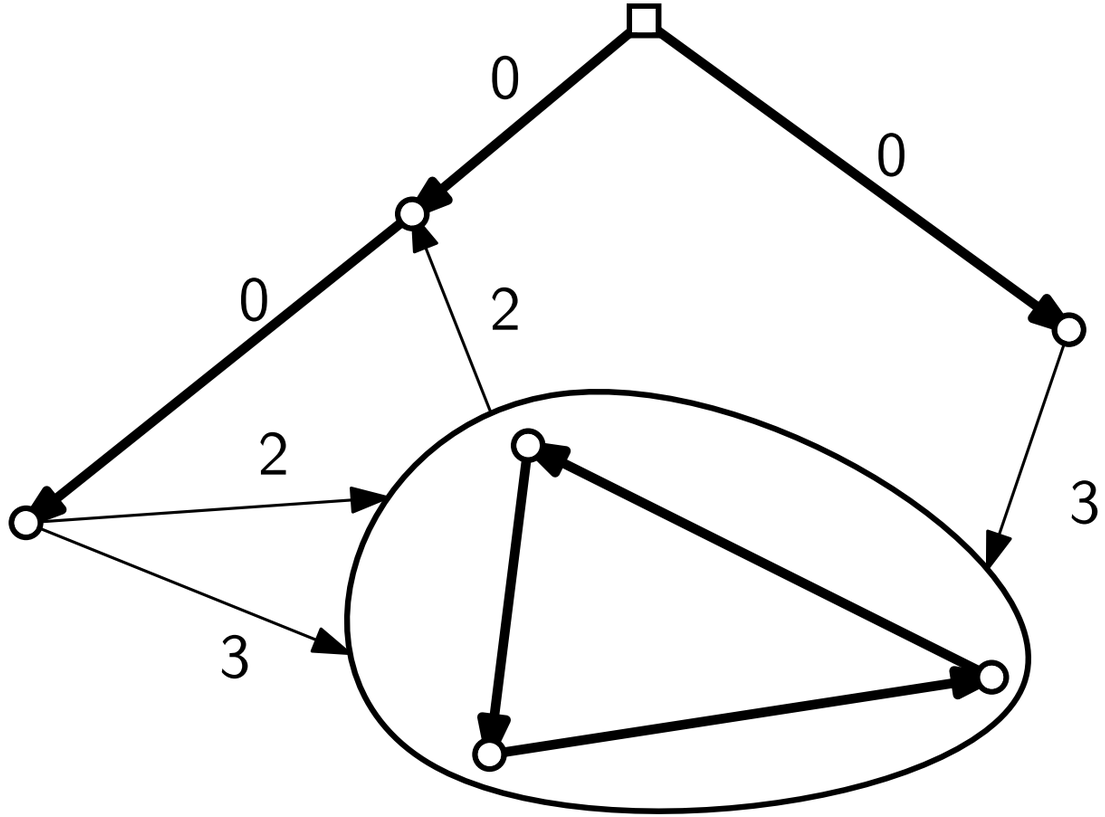
Edmonds Algorithmus – Illustration



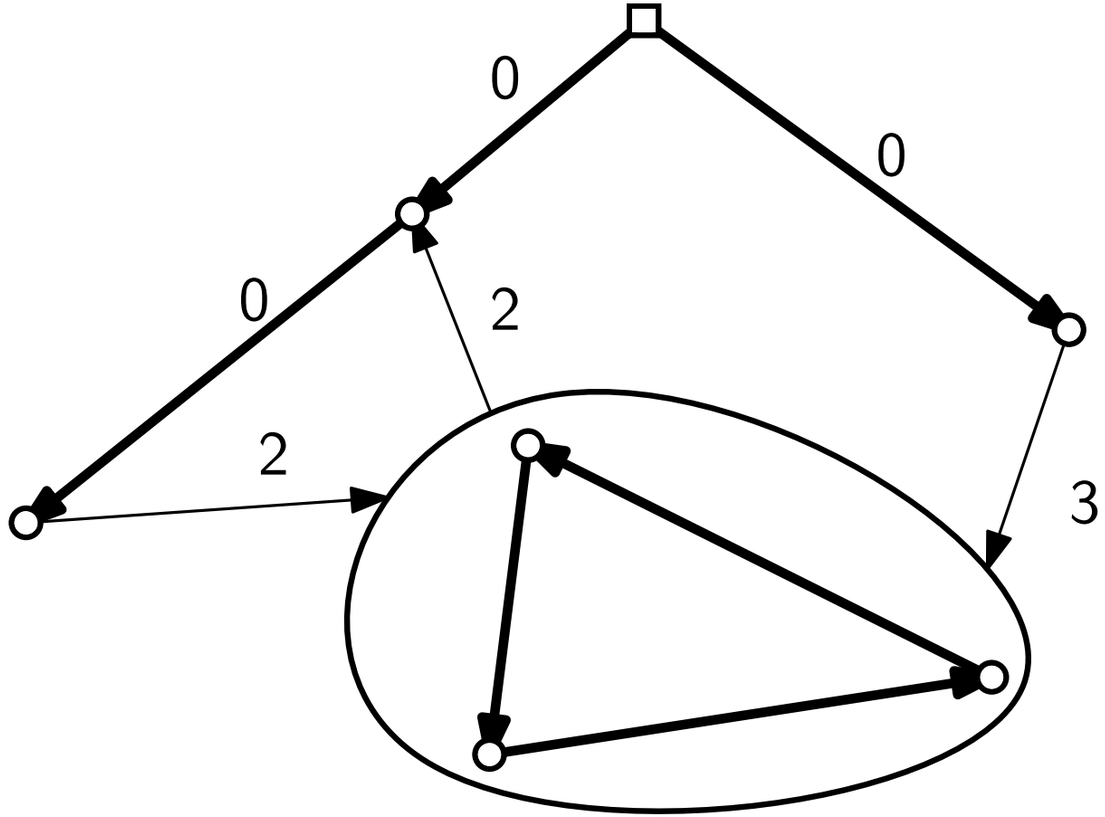
Edmonds Algorithmus – Illustration



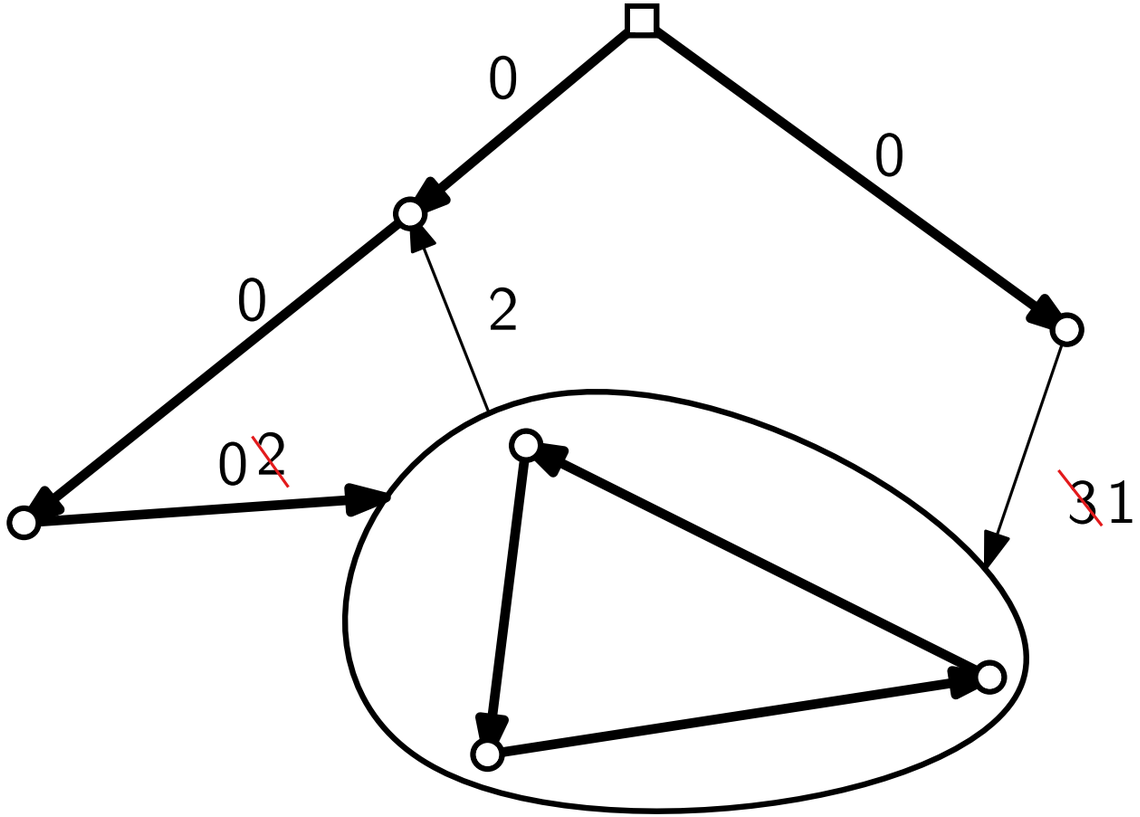
Edmonds Algorithmus – Illustration



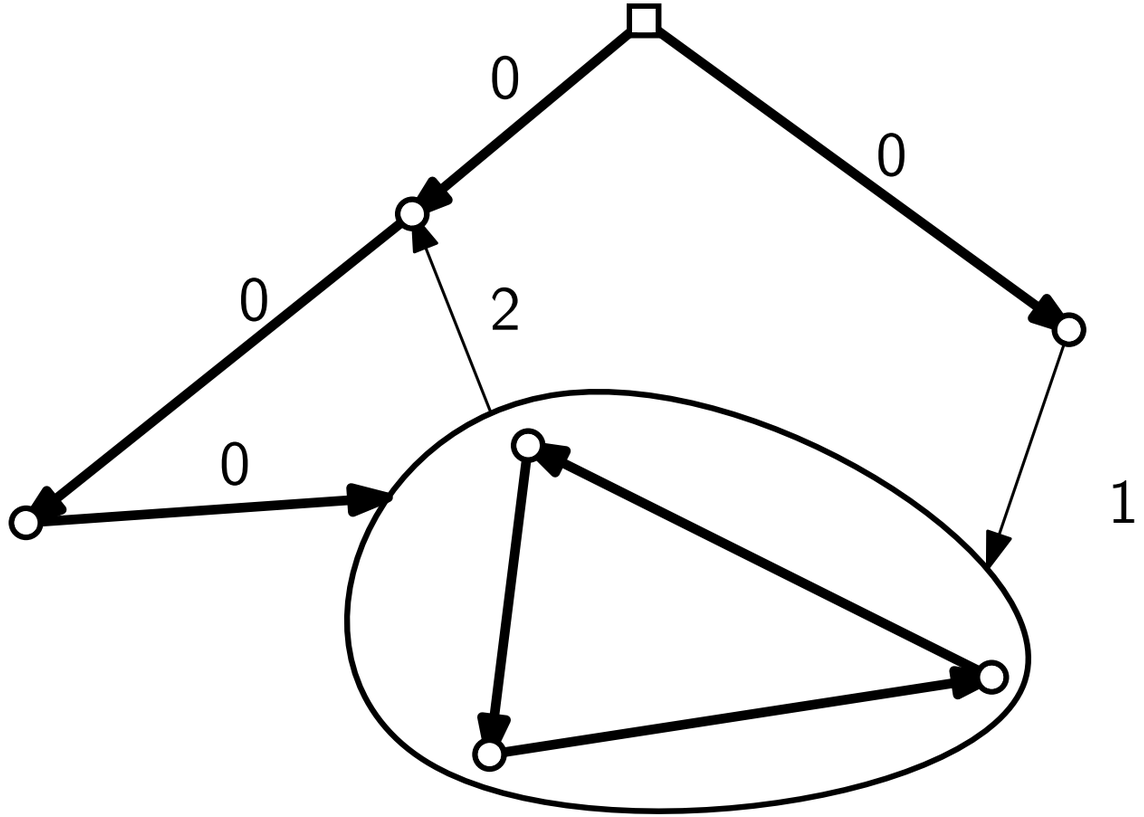
Edmonds Algorithmus – Illustration



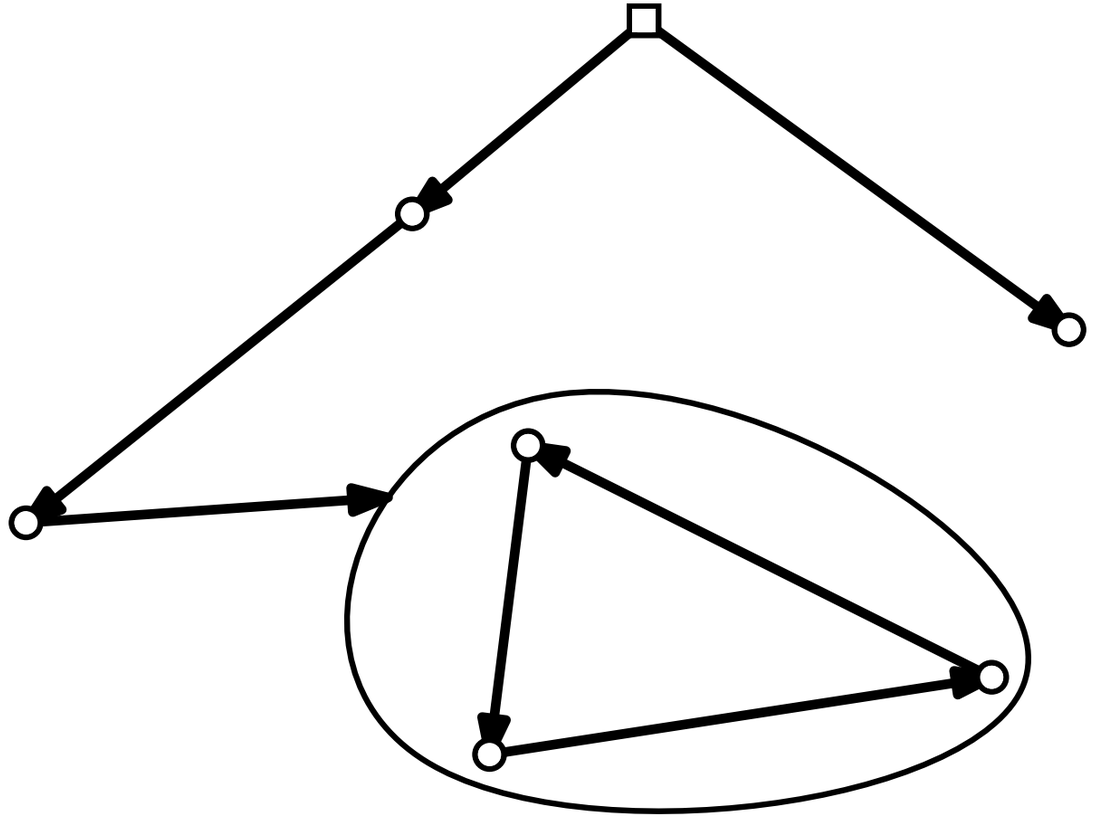
Edmonds Algorithmus – Illustration



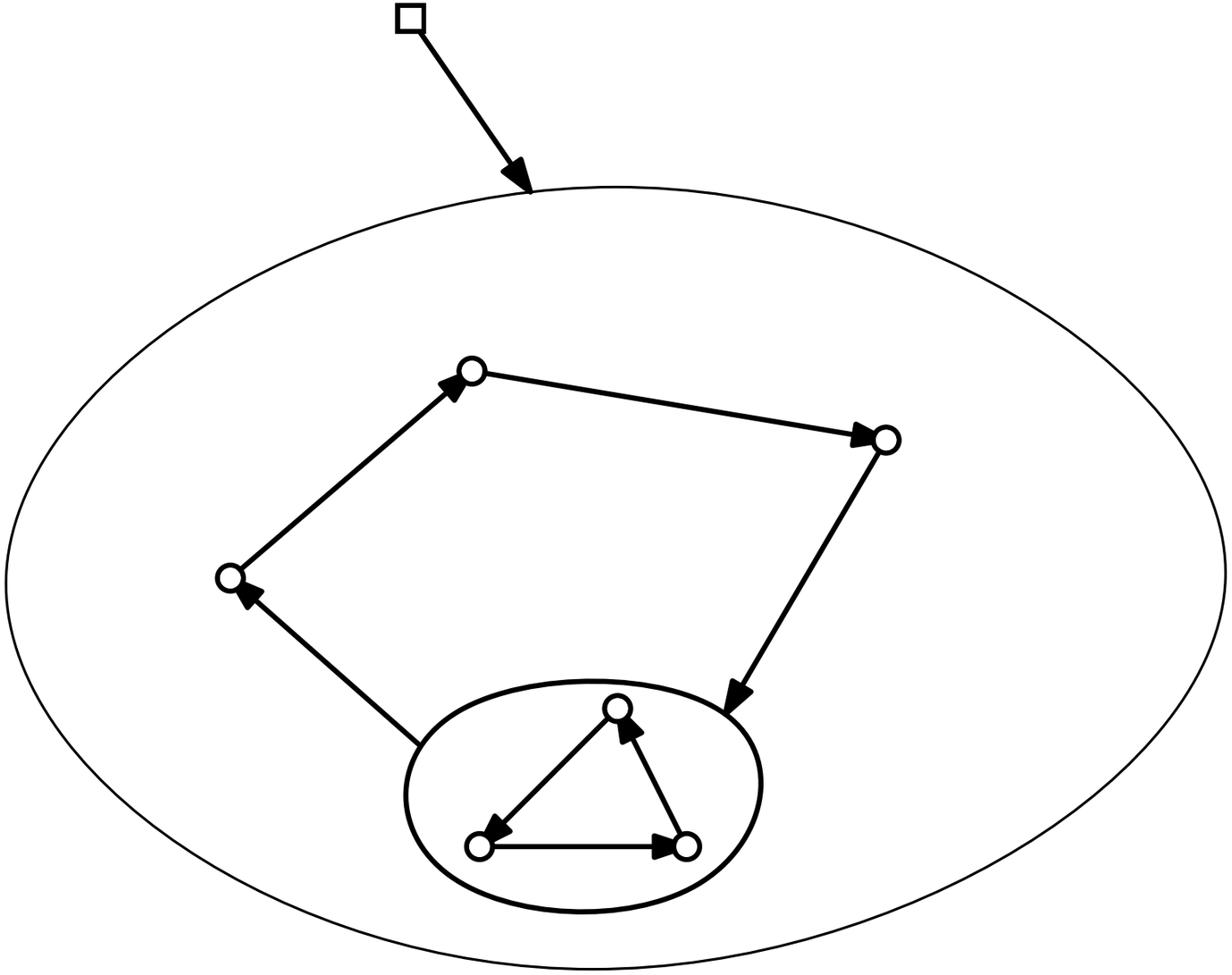
Edmonds Algorithmus – Illustration



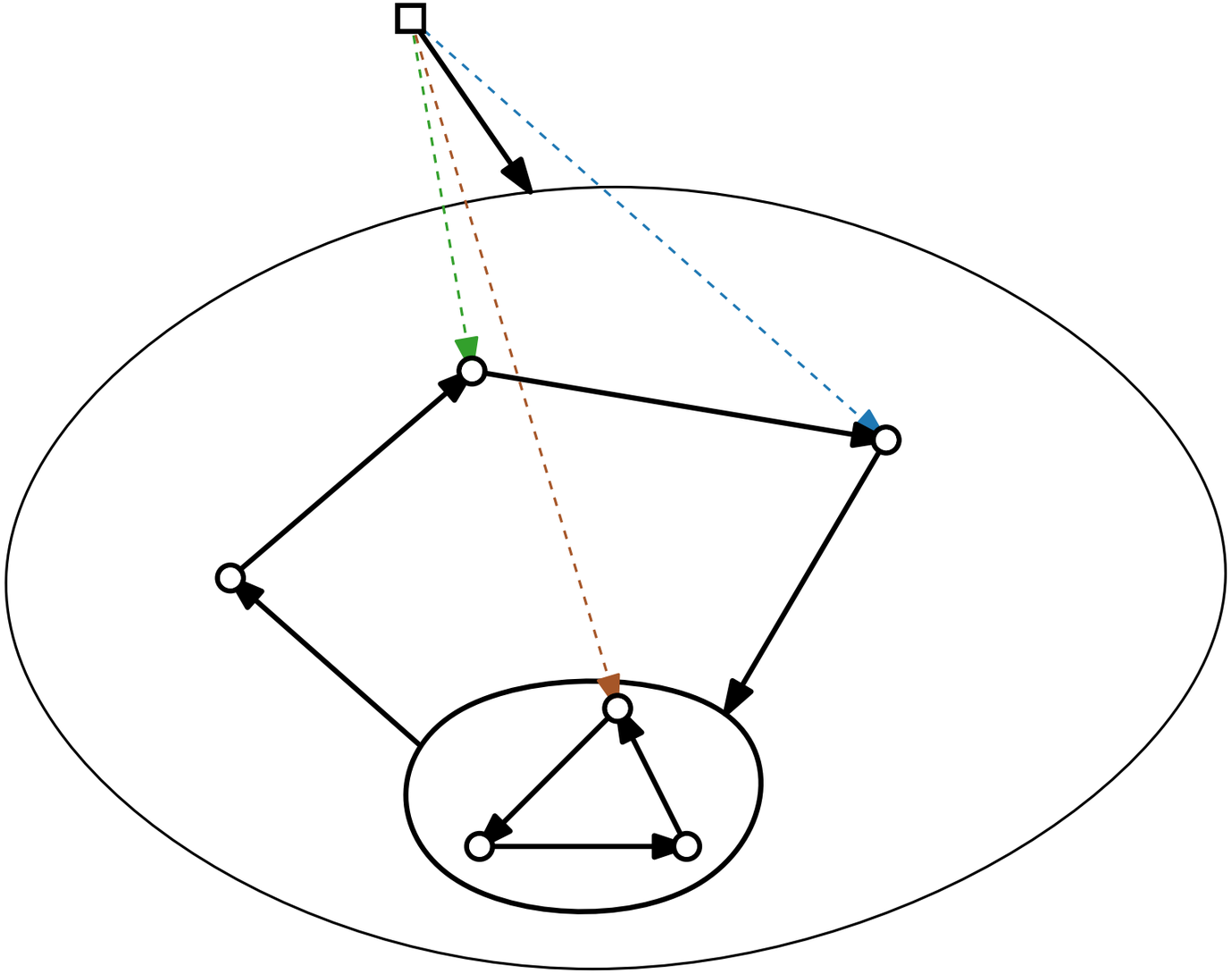
Edmonds Algorithmus – Illustration



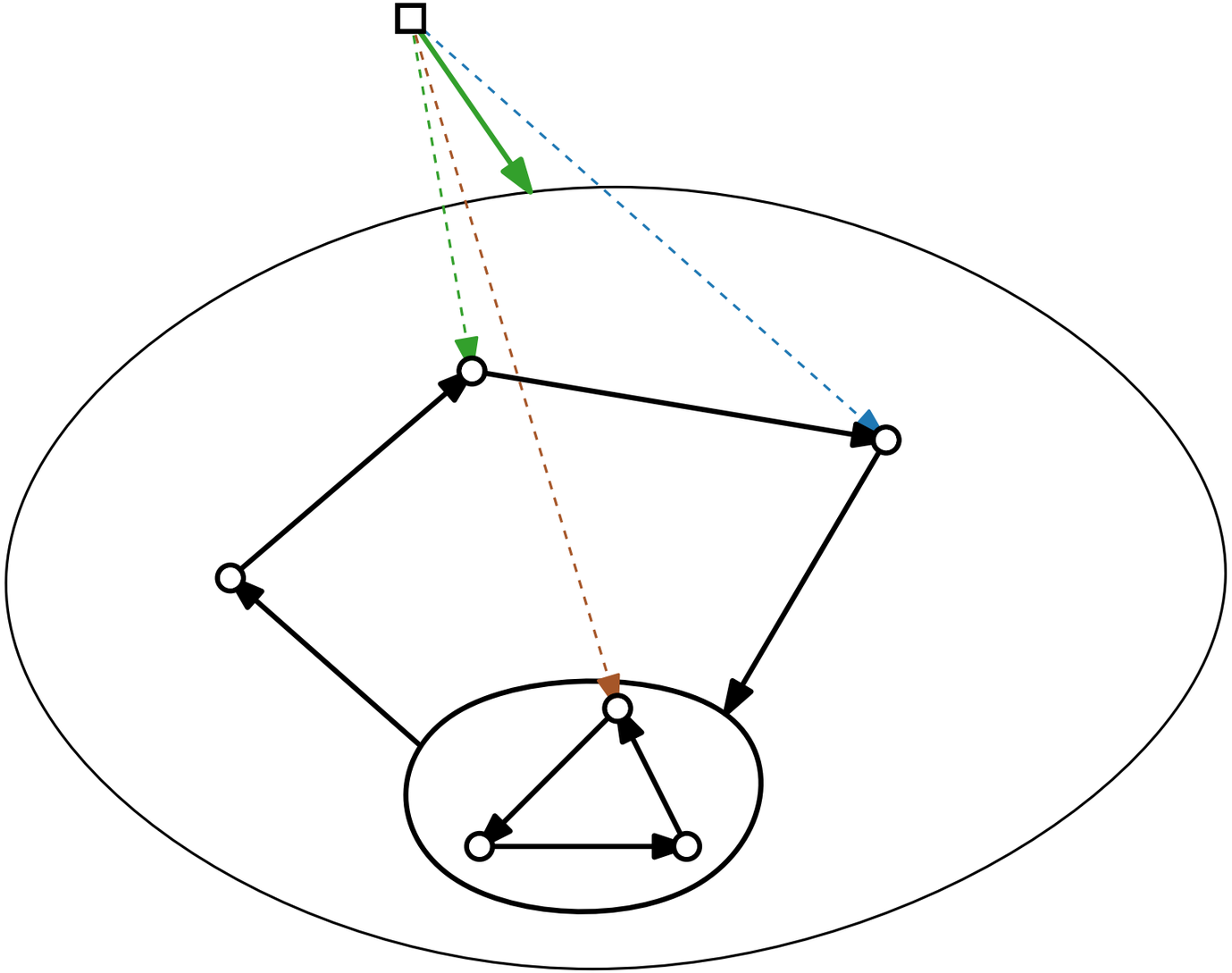
Edmonds Algorithmus – Illustration



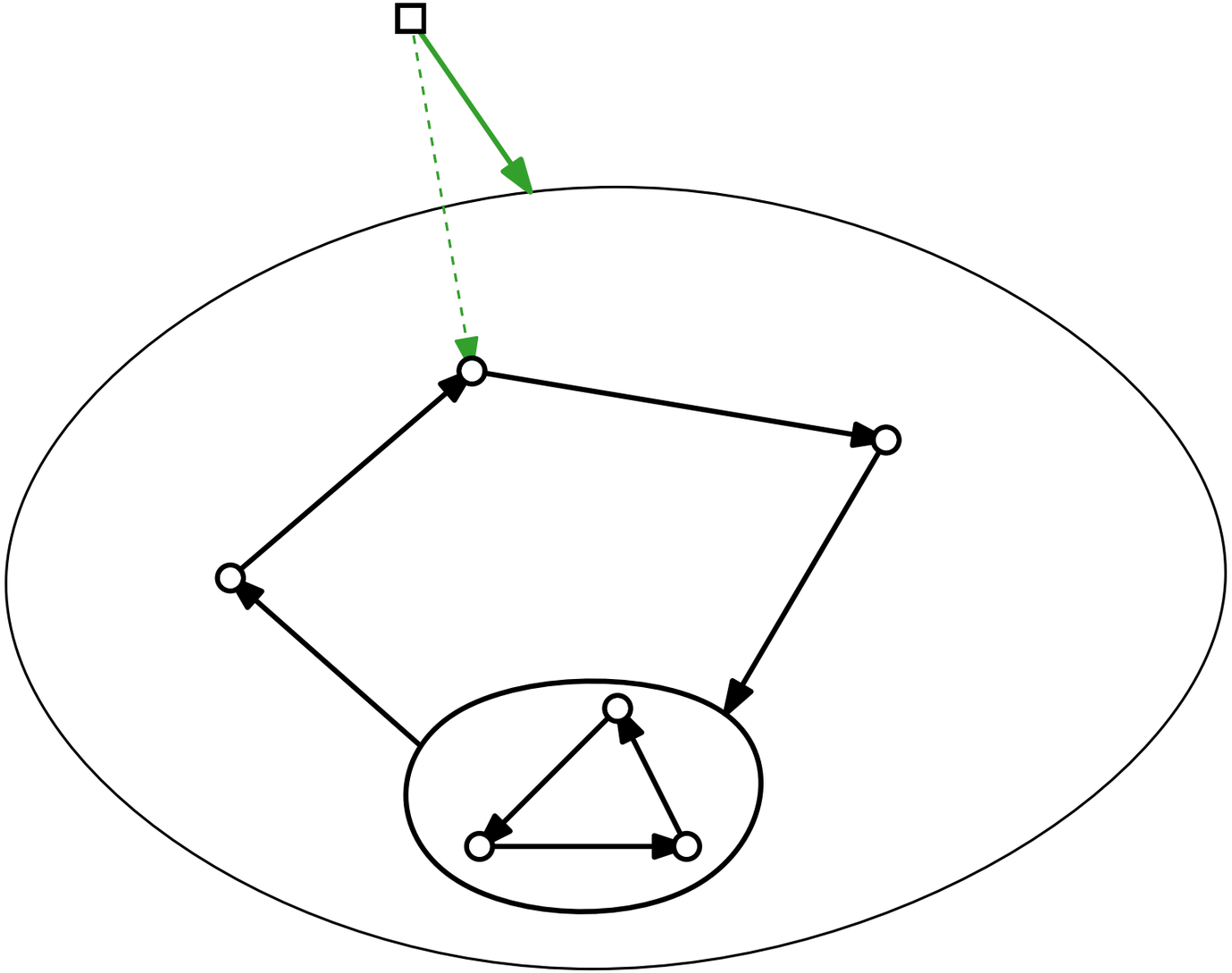
Edmonds Algorithmus – Illustration



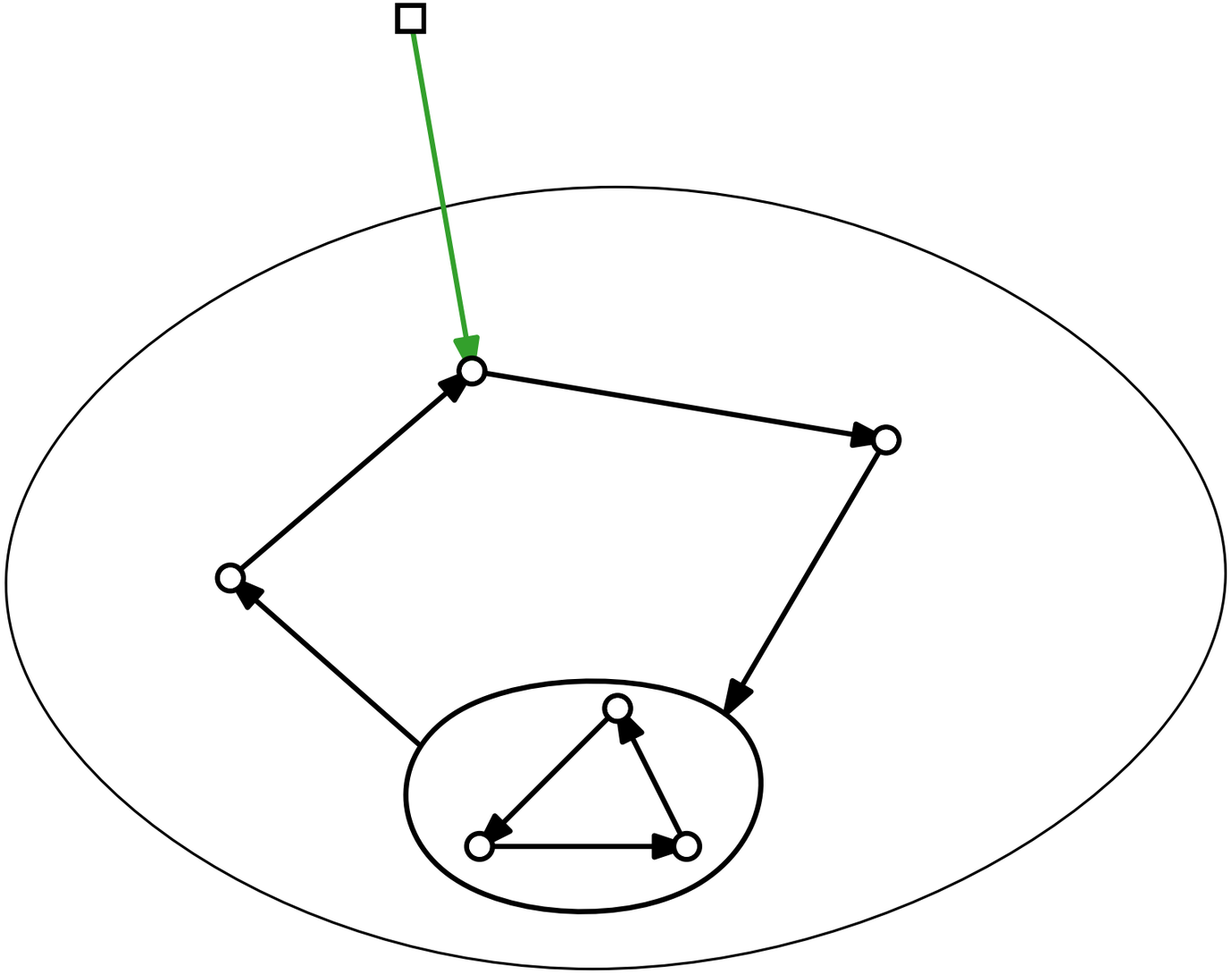
Edmonds Algorithmus – Illustration



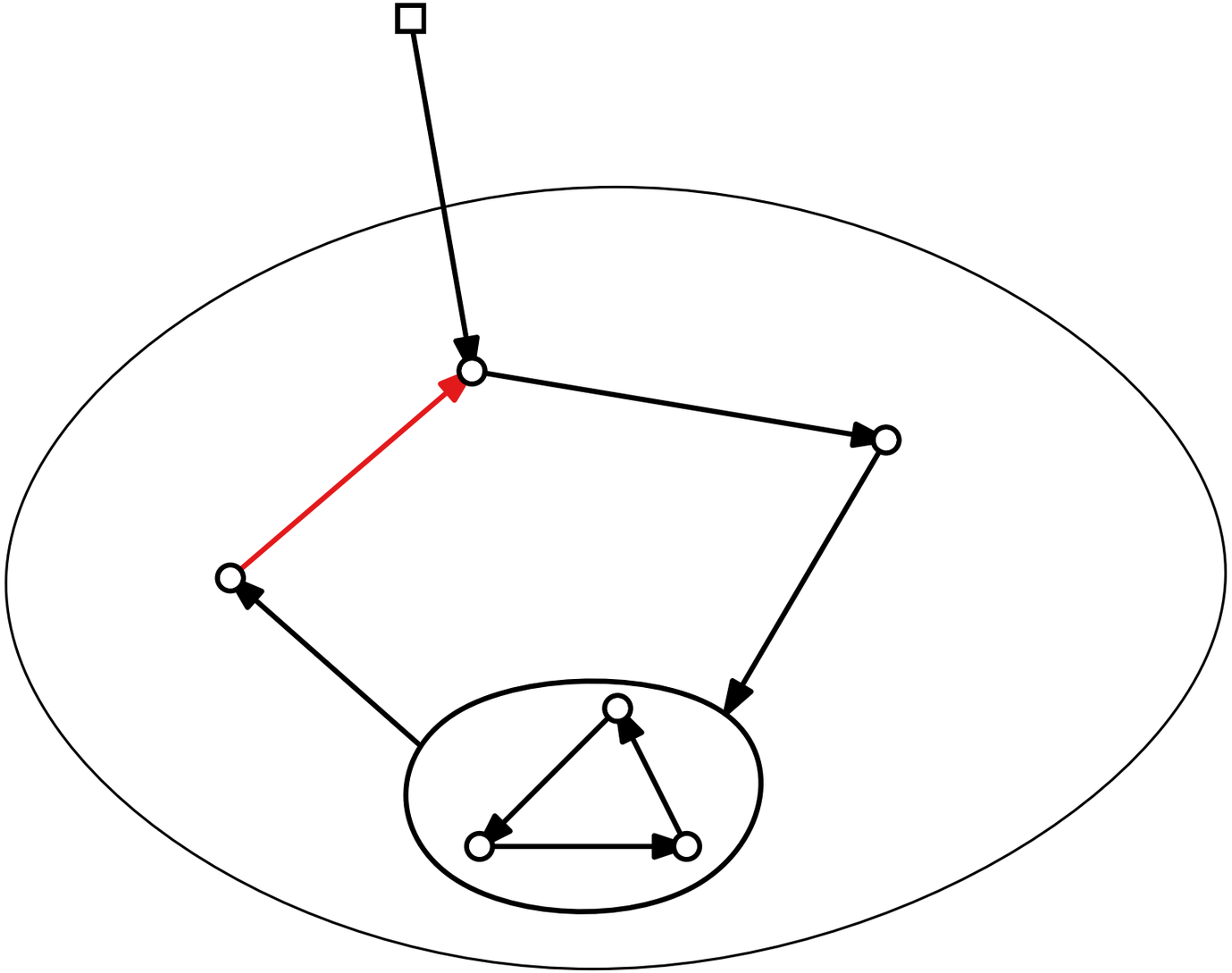
Edmonds Algorithmus – Illustration



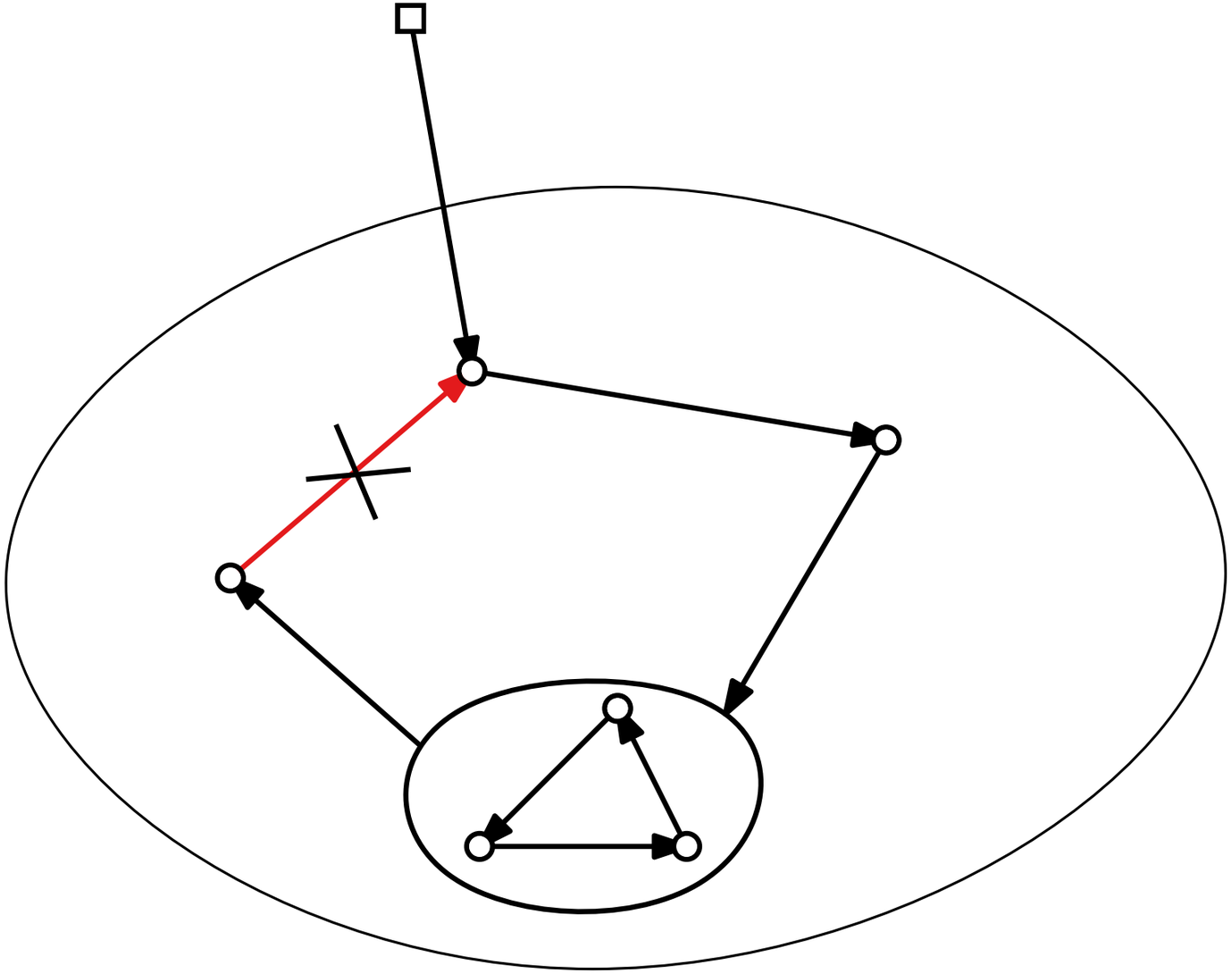
Edmonds Algorithmus – Illustration



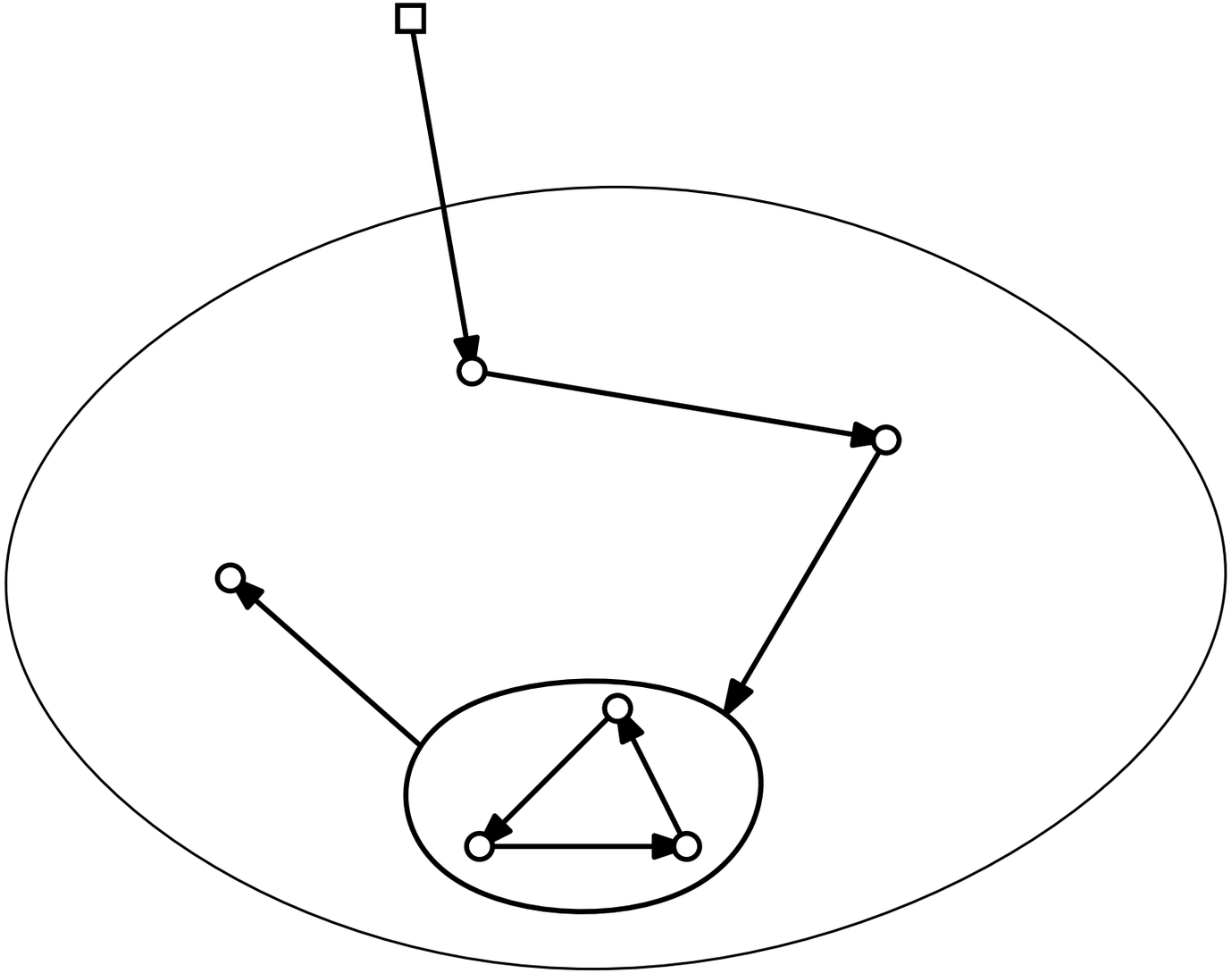
Edmonds Algorithmus – Illustration



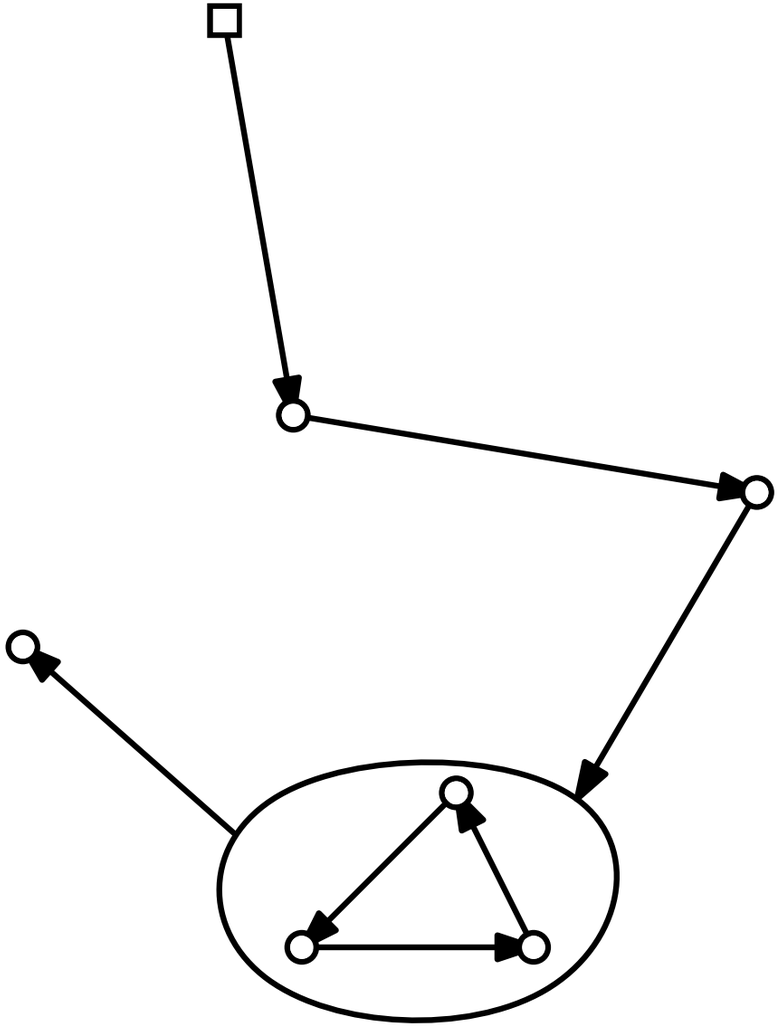
Edmonds Algorithmus – Illustration



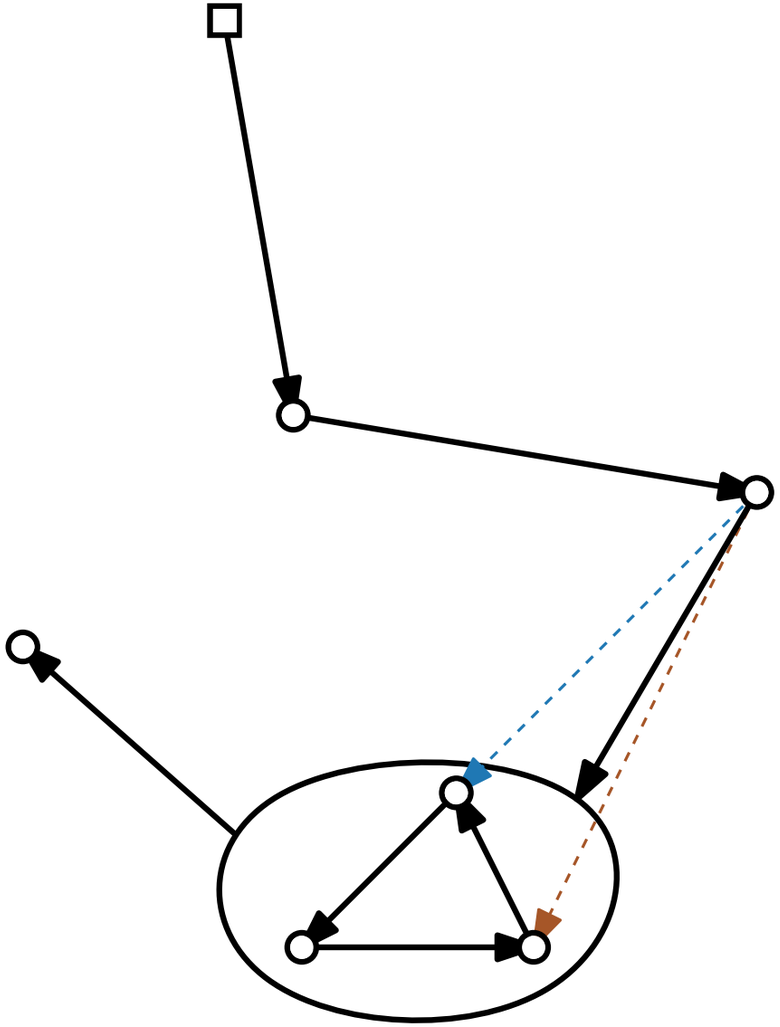
Edmonds Algorithmus – Illustration



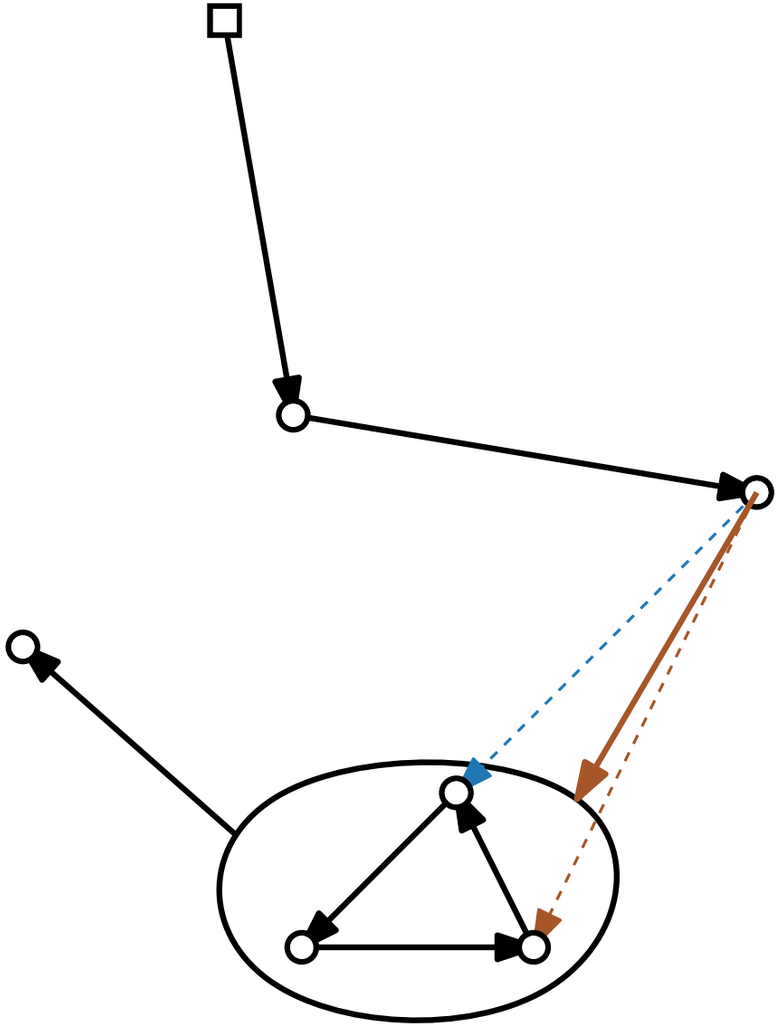
Edmonds Algorithmus – Illustration



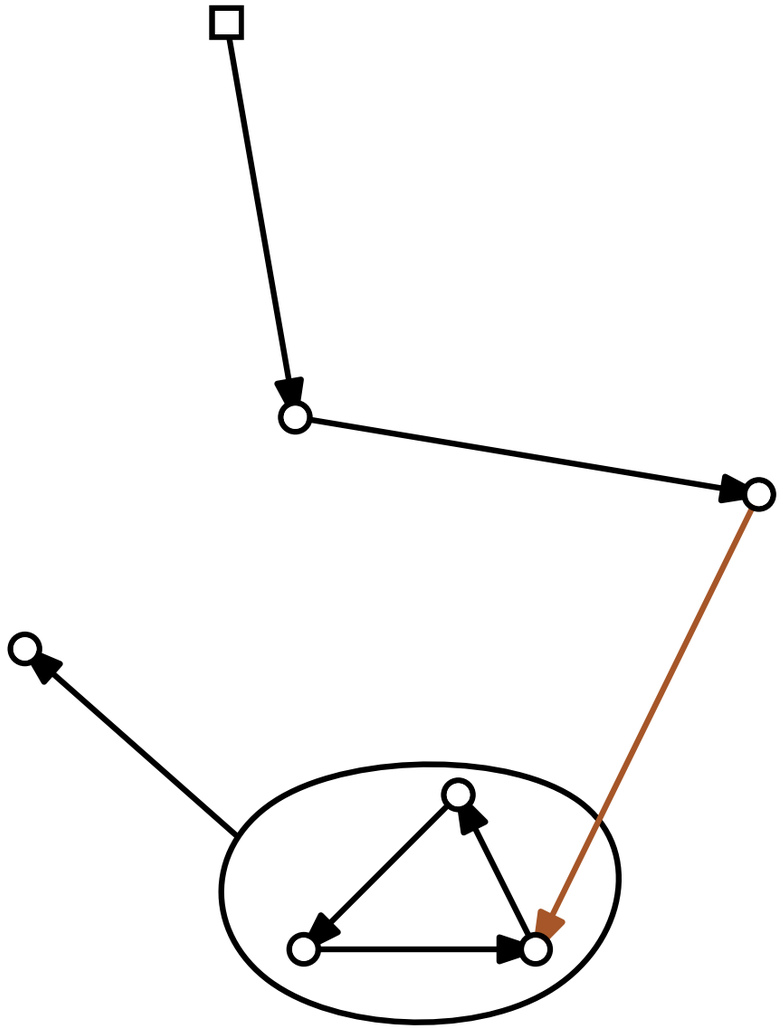
Edmonds Algorithmus – Illustration



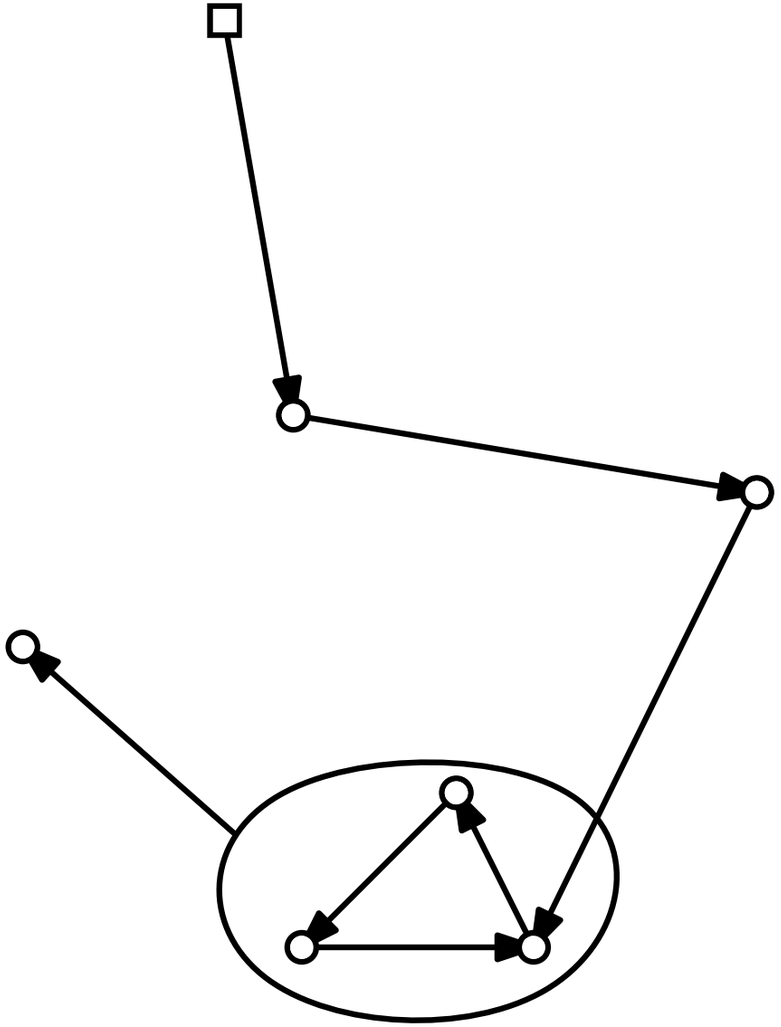
Edmonds Algorithmus – Illustration



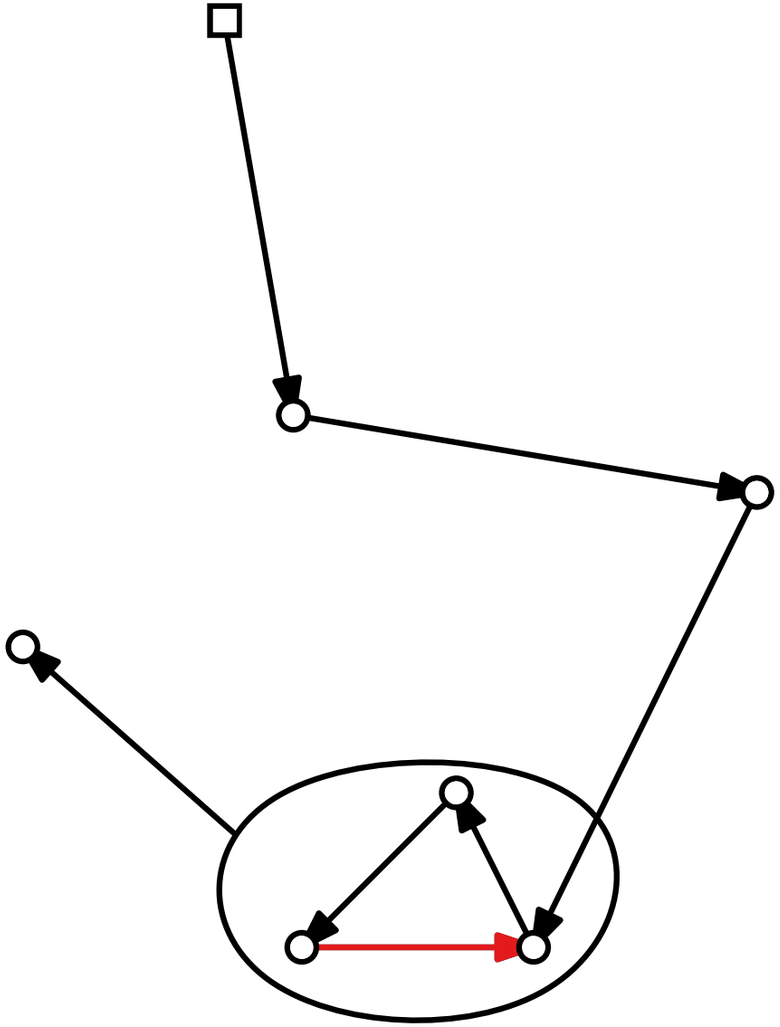
Edmonds Algorithmus – Illustration



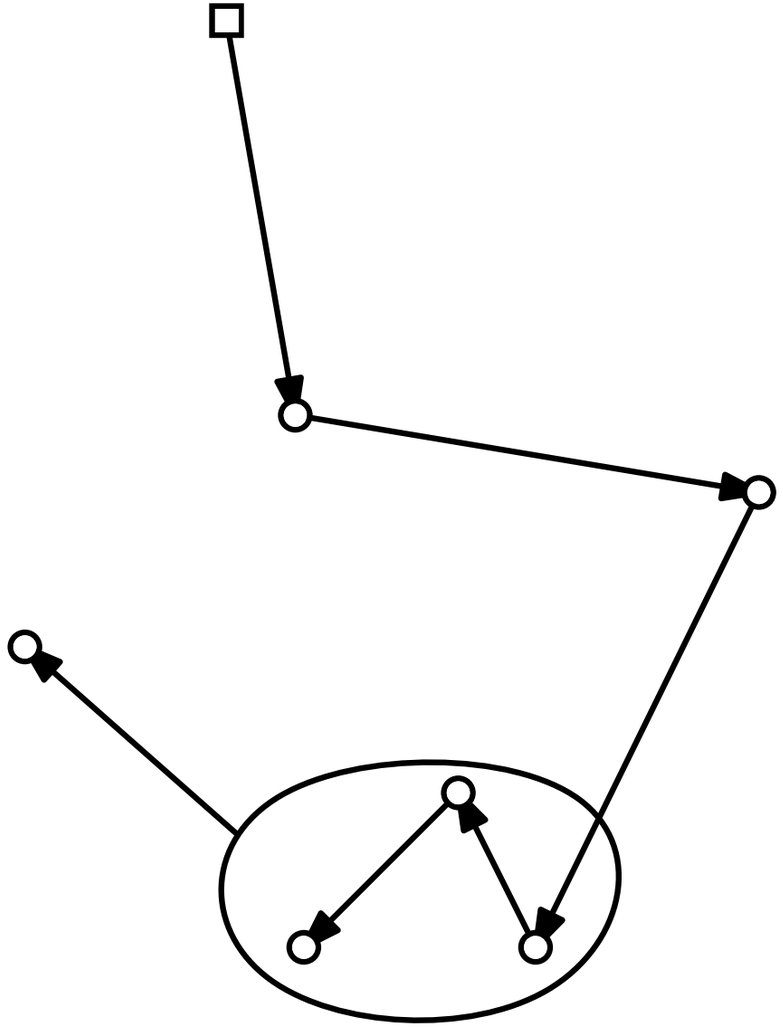
Edmonds Algorithmus – Illustration



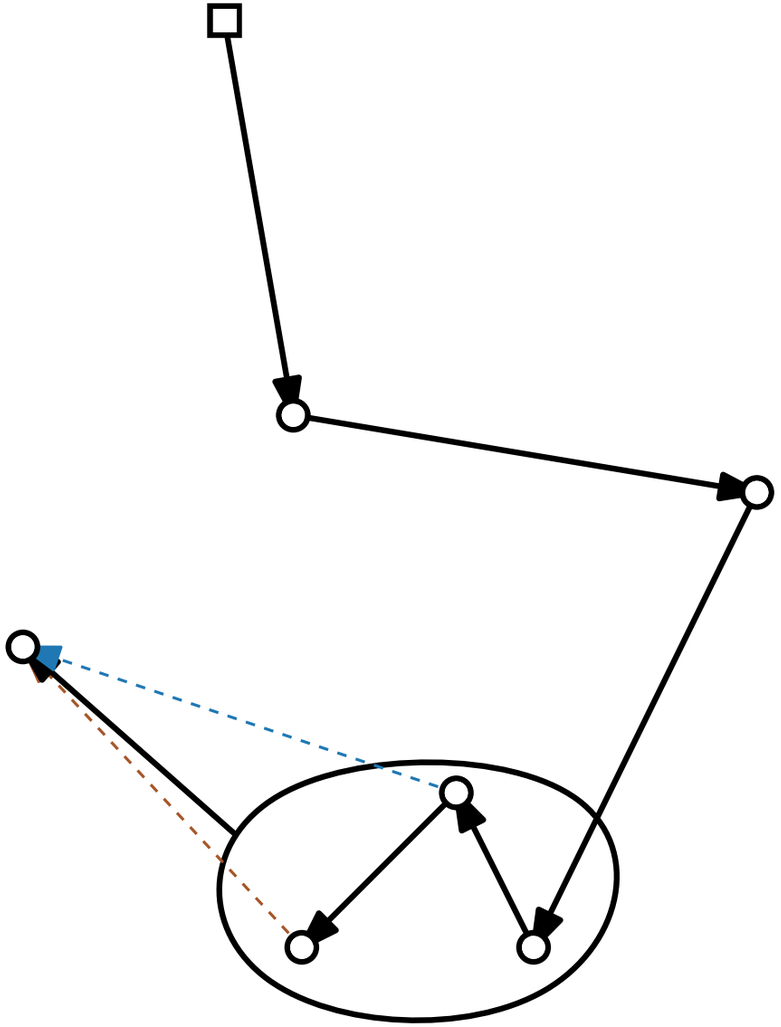
Edmonds Algorithmus – Illustration



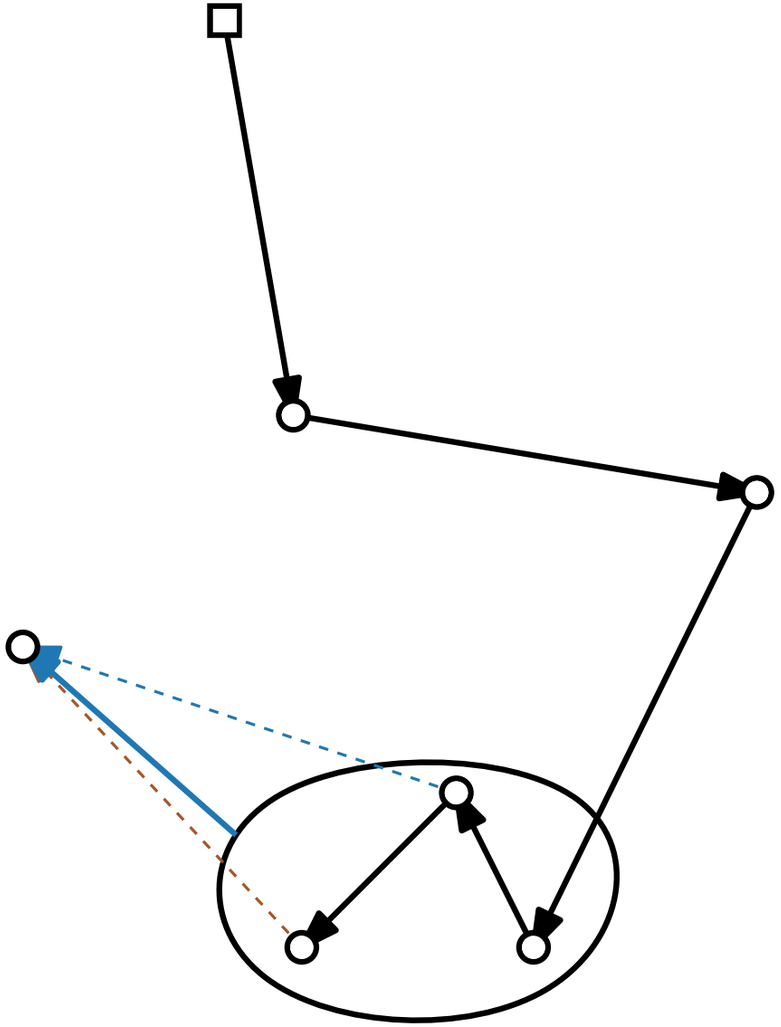
Edmonds Algorithmus – Illustration



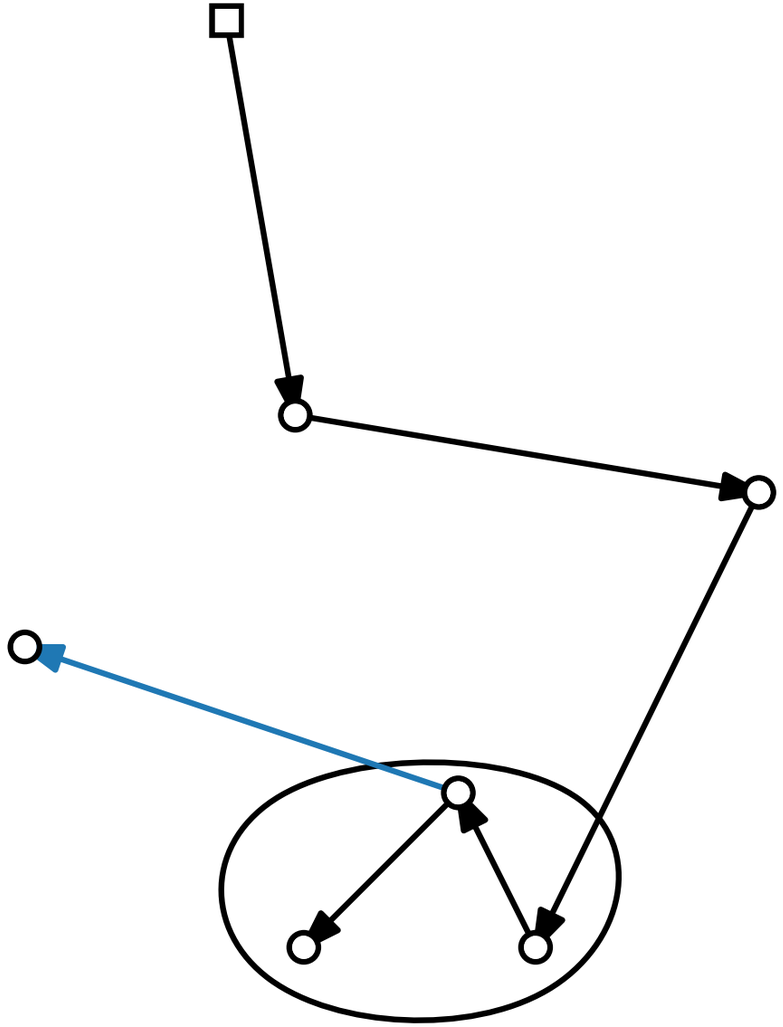
Edmonds Algorithmus – Illustration



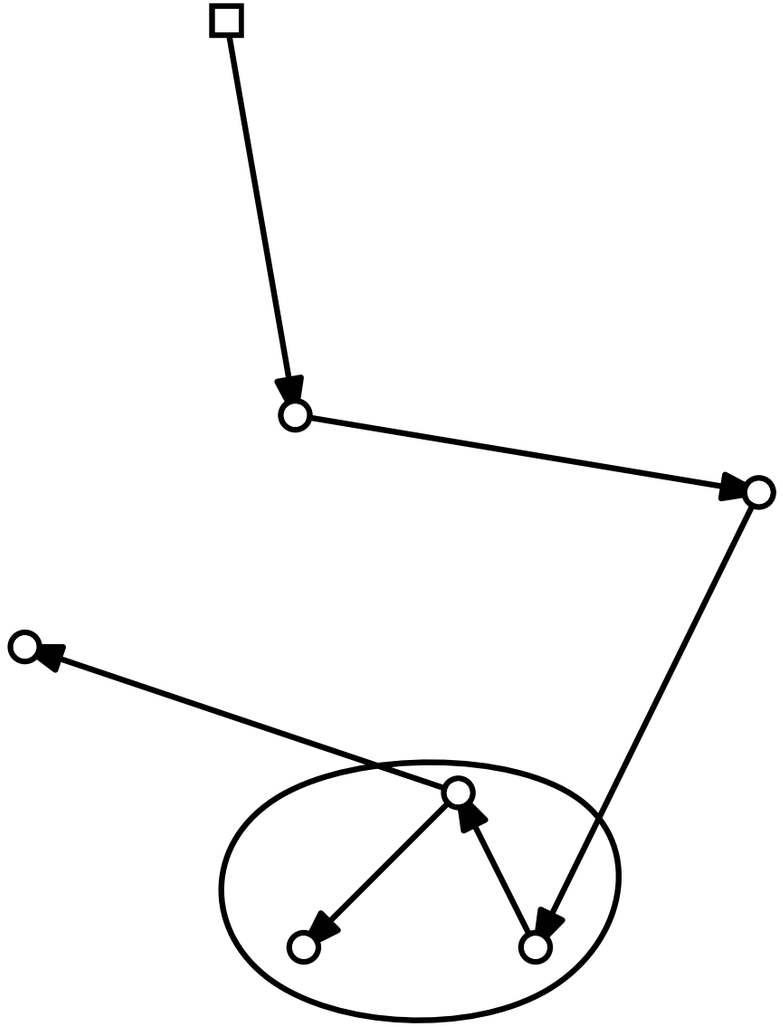
Edmonds Algorithmus – Illustration



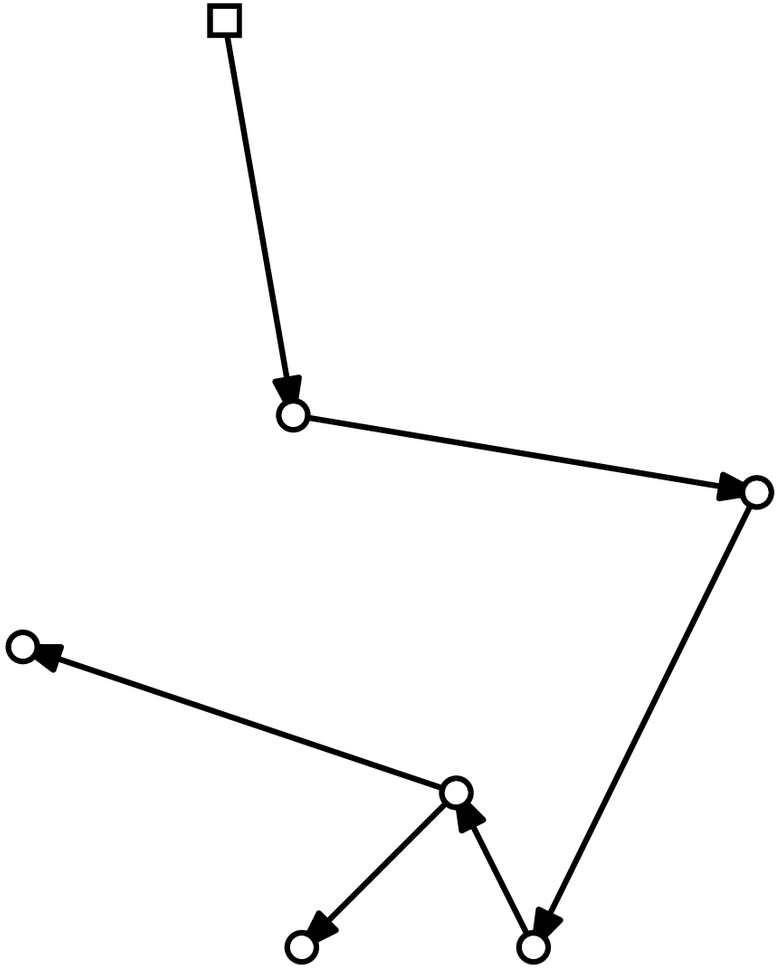
Edmonds Algorithmus – Illustration



Edmonds Algorithmus – Illustration



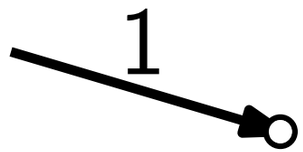
Edmonds Algorithmus – Illustration



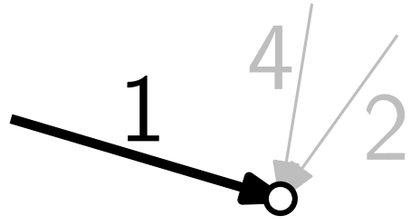
Edmonds Algorithmus – Normalisieren

Edmonds Algorithmus – Normalisieren

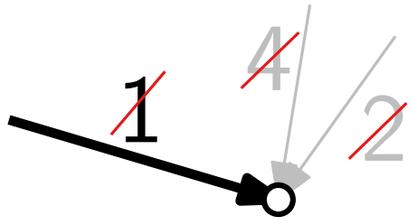
Edmonds Algorithmus – Normalisieren



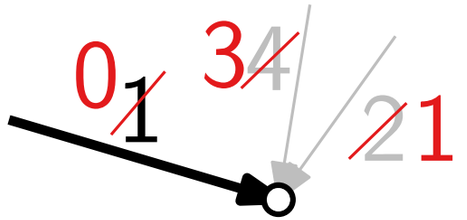
Edmonds Algorithmus – Normalisieren



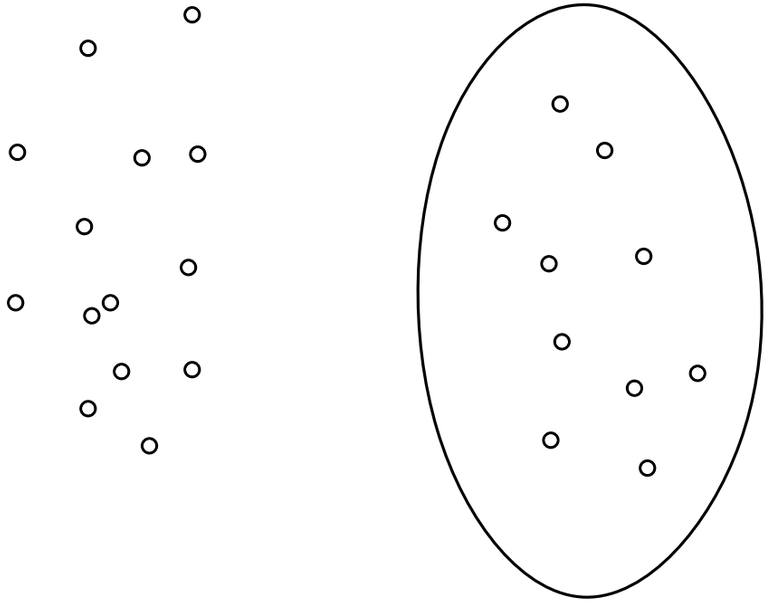
Edmonds Algorithmus – Normalisieren

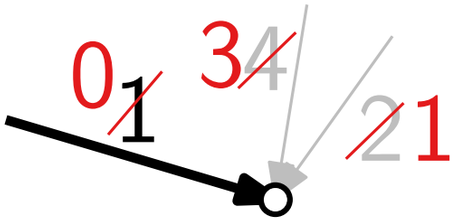


Edmonds Algorithmus – Normalisieren

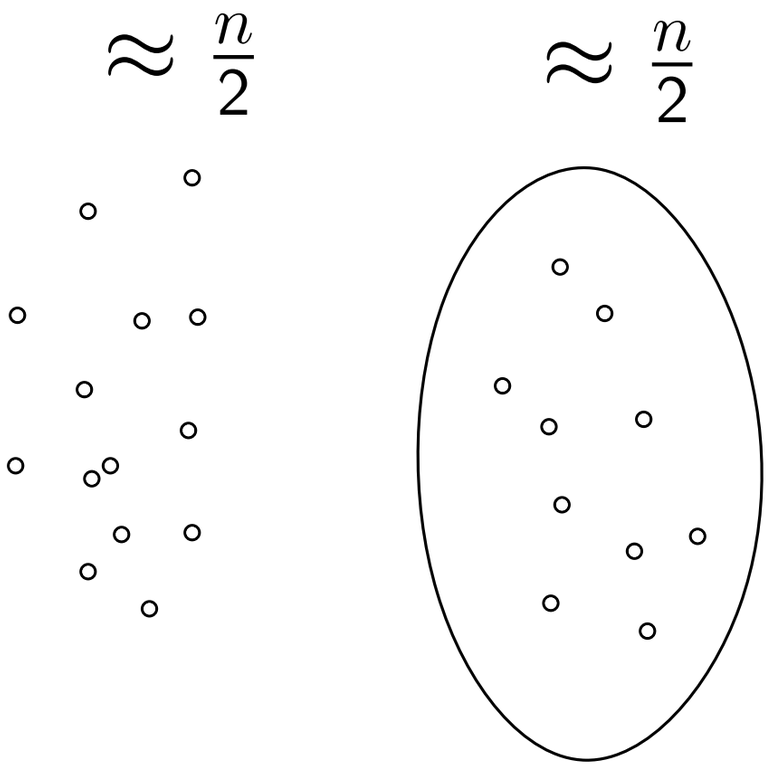


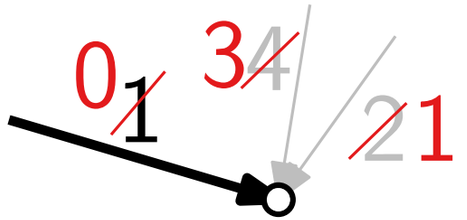
Edmonds Algorithmus – Normalisieren



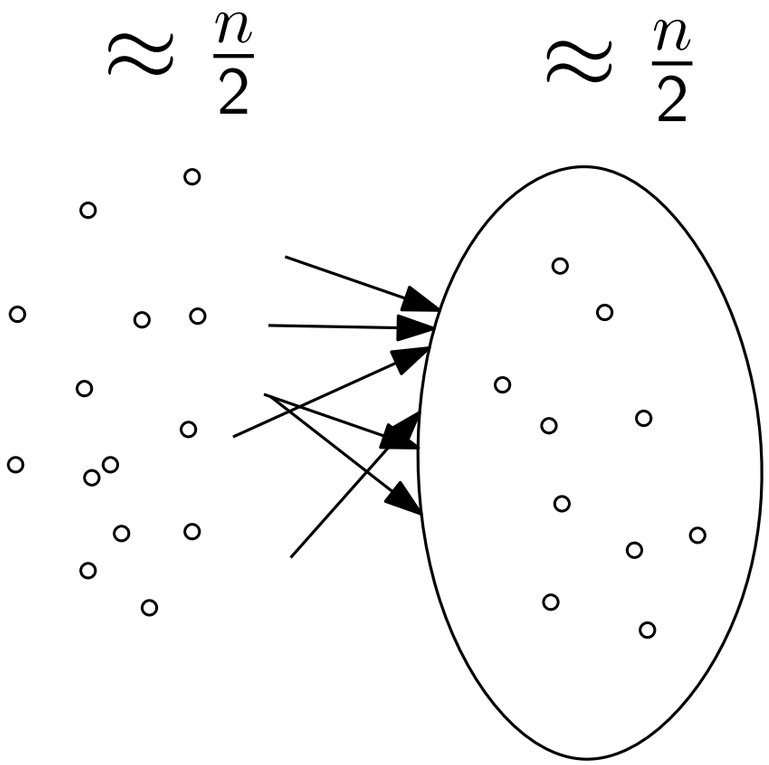


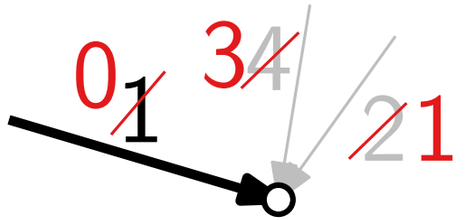
Edmonds Algorithmus – Normalisieren



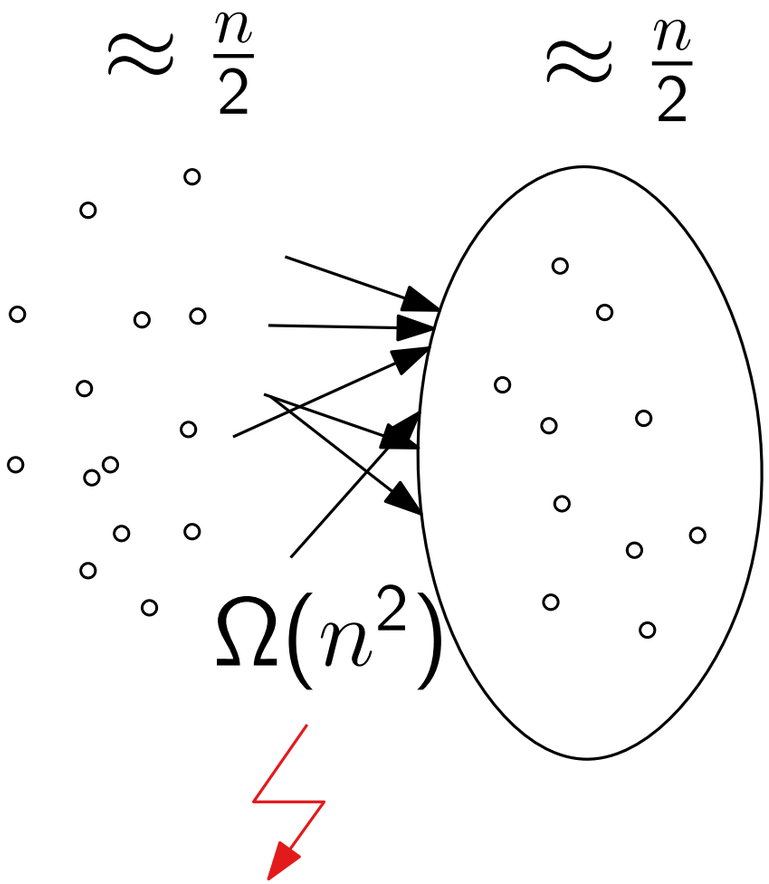


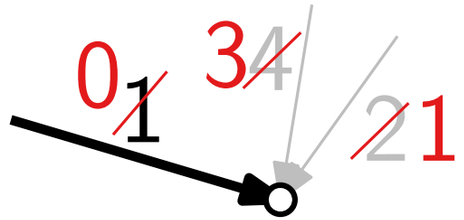
Edmonds Algorithmus – Normalisieren



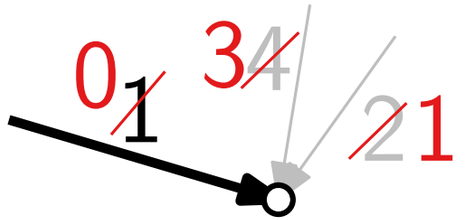


Edmonds Algorithmus – Normalisieren

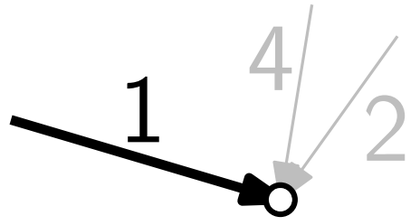




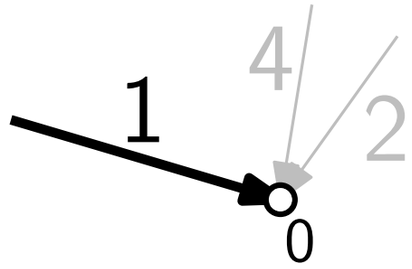
Edmonds Algorithmus – Normalisieren



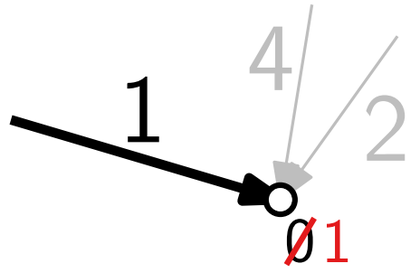
Edmonds Algorithmus – Normalisieren



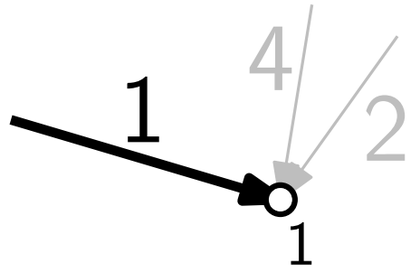
Edmonds Algorithmus – Normalisieren



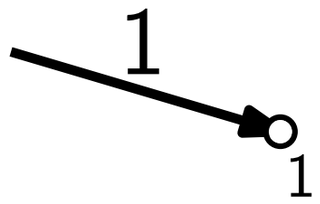
Edmonds Algorithmus – Normalisieren



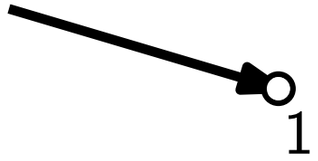
Edmonds Algorithmus – Normalisieren



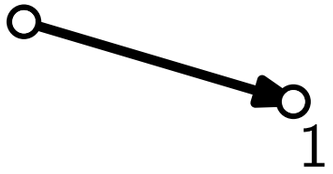
Edmonds Algorithmus – Normalisieren



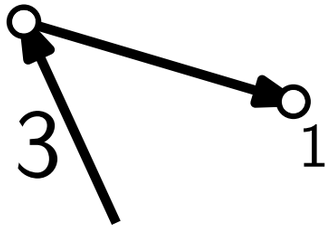
Edmonds Algorithmus – Normalisieren



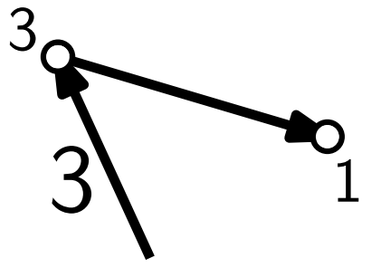
Edmonds Algorithmus – Normalisieren



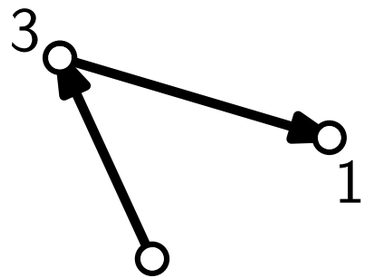
Edmonds Algorithmus – Normalisieren



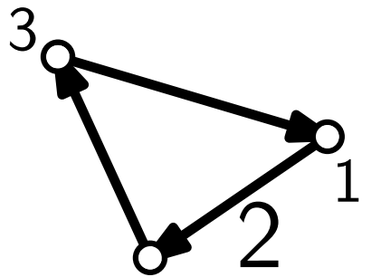
Edmonds Algorithmus – Normalisieren



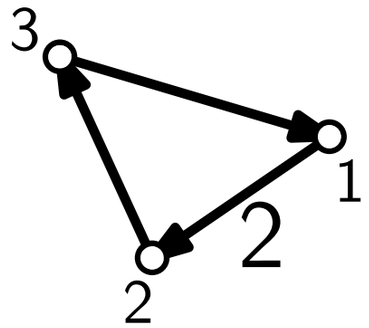
Edmonds Algorithmus – Normalisieren



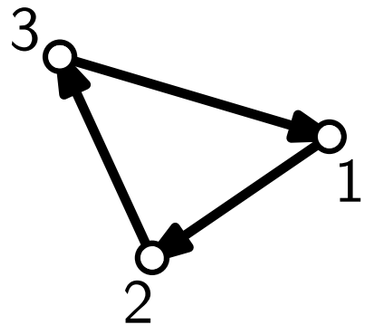
Edmonds Algorithmus – Normalisieren



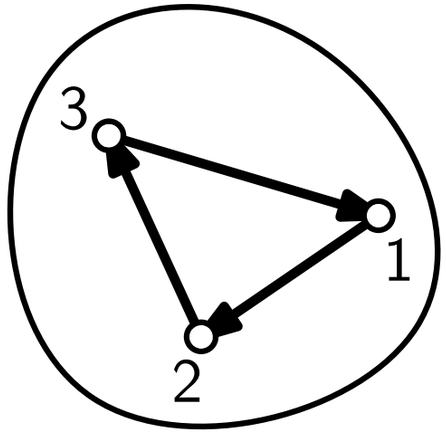
Edmonds Algorithmus – Normalisieren



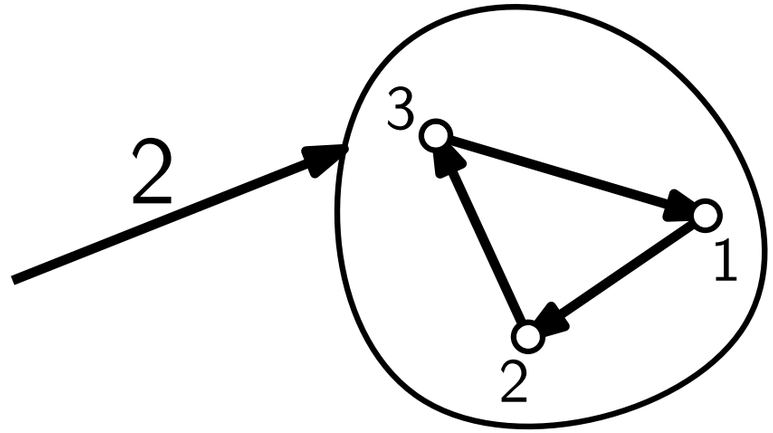
Edmonds Algorithmus – Normalisieren



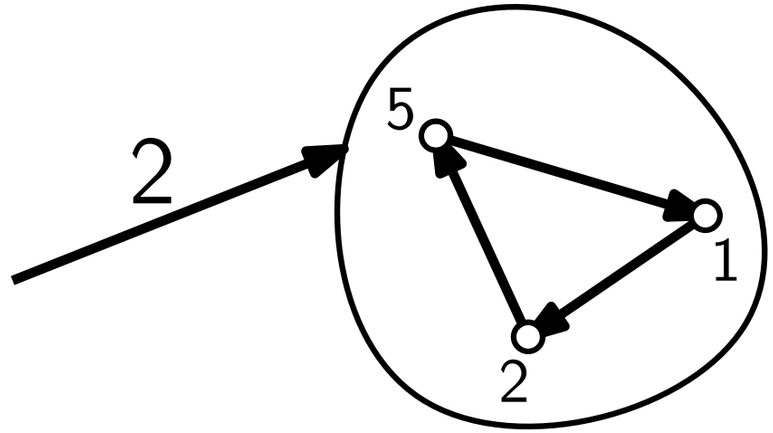
Edmonds Algorithmus – Normalisieren



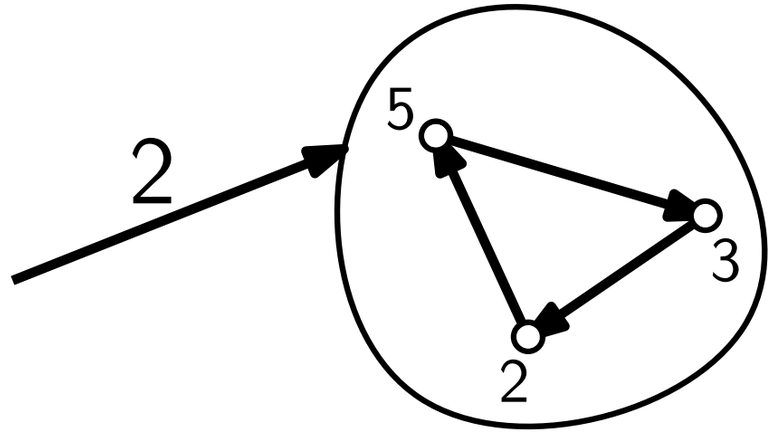
Edmonds Algorithmus – Normalisieren



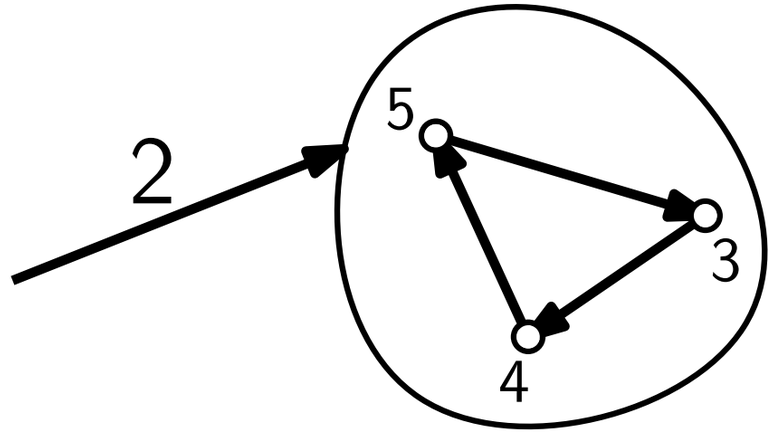
Edmonds Algorithmus – Normalisieren



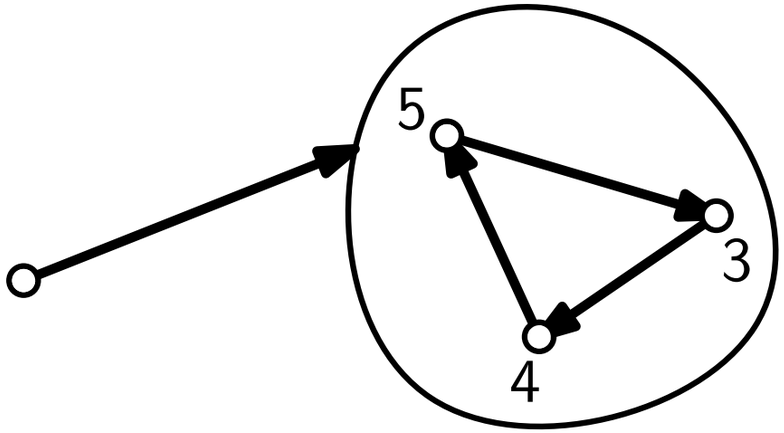
Edmonds Algorithmus – Normalisieren



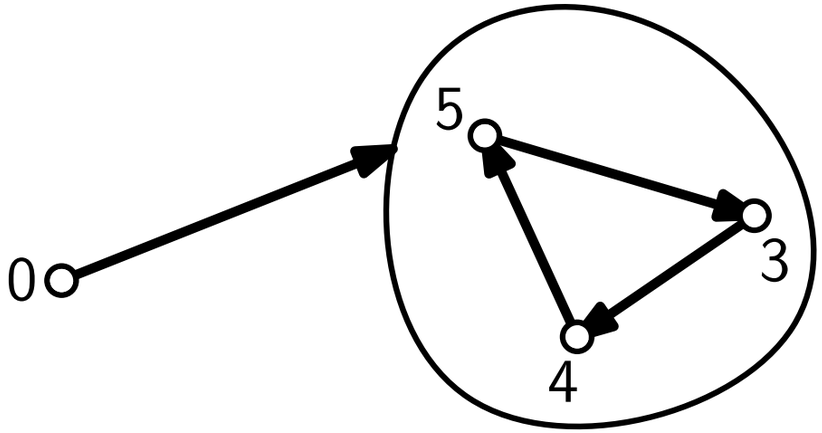
Edmonds Algorithmus – Normalisieren



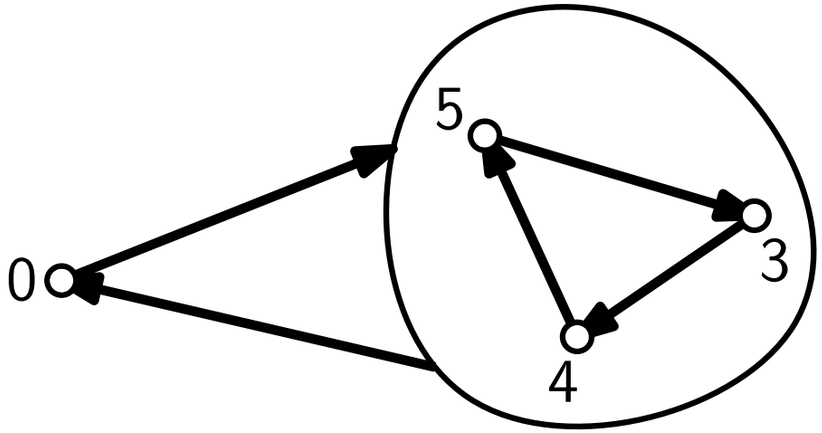
Edmonds Algorithmus – Normalisieren



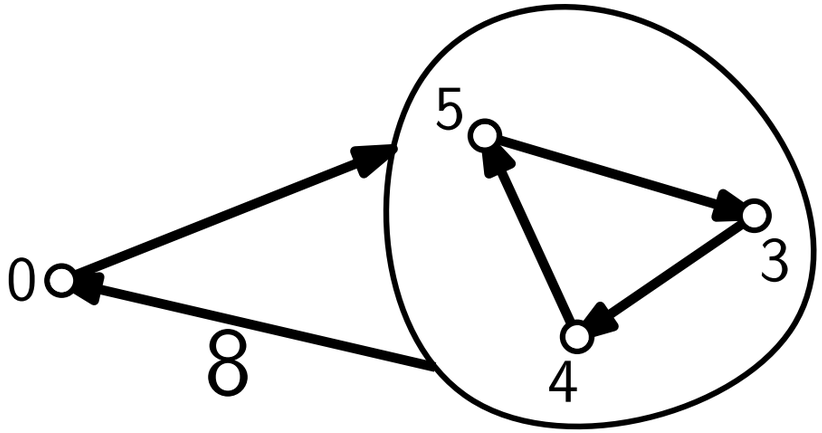
Edmonds Algorithmus – Normalisieren



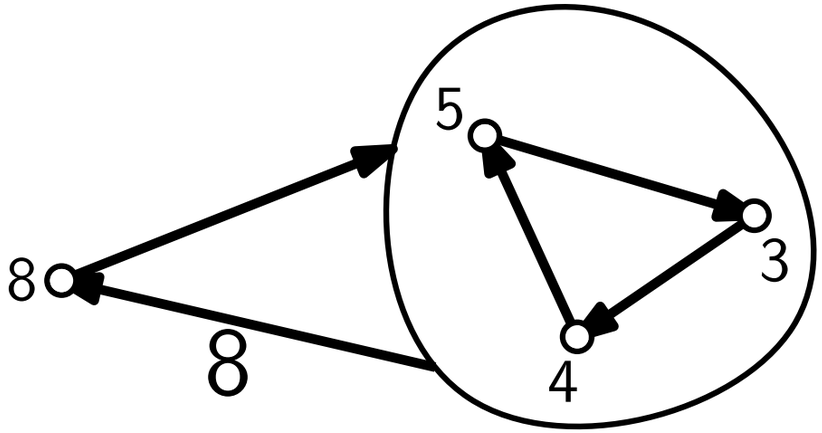
Edmonds Algorithmus – Normalisieren



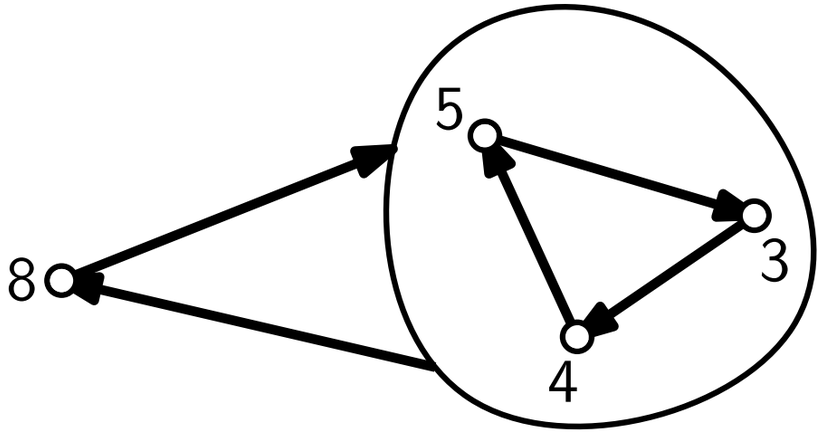
Edmonds Algorithmus – Normalisieren



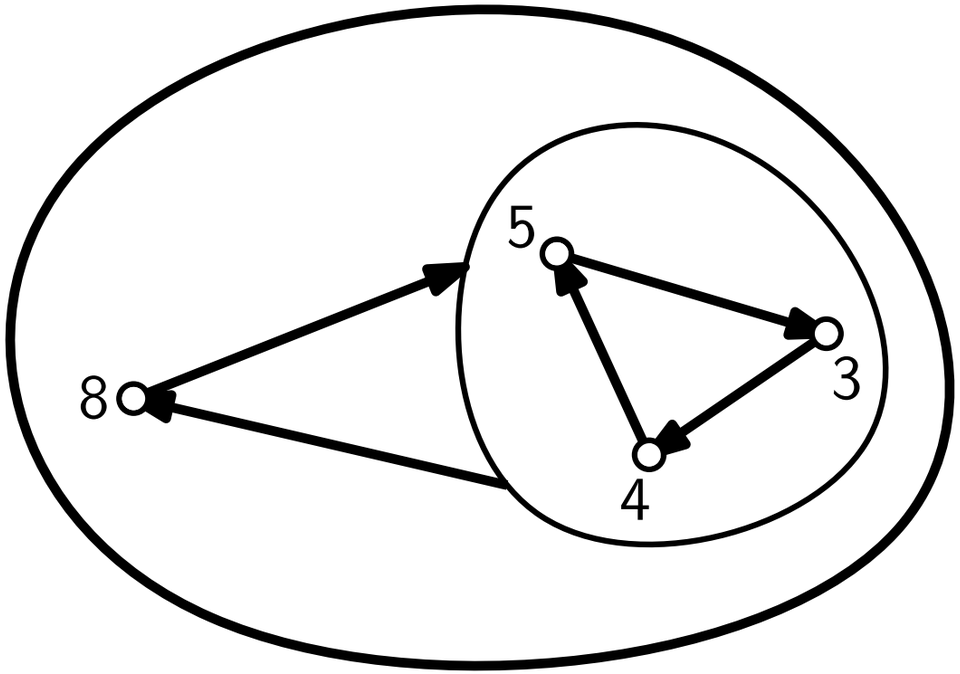
Edmonds Algorithmus – Normalisieren



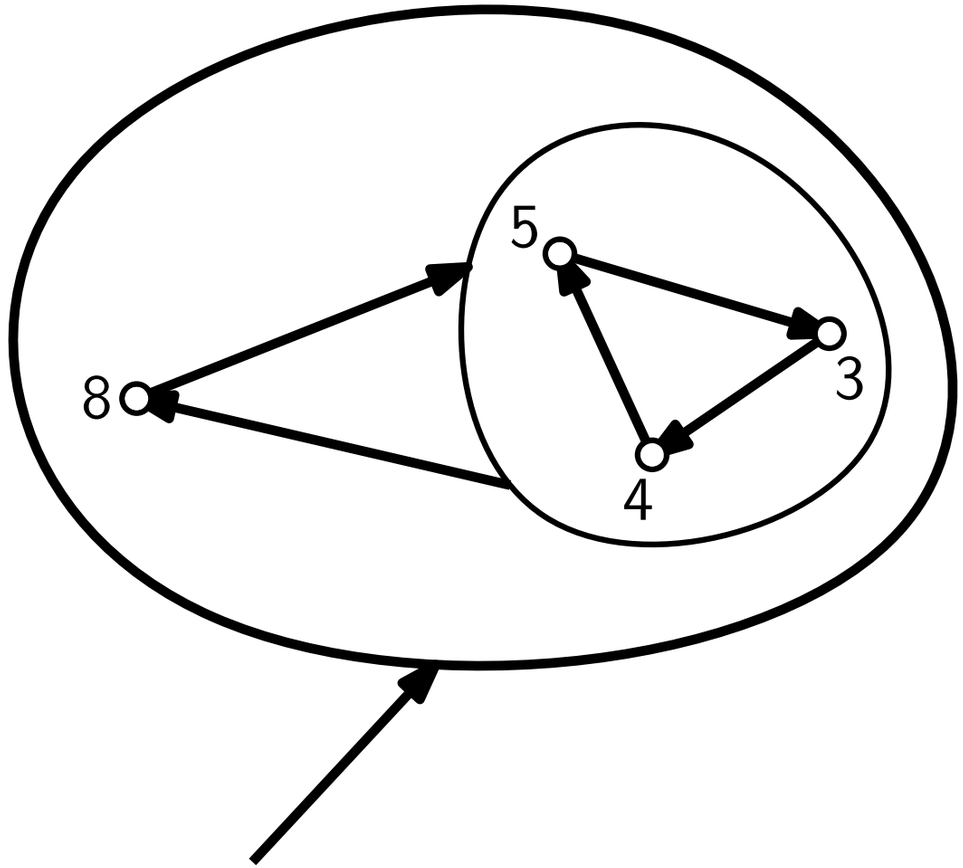
Edmonds Algorithmus – Normalisieren



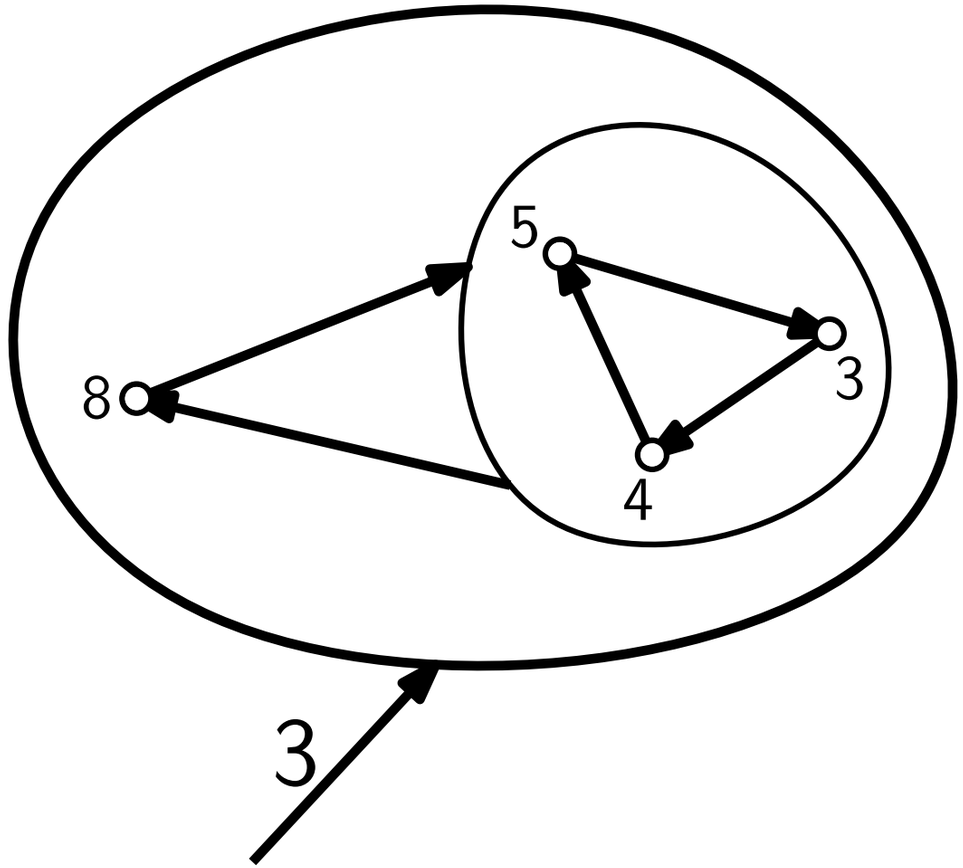
Edmonds Algorithmus – Normalisieren



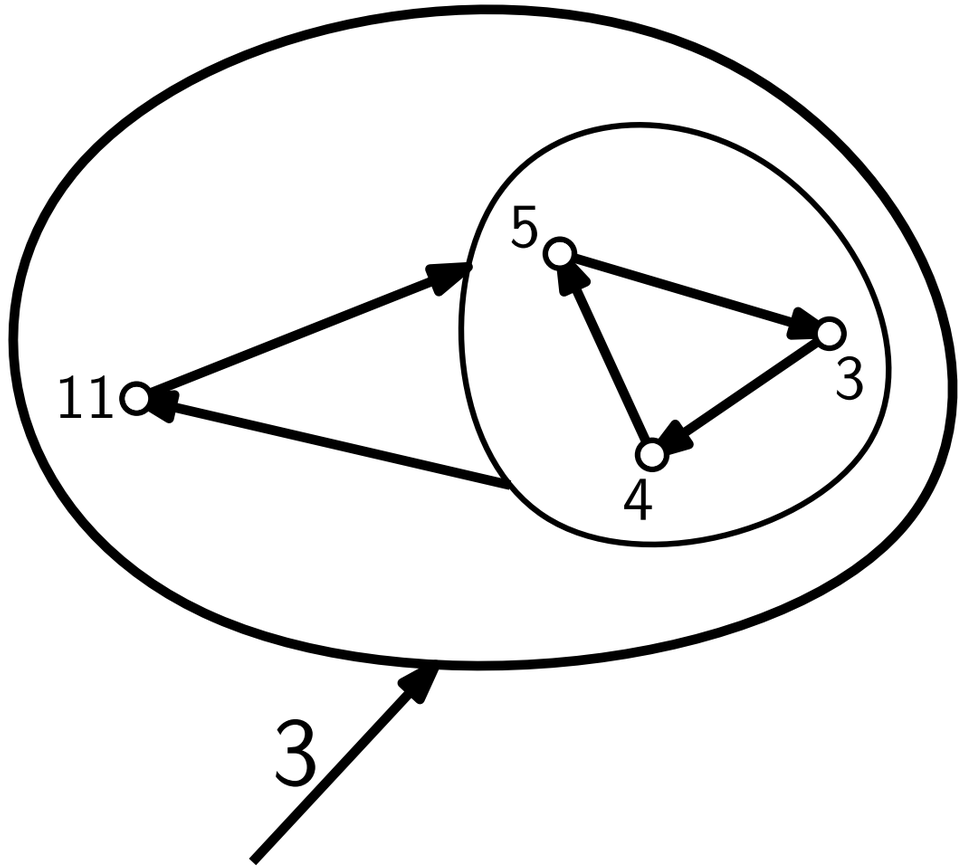
Edmonds Algorithmus – Normalisieren



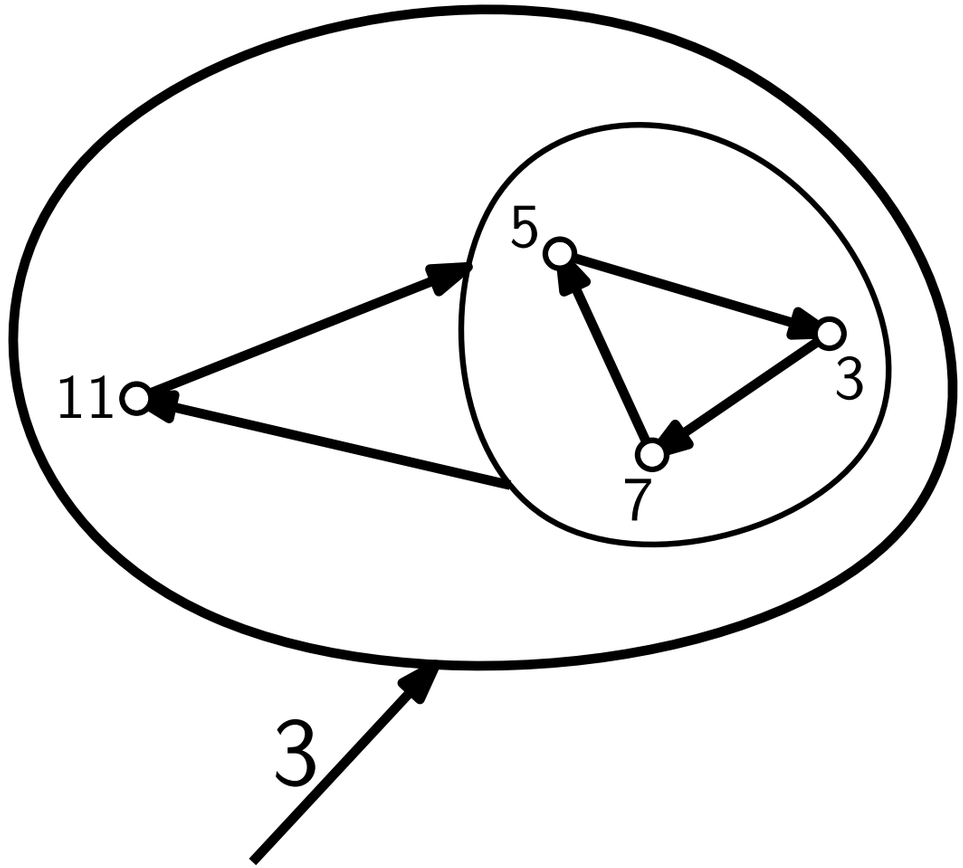
Edmonds Algorithmus – Normalisieren



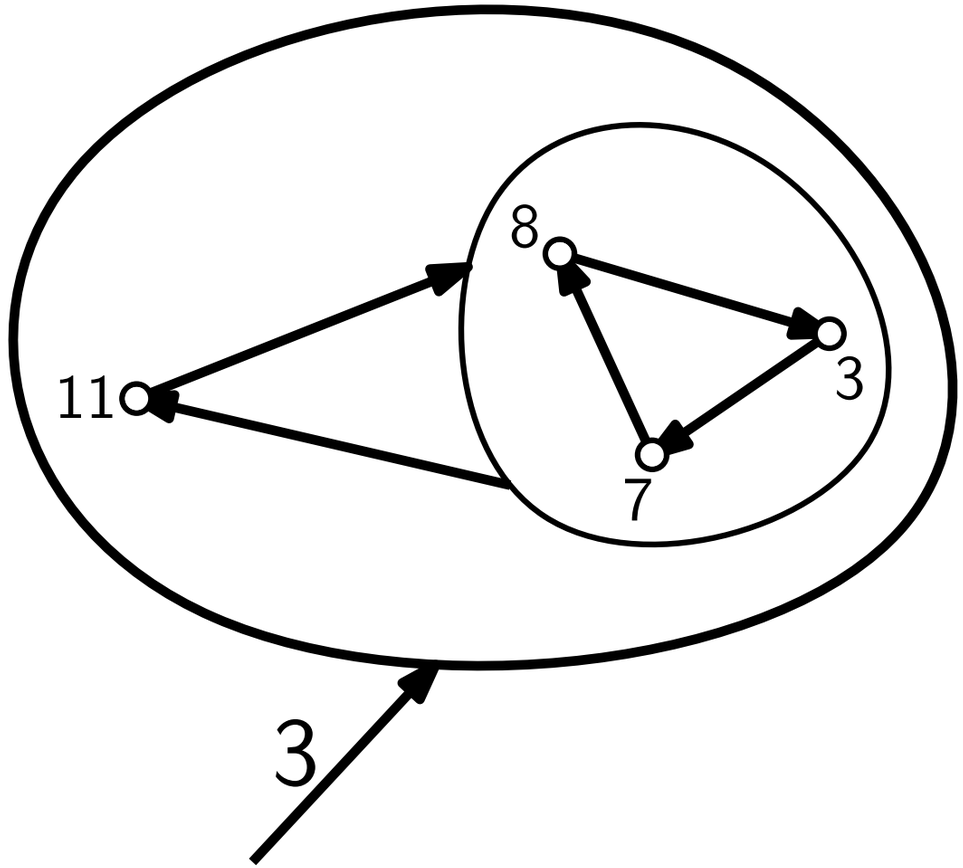
Edmonds Algorithmus – Normalisieren



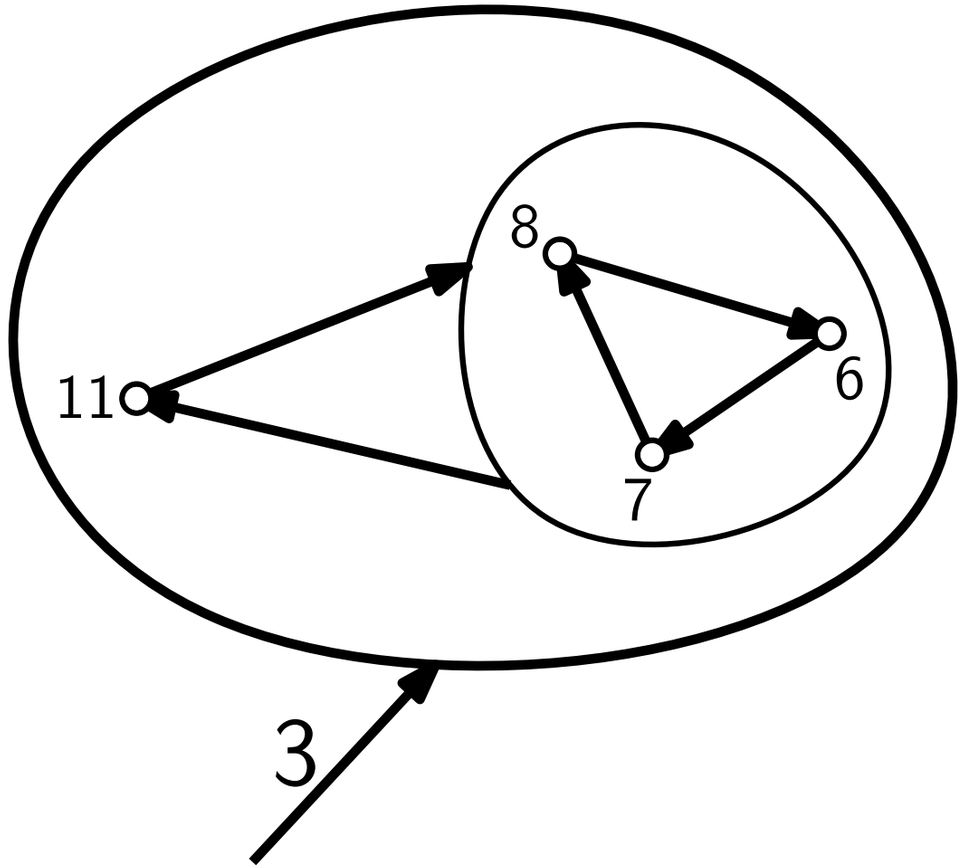
Edmonds Algorithmus – Normalisieren



Edmonds Algorithmus – Normalisieren



Edmonds Algorithmus – Normalisieren

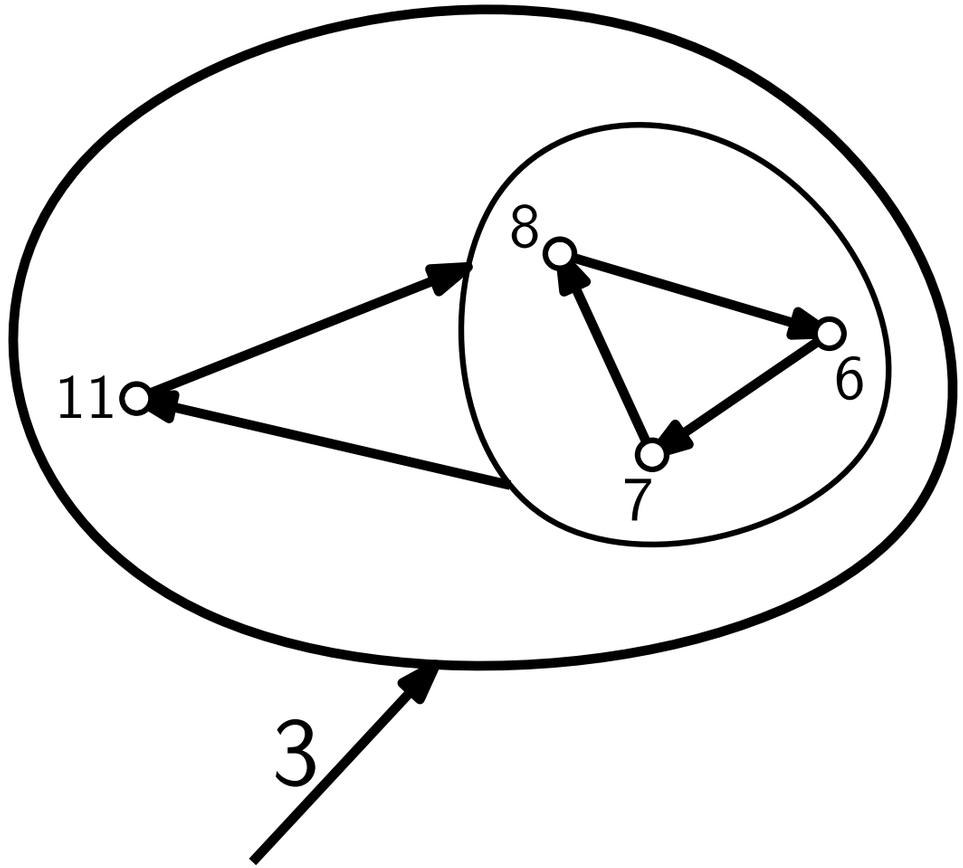


Edmonds Algorithmus – Normalisieren

normalisiere(Knoten v, Int c)

foreach u **in** v **do**

$u.c_0 \leftarrow u.c_0 + c$



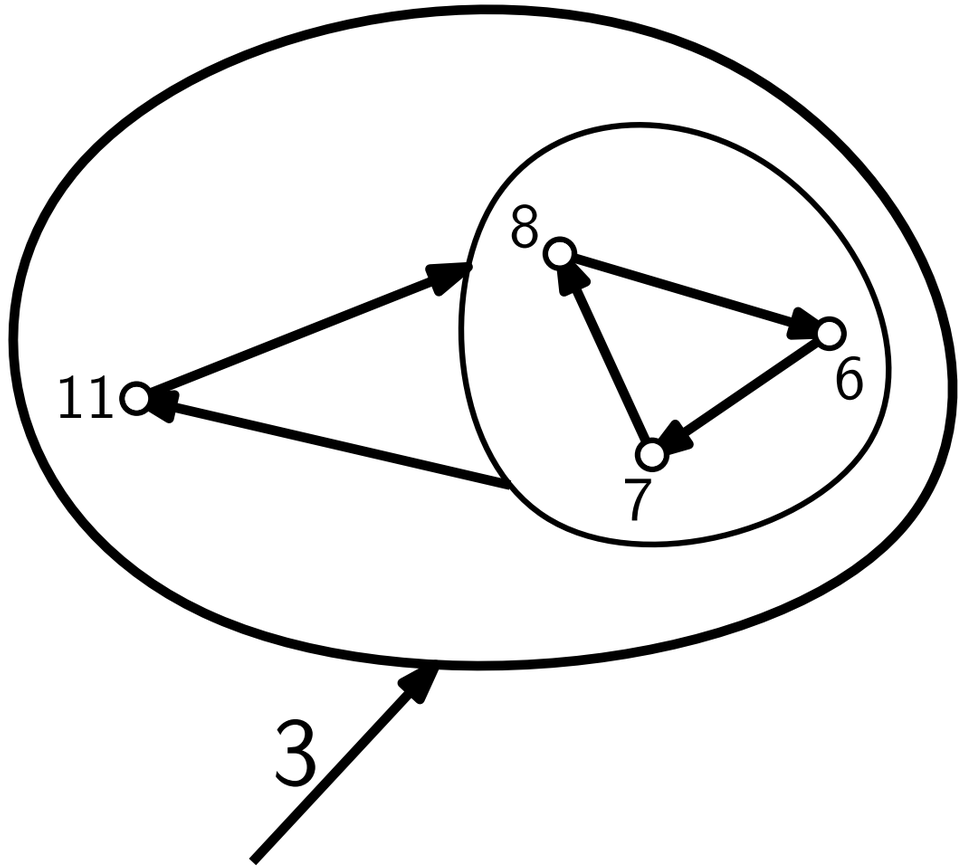
Edmonds Algorithmus – Normalisieren

normalisiere(Knoten v, Int c)

foreach u **in** v **do**

$u.c_0 \leftarrow u.c_0 + c$

Laufzeit $O(n)$

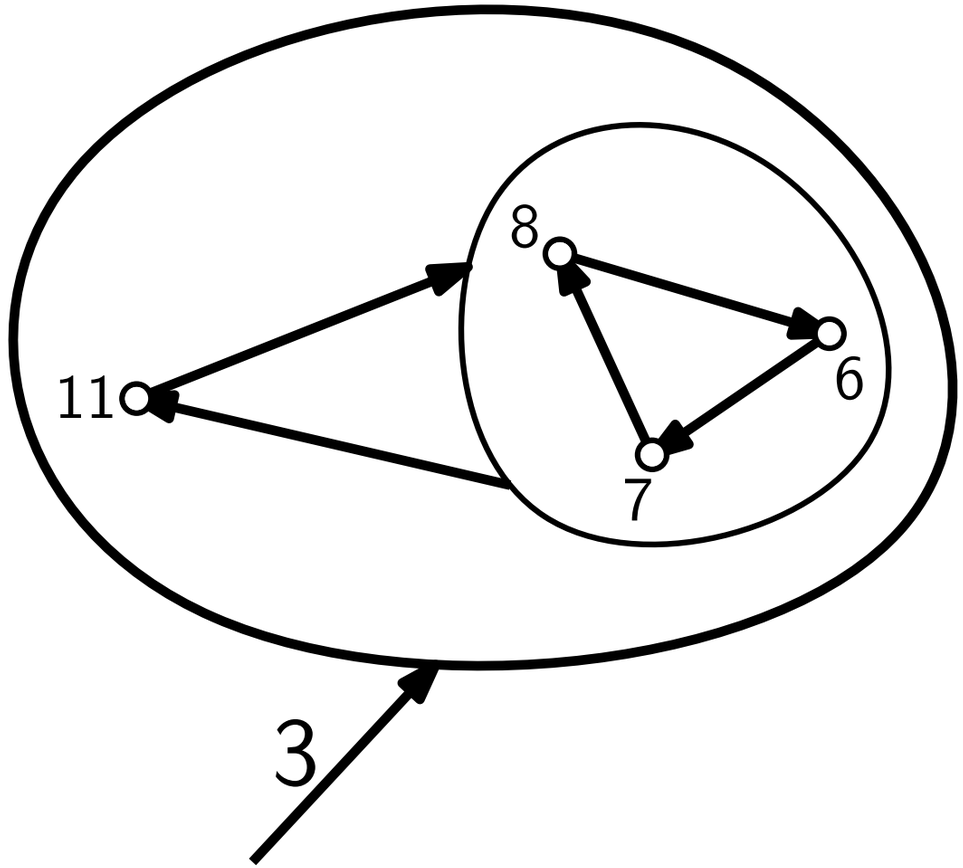


Edmonds Algorithmus – Normalisieren

normalisiere(Knoten v, Int c)

foreach u **in** v **do**

$u.c_0 \leftarrow u.c_0 + c$



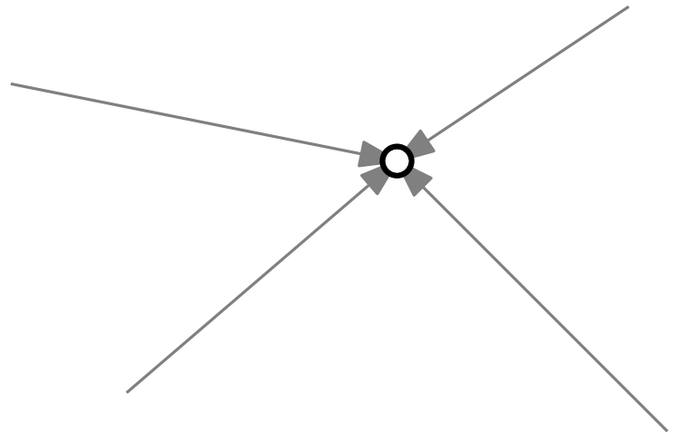
Laufzeit $O(n)$

$\Rightarrow$ Items in v in $O(n)$ iterierbar.

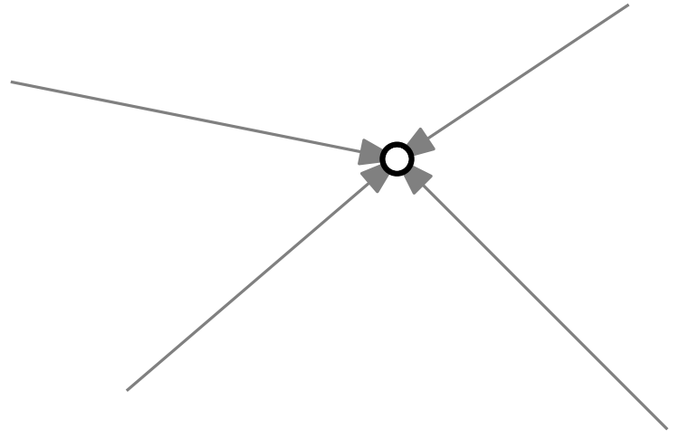
Edmonds Algorithmus – Billigeste Kante

Edmonds Algorithmus – Billigeste Kante

Edmonds Algorithmus – Billigeste Kante

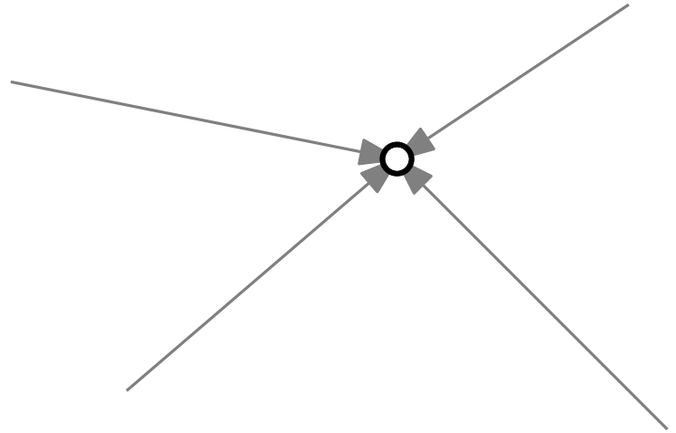


Edmonds Algorithmus – Billigeste Kante



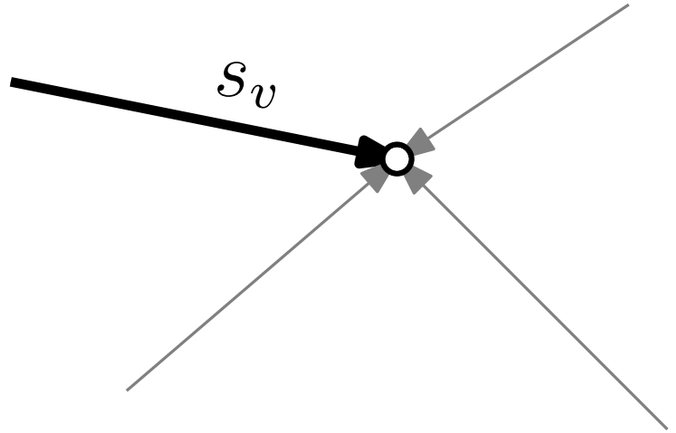
SC-Item ist mind. so gut wie alle anderen.

Edmonds Algorithmus – Billigeste Kante



SC-Item ist mind. so gut wie alle anderen.

Edmonds Algorithmus – Billigeste Kante

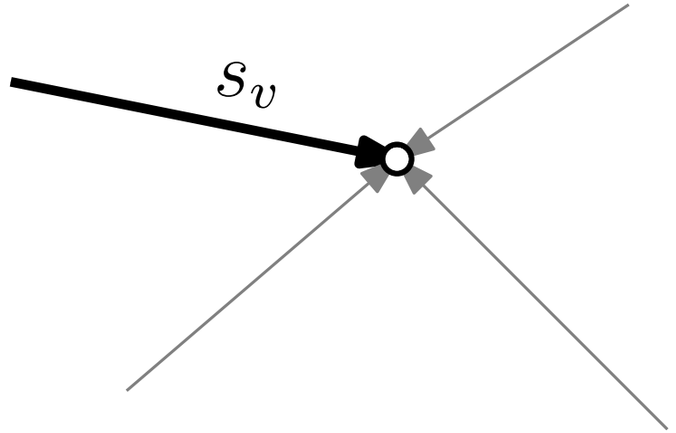


SC-Item ist mind. so gut wie alle anderen.

Edmonds Algorithmus – Billigeste Kante

billigsteKante(Knoten v, Int c)

if v ist ein Item **then**
  return (x_v, v)

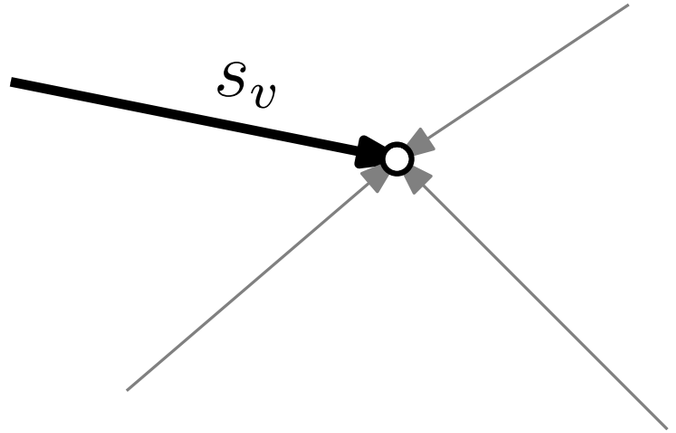


SC-Item ist mind. so gut wie alle anderen.

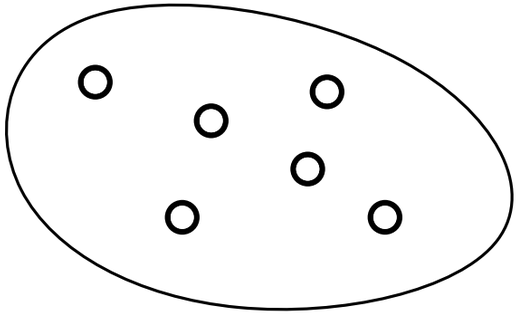
Edmonds Algorithmus – Billigeste Kante

billigsteKante(Knoten v, Int c)

if v ist ein Item **then**
  return (x_v, v)



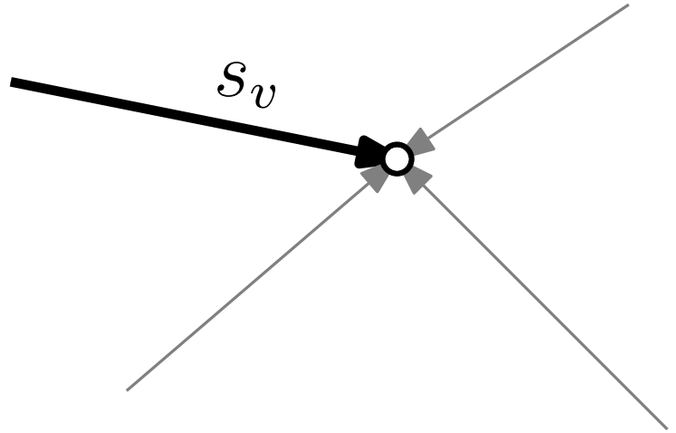
SC-Item ist mind. so gut wie alle anderen.



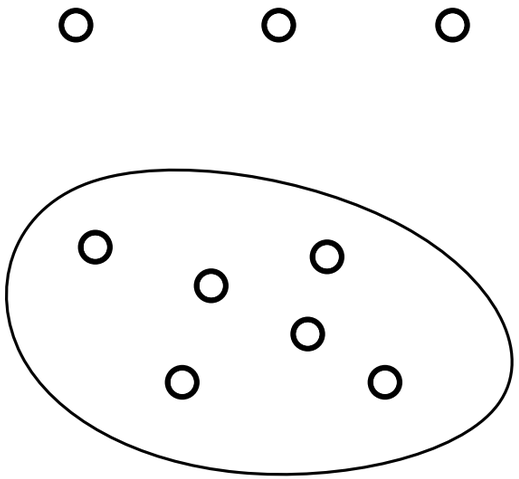
Edmonds Algorithmus – Billigeste Kante

billigsteKante(Knoten v, Int c)

if v ist ein Item **then**
  return (x_v, v)



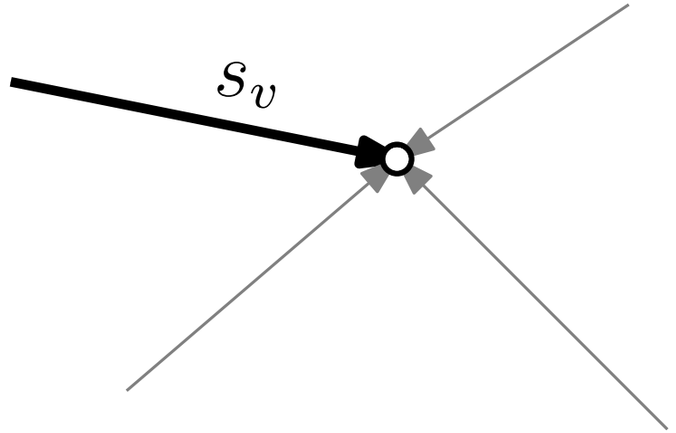
SC-Item ist mind. so gut wie alle anderen.



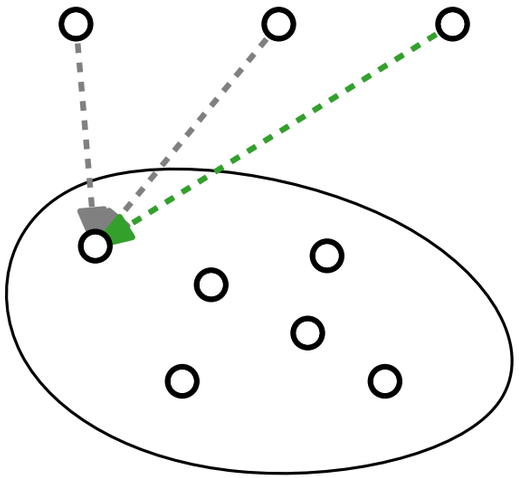
Edmonds Algorithmus – Billigeste Kante

billigsteKante(Knoten v, Int c)

if v ist ein Item **then**
  return (x_v, v)



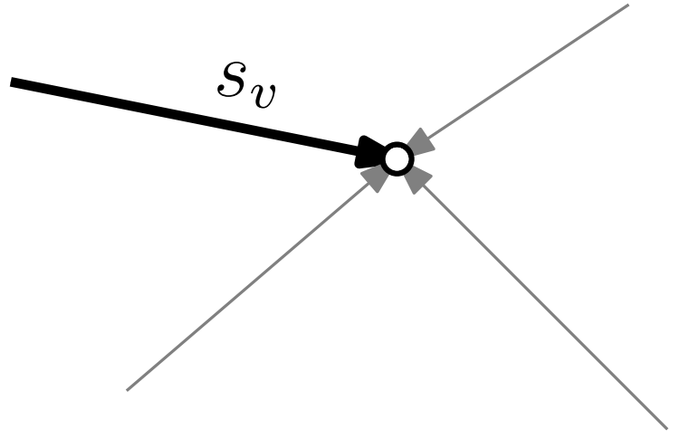
SC-Item ist mind. so gut wie alle anderen.



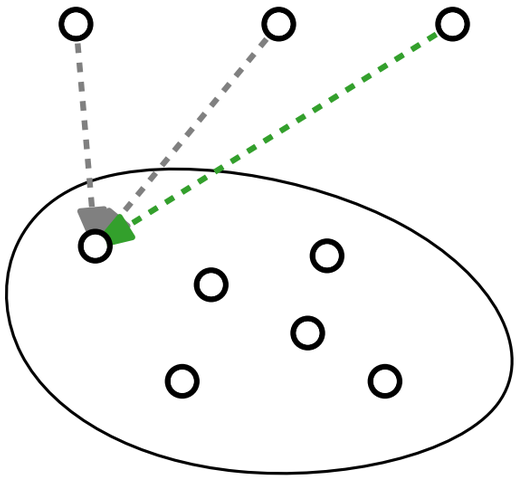
Edmonds Algorithmus – Billigeste Kante

billigsteKante(Knoten v, Int c)

if v ist ein Item **then**
  return (x_v, v)



SC-Item ist mind. so gut wie alle anderen.



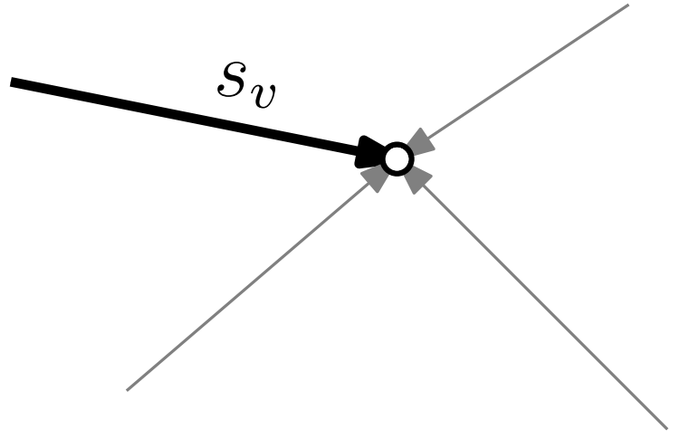
$a_{i,j} \leq a_{i,j'}$ für $j > j'$

Edmonds Algorithmus – Billigeste Kante

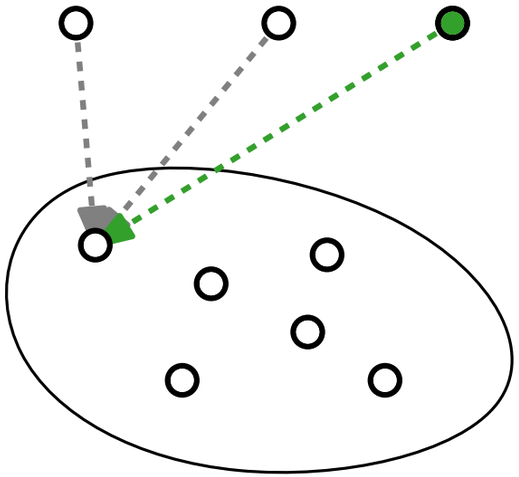
billigsteKante(Knoten v, Int c)

if v ist ein Item **then**

  return (x_v, v)



SC-Item ist mind. so gut wie alle anderen.

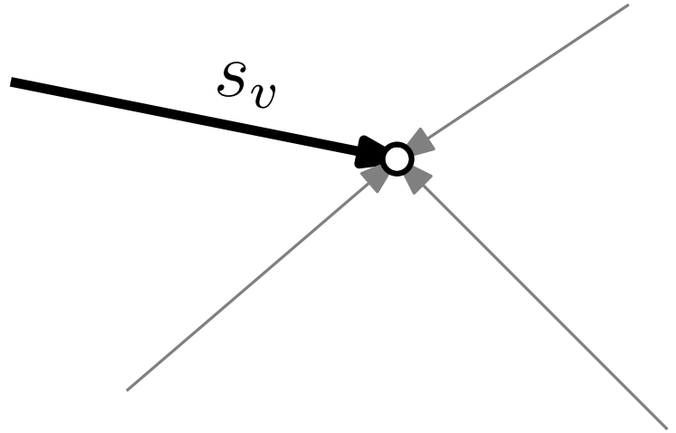


$a_{i,j} \leq a_{i,j'}$ für $j > j'$

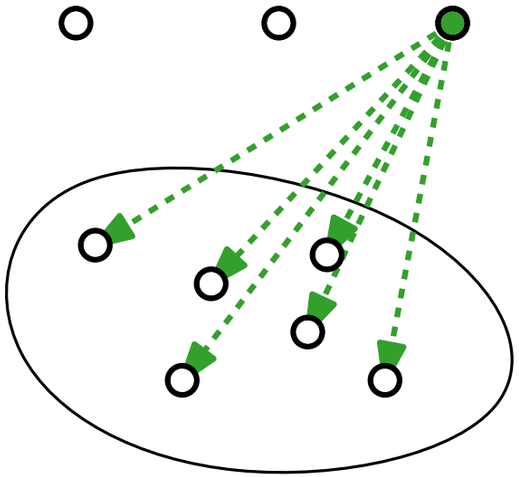
Edmonds Algorithmus – Billigeste Kante

billigsteKante(Knoten v, Int c)

if v ist ein Item **then**
  return (x_v, v)



SC-Item ist mind. so gut wie alle anderen.



$a_{i,j} \leq a_{i,j'}$ für $j > j'$

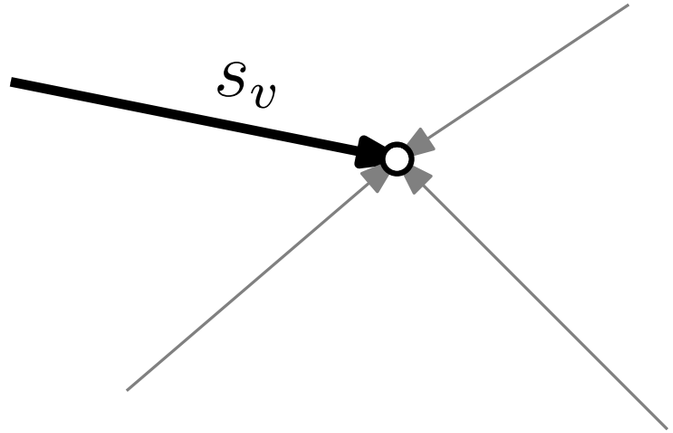
Edmonds Algorithmus – Billigeste Kante

billigsteKante(Knoten v, Int c)

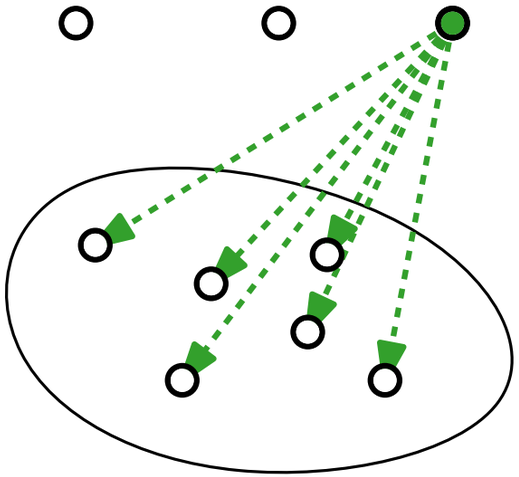
if v ist ein Item **then**
  return (x_v, v)

$u' \leftarrow \max\{u \notin v\}$

return $(u', argmin_u \text{ in } v\{a_{u,u'} - u.c_0\})$



SC-Item ist mind. so gut wie alle anderen.



$a_{i,j} \leq a_{i,j'}$ für $j > j'$

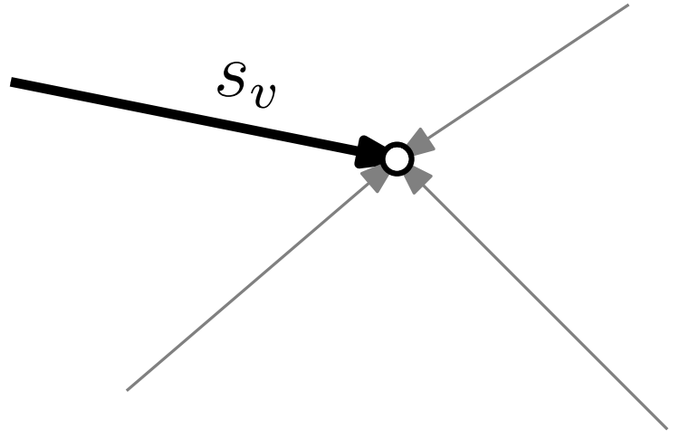
Edmonds Algorithmus – Billigeste Kante

billigsteKante(Knoten v, Int c)

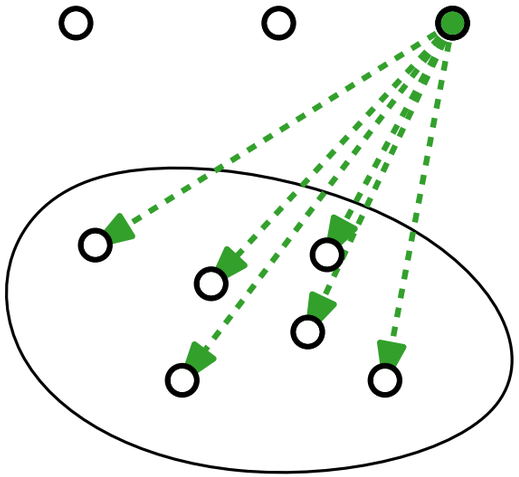
if v ist ein Item **then**
  return (x_v, v)

$u' \leftarrow \max\{u \notin v\}$ } $O(n \log n)$

return $(u', argmin_{u \text{ in } v}\{a_{u,u'} - u.c_0\})$



SC-Item ist mind. so gut wie alle anderen.



$a_{i,j} \leq a_{i,j'}$ für $j > j'$

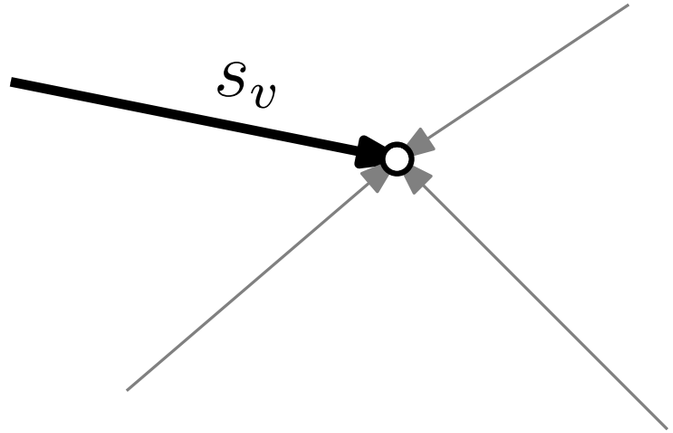
Edmonds Algorithmus – Billigeste Kante

billigsteKante(Knoten v, Int c)

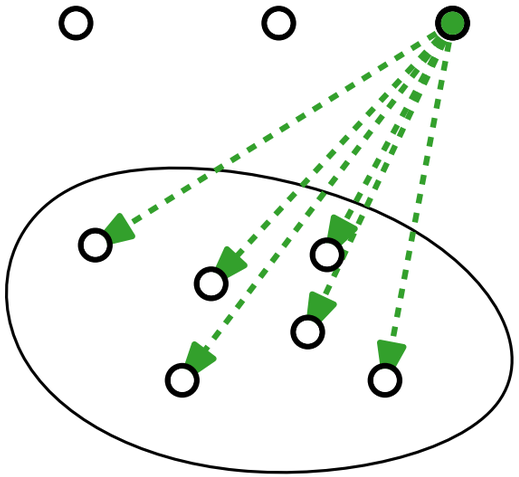
if v ist ein Item **then**
  return (x_v, v)

$u' \leftarrow \max\{u \notin v\}$ } $O(n \log n)$

return $(u', argmin_{u \text{ in } v}\{a_{u,u'} - u.c_0\})$ } $O(n)$



SC-Item ist mind. so gut wie alle anderen.



$a_{i,j} \leq a_{i,j'}$ für $j > j'$

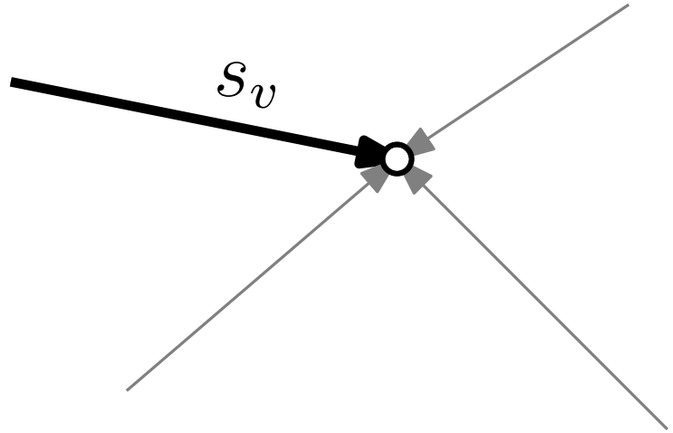
Edmonds Algorithmus – Billigeste Kante

billigsteKante(Knoten v, Int c)

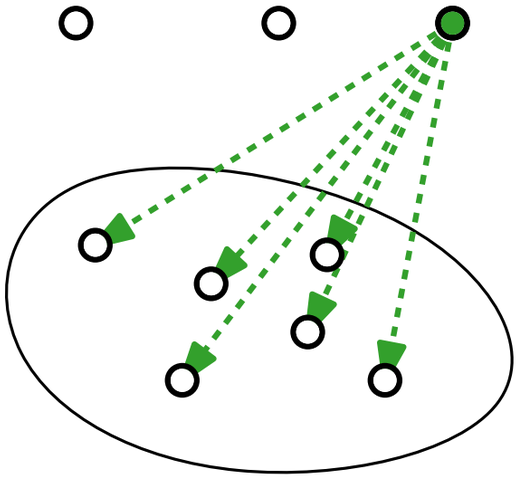
if v ist ein Item **then**
  return (x_v, v)

$u' \leftarrow \max\{u \notin v\}$ } $O(n \log n)$

return $(u', argmin_{u \text{ in } v}\{a_{u,u'} - u.c_0\})$ } $O(n)$



SC-Item ist mind. so gut wie alle anderen.

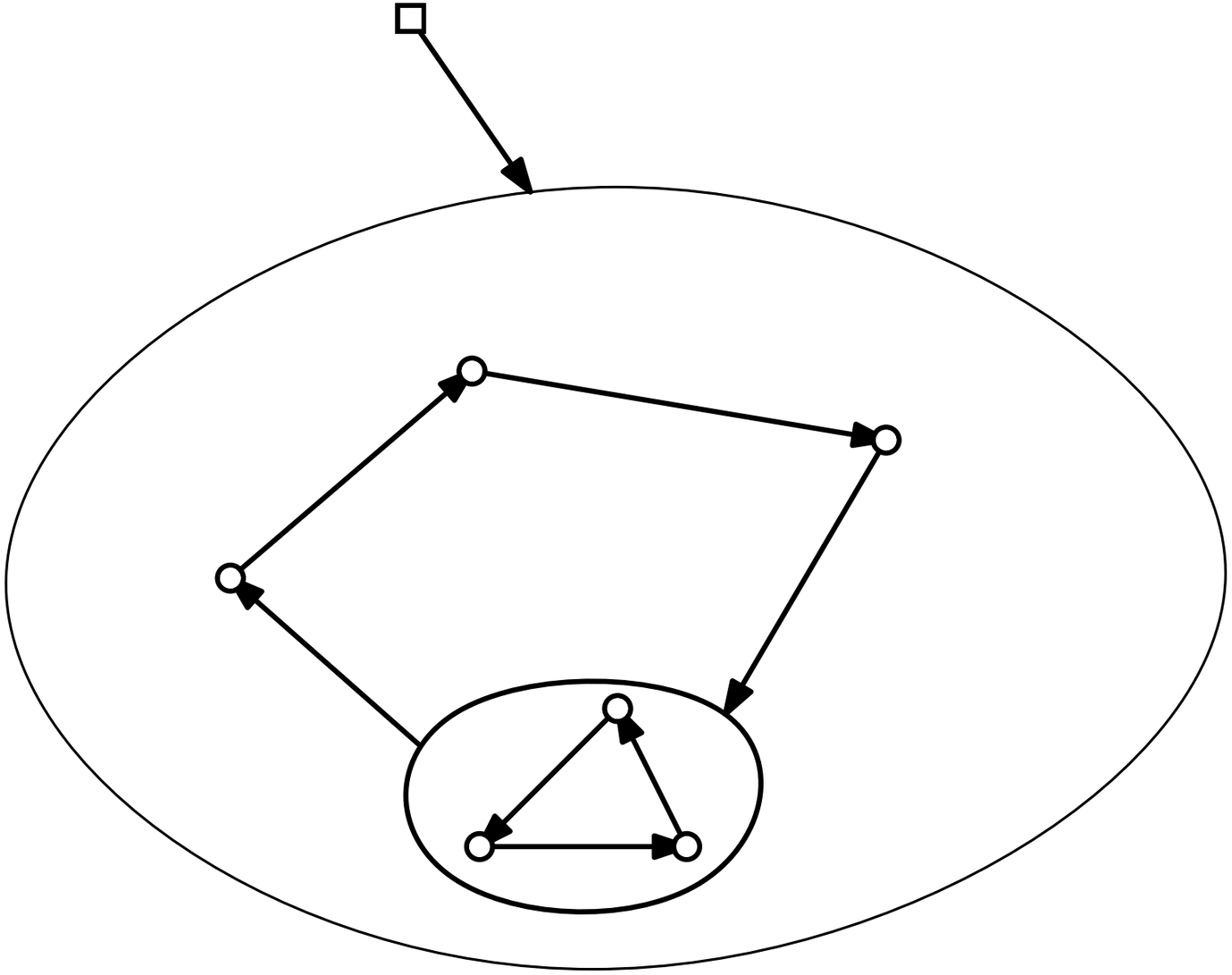


$a_{i,j} \leq a_{i,j'}$ für $j > j'$

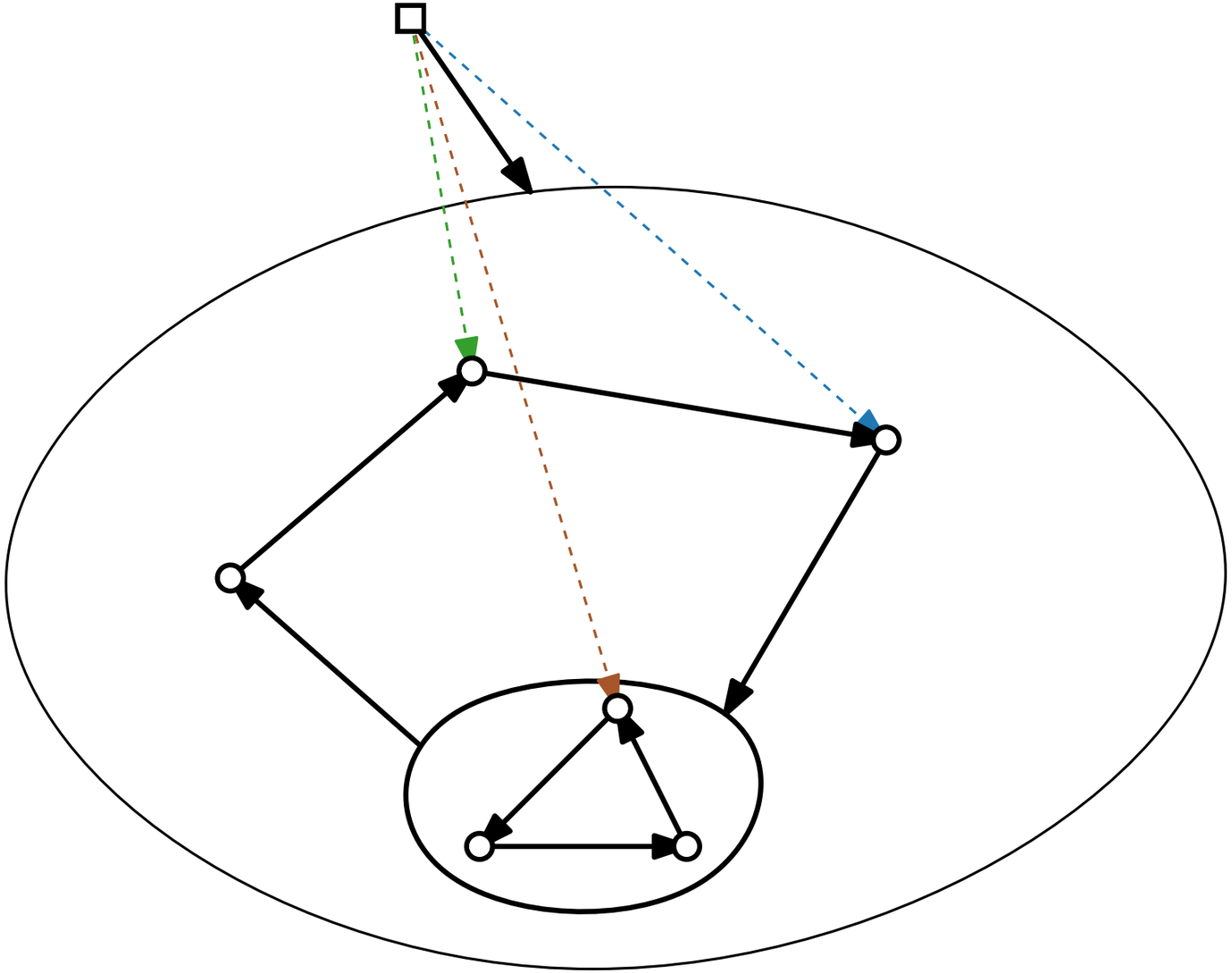
Laufzeit $O(n \log n)$

$\Rightarrow$ Test $u \in v$ in $O(\log n)$

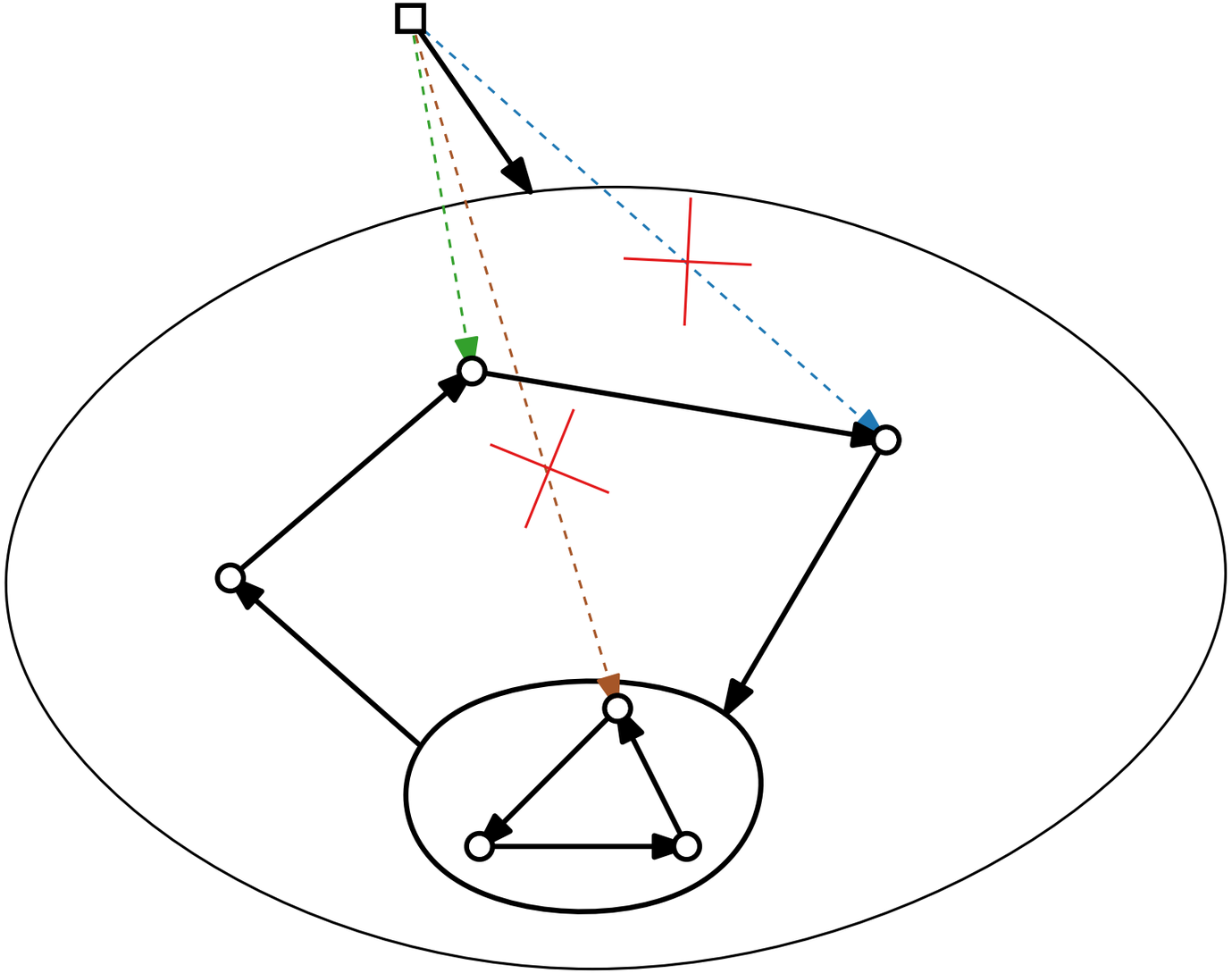
Edmonds Algorithmus – Expandieren



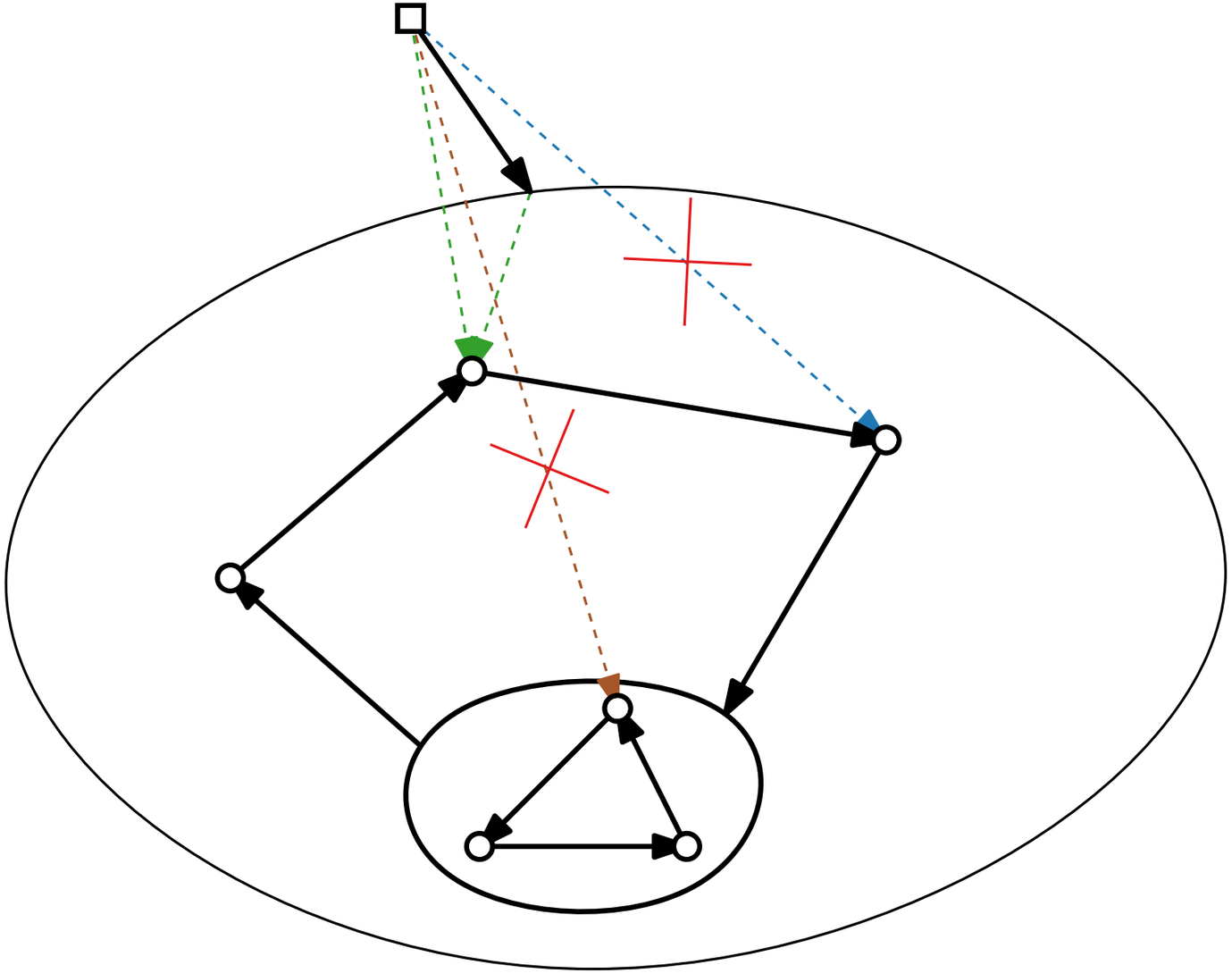
Edmonds Algorithmus – Expandieren



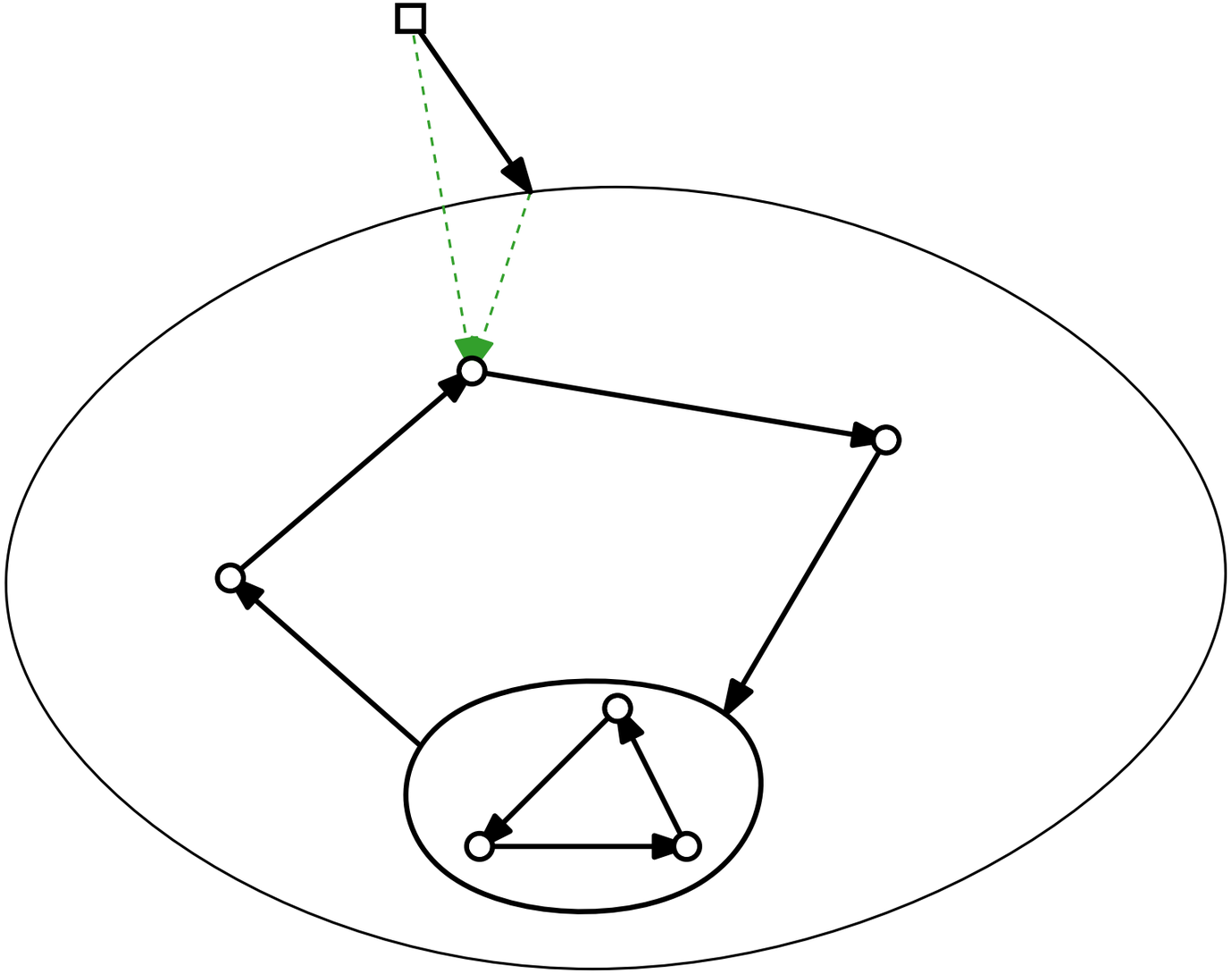
Edmonds Algorithmus – Expandieren



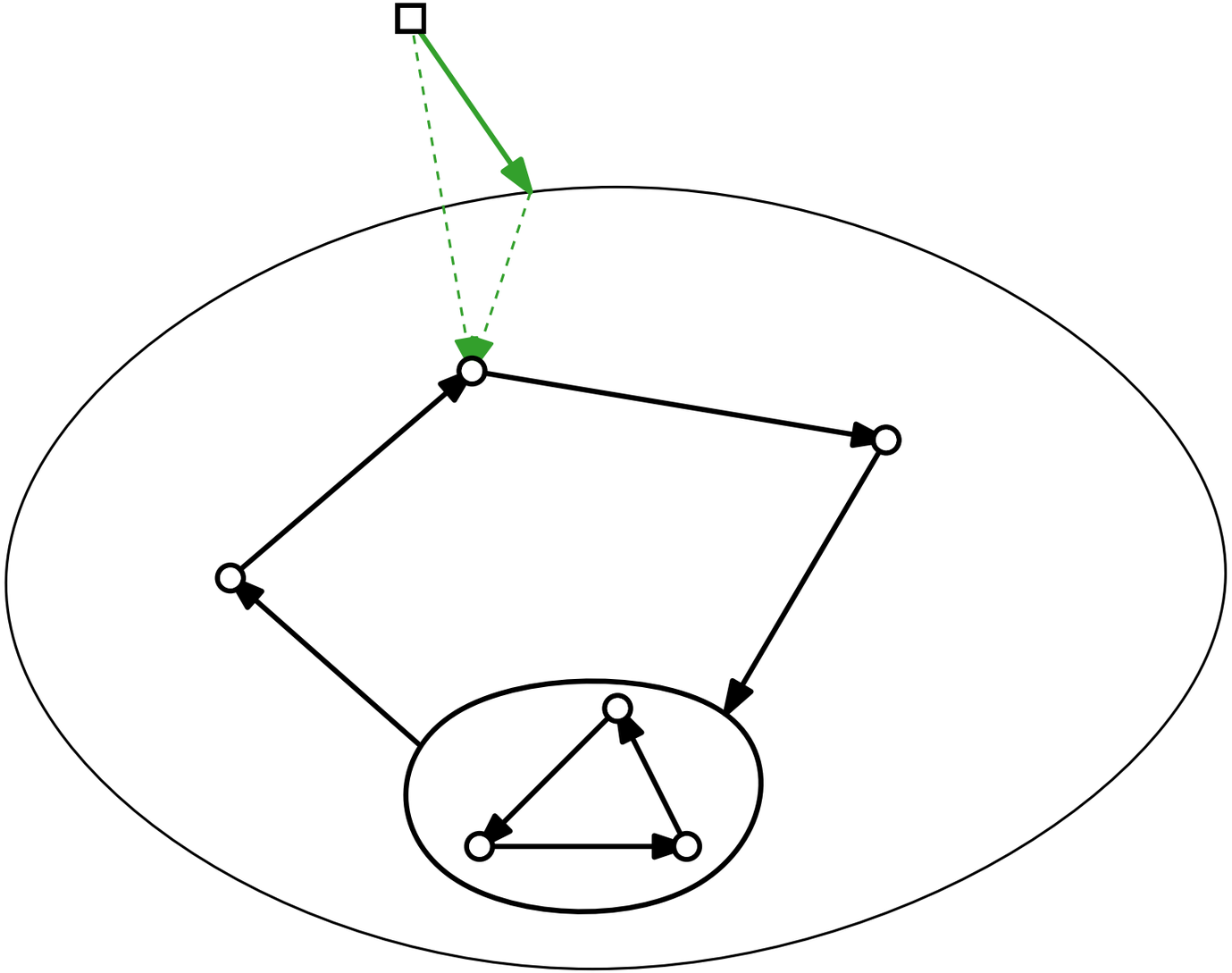
Edmonds Algorithmus – Expandieren



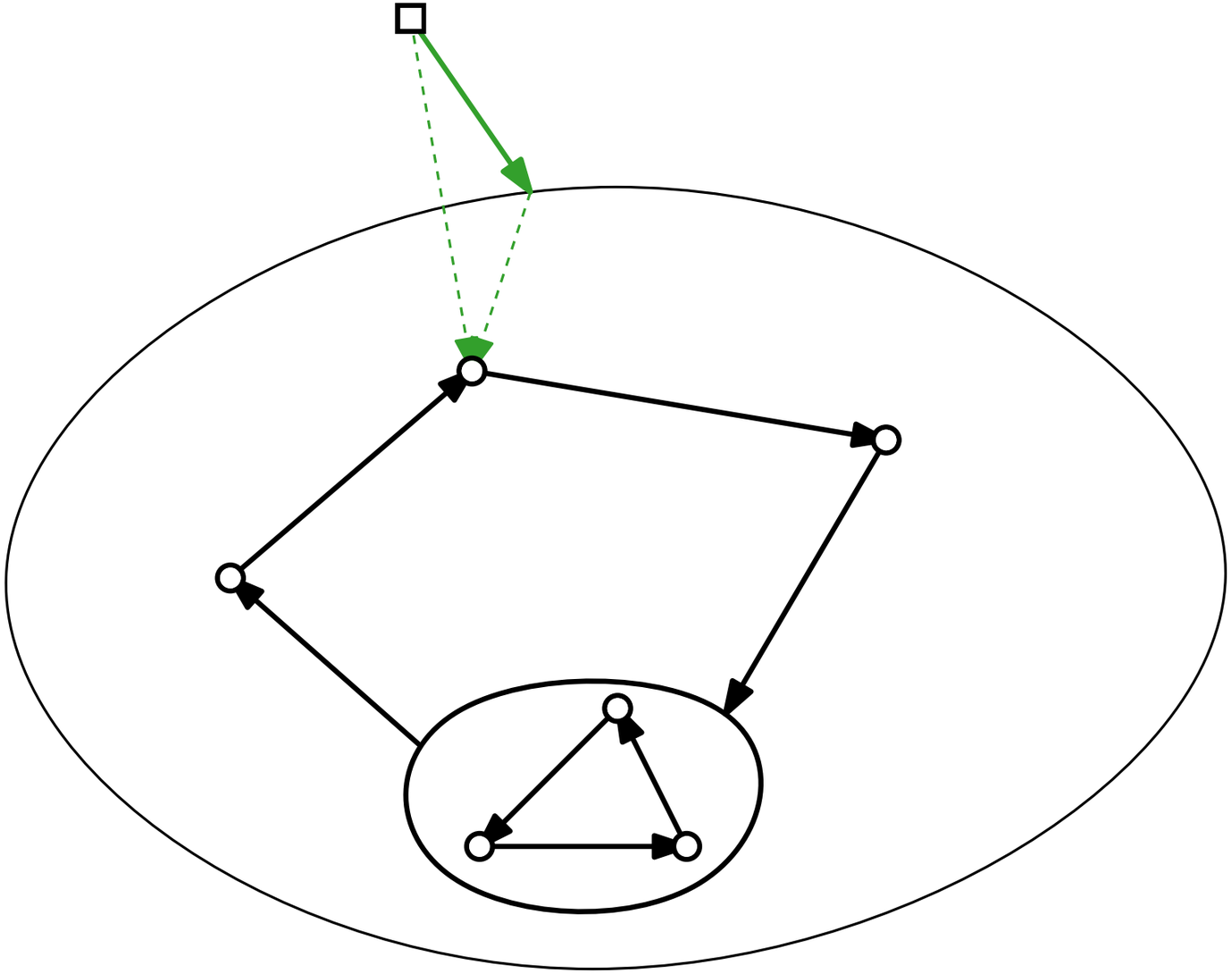
Edmonds Algorithmus – Expandieren



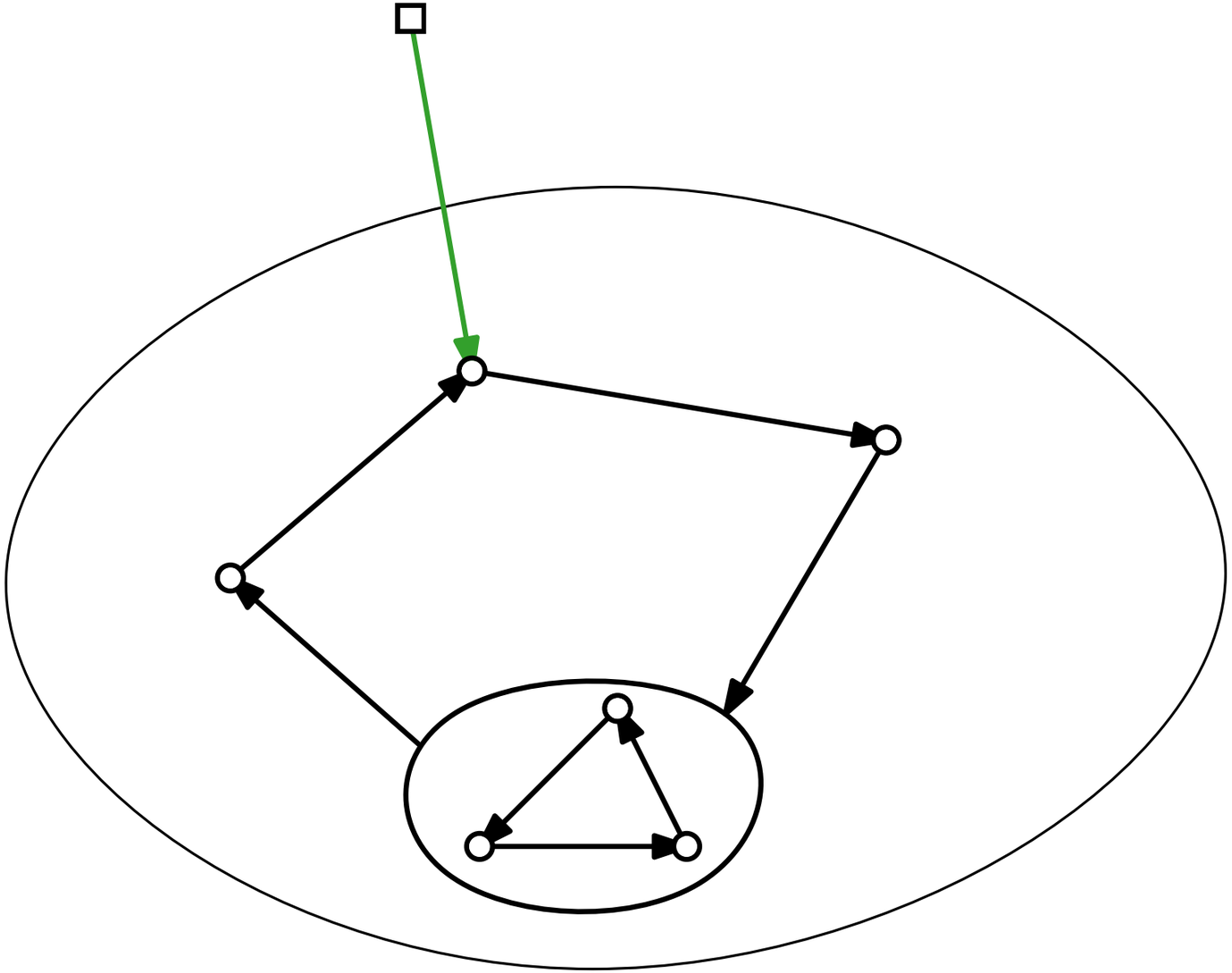
Edmonds Algorithmus – Expandieren



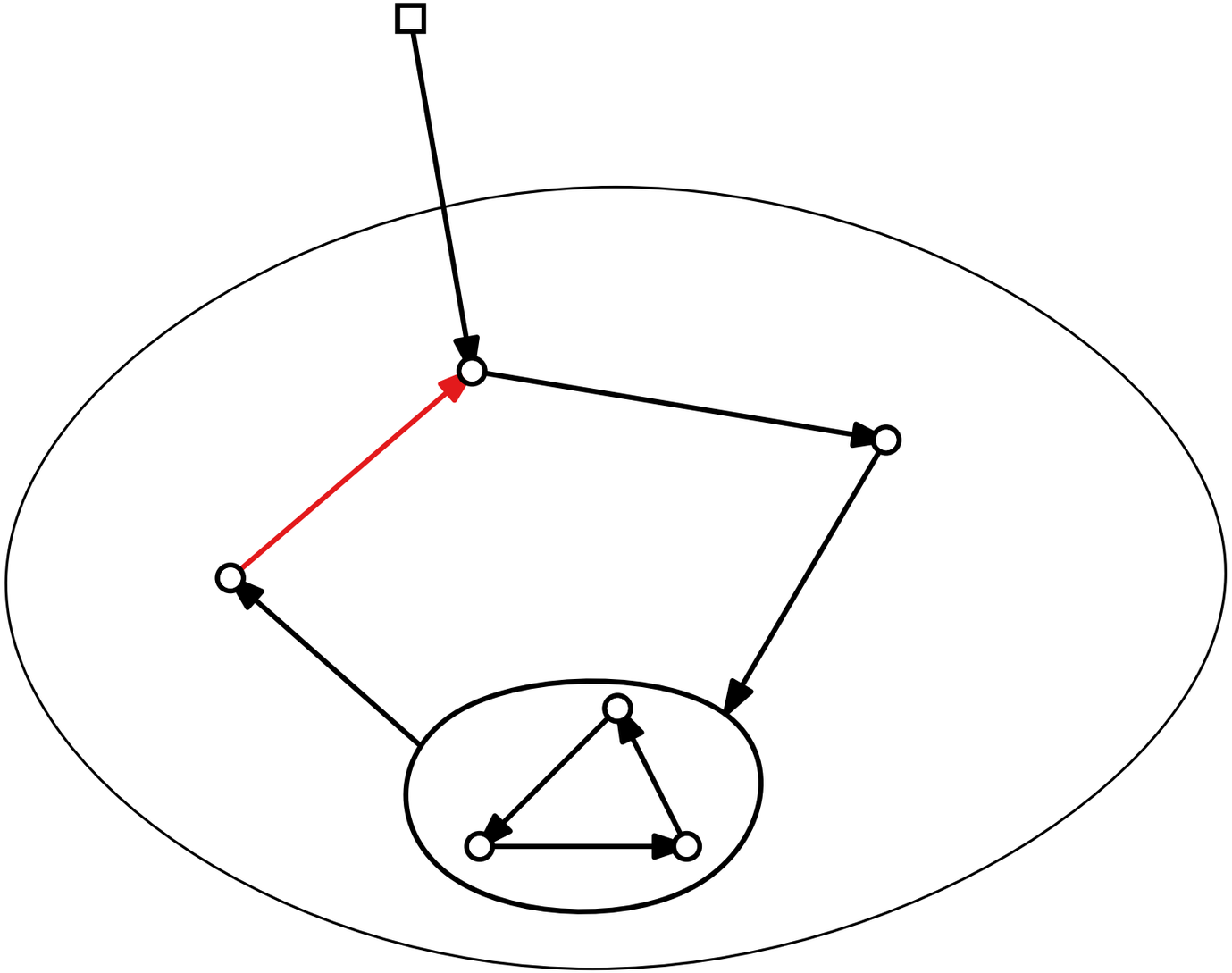
Edmonds Algorithmus – Expandieren



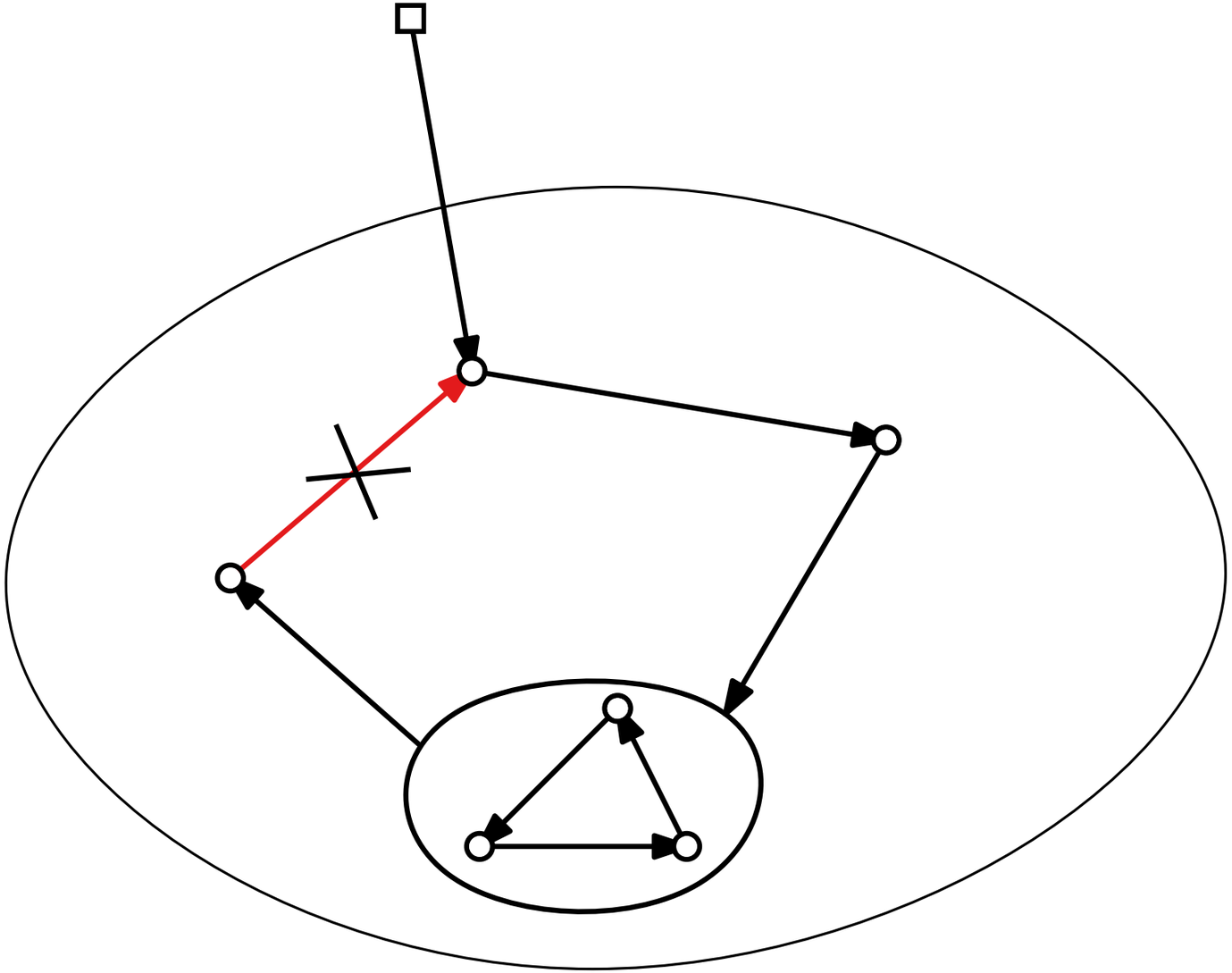
Edmonds Algorithmus – Expandieren



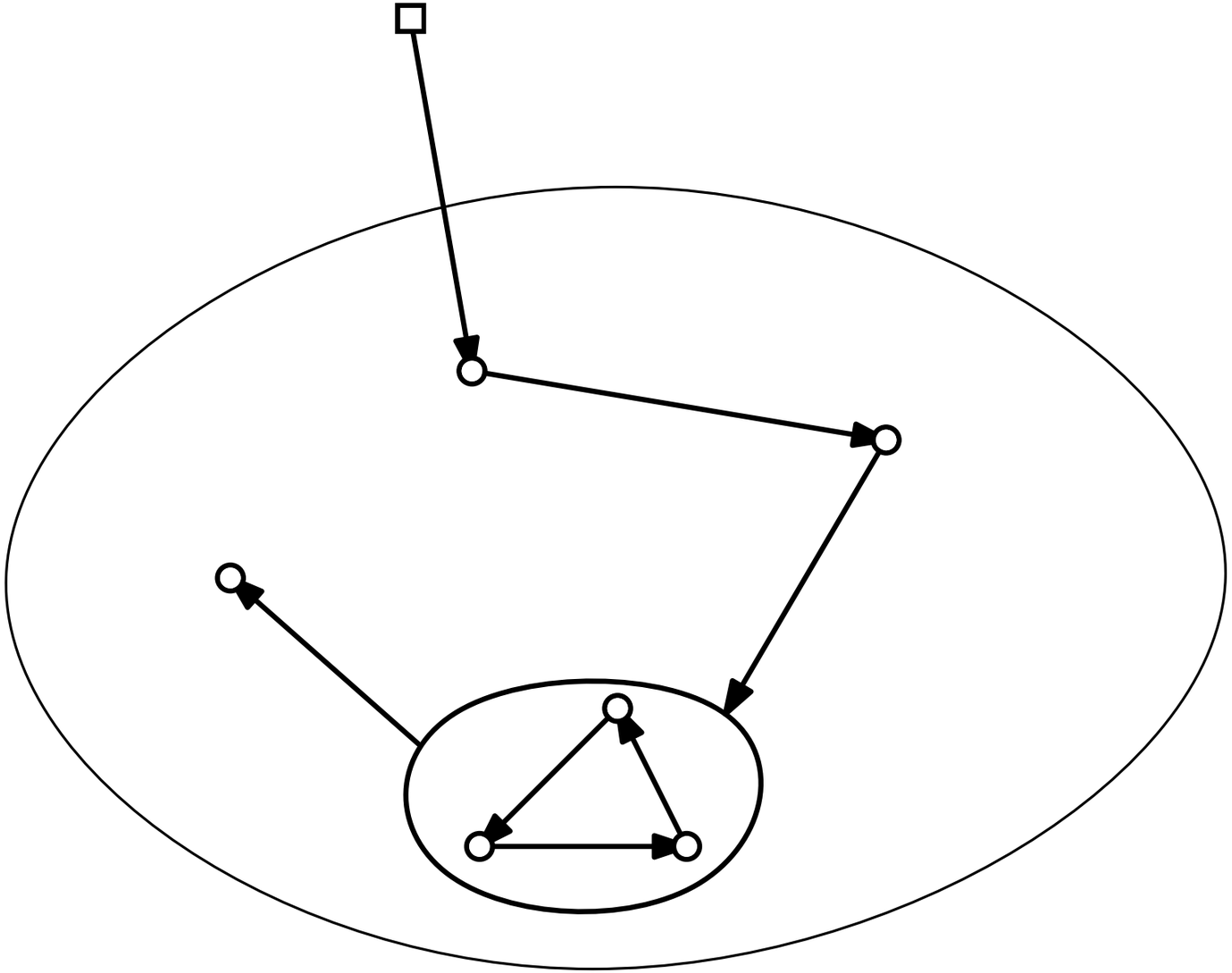
Edmonds Algorithmus – Expandieren



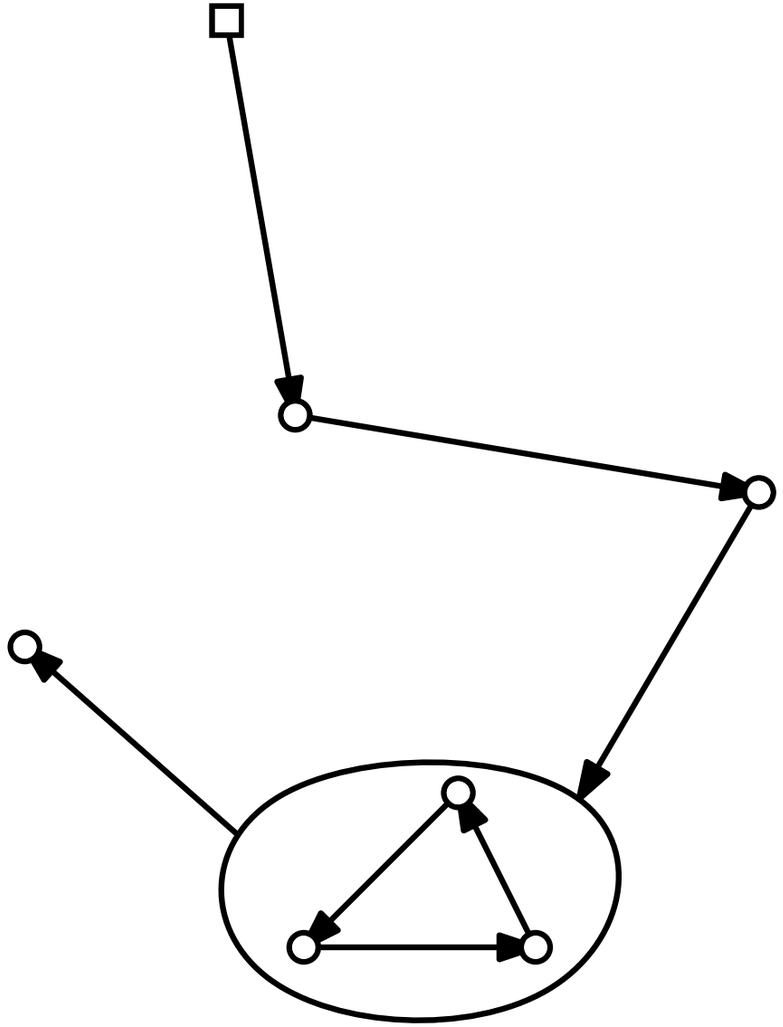
Edmonds Algorithmus – Expandieren



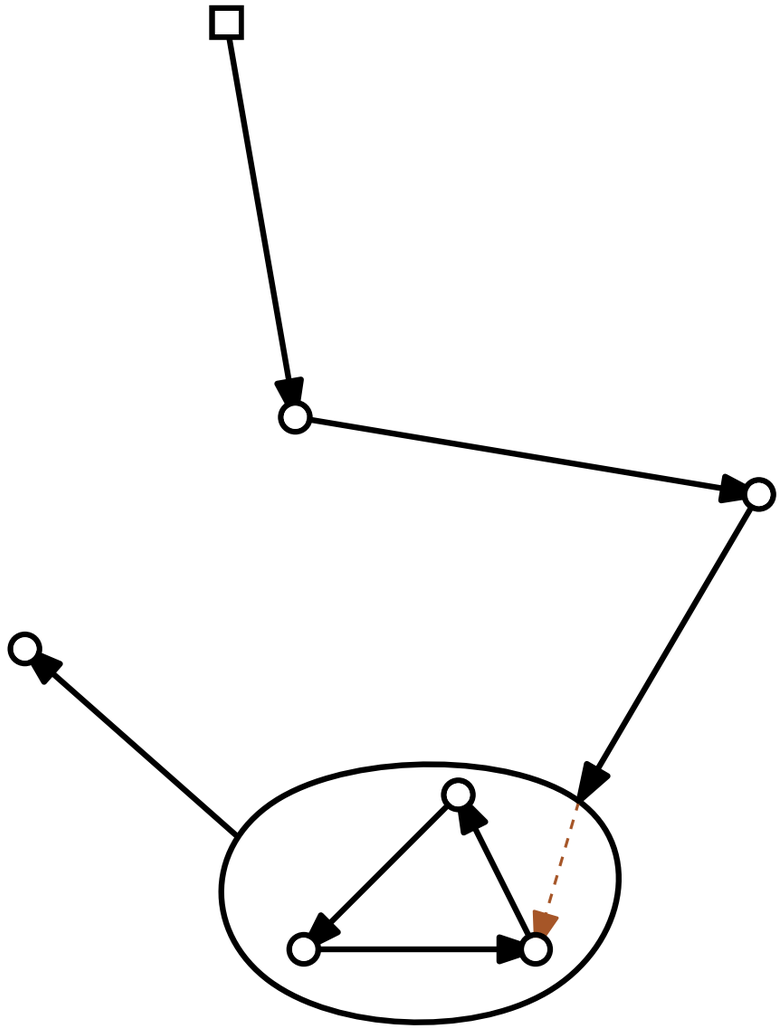
Edmonds Algorithmus – Expandieren



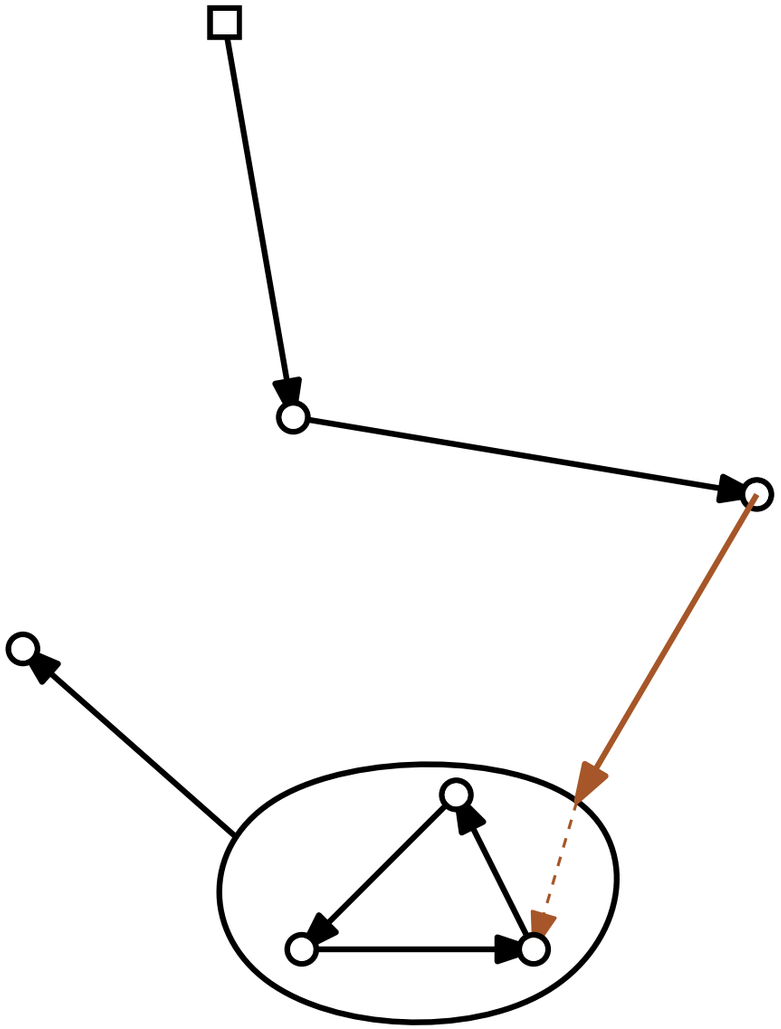
Edmonds Algorithmus – Expandieren



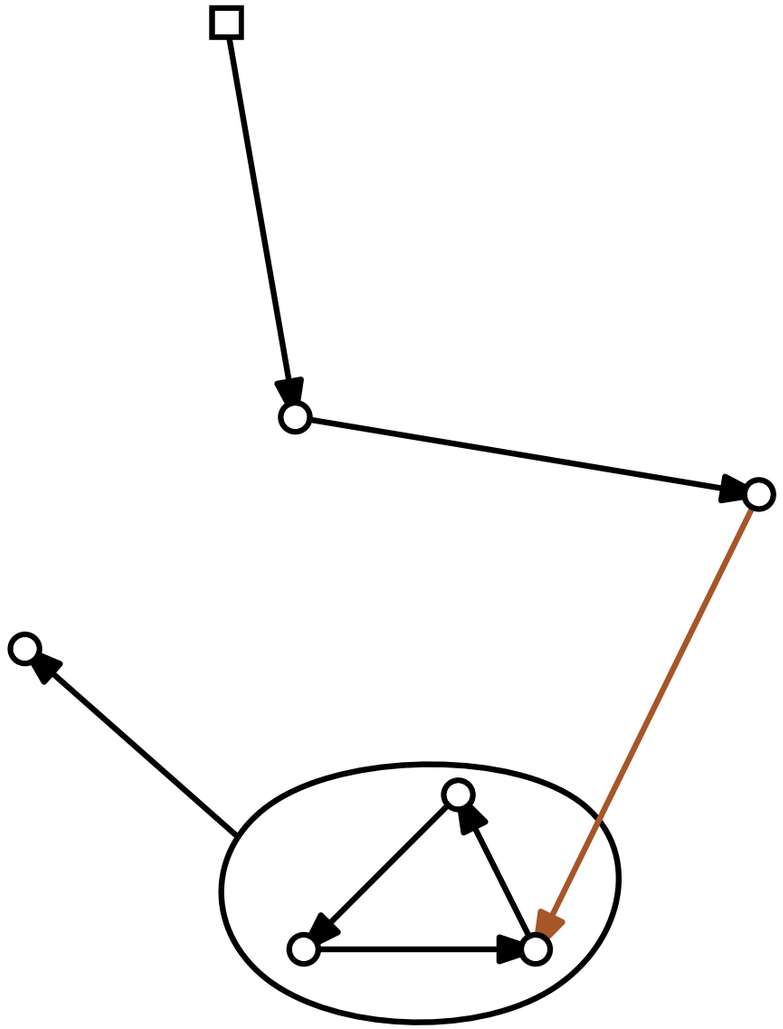
Edmonds Algorithmus – Expandieren



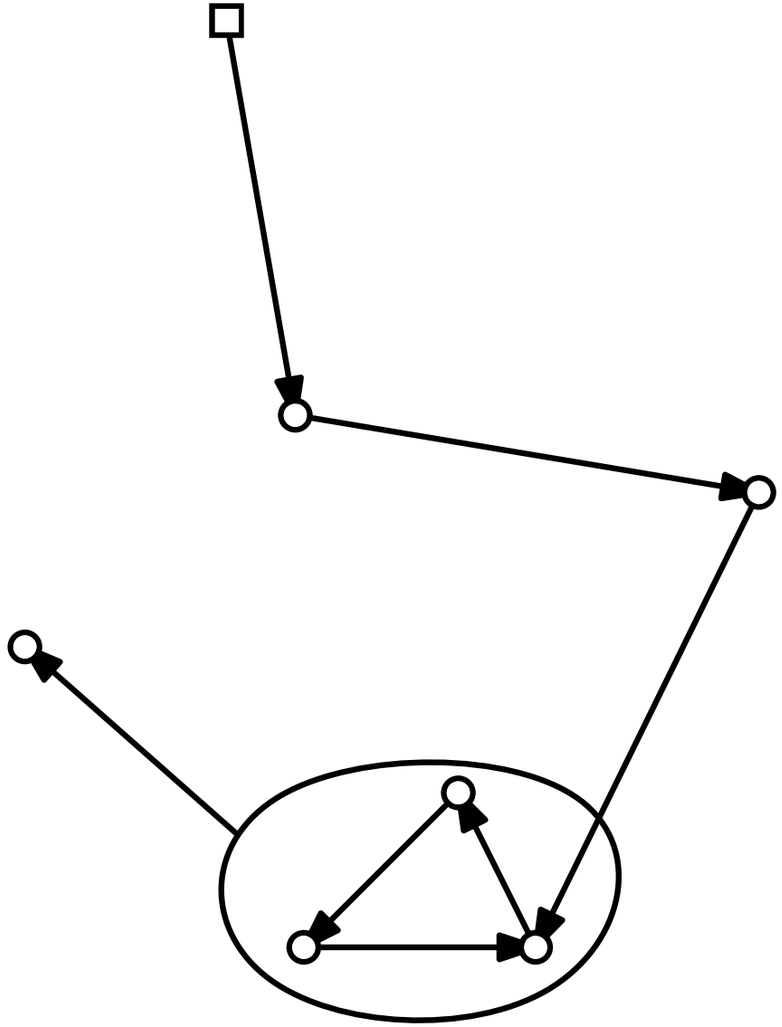
Edmonds Algorithmus – Expandieren



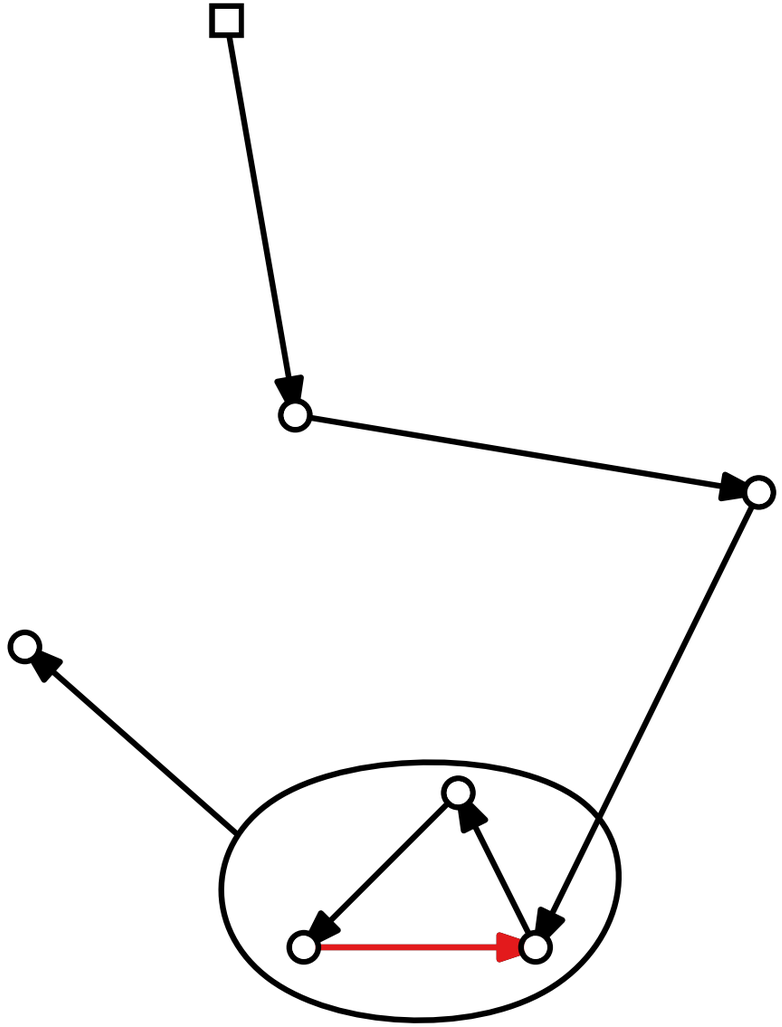
Edmonds Algorithmus – Expandieren



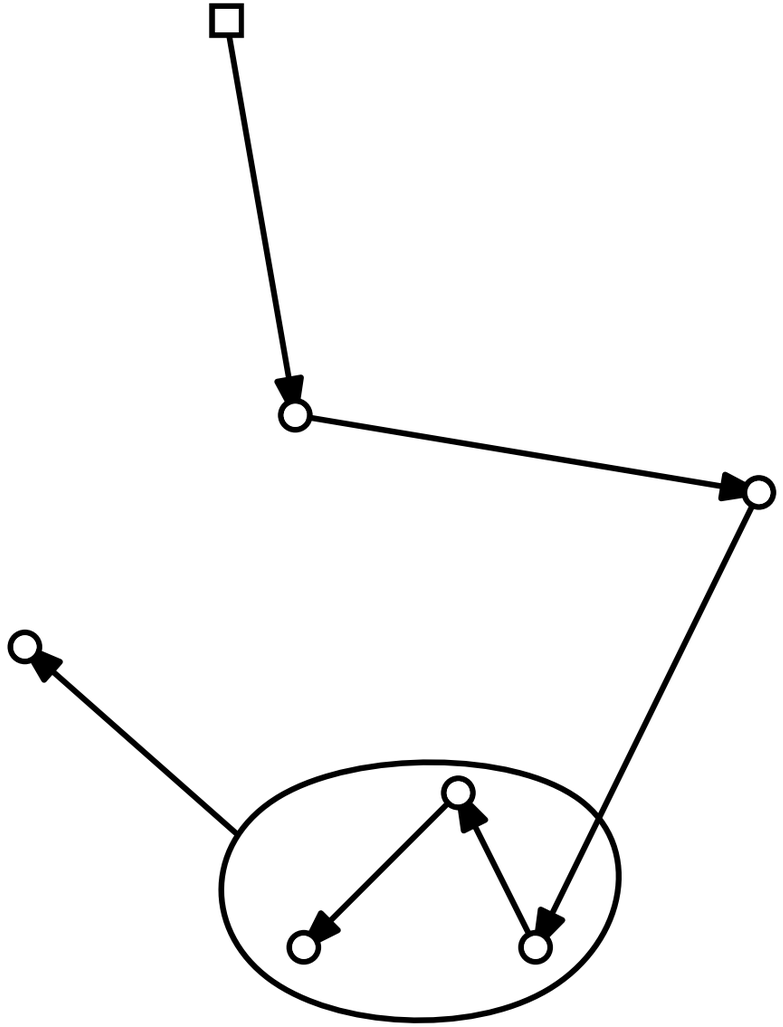
Edmonds Algorithmus – Expandieren



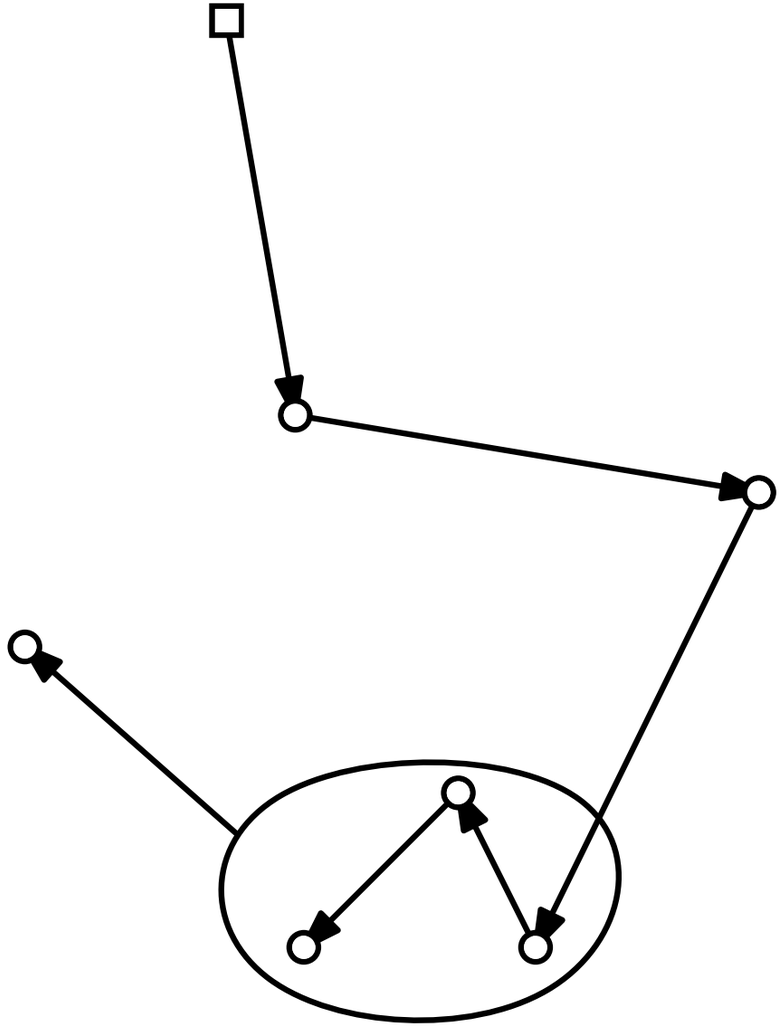
Edmonds Algorithmus – Expandieren



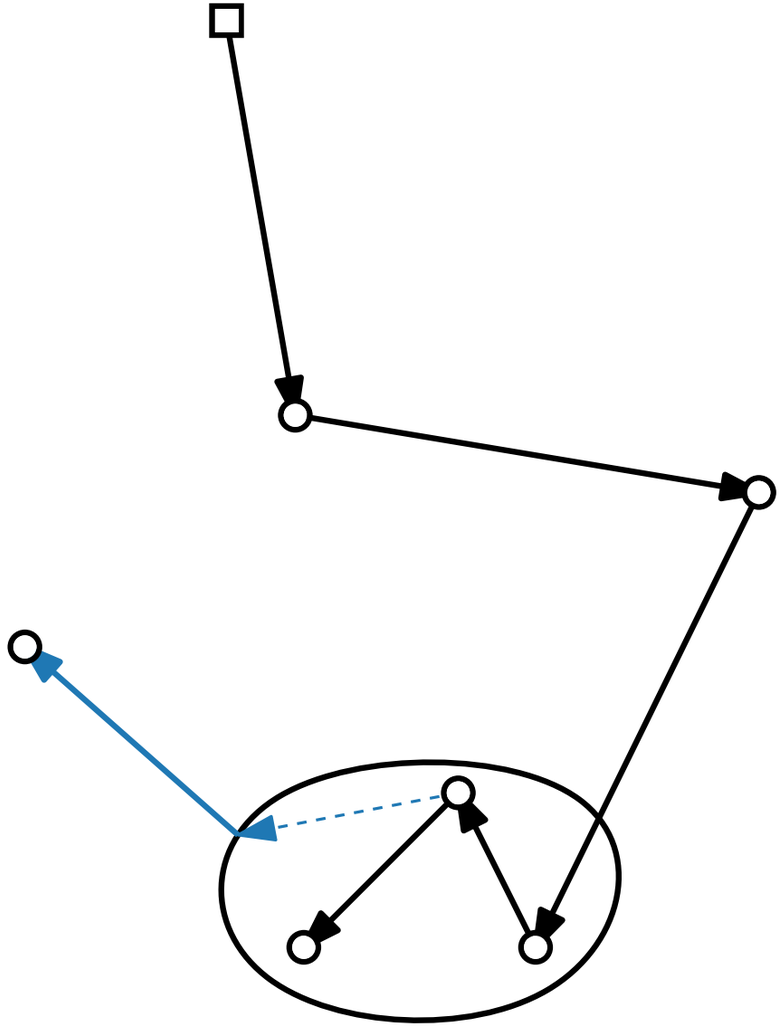
Edmonds Algorithmus – Expandieren



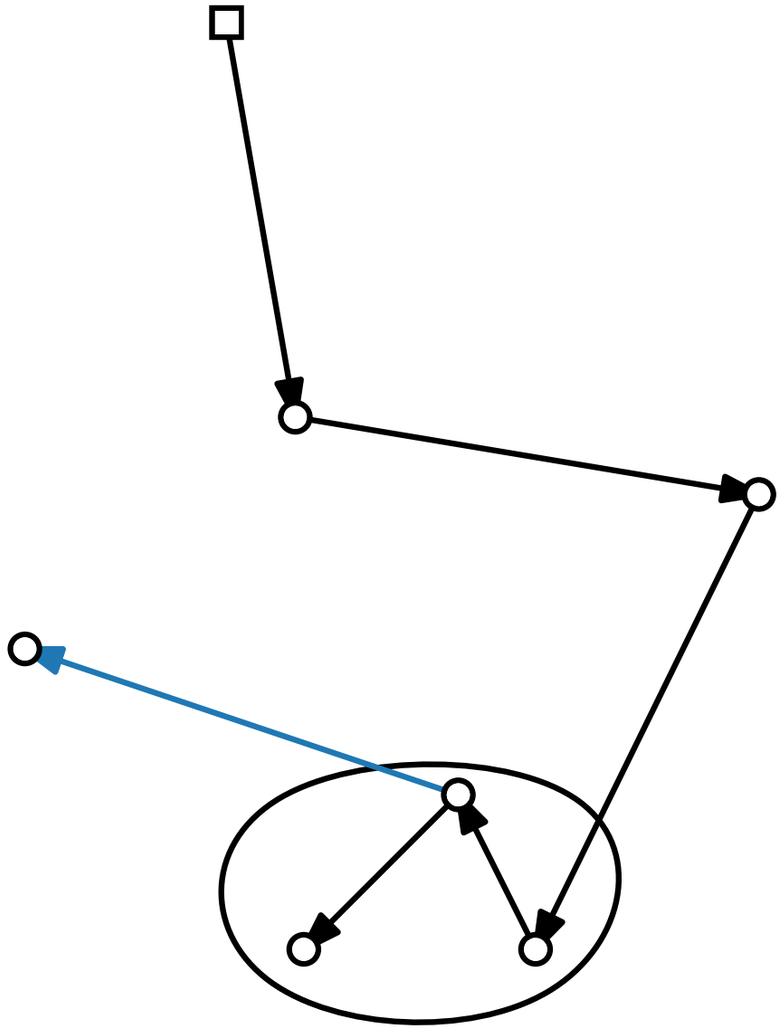
Edmonds Algorithmus – Expandieren



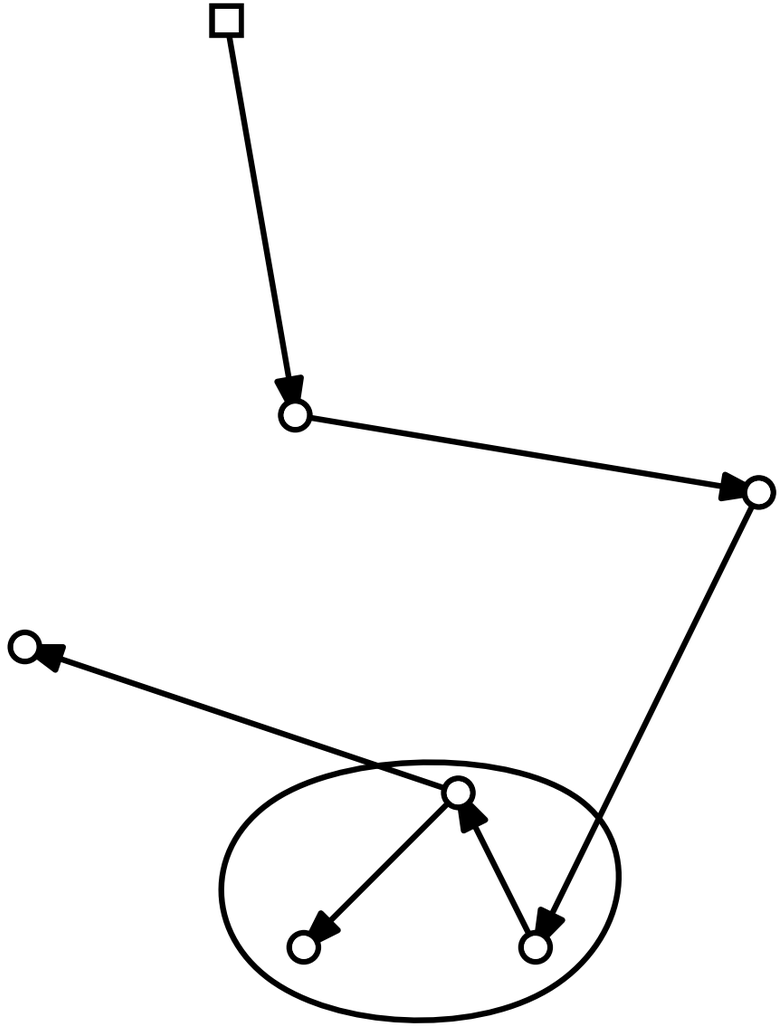
Edmonds Algorithmus – Expandieren



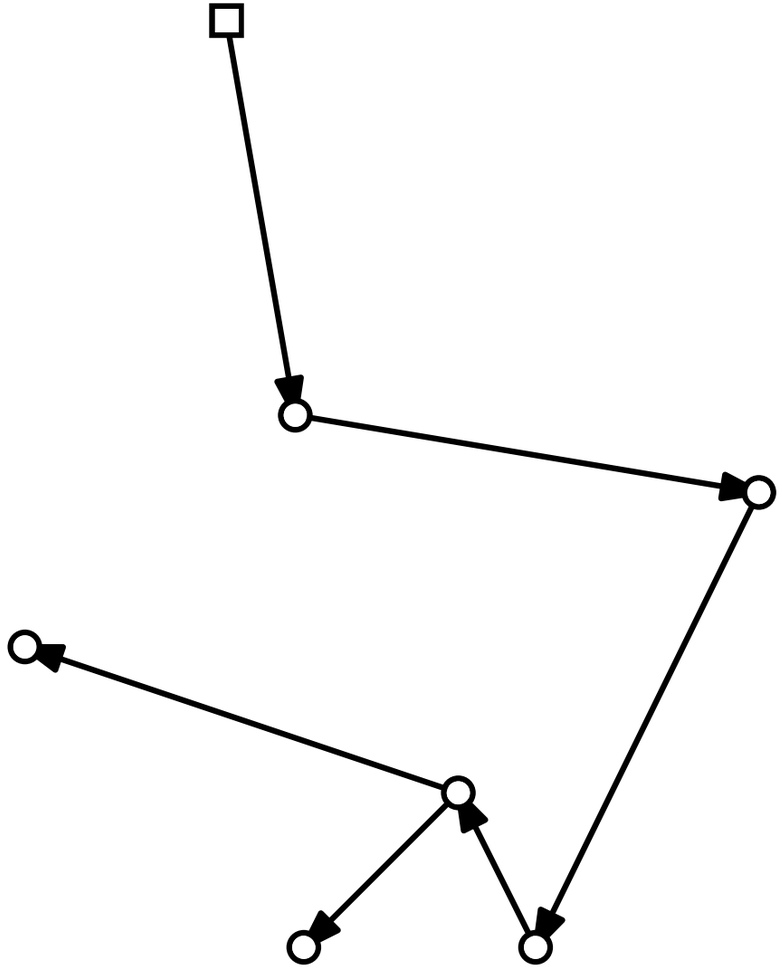
Edmonds Algorithmus – Expandieren



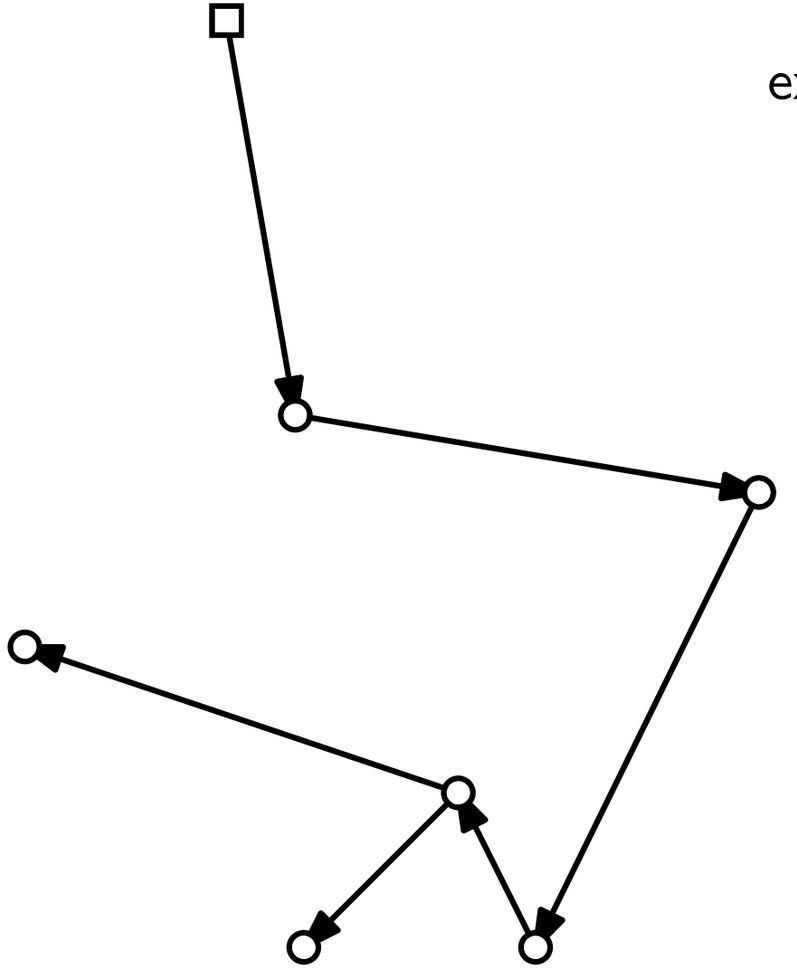
Edmonds Algorithmus – Expandieren



Edmonds Algorithmus – Expandieren

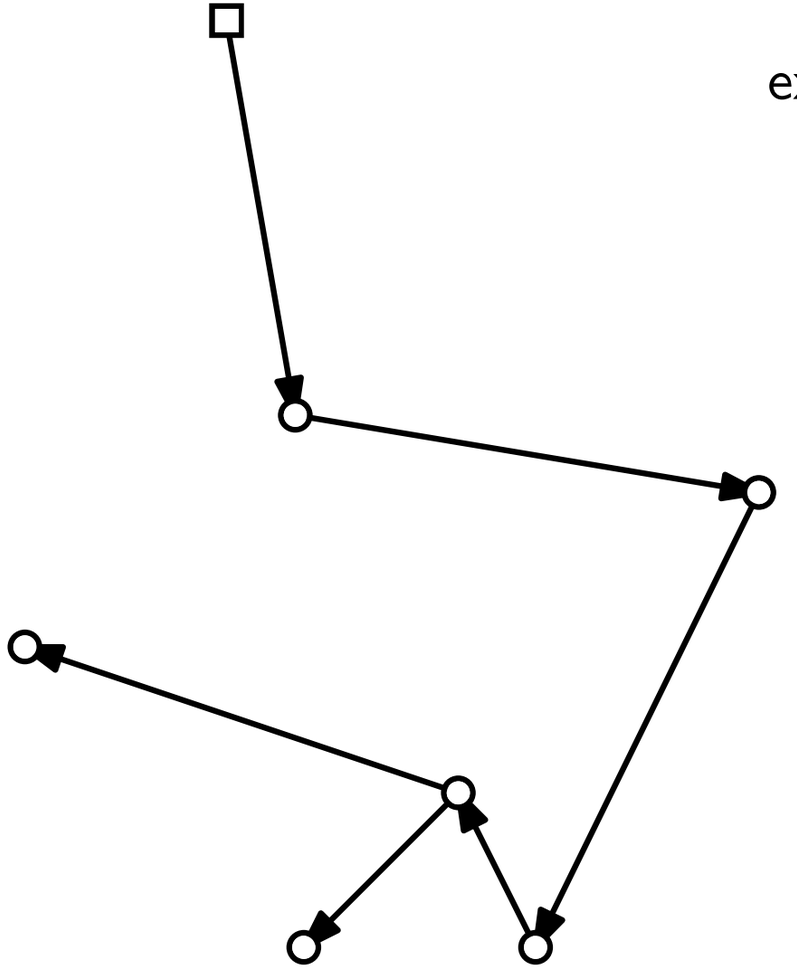


Edmonds Algorithmus – Expandieren



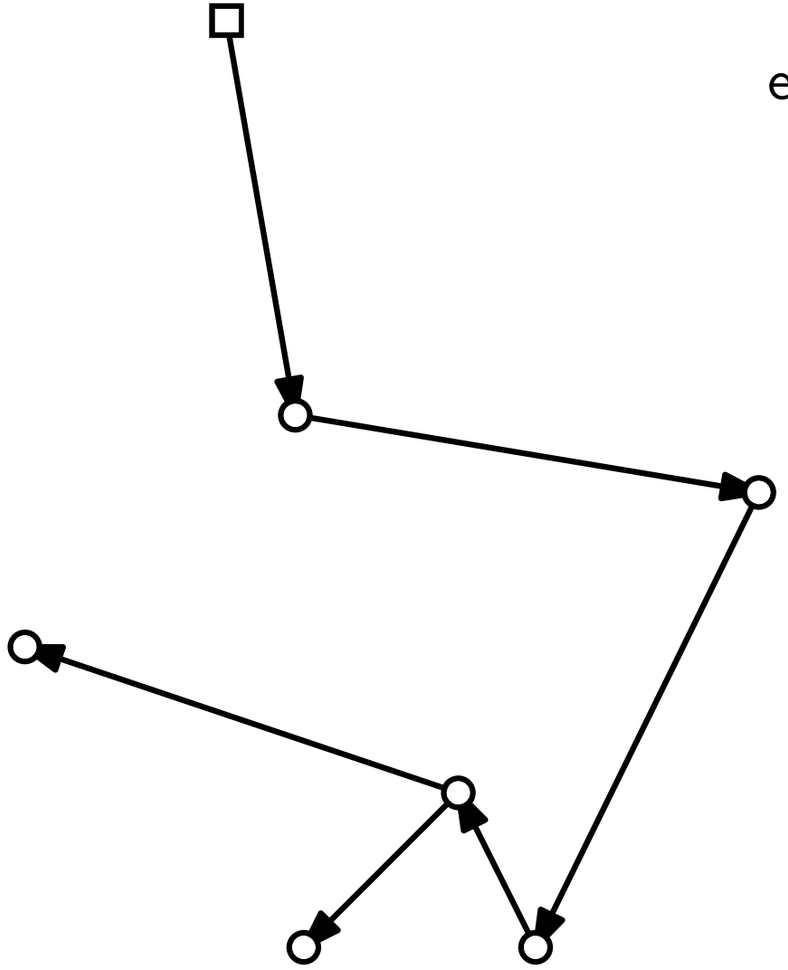
```
expandiere(Knoten v)
    if v kein Item then
        foreach u Subknoten von v do
            if v.inKante zeigt nach u then
                u.inKante ← v.inKante
            expandiere(u)
```

Edmonds Algorithmus – Expandieren



```
expandiere(Knoten v)
  if v kein Item then
    foreach u Subknoten von v do
      if v.inKante zeigt nach u then } O(log n)
        u.inKante ← v.inKante
      expandiere(u)
```

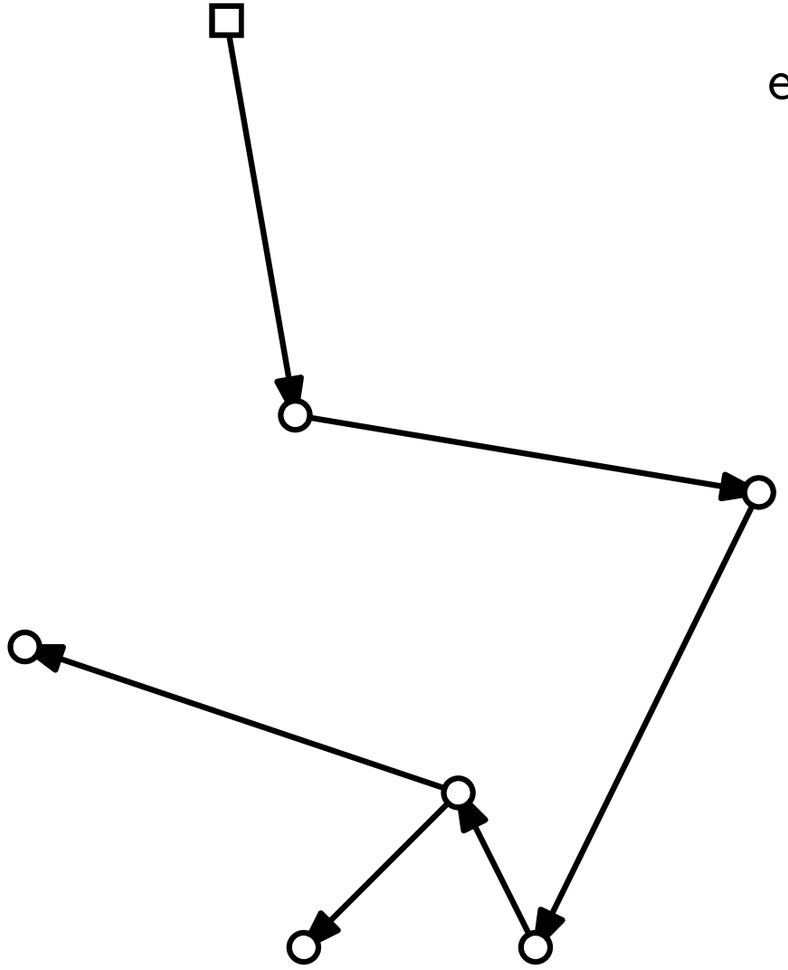
Edmonds Algorithmus – Expandieren



```
expandiere(Knoten v)
  if v kein Item then
    foreach u Subknoten von v do
      if v.inKante zeigt nach u then } O(log n)
        u.inKante ← v.inKante
      expandiere(u)
```

$O(n)$ Subknoten im ganzen Graphen.

Edmonds Algorithmus – Expandieren

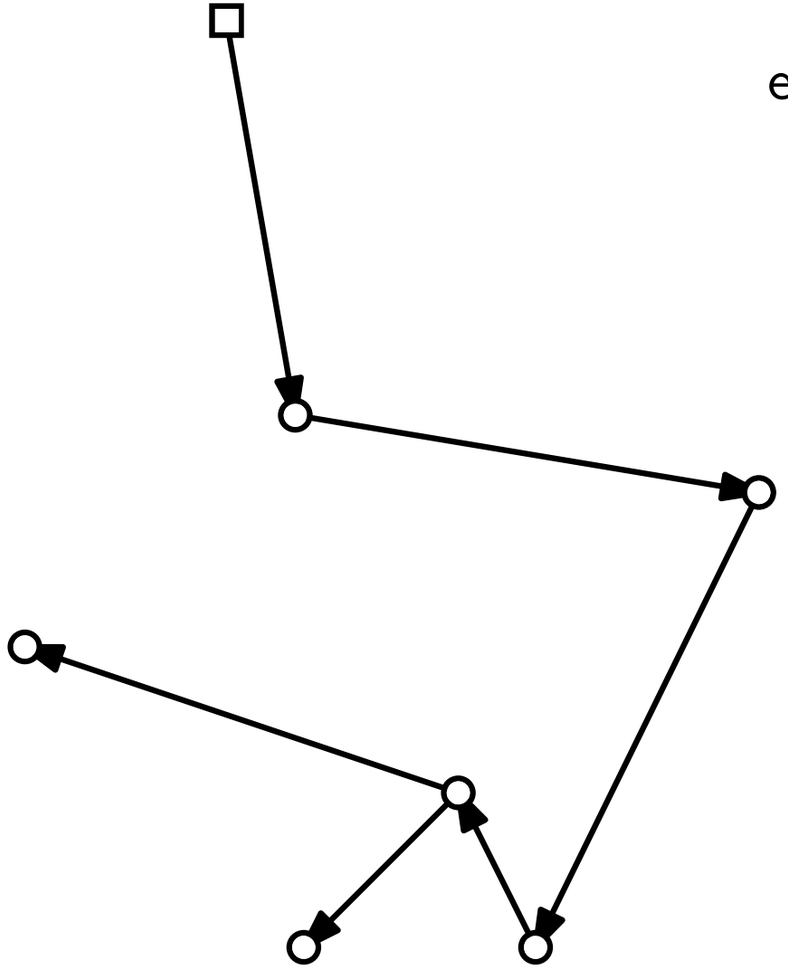


```
expandiere(Knoten v)
  if v kein Item then
    foreach u Subknoten von v do
      if v.inKante zeigt nach u then } O(log n)
        u.inKante ← v.inKante
      expandiere(u)
```

$O(n)$ Subknoten im ganzen Graphen.

Laufzeit $O(n \log n)$

Edmonds Algorithmus – Expandieren



```
expandiere(Knoten v)
  if v kein Item then
    foreach u Subknoten von v do
      if v.inKante zeigt nach u then } O(log n)
        u.inKante ← v.inKante
      expandiere(u)
```

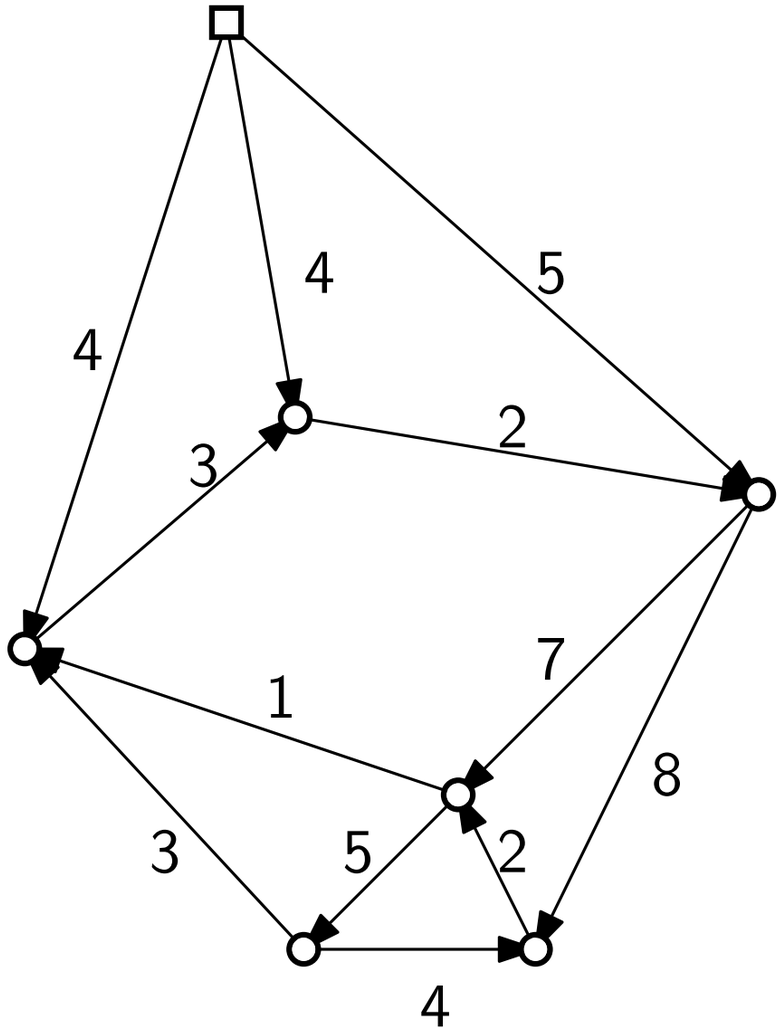
$O(n)$ Subknoten im ganzen Graphen.

Laufzeit $O(n \log n)$

$\Rightarrow$ For-Schleife iteriert in $O(1)$ weiter.

Edmonds Algorithmus – Implementierung

Edmonds Algorithmus – Implementierung

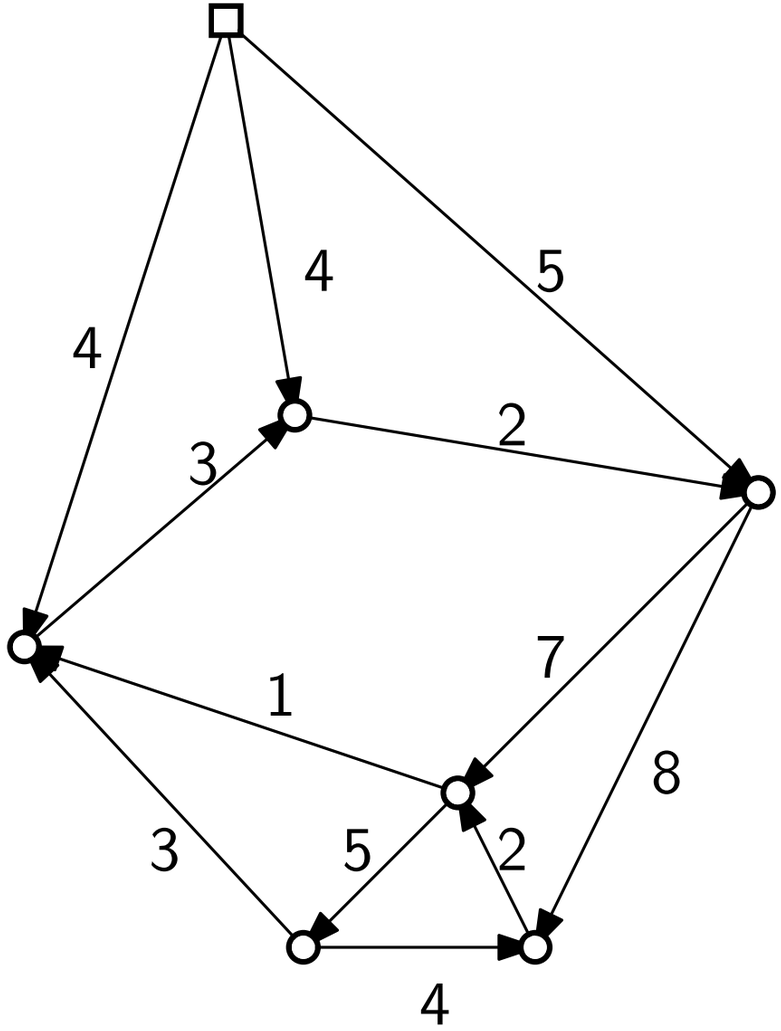


Edmonds Algorithmus – Implementierung

knoten: array

status: c_0:
status: c_0:
status: c_0:
status: c_0:
status: c_0:
status: c_0:
status: c_0:

ast: stack

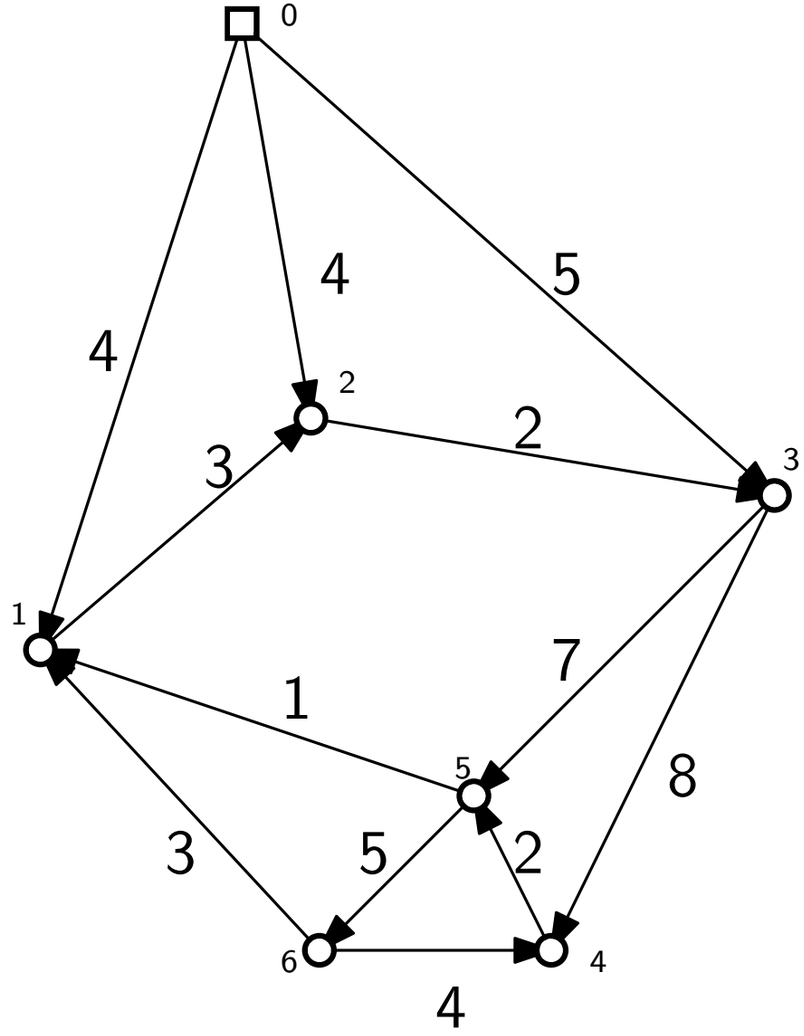


Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0

ast: stack

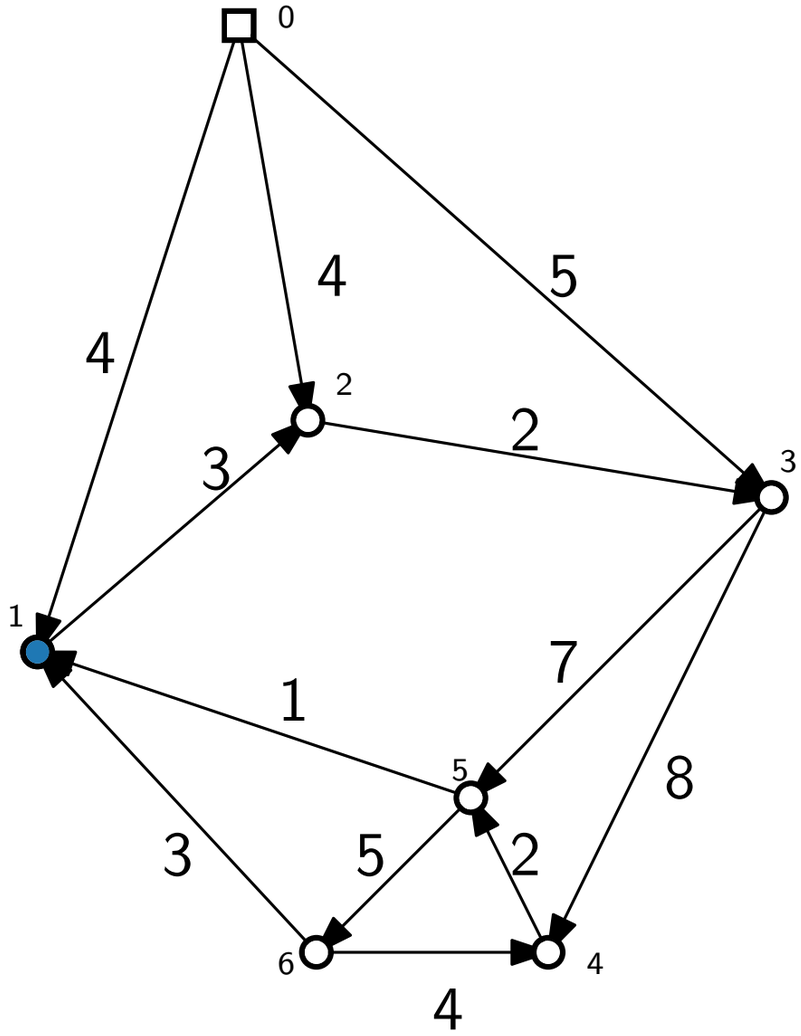


Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0

ast: stack

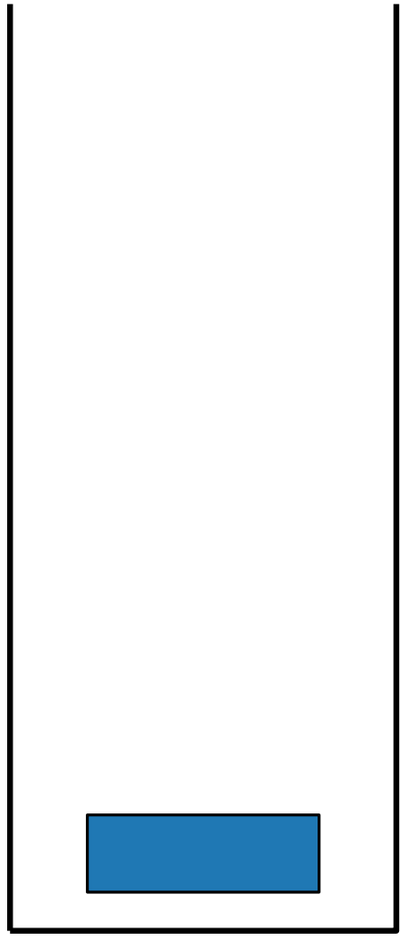


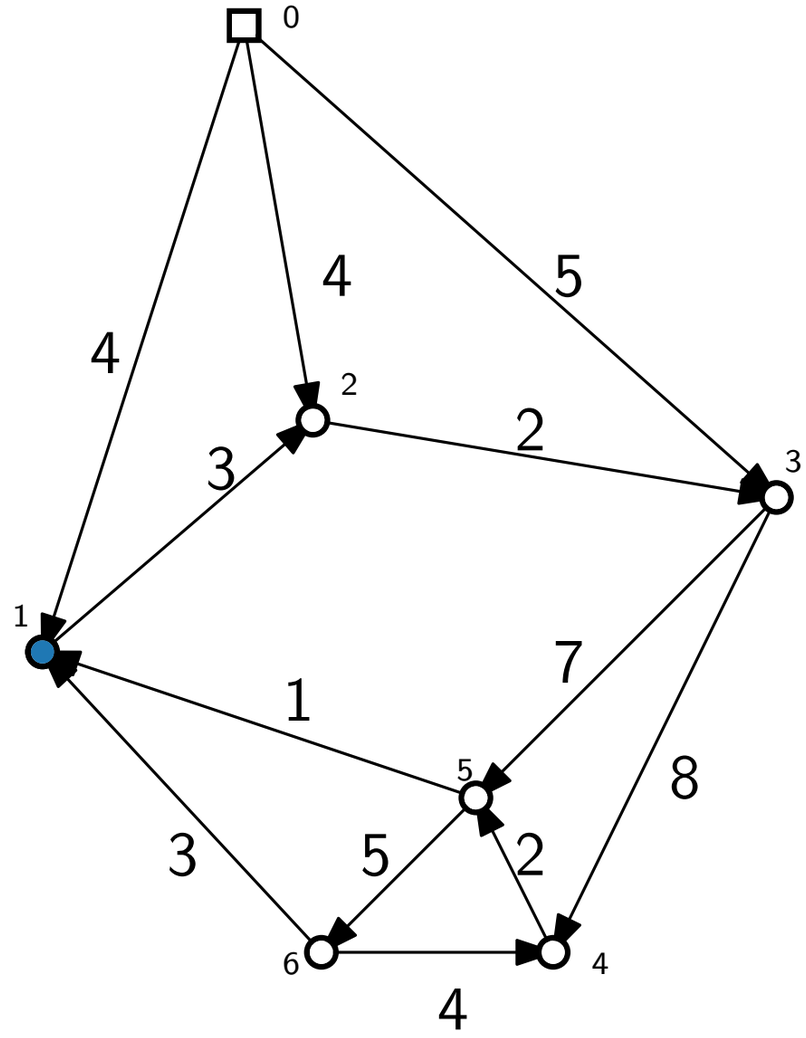
Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0

ast: stack



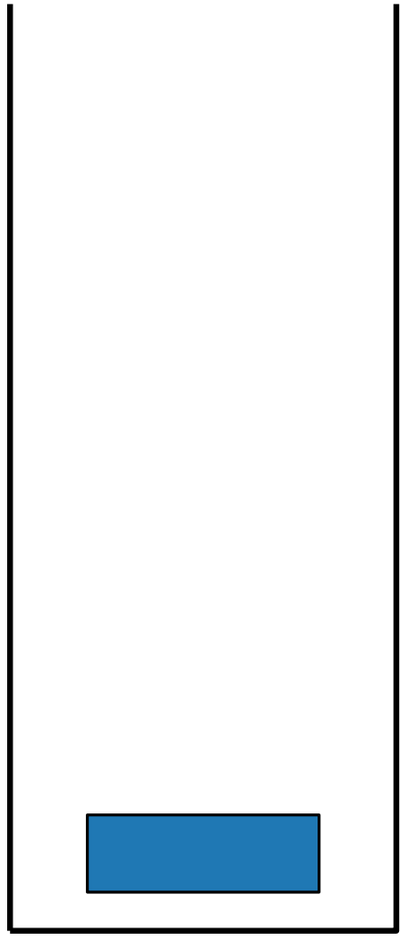


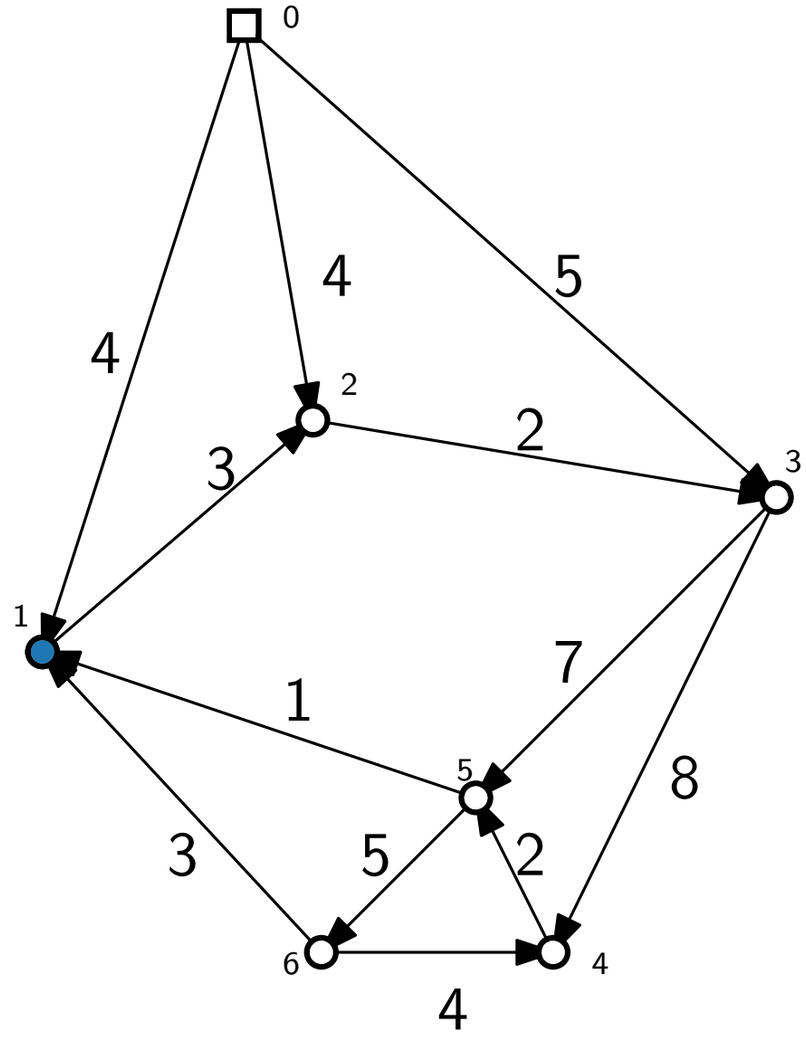
Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	besucht
c_0:	0
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0

ast: stack



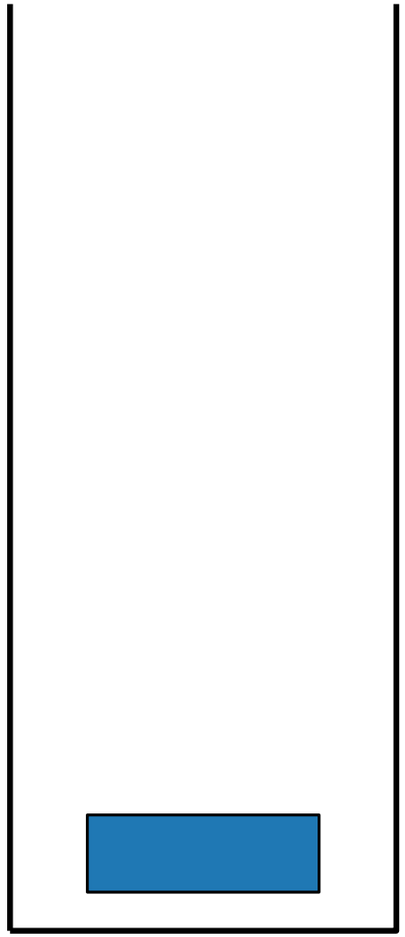


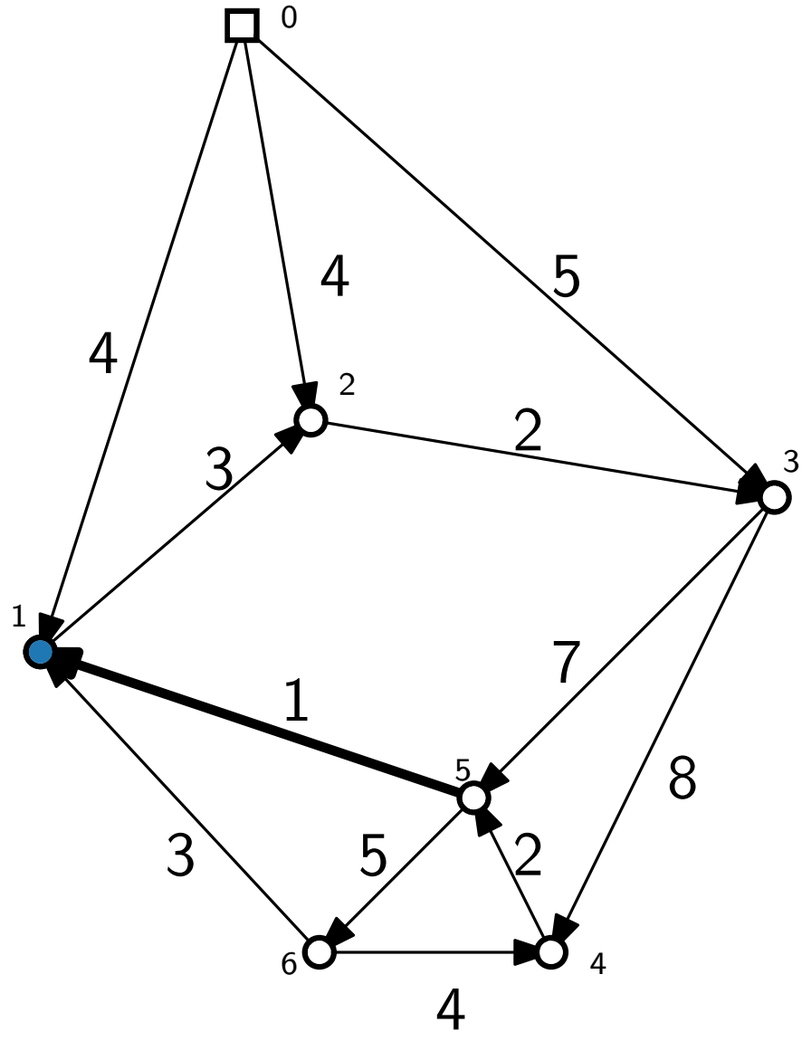
Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	besucht
c_0:	0
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0

ast: stack



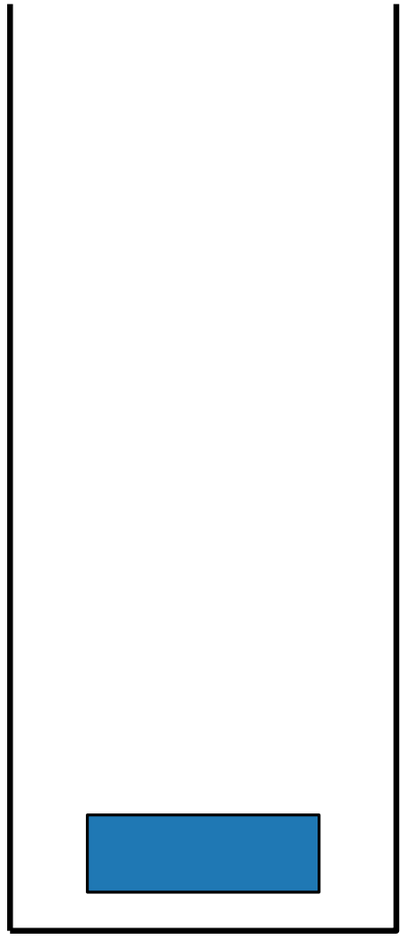


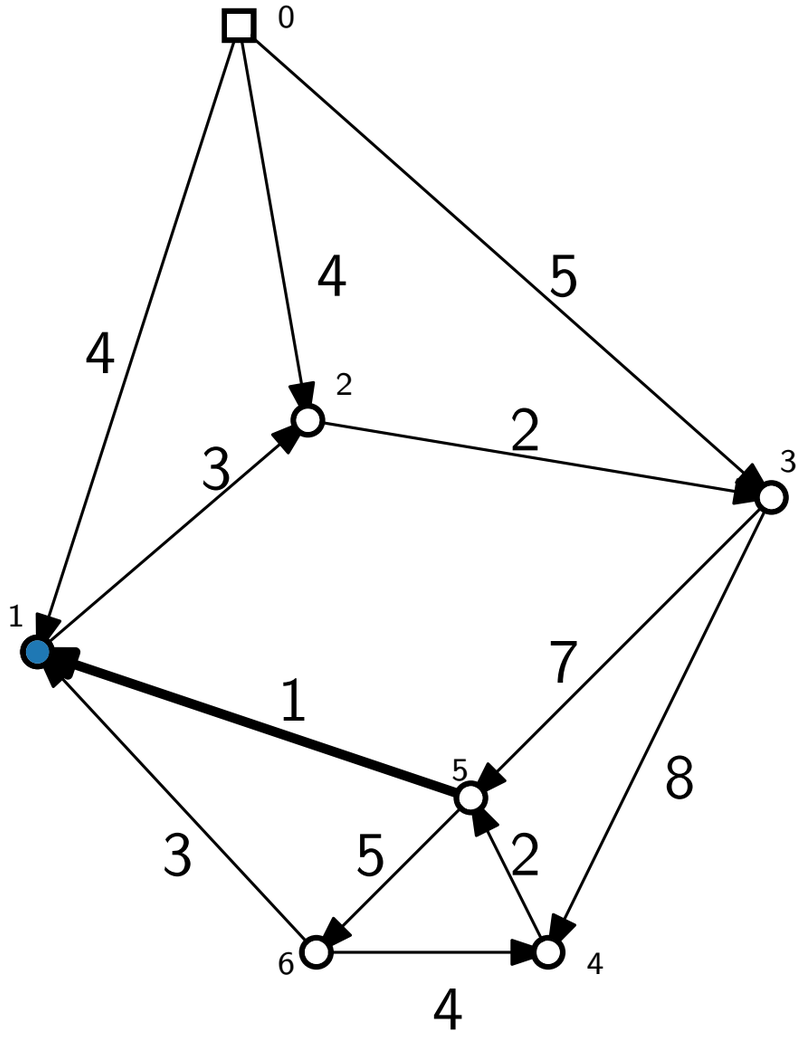
Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	besucht
c_0:	1
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0

ast: stack



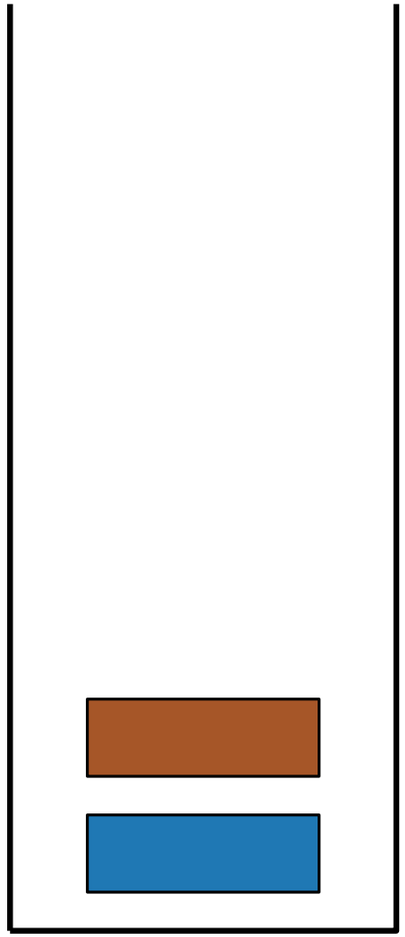


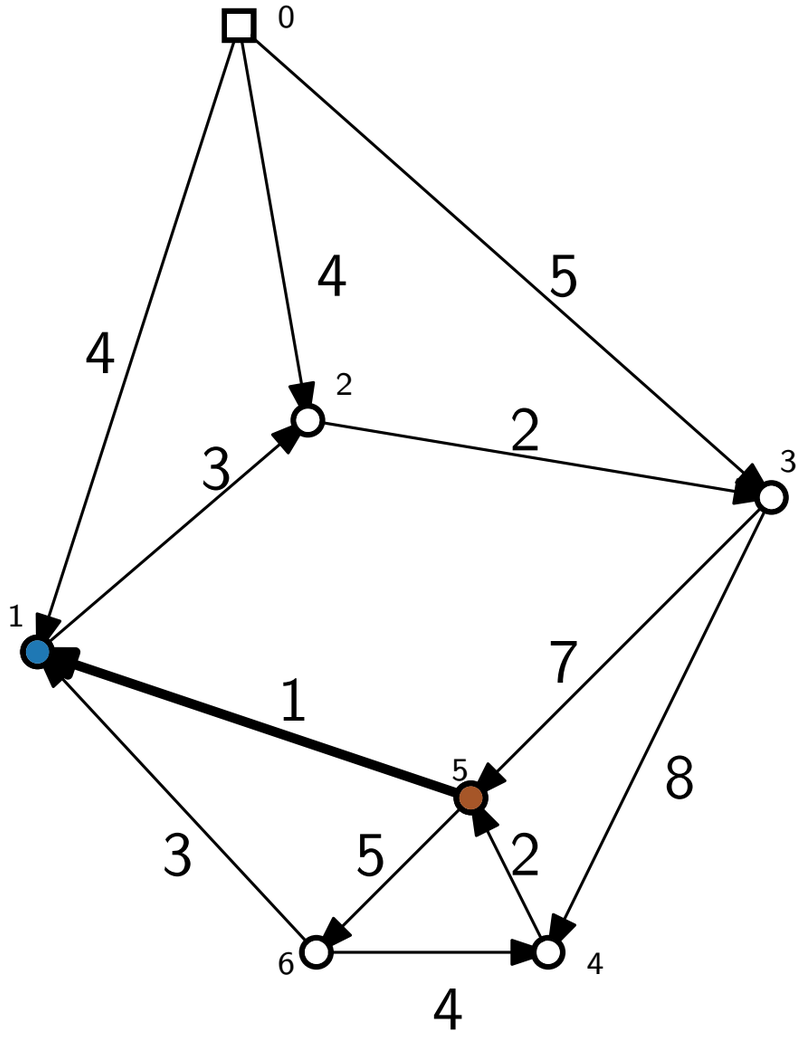
Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	besucht
c_0:	1
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0

ast: stack



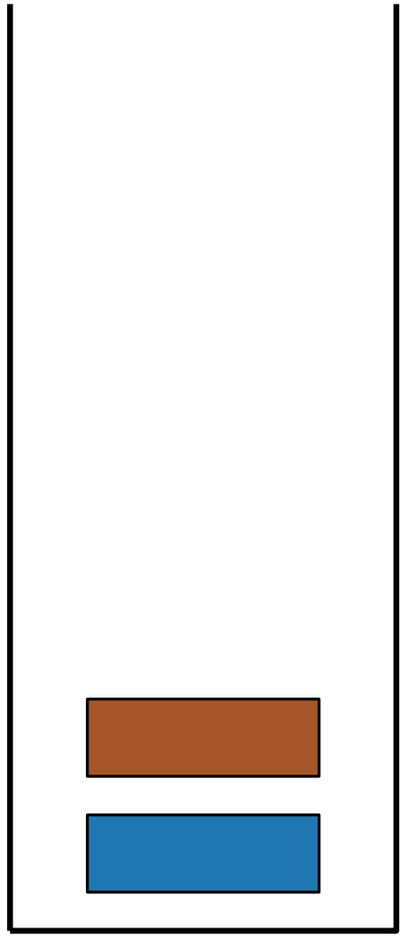


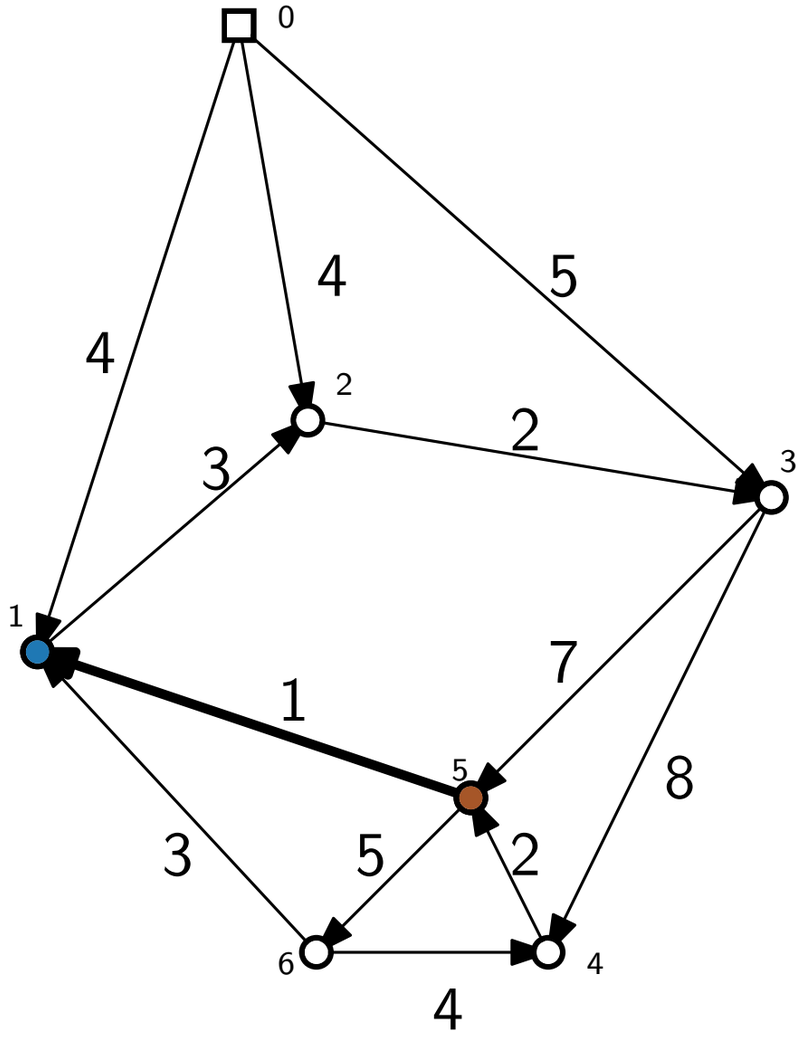
Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	besucht
c_0:	1
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	besucht
c_0:	0
status:	unbesucht
c_0:	0

ast: stack



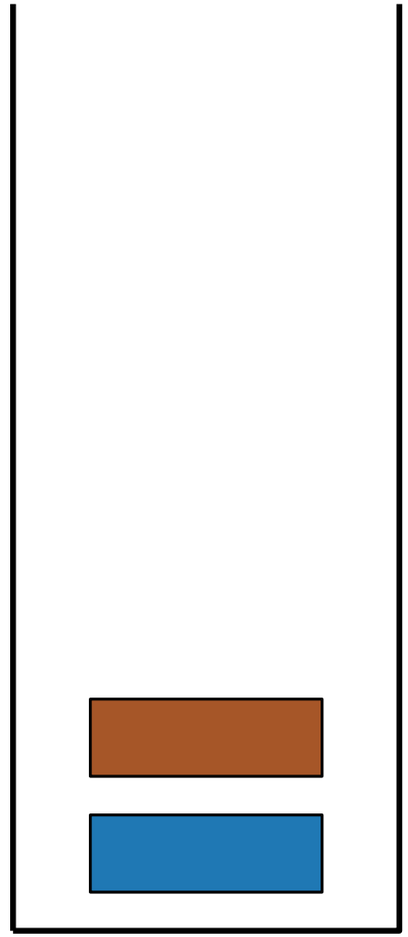


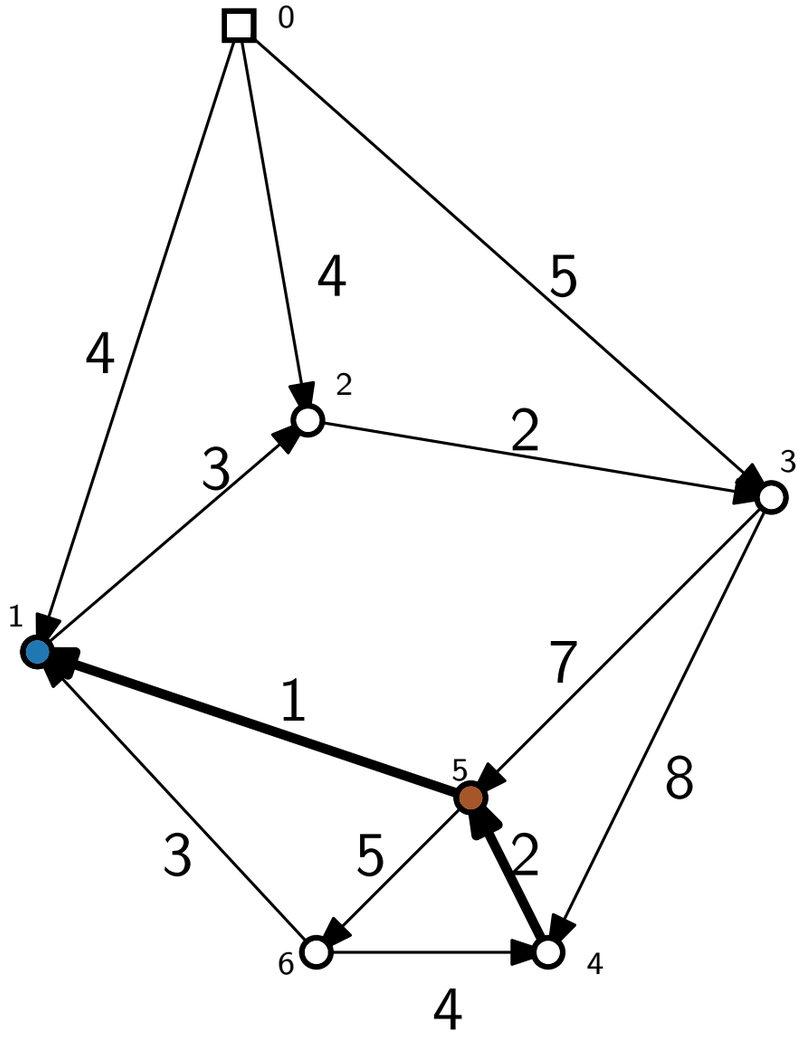
Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	besucht
c_0:	1
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	besucht
c_0:	0
status:	unbesucht
c_0:	0

ast: stack



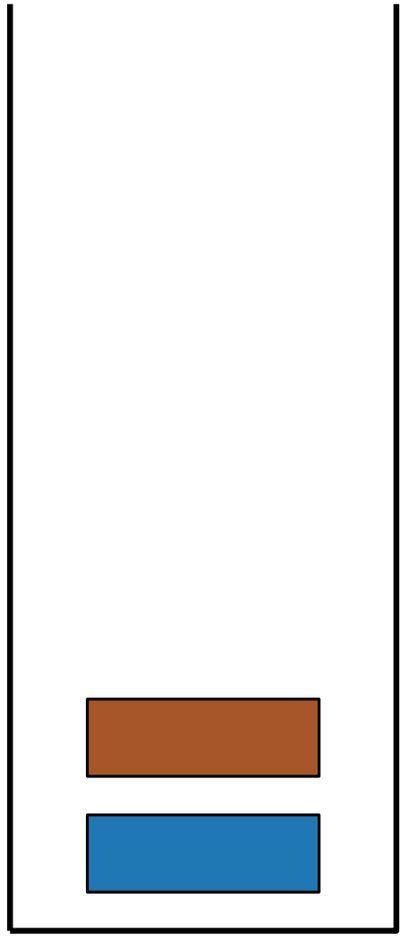


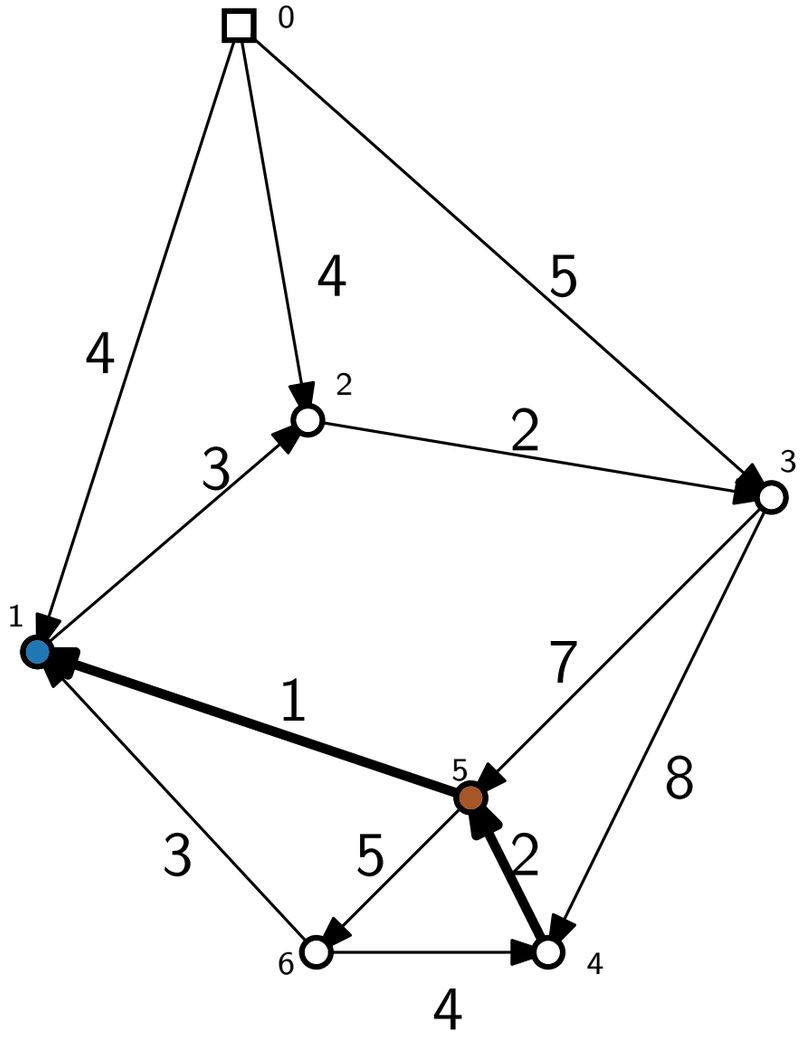
Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	besucht
c_0:	1
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	besucht
c_0:	2
status:	unbesucht
c_0:	0

ast: stack



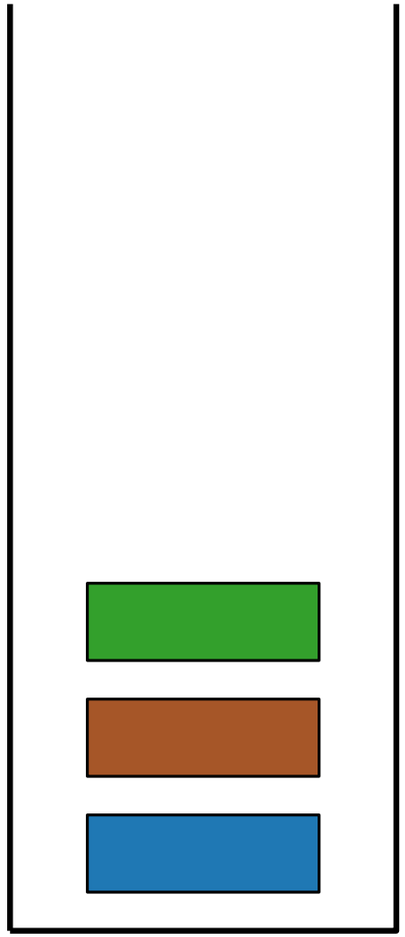


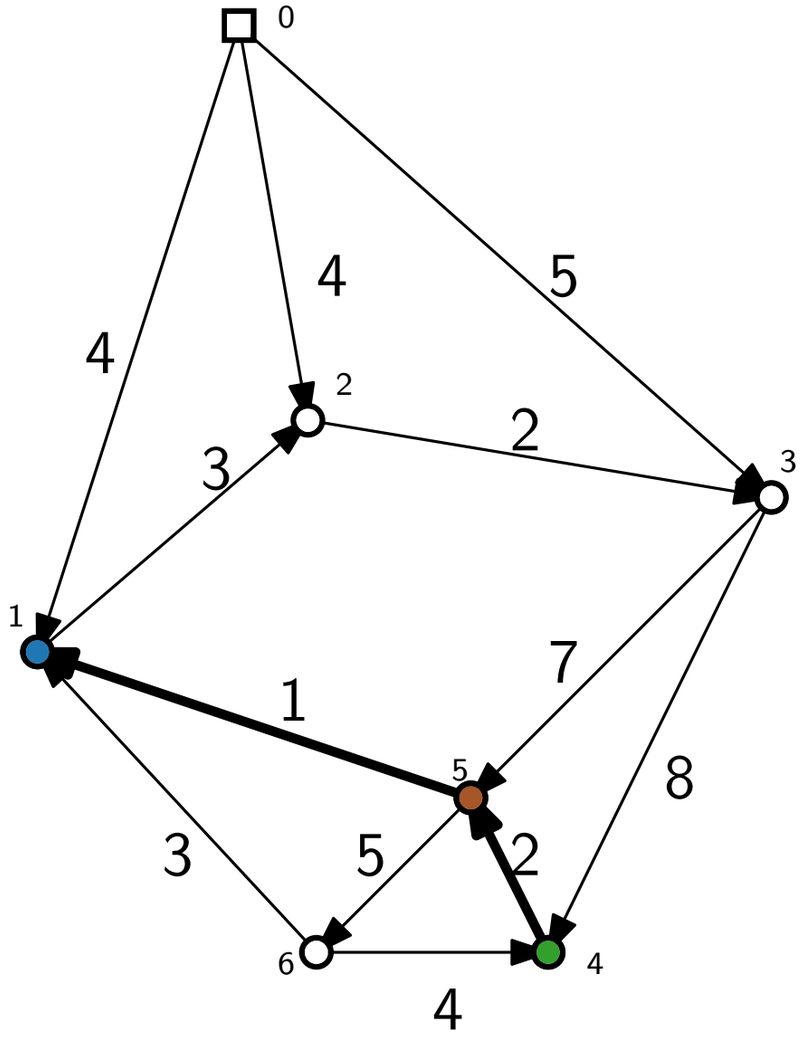
Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	besucht
c_0:	1
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	besucht
c_0:	2
status:	unbesucht
c_0:	0

ast: stack



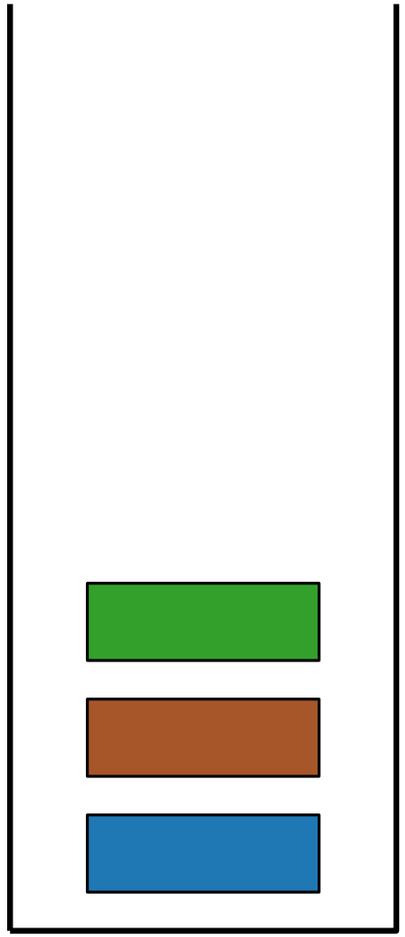


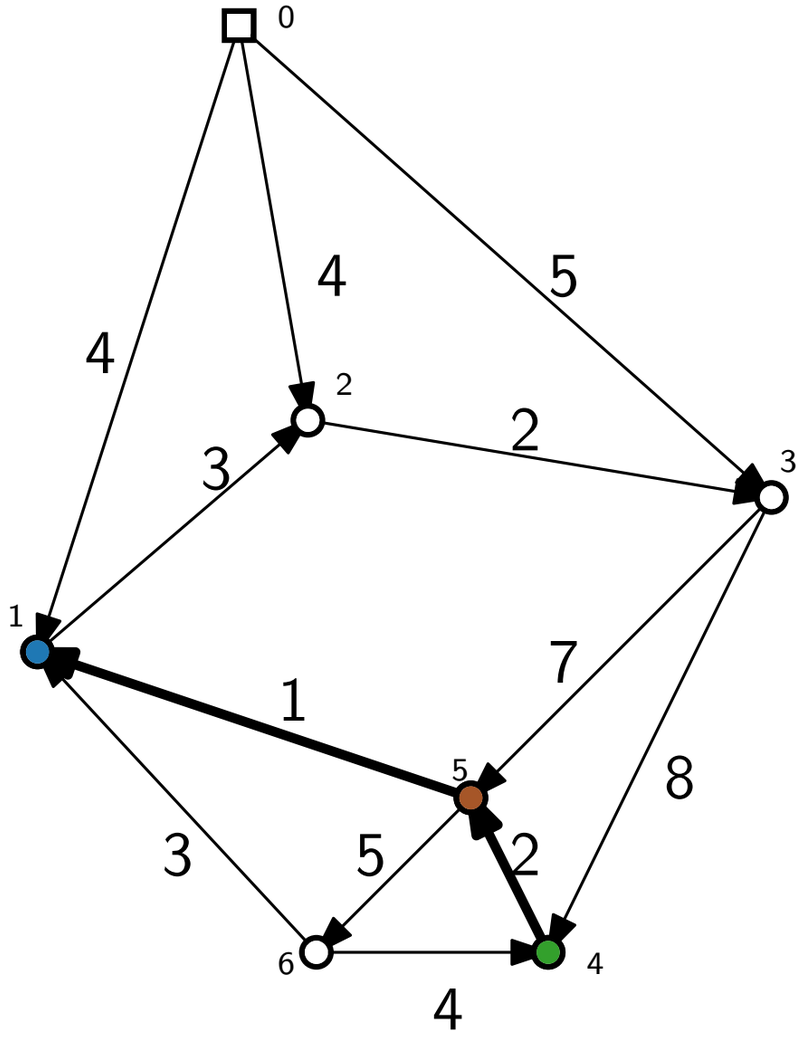
Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	besucht
c_0:	1
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	besucht
c_0:	0
status:	besucht
c_0:	2
status:	unbesucht
c_0:	0

ast: stack



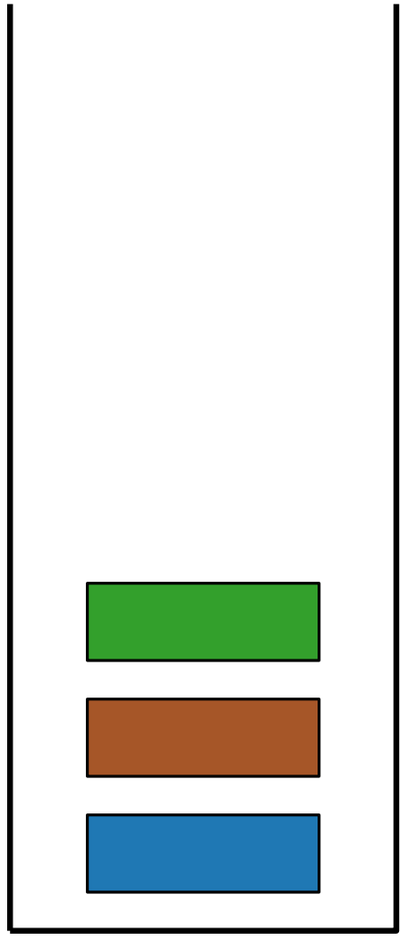


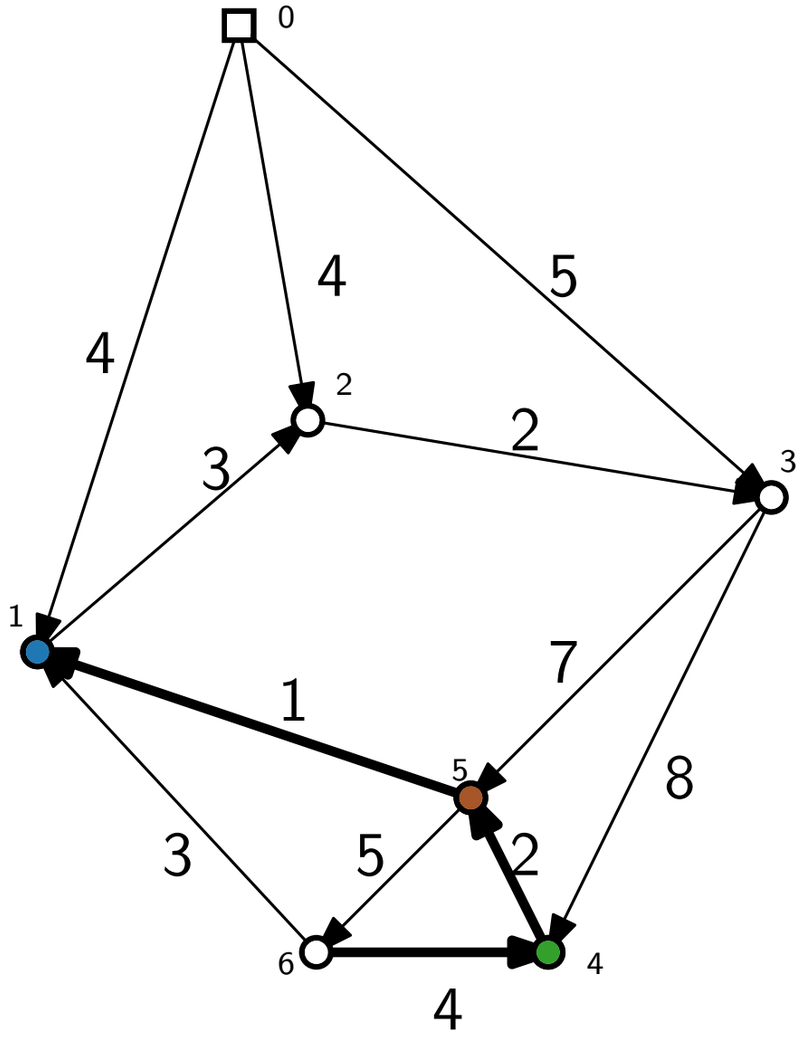
Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	besucht
c_0:	1
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	besucht
c_0:	0
status:	besucht
c_0:	2
status:	unbesucht
c_0:	0

ast: stack



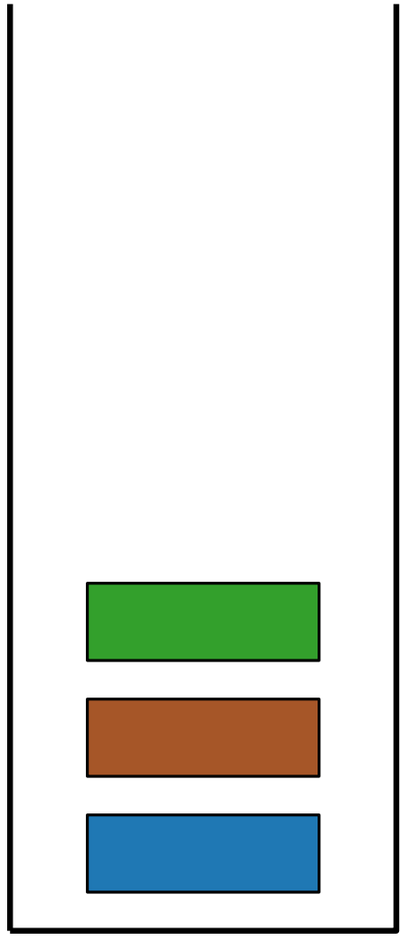


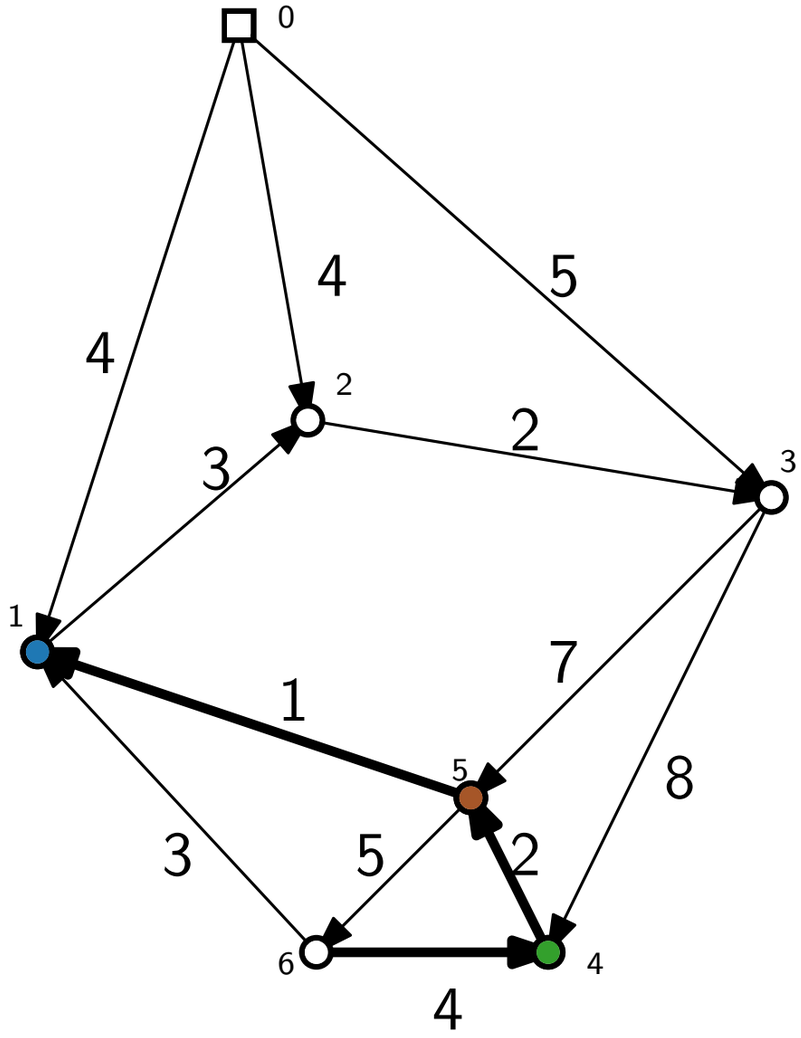
Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	besucht
c_0:	1
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	besucht
c_0:	4
status:	besucht
c_0:	2
status:	unbesucht
c_0:	0

ast: stack



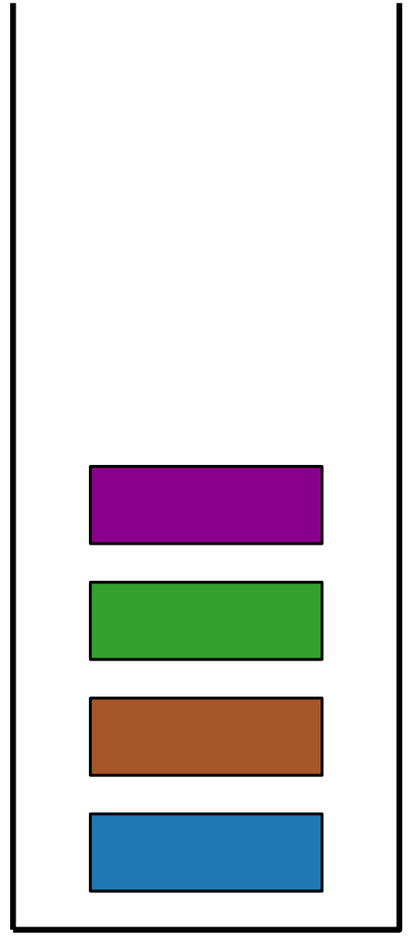


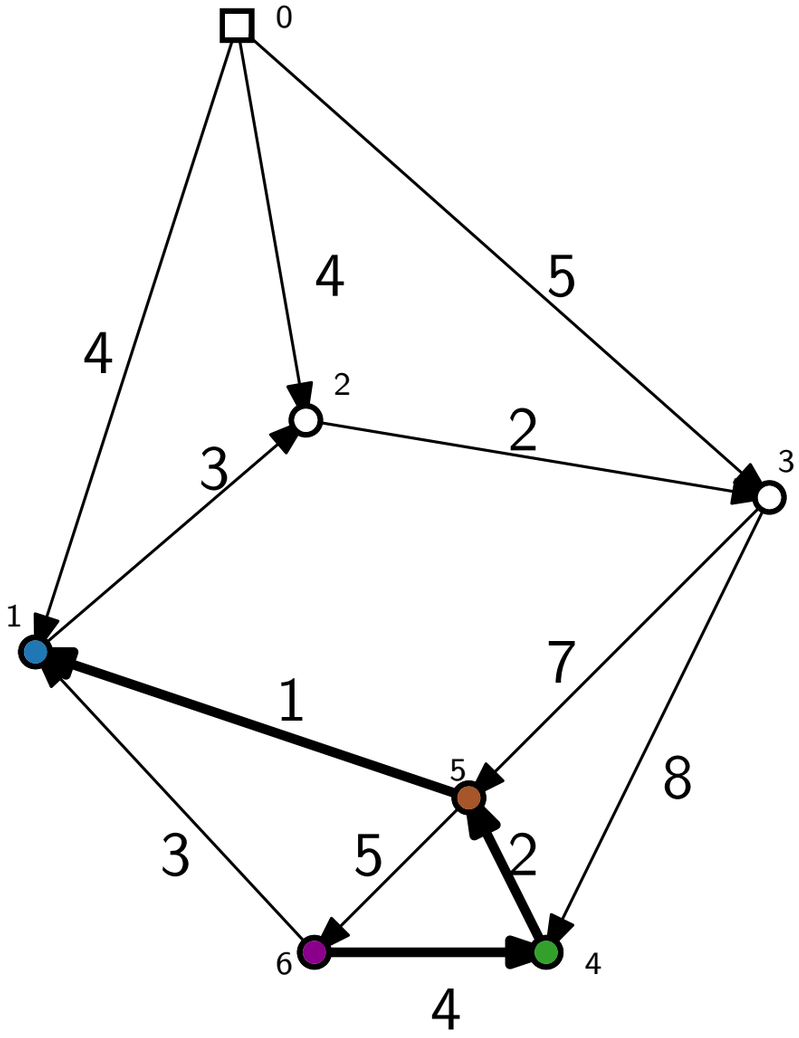
Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	besucht
c_0:	1
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	besucht
c_0:	4
status:	besucht
c_0:	2
status:	unbesucht
c_0:	0

ast: stack



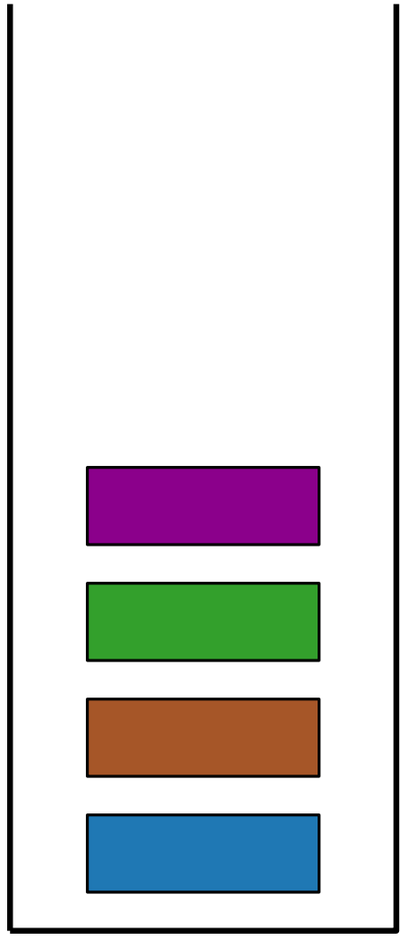


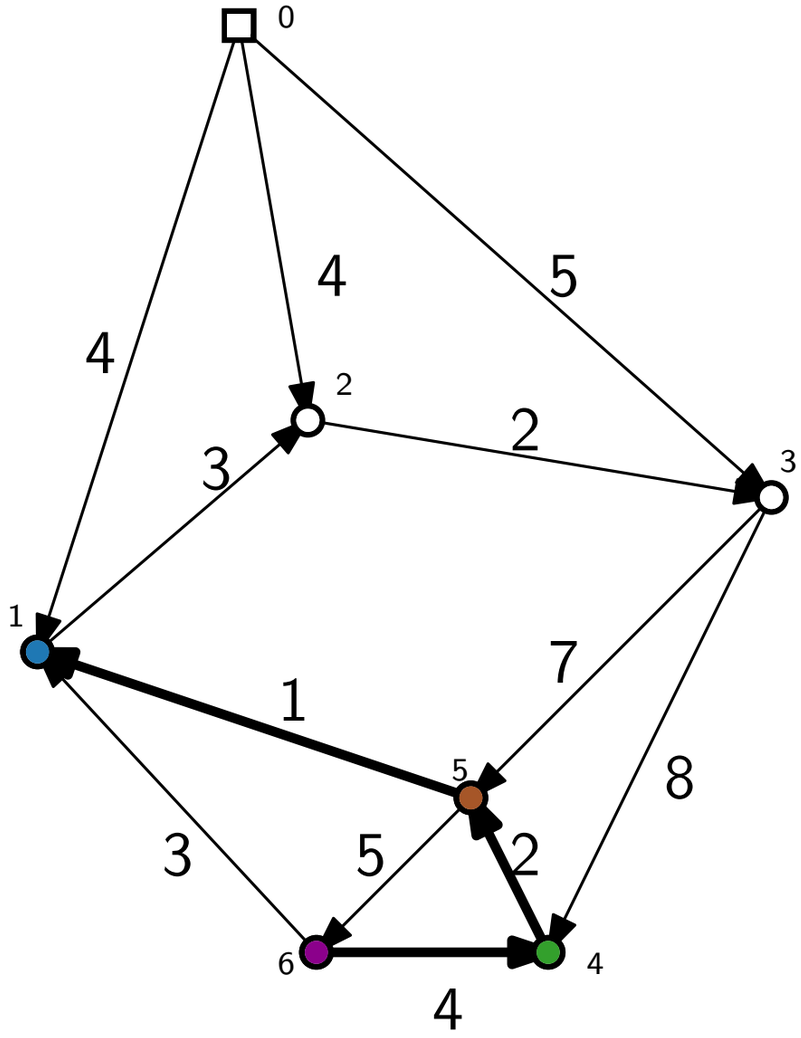
Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	besucht
c_0:	1
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	besucht
c_0:	4
status:	besucht
c_0:	2
status:	besucht
c_0:	0

ast: stack



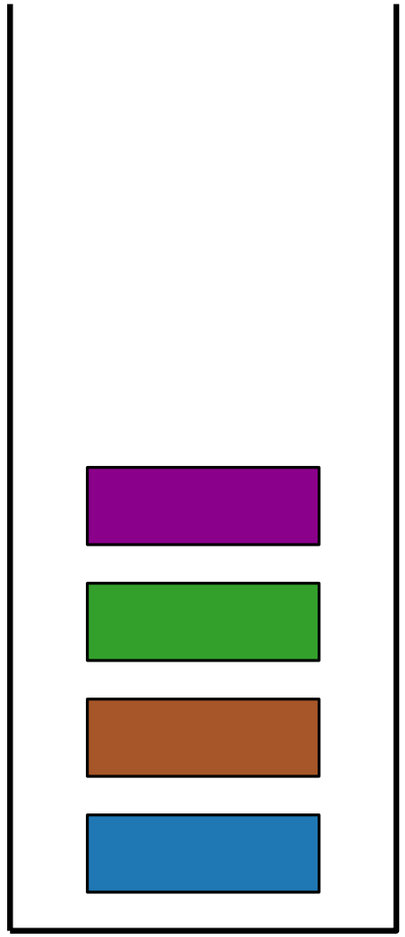


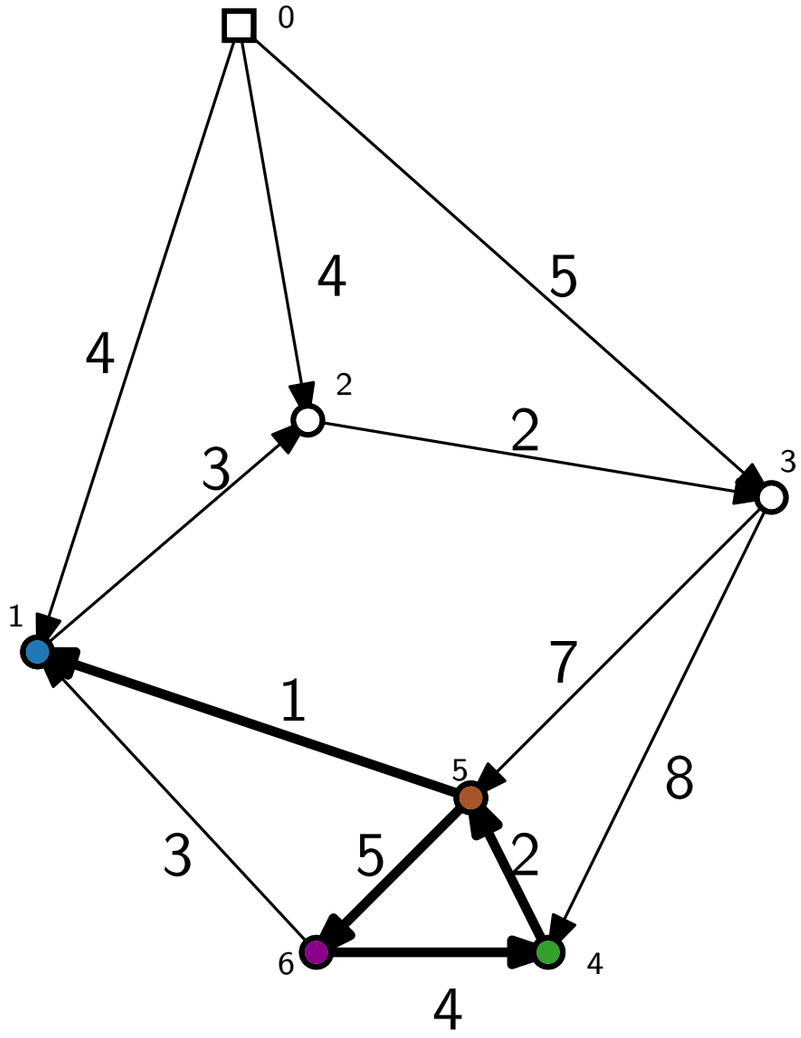
Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	besucht
c_0:	1
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	besucht
c_0:	4
status:	besucht
c_0:	2
status:	besucht
c_0:	0

ast: stack



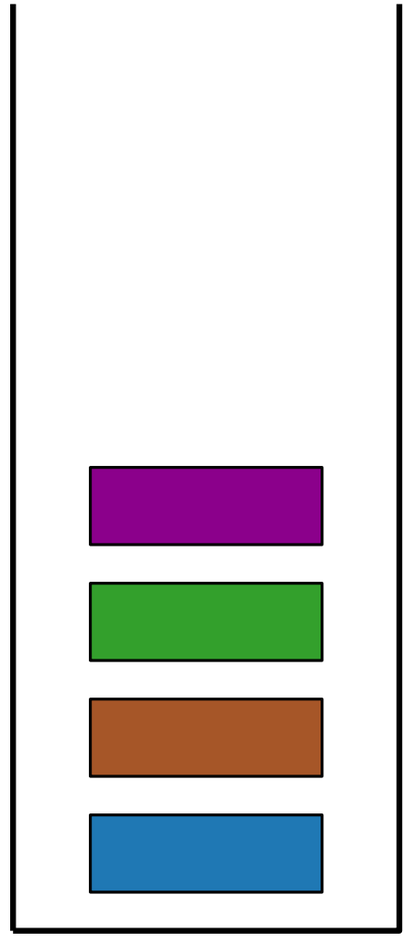


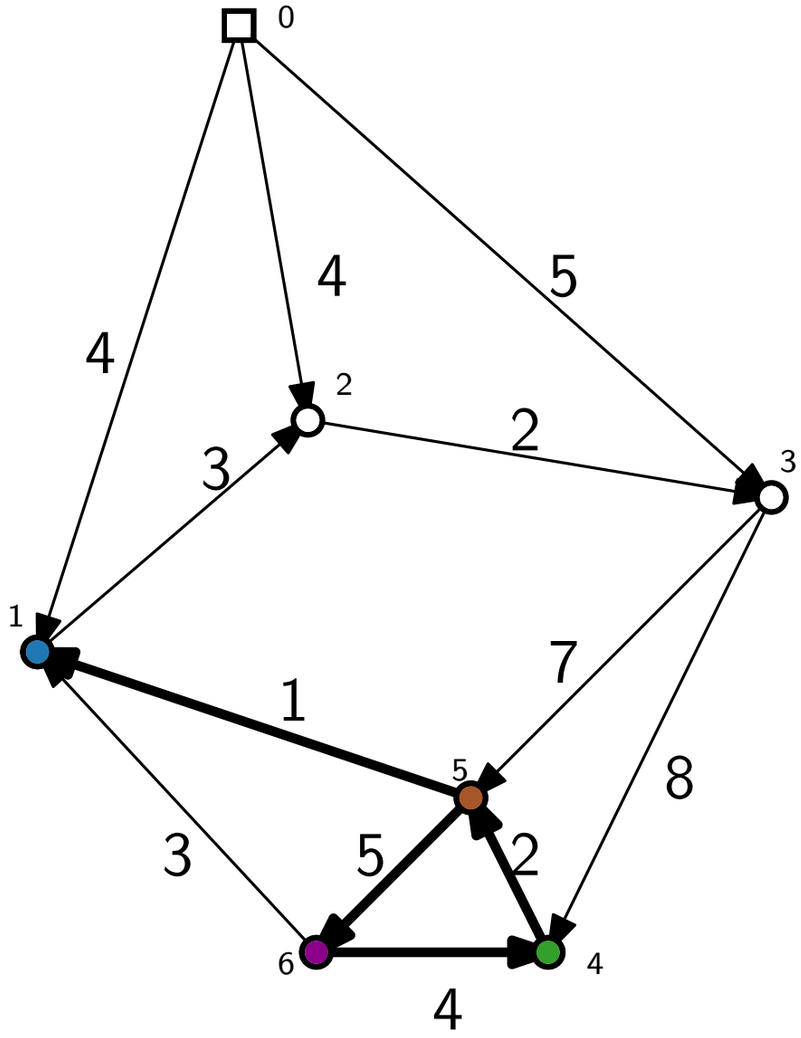
Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	besucht
c_0:	1
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	besucht
c_0:	4
status:	besucht
c_0:	2
status:	besucht
c_0:	5

ast: stack



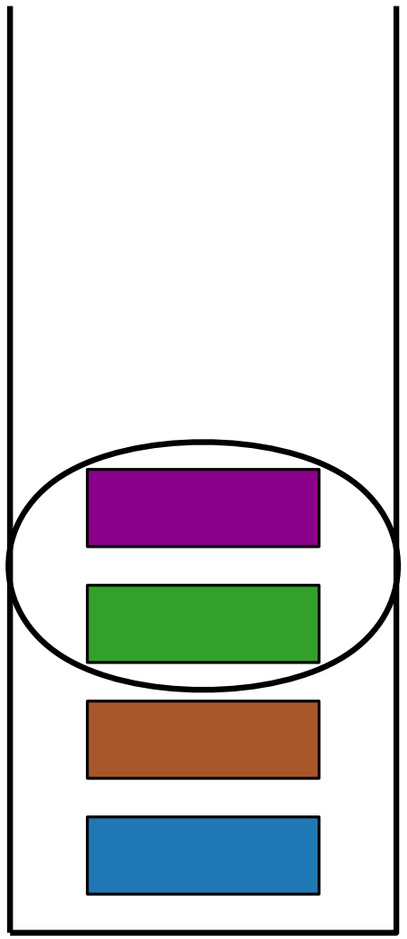


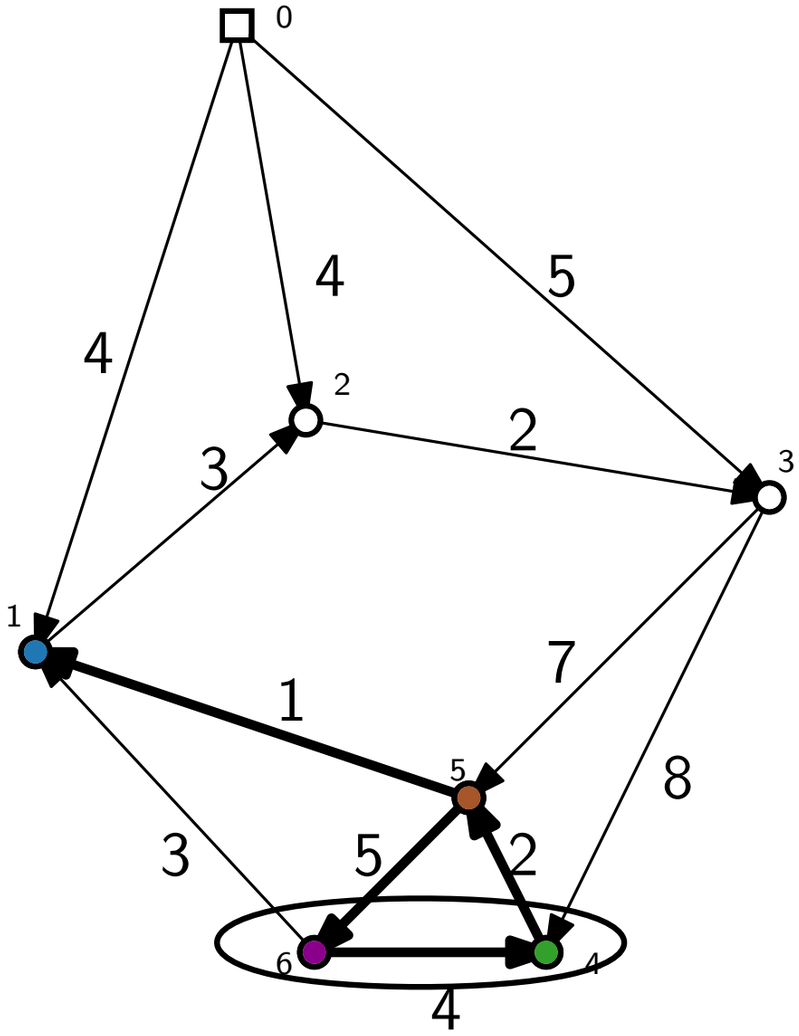
Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	besucht
c_0:	1
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	besucht
c_0:	4
status:	besucht
c_0:	2
status:	besucht
c_0:	5

ast: stack



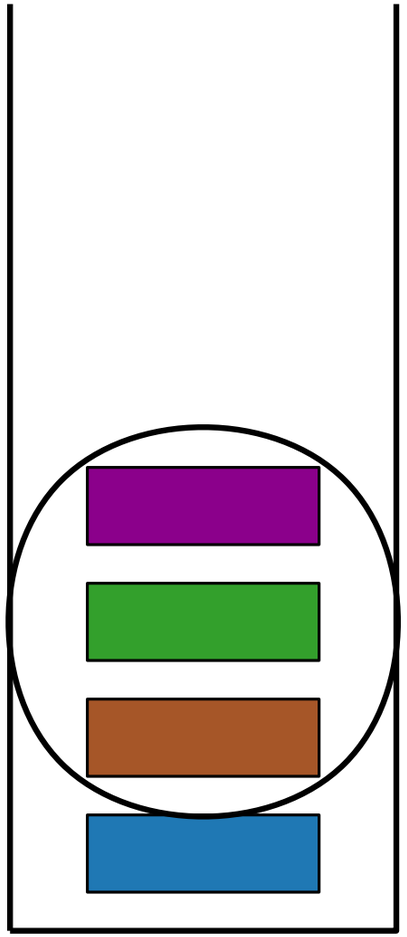


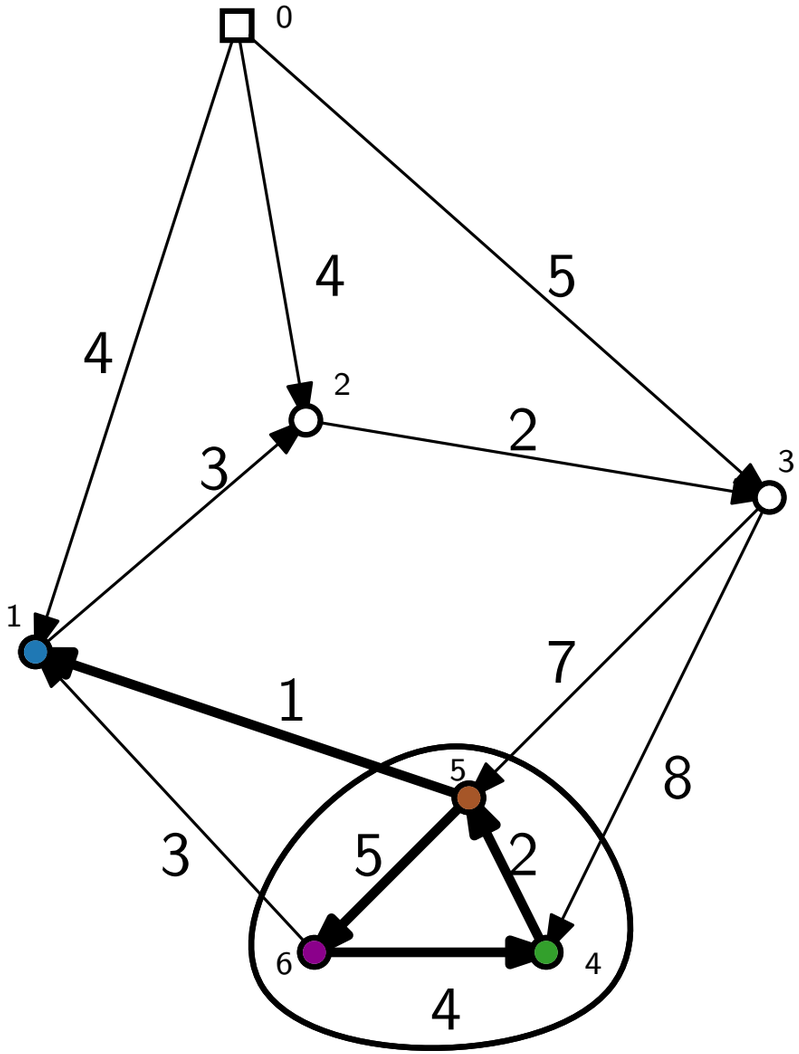
Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	besucht
c_0:	1
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	besucht
c_0:	4
status:	besucht
c_0:	2
status:	besucht
c_0:	5

ast: stack



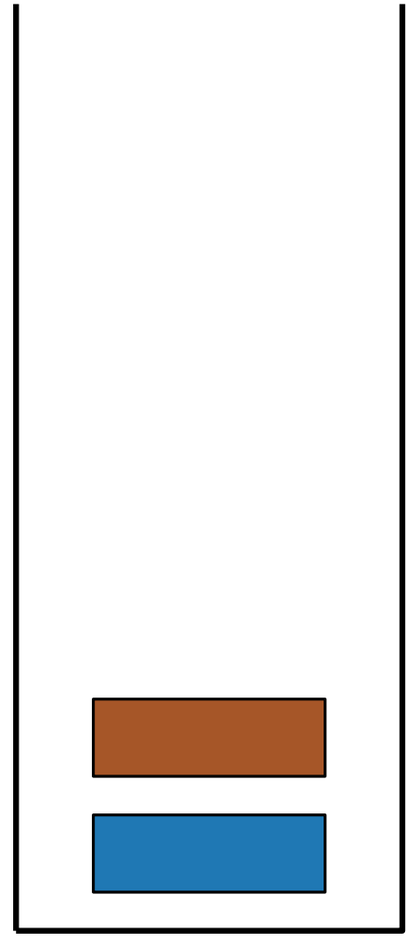


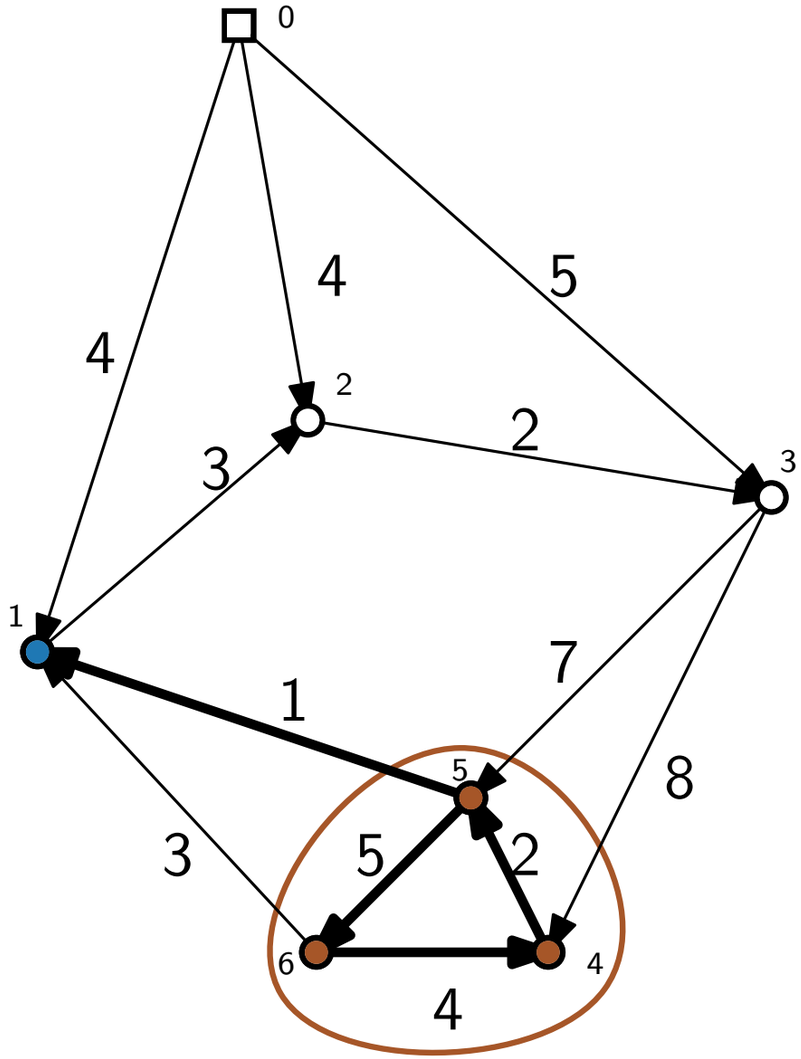
Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	besucht
c_0:	1
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	besucht
c_0:	4
status:	besucht
c_0:	2
status:	besucht
c_0:	5

ast: stack



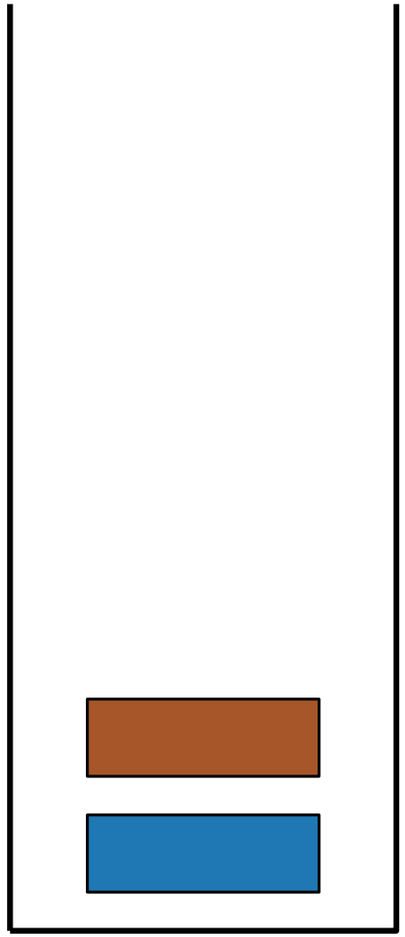


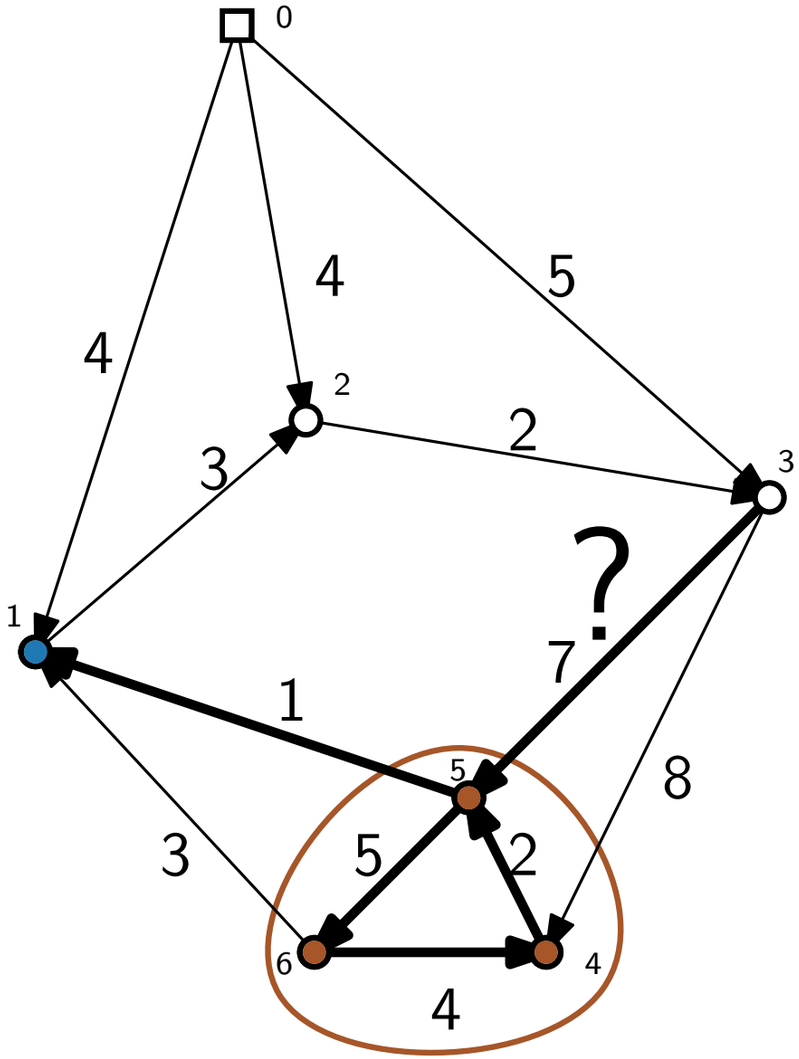
Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	besucht
c_0:	1
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	besucht
c_0:	4
status:	besucht
c_0:	2
status:	besucht
c_0:	5

ast: stack



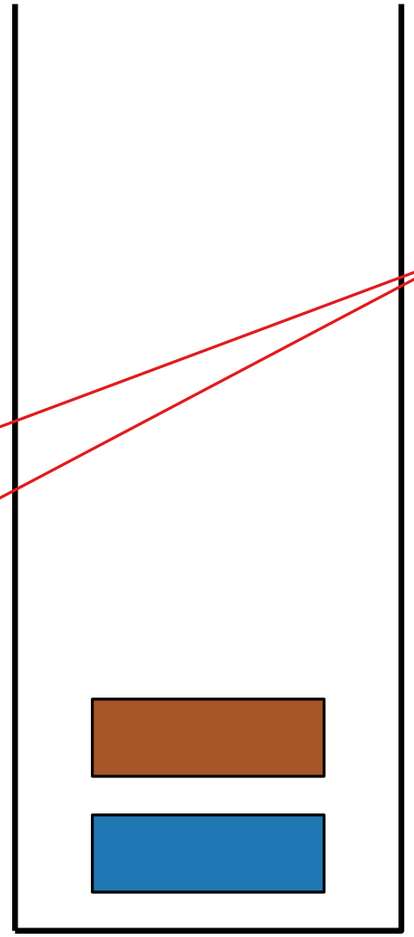


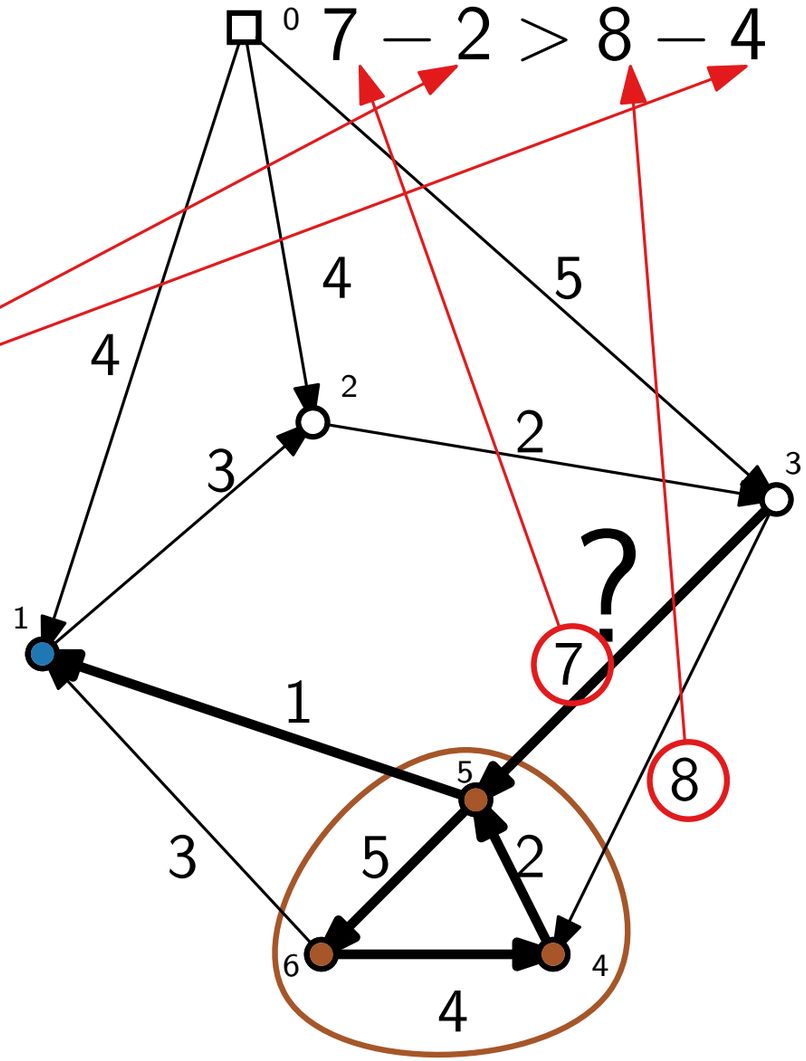
Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	besucht
c_0:	1
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	besucht
c_0:	4
status:	besucht
c_0:	2
status:	besucht
c_0:	5

ast: stack



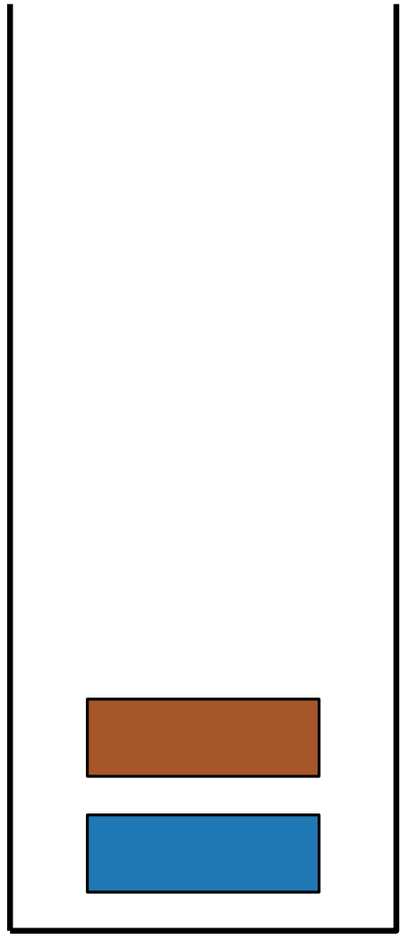


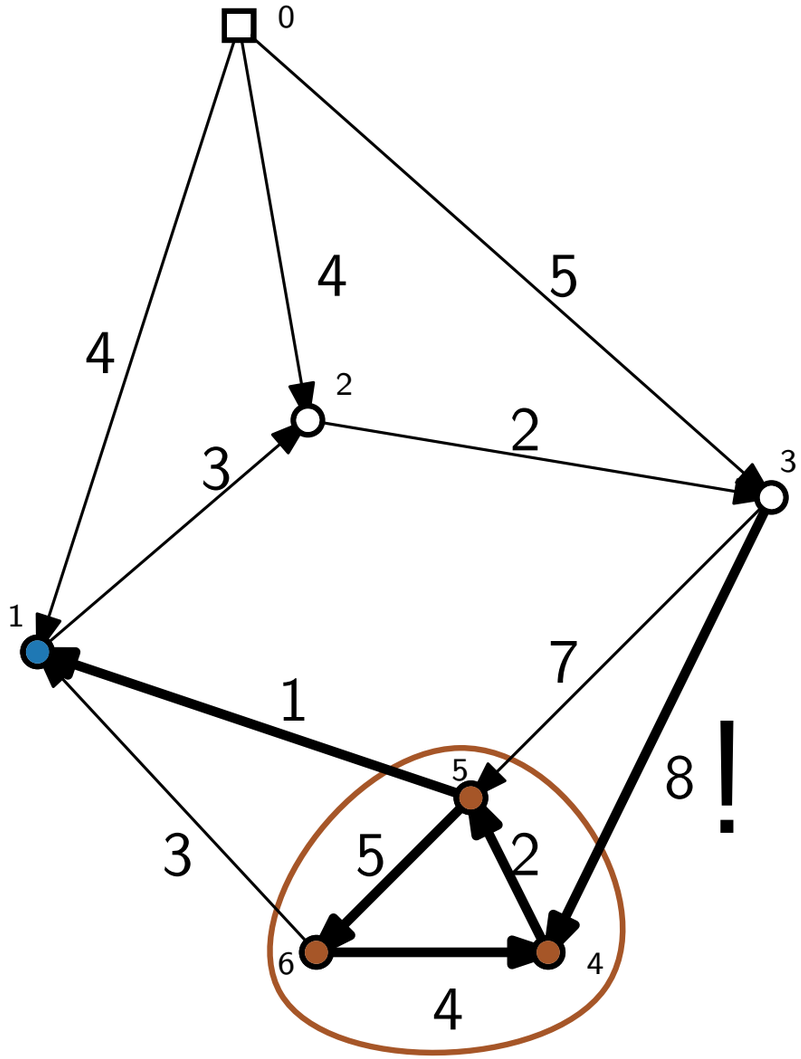
Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	besucht
c_0:	1
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	besucht
c_0:	4
status:	besucht
c_0:	2
status:	besucht
c_0:	5

ast: stack



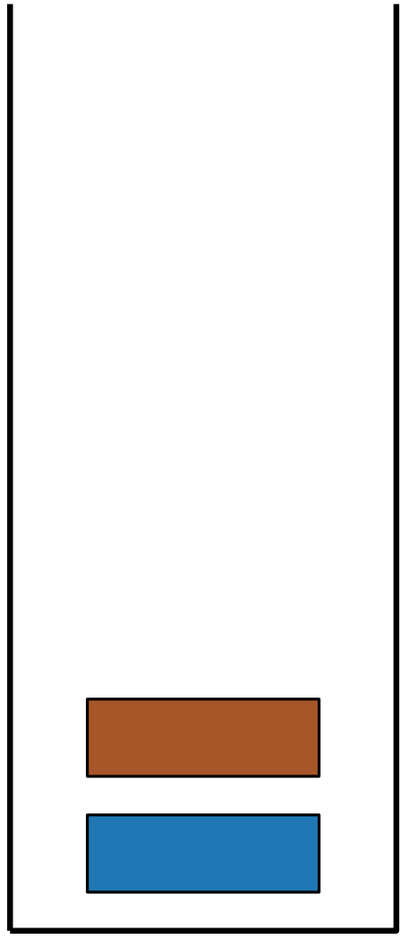


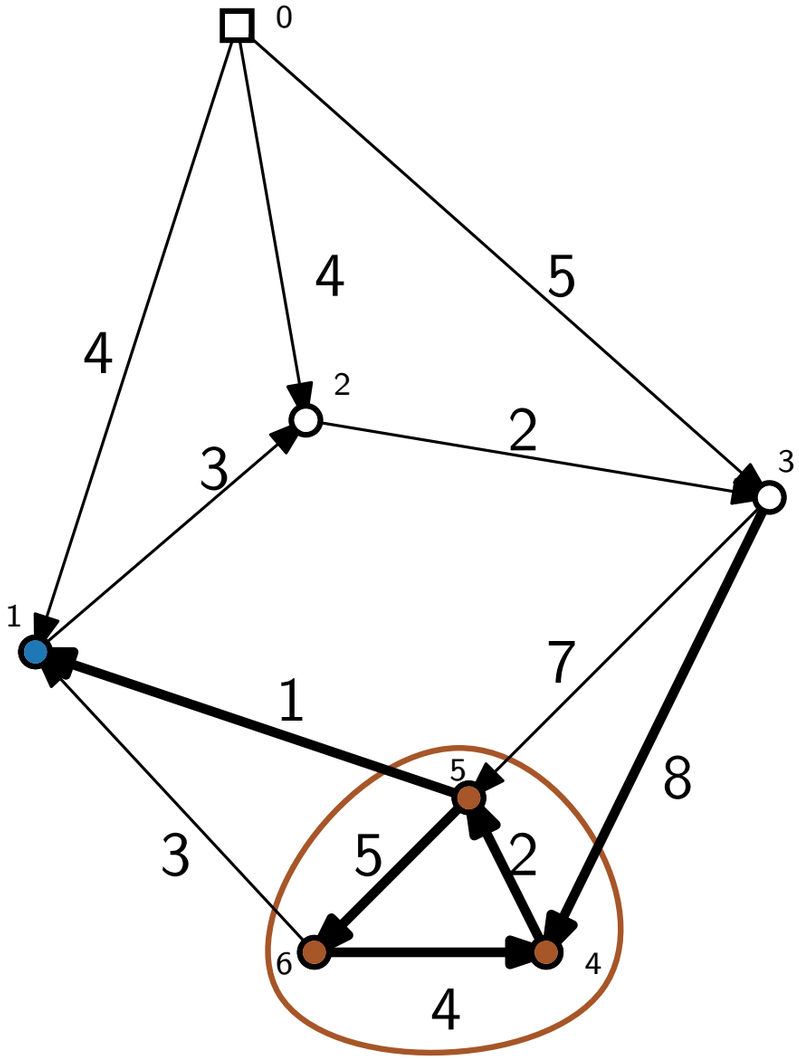
Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	besucht
c_0:	1
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	besucht
c_0:	8
status:	besucht
c_0:	8
status:	besucht
c_0:	8

ast: stack



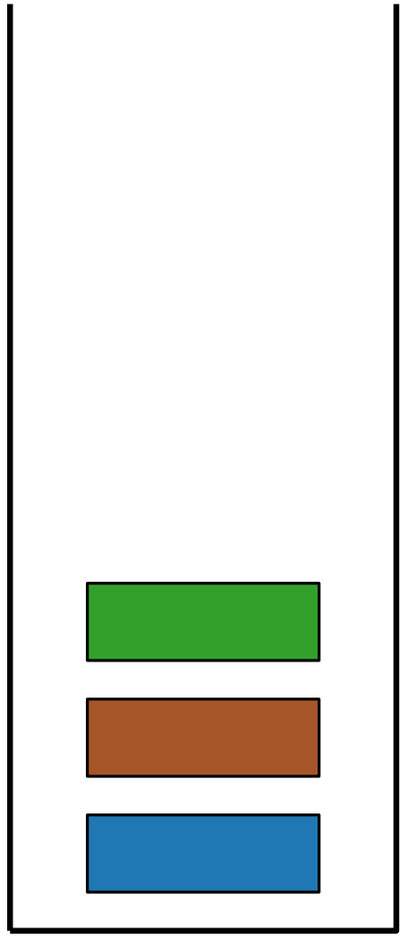


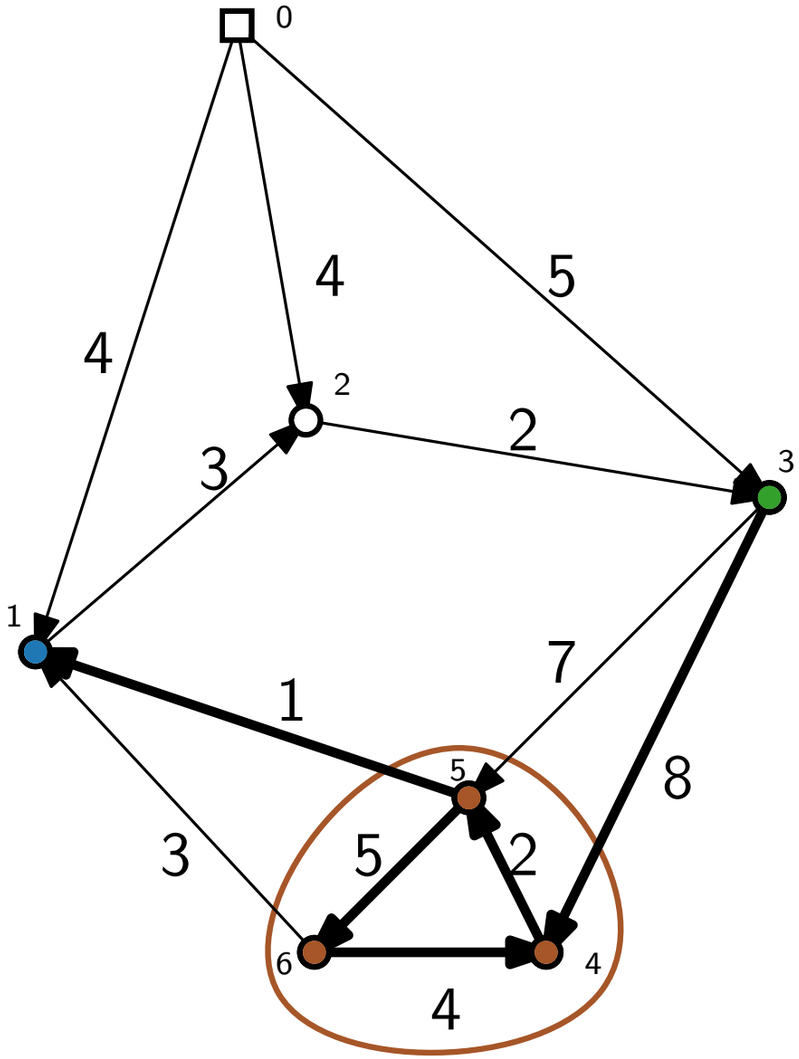
Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	besucht
c_0:	1
status:	unbesucht
c_0:	0
status:	unbesucht
c_0:	0
status:	besucht
c_0:	8
status:	besucht
c_0:	8
status:	besucht
c_0:	8

ast: stack



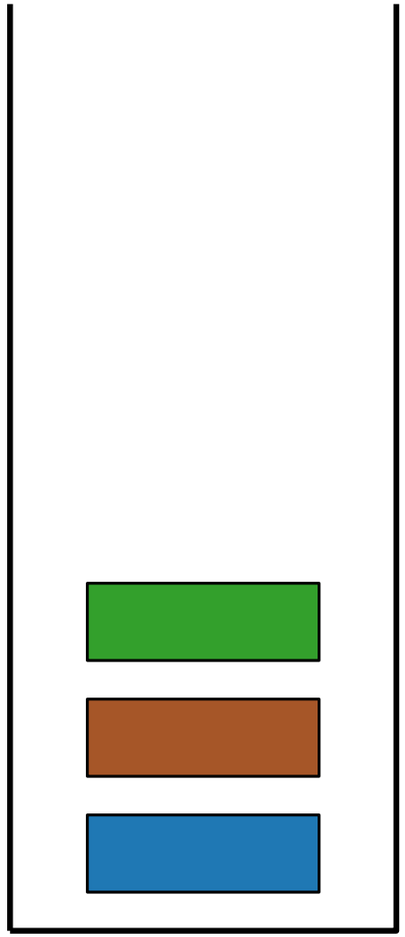


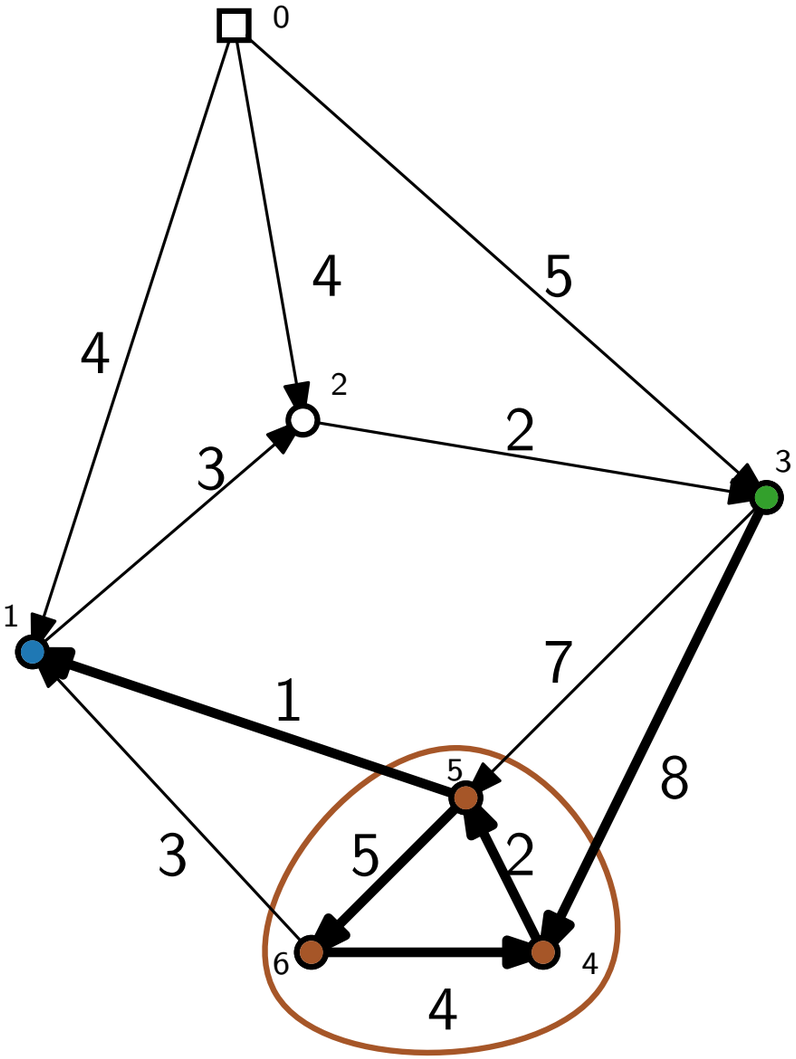
Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	besucht
c_0:	1
status:	unbesucht
c_0:	0
status:	besucht
c_0:	0
status:	besucht
c_0:	8
status:	besucht
c_0:	8
status:	besucht
c_0:	8

ast: stack



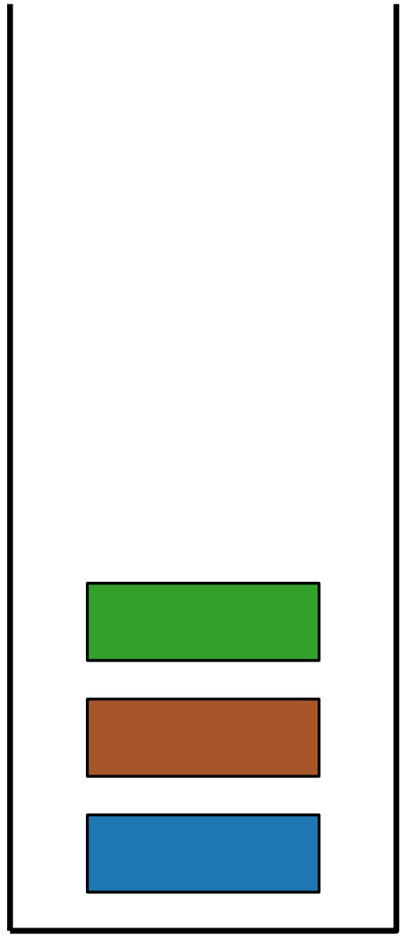


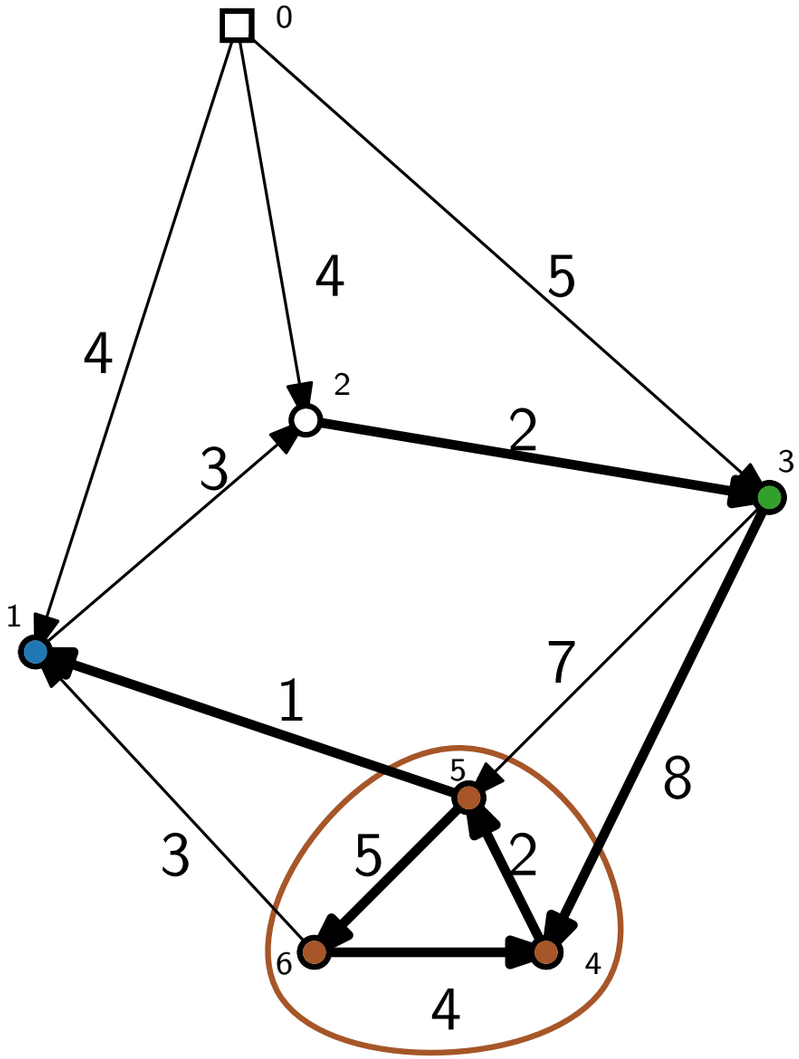
Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	besucht
c_0:	1
status:	unbesucht
c_0:	0
status:	besucht
c_0:	0
status:	besucht
c_0:	8
status:	besucht
c_0:	8
status:	besucht
c_0:	8

ast: stack



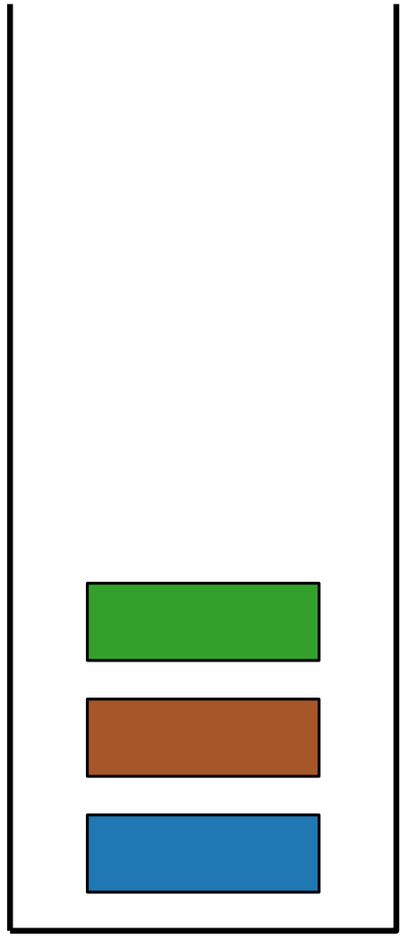


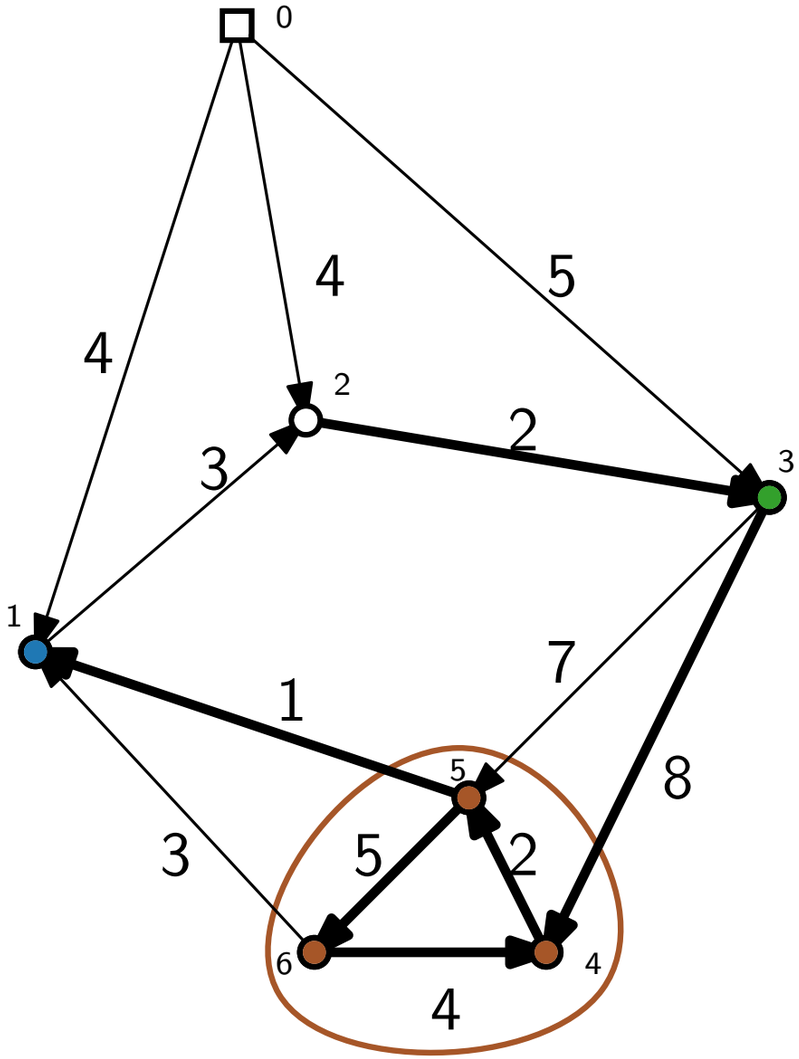
Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	besucht
c_0:	1
status:	unbesucht
c_0:	0
status:	besucht
c_0:	2
status:	besucht
c_0:	8
status:	besucht
c_0:	8
status:	besucht
c_0:	8

ast: stack



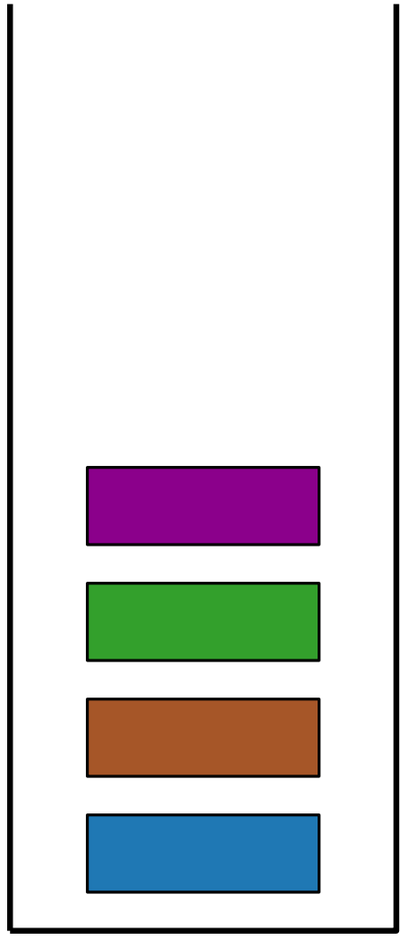


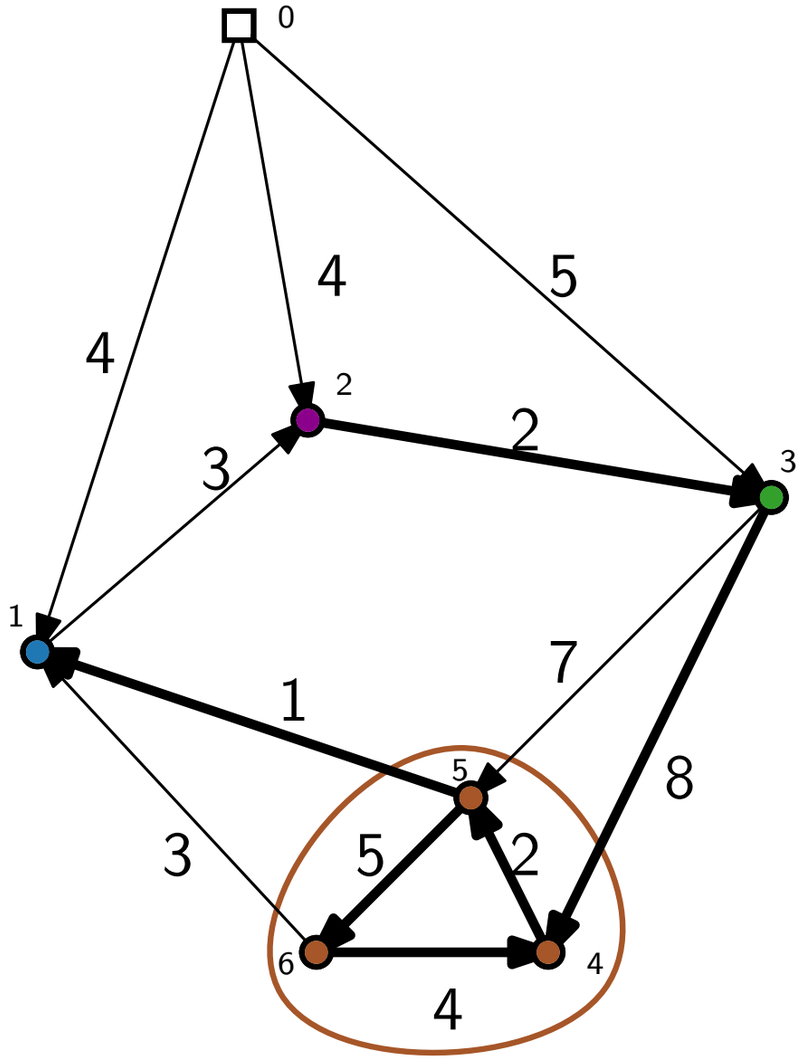
Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	besucht
c_0:	1
status:	unbesucht
c_0:	0
status:	besucht
c_0:	2
status:	besucht
c_0:	8
status:	besucht
c_0:	8
status:	besucht
c_0:	8

ast: stack



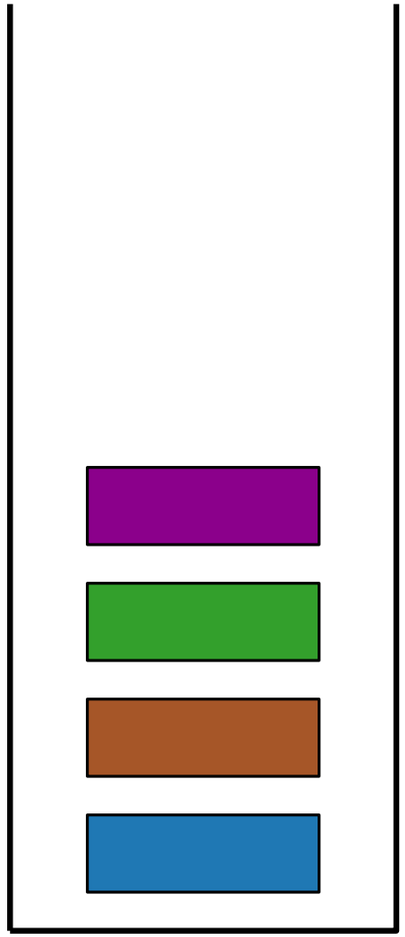


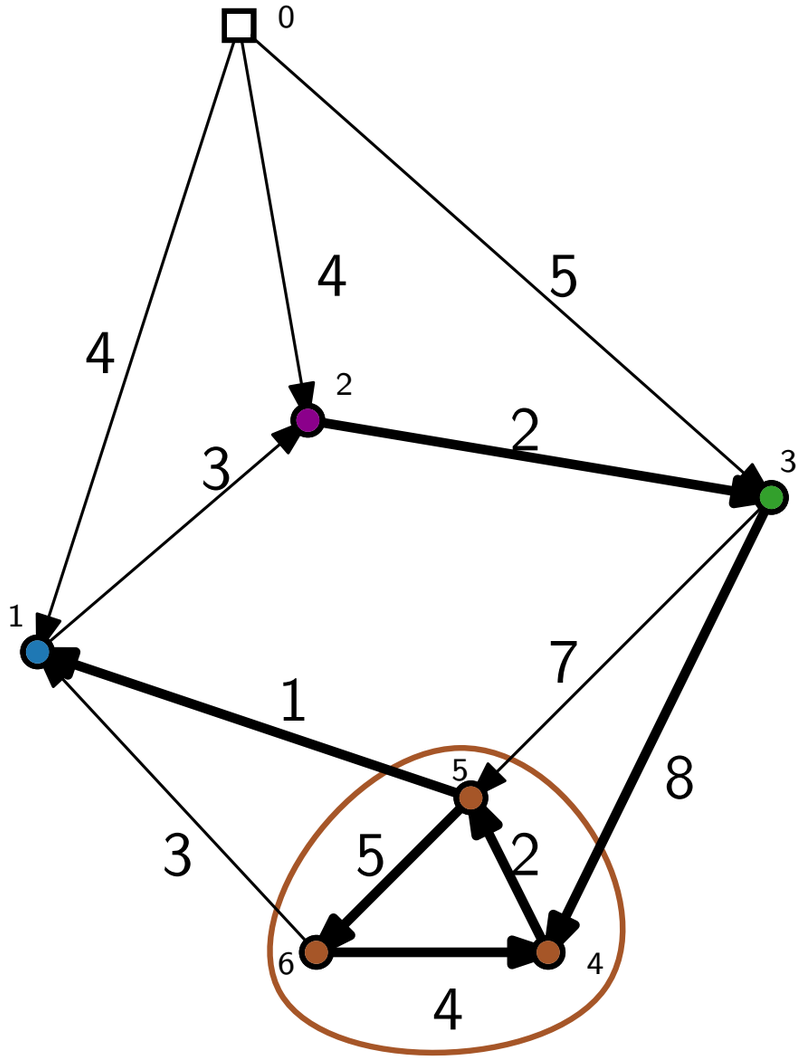
Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	besucht
c_0:	1
status:	besucht
c_0:	0
status:	besucht
c_0:	2
status:	besucht
c_0:	8
status:	besucht
c_0:	8
status:	besucht
c_0:	8

ast: stack



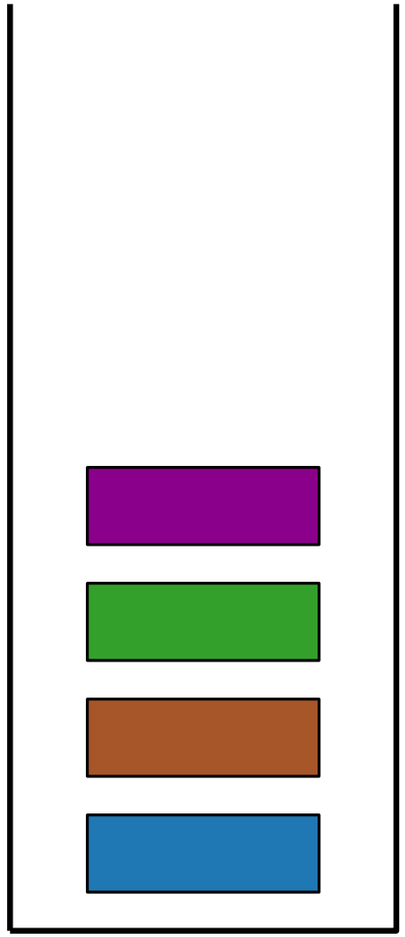


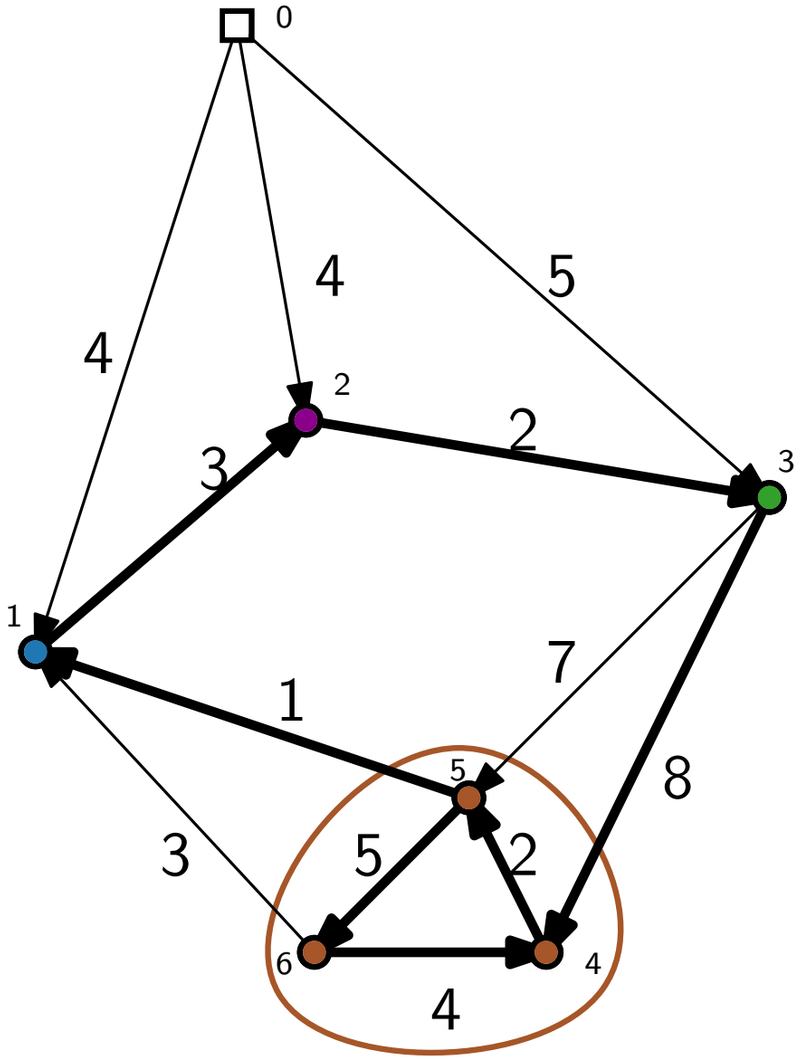
Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	besucht
c_0:	1
status:	besucht
c_0:	0
status:	besucht
c_0:	2
status:	besucht
c_0:	8
status:	besucht
c_0:	8
status:	besucht
c_0:	8

ast: stack



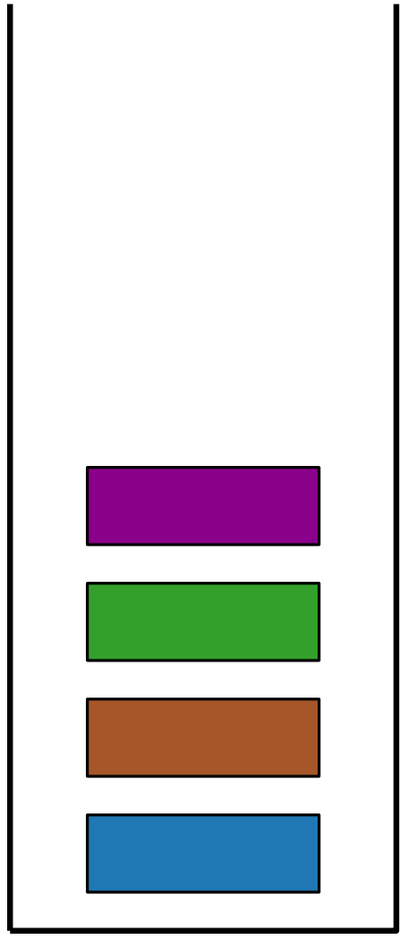


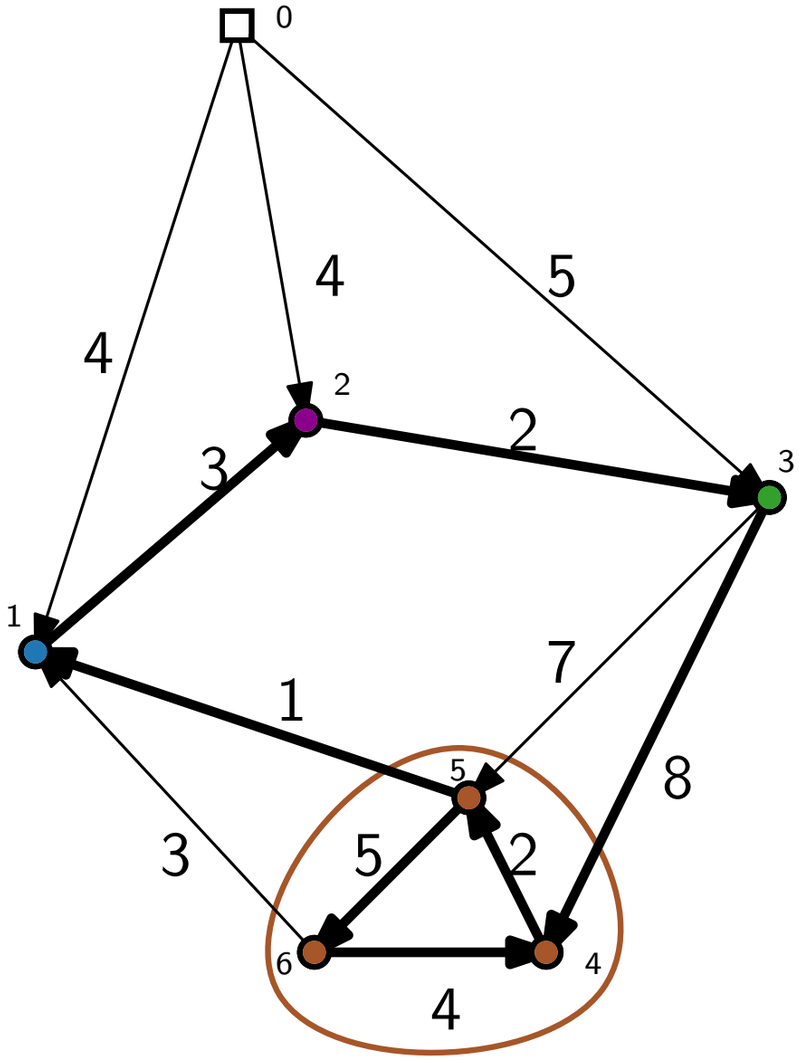
Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	besucht
c_0:	1
status:	besucht
c_0:	3
status:	besucht
c_0:	2
status:	besucht
c_0:	8
status:	besucht
c_0:	8
status:	besucht
c_0:	8

ast: stack



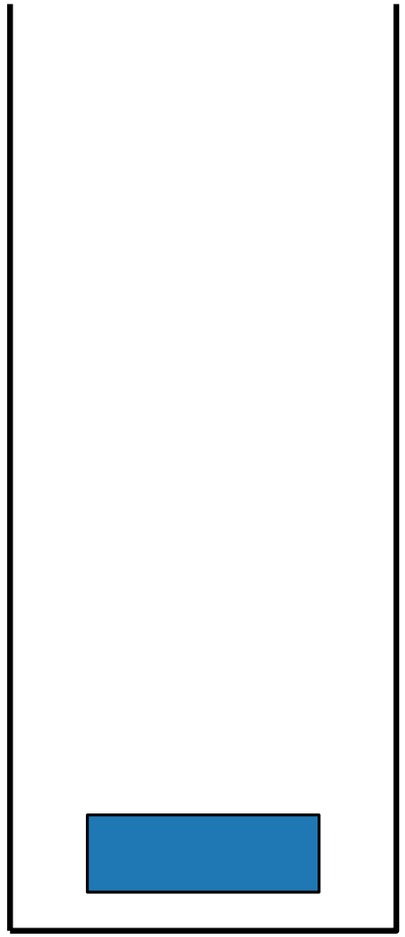


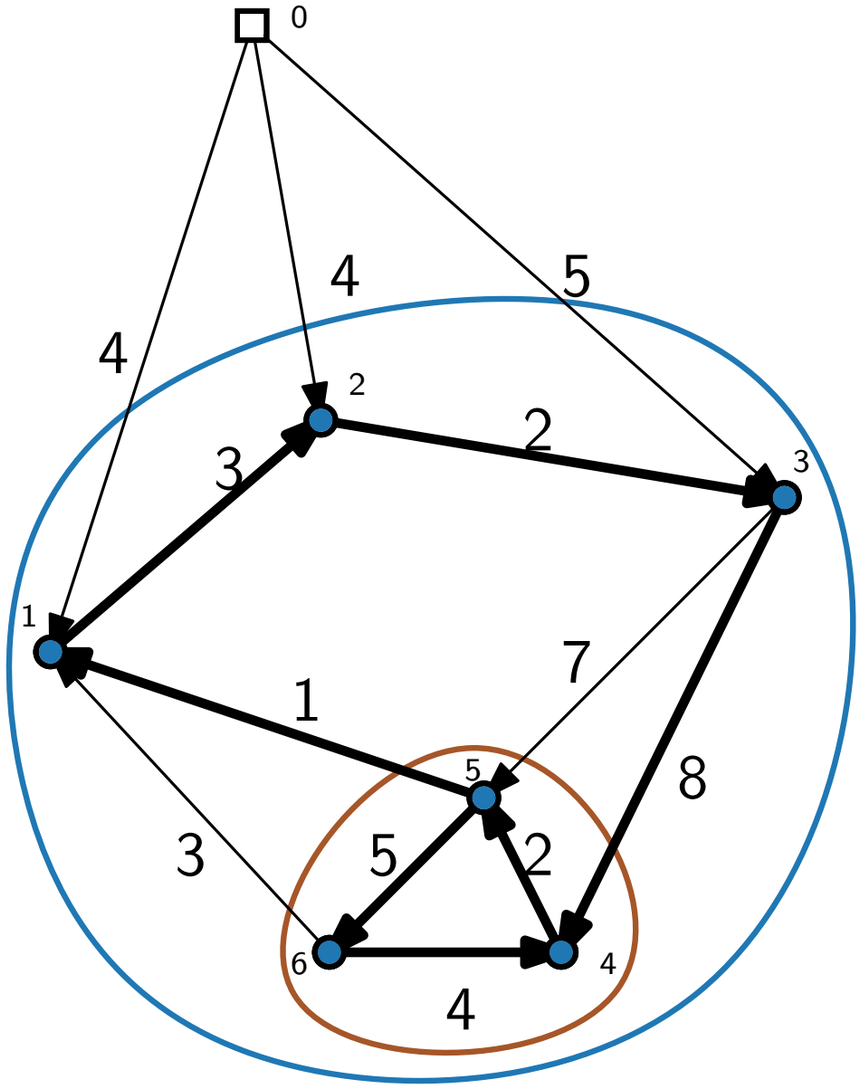
Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	besucht
c_0:	1
status:	besucht
c_0:	3
status:	besucht
c_0:	2
status:	besucht
c_0:	8
status:	besucht
c_0:	8
status:	besucht
c_0:	8

ast: stack



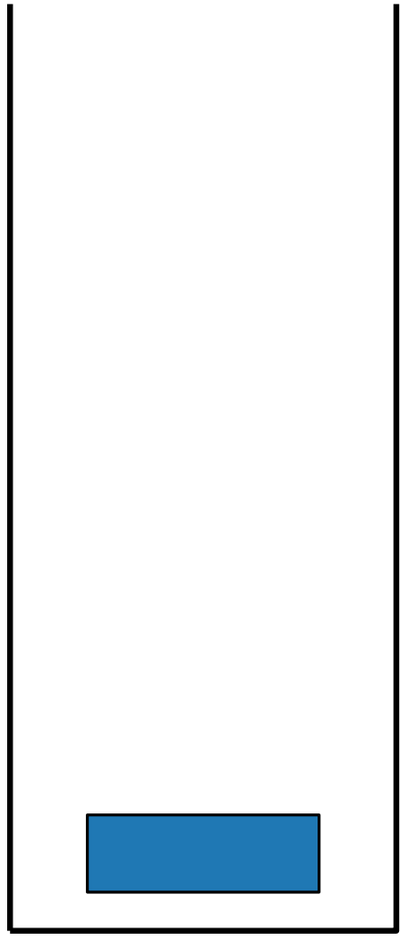


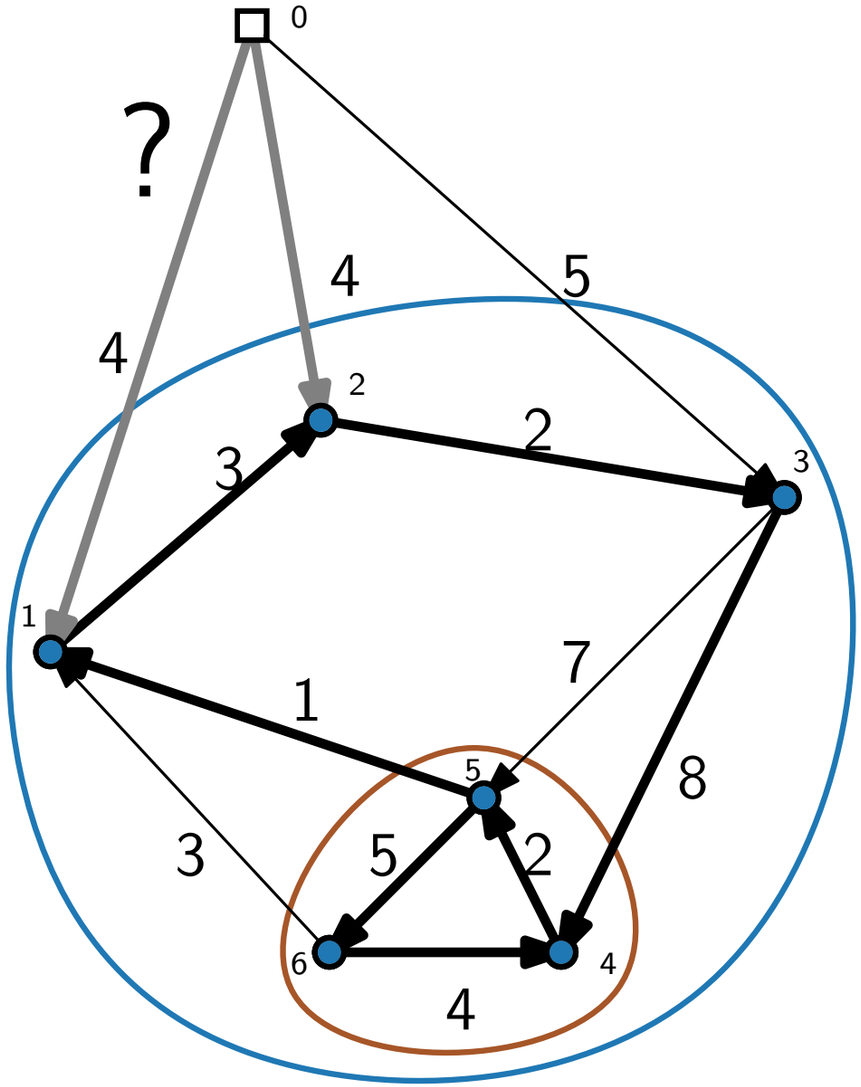
Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	besucht
c_0:	1
status:	besucht
c_0:	3
status:	besucht
c_0:	2
status:	besucht
c_0:	8
status:	besucht
c_0:	8
status:	besucht
c_0:	8

ast: stack



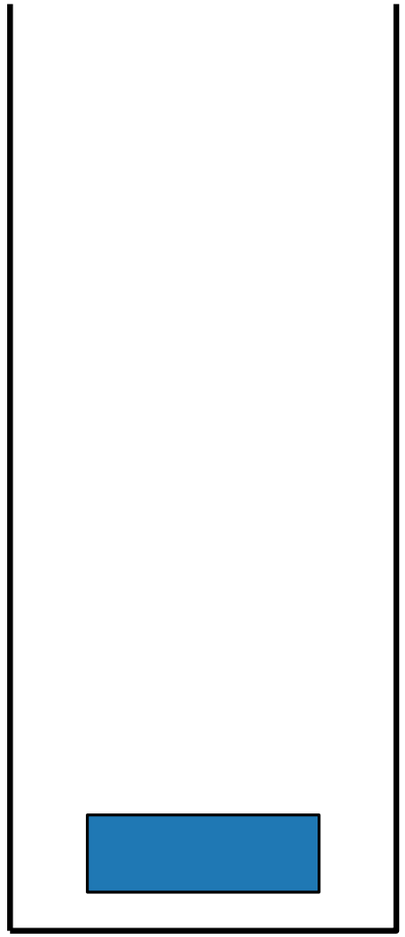


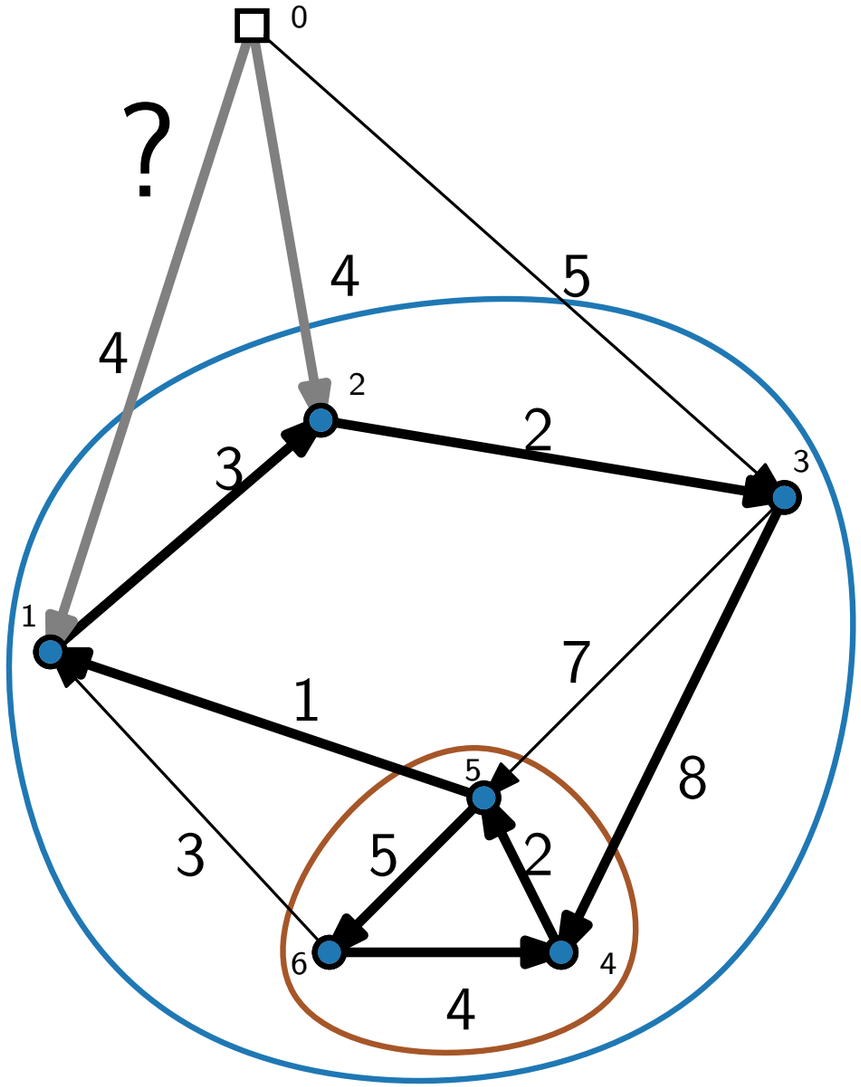
Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	besucht
c_0:	1
status:	besucht
c_0:	3
status:	besucht
c_0:	2
status:	besucht
c_0:	8
status:	besucht
c_0:	8
status:	besucht
c_0:	8

ast: stack



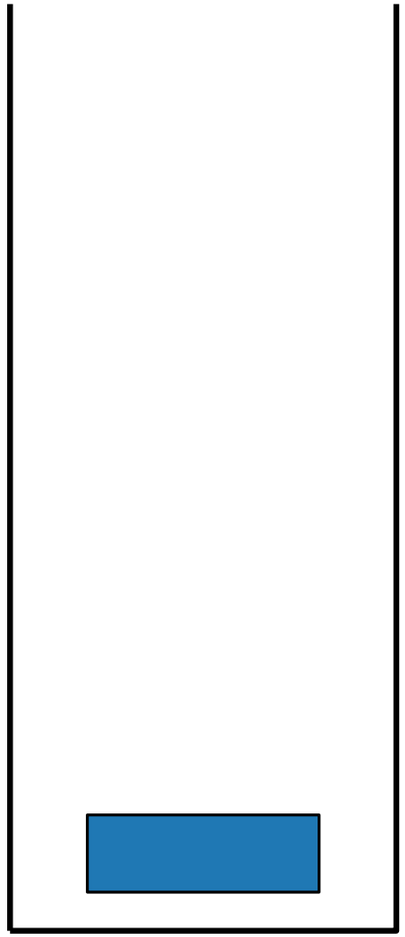


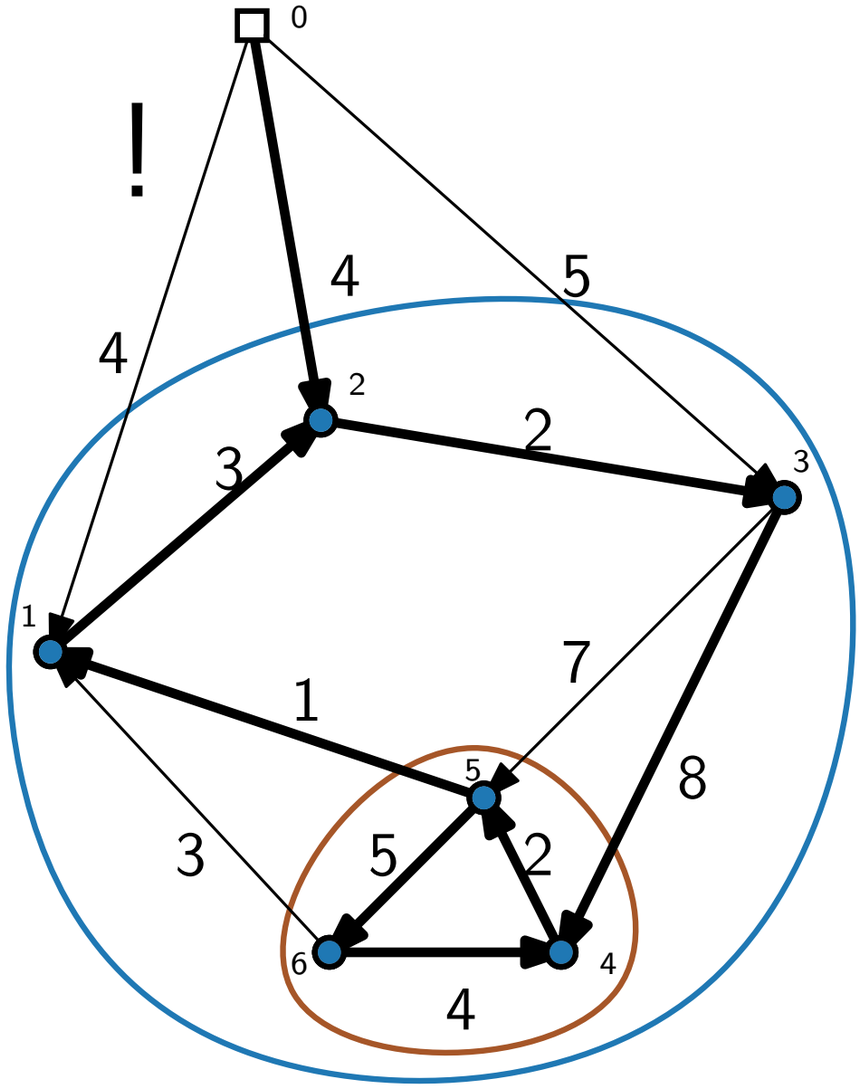
Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	besucht
c_0:	1
status:	besucht
c_0:	3
status:	besucht
c_0:	2
status:	besucht
c_0:	8
status:	besucht
c_0:	8
status:	besucht
c_0:	8

ast: stack



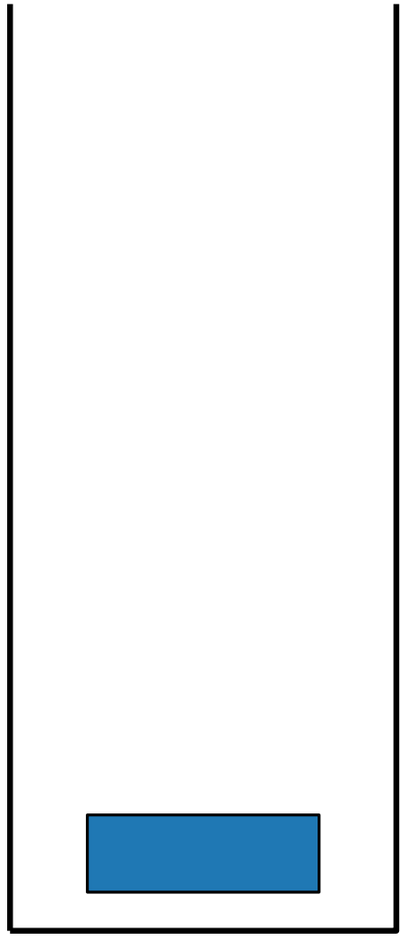


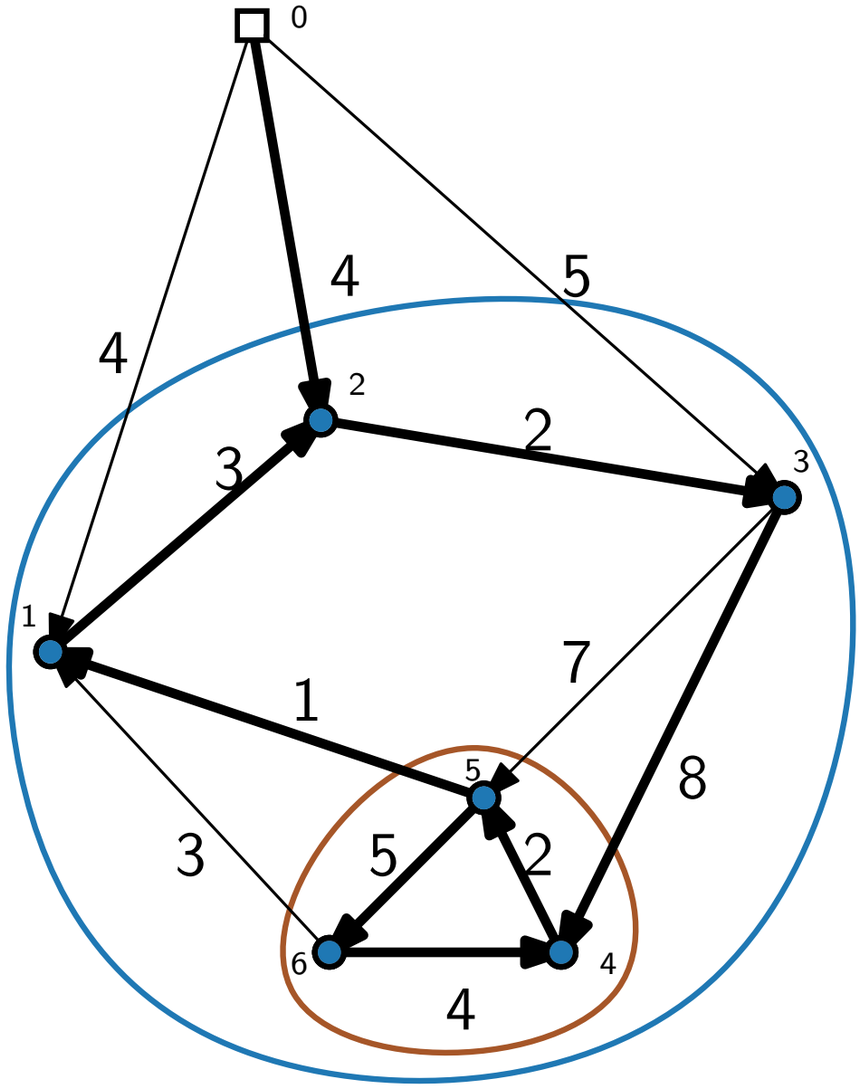
Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	besucht
c_0:	1
status:	besucht
c_0:	3
status:	besucht
c_0:	2
status:	besucht
c_0:	8
status:	besucht
c_0:	8
status:	besucht
c_0:	8

ast: stack



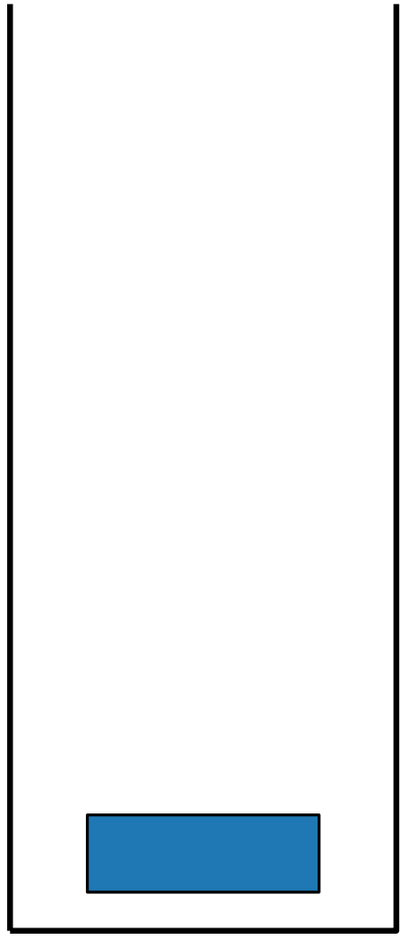


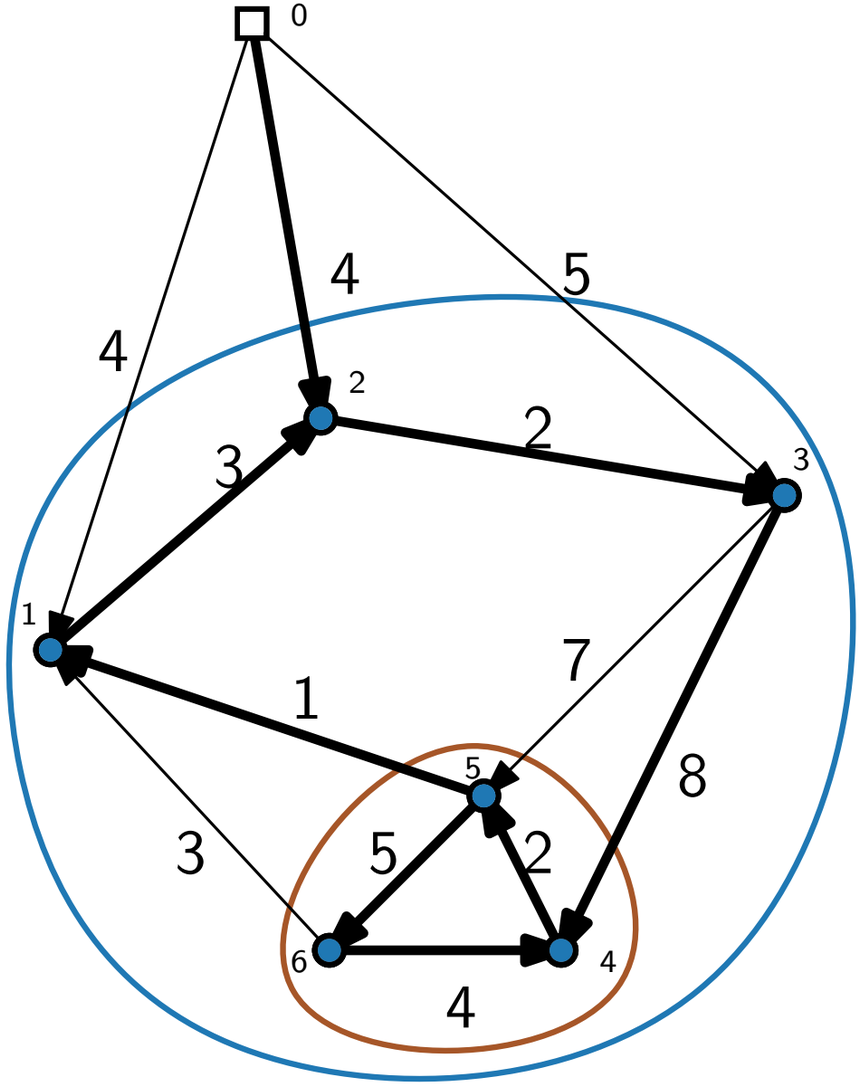
Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	besucht
c_0:	1+(4-3)
status:	besucht
c_0:	3
status:	besucht
c_0:	2
status:	besucht
c_0:	8
status:	besucht
c_0:	8
status:	besucht
c_0:	8

ast: stack



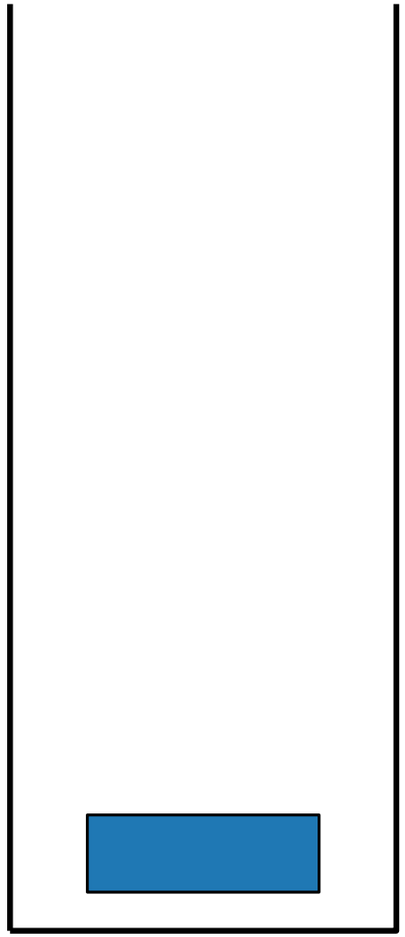


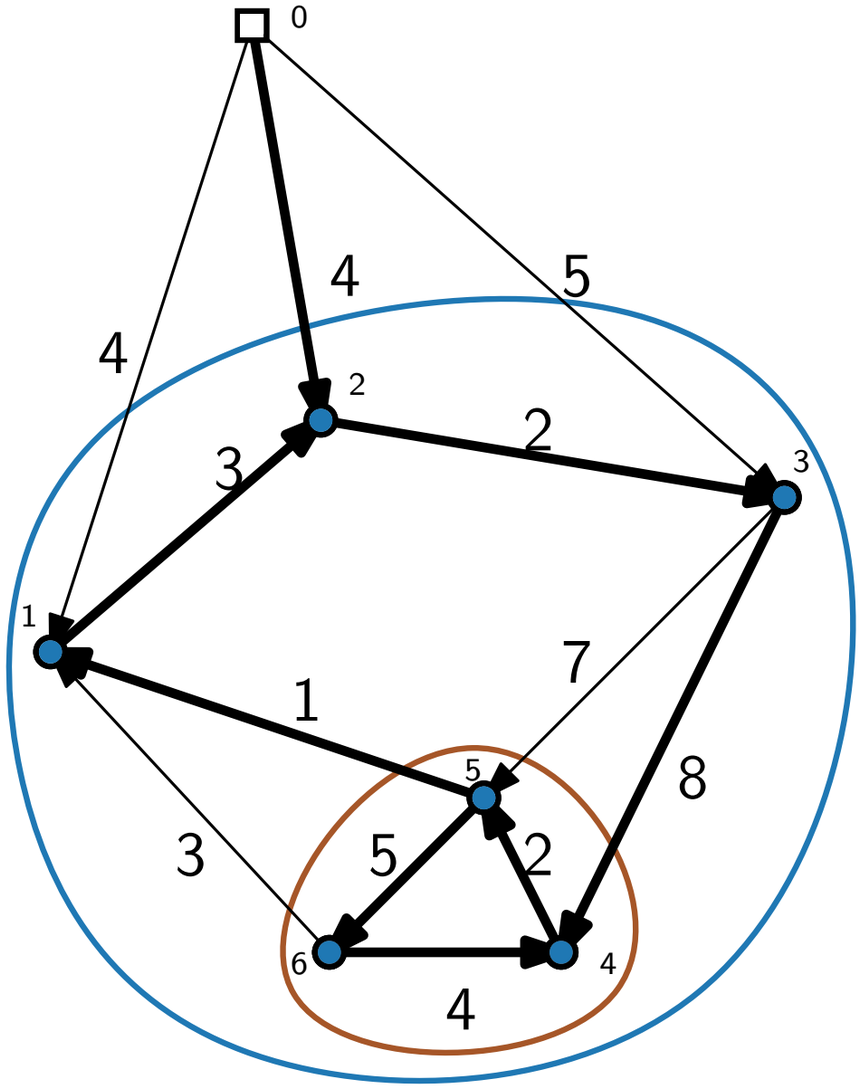
Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	besucht
c_0:	2
status:	besucht
c_0:	3
status:	besucht
c_0:	2
status:	besucht
c_0:	8
status:	besucht
c_0:	8
status:	besucht
c_0:	8

ast: stack



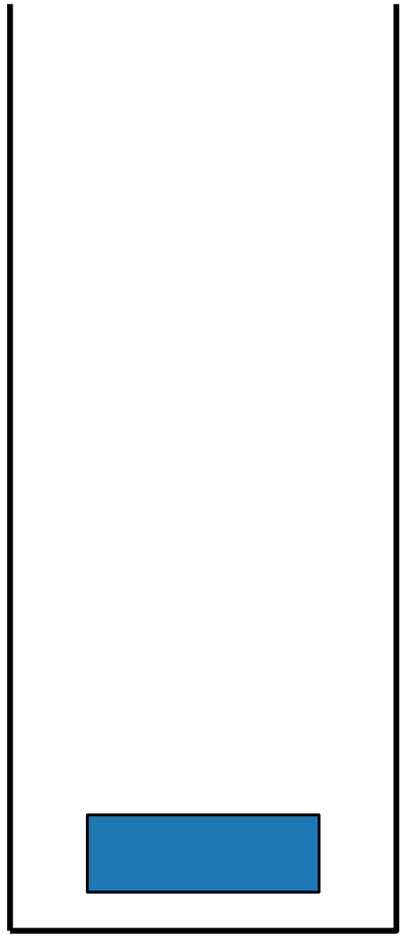


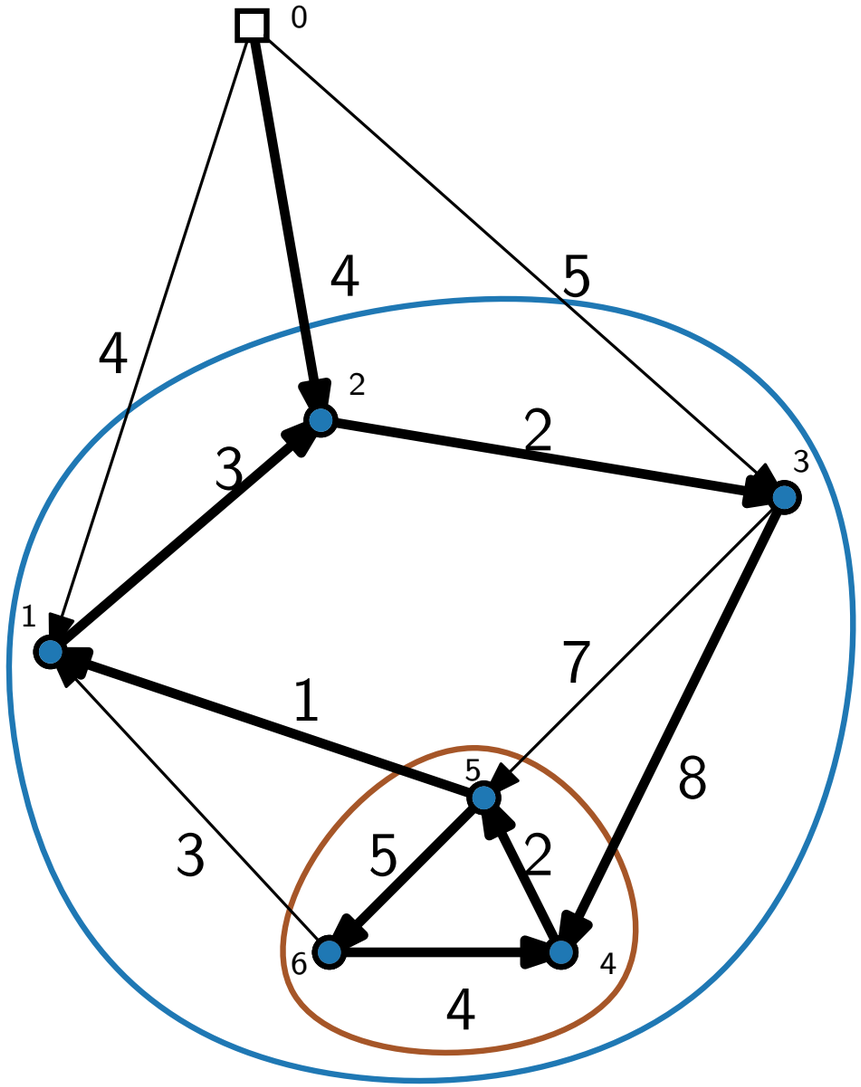
Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	besucht
c_0:	2
status:	besucht
c_0:	4
status:	besucht
c_0:	3
status:	besucht
c_0:	9
status:	besucht
c_0:	9
status:	besucht
c_0:	9

ast: stack



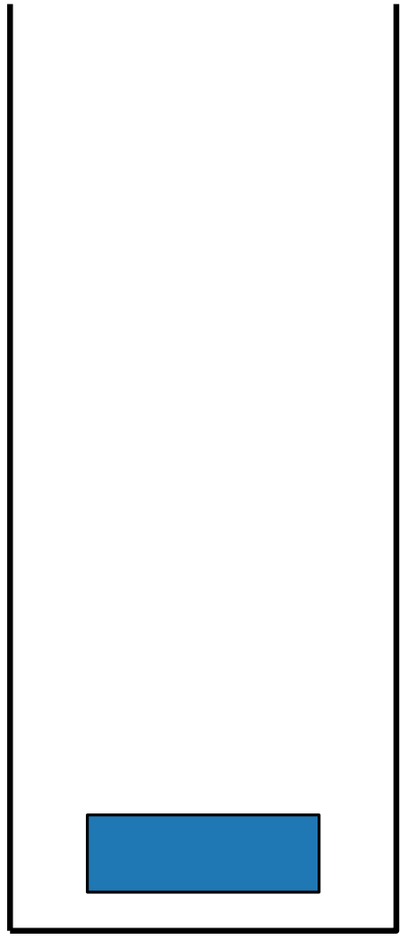


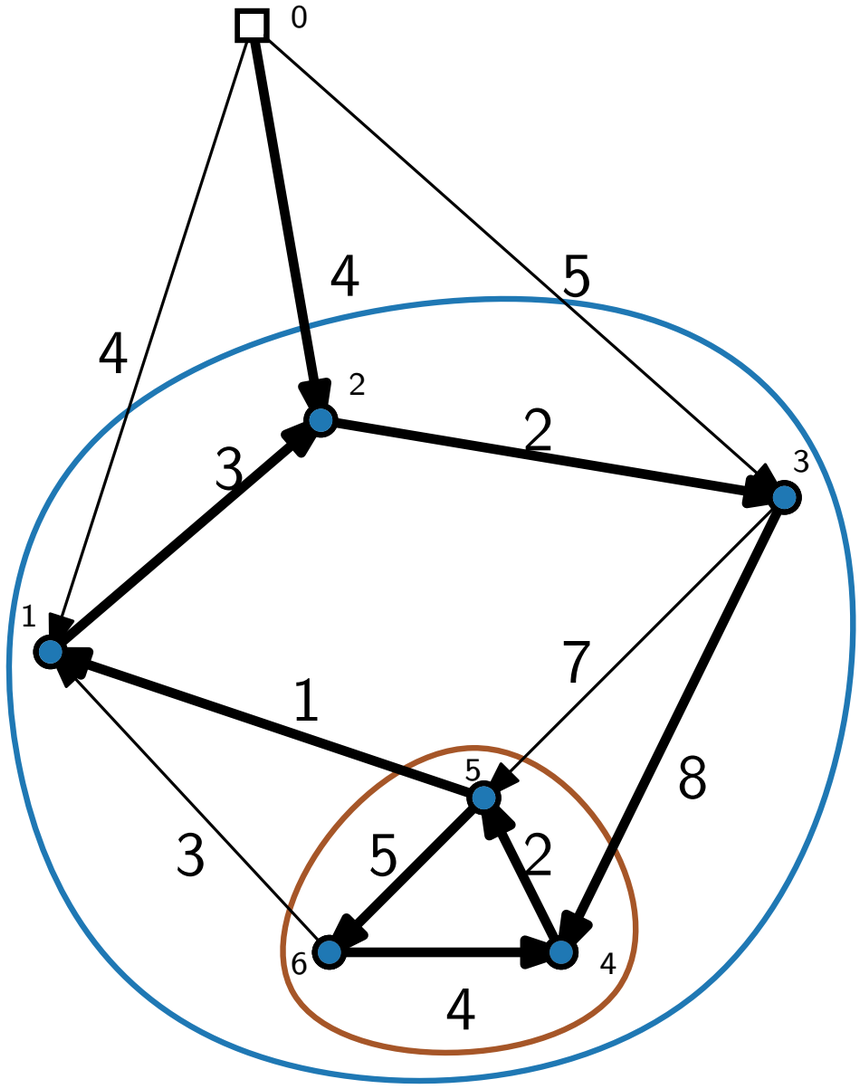
Edmonds Algorithmus – Implementierung

knoten: array

status:	gewurzelt
c_0:	0
status:	gewurzelt
c_0:	2
status:	gewurzelt
c_0:	4
status:	gewurzelt
c_0:	3
status:	gewurzelt
c_0:	9
status:	gewurzelt
c_0:	9
status:	gewurzelt
c_0:	9

ast: stack





Edmonds Algorithmus – Pseudocode

Edmonds(Shortcuts x, Shortcut-Zeit s, Gewichtsmatrix a)

```
while ∃ nicht gewurzeltes v do
    Stack ast = Stack(v)
    do
        (u, v') = billigsteKante(ast.top())
        ast.top().inEdge = (u, v')
        normalisiere(ast.top())
        if u besucht then
            kontrahiereZyklus()
        if u unbesucht then
            besuche(u)
            ast.push(u)
    while u nicht gewurzelt
    foreach u ∈ ast do
        wurzel(u)
        expandiere(u)
return    ?
```

Edmonds Algorithmus – Pseudocode

Edmonds(Shortcuts x, Shortcut-Zeit s, Gewichtsmatrix a)

```
while ∃ nicht gewurzeltes v do
    Stack ast = Stack(v)
    do
        (u, v') = billigsteKante(ast.top())
        ast.top().inEdge = (u, v')
        normalisiere(ast.top())
        if u besucht then
            kontrahiereZyklus()
        if u unbesucht then
            besuche(u)
            ast.push(u)
    while u nicht gewurzelt
    foreach u ∈ ast do
        wurzel(u)
        expandiere(u)
return    ?
```

Edmonds Algorithmus – Pseudocode

Edmonds(Shortcuts x, Shortcut-Zeit s, Gewichtsmatrix a)

```
while ∃ nicht gewurzeltes v do
    Stack ast = Stack(v)
    do
        (u, v') = billigsteKante(ast.top())
        ast.top().inEdge = (u, v')
        normalisiere(ast.top())
        if u besucht then
            kontrahiereZyklus()
        if u unbesucht then
            besuche(u)
            ast.push(u)
    while u nicht gewurzelt
    foreach u ∈ ast do
        wurzel(u)
        expandiere(u)
return    ?
```

1. Abgedeckte Items in $O(n)$ iterierbar.
2. Zugehörigkeit von Items in $O(\log n)$ überprüfbar.
3. Von einem zum nächsten Subknoten in $O(1)$ iterierbar

Edmonds Algorithmus – Pseudocode

Edmonds(Shortcuts x, Shortcut-Zeit s, Gewichtsmatrix a)

```
while ∃ nicht gewurzeltes v do
    Stack ast = Stack(v)
    do
        (u, v') = billigsteKante(ast.top())
        ast.top().inEdge = (u, v')
        normalisiere(ast.top())
        if u besucht then
            kontrahiereZyklus()
        if u unbesucht then
            besuche(u)
            ast.push(u)
    while u nicht gewurzelt
    foreach u ∈ ast do
        wurzel(u)
        expandiere(u)
return    ?
```

1. Abgedeckte Items in $O(n)$ iterierbar.
2. Zugehörigkeit von Items in $O(\log n)$ überprüfbar.
3. Von einem zum nächsten Subknoten in $O(1)$ iterierbar

$$O(n \log n)$$

Edmonds Algorithmus – Pseudocode

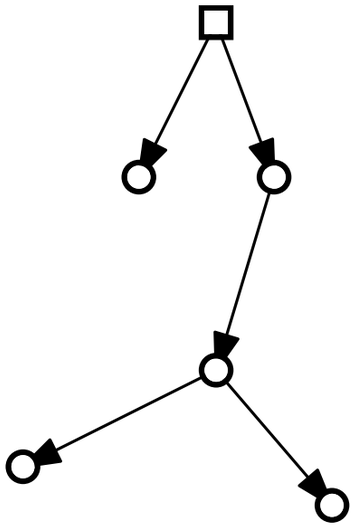
Edmonds(Shortcuts x, Shortcut-Zeit s, Gewichtsmatrix a)

```
while ∃ nicht gewurzeltes v do
    Stack ast = Stack(v)
    do
        (u, v') = billigsteKante(ast.top())
        ast.top().inEdge = (u, v')
        normalisiere(ast.top())
        if u besucht then
            kontrahiereZyklus()
        if u unbesucht then
            besuche(u)
            ast.push(u)
    while u nicht gewurzelt
    foreach u ∈ ast do
        wurzel(u)
        expandiere(u)
return    ?
```

Edmonds Algorithmus – Pseudocode

Edmonds(Shortcuts x, Shortcut-Zeit s, Gewichtsmatrix a)

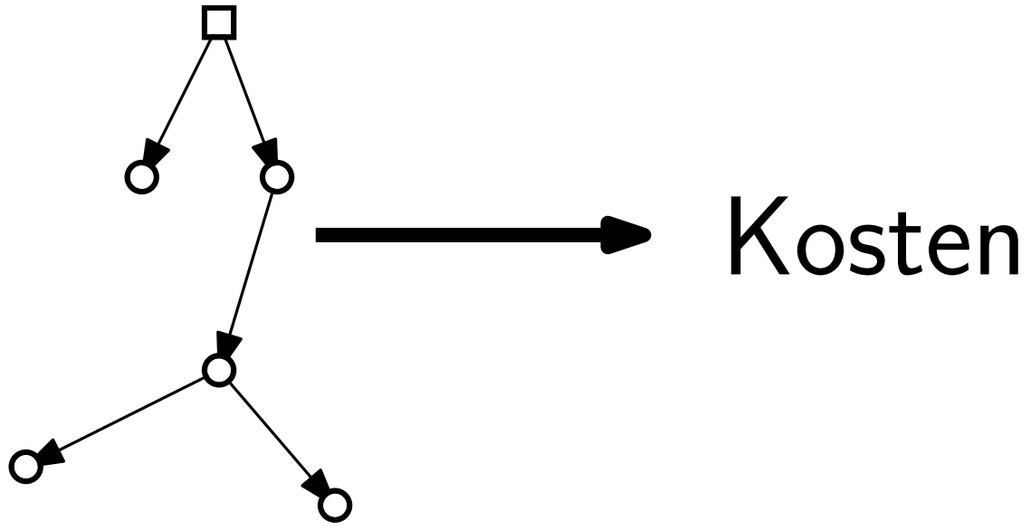
```
while ∃ nicht gewurzeltes v do
    Stack ast = Stack(v)
    do
        (u, v') = billigsteKante(ast.top())
        ast.top().inEdge = (u, v')
        normalisiere(ast.top())
        if u besucht then
            kontrahiereZyklus()
        if u unbesucht then
            besuche(u)
            ast.push(u)
    while u nicht gewurzelt
    foreach u ∈ ast do
        wurzel(u)
        expandiere(u)
return    ?
```



Edmonds Algorithmus – Pseudocode

Edmonds(Shortcuts x, Shortcut-Zeit s, Gewichtsmatrix a)

```
while ∃ nicht gewurzeltes v do
    Stack ast = Stack(v)
    do
        (u, v') = billigsteKante(ast.top())
        ast.top().inEdge = (u, v')
        normalisiere(ast.top())
        if u besucht then
            kontrahiereZyklus()
        if u unbesucht then
            besuche(u)
            ast.push(u)
    while u nicht gewurzelt
    foreach u ∈ ast do
        wurzel(u)
        expandiere(u)
return    ?
```



Edmonds Algorithmus – Pseudocode

Edmonds(Shortcuts x, Shortcut-Zeit s, Gewichtsmatrix a)

```
while ∃ nicht gewurzeltes v do
    Stack ast = Stack(v)
    do
        (u, v') = billigsteKante(ast.top())        } O(n log n)
        ast.top().inEdge = (u, v')
        normalisiere(ast.top())
        if u besucht then
            kontrahiereZyklus()
        if u unbesucht then
            besuche(u)
            ast.push(u)
    while u nicht gewurzelt
    foreach u ∈ ast do
        wurzel(u)
        expandiere(u)
return    ?
```

Edmonds Algorithmus – Pseudocode

Edmonds(Shortcuts x, Shortcut-Zeit s, Gewichtsmatrix a)

```
while ∃ nicht gewurzeltes v do
    Stack ast = Stack(v)
    do
        (u, v') = billigsteKante(ast.top())        } O(n log n)
        ast.top().inEdge = (u, v')                 } O(1)
        normalisiere(ast.top())
        if u besucht then
            kontrahiereZyklus()
        if u unbesucht then
            besuche(u)
            ast.push(u)
    while u nicht gewurzelt
    foreach u ∈ ast do
        wurzel(u)
        expandiere(u)
return    ?
```

Edmonds Algorithmus – Pseudocode

Edmonds(Shortcuts x, Shortcut-Zeit s, Gewichtsmatrix a)

```
while ∃ nicht gewurzeltes v do
    Stack ast = Stack(v)
    do
        (u, v') = billigsteKante(ast.top())        } O(n log n)
        ast.top().inEdge = (u, v')                 } O(1)
        normalisiere(ast.top())                    } O(n)
        if u besucht then
            kontrahiereZyklus()
        if u unbesucht then
            besuche(u)
            ast.push(u)
    while u nicht gewurzelt
    foreach u ∈ ast do
        wurzel(u)
        expandiere(u)
return    ?
```

Edmonds Algorithmus – Pseudocode

Edmonds(Shortcuts x, Shortcut-Zeit s, Gewichtsmatrix a)

```
while ∃ nicht gewurzeltes v do
    Stack ast = Stack(v)
    do
        (u, v') = billigsteKante(ast.top())        } O(n log n)
        ast.top().inEdge = (u, v')                 } O(1)
        normalisiere(ast.top())                    } O(n)
        if u besucht then
            kontrahiereZyklus()                    } O(n log n)
        if u unbesucht then
            besuche(u)
            ast.push(u)
    while u nicht gewurzelt
    foreach u ∈ ast do
        wurzel(u)
        expandiere(u)
return    ?
```

Edmonds Algorithmus – Pseudocode

Edmonds(Shortcuts x, Shortcut-Zeit s, Gewichtsmatrix a)

```
while ∃ nicht gewurzeltes v do                                  }O(n log n)
    Stack ast = Stack(v)
    do
        (u, v') = billigsteKante(ast.top())                     }O(n log n)
        ast.top().inEdge = (u, v')                              }O(1)
        normalisiere(ast.top())                                 }O(n)
        if u besucht then
            kontrahiereZyklus()                                 }O(n log n)
        if u unbesucht then
            besuche(u)                                          }
            ast.push(u)                                         } O(1)
    while u nicht gewurzelt
    foreach u ∈ ast do
        wurzel(u)
        expandiere(u)
return    ?
```

Edmonds Algorithmus – Pseudocode

Edmonds(Shortcuts x, Shortcut-Zeit s, Gewichtsmatrix a)

```
while ∃ nicht gewurzeltes v do
    Stack ast = Stack(v)
    do
        (u, v') = billigsteKante(ast.top())        } O(n log n)
        ast.top().inEdge = (u, v')                 } O(1)
        normalisiere(ast.top())                    } O(n)
        if u besucht then
            kontrahiereZyklus()                    } O(n log n)
        if u unbesucht then
            besuche(u)                             } O(1)
            ast.push(u)
    while u nicht gewurzelt
    foreach u ∈ ast do
        wurzel(u)
        expandiere(u)
return    ?
```

} Insgesamt $O(n)$ Durchläufe

Edmonds Algorithmus – Pseudocode

Edmonds(Shortcuts x, Shortcut-Zeit s, Gewichtsmatrix a)

```
while ∃ nicht gewurzeltes v do
    Stack ast = Stack(v)
    do
        (u, v') = billigsteKante(ast.top())          } O(n log n)
        ast.top().inEdge = (u, v')                   } O(1)
        normalisiere(ast.top())                      } O(n)
        if u besucht then
            kontrahiereZyklus()                      } O(n log n)
        if u unbesucht then
            besuche(u)                               } O(1)
            ast.push(u)
    while u nicht gewurzelt                          } Insgesamt O(n) Durchläufe
    foreach u ∈ ast do
        wurzel(u)                                    } Insgesamt O(n log n)
        expandiere(u)
return    ?
```

Edmonds Algorithmus – Pseudocode

Edmonds(Shortcuts x, Shortcut-Zeit s, Gewichtsmatrix a)

```
while ∃ nicht gewurzeltes v do
    Stack ast = Stack(v)
    do
        (u, v') = billigsteKante(ast.top())        } O(n log n)
        ast.top().inEdge = (u, v')                 } O(1)
        normalisiere(ast.top())                    } O(n)
        if u besucht then
            kontrahiereZyklus()                    } O(n log n)
        if u unbesucht then
            besuche(u)                             } O(1)
            ast.push(u)
    while u nicht gewurzelt
    foreach u ∈ ast do
        wurzel(u)
        expandiere(u)
return ?
```

Insgesamt $O(n)$ Durchläufe

Insgesamt $O(n \log n)$

$\Rightarrow$ Laufzeit $O(n^2 \log n)$

Edmonds Algorithmus – Pseudocode

Edmonds(Shortcuts x, Shortcut-Zeit s, Gewichtsmatrix a)

```
while ∃ nicht gewurzeltes v do
    Stack ast = Stack(v)
    do
        (u, v') = billigsteKante(ast.top())        } O(n log n)
        ast.top().inEdge = (u, v')                 } O(1)
        normalisiere(ast.top())                    } O(n)
        if u besucht then
            kontrahiereZyklus()                    } O(n log n)
        if u unbesucht then
            besuche(u)                             } O(1)
            ast.push(u)
    while u nicht gewurzelt
    foreach u ∈ ast do
        wurzel(u)
        expandiere(u)
return    ?
```

Insgesamt $O(n)$ Durchläufe

Insgesamt $O(n \log n)$

$\Rightarrow$ Laufzeit $O(n^2 \log n)$ ✓

Kombinatorischer Ansatz

Kombinatorischer Ansatz

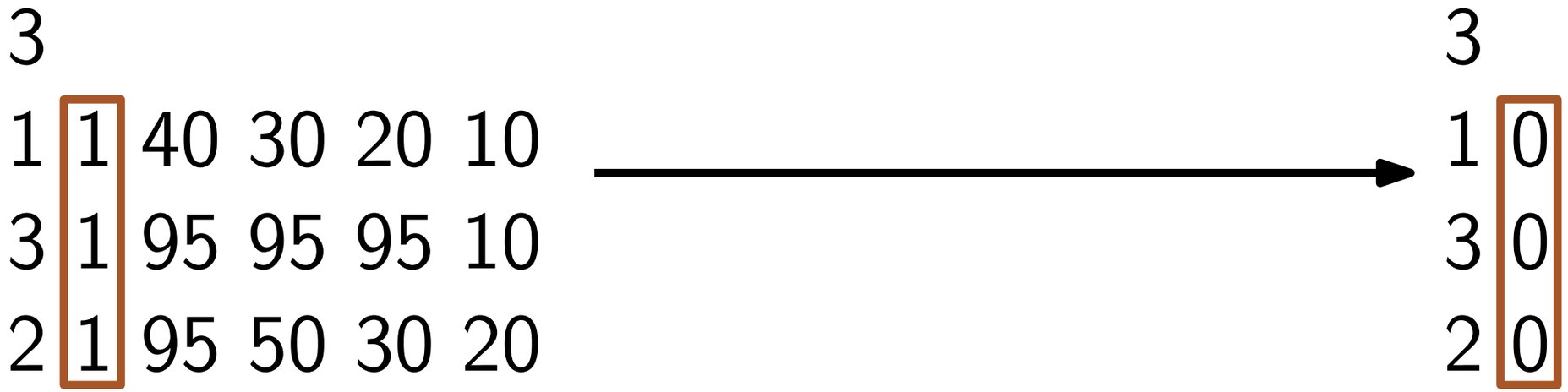
3

1 1 40 30 20 10

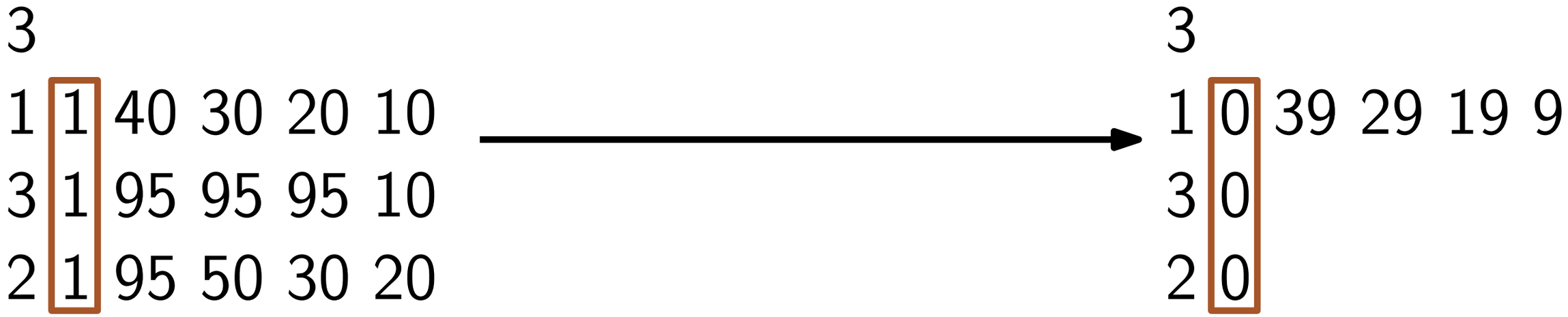
3 1 95 95 95 10

2 1 95 50 30 20

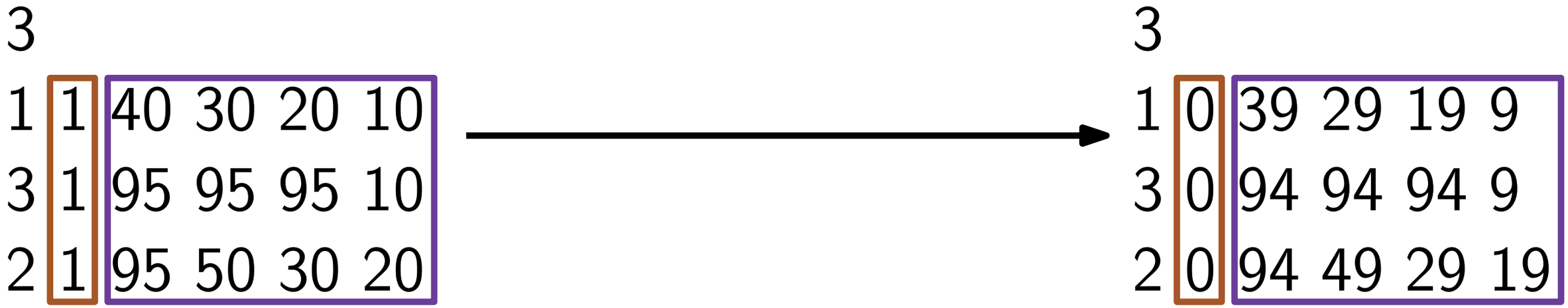
Kombinatorischer Ansatz



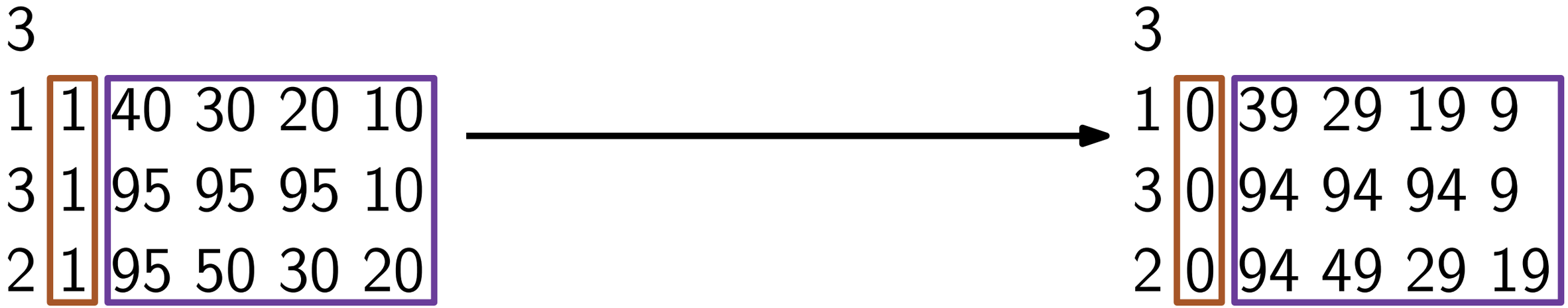
Kombinatorischer Ansatz



Kombinatorischer Ansatz

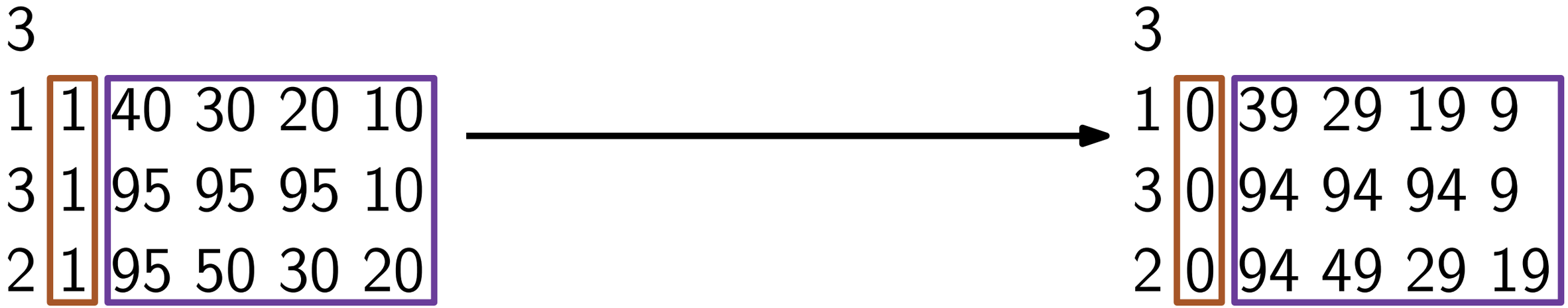


Kombinatorischer Ansatz



$$a_{i,j} \leftarrow a_{i,j} - s_j$$

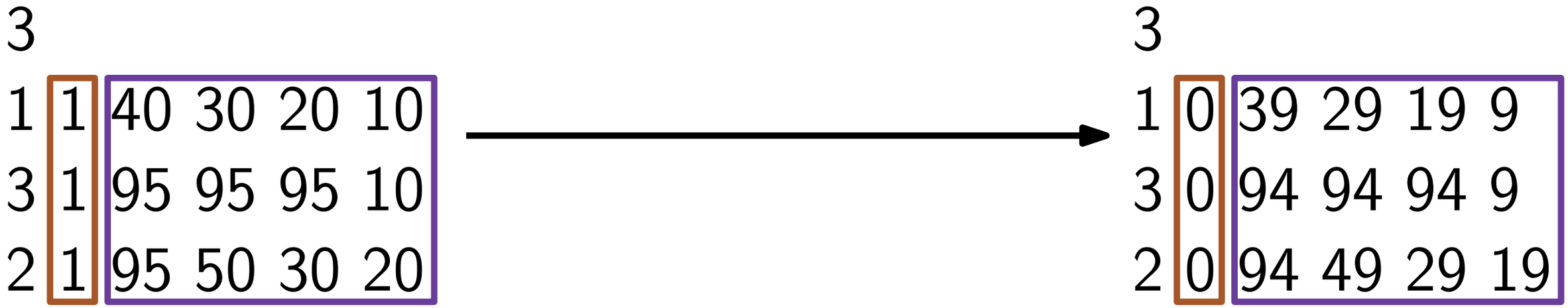
Kombinatorischer Ansatz



$$a_{i,j} \leftarrow a_{i,j} - s_j$$

$$s_j \leftarrow 0$$

Kombinatorischer Ansatz

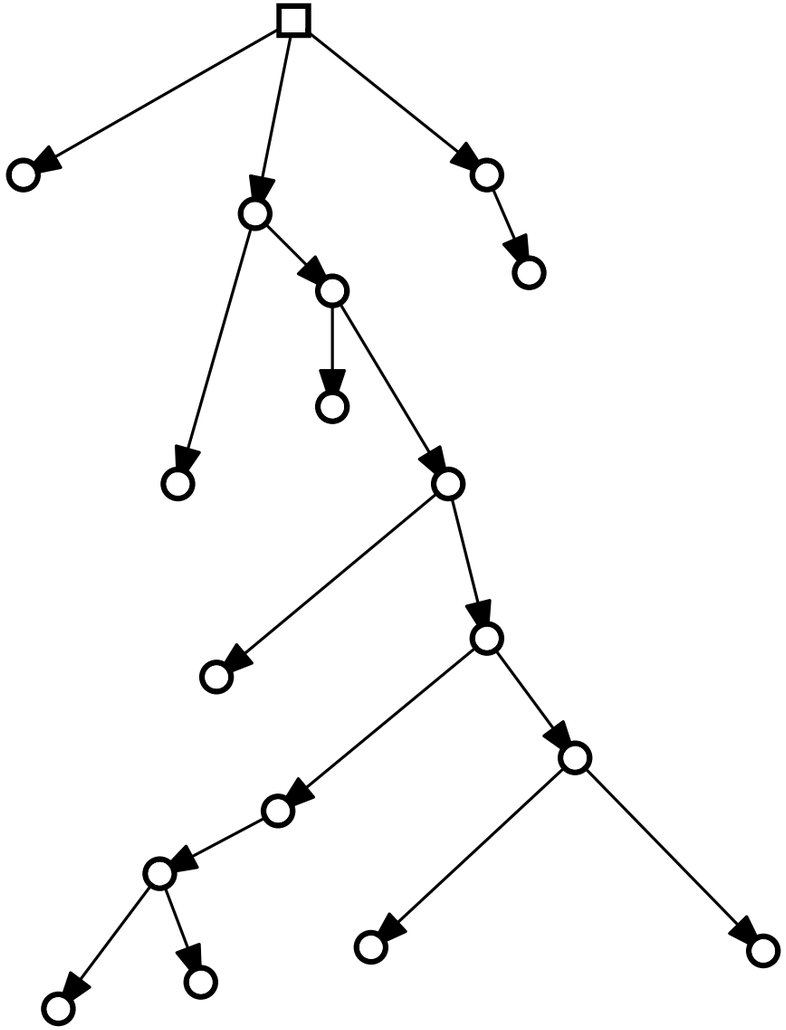


$$a_{i,j} \leftarrow a_{i,j} - s_j$$

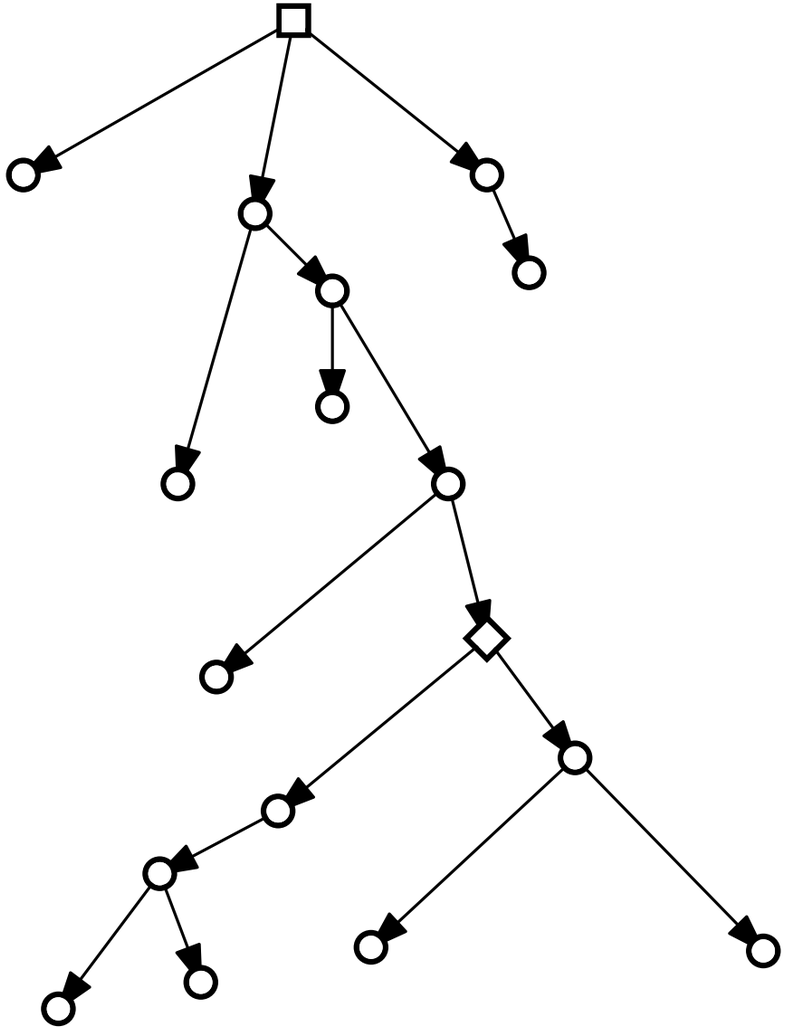
$$s_j \leftarrow 0$$

$$O(n^2)$$

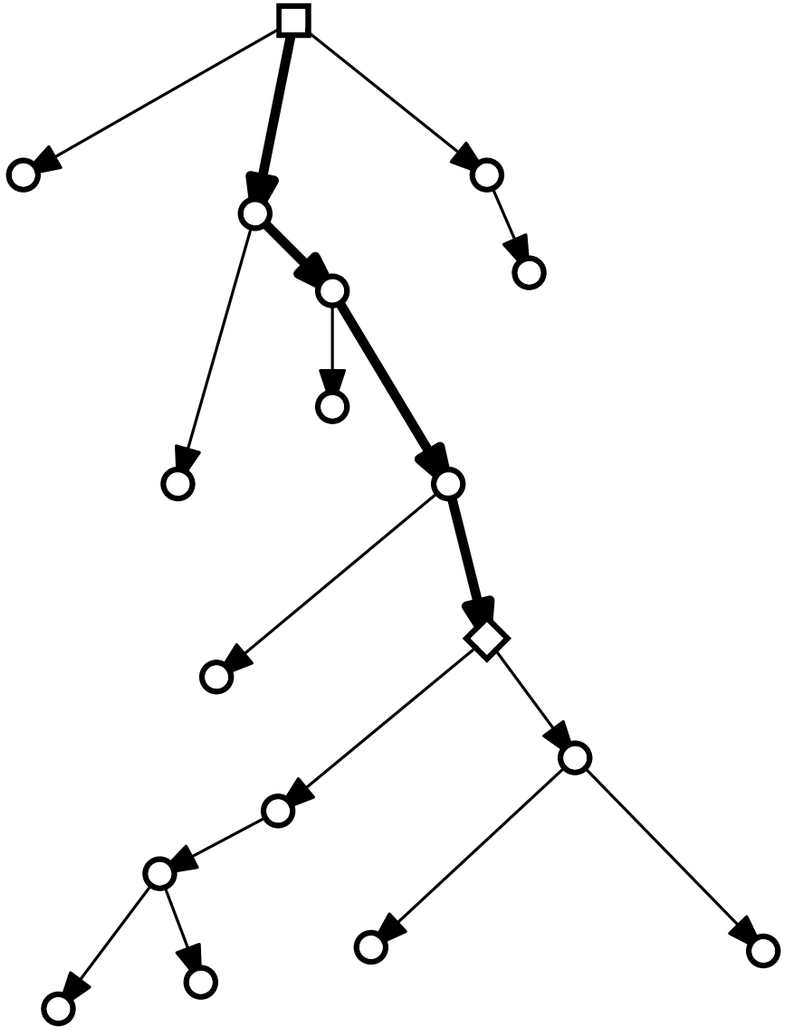
Eigenschaften optimaler Lösungen



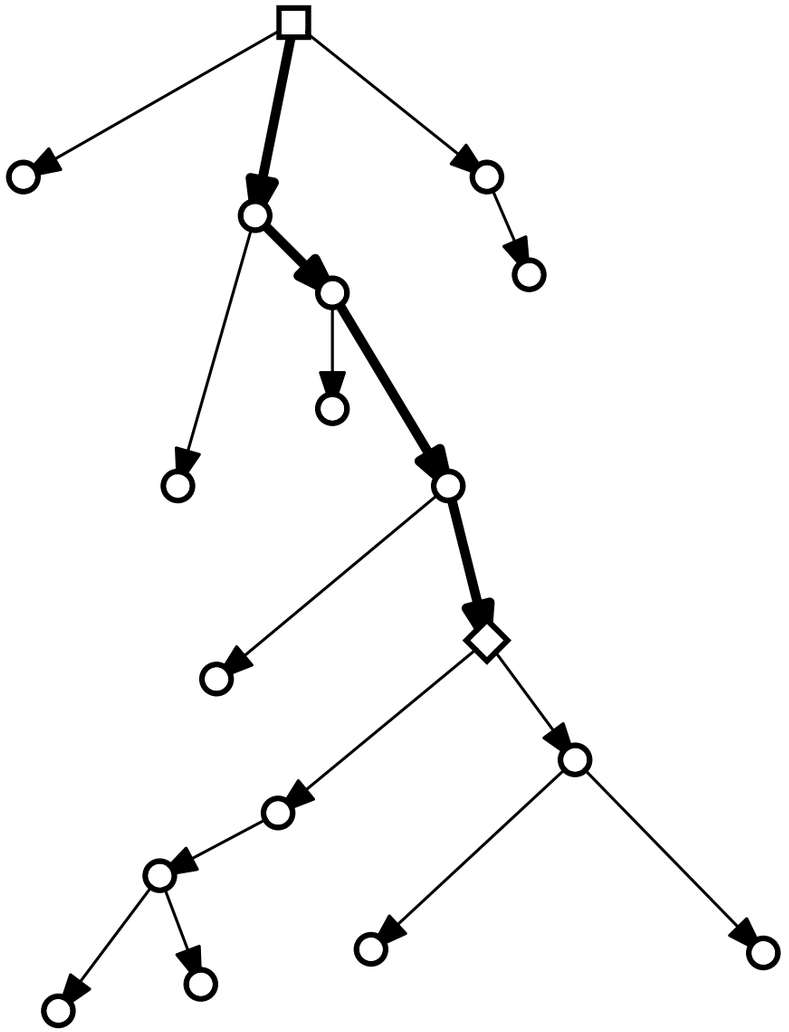
Eigenschaften optimaler Lösungen



Eigenschaften optimaler Lösungen



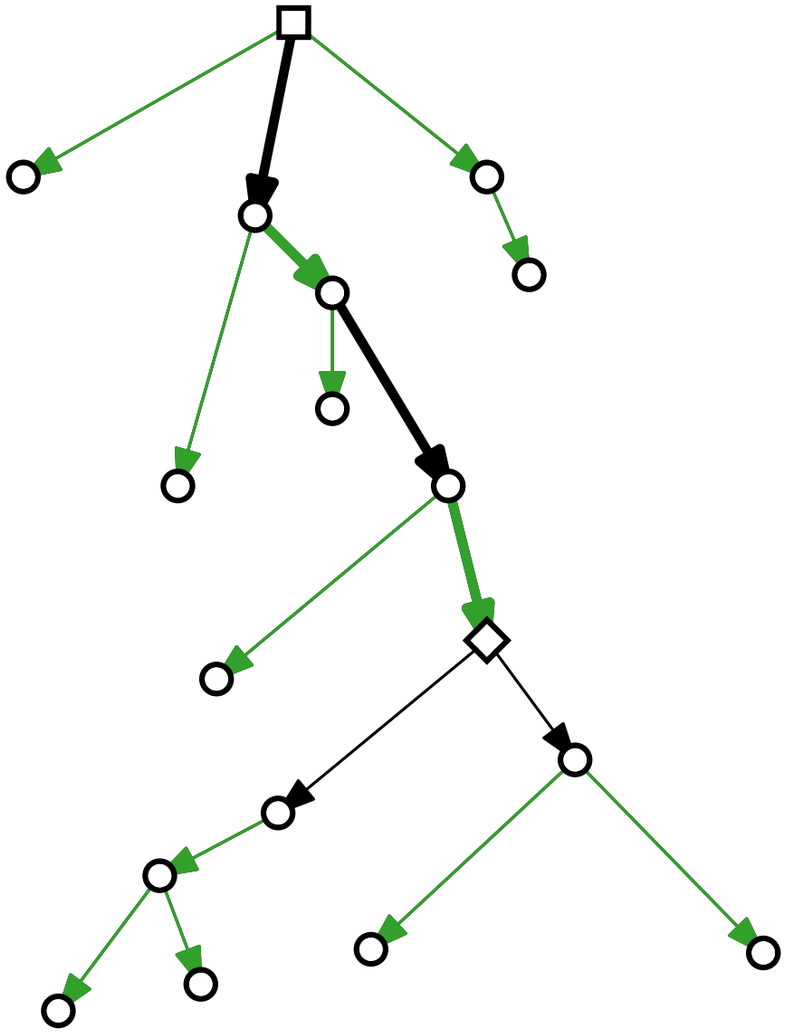
Eigenschaften optimaler Lösungen



Jede Kante ist entweder

Eine Abkürzung

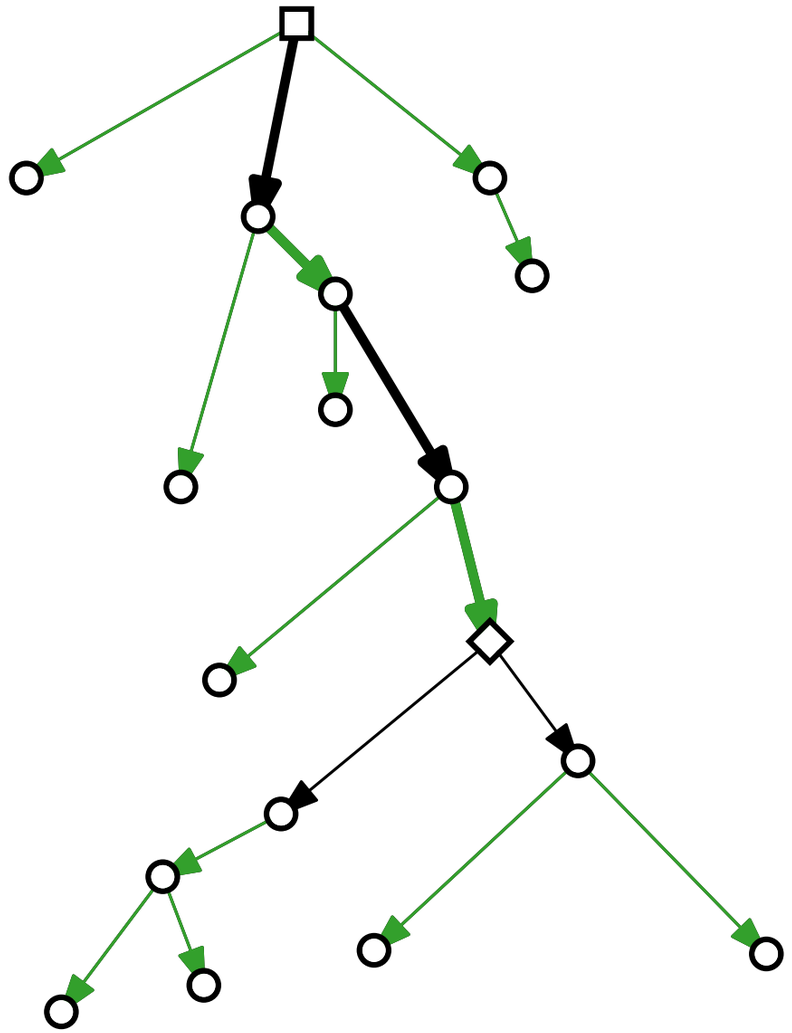
Eigenschaften optimaler Lösungen



Jede Kante ist entweder

Eine Abkürzung

Eigenschaften optimaler Lösungen

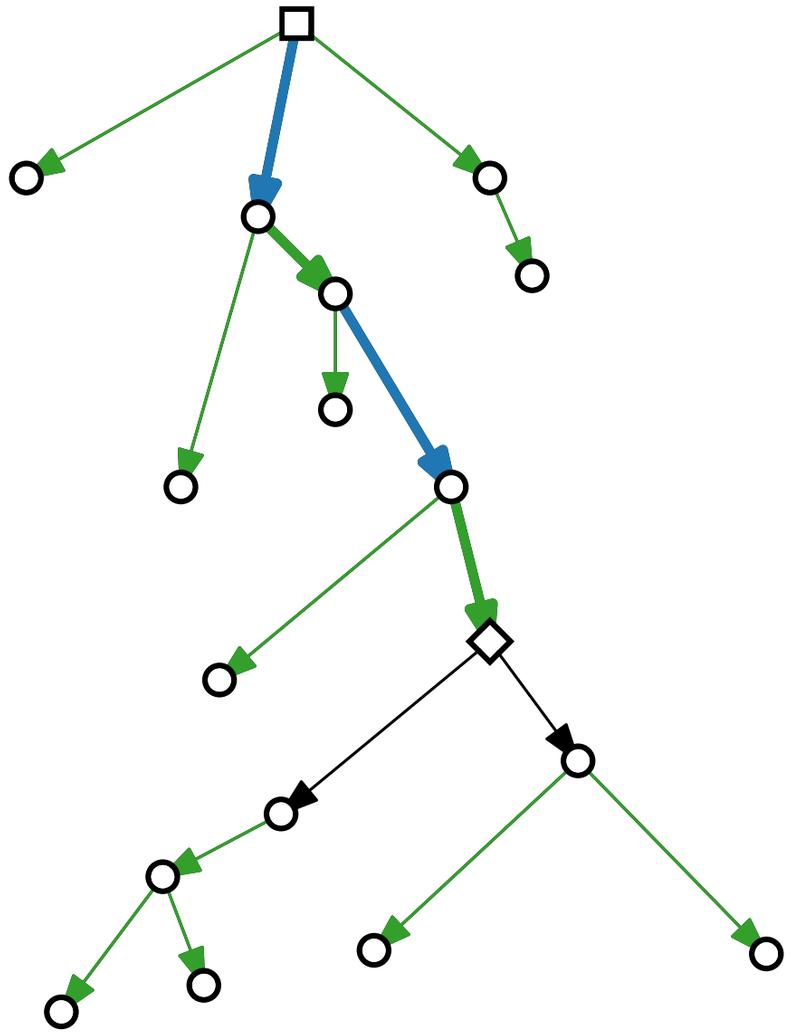


Jede Kante ist entweder

Eine Abkürzung

Auf dem Pfad von 0 zu n

Eigenschaften optimaler Lösungen

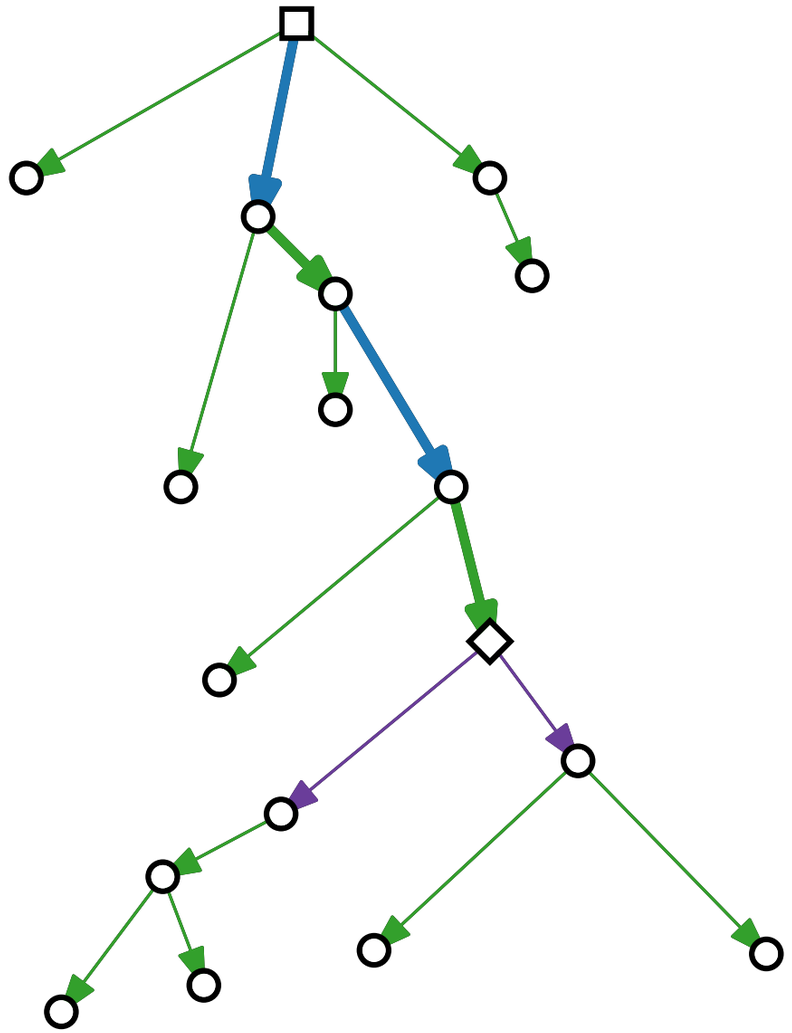


Jede Kante ist entweder

Eine Abkürzung

Auf dem Pfad von 0 zu n

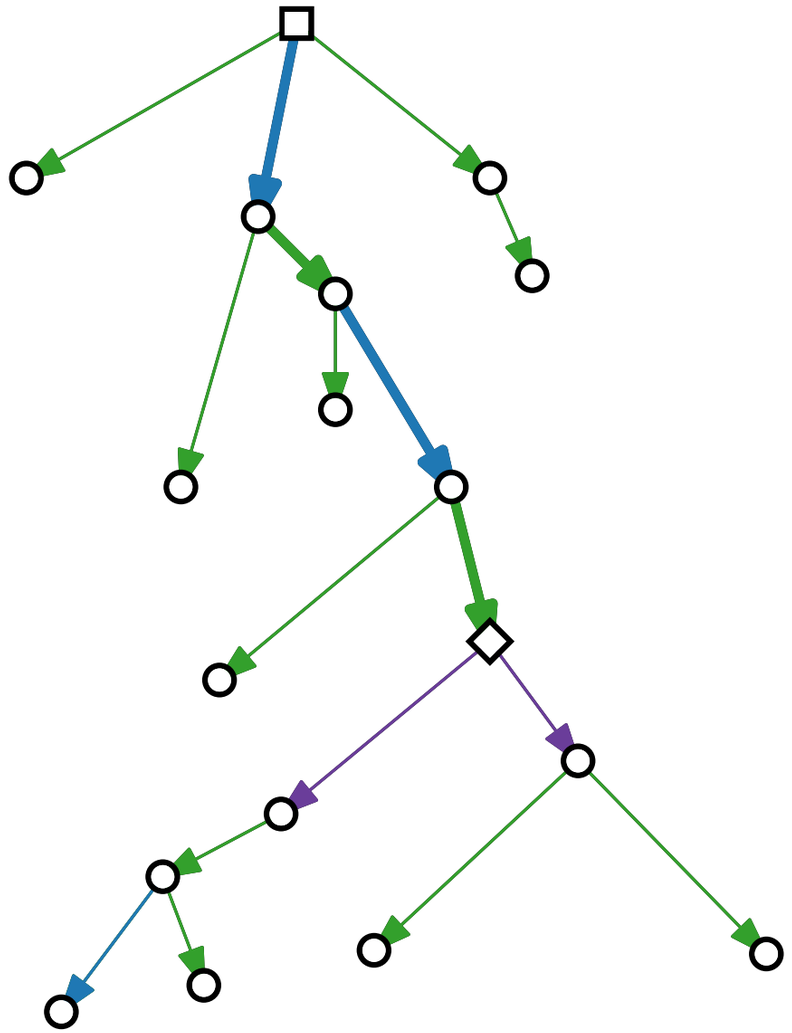
Eigenschaften optimaler Lösungen



Jede Kante ist entweder

- Eine Abkürzung
- Auf dem Pfad von 0 zu n
- Ausgehend von n

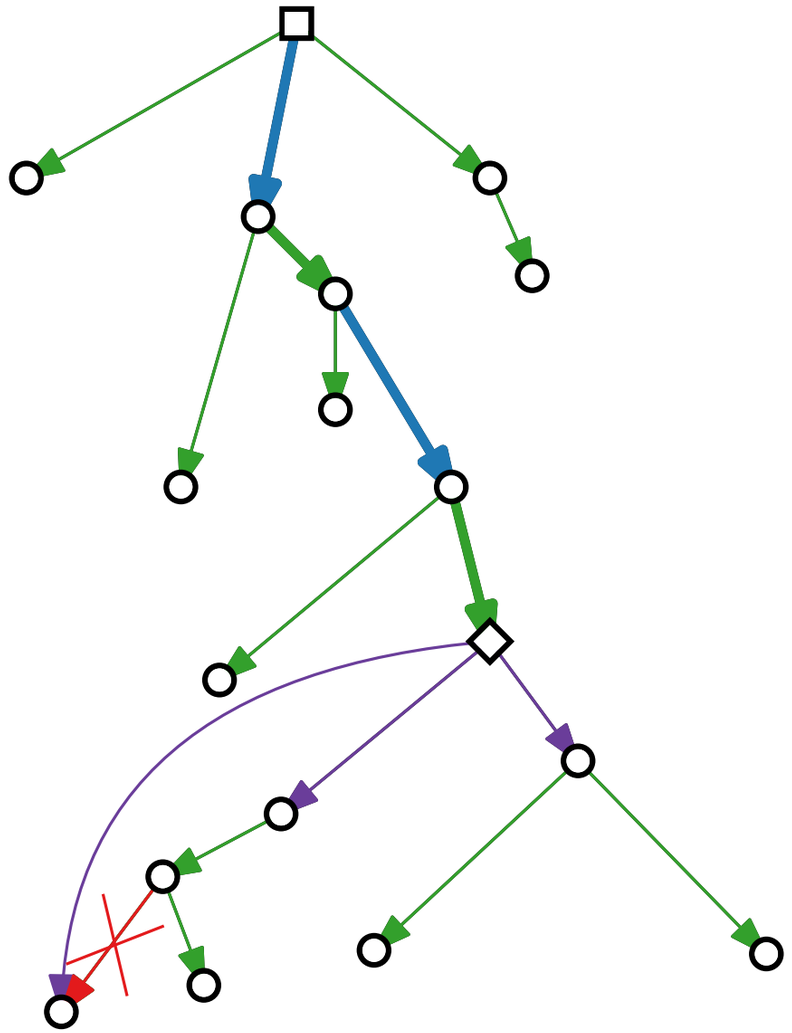
Eigenschaften optimaler Lösungen



Jede Kante ist entweder

- Eine Abkürzung
- Auf dem Pfad von 0 zu n
- Ausgehend von n

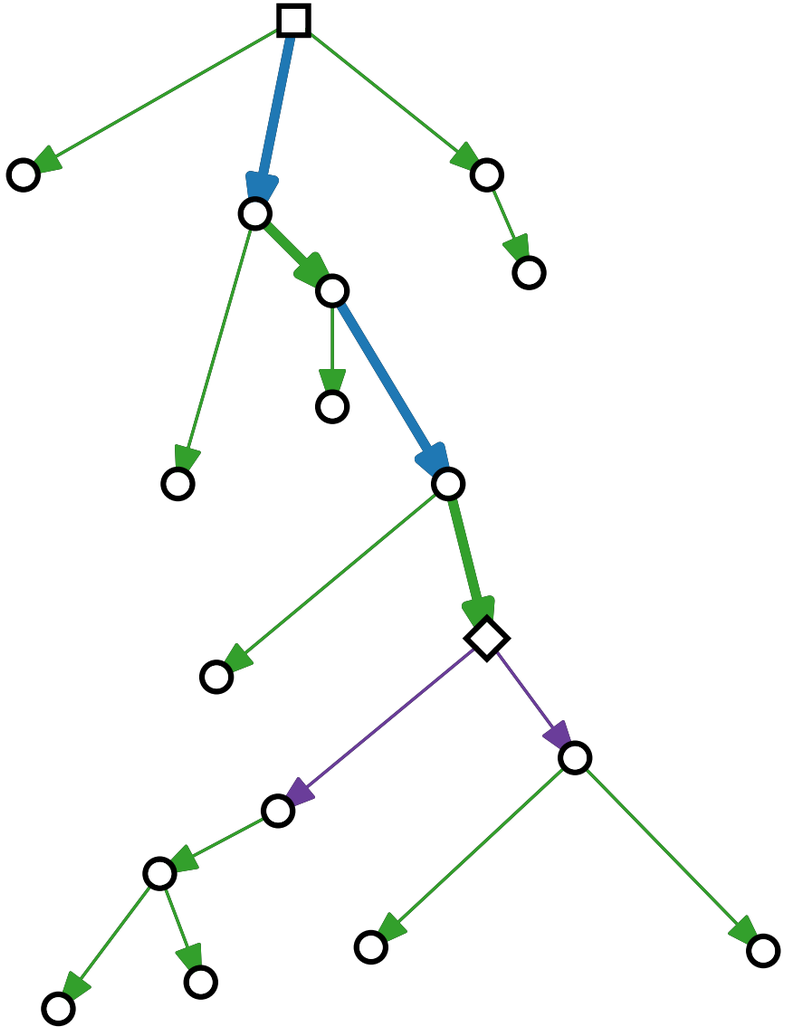
Eigenschaften optimaler Lösungen



Jede Kante ist entweder

- Eine Abkürzung
- Auf dem Pfad von 0 zu n
- Ausgehend von n

Eigenschaften optimaler Lösungen

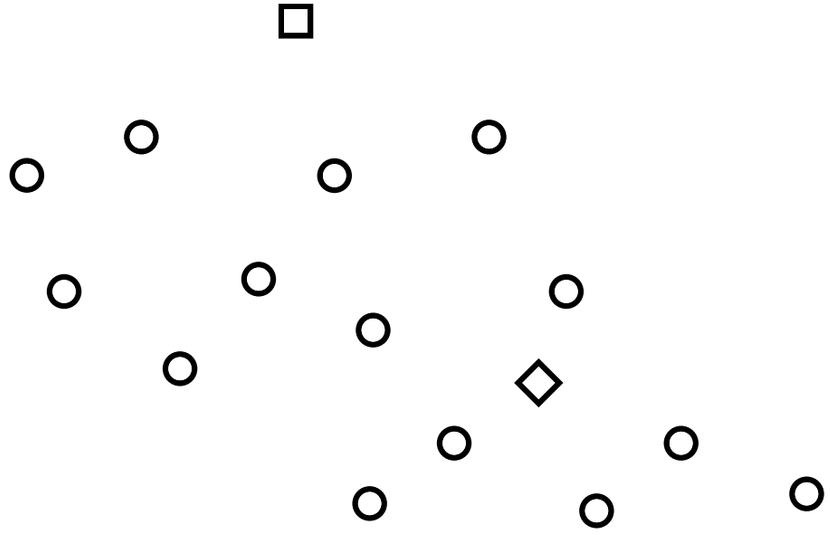


Jede Kante ist entweder

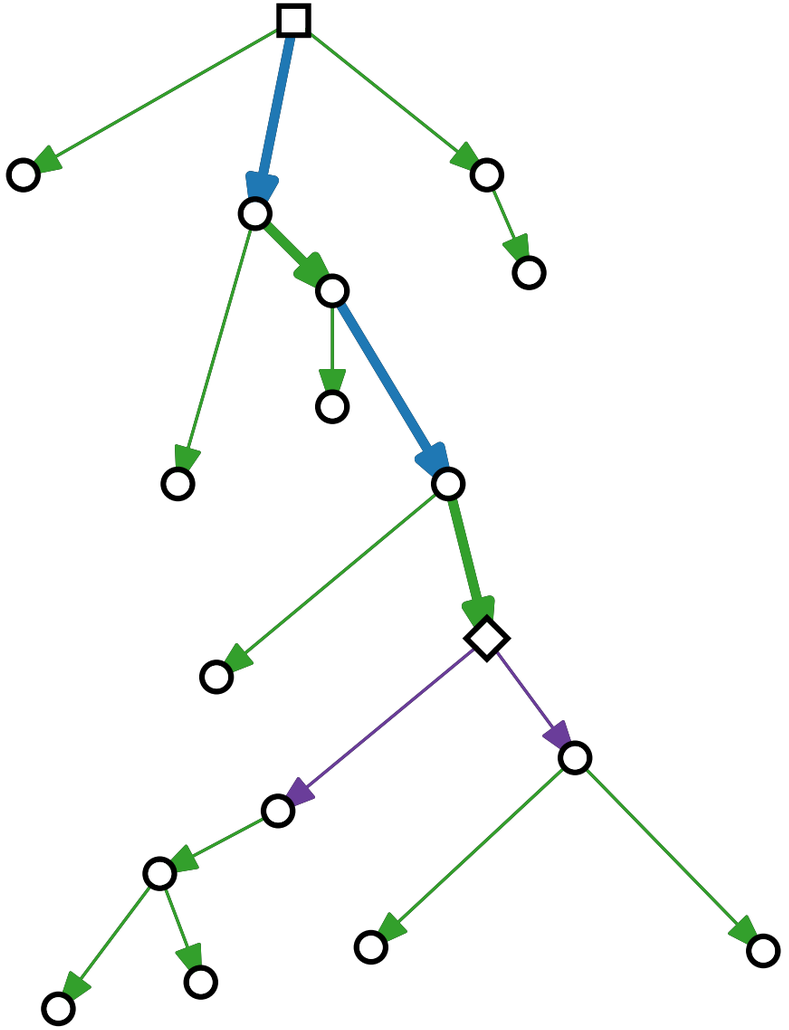
Eine Abkürzung

Auf dem Pfad von 0 zu n

Ausgehend von n



Eigenschaften optimaler Lösungen

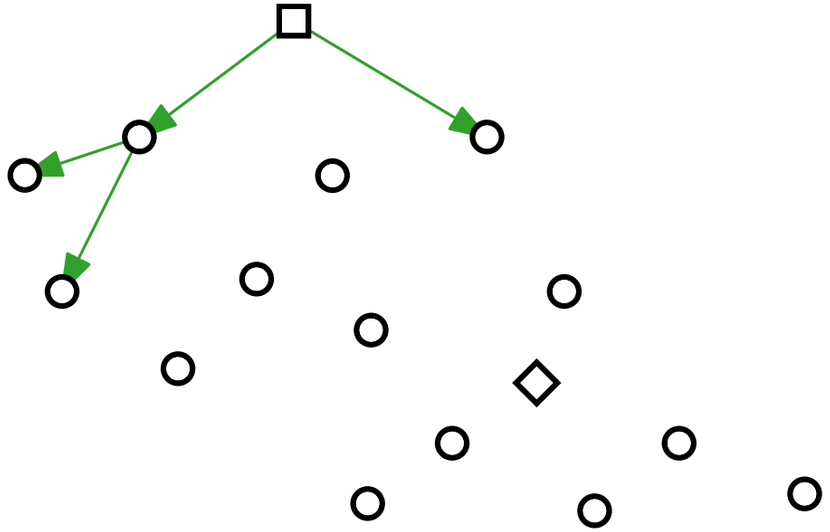


Jede Kante ist entweder

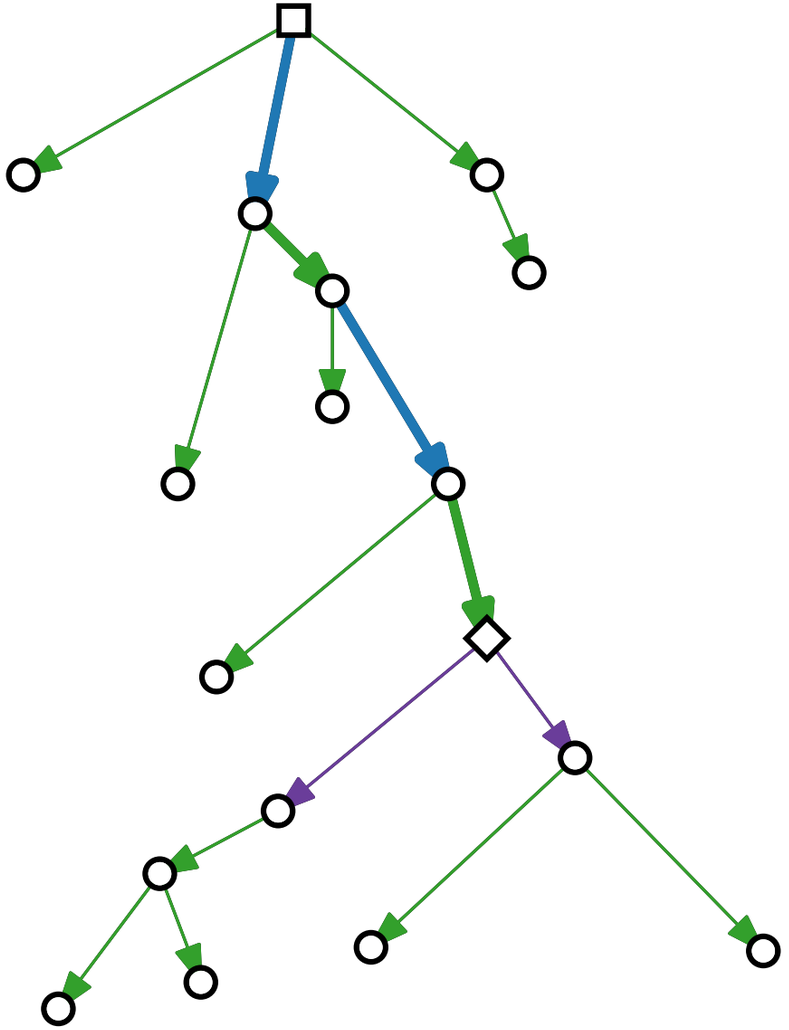
Eine Abkürzung

Auf dem Pfad von 0 zu n

Ausgehend von n



Eigenschaften optimaler Lösungen

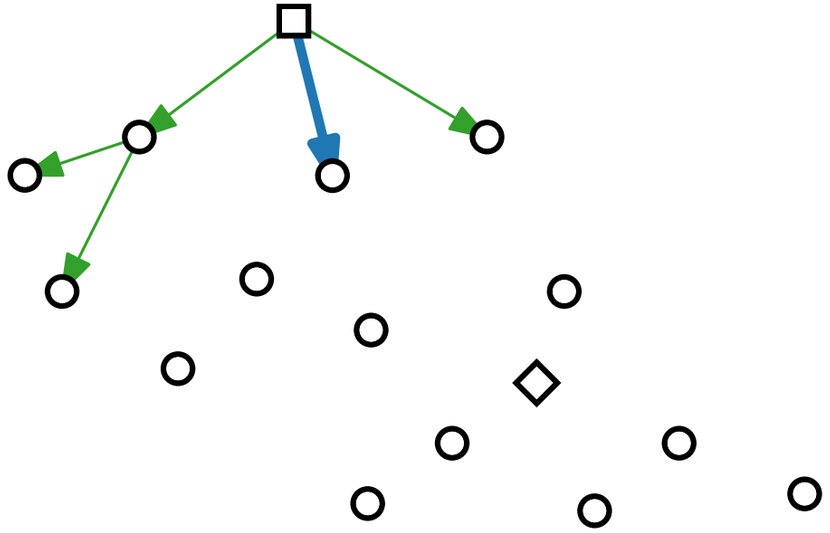


Jede Kante ist entweder

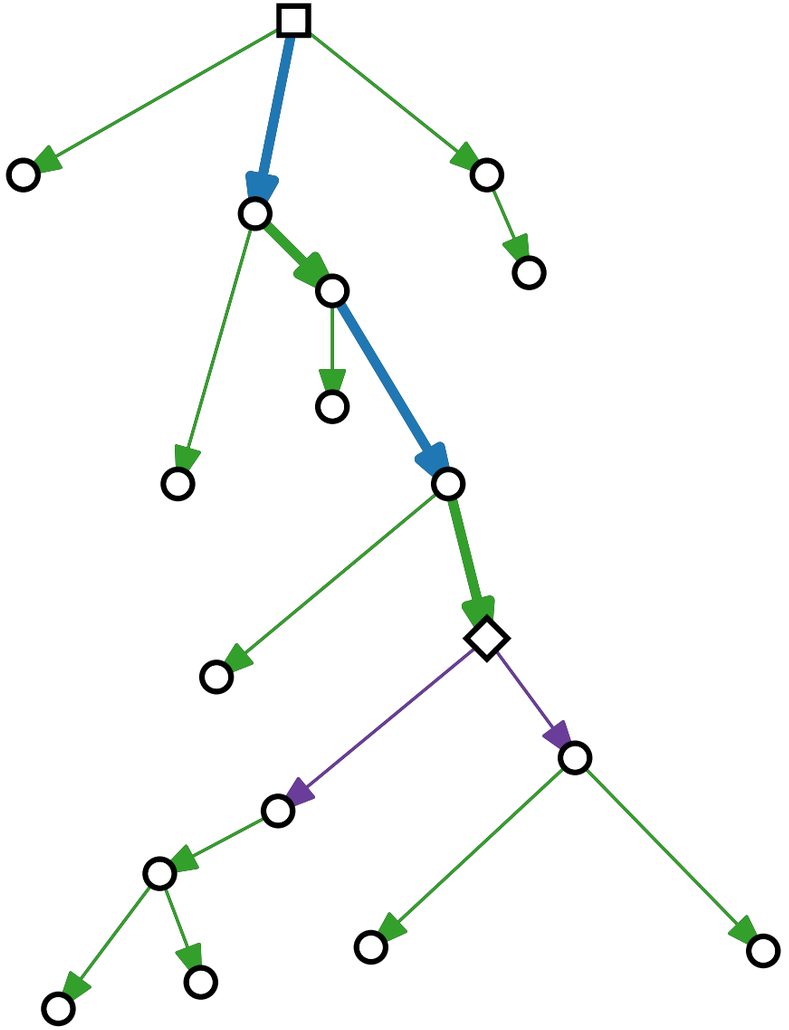
Eine Abkürzung

Auf dem Pfad von 0 zu n

Ausgehend von n



Eigenschaften optimaler Lösungen

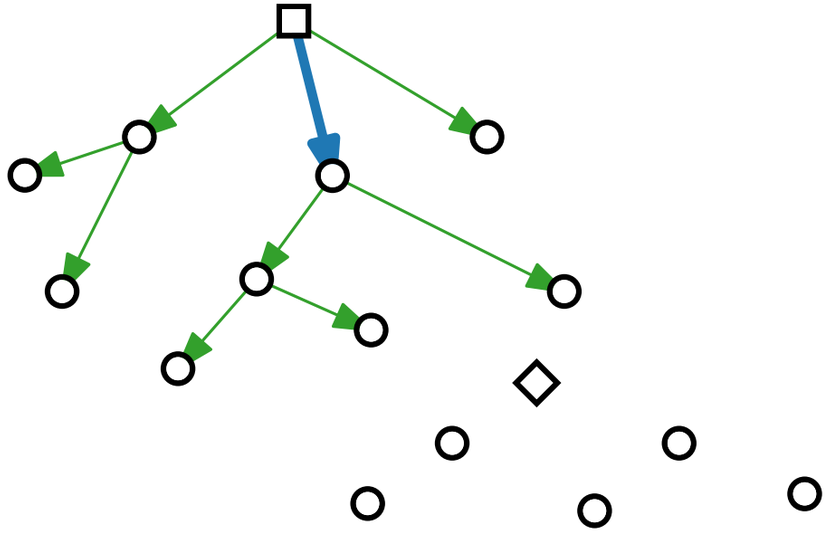


Jede Kante ist entweder

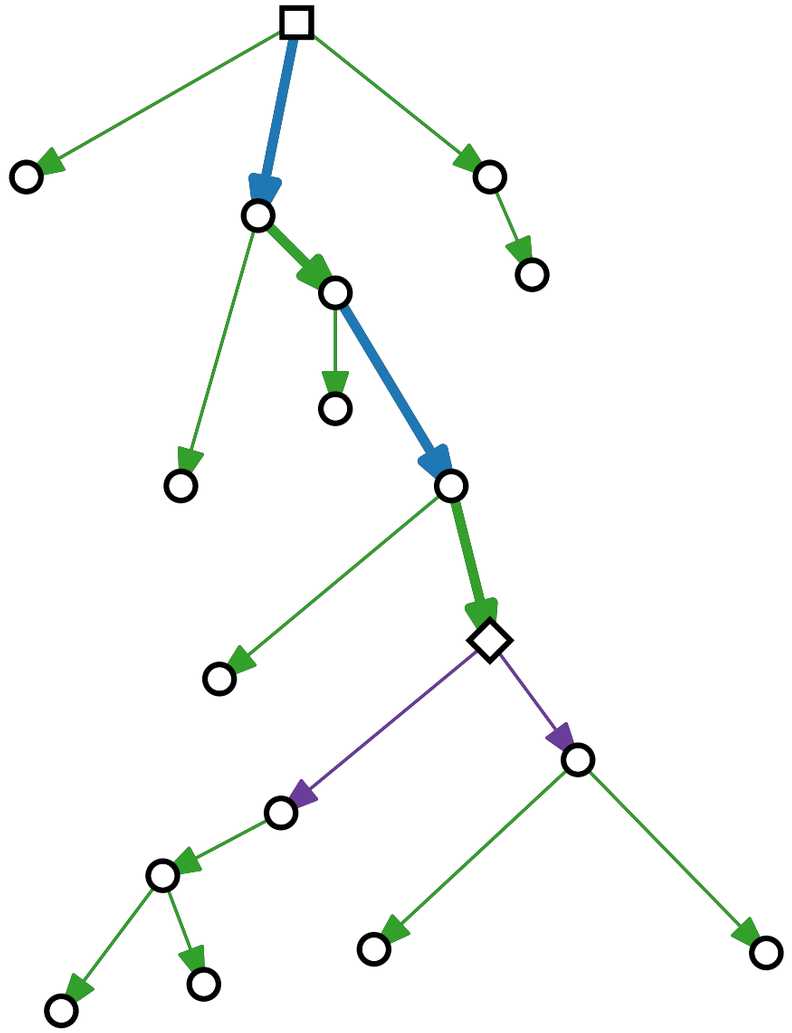
Eine Abkürzung

Auf dem Pfad von 0 zu n

Ausgehend von n



Eigenschaften optimaler Lösungen

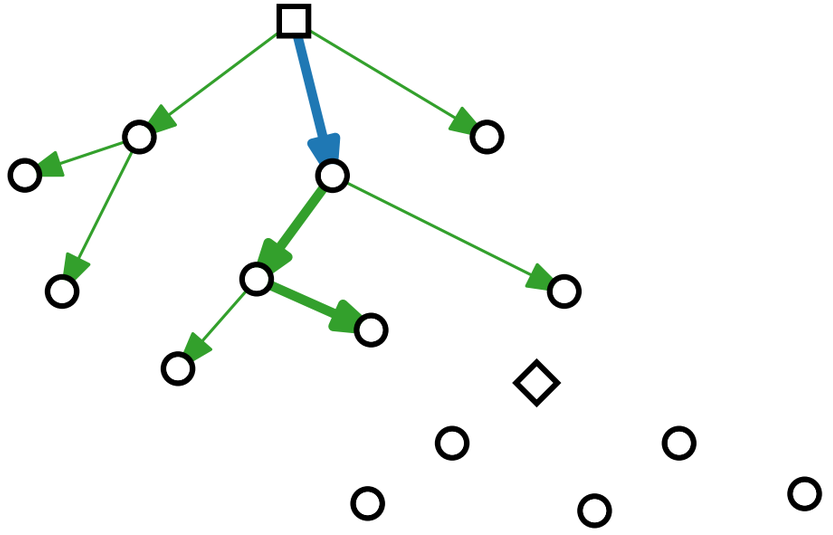


Jede Kante ist entweder

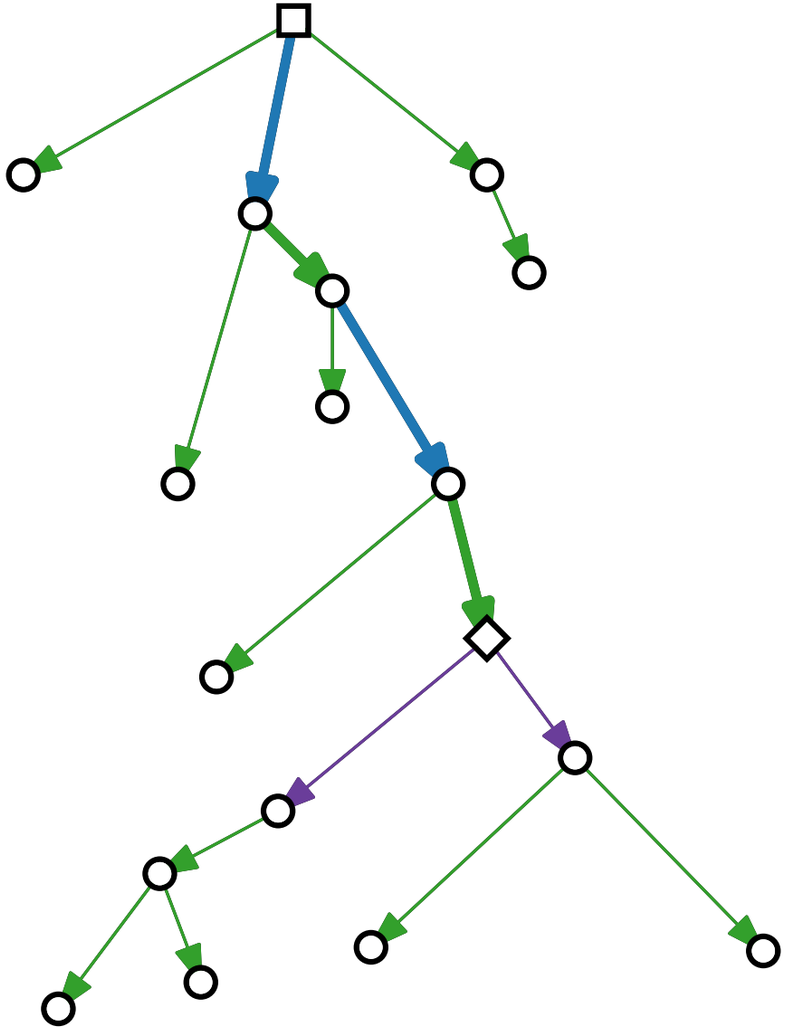
Eine Abkürzung

Auf dem Pfad von 0 zu n

Ausgehend von n



Eigenschaften optimaler Lösungen

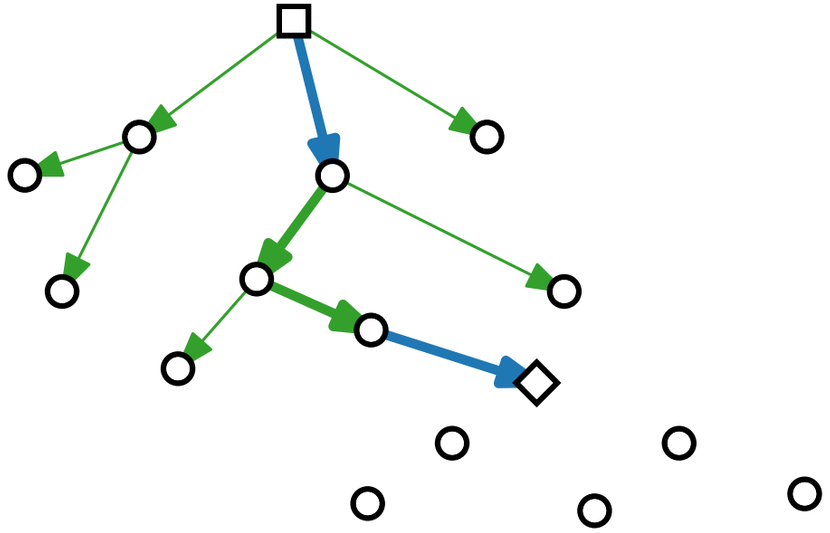


Jede Kante ist entweder

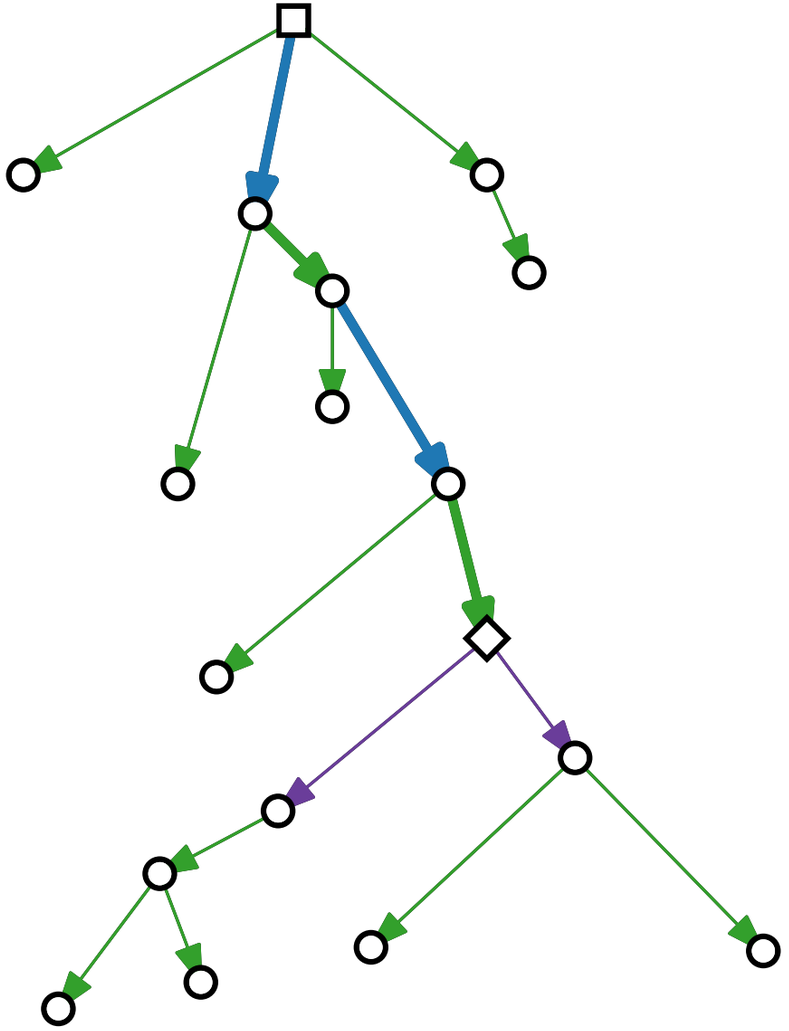
Eine Abkürzung

Auf dem Pfad von 0 zu n

Ausgehend von n



Eigenschaften optimaler Lösungen

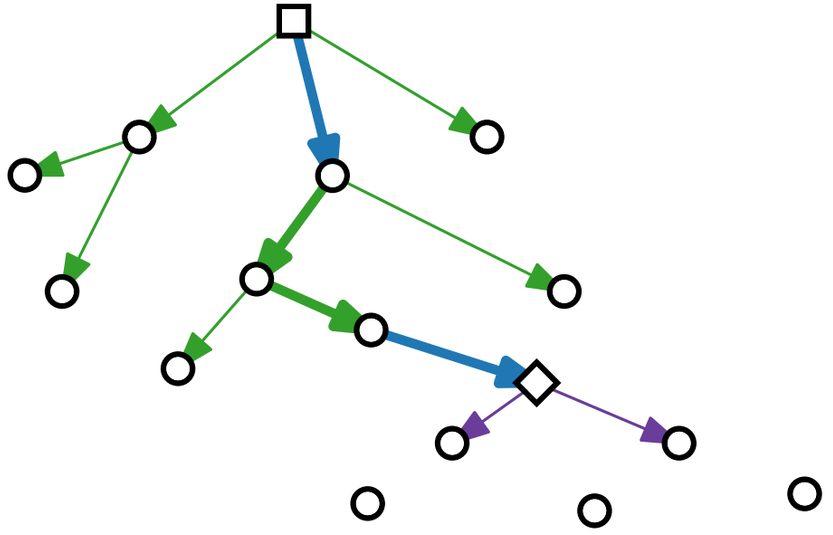


Jede Kante ist entweder

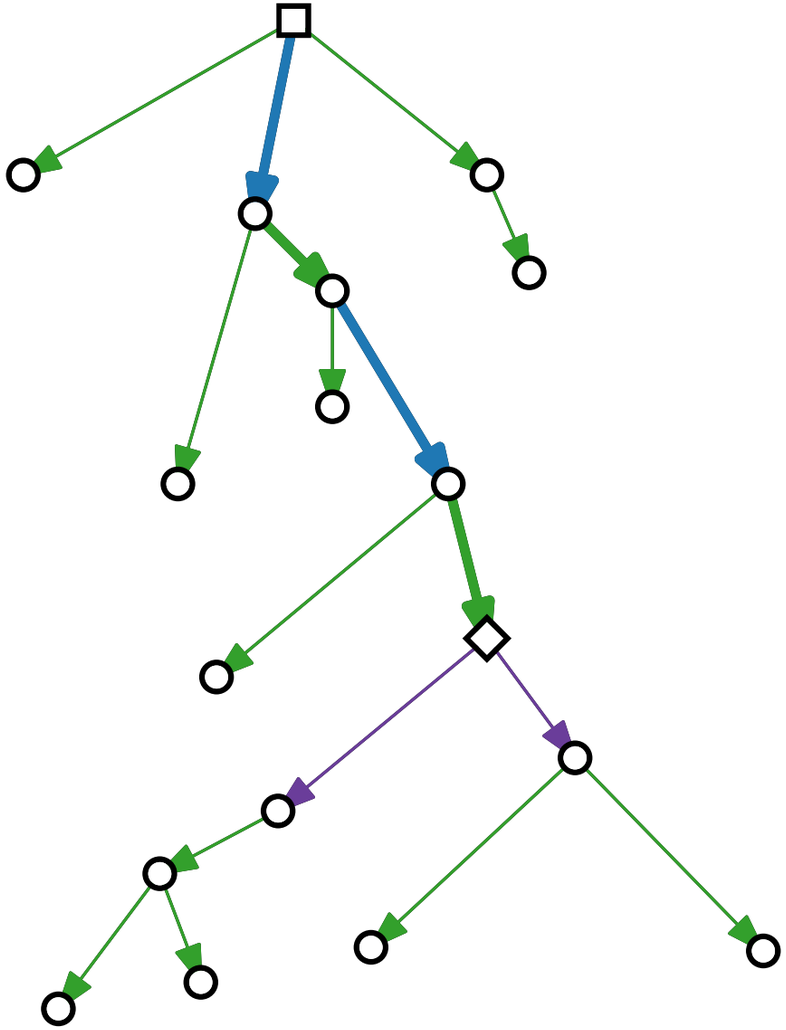
Eine Abkürzung

Auf dem Pfad von 0 zu n

Ausgehend von n



Eigenschaften optimaler Lösungen

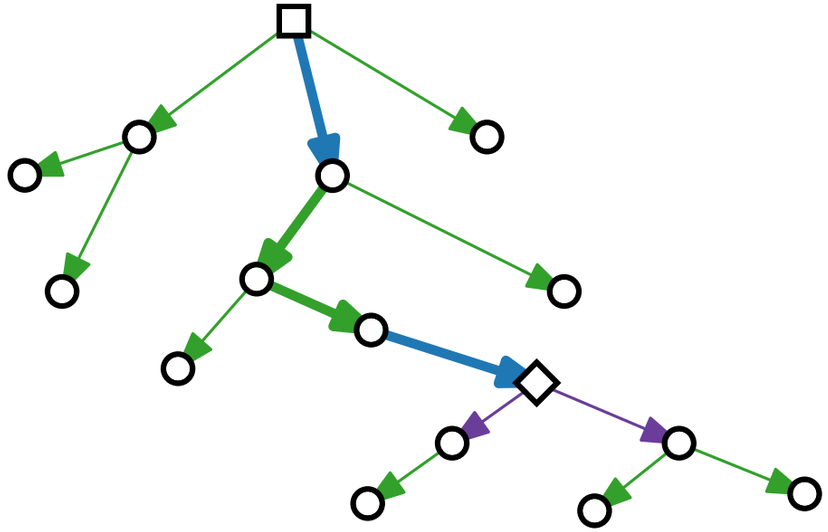


Jede Kante ist entweder

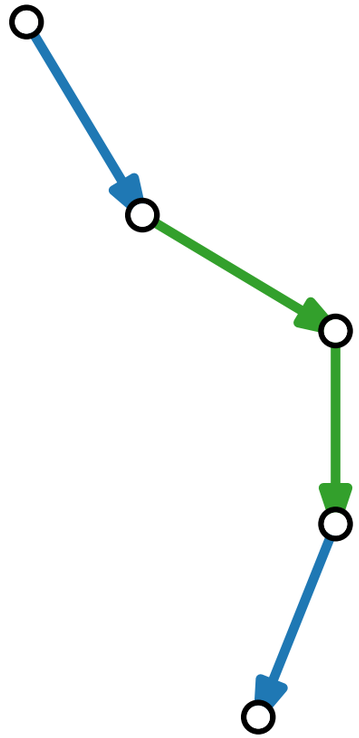
Eine Abkürzung

Auf dem Pfad von 0 zu n

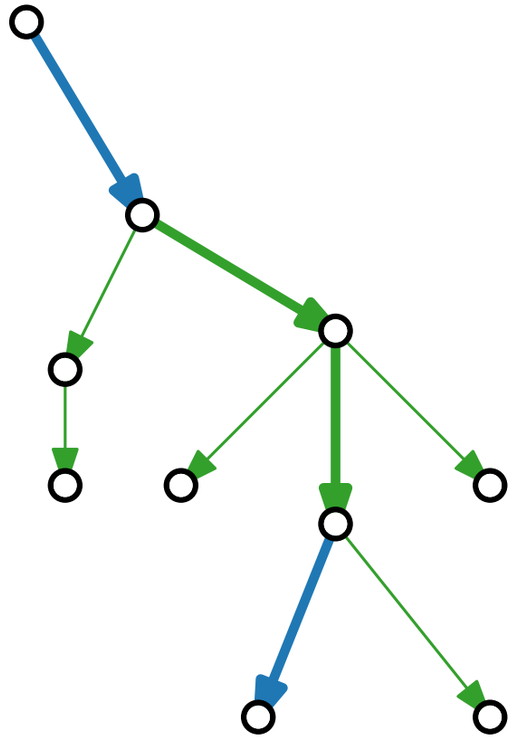
Ausgehend von n



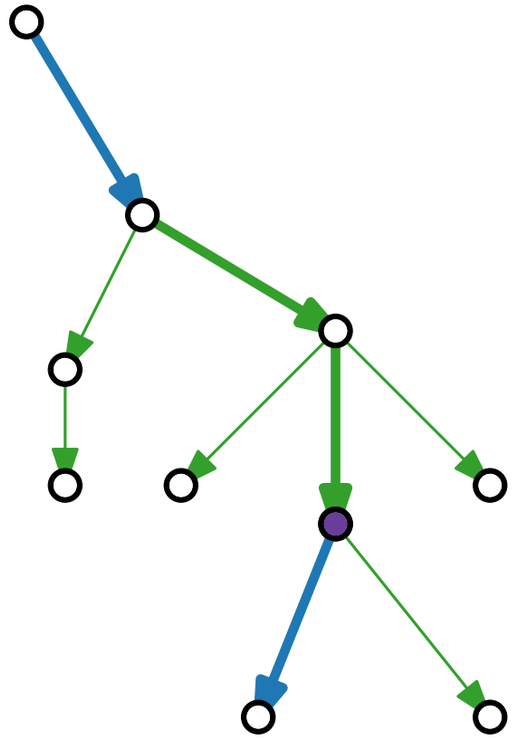
Eigenschaften optimaler Lösungen



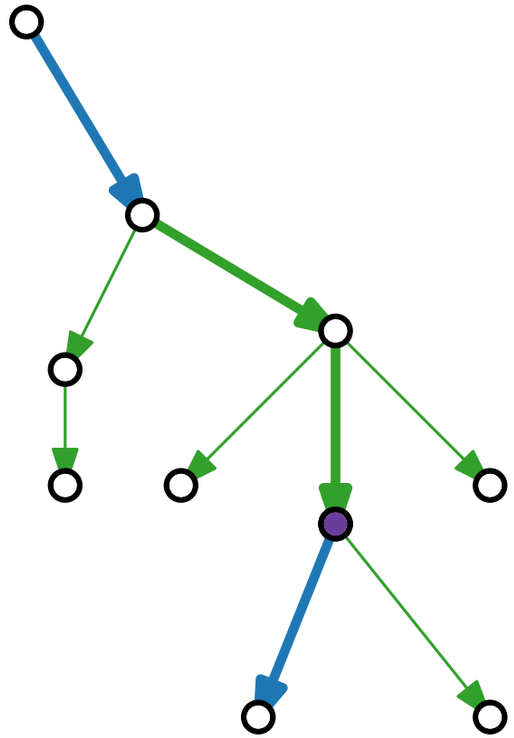
Eigenschaften optimaler Lösungen



Eigenschaften optimaler Lösungen

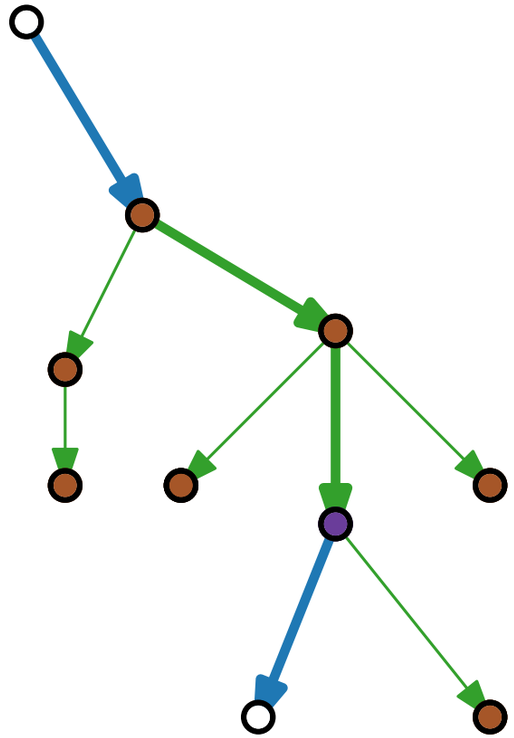


Eigenschaften optimaler Lösungen



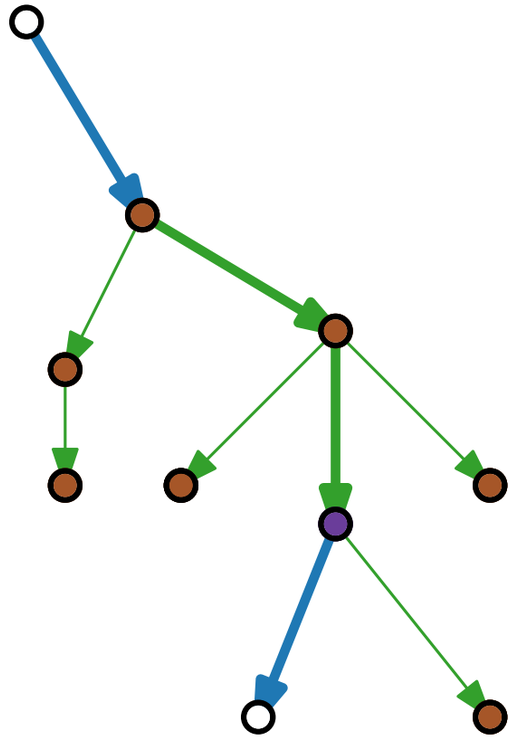
maximaler Knoten

Eigenschaften optimaler Lösungen



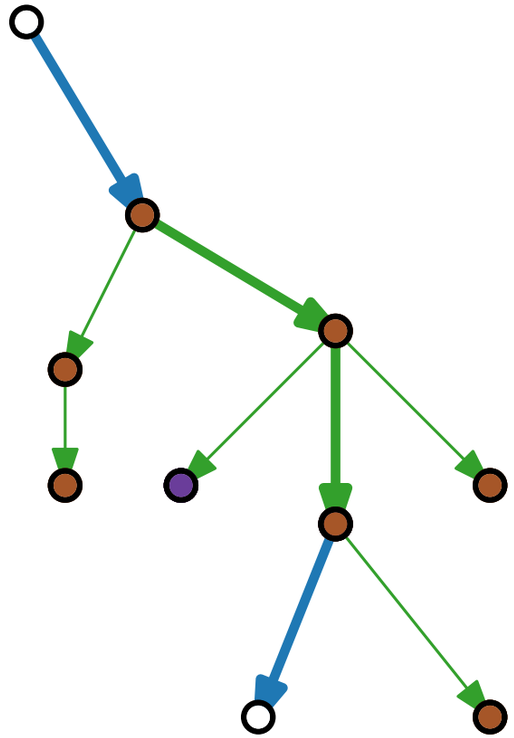
maximaler Knoten

Eigenschaften optimaler Lösungen



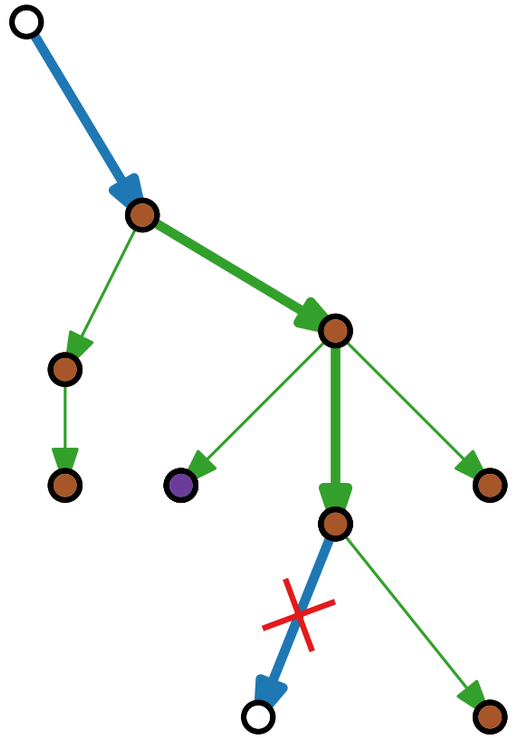
maximaler Knoten
von allen erreichbaren

Eigenschaften optimaler Lösungen



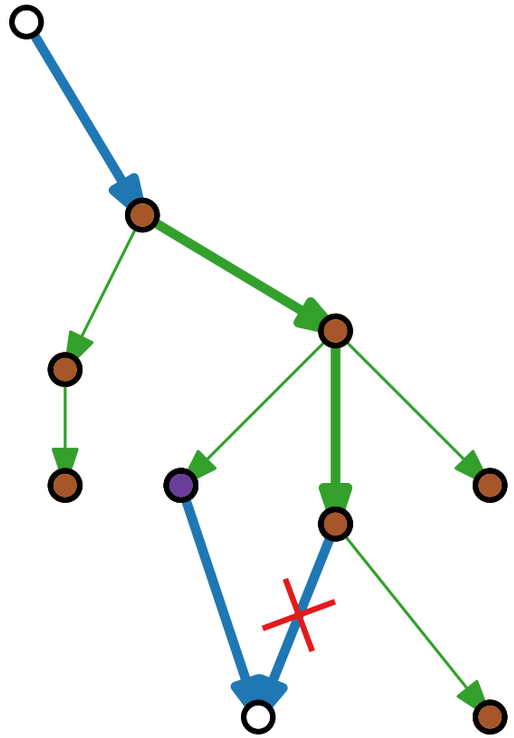
maximaler Knoten
von allen erreichbaren

Eigenschaften optimaler Lösungen



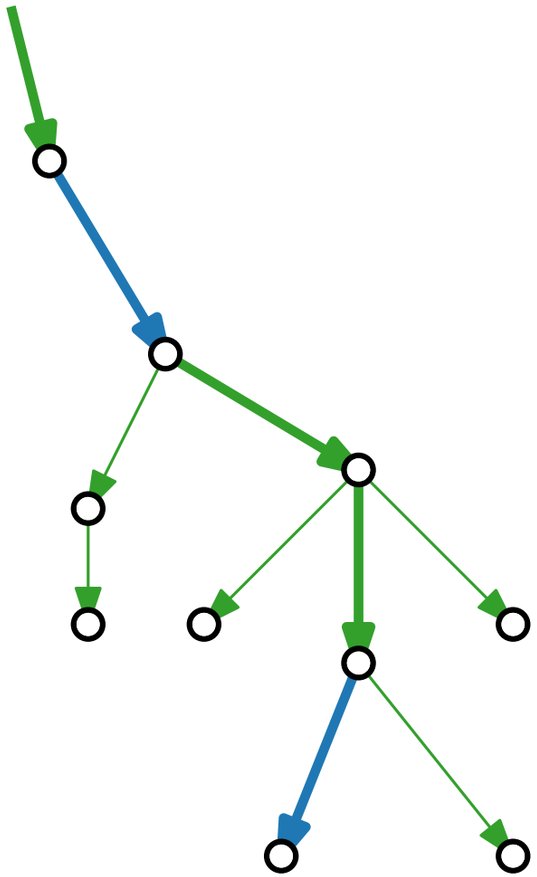
maximaler Knoten
von allen erreichbaren

Eigenschaften optimaler Lösungen

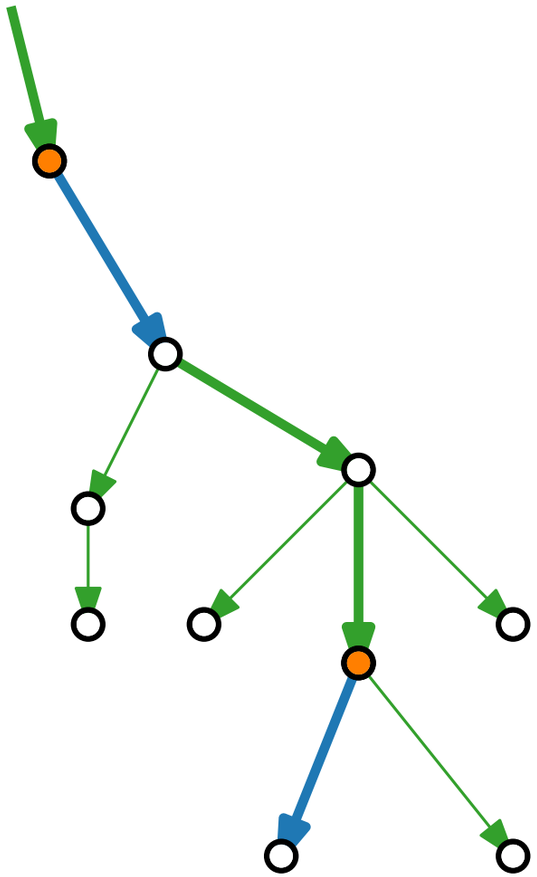


maximaler Knoten
von allen erreichbaren

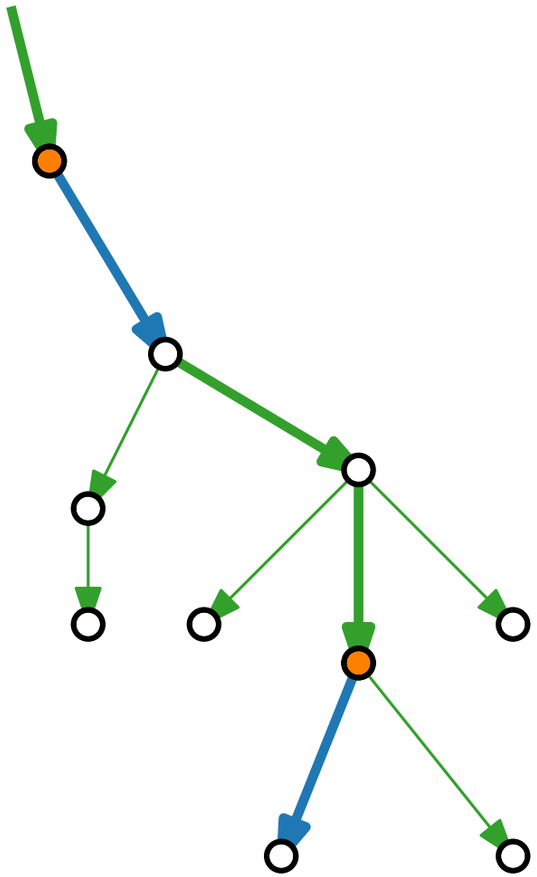
Eigenschaften optimaler Lösungen



Eigenschaften optimaler Lösungen

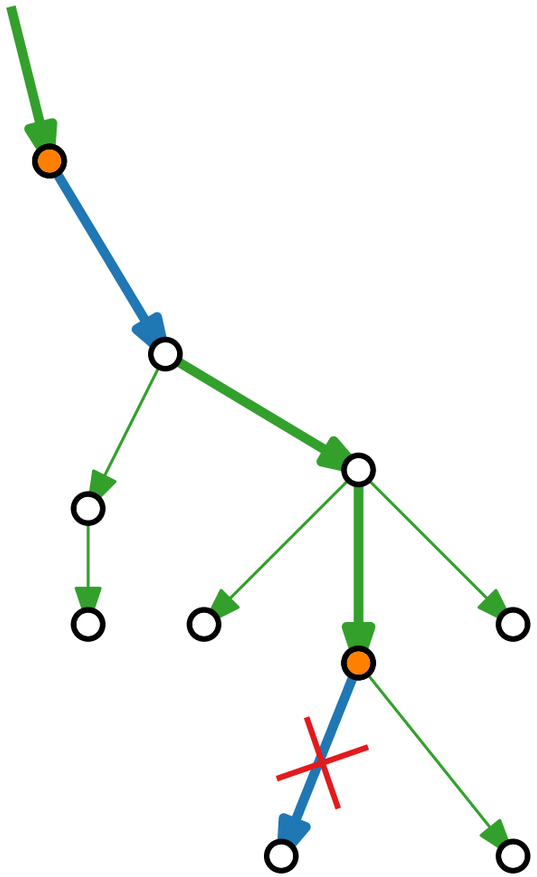


Eigenschaften optimaler Lösungen



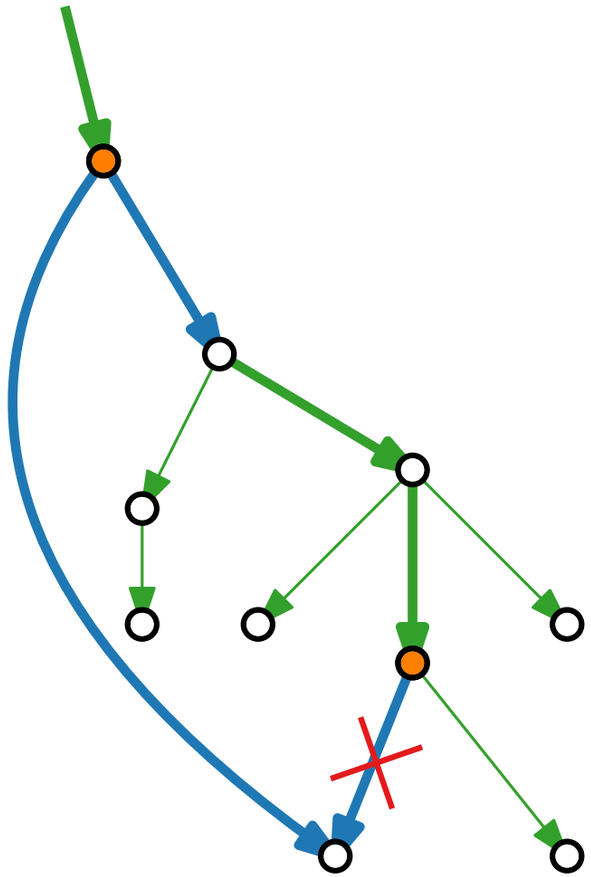
Quellen blauer Kanten sind aufsteigend

Eigenschaften optimaler Lösungen



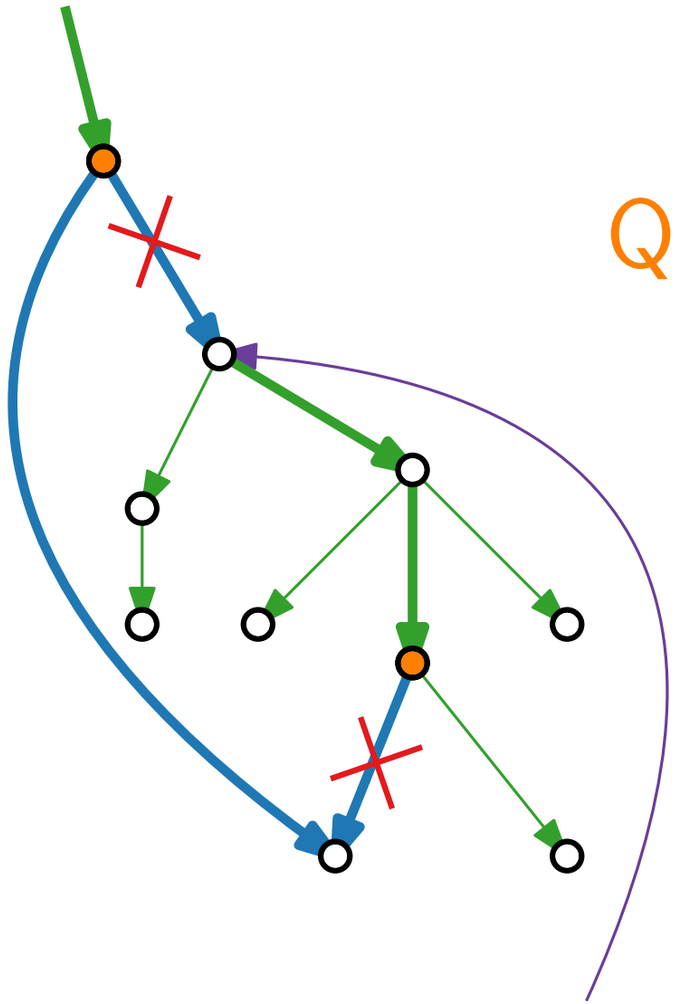
Quellen blauer Kanten sind aufsteigend

Eigenschaften optimaler Lösungen



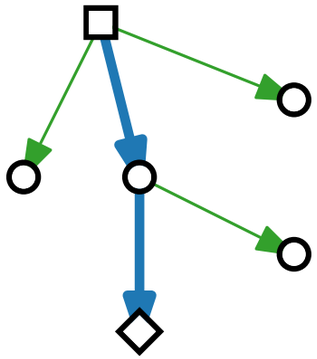
Quellen blauer Kanten sind aufsteigend

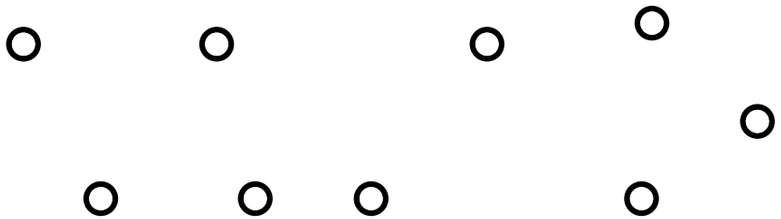
Eigenschaften optimaler Lösungen



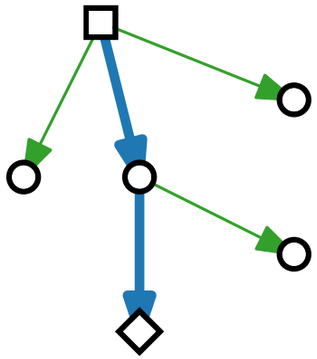
Quellen blauer Kanten sind aufsteigend

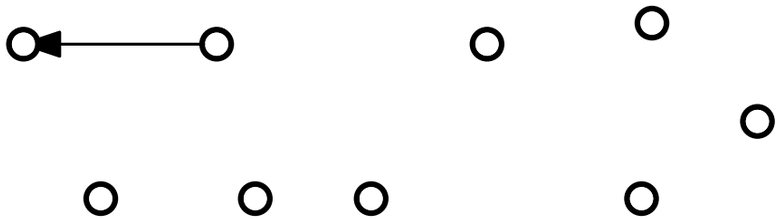
Eigenschaften optimaler Lösungen



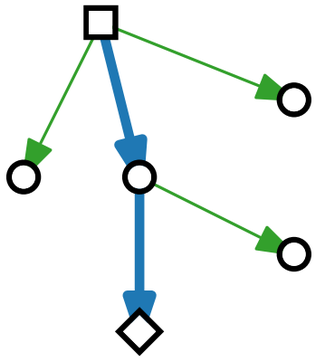


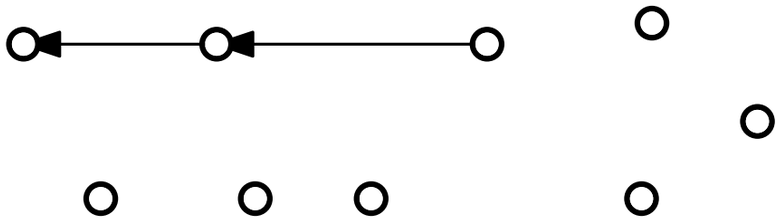
Eigenschaften optimaler Lösungen



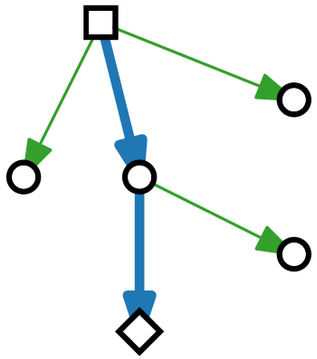


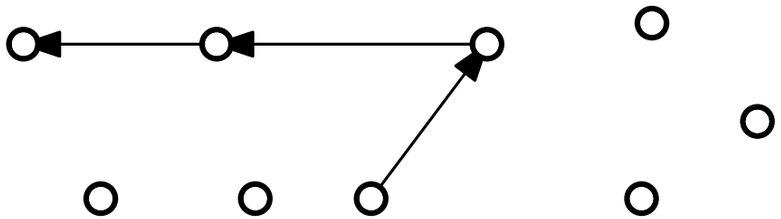
Eigenschaften optimaler Lösungen



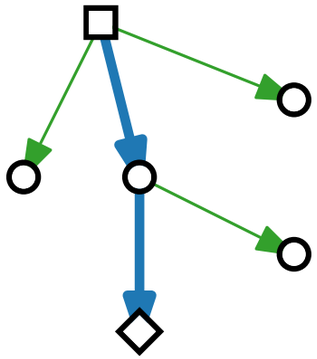


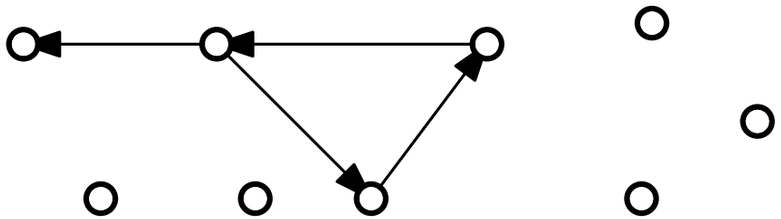
Eigenschaften optimaler Lösungen



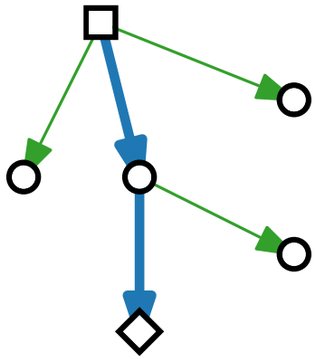


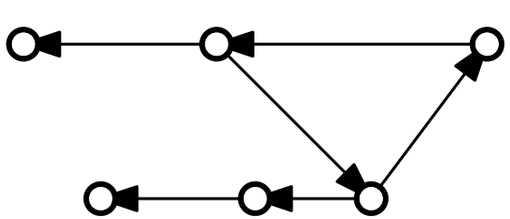
Eigenschaften optimaler Lösungen

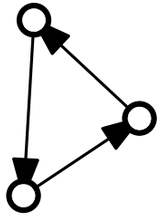




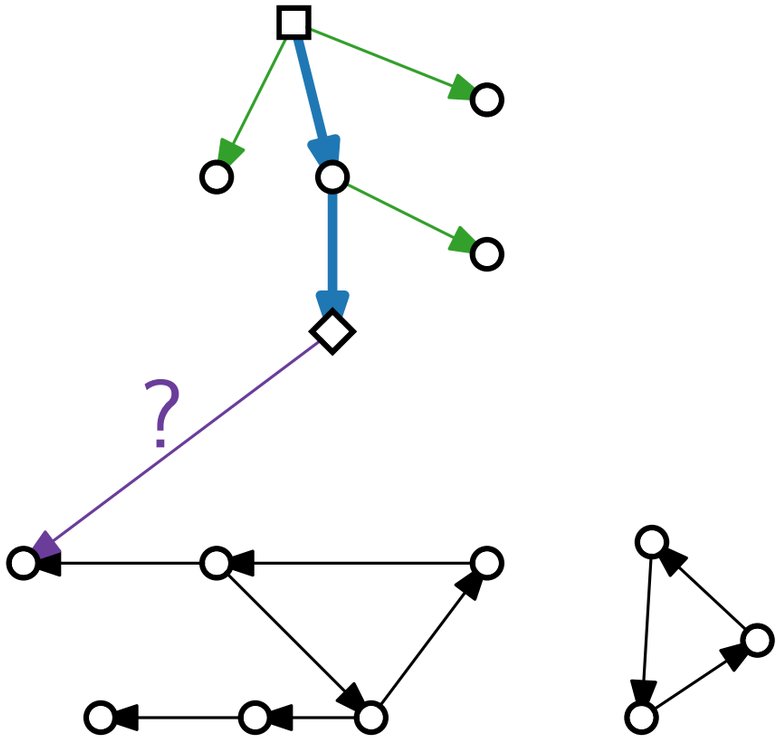
Eigenschaften optimaler Lösungen



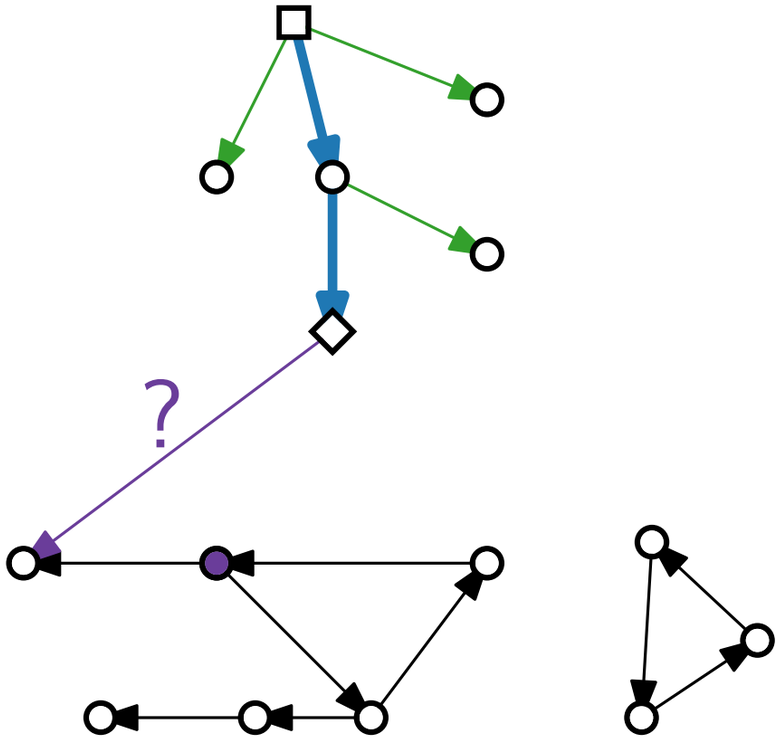




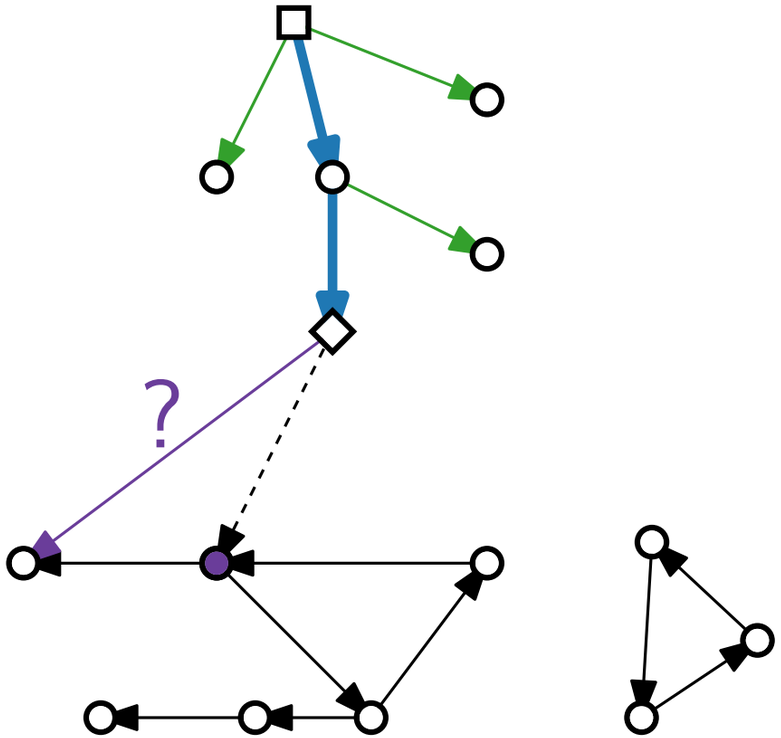
Eigenschaften optimaler Lösungen



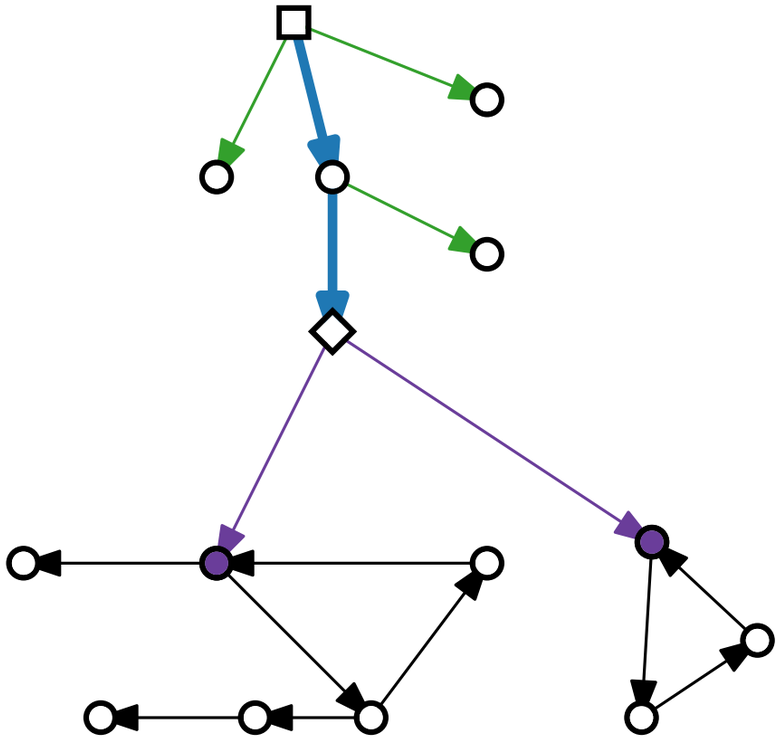
Eigenschaften optimaler Lösungen



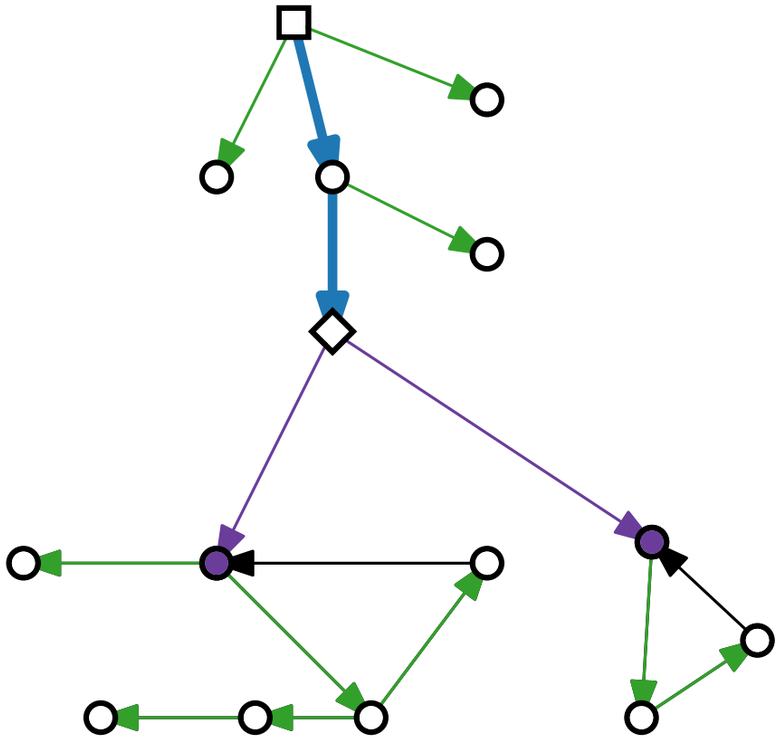
Eigenschaften optimaler Lösungen



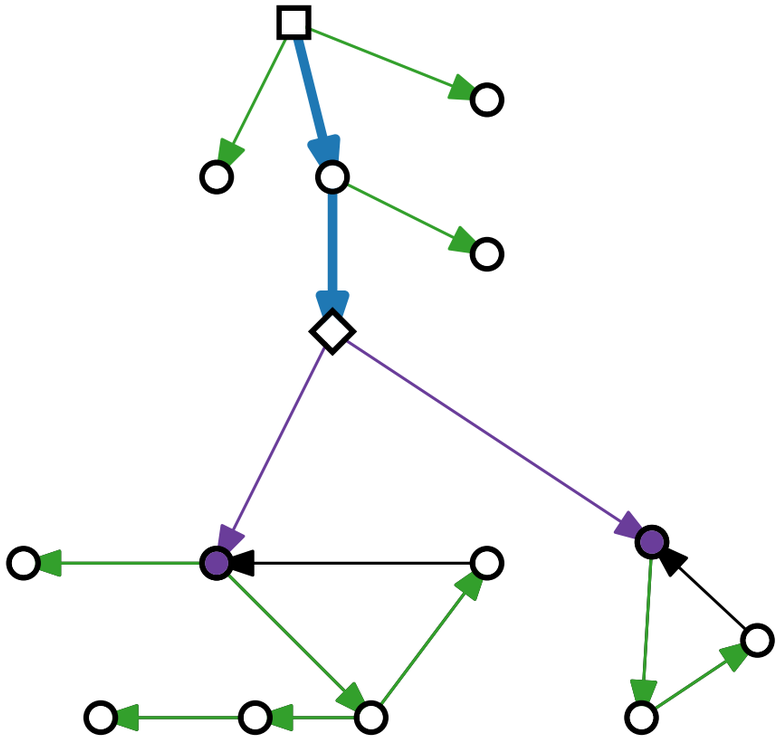
Eigenschaften optimaler Lösungen



Eigenschaften optimaler Lösungen

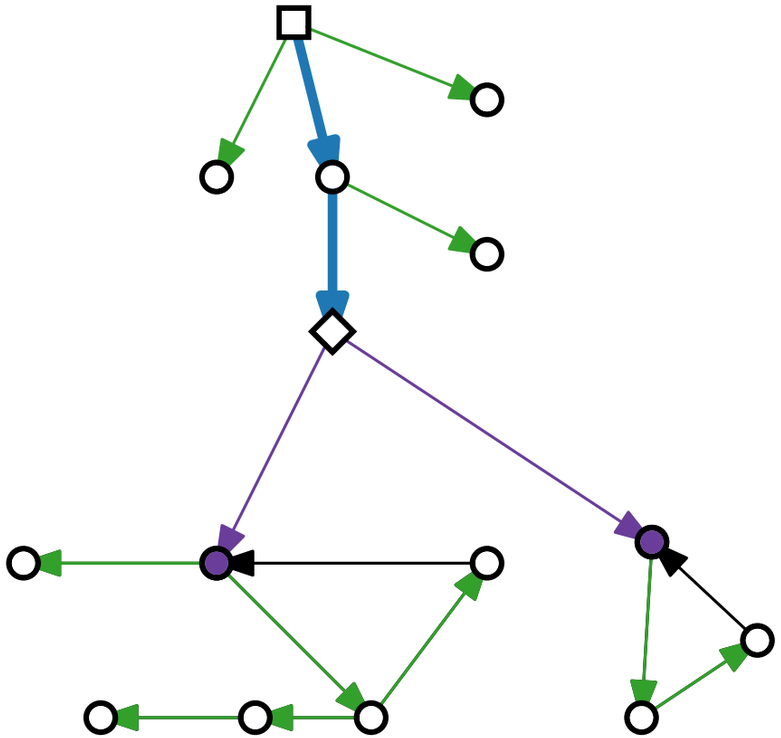


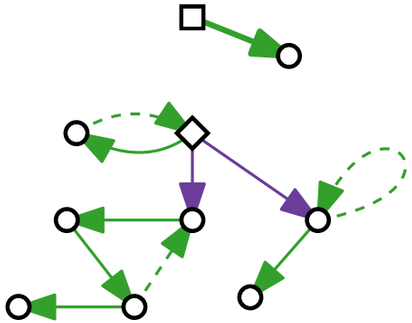
Eigenschaften optimaler Lösungen



Kante mit minimalen Kosten in jeden Zyklus

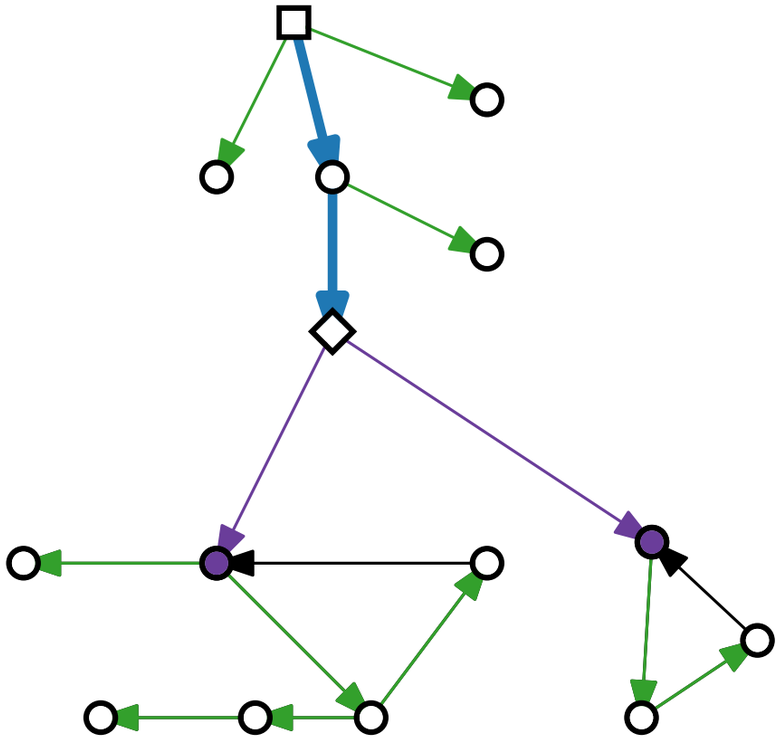
Eigenschaften optimaler Lösungen

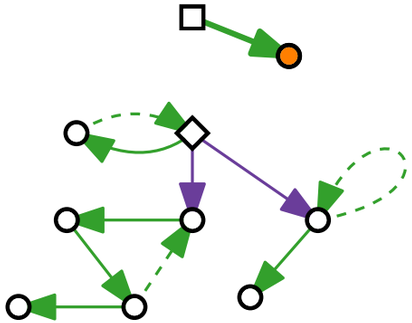




Kante mit minimalen Kosten in jeden Zyklus

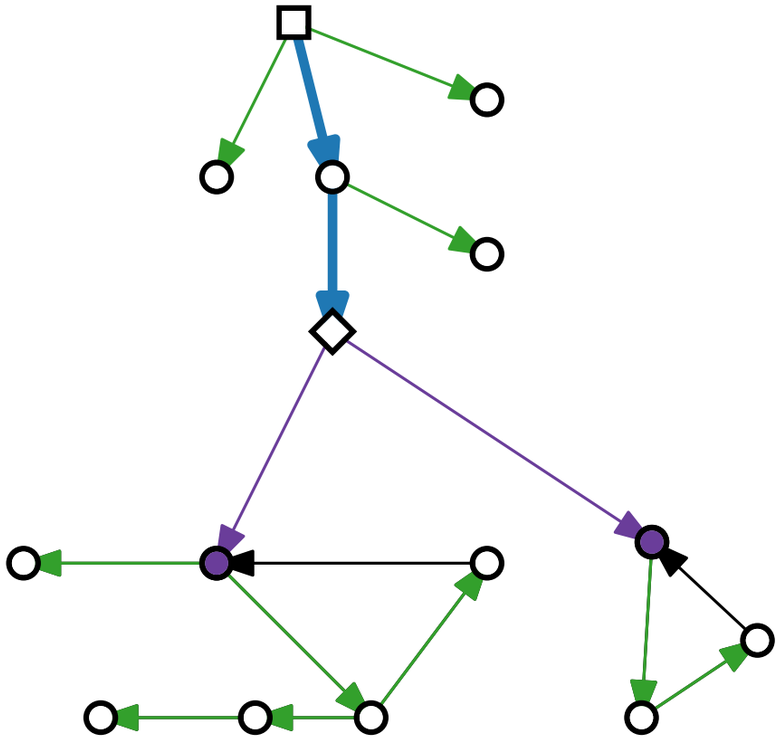
Eigenschaften optimaler Lösungen

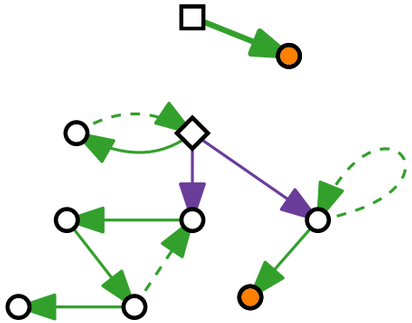




Kante mit minimalen Kosten in jeden Zyklus

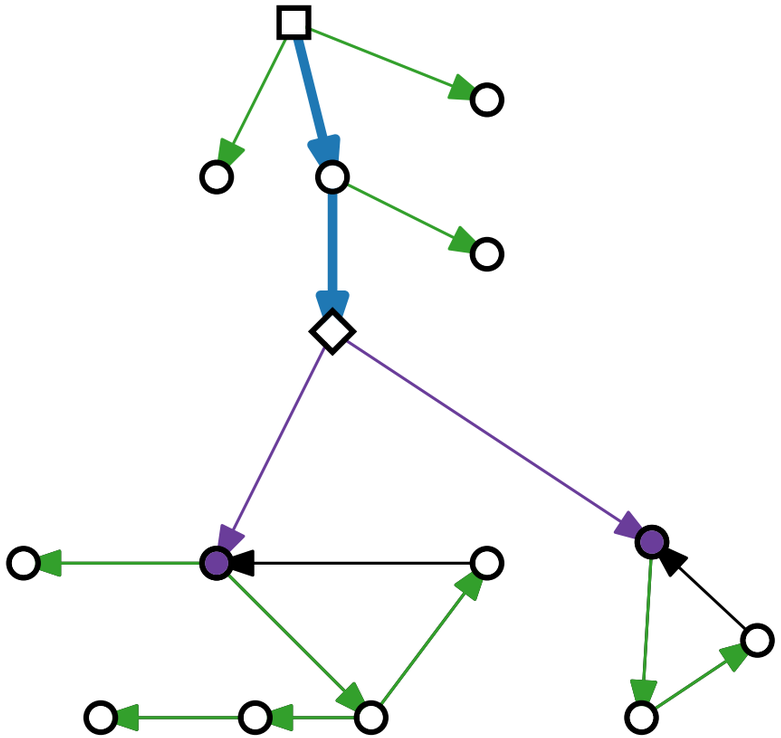
Eigenschaften optimaler Lösungen

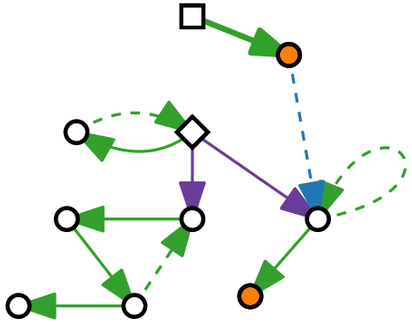




Kante mit minimalen Kosten in jeden Zyklus

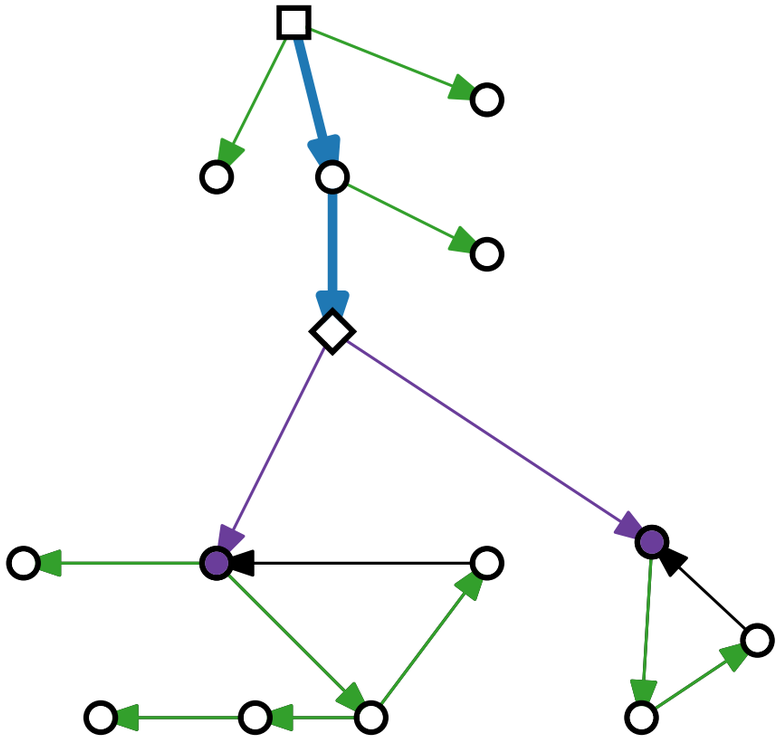
Eigenschaften optimaler Lösungen

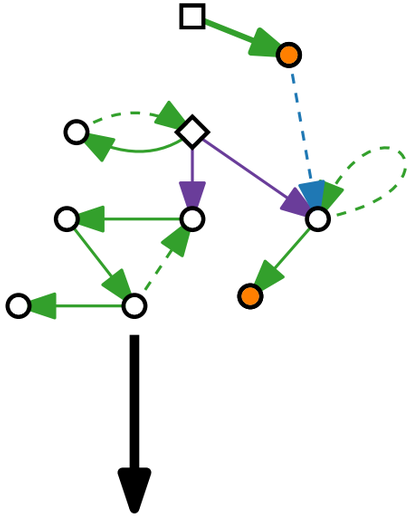




Kante mit minimalen Kosten in jeden Zyklus

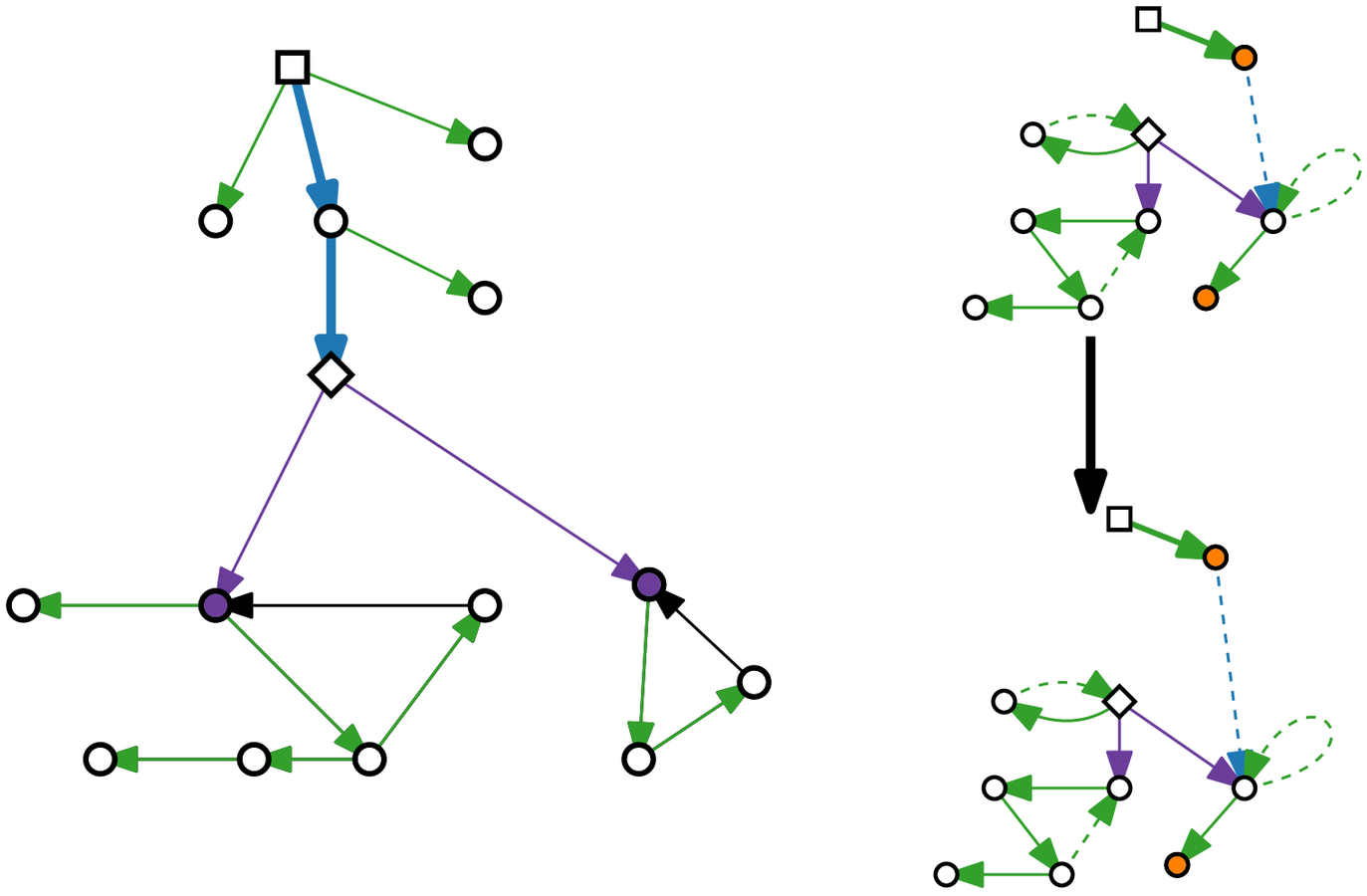
Eigenschaften optimaler Lösungen





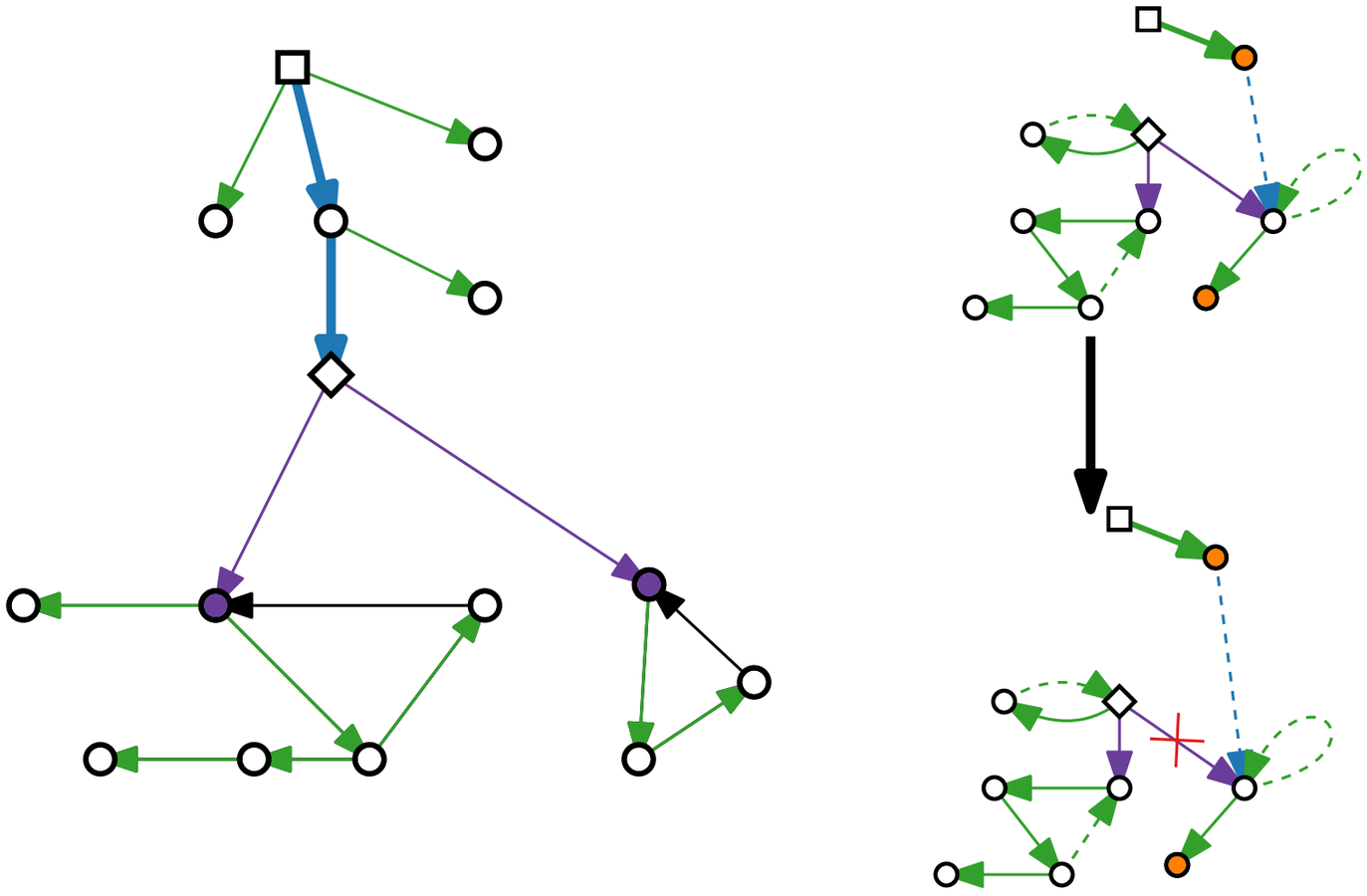
Kante mit minimalen Kosten in jeden Zyklus

Eigenschaften optimaler Lösungen



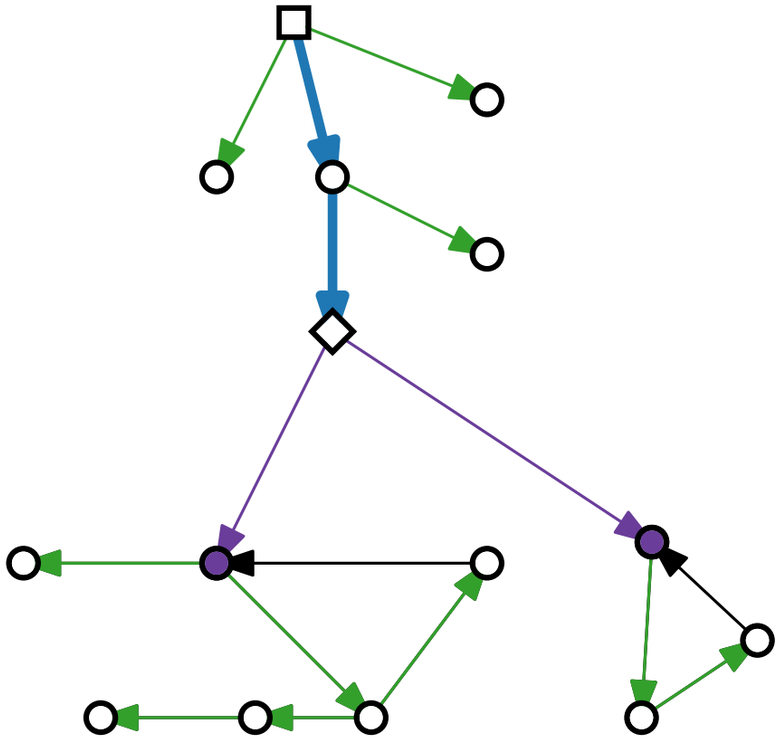
Kante mit minimalen Kosten in jeden Zyklus

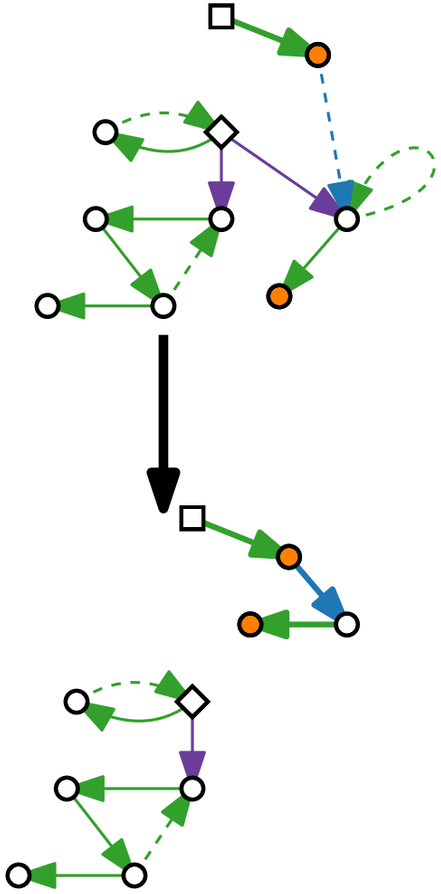
Eigenschaften optimaler Lösungen



Kante mit minimalen Kosten in jeden Zyklus

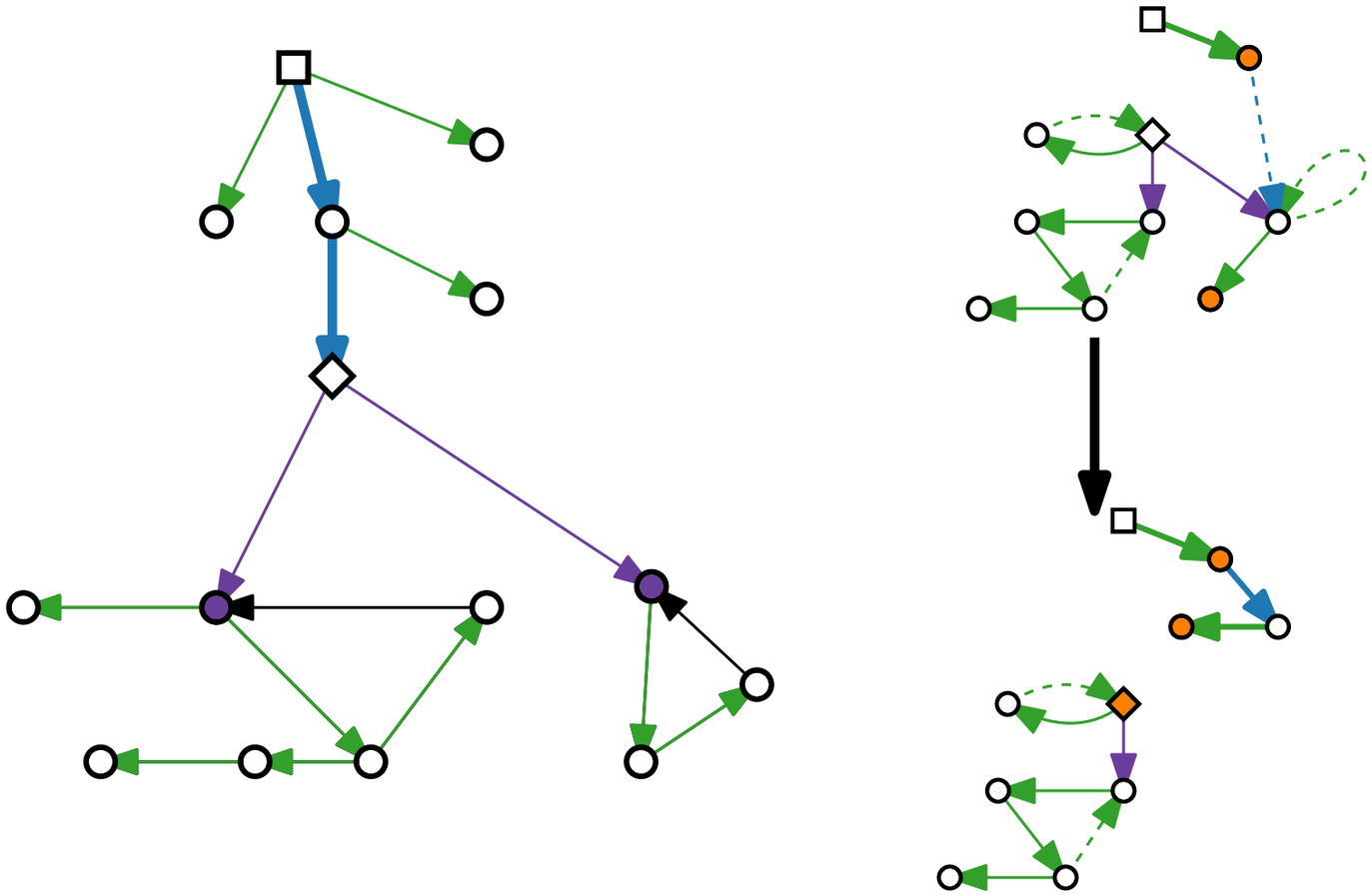
Eigenschaften optimaler Lösungen





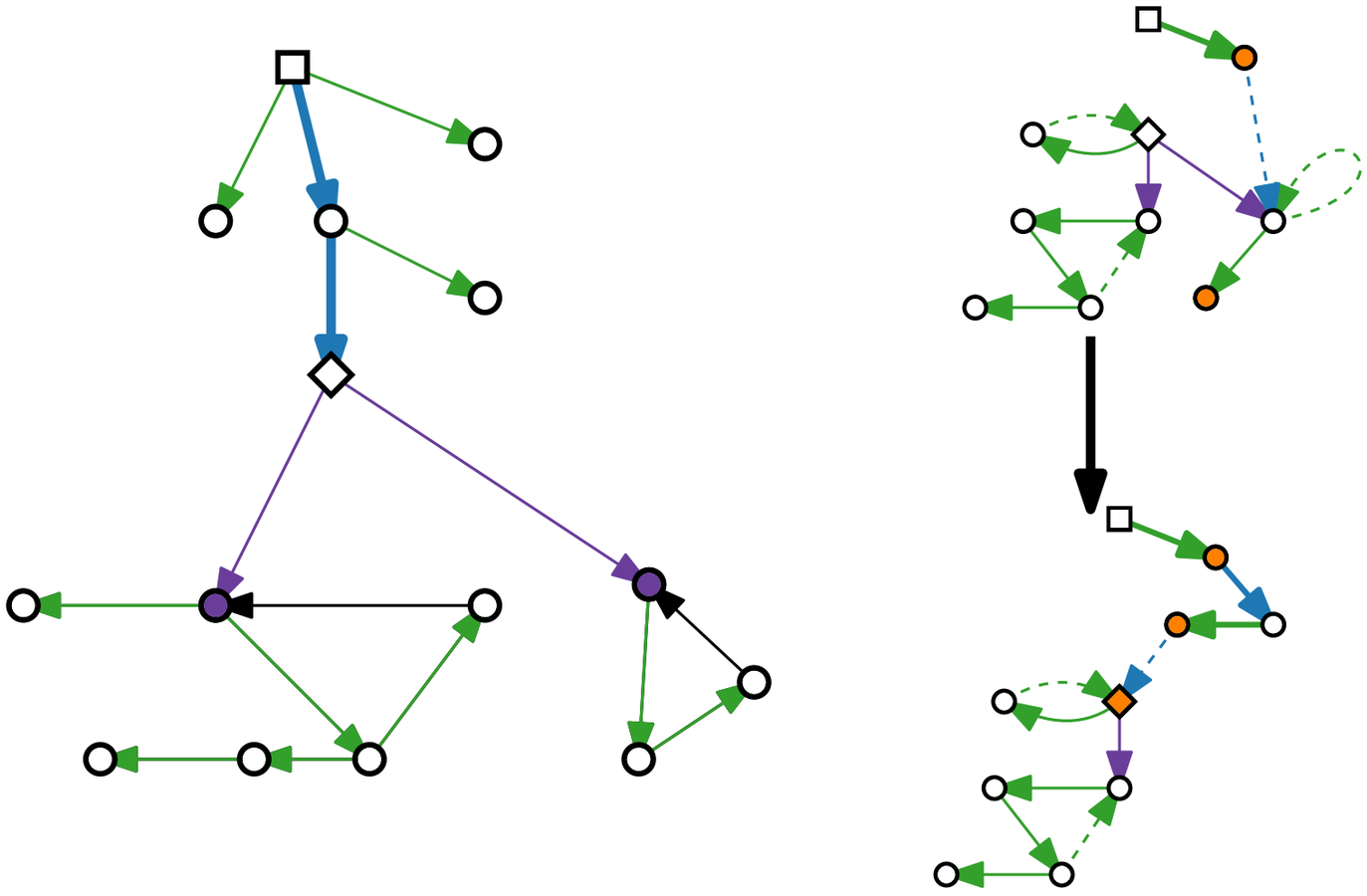
Kante mit minimalen Kosten in jeden Zyklus

Eigenschaften optimaler Lösungen



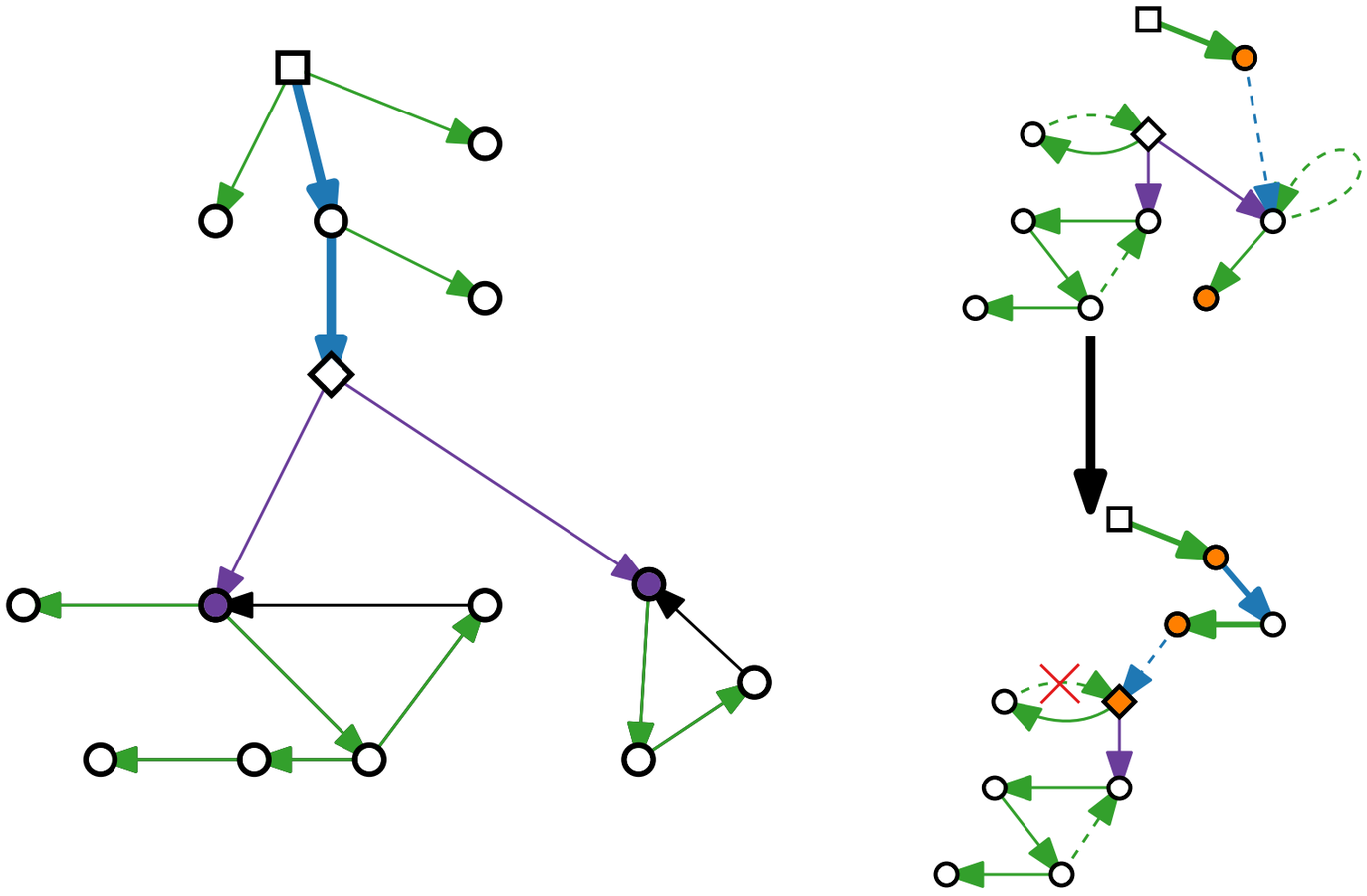
Kante mit minimalen Kosten in jeden Zyklus

Eigenschaften optimaler Lösungen



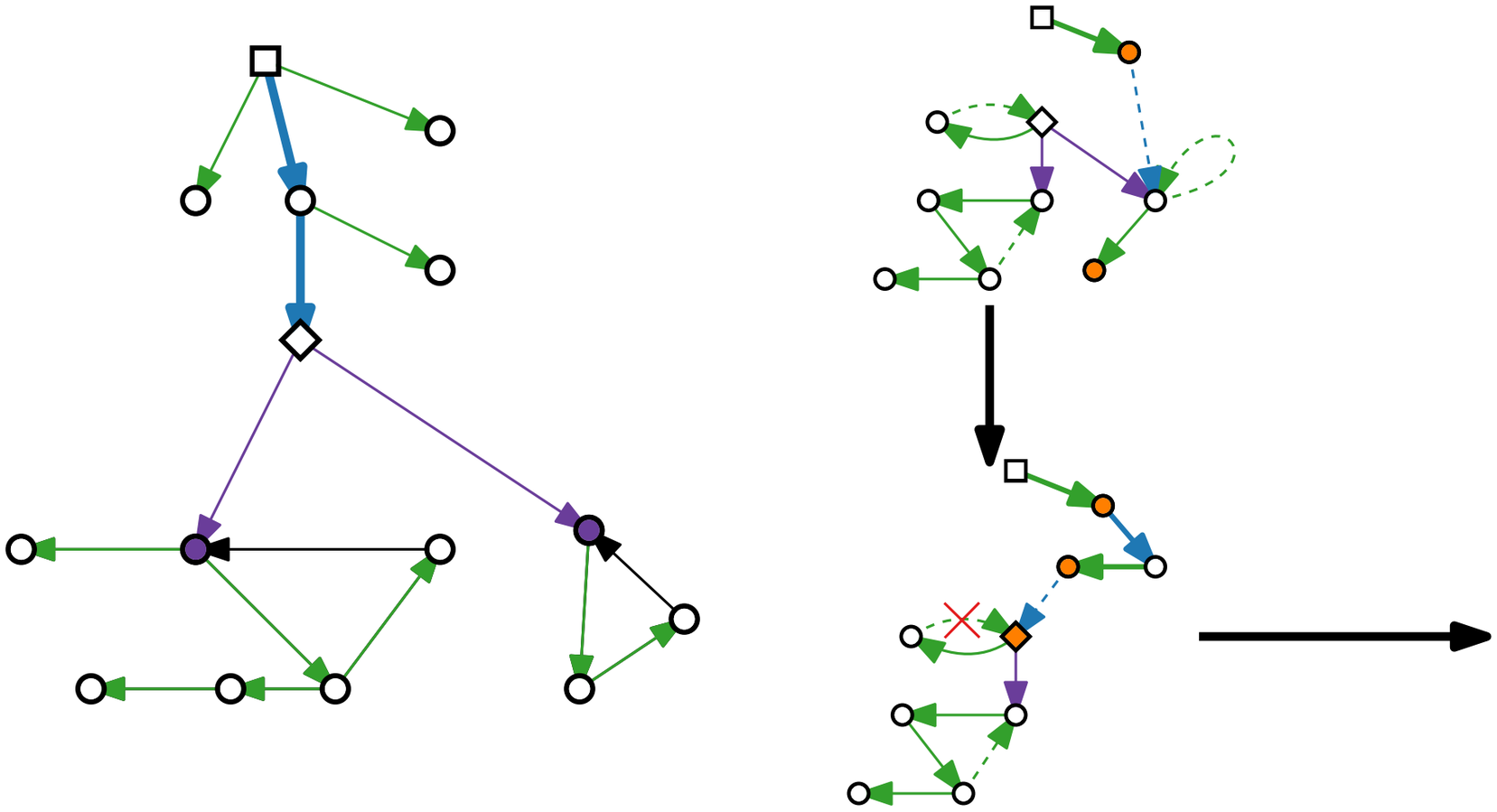
Kante mit minimalen Kosten in jeden Zyklus

Eigenschaften optimaler Lösungen



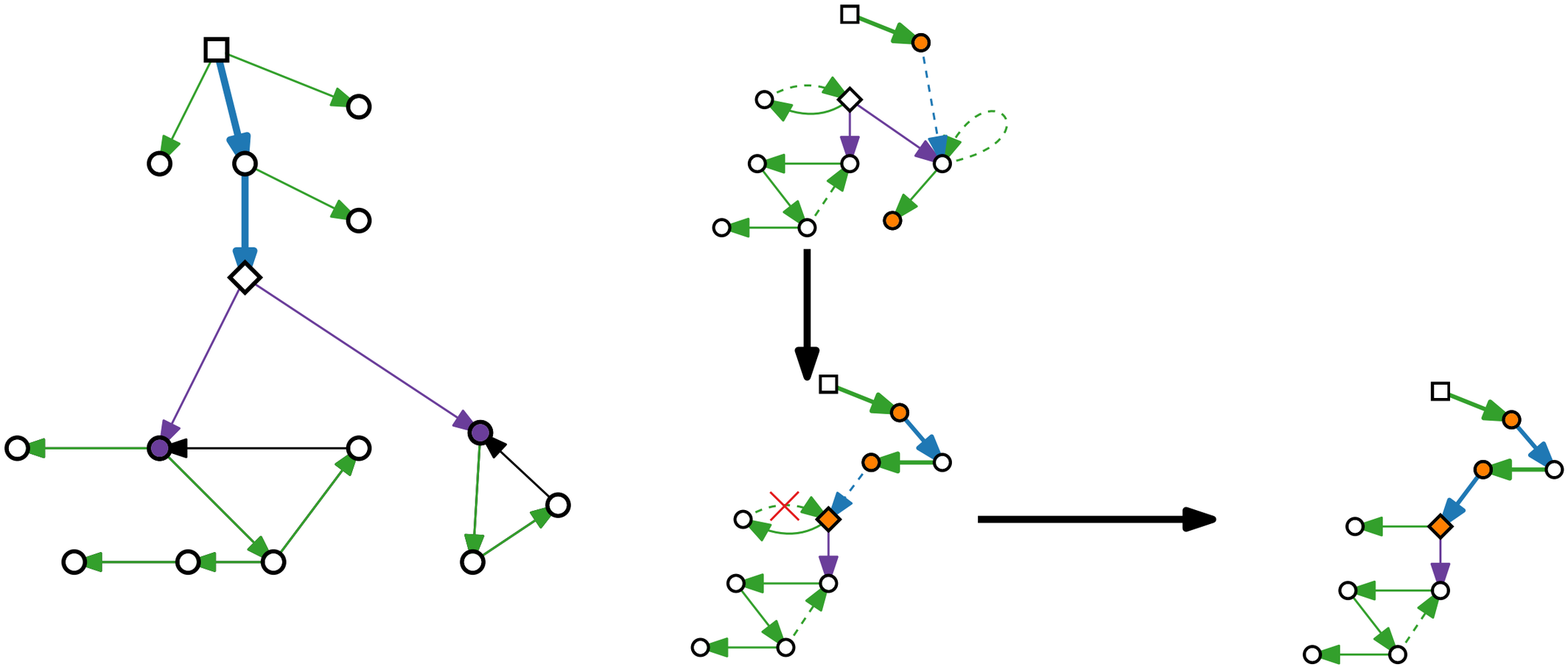
Kante mit minimalen Kosten in jeden Zyklus

Eigenschaften optimaler Lösungen



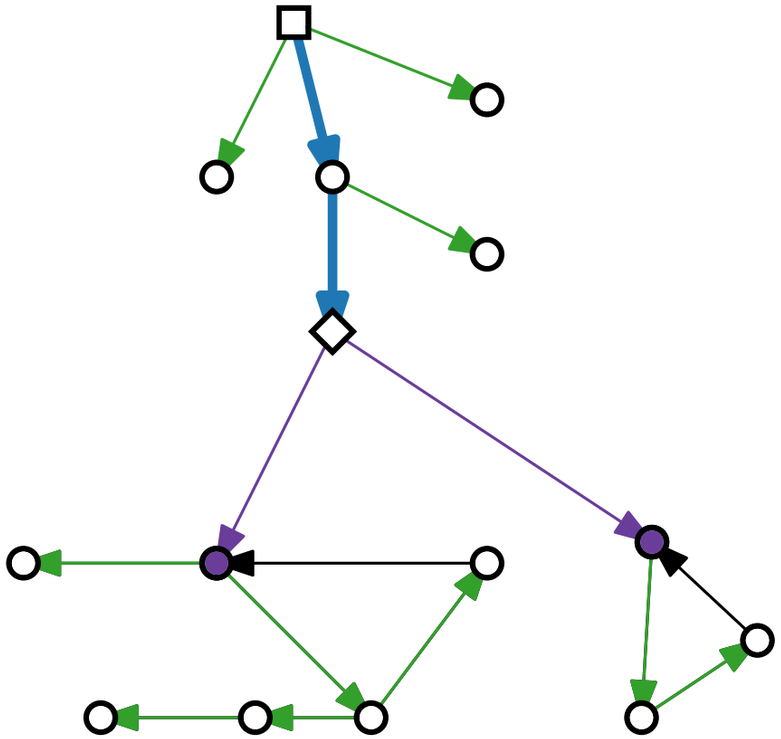
Kante mit minimalen Kosten in jeden Zyklus

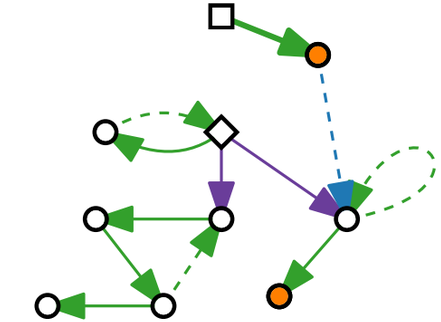
Eigenschaften optimaler Lösungen

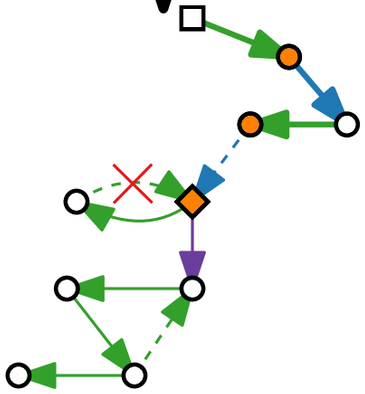


Kante mit minimalen Kosten in jeden Zyklus

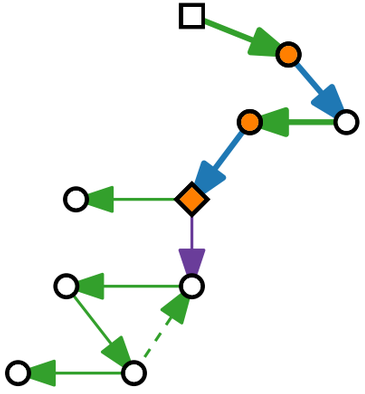
Eigenschaften optimaler Lösungen





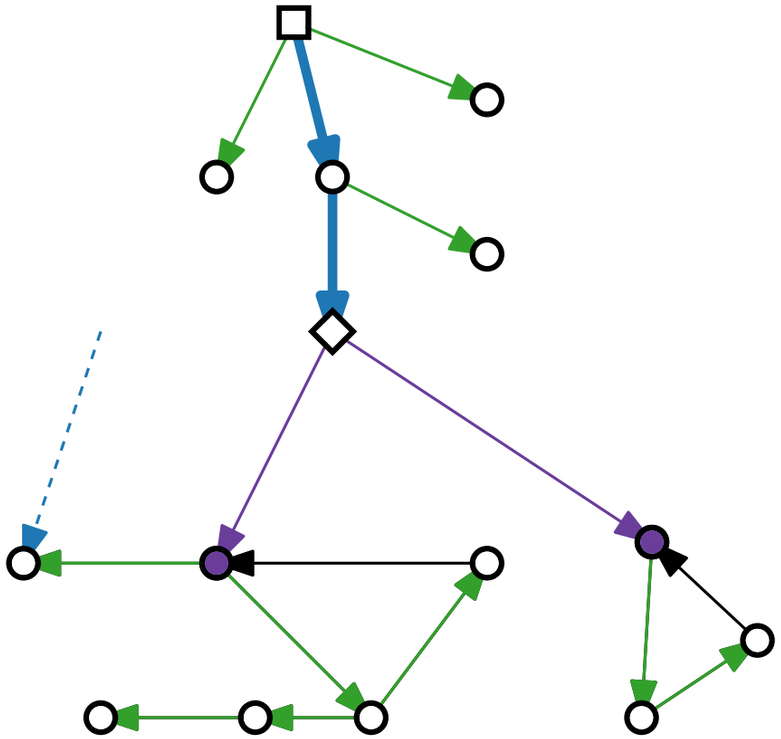


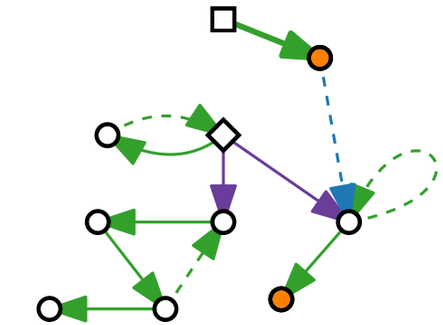
e_i – Ersparnis

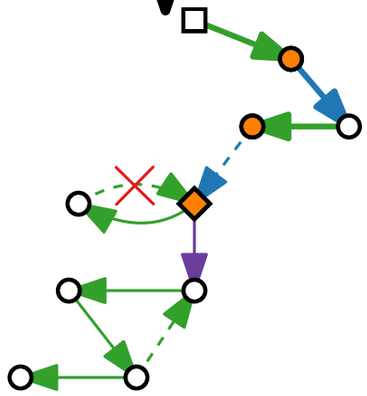


Kante mit minimalen Kosten in jeden Zyklus

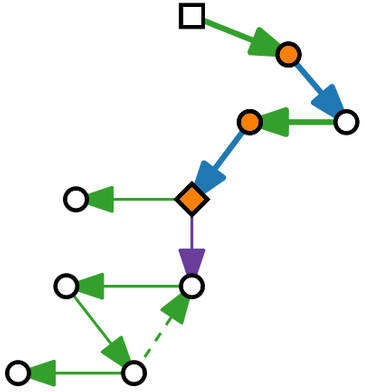
Eigenschaften optimaler Lösungen





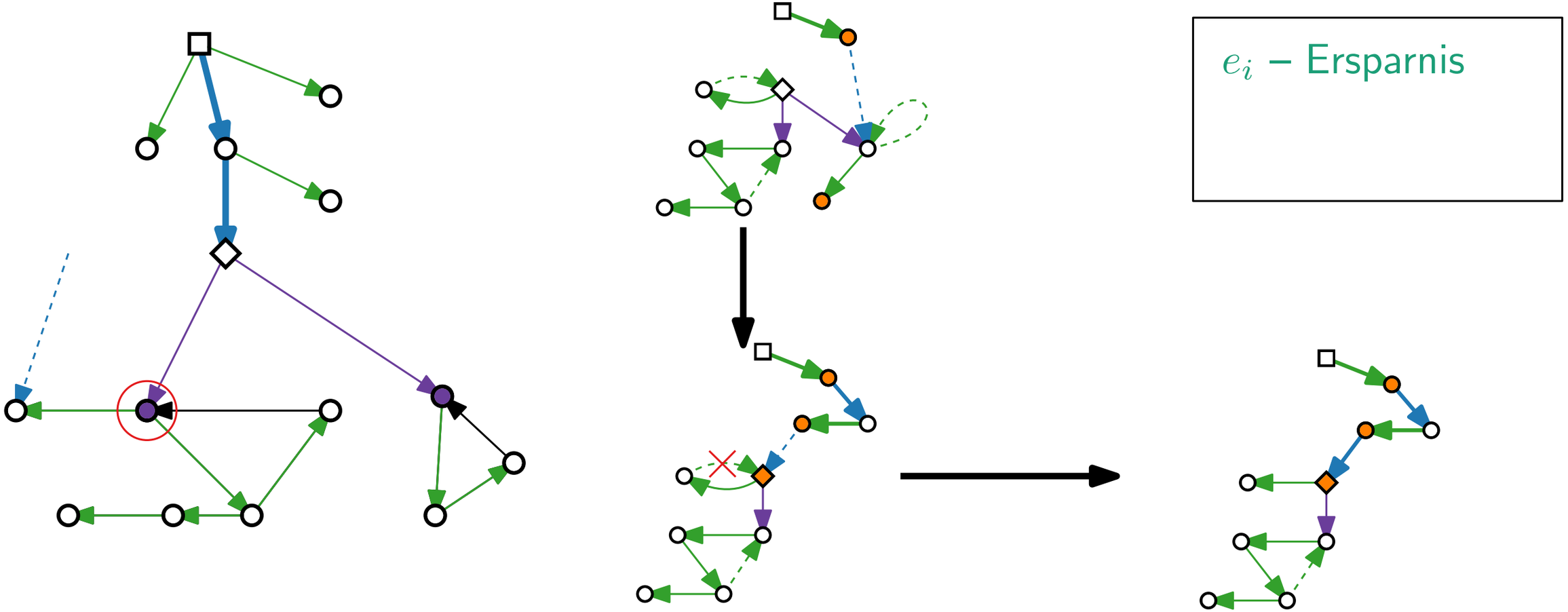


e_i – Ersparnis



Kante mit minimalen Kosten in jeden Zyklus

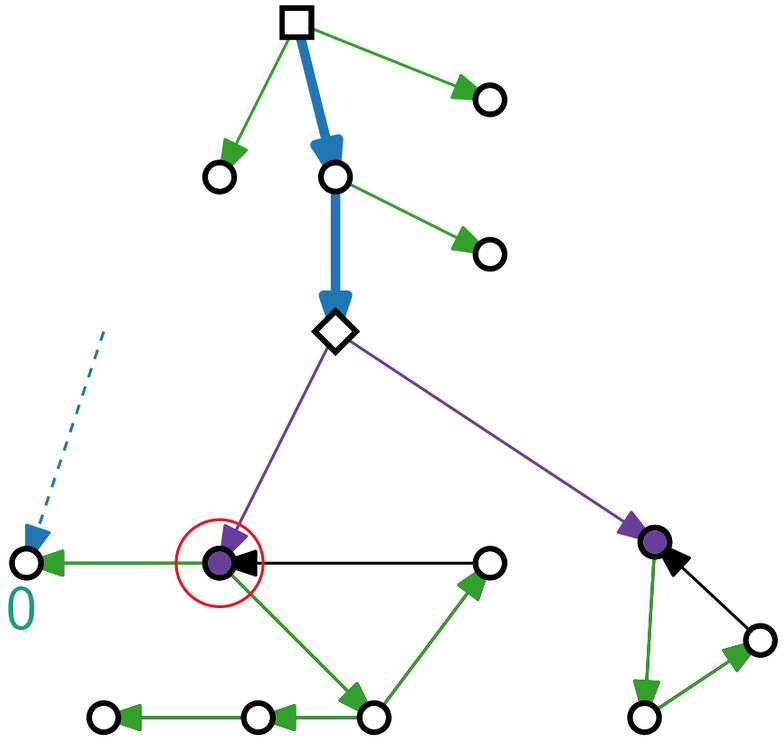
Eigenschaften optimaler Lösungen

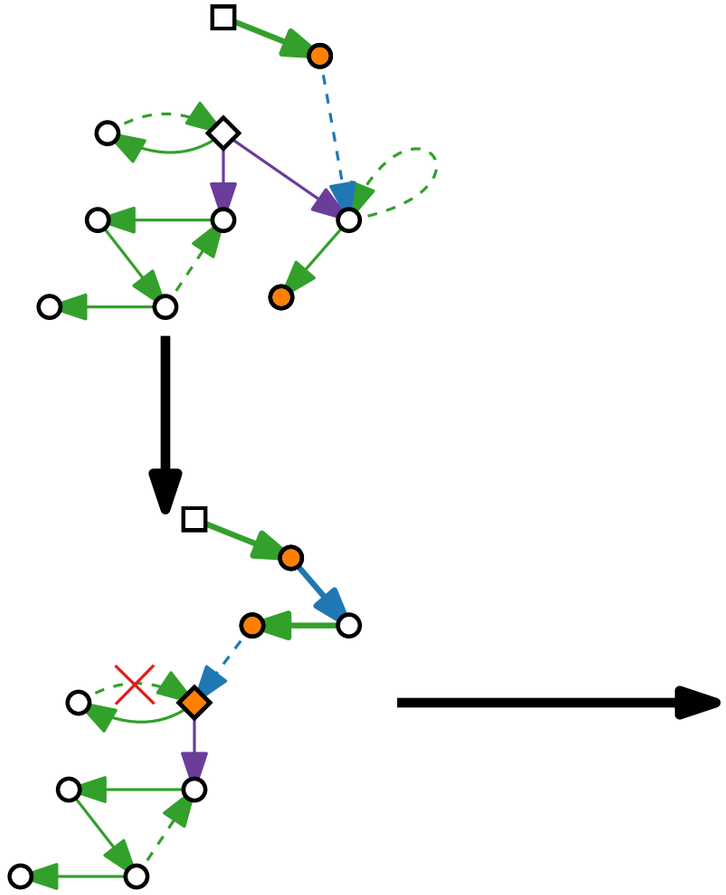


e_i – Ersparnis

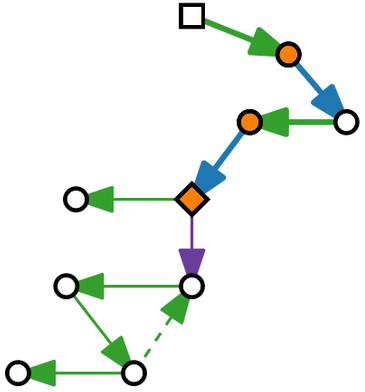
Kante mit minimalen Kosten in jeden Zyklus

Eigenschaften optimaler Lösungen



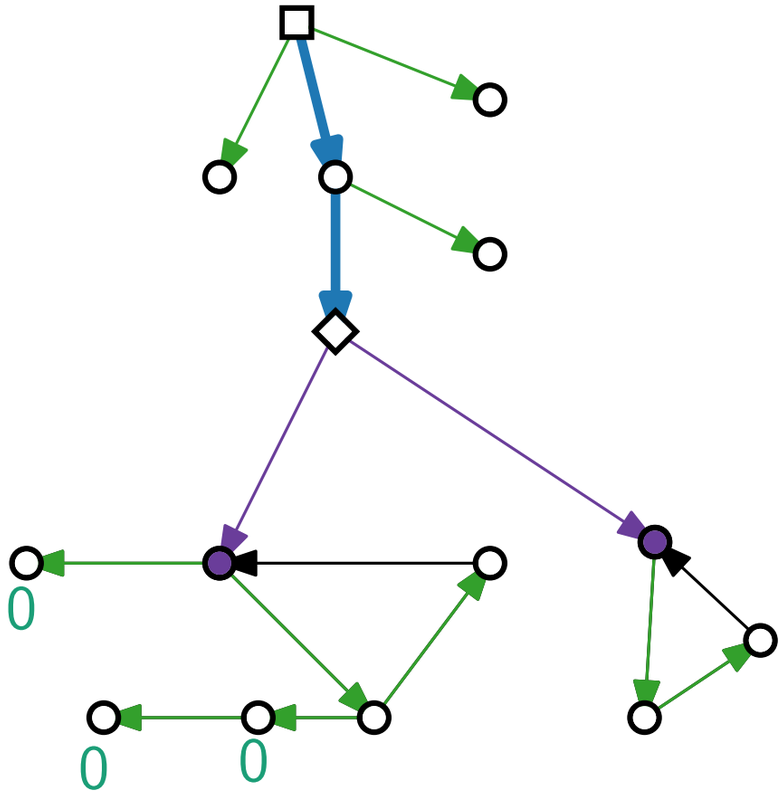


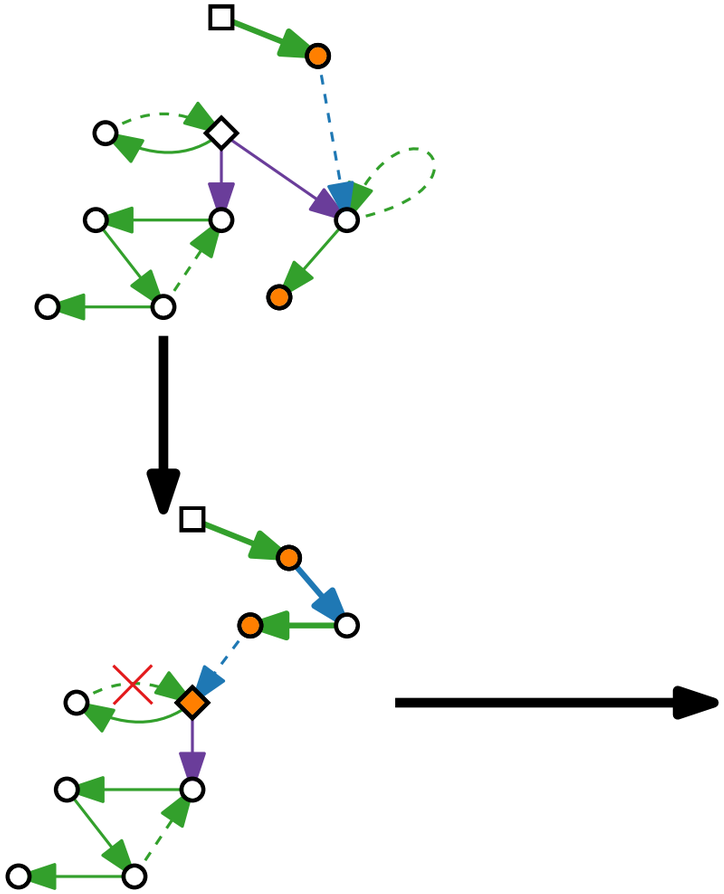
e_i – Ersparnis



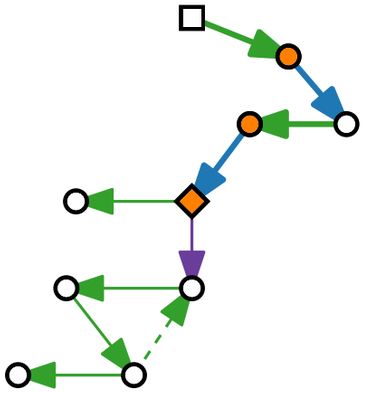
Kante mit minimalen Kosten in jeden Zyklus

Eigenschaften optimaler Lösungen



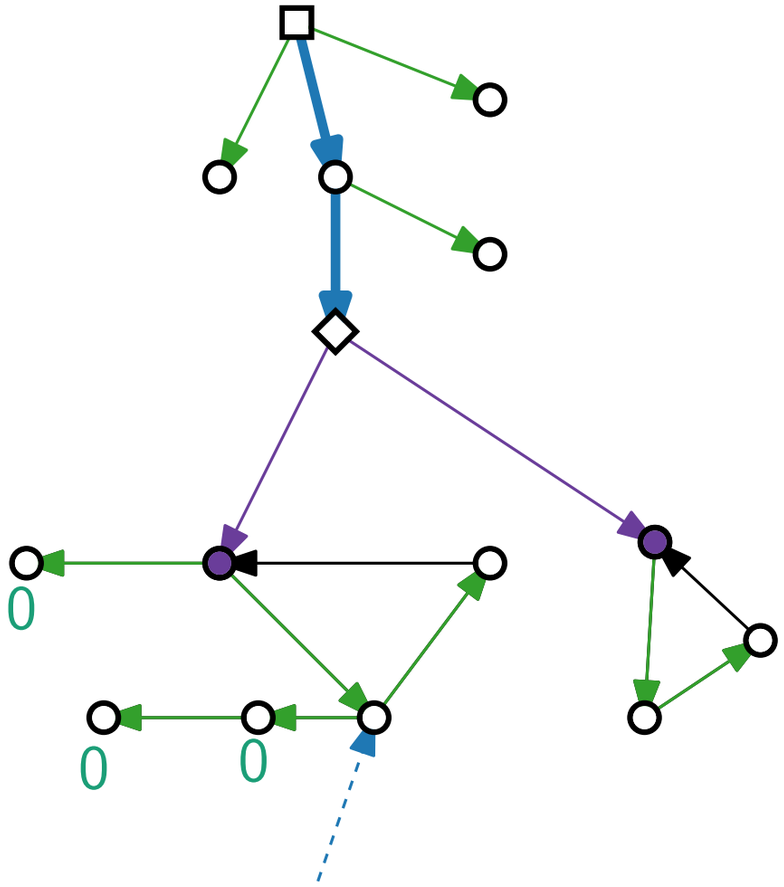


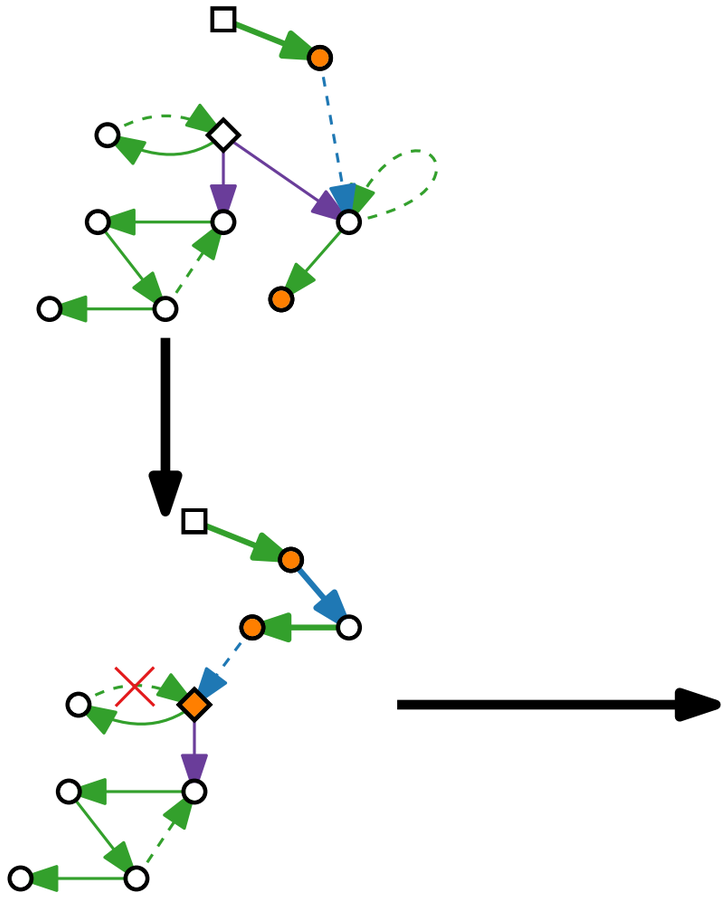
e_i – Ersparnis



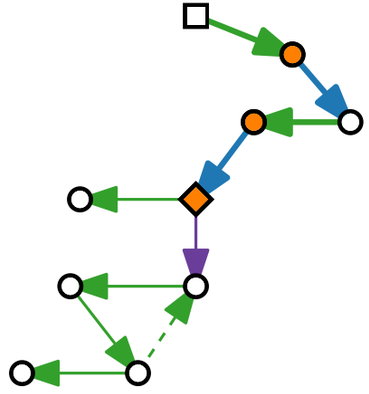
Kante mit minimalen Kosten in jeden Zyklus

Eigenschaften optimaler Lösungen



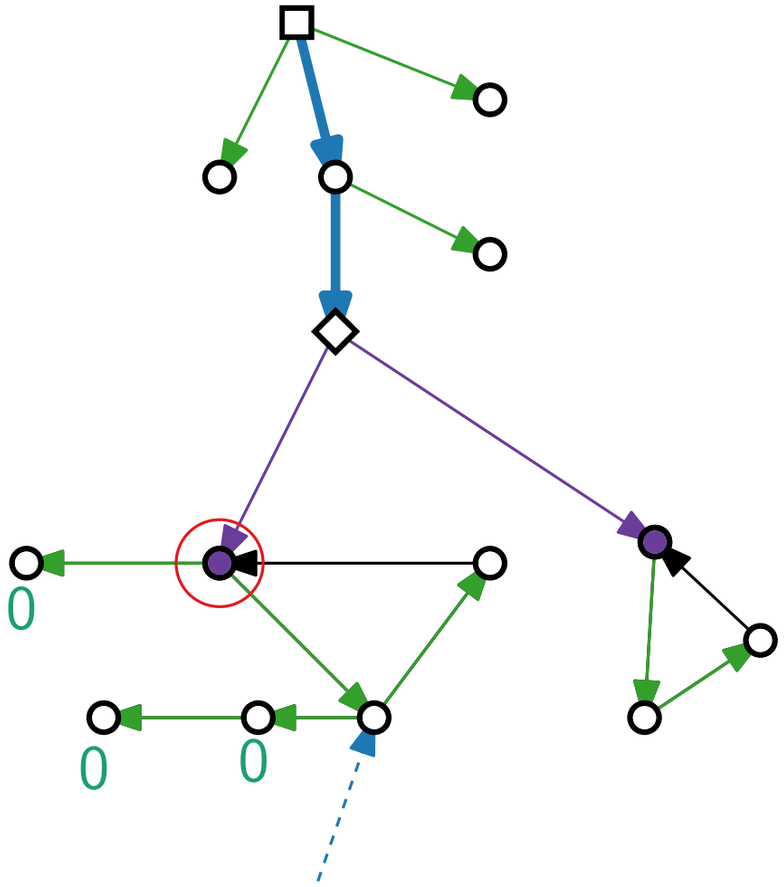


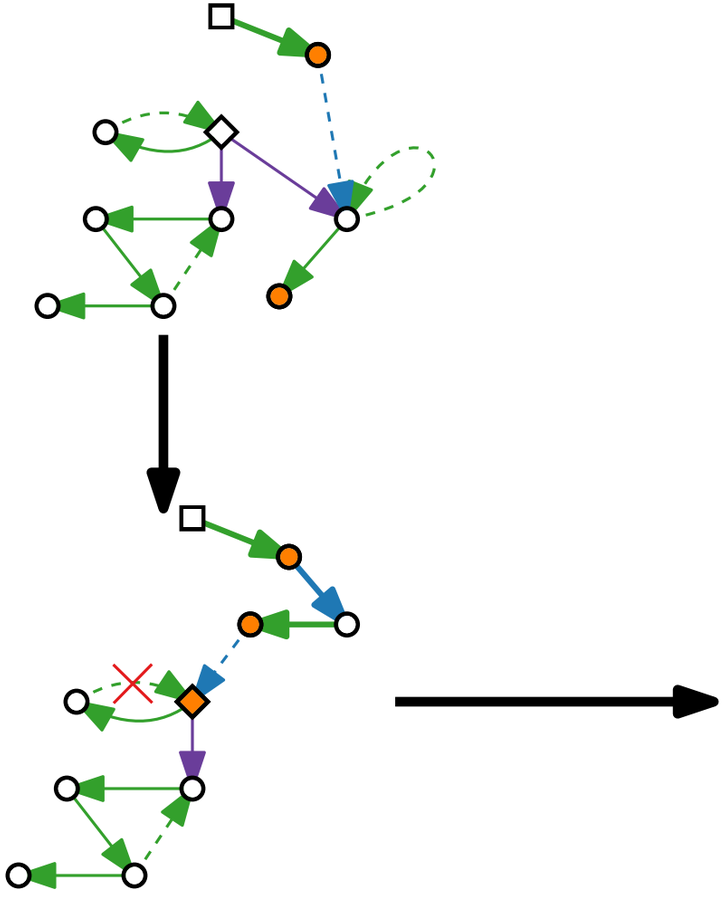
e_i – Ersparnis



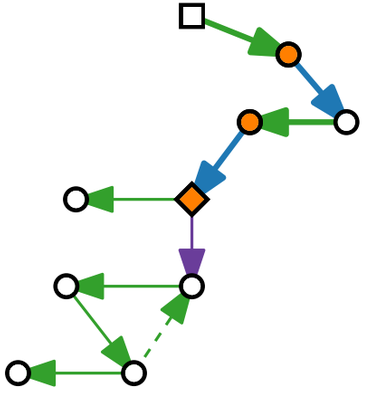
Kante mit minimalen Kosten in jeden Zyklus

Eigenschaften optimaler Lösungen



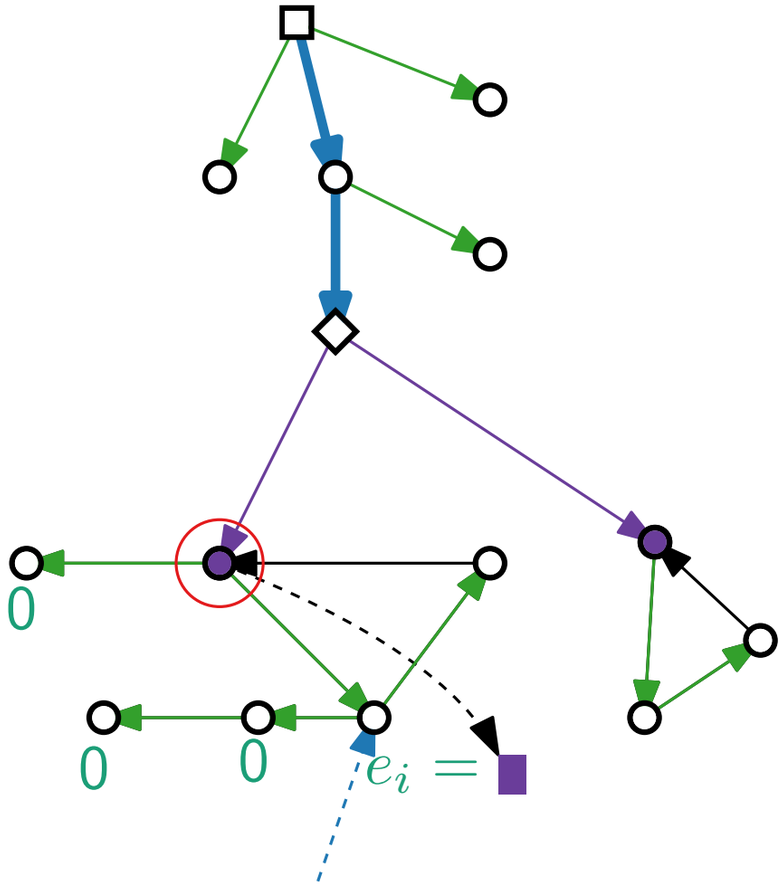


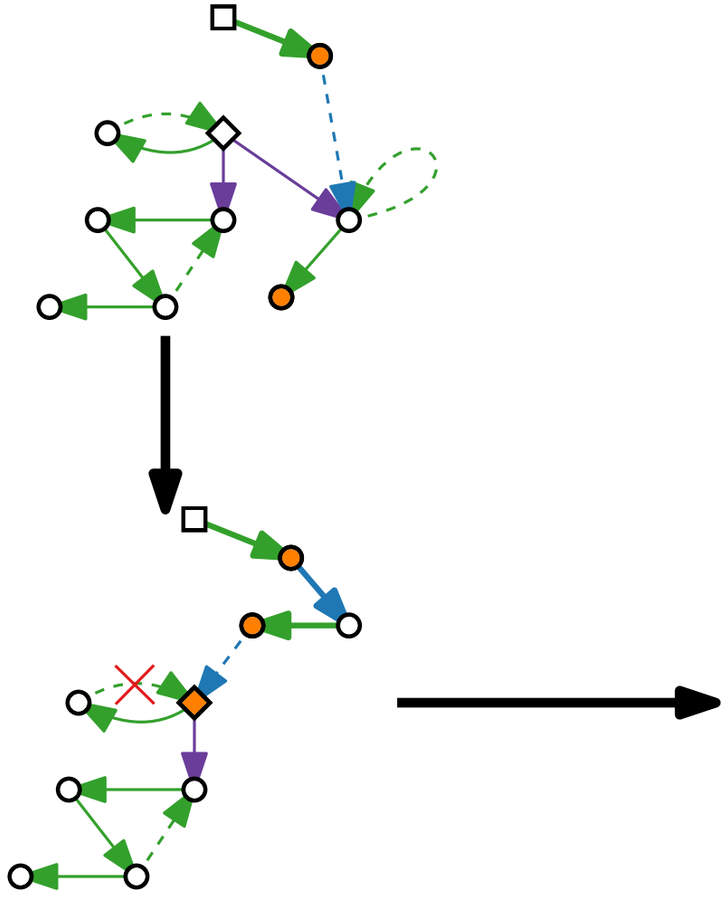
e_i – Ersparnis



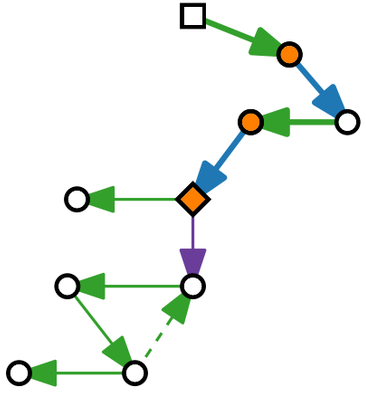
Kante mit minimalen Kosten in jeden Zyklus

Eigenschaften optimaler Lösungen



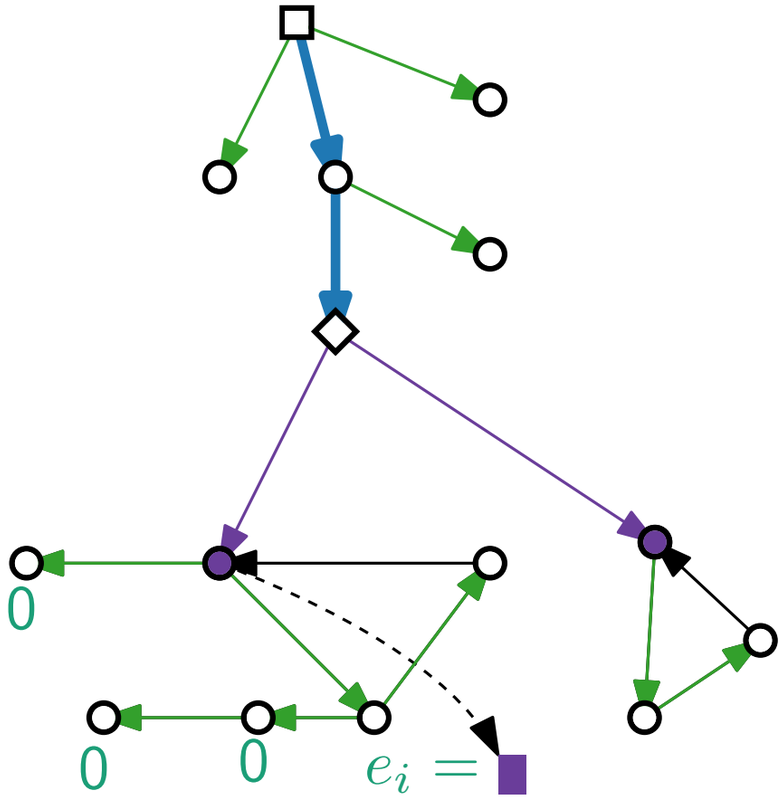


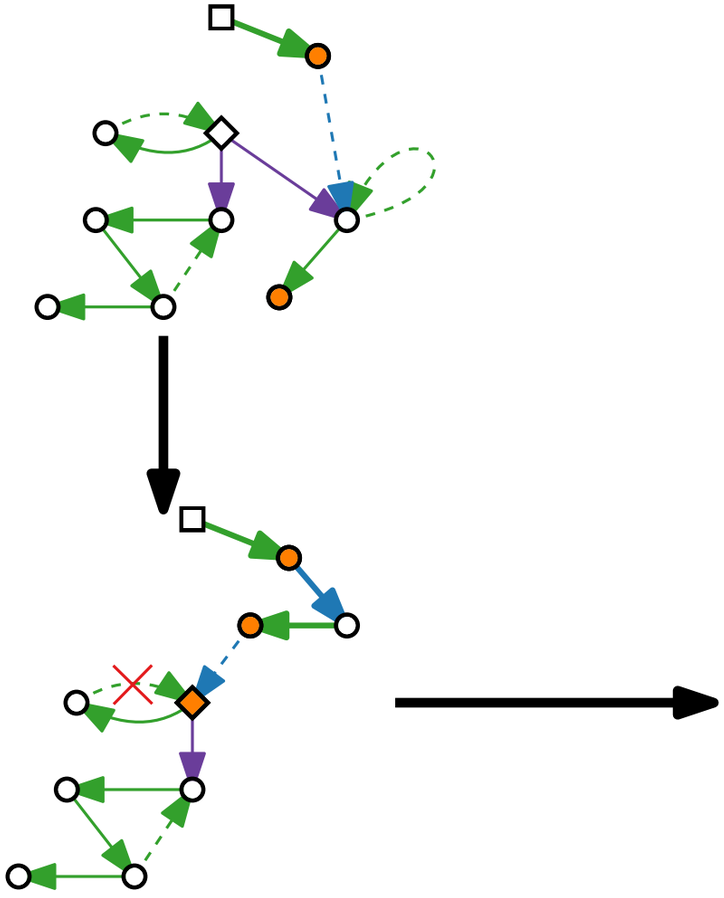
e_i – Ersparnis



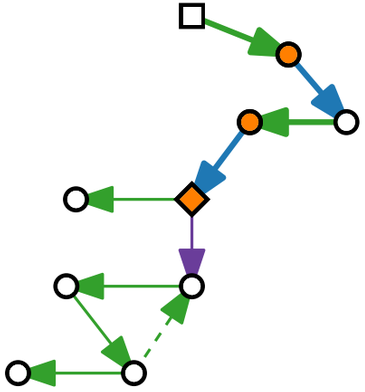
Kante mit minimalen Kosten in jeden Zyklus

Eigenschaften optimaler Lösungen



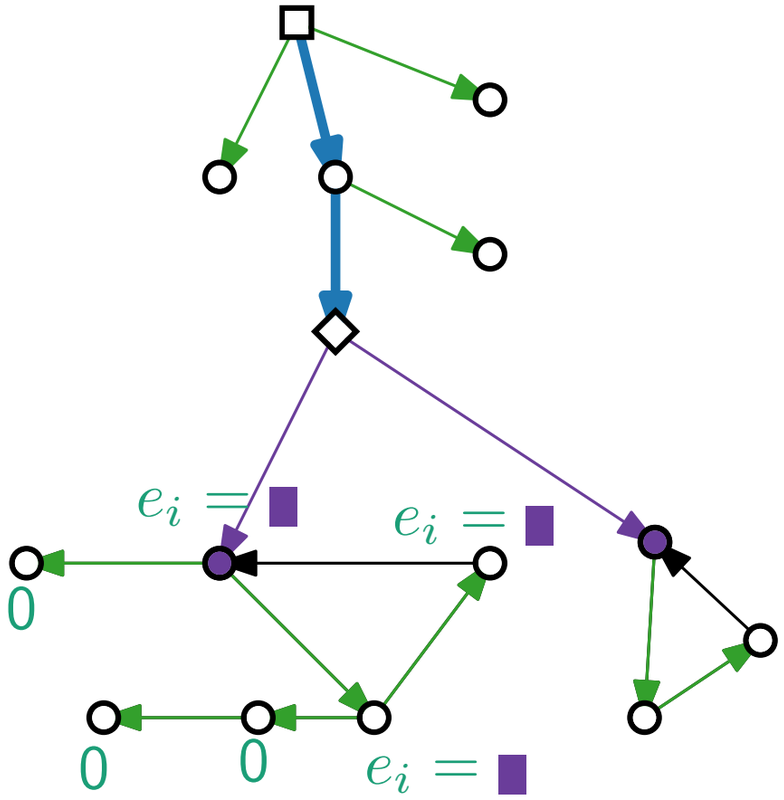


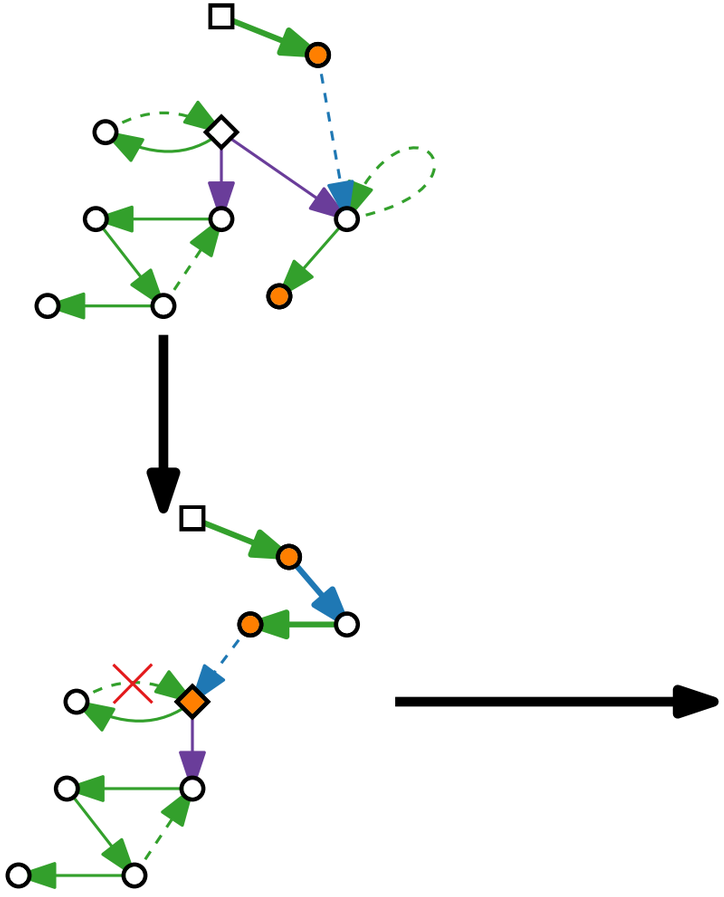
e_i – Ersparnis



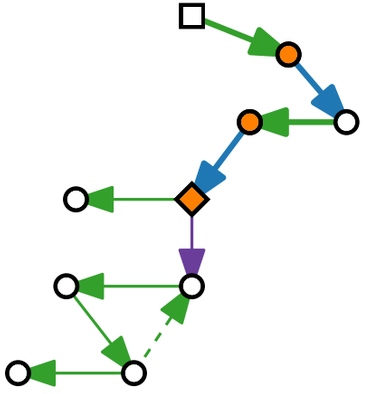
Kante mit minimalen Kosten in jeden Zyklus

Eigenschaften optimaler Lösungen



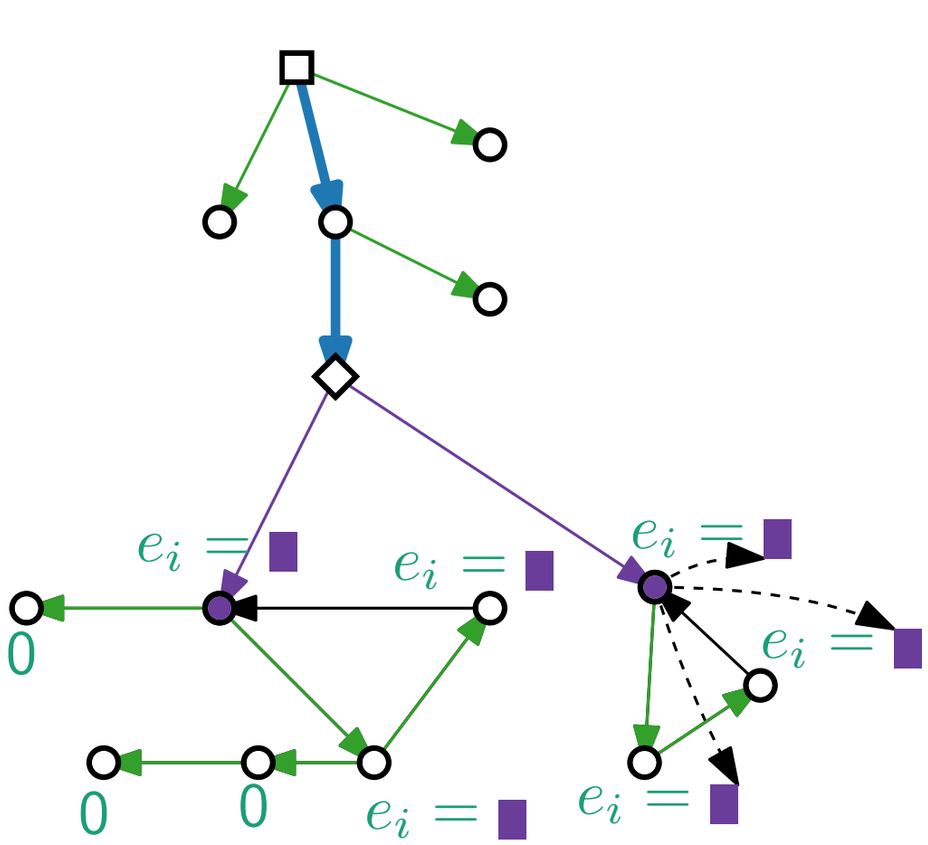


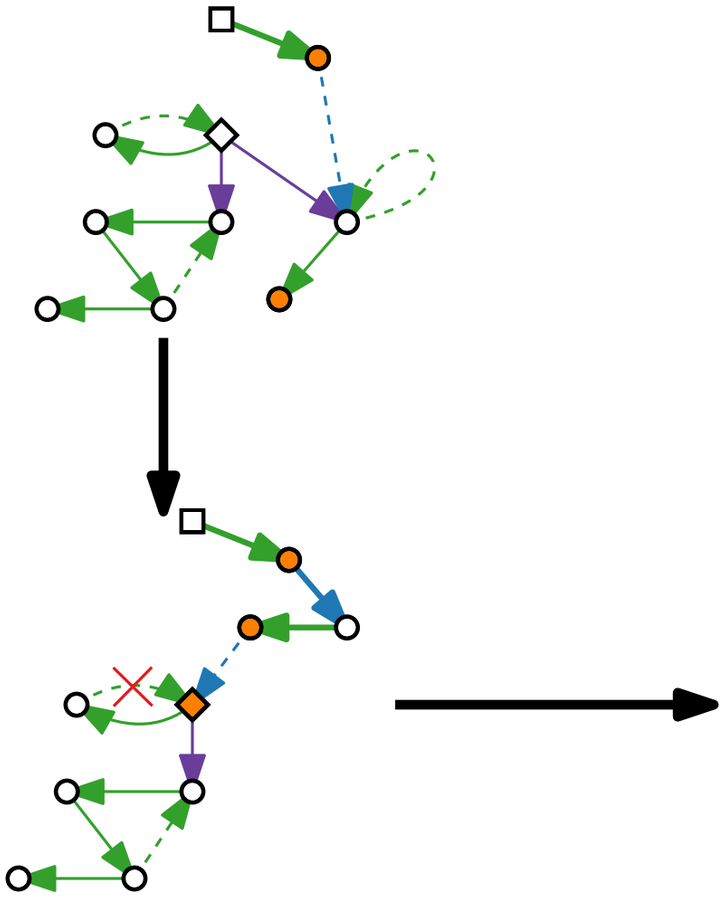
e_i – Ersparnis

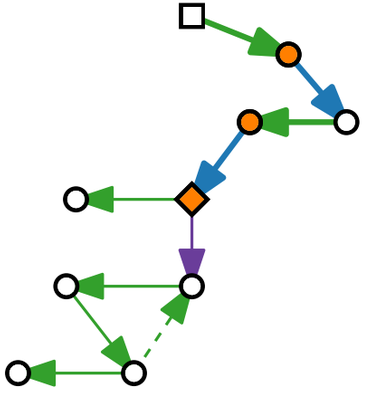


Kante mit minimalen Kosten in jeden Zyklus

Eigenschaften optimaler Lösungen



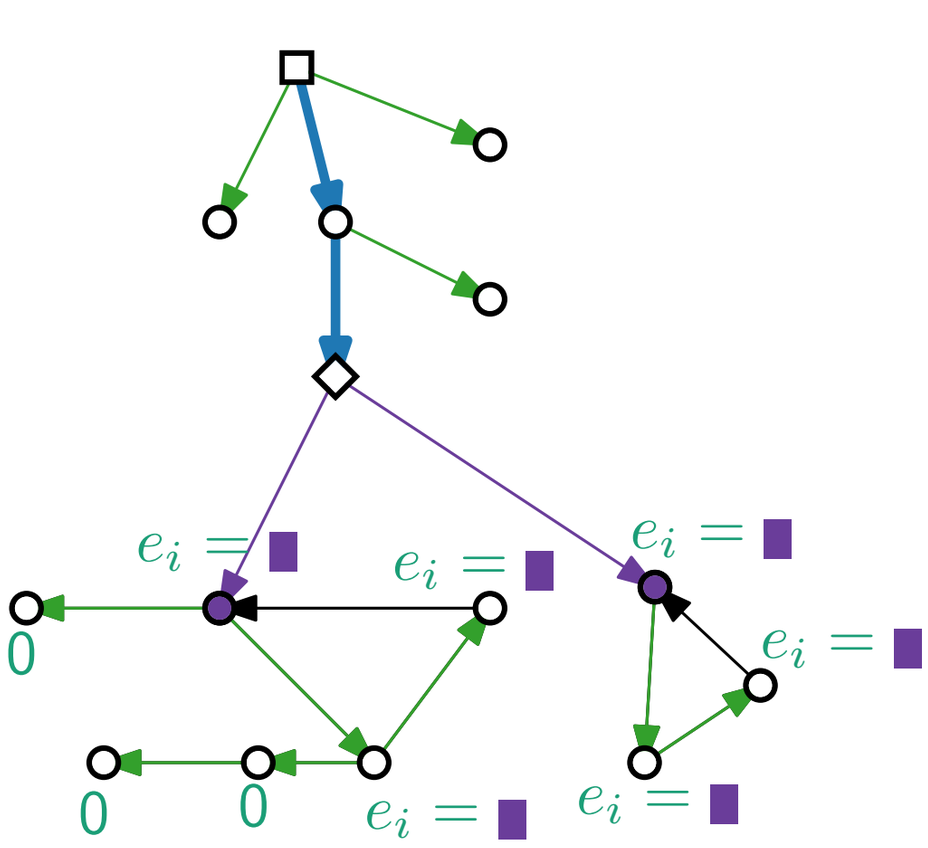


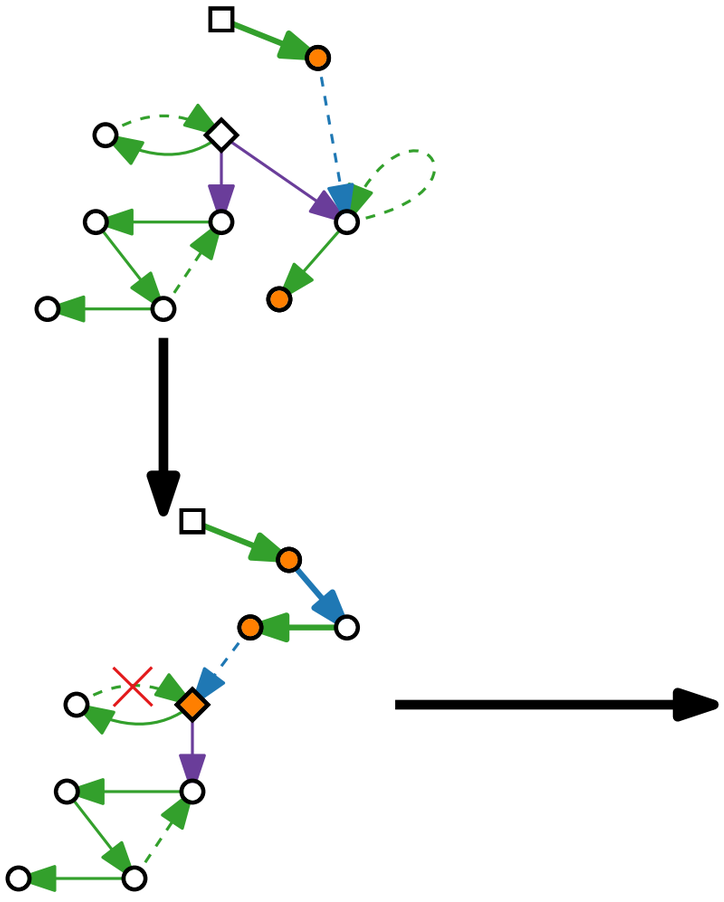


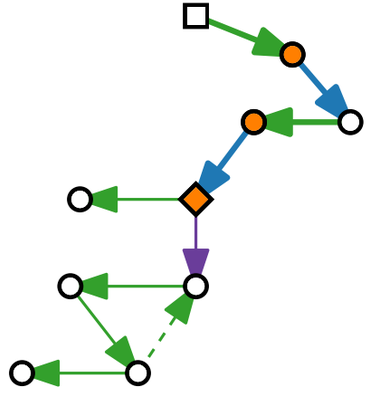
e_i – Ersparnis

Kante mit minimalen Kosten in jeden Zyklus

Eigenschaften optimaler Lösungen

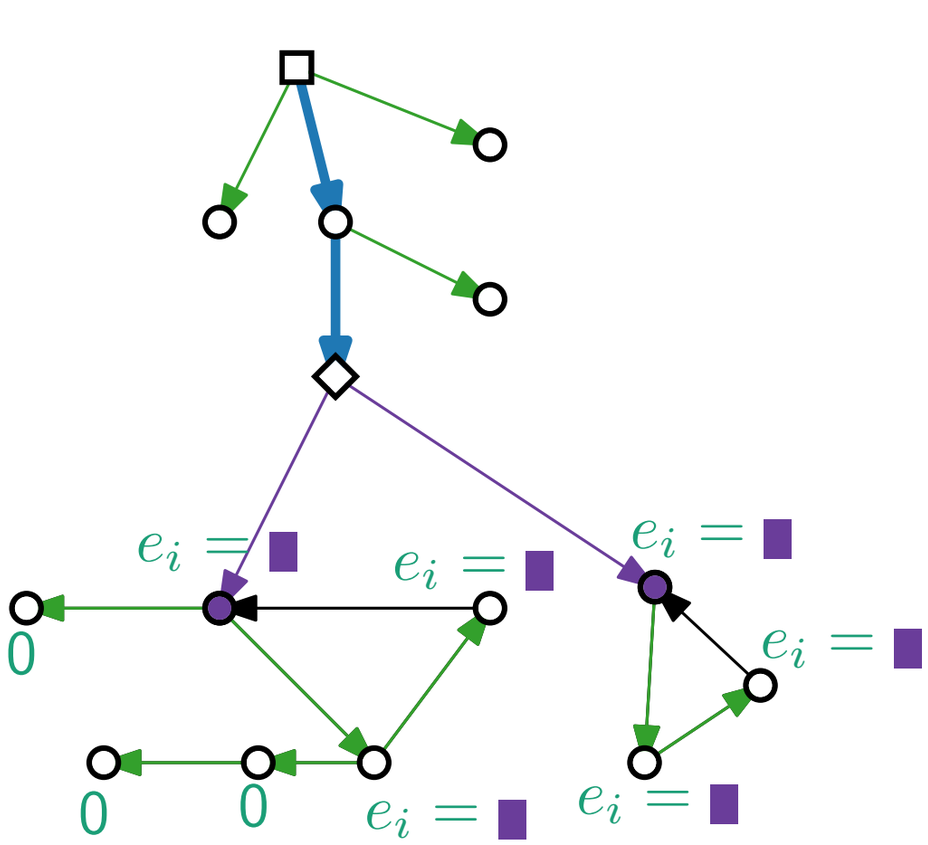


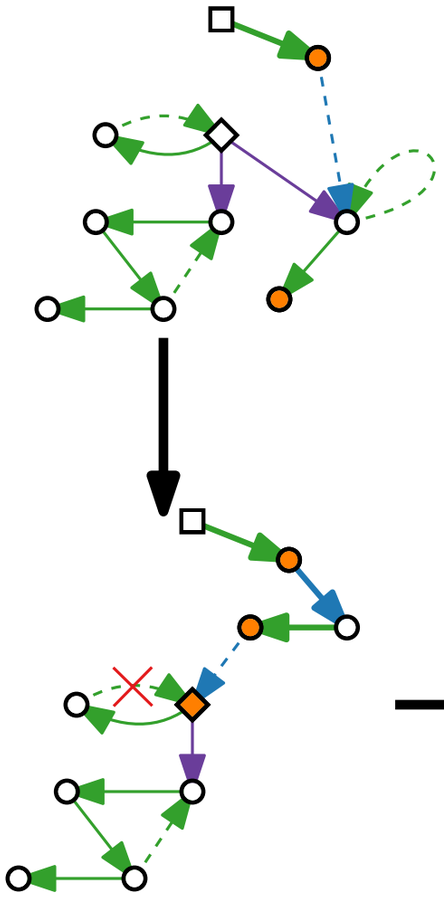


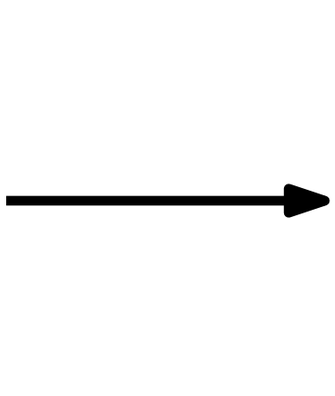


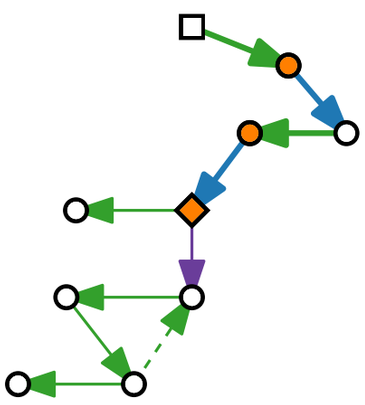
Kante mit minimalen Kosten in jeden Zyklus

Eigenschaften optimaler Lösungen









e_i – Ersparnis

Kante mit minimalen Kosten in jeden Zyklus

Berechnungsansatz

e_i – Ersparnis

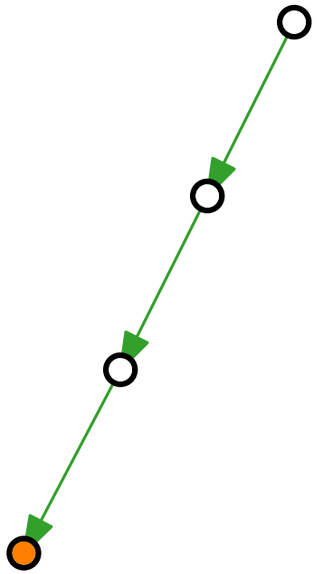
Berechnungsansatz

e_i – Ersparnis

Berechnungsansatz

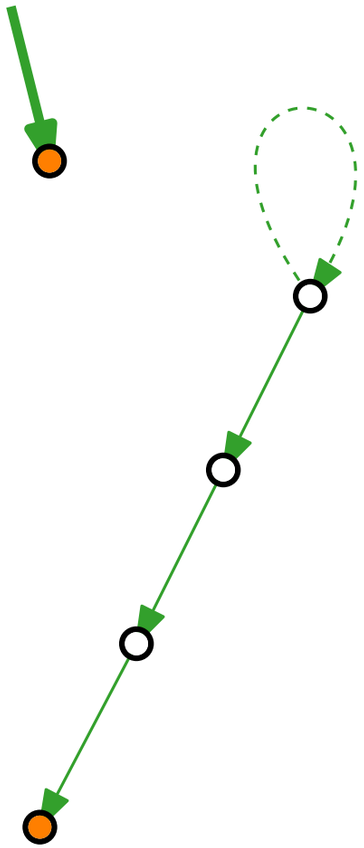
e_i – Ersparnis

Berechnungsansatz



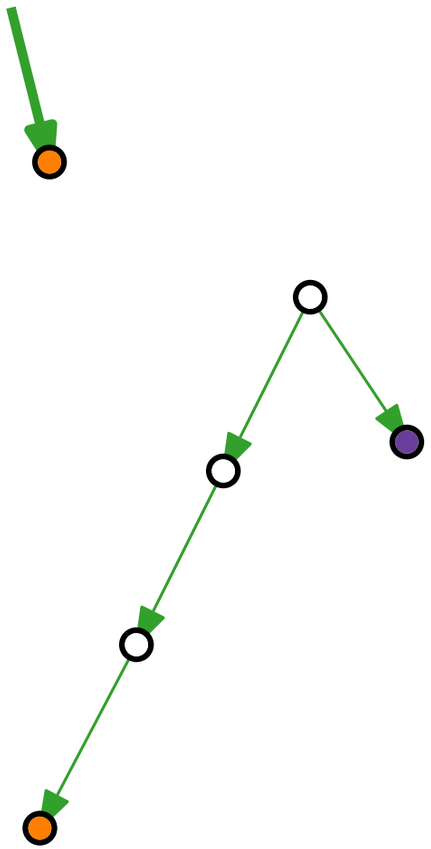
e_i – Ersparnis

Berechnungsansatz



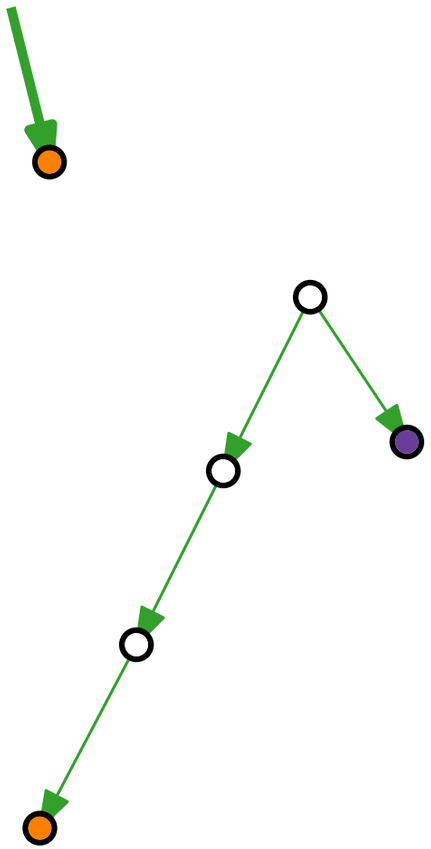
e_i – Ersparnis

Berechnungsansatz



e_i – Ersparnis

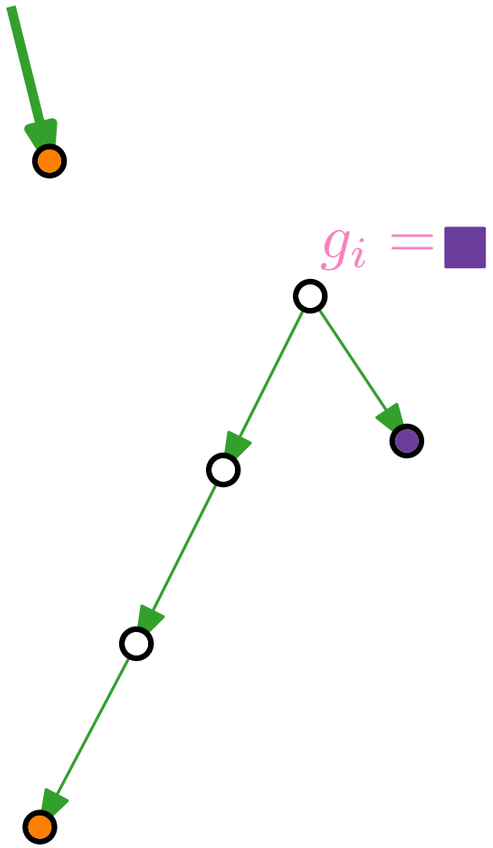
Berechnungsansatz



e_i – Ersparnis

g_i – größtes Kind

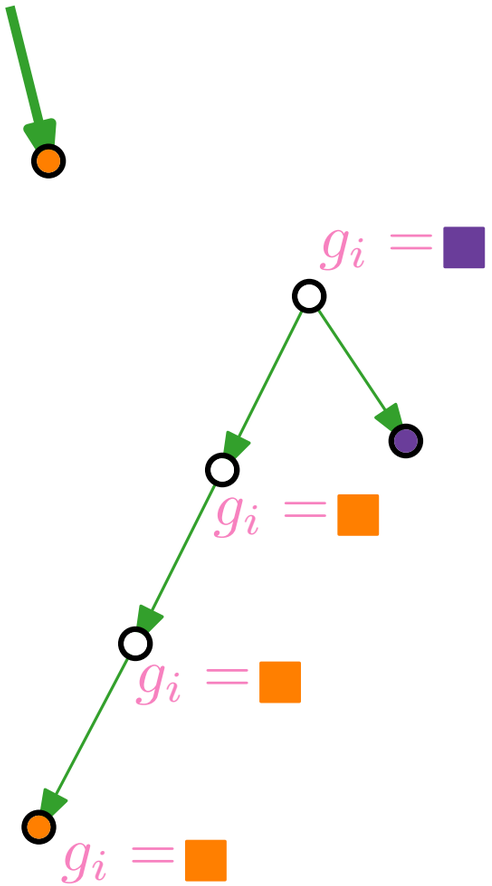
Berechnungsansatz



e_i – Ersparnis

g_i – größtes Kind

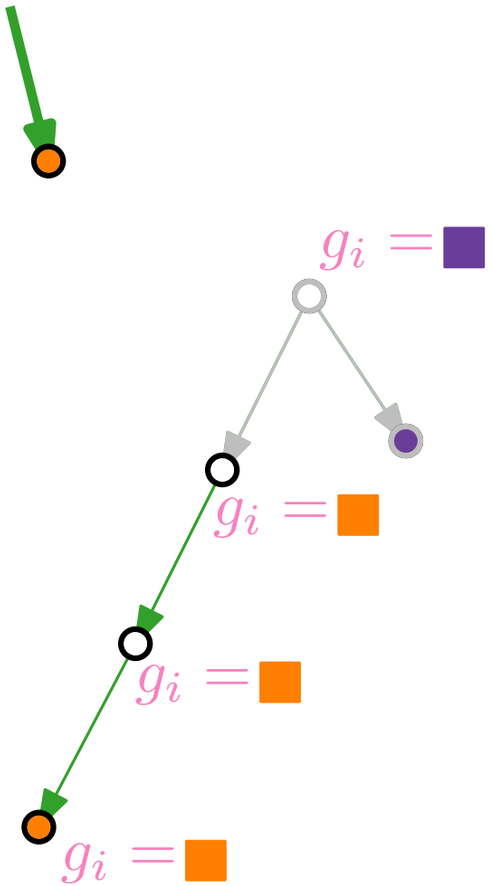
Berechnungsansatz



e_i – Ersparnis

g_i – größtes Kind

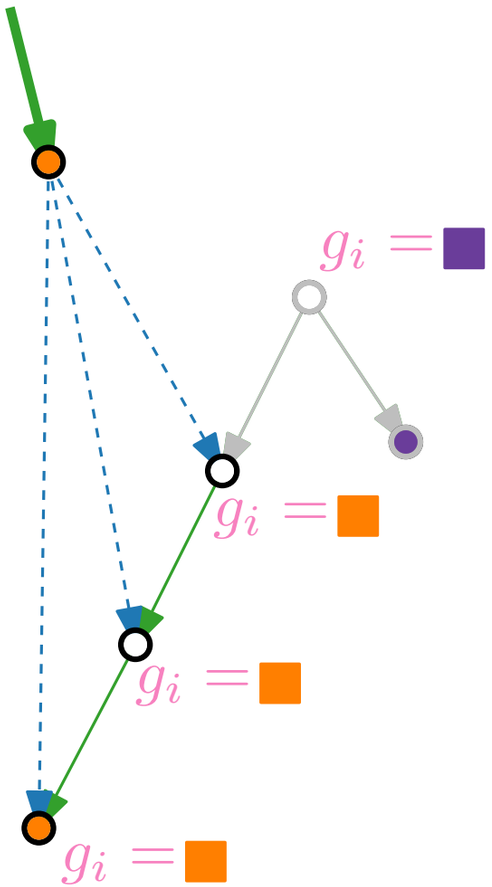
Berechnungsansatz



e_i – Ersparnis

g_i – größtes Kind

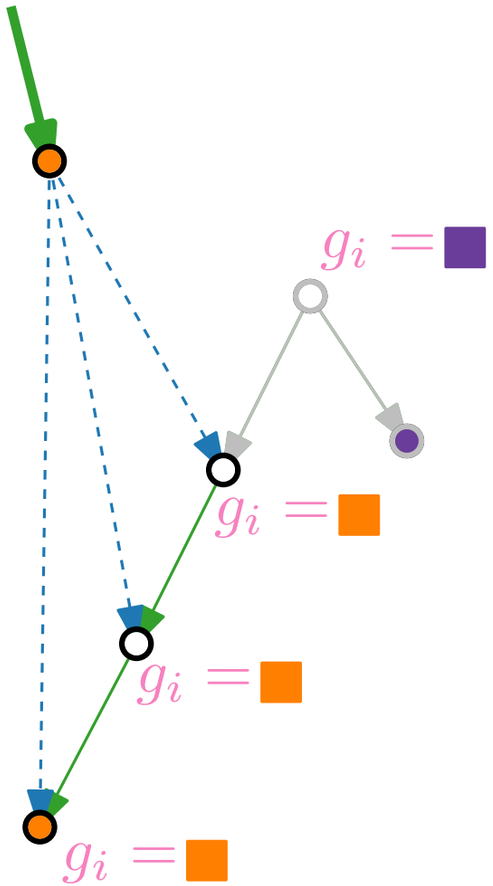
Berechnungsansatz



e_i – Ersparnis

g_i – größtes Kind

Berechnungsansatz

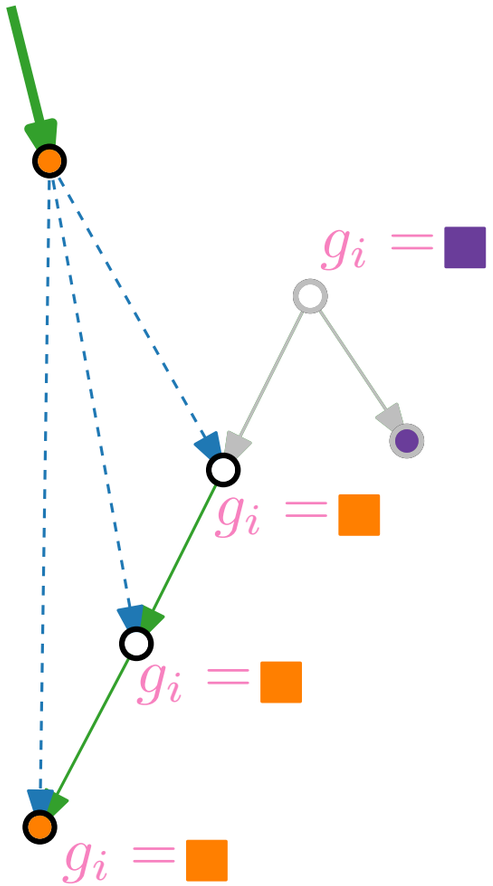


e_i – Ersparnis

g_i – größtes Kind

$$OPT_{i,j} = OPT_i + \min_{k:g_k=j} \{a_{k,i} - e_k\}$$

Berechnungsansatz



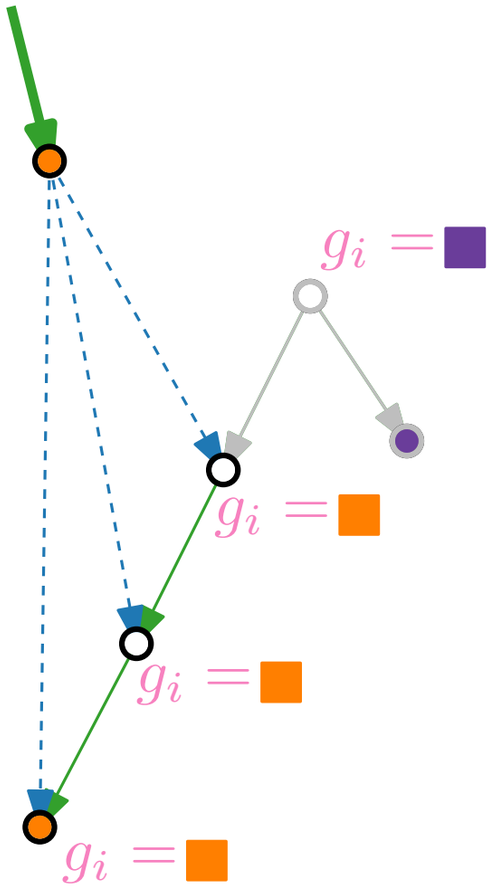
e_i – Ersparnis

g_i – größtes Kind

$$OPT_{i,j} = OPT_i + \min_{k:g_k=j} \{a_{k,i} - e_k\}$$

$$OPT_j = \min_{i<j \wedge i=g_i} \{OPT_{i,j}\}$$

Berechnungsansatz



e_i – Ersparnis

g_i – größtes Kind

$$OPT_{i,j} = OPT_i + \min_{k:g_k=j} \{a_{k,i} - e_k\}$$

$$OPT_j = \min_{i<j \wedge i=g_i} \{OPT_{i,j}\}$$

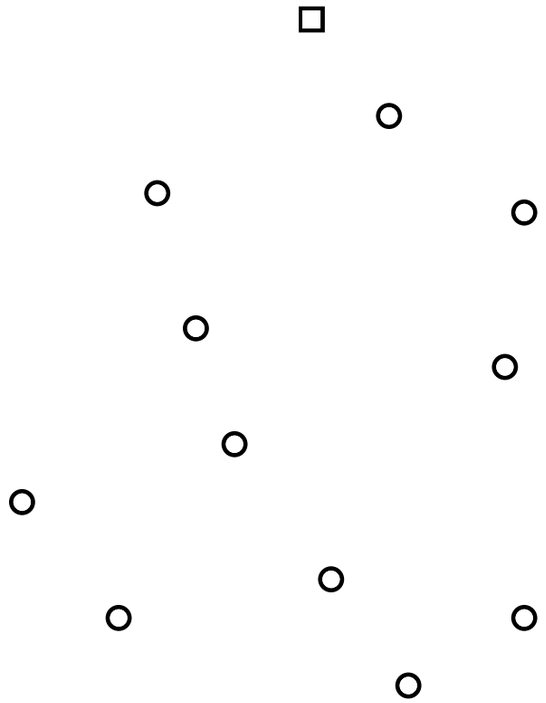
$$OPT = OPT_n$$

Berechnung Hilfsvariablen

e_i – Ersparnis

g_i – größtes Kind

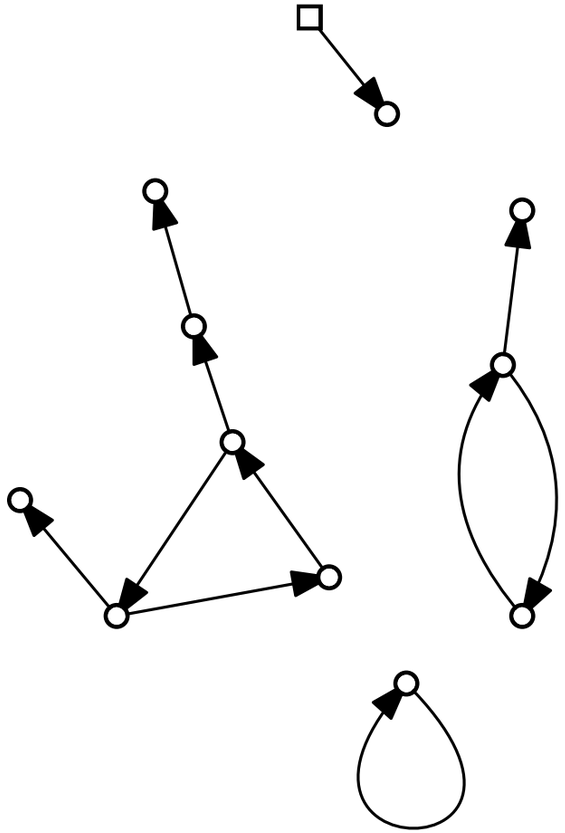
Berechnung Hilfsvariablen



e_i – Ersparnis

g_i – größtes Kind

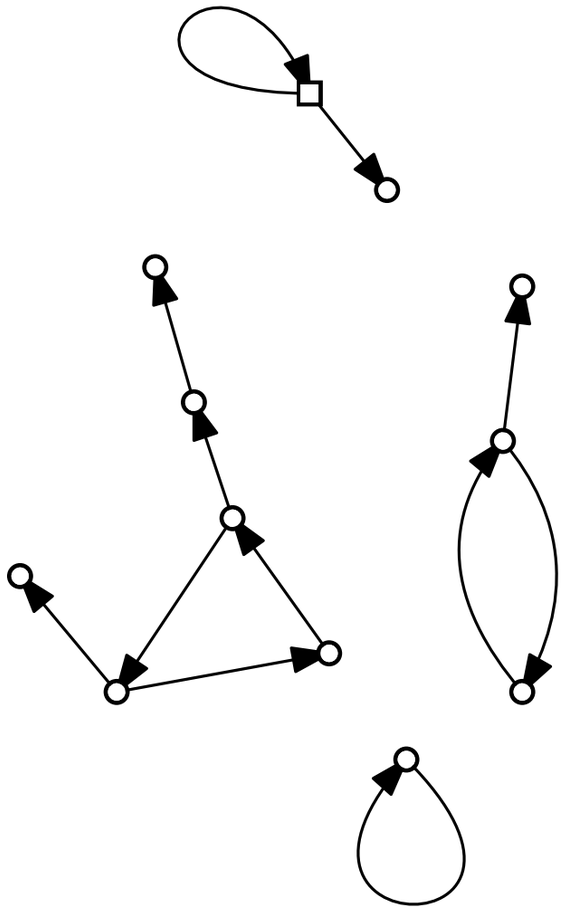
Berechnung Hilfsvariablen



e_i – Ersparnis

g_i – größtes Kind

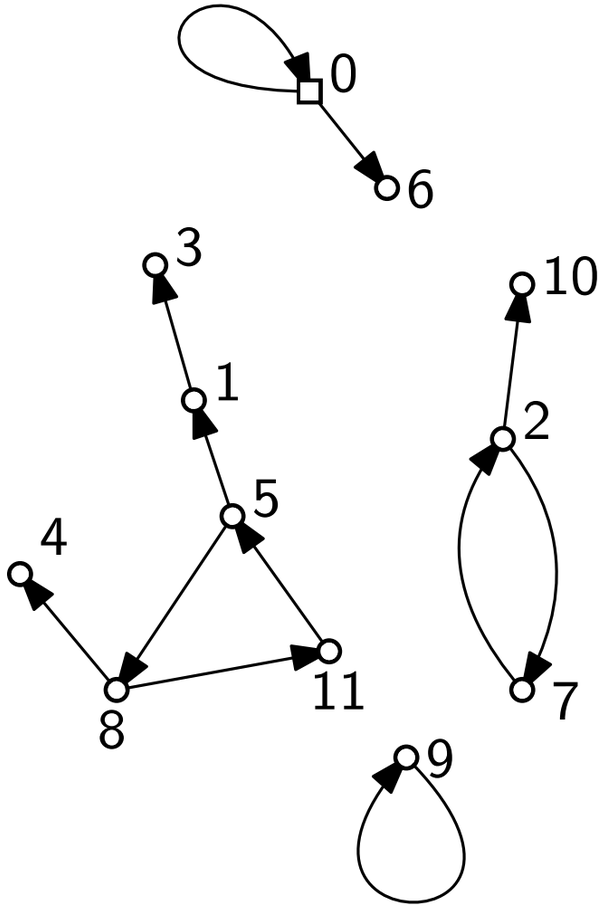
Berechnung Hilfsvariablen



e_i – Ersparnis

g_i – größtes Kind

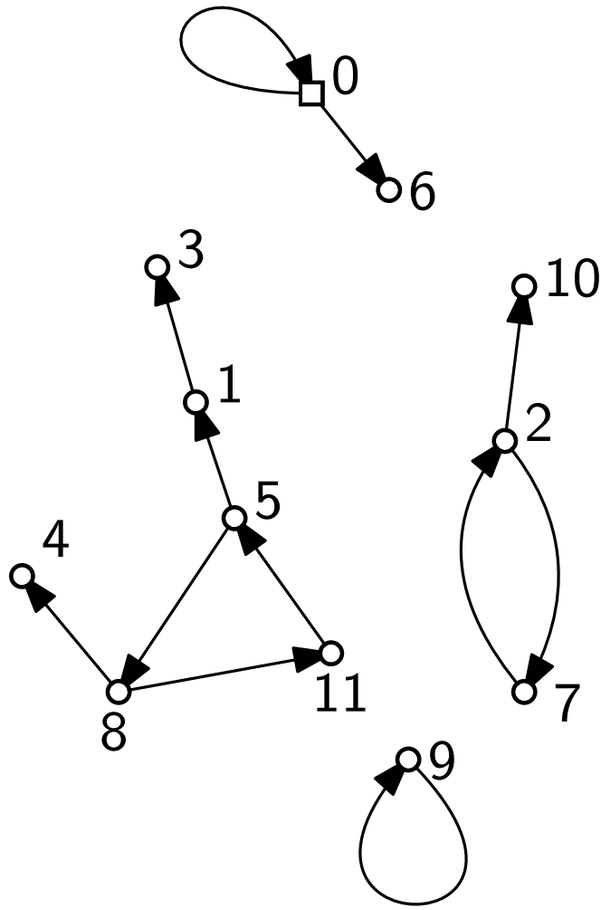
Berechnung Hilfsvariablen



e_i – Ersparnis

g_i – größtes Kind

Berechnung Hilfsvariablen

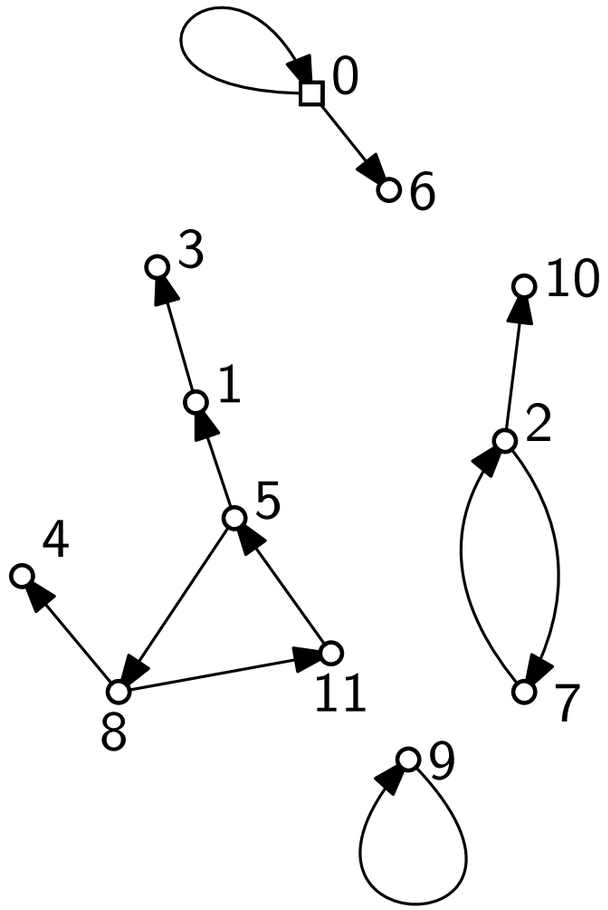


Knoten i	e_i	g_i
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		

e_i – Ersparnis

g_i – größtes Kind

Berechnung Hilfsvariablen

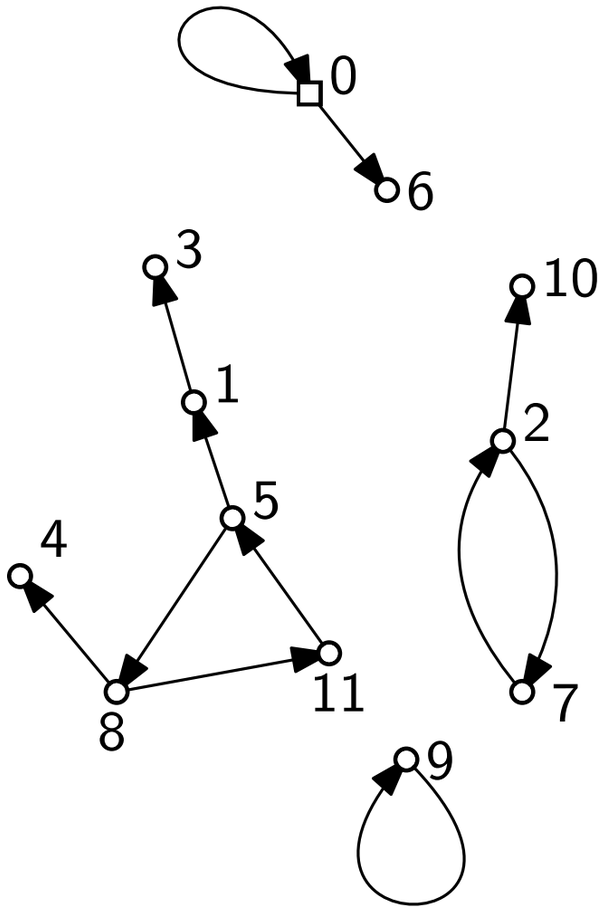


Knoten i	e_i	g_i
0	0	
1	0	
2	0	
3	0	
4	0	
5	0	
6	0	
7	0	
8	0	
9	0	
10	0	
11	0	

e_i – Ersparnis

g_i – größtes Kind

Berechnung Hilfsvariablen

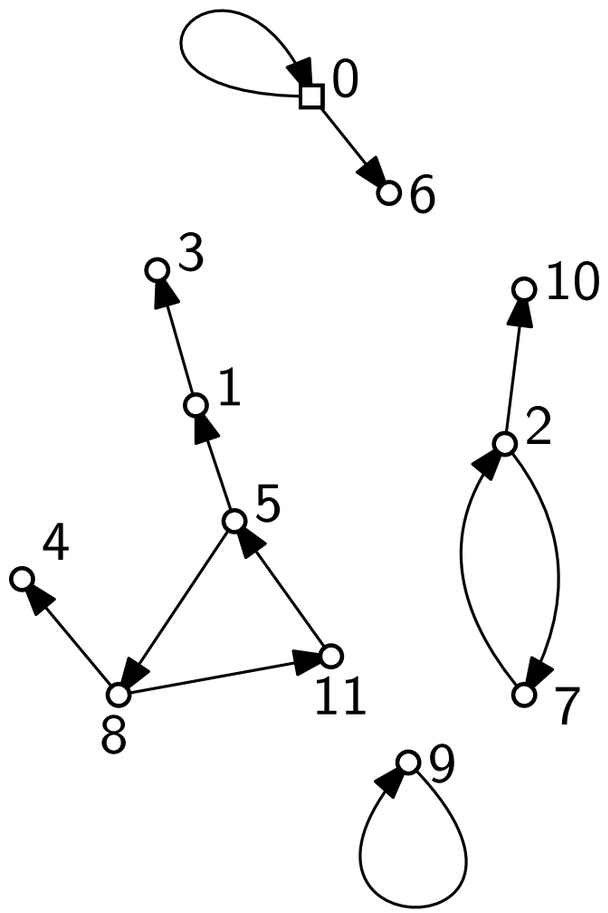


Knoten i	e_i	g_i
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	0	0
9	0	0
10	0	0
11	0	0

e_i – Ersparnis

g_i – größtes Kind

Berechnung Hilfsvariablen



Knoten i	e_i	g_i
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	0	0
9	0	0
10	0	0
11	0	0

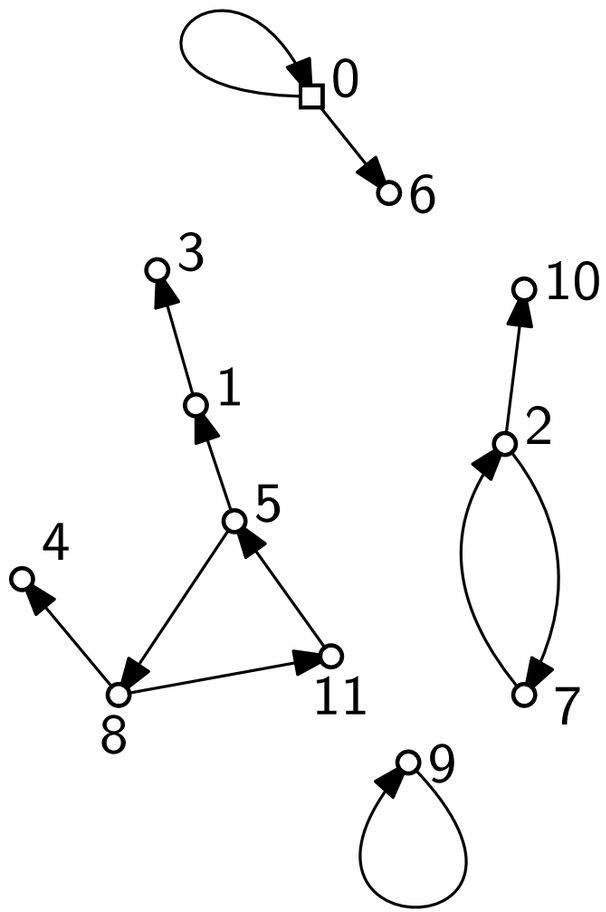
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**

Berechnung Hilfsvariablen



Knoten i	e_i	g_i
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	0	0
9	0	0
10	0	0
11	0	0

e_i – Ersparnis

g_i – größtes Kind

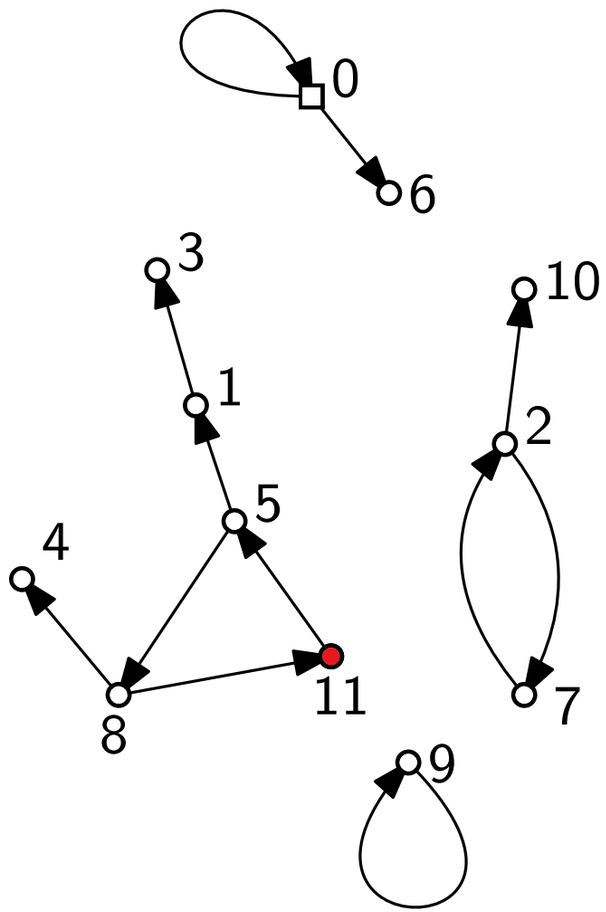
initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**

$j \leftarrow i$

while $g_j = 0$ **do**

Berechnung Hilfsvariablen



Knoten i	e_i	g_i
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	0	0
9	0	0
10	0	0
11	0	0

e_i – Ersparnis

g_i – größtes Kind

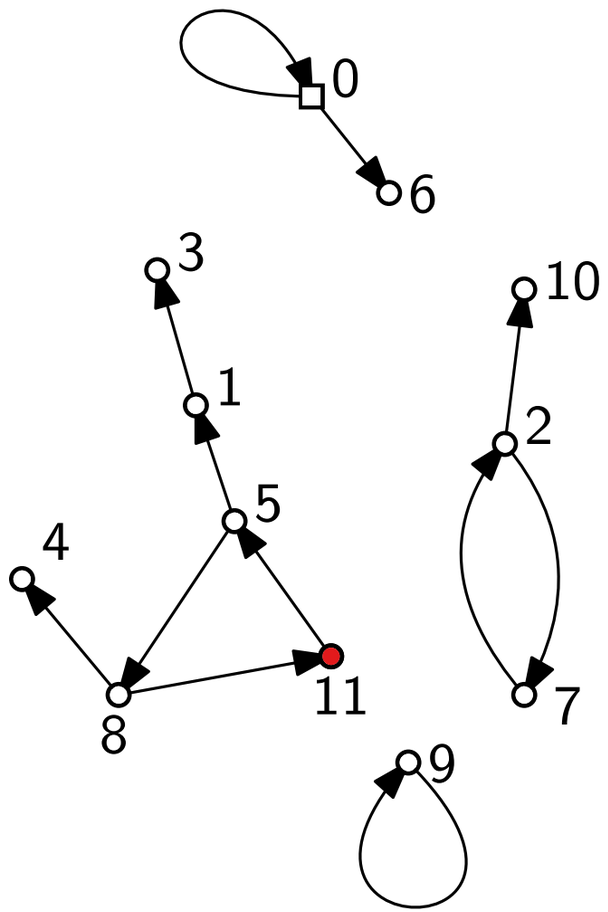
initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**

$j \leftarrow i$

while $g_j = 0$ **do**

Berechnung Hilfsvariablen



Knoten i	e_i	g_i
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	0	0
9	0	0
10	0	0
11	0	0

e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**

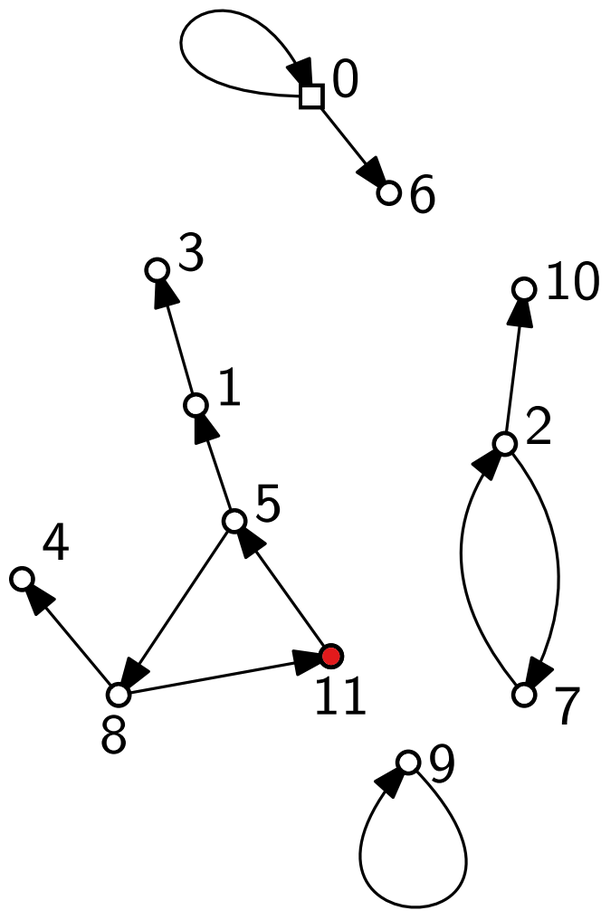
  $j \leftarrow i$

  while $g_j = 0$ **do**

    $g_j \leftarrow i$

    $j \leftarrow x_j$

Berechnung Hilfsvariablen



Knoten i	e_i	g_i
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	0	0
9	0	0
10	0	0
11	0	11

e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**

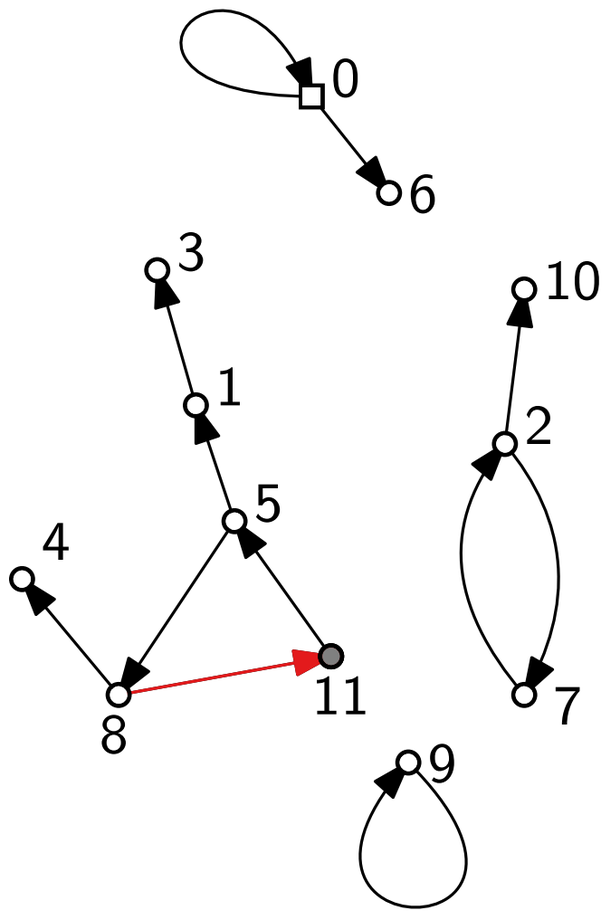
  $j \leftarrow i$

  while $g_j = 0$ **do**

    $g_j \leftarrow i$

    $j \leftarrow x_j$

Berechnung Hilfsvariablen



Knoten i	e_i	g_i
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	0	0
9	0	0
10	0	0
11	0	11

e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**

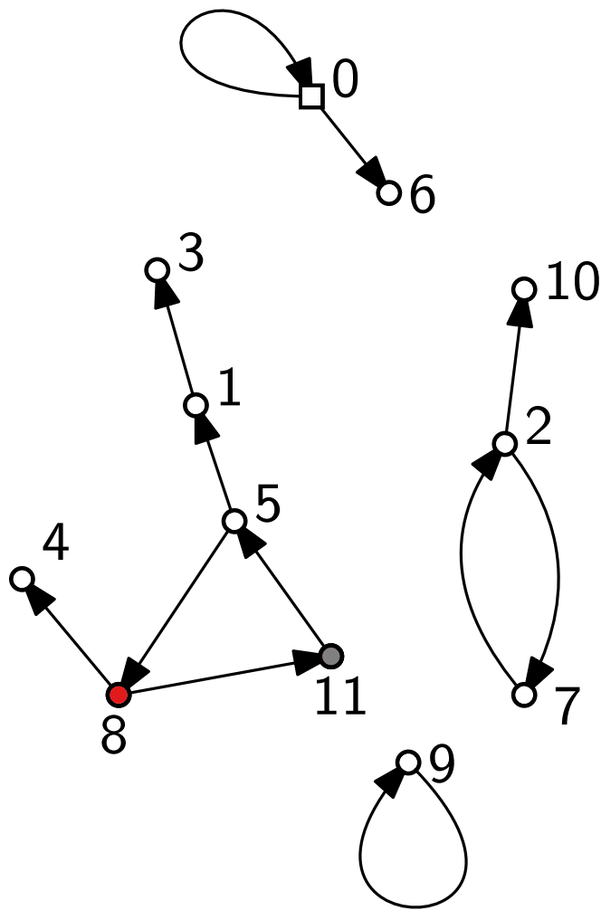
  $j \leftarrow i$

  while $g_j = 0$ **do**

    $g_j \leftarrow i$

    $j \leftarrow x_j$

Berechnung Hilfsvariablen



Knoten i	e_i	g_i
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	0	0
9	0	0
10	0	0
11	0	11

e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**

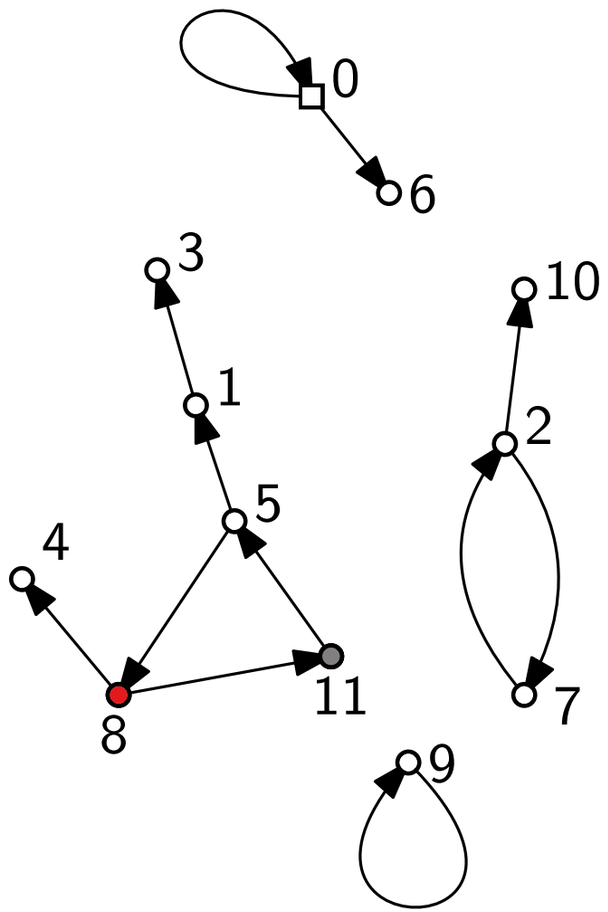
  $j \leftarrow i$

  while $g_j = 0$ **do**

    $g_j \leftarrow i$

    $j \leftarrow x_j$

Berechnung Hilfsvariablen



Knoten i	e_i	g_i
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	0	11
9	0	0
10	0	0
11	0	11

e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**

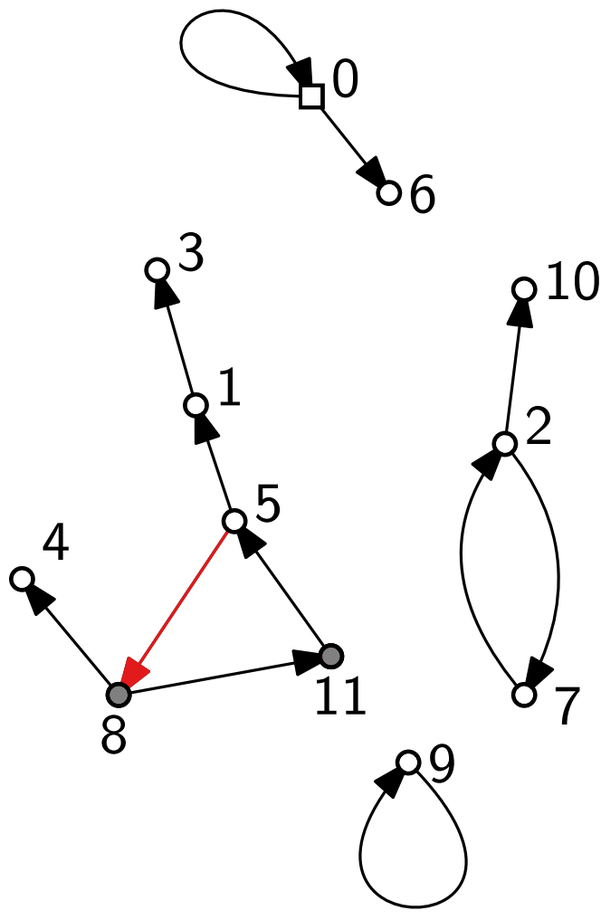
  $j \leftarrow i$

  while $g_j = 0$ **do**

    $g_j \leftarrow i$

    $j \leftarrow x_j$

Berechnung Hilfsvariablen



Knoten i	e_i	g_i
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	0	11
9	0	0
10	0	0
11	0	11

e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**

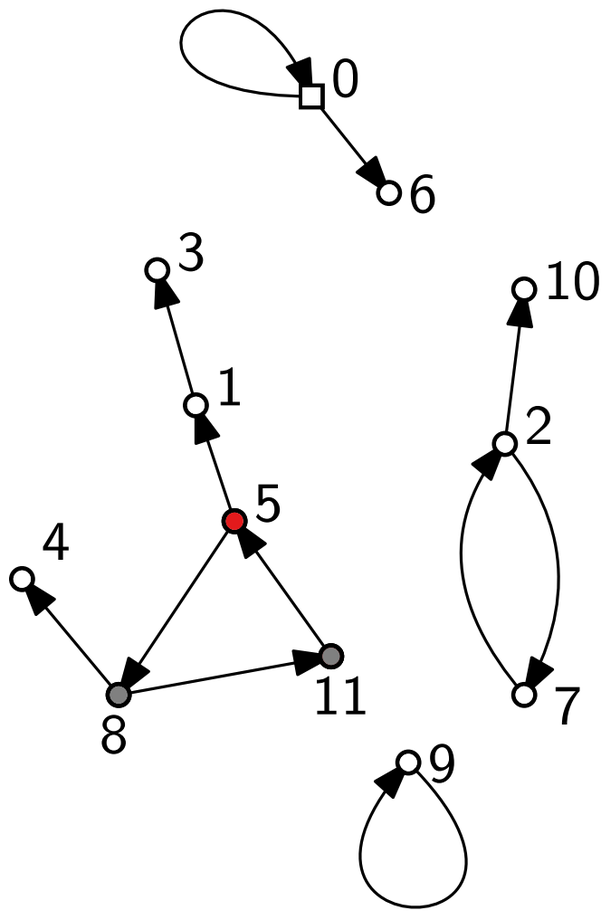
  $j \leftarrow i$

  while $g_j = 0$ **do**

    $g_j \leftarrow i$

    $j \leftarrow x_j$

Berechnung Hilfsvariablen



Knoten i	e_i	g_i
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	0	11
9	0	0
10	0	0
11	0	11

e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**

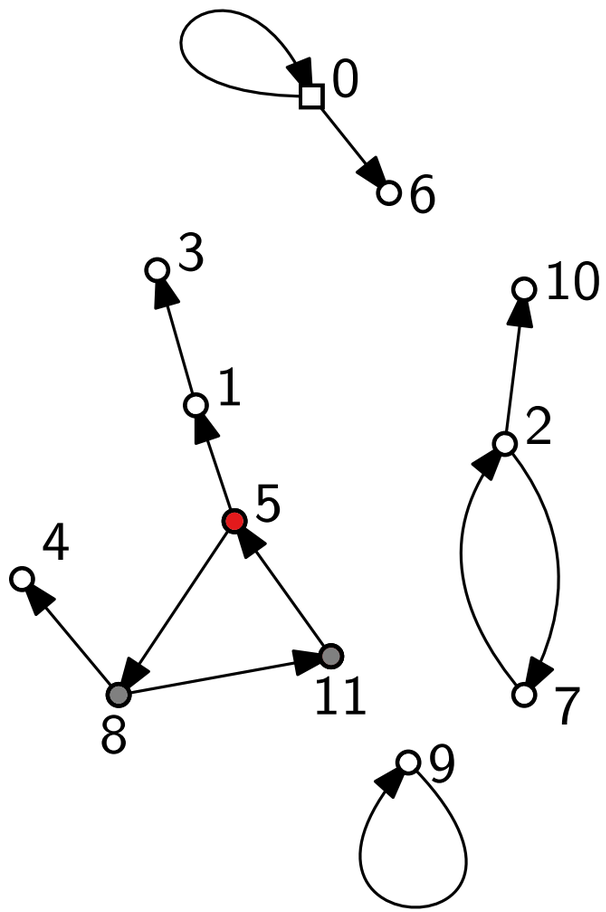
  $j \leftarrow i$

  while $g_j = 0$ **do**

    $g_j \leftarrow i$

    $j \leftarrow x_j$

Berechnung Hilfsvariablen



Knoten i	e_i	g_i
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	11
6	0	0
7	0	0
8	0	11
9	0	0
10	0	0
11	0	11

e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**

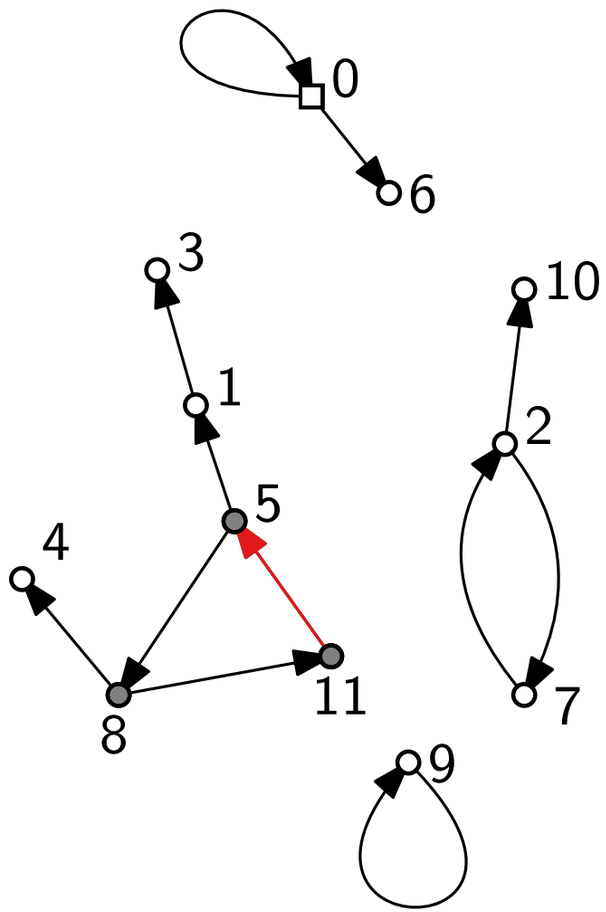
  $j \leftarrow i$

  while $g_j = 0$ **do**

    $g_j \leftarrow i$

    $j \leftarrow x_j$

Berechnung Hilfsvariablen



Knoten i	e_i	g_i
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	11
6	0	0
7	0	0
8	0	11
9	0	0
10	0	0
11	0	11

e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**

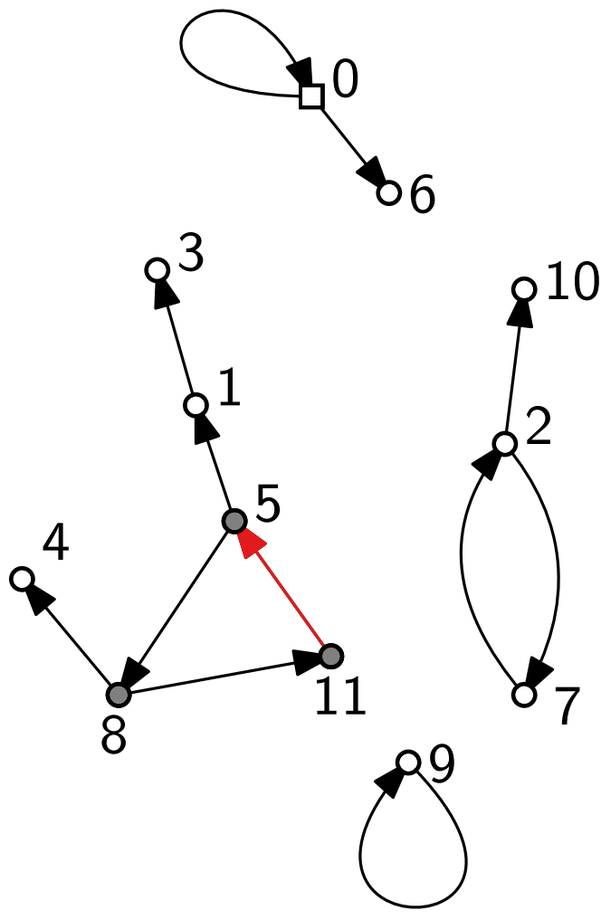
$j \leftarrow i$

while $g_j = 0$ **do**

$g_j \leftarrow i$

$j \leftarrow x_j$

Berechnung Hilfsvariablen



Knoten i	e_i	g_i
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	11
6	0	0
7	0	0
8	0	11
9	0	0
10	0	0
11	0	11

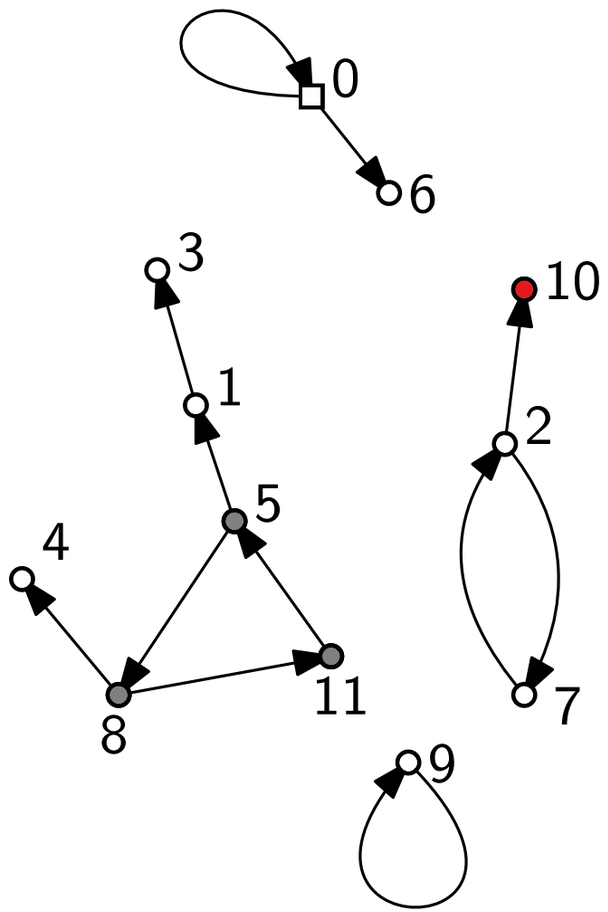
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**

Berechnung Hilfsvariablen



Knoten i	e_i	g_i
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	11
6	0	0
7	0	0
8	0	11
9	0	0
10	0	0
11	0	11

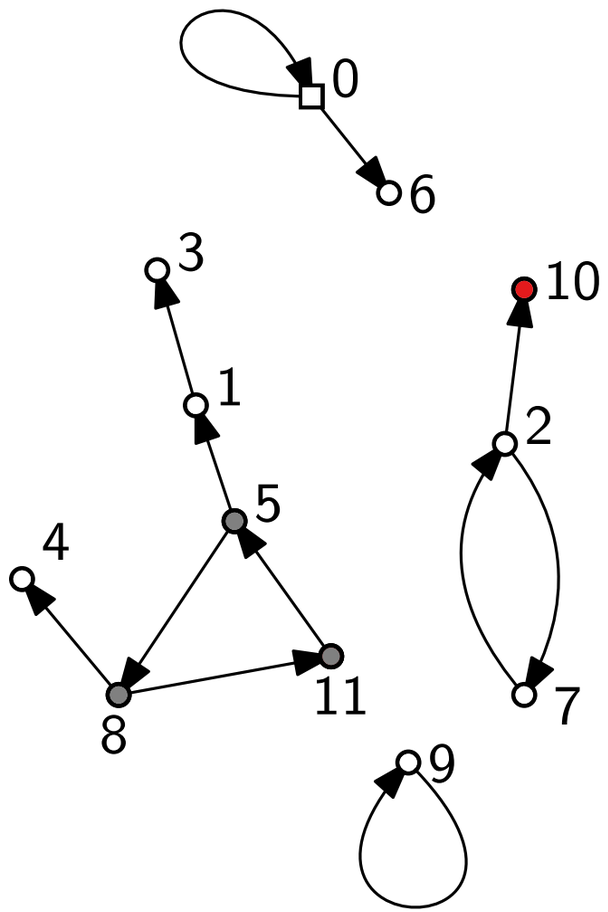
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**

Berechnung Hilfsvariablen



Knoten i	e_i	g_i
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	11
6	0	0
7	0	0
8	0	11
9	0	0
10	0	10
11	0	11

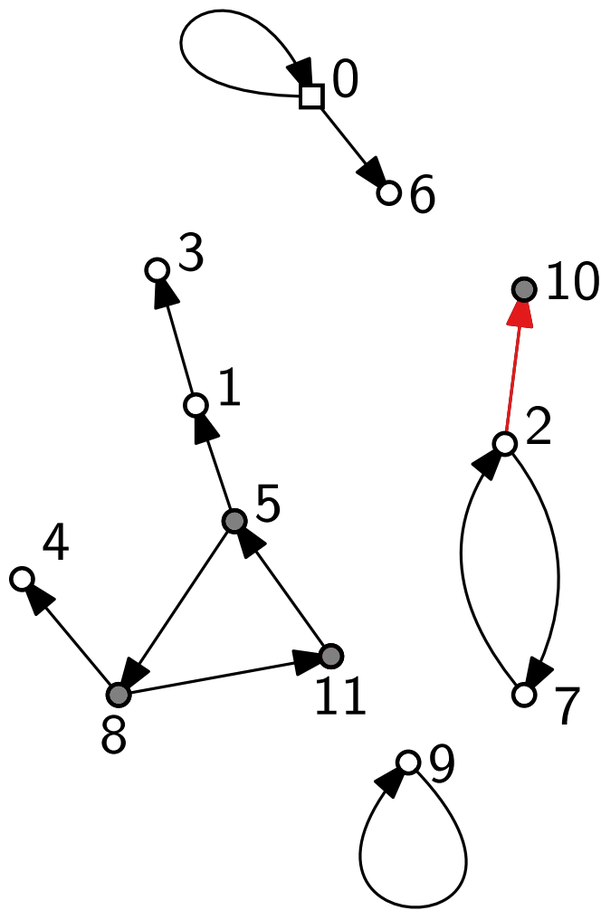
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**

Berechnung Hilfsvariablen



Knoten i	e_i	g_i
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	11
6	0	0
7	0	0
8	0	11
9	0	0
10	0	10
11	0	11

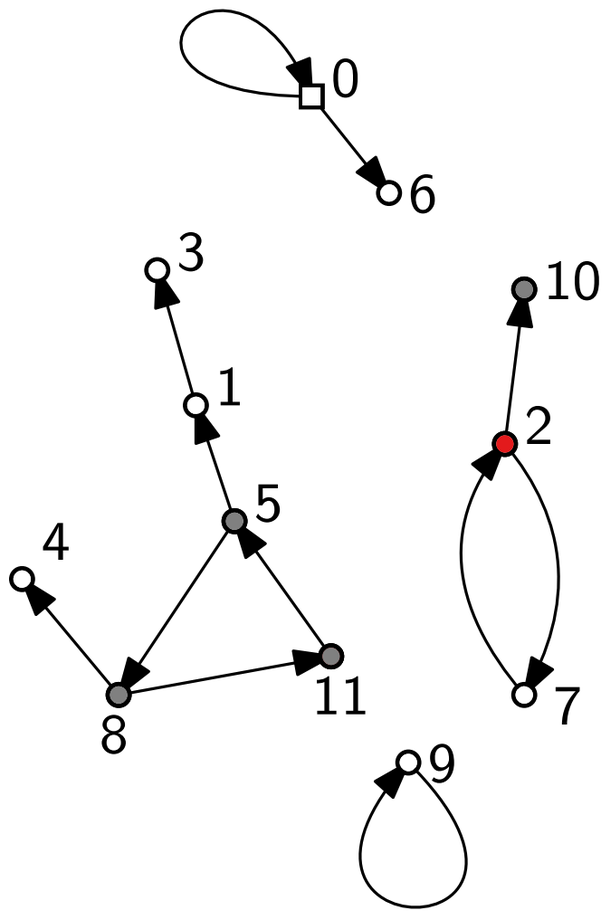
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**

Berechnung Hilfsvariablen



Knoten i	e_i	g_i
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	11
6	0	0
7	0	0
8	0	11
9	0	0
10	0	10
11	0	11

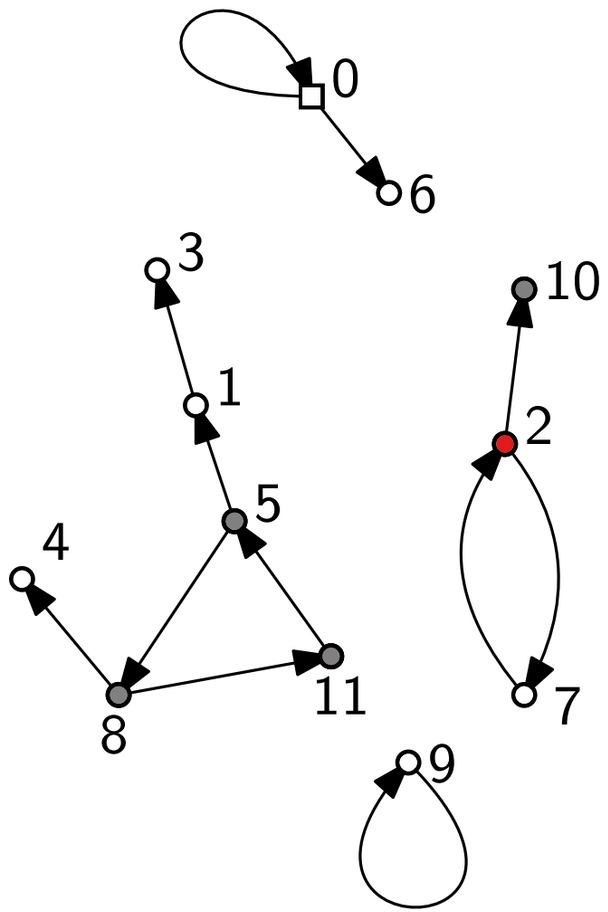
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**

Berechnung Hilfsvariablen



Knoten i	e_i	g_i
0	0	0
1	0	0
2	0	10
3	0	0
4	0	0
5	0	11
6	0	0
7	0	0
8	0	11
9	0	0
10	0	10
11	0	11

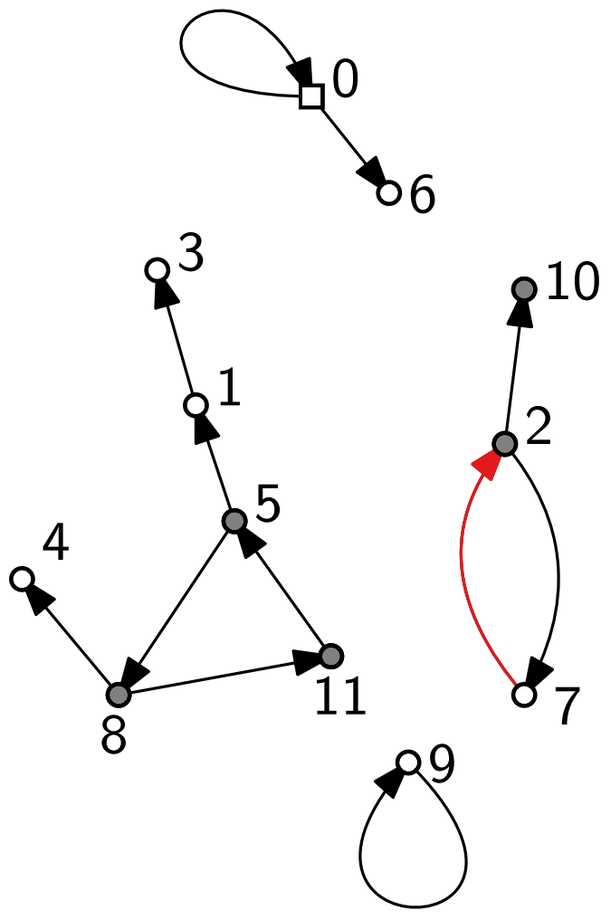
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**

Berechnung Hilfsvariablen



Knoten i	e_i	g_i
0	0	0
1	0	0
2	0	10
3	0	0
4	0	0
5	0	11
6	0	0
7	0	0
8	0	11
9	0	0
10	0	10
11	0	11

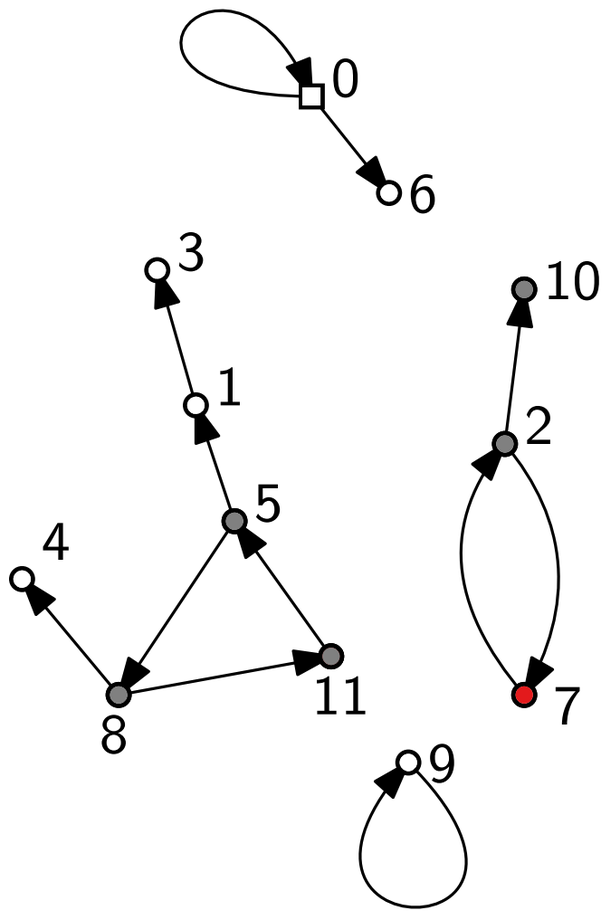
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**

Berechnung Hilfsvariablen



Knoten i	e_i	g_i
0	0	0
1	0	0
2	0	10
3	0	0
4	0	0
5	0	11
6	0	0
7	0	0
8	0	11
9	0	0
10	0	10
11	0	11

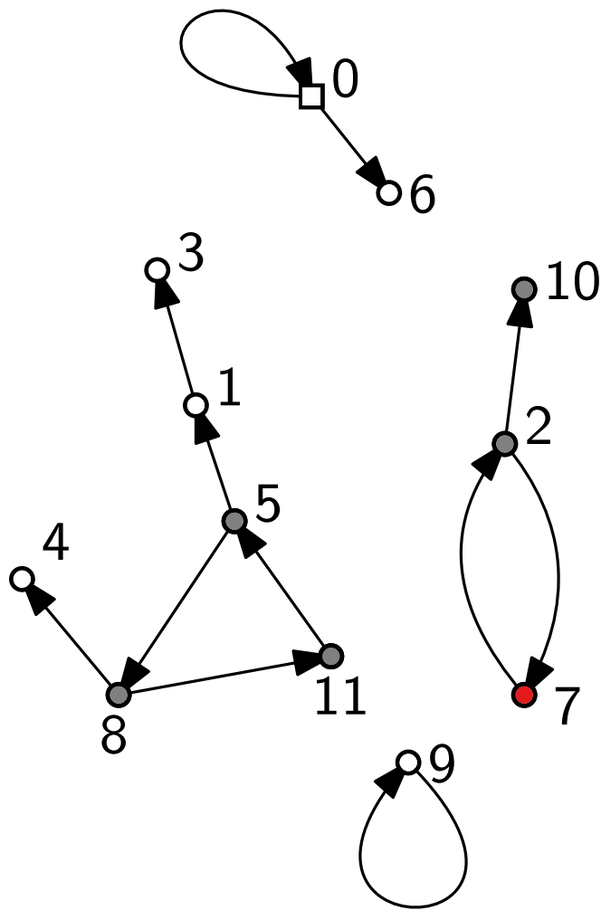
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**

Berechnung Hilfsvariablen



Knoten i	e_i	g_i
0	0	0
1	0	0
2	0	10
3	0	0
4	0	0
5	0	11
6	0	0
7	0	10
8	0	11
9	0	0
10	0	10
11	0	11

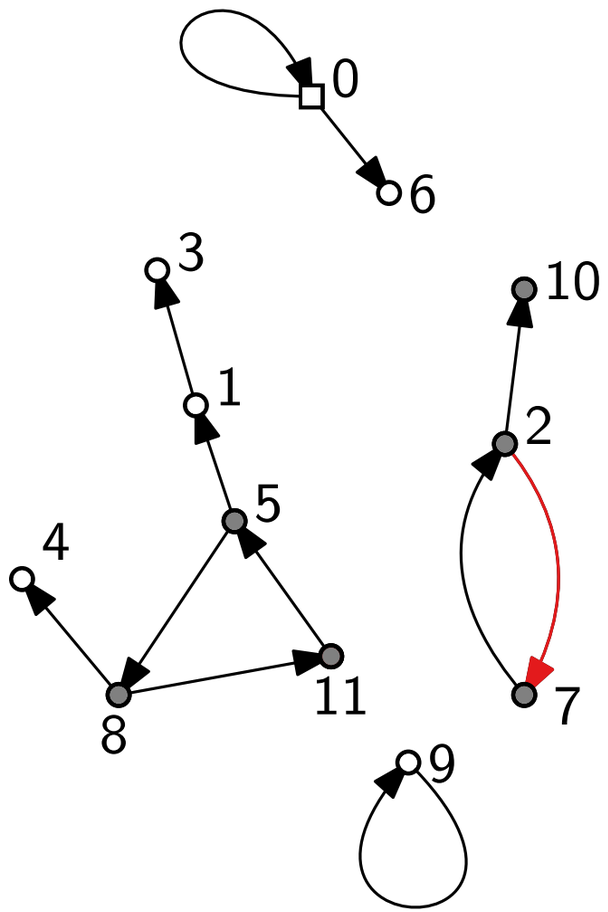
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**

Berechnung Hilfsvariablen



Knoten i	e_i	g_i
0	0	0
1	0	0
2	0	10
3	0	0
4	0	0
5	0	11
6	0	0
7	0	10
8	0	11
9	0	0
10	0	10
11	0	11

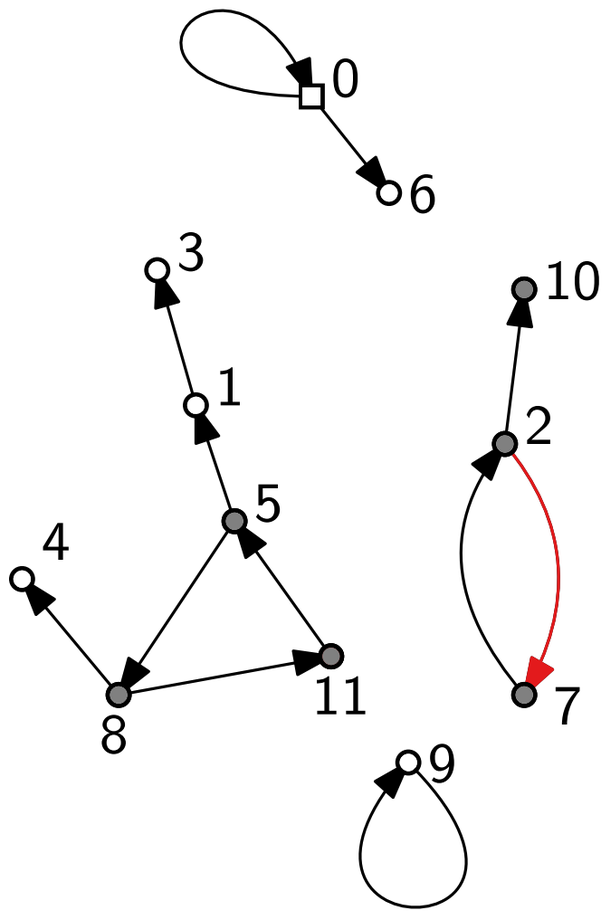
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**

Berechnung Hilfsvariablen



Knoten i	e_i	g_i
0	0	0
1	0	0
2	0	10
3	0	0
4	0	0
5	0	11
6	0	0
7	0	10
8	0	11
9	0	0
10	0	10
11	0	11

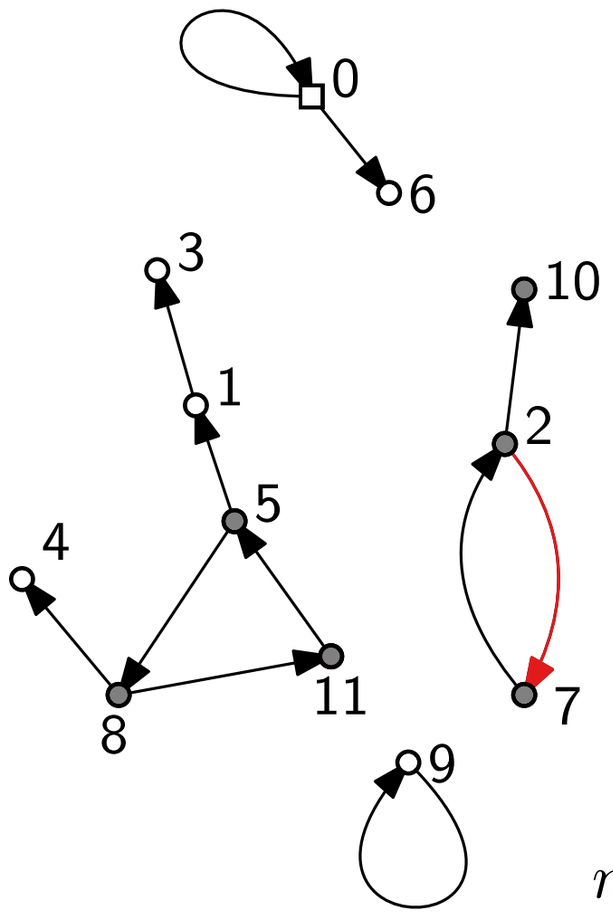
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**
    - **do**
      - $m \leftarrow \min\{m, a_{j,n}\}$
    - **while** $(j \leftarrow x_j) \neq i'$

Berechnung Hilfsvariablen



m:

$a_{\_,n}$:

5 3 20 8 15 4 16 13 7 2 17

Knoten i	e_i	g_i
0	0	0
1	0	0
2	0	10
3	0	0
4	0	0
5	0	11
6	0	0
7	0	10
8	0	11
9	0	0
10	0	10
11	0	11

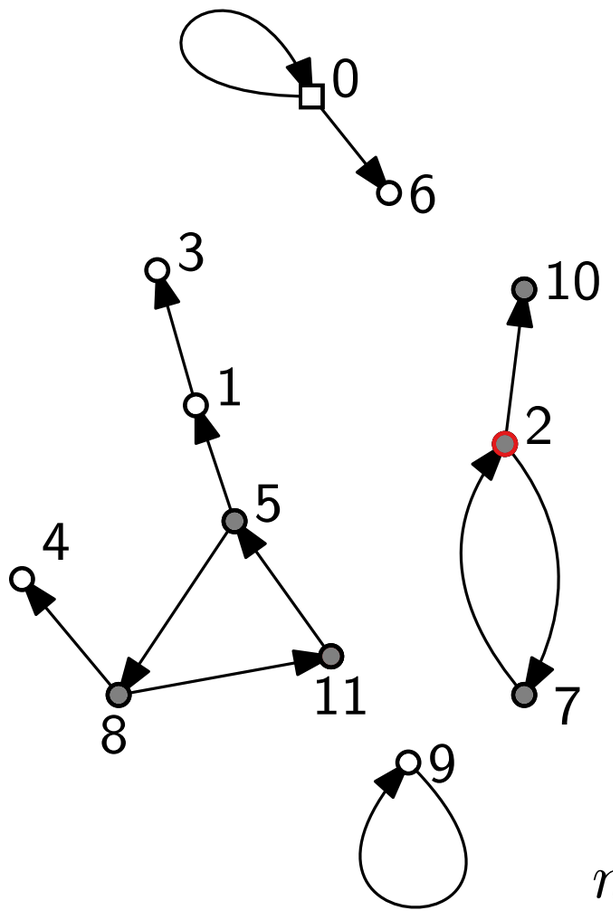
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**
    - **do**
      - $m \leftarrow \min\{m, a_{j,n}\}$
    - **while** $(j \leftarrow x_j) \neq i'$

Berechnung Hilfsvariablen



m:

$a_{\_,n}$:

5	3	20	8	15	4	16	13	7	2	17

Knoten i	e_i	g_i
0	0	0
1	0	0
2	0	10
3	0	0
4	0	0
5	0	11
6	0	0
7	0	10
8	0	11
9	0	0
10	0	10
11	0	11

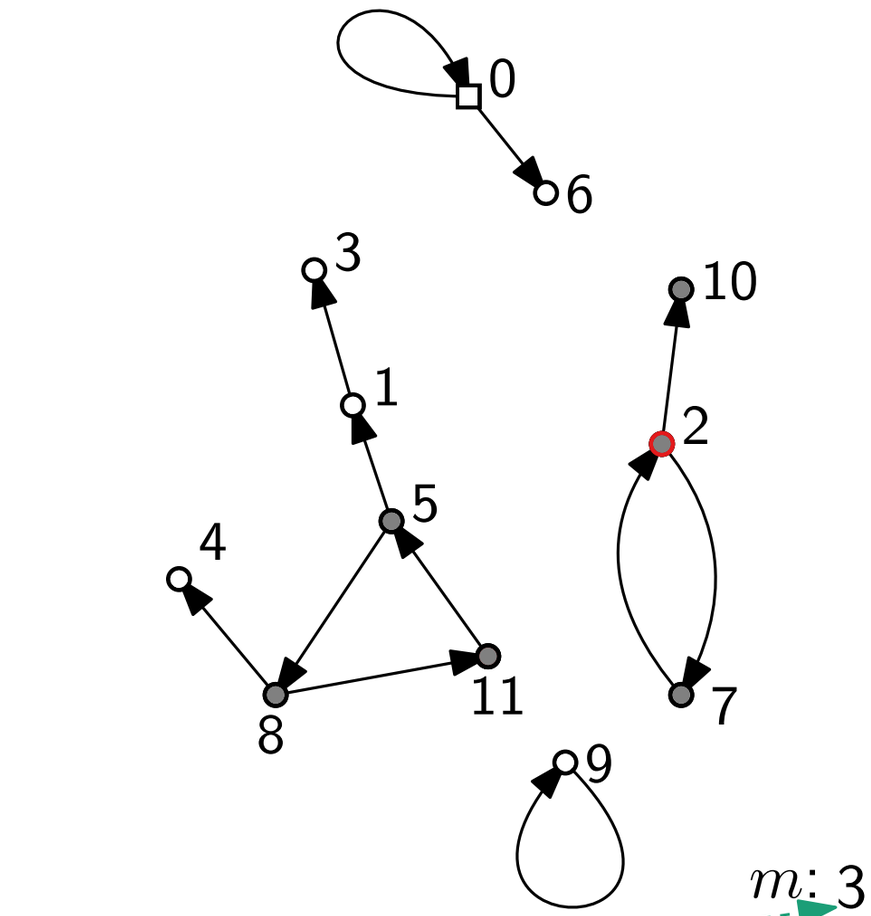
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**
    - **do**
      - $m \leftarrow \min\{m, a_{j,n}\}$
    - **while** $(j \leftarrow x_j) \neq i'$

Berechnung Hilfsvariablen



Knoten i	e_i	g_i
0	0	0
1	0	0
2	0	10
3	0	0
4	0	0
5	0	11
6	0	0
7	0	10
8	0	11
9	0	0
10	0	10
11	0	11

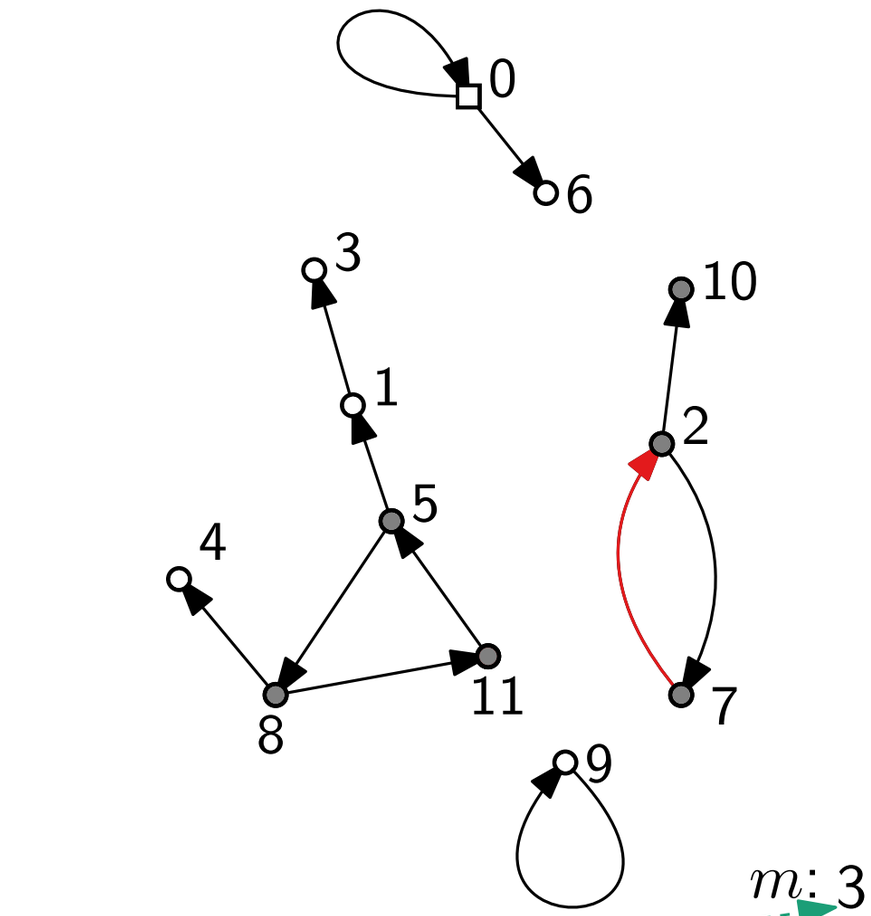
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**
    - **do**
      - $m \leftarrow \min\{m, a_{j,n}\}$
    - **while** $(j \leftarrow x_j) \neq i'$

Berechnung Hilfsvariablen



Knoten i	e_i	g_i
0	0	0
1	0	0
2	0	10
3	0	0
4	0	0
5	0	11
6	0	0
7	0	10
8	0	11
9	0	0
10	0	10
11	0	11

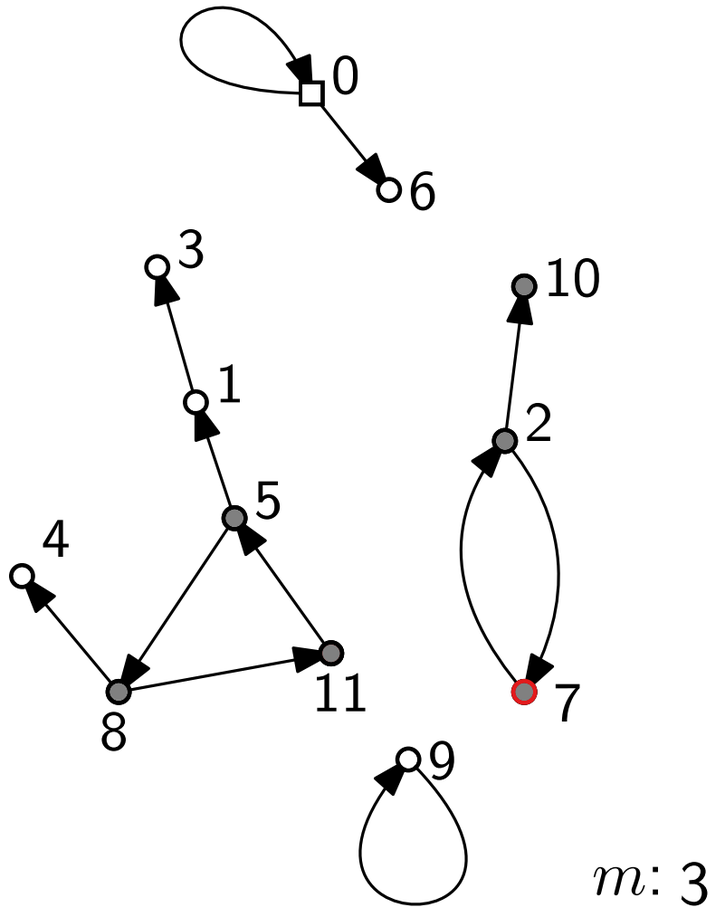
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**
    - **do**
      - $m \leftarrow \min\{m, a_{j,n}\}$
    - **while** $(j \leftarrow x_j) \neq i'$

Berechnung Hilfsvariablen



$a_{\_,n}$:

5 3 20 8 15 4 16 13 7 2 17

Knoten i	e_i	g_i
0	0	0
1	0	0
2	0	10
3	0	0
4	0	0
5	0	11
6	0	0
7	0	10
8	0	11
9	0	0
10	0	10
11	0	11

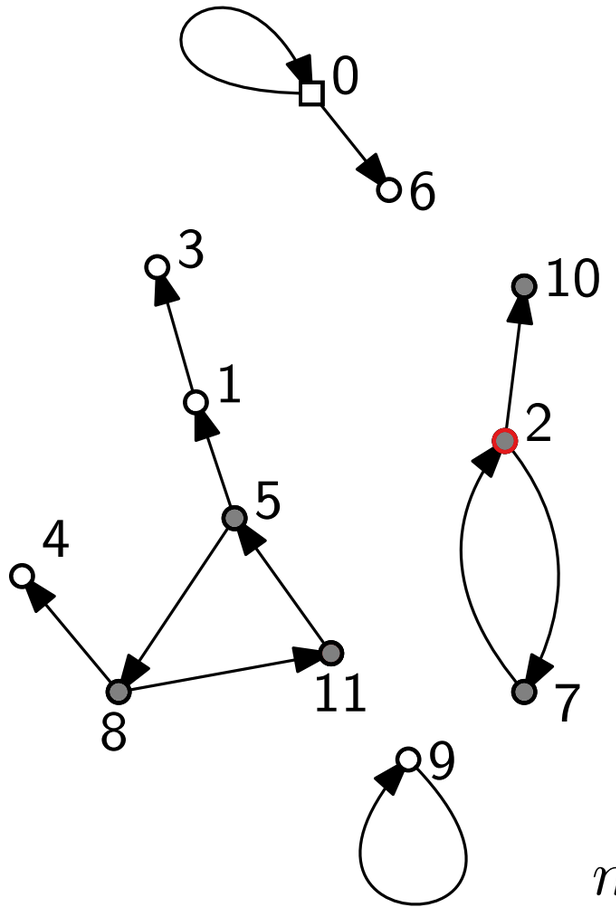
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**
    - **do**
      - $m \leftarrow \min\{m, a_{j,n}\}$
    - **while** $(j \leftarrow x_j) \neq i'$

Berechnung Hilfsvariablen



m: 3

$a_{\_,n}$:

5 3 20 8 15 4 16 13 7 2 17

Knoten i	e_i	g_i
0	0	0
1	0	0
2	0	10
3	0	0
4	0	0
5	0	11
6	0	0
7	0	10
8	0	11
9	0	0
10	0	10
11	0	11

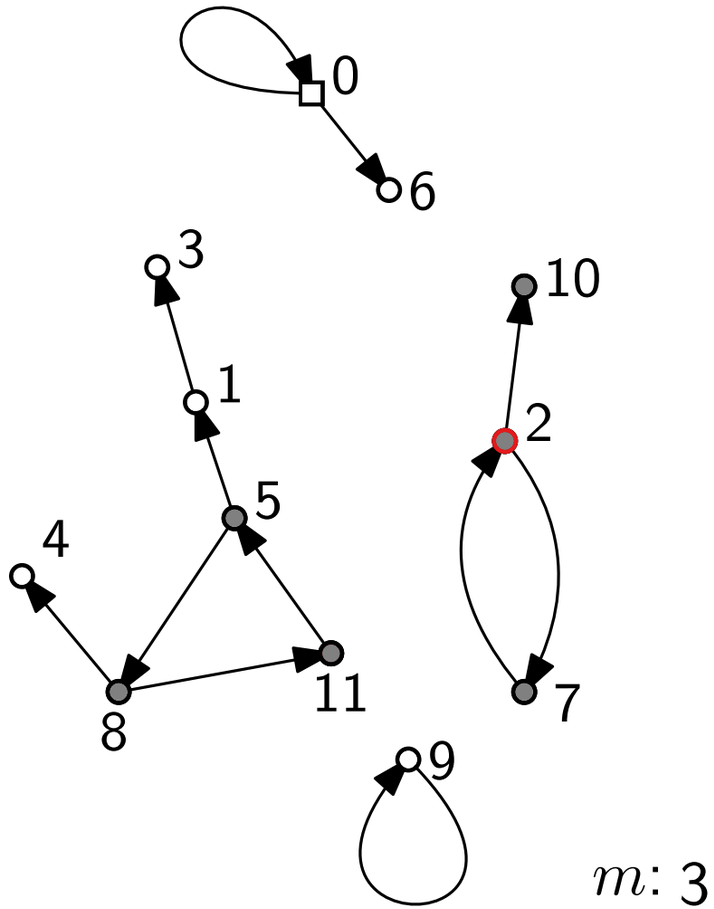
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**
    - **do**
      - $m \leftarrow \min\{m, a_{j,n}\}$
    - **while** $(j \leftarrow x_j) \neq i'$

Berechnung Hilfsvariablen



m: 3

$a_{\_,n}$:

5	3	20	8	15	4	16	13	7	2	17

Knoten i	e_i	g_i
0	0	0
1	0	0
2	0	10
3	0	0
4	0	0
5	0	11
6	0	0
7	0	10
8	0	11
9	0	0
10	0	10
11	0	11

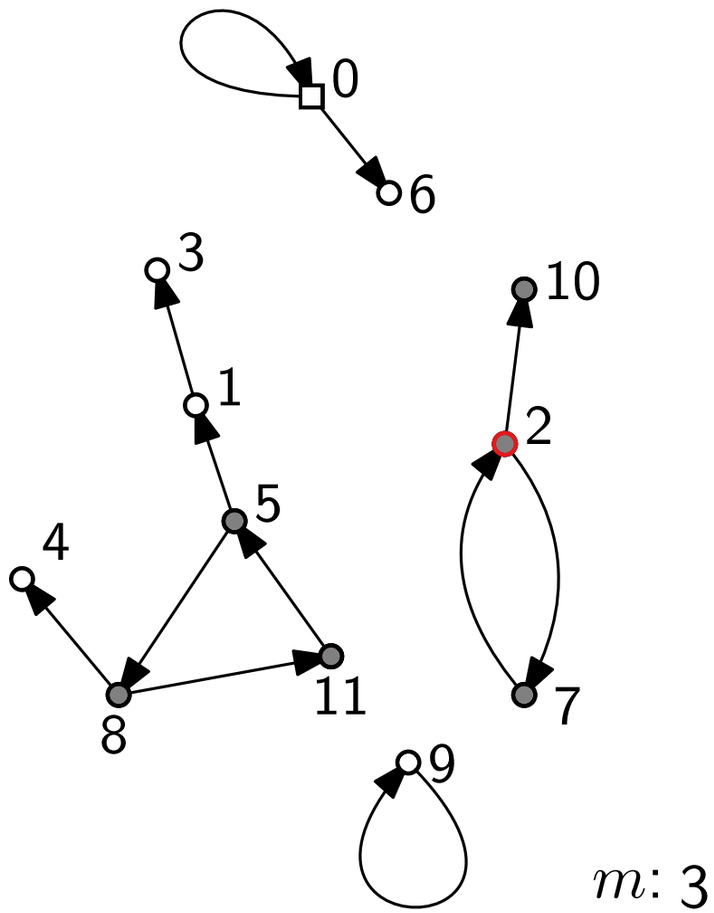
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**
    - **do**
      - $m \leftarrow \min\{m, a_{j,n}\}$
    - **while** $(j \leftarrow x_j) \neq i'$
    - **do**
      - $e_j \leftarrow m$
    - **while** $(j \leftarrow x_j) \neq i'$

Berechnung Hilfsvariablen



m: 3

$a_{\_,n}$:

5	3	20	8	15	4	16	13	7	2	17

Knoten i	e_i	g_i
0	0	0
1	0	0
2	3	10
3	0	0
4	0	0
5	0	11
6	0	0
7	0	10
8	0	11
9	0	0
10	0	10
11	0	11

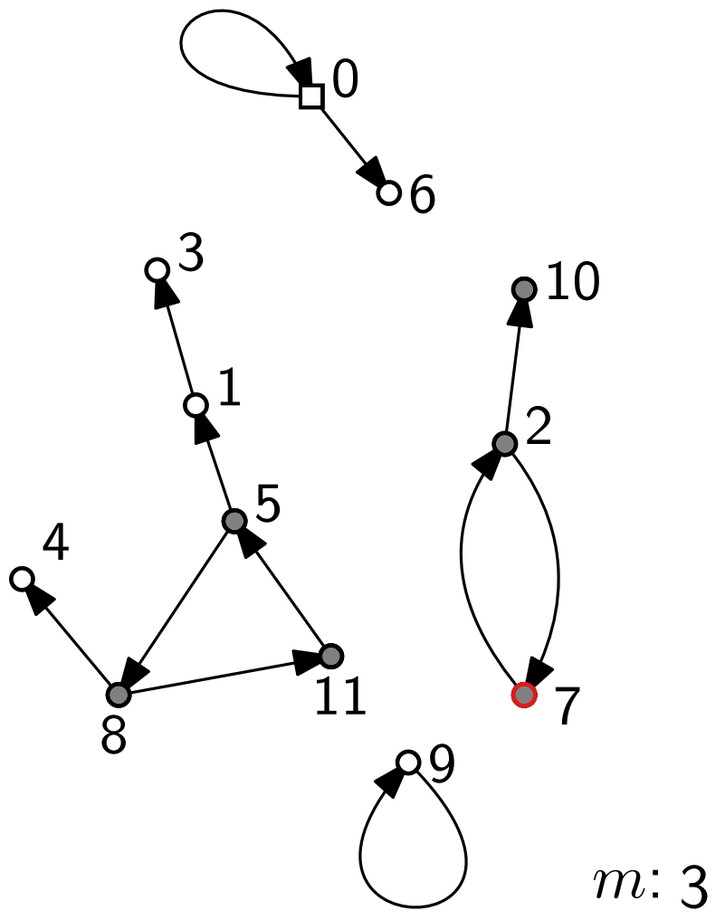
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**
    - **do**
      - $m \leftarrow \min\{m, a_{j,n}\}$
    - **while** $(j \leftarrow x_j) \neq i'$
    - **do**
      - $e_j \leftarrow m$
    - **while** $(j \leftarrow x_j) \neq i'$

Berechnung Hilfsvariablen



m: 3

$a_{\_,n}$:

5	3	20	8	15	4	16	13	7	2	17

Knoten i	e_i	g_i
0	0	0
1	0	0
2	3	10
3	0	0
4	0	0
5	0	11
6	0	0
7	3	10
8	0	11
9	0	0
10	0	10
11	0	11

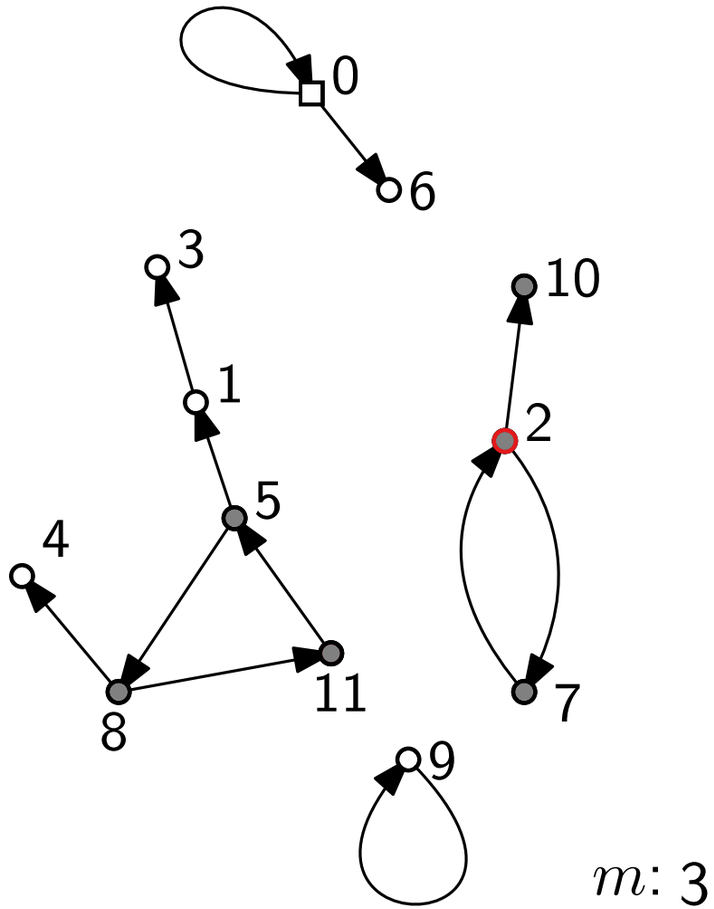
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**
    - **do**
      - $m \leftarrow \min\{m, a_{j,n}\}$
    - **while** $(j \leftarrow x_j) \neq i'$
    - **do**
      - $e_j \leftarrow m$
    - **while** $(j \leftarrow x_j) \neq i'$

Berechnung Hilfsvariablen



m: 3

$a_{\_,n}$:

5	3	20	8	15	4	16	13	7	2	17

Knoten i	e_i	g_i
0	0	0
1	0	0
2	3	10
3	0	0
4	0	0
5	0	11
6	0	0
7	3	10
8	0	11
9	0	0
10	0	10
11	0	11

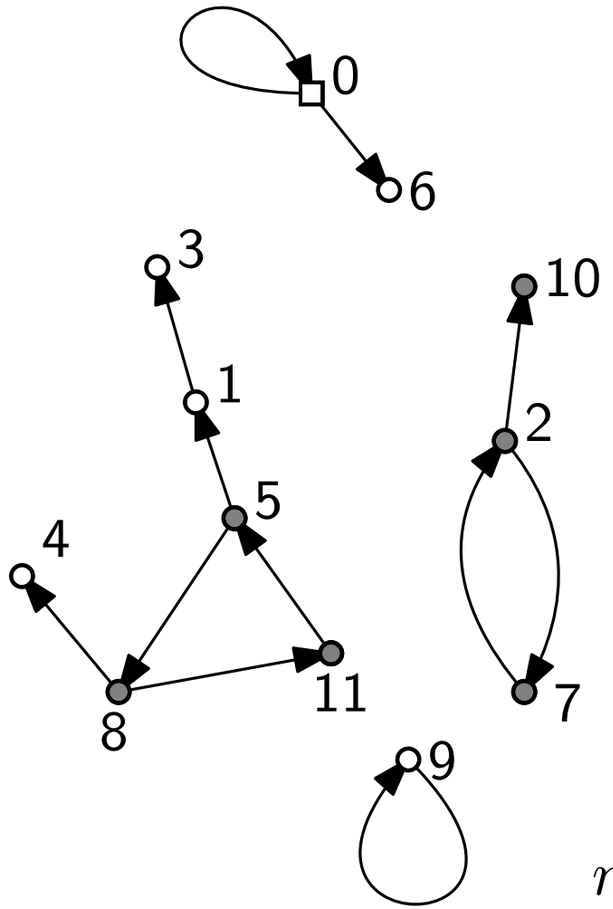
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**
    - **do**
      - $m \leftarrow \min\{m, a_{j,n}\}$
    - **while** $(j \leftarrow x_j) \neq i'$
    - **do**
      - $e_j \leftarrow m$
    - **while** $(j \leftarrow x_j) \neq i'$

Berechnung Hilfsvariablen



m:

$a_{\_,n}$:

5 3 20 8 15 4 16 13 7 2 17

Knoten i	e_i	g_i
0	0	0
1	0	0
2	3	10
3	0	0
4	0	0
5	0	11
6	0	0
7	3	10
8	0	11
9	0	0
10	0	10
11	0	11

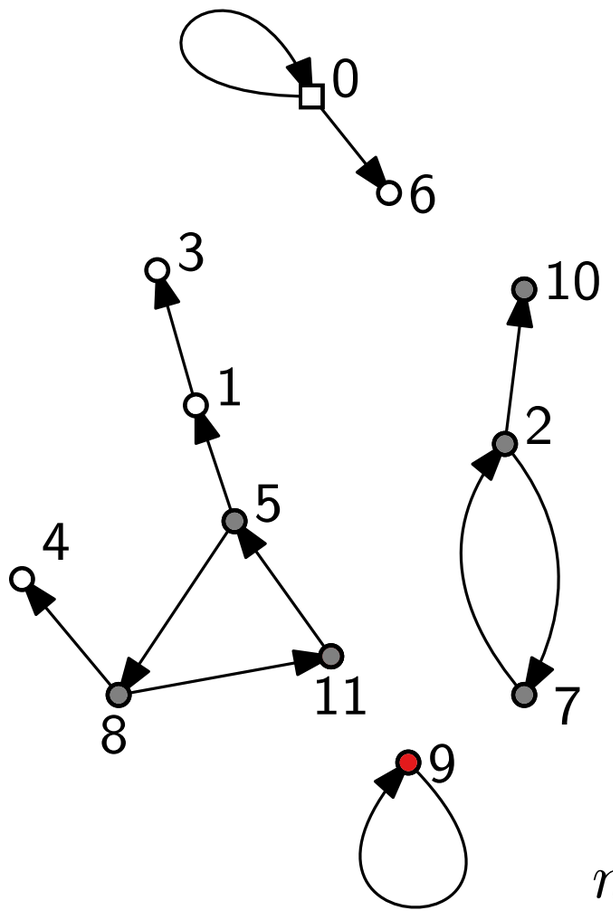
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**
    - **do**
      - $m \leftarrow \min\{m, a_{j,n}\}$
    - **while** $(j \leftarrow x_j) \neq i'$
    - **do**
      - $e_j \leftarrow m$
    - **while** $(j \leftarrow x_j) \neq i'$

Berechnung Hilfsvariablen



m:

$a_{\_,n}$:

5	3	20	8	15	4	16	13	7	2	17

Knoten i	e_i	g_i
0	0	0
1	0	0
2	3	10
3	0	0
4	0	0
5	0	11
6	0	0
7	3	10
8	0	11
9	0	0
10	0	10
11	0	11

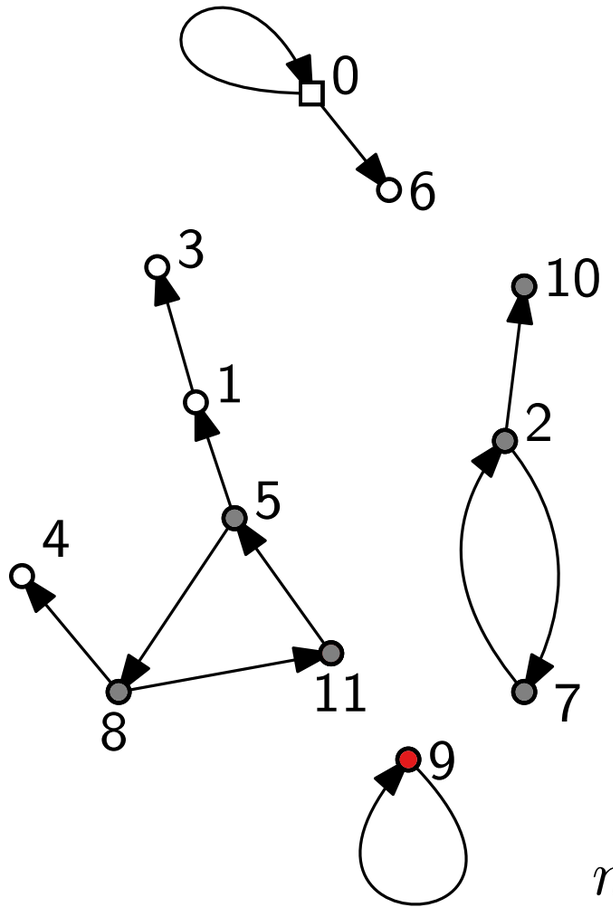
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**
    - **do**
      - $m \leftarrow \min\{m, a_{j,n}\}$
    - **while** $(j \leftarrow x_j) \neq i'$
    - **do**
      - $e_j \leftarrow m$
    - **while** $(j \leftarrow x_j) \neq i'$

Berechnung Hilfsvariablen



m:

$a_{\_,n}$:

5 3 20 8 15 4 16 13 7 2 17

Knoten i	e_i	g_i
0	0	0
1	0	0
2	3	10
3	0	0
4	0	0
5	0	11
6	0	0
7	3	10
8	0	11
9	0	9
10	0	10
11	0	11

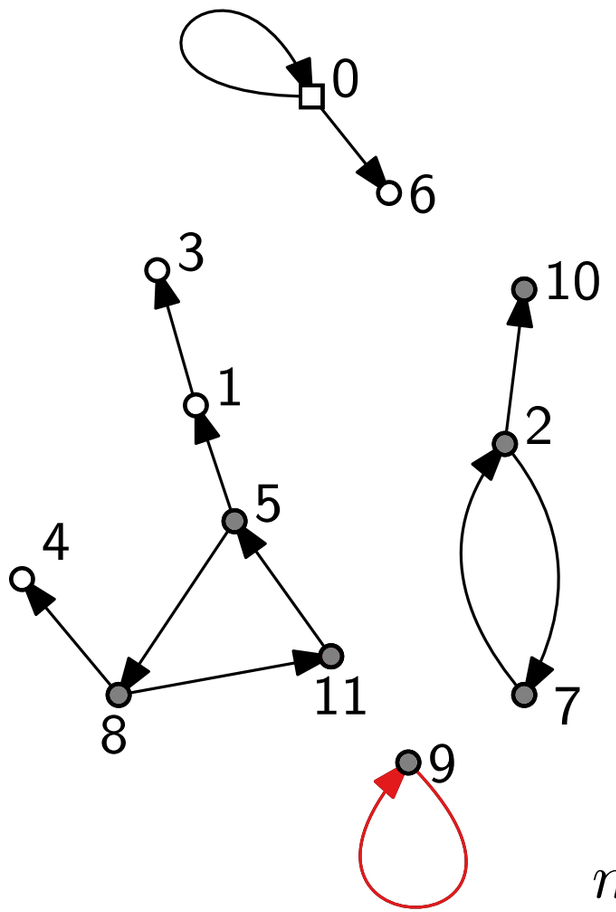
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**
    - **do**
      - $m \leftarrow \min\{m, a_{j,n}\}$
    - **while** $(j \leftarrow x_j) \neq i'$
    - **do**
      - $e_j \leftarrow m$
    - **while** $(j \leftarrow x_j) \neq i'$

Berechnung Hilfsvariablen



m:

$a_{\_,n}$:

5 3 20 8 15 4 16 13 7 2 17

Knoten i	e_i	g_i
0	0	0
1	0	0
2	3	10
3	0	0
4	0	0
5	0	11
6	0	0
7	3	10
8	0	11
9	0	9
10	0	10
11	0	11

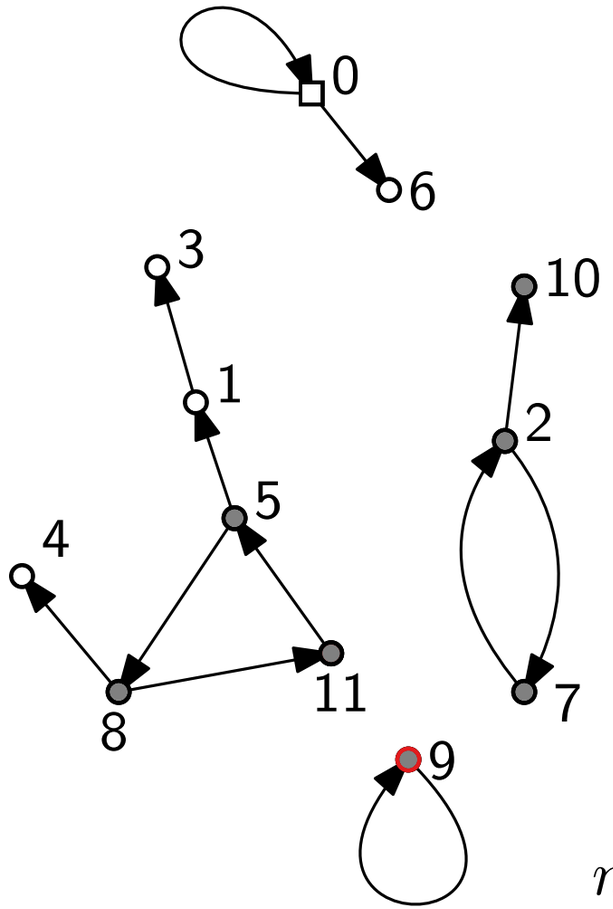
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**
    - **do**
      - $m \leftarrow \min\{m, a_{j,n}\}$
    - **while** $(j \leftarrow x_j) \neq i'$
    - **do**
      - $e_j \leftarrow m$
    - **while** $(j \leftarrow x_j) \neq i'$

Berechnung Hilfsvariablen



m:

$a_{\_,n}$:

5 3 20 8 15 4 16 13 7 2 17

Knoten i	e_i	g_i
0	0	0
1	0	0
2	3	10
3	0	0
4	0	0
5	0	11
6	0	0
7	3	10
8	0	11
9	0	9
10	0	10
11	0	11

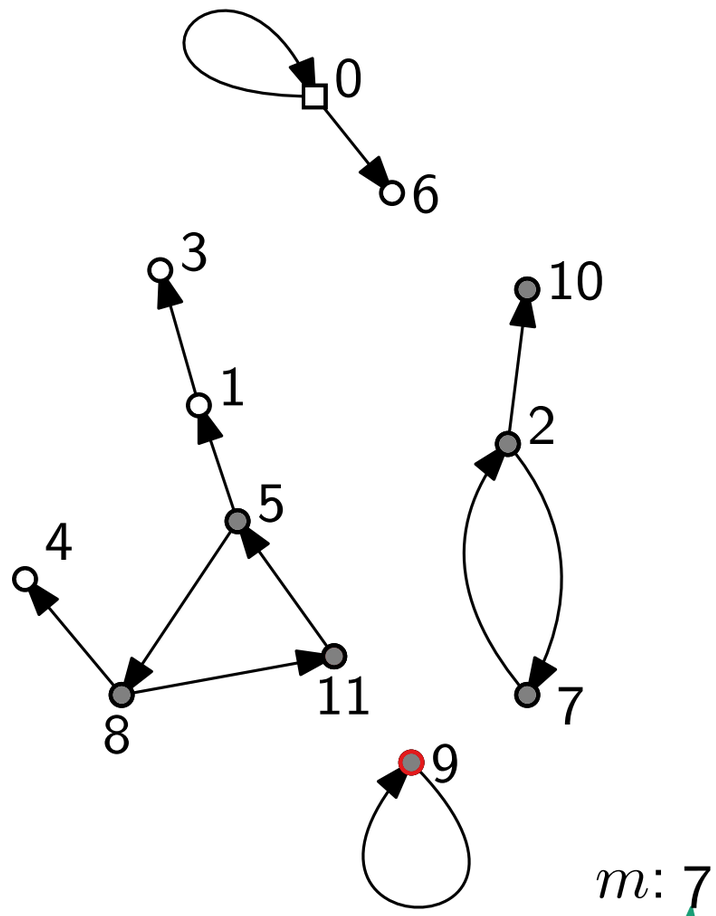
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

```
for i = n downto 1 do
    j ← i
    while g_j = 0 do
        g_j ← i
        j ← x_j
    if g_j = i then
        i', m ← (j, a_{j,n})
        if i' ≠ n then
            do
                m ← min{m, a_{j,n}}
            while (j ← x_j) ≠ i'
            do
                e_j ← m
            while (j ← x_j) ≠ i'
```

Berechnung Hilfsvariablen



Knoten i	e_i	g_i
0	0	0
1	0	0
2	3	10
3	0	0
4	0	0
5	0	11
6	0	0
7	3	10
8	0	11
9	0	9
10	0	10
11	0	11

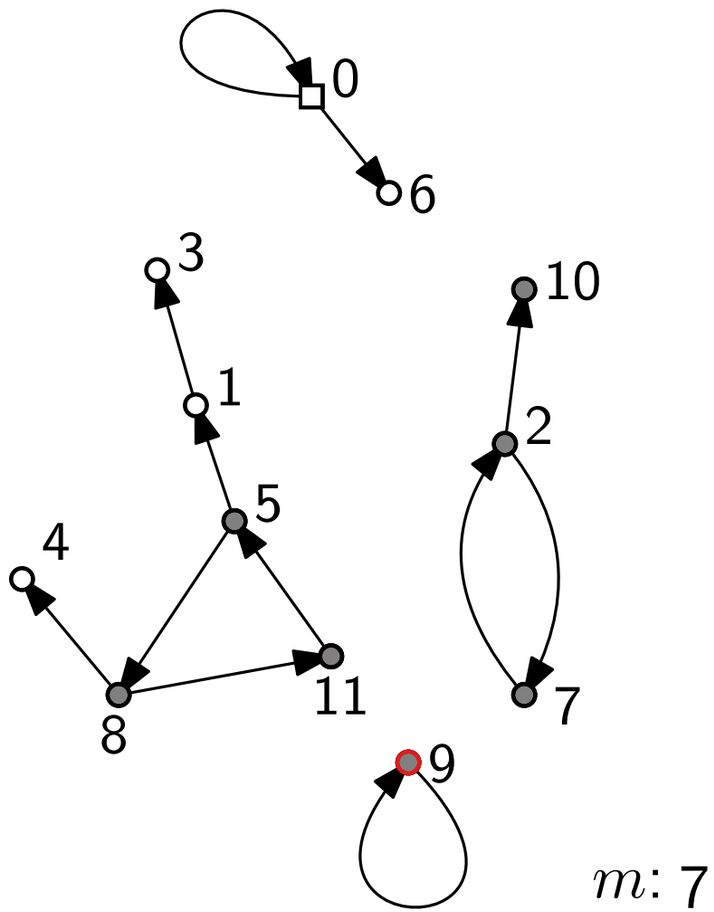
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**
    - **do**
      - $m \leftarrow \min\{m, a_{j,n}\}$
    - **while** $(j \leftarrow x_j) \neq i'$
    - **do**
      - $e_j \leftarrow m$
    - **while** $(j \leftarrow x_j) \neq i'$

Berechnung Hilfsvariablen



m: 7

$a_{\_,n}$:

5	3	20	8	15	4	16	13	7	2	17

Knoten i	e_i	g_i
0	0	0
1	0	0
2	3	10
3	0	0
4	0	0
5	0	11
6	0	0
7	3	10
8	0	11
9	7	9
10	0	10
11	0	11

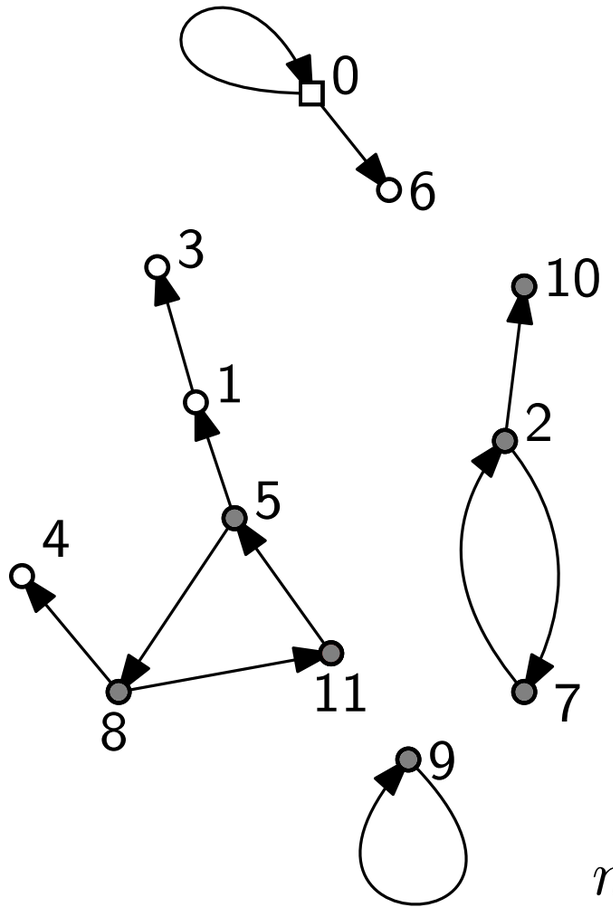
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**
    - **do**
      - $m \leftarrow \min\{m, a_{j,n}\}$
    - **while** $(j \leftarrow x_j) \neq i'$
    - **do**
      - $e_j \leftarrow m$
    - **while** $(j \leftarrow x_j) \neq i'$

Berechnung Hilfsvariablen



m:

$a_{\_,n}$:

5 3 20 8 15 4 16 13 7 2 17

Knoten i	e_i	g_i
0	0	0
1	0	0
2	3	10
3	0	0
4	0	0
5	0	11
6	0	0
7	3	10
8	0	11
9	7	9
10	0	10
11	0	11

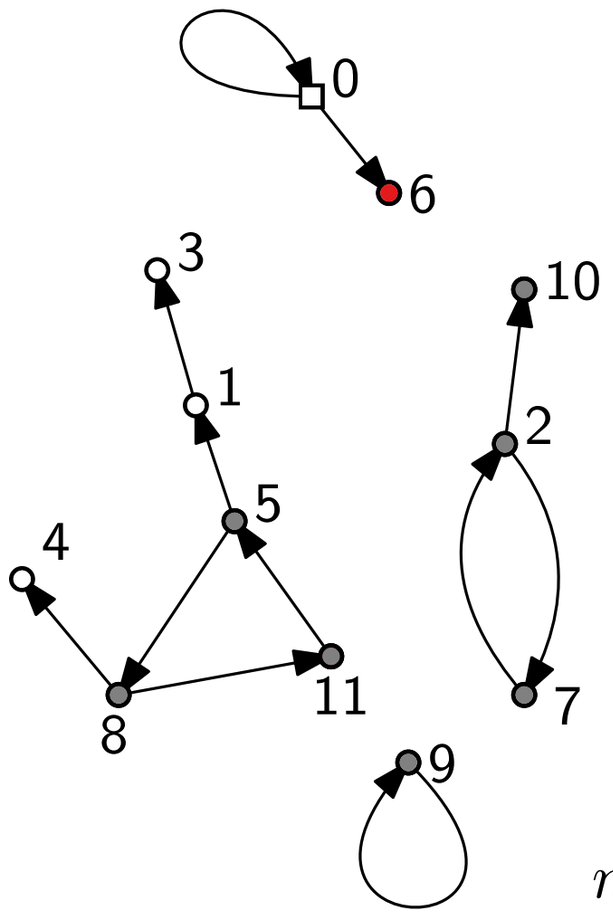
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**
    - **do**
      - $m \leftarrow \min\{m, a_{j,n}\}$
    - **while** $(j \leftarrow x_j) \neq i'$
    - **do**
      - $e_j \leftarrow m$
    - **while** $(j \leftarrow x_j) \neq i'$

Berechnung Hilfsvariablen



m:

$a_{\_,n}$:

5 3 20 8 15 4 16 13 7 2 17

Knoten i	e_i	g_i
0	0	0
1	0	0
2	3	10
3	0	0
4	0	0
5	0	11
6	0	0
7	3	10
8	0	11
9	7	9
10	0	10
11	0	11

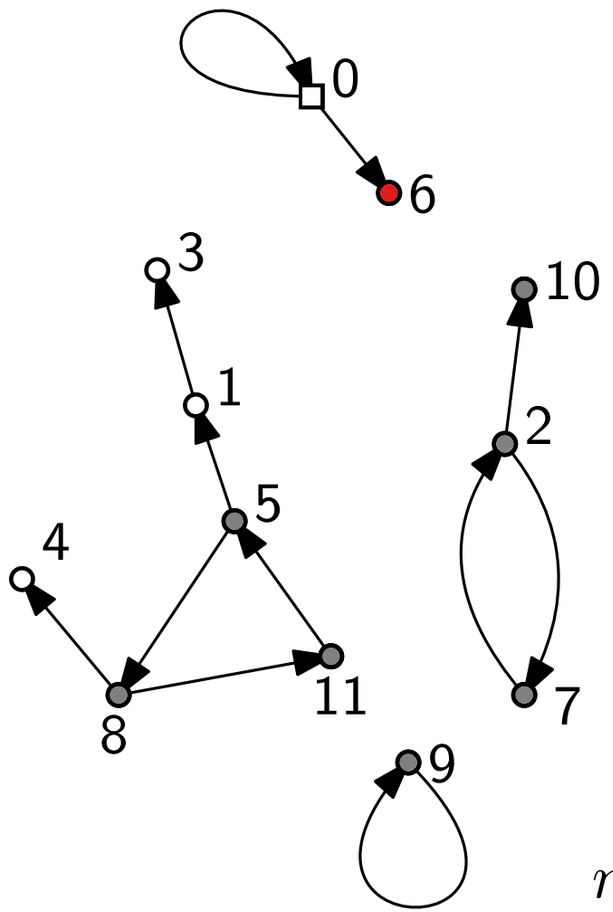
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**
    - **do**
      - $m \leftarrow \min\{m, a_{j,n}\}$
    - **while** $(j \leftarrow x_j) \neq i'$
    - **do**
      - $e_j \leftarrow m$
    - **while** $(j \leftarrow x_j) \neq i'$

Berechnung Hilfsvariablen



m:

$a_{\_,n}$:

5 3 20 8 15 4 16 13 7 2 17

Knoten i	e_i	g_i
0	0	0
1	0	0
2	3	10
3	0	0
4	0	0
5	0	11
6	0	6
7	3	10
8	0	11
9	7	9
10	0	10
11	0	11

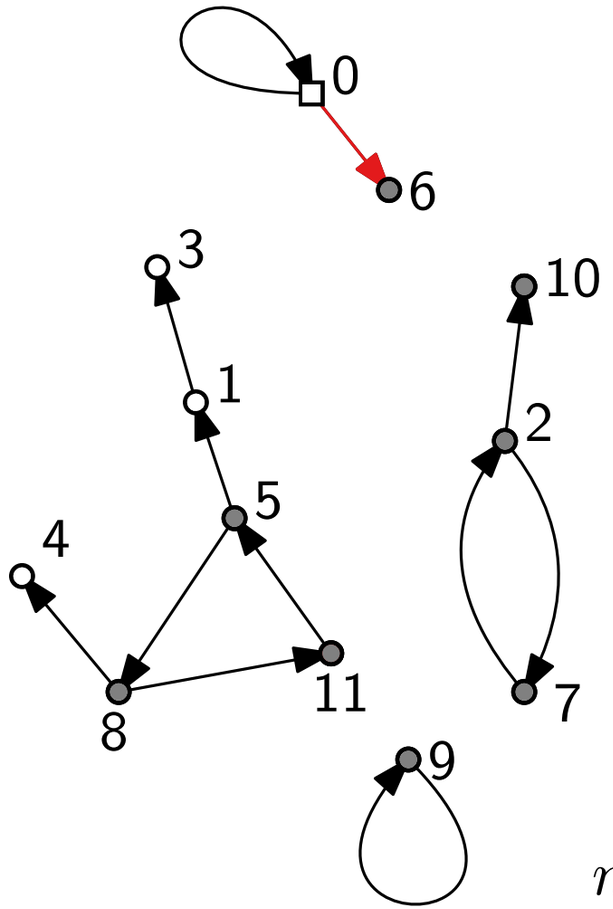
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**
    - **do**
      - $m \leftarrow \min\{m, a_{j,n}\}$
    - **while** $(j \leftarrow x_j) \neq i'$
    - **do**
      - $e_j \leftarrow m$
    - **while** $(j \leftarrow x_j) \neq i'$

Berechnung Hilfsvariablen



m:

$a_{\_,n}$:

5 3 20 8 15 4 16 13 7 2 17

Knoten i	e_i	g_i
0	0	0
1	0	0
2	3	10
3	0	0
4	0	0
5	0	11
6	0	6
7	3	10
8	0	11
9	7	9
10	0	10
11	0	11

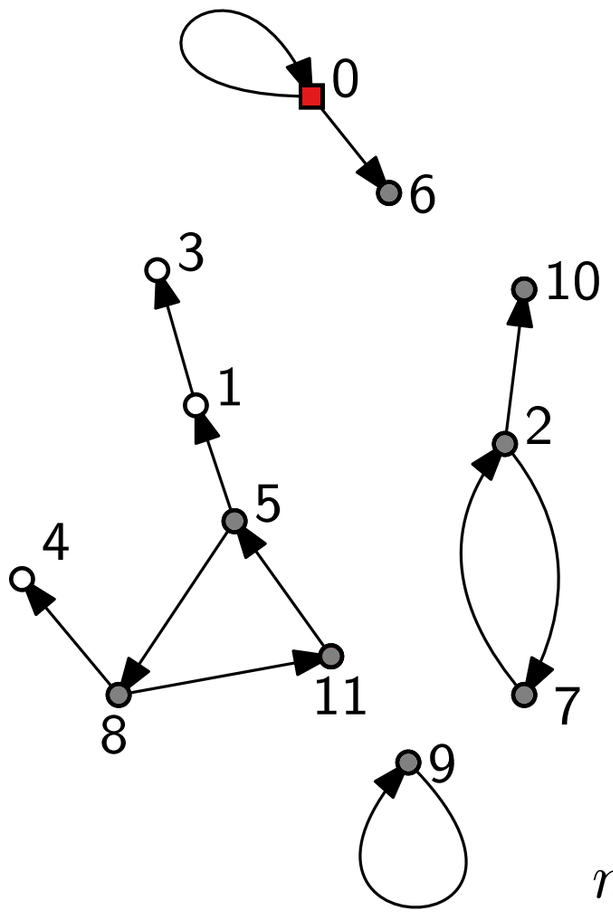
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**
    - **do**
      - $m \leftarrow \min\{m, a_{j,n}\}$
    - **while** $(j \leftarrow x_j) \neq i'$
    - **do**
      - $e_j \leftarrow m$
    - **while** $(j \leftarrow x_j) \neq i'$

Berechnung Hilfsvariablen



m:

$a_{\_,n}$:

5 3 20 8 15 4 16 13 7 2 17

Knoten i	e_i	g_i
0	0	0
1	0	0
2	3	10
3	0	0
4	0	0
5	0	11
6	0	6
7	3	10
8	0	11
9	7	9
10	0	10
11	0	11

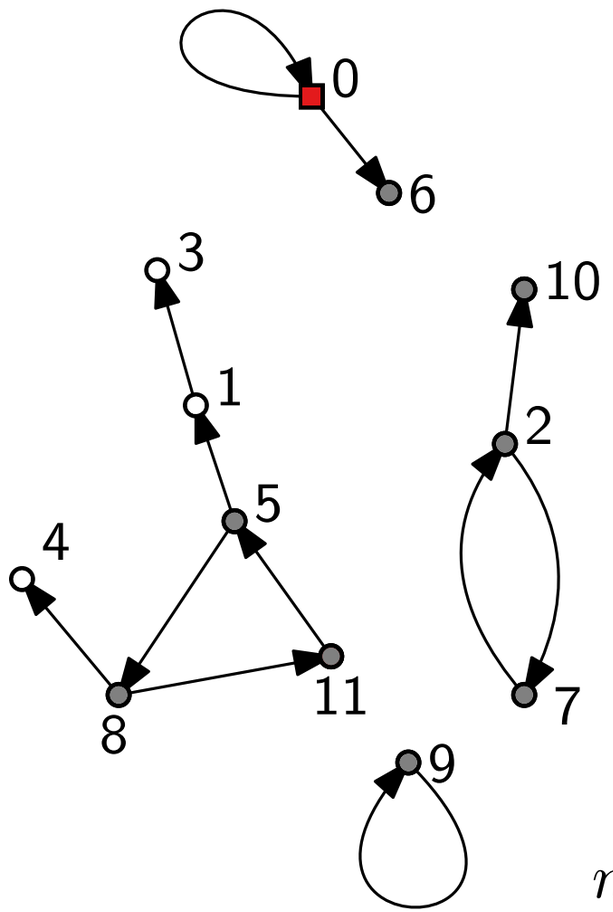
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**
    - **do**
      - $m \leftarrow \min\{m, a_{j,n}\}$
    - **while** $(j \leftarrow x_j) \neq i'$
    - **do**
      - $e_j \leftarrow m$
    - **while** $(j \leftarrow x_j) \neq i'$

Berechnung Hilfsvariablen



m:

$a_{\_,n}$:

5 3 20 8 15 4 16 13 7 2 17

Knoten i	e_i	g_i
0	0	6
1	0	0
2	3	10
3	0	0
4	0	0
5	0	11
6	0	6
7	3	10
8	0	11
9	7	9
10	0	10
11	0	11

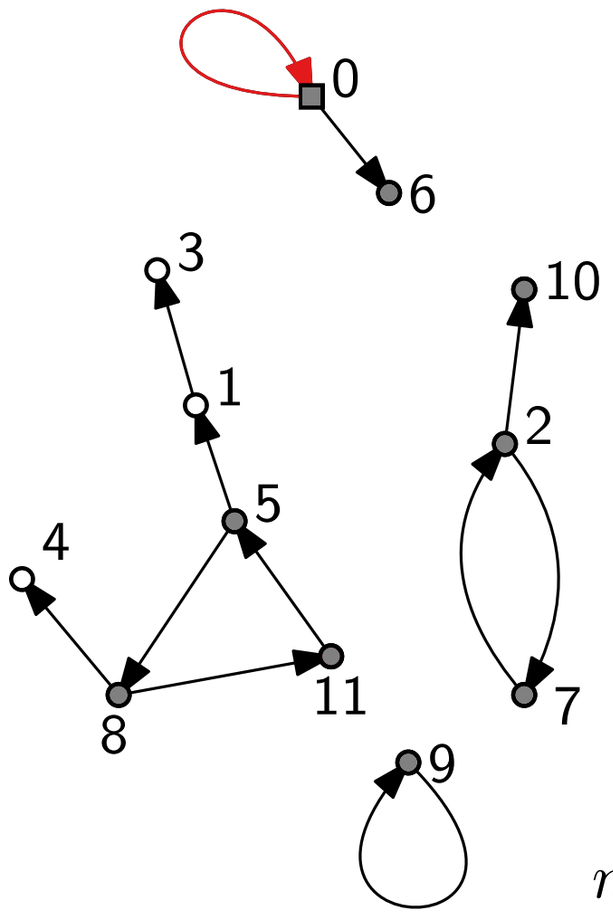
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**
    - **do**
      - $m \leftarrow \min\{m, a_{j,n}\}$
    - **while** $(j \leftarrow x_j) \neq i'$
    - **do**
      - $e_j \leftarrow m$
    - **while** $(j \leftarrow x_j) \neq i'$

Berechnung Hilfsvariablen



m:

$a_{\_,n}$:

5 3 20 8 15 4 16 13 7 2 17

Knoten i	e_i	g_i
0	0	6
1	0	0
2	3	10
3	0	0
4	0	0
5	0	11
6	0	6
7	3	10
8	0	11
9	7	9
10	0	10
11	0	11

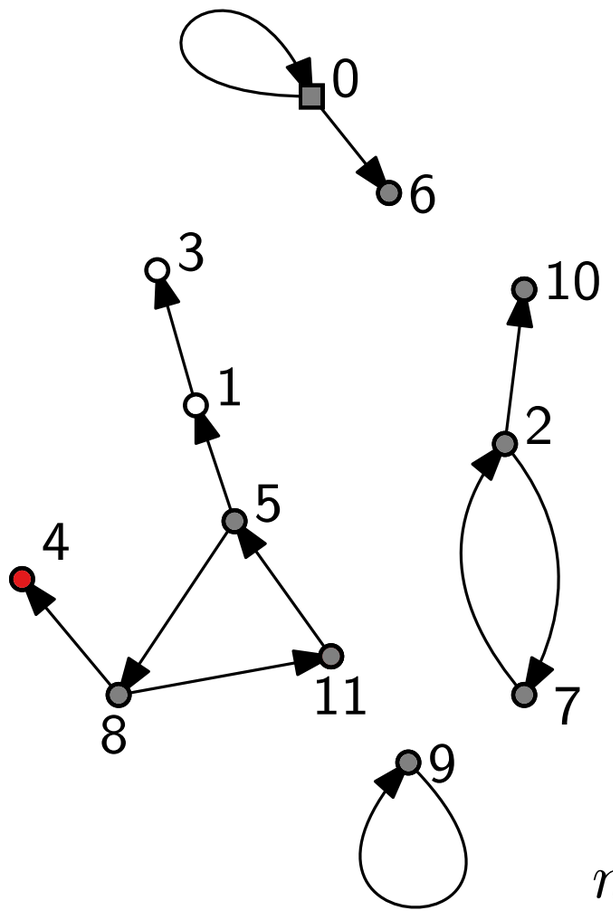
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**
    - **do**
      - $m \leftarrow \min\{m, a_{j,n}\}$
    - **while** $(j \leftarrow x_j) \neq i'$
    - **do**
      - $e_j \leftarrow m$
    - **while** $(j \leftarrow x_j) \neq i'$

Berechnung Hilfsvariablen



m:

$a_{\_,n}$:

5 3 20 8 15 4 16 13 7 2 17

Knoten i	e_i	g_i
0	0	6
1	0	0
2	3	10
3	0	0
4	0	0
5	0	11
6	0	6
7	3	10
8	0	11
9	7	9
10	0	10
11	0	11

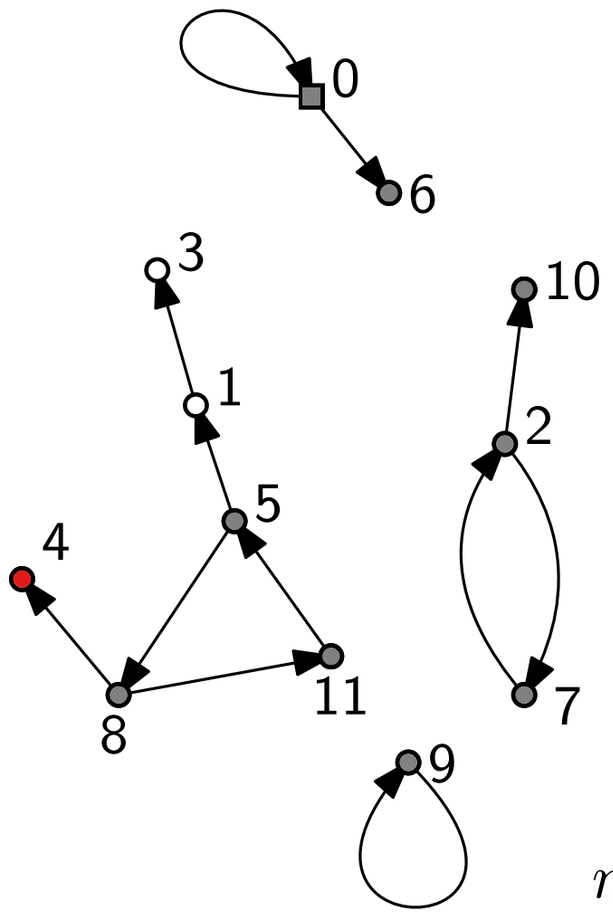
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**
    - **do**
      - $m \leftarrow \min\{m, a_{j,n}\}$
    - **while** $(j \leftarrow x_j) \neq i'$
    - **do**
      - $e_j \leftarrow m$
    - **while** $(j \leftarrow x_j) \neq i'$

Berechnung Hilfsvariablen



m:

$a_{\_,n}$:

5 3 20 8 15 4 16 13 7 2 17

Knoten i	e_i	g_i
0	0	6
1	0	0
2	3	10
3	0	0
4	0	4
5	0	11
6	0	6
7	3	10
8	0	11
9	7	9
10	0	10
11	0	11

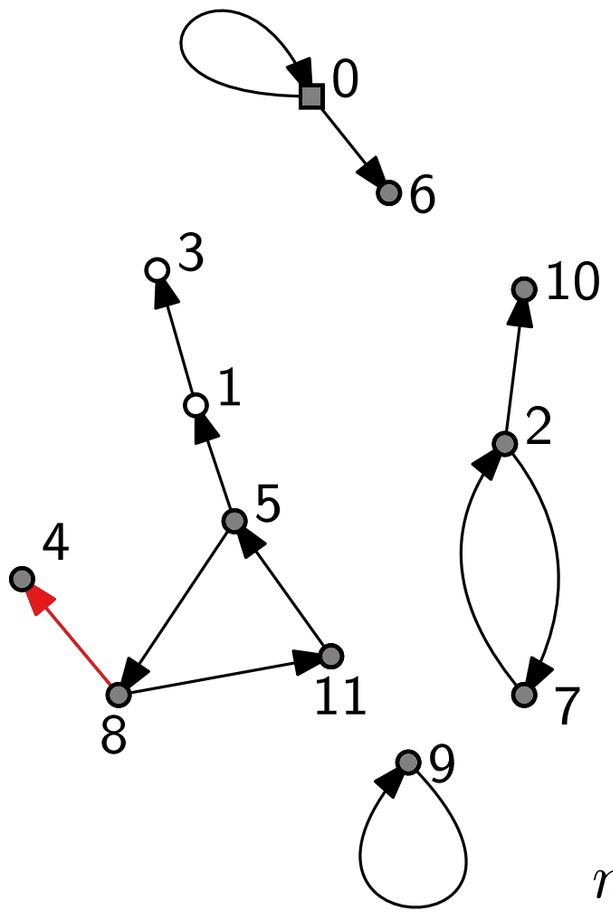
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**
    - **do**
      - $m \leftarrow \min\{m, a_{j,n}\}$
    - **while** $(j \leftarrow x_j) \neq i'$
    - **do**
      - $e_j \leftarrow m$
    - **while** $(j \leftarrow x_j) \neq i'$

Berechnung Hilfsvariablen



m:

$a_{\_,n}$:

5 3 20 8 15 4 16 13 7 2 17

Knoten i	e_i	g_i
0	0	6
1	0	0
2	3	10
3	0	0
4	0	4
5	0	11
6	0	6
7	3	10
8	0	11
9	7	9
10	0	10
11	0	11

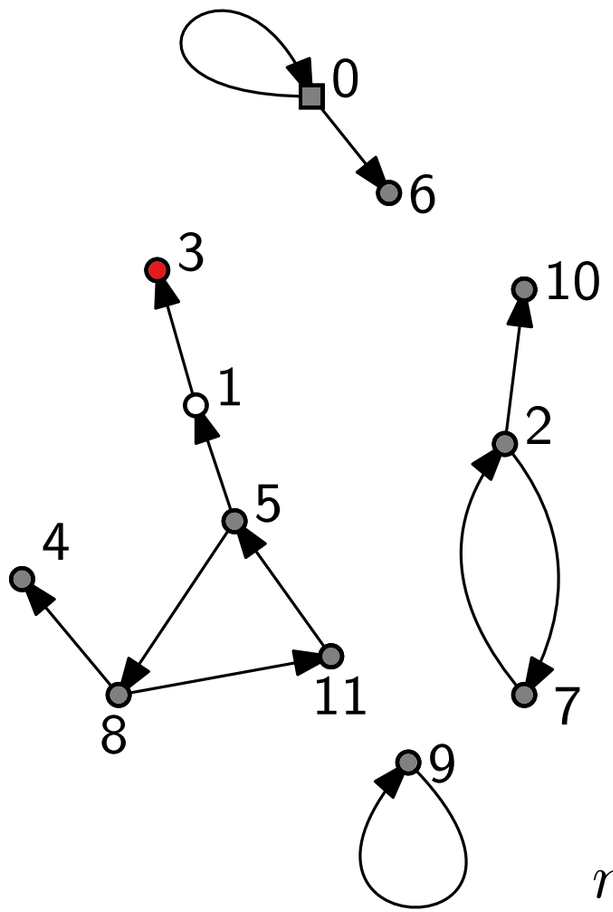
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**
    - **do**
      - $m \leftarrow \min\{m, a_{j,n}\}$
    - **while** $(j \leftarrow x_j) \neq i'$
    - **do**
      - $e_j \leftarrow m$
    - **while** $(j \leftarrow x_j) \neq i'$

Berechnung Hilfsvariablen



m:

$a_{\_,n}$:

5 3 20 8 15 4 16 13 7 2 17

Knoten i	e_i	g_i
0	0	6
1	0	0
2	3	10
3	0	0
4	0	4
5	0	11
6	0	6
7	3	10
8	0	11
9	7	9
10	0	10
11	0	11

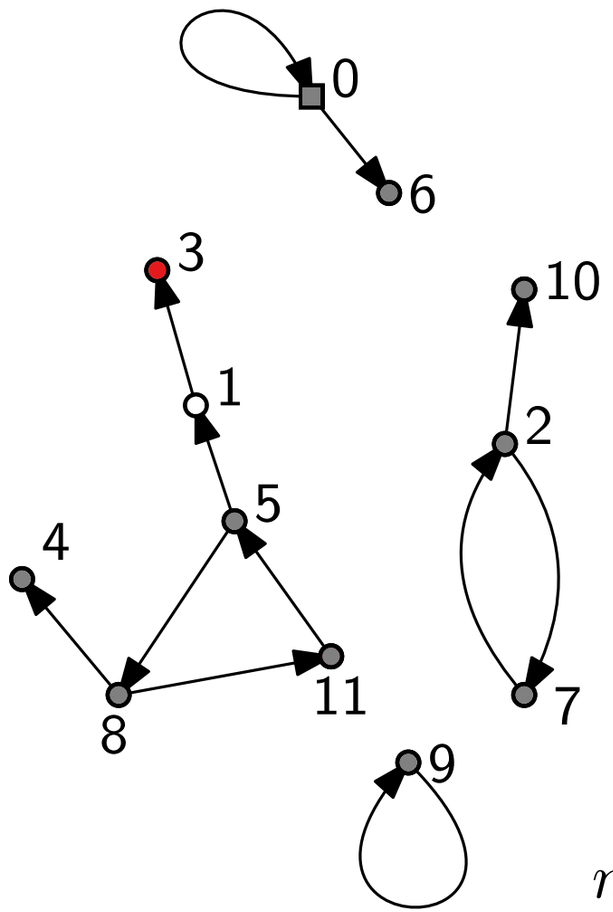
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**
    - **do**
      - $m \leftarrow \min\{m, a_{j,n}\}$
    - **while** $(j \leftarrow x_j) \neq i'$
    - **do**
      - $e_j \leftarrow m$
    - **while** $(j \leftarrow x_j) \neq i'$

Berechnung Hilfsvariablen



m:

$a_{\_,n}$:

5 3 20 8 15 4 16 13 7 2 17

Knoten i	e_i	g_i
0	0	6
1	0	0
2	3	10
3	0	3
4	0	4
5	0	11
6	0	6
7	3	10
8	0	11
9	7	9
10	0	10
11	0	11

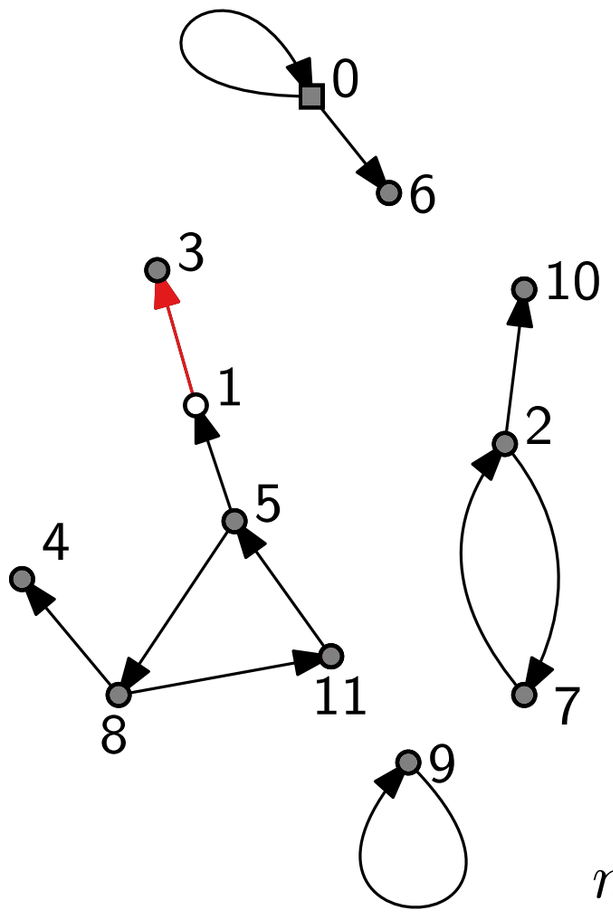
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**
    - **do**
      - $m \leftarrow \min\{m, a_{j,n}\}$
    - **while** $(j \leftarrow x_j) \neq i'$
    - **do**
      - $e_j \leftarrow m$
    - **while** $(j \leftarrow x_j) \neq i'$

Berechnung Hilfsvariablen



m:

$a_{\_,n}$:

5 3 20 8 15 4 16 13 7 2 17

Knoten i	e_i	g_i
0	0	6
1	0	0
2	3	10
3	0	3
4	0	4
5	0	11
6	0	6
7	3	10
8	0	11
9	7	9
10	0	10
11	0	11

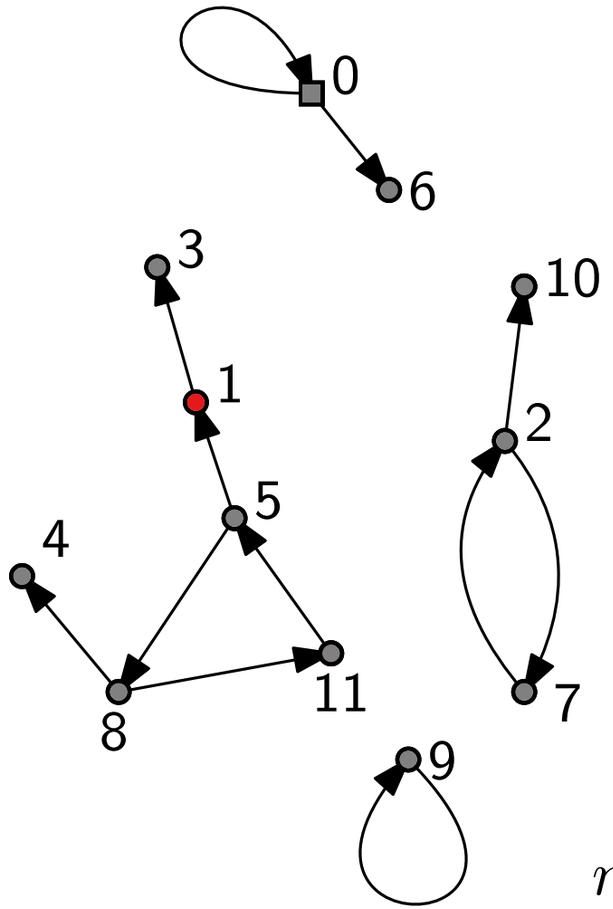
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**
    - **do**
      - $m \leftarrow \min\{m, a_{j,n}\}$
    - **while** $(j \leftarrow x_j) \neq i'$
    - **do**
      - $e_j \leftarrow m$
    - **while** $(j \leftarrow x_j) \neq i'$

Berechnung Hilfsvariablen



m:

$a_{\_,n}$:

5 3 20 8 15 4 16 13 7 2 17

Knoten i	e_i	g_i
0	0	6
1	0	0
2	3	10
3	0	3
4	0	4
5	0	11
6	0	6
7	3	10
8	0	11
9	7	9
10	0	10
11	0	11

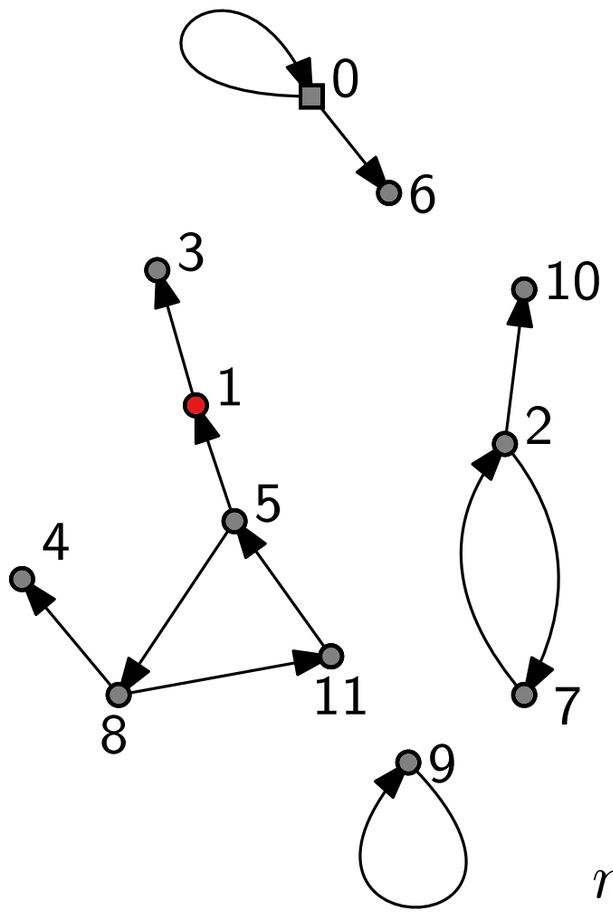
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**
    - **do**
      - $m \leftarrow \min\{m, a_{j,n}\}$
    - **while** $(j \leftarrow x_j) \neq i'$
    - **do**
      - $e_j \leftarrow m$
    - **while** $(j \leftarrow x_j) \neq i'$

Berechnung Hilfsvariablen



m:

$a_{\_,n}$:

5 3 20 8 15 4 16 13 7 2 17

Knoten i	e_i	g_i
0	0	6
1	0	3
2	3	10
3	0	3
4	0	4
5	0	11
6	0	6
7	3	10
8	0	11
9	7	9
10	0	10
11	0	11

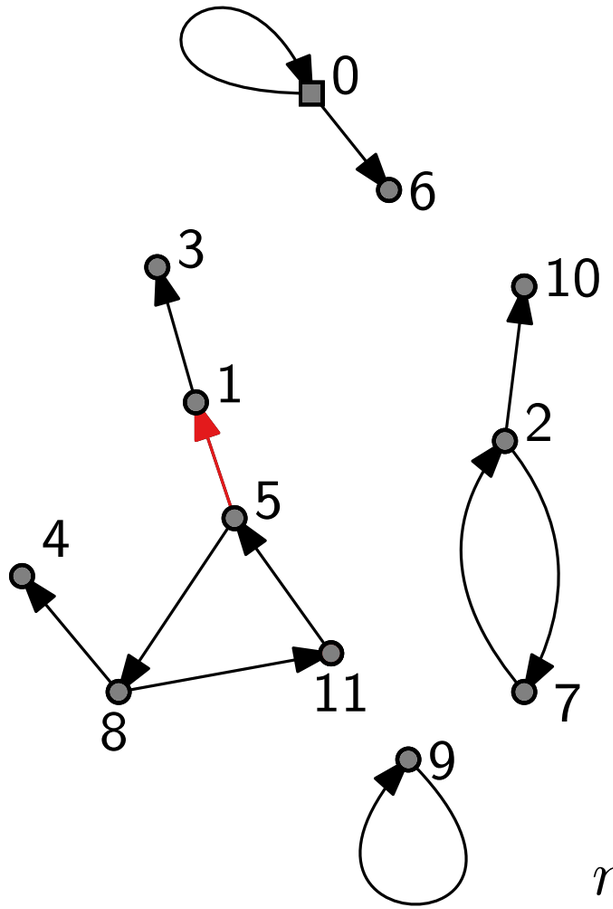
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**
    - **do**
      - $m \leftarrow \min\{m, a_{j,n}\}$
    - **while** $(j \leftarrow x_j) \neq i'$
    - **do**
      - $e_j \leftarrow m$
    - **while** $(j \leftarrow x_j) \neq i'$

Berechnung Hilfsvariablen



m:

$a_{\_,n}$:

5 3 20 8 15 4 16 13 7 2 17

Knoten i	e_i	g_i
0	0	6
1	0	3
2	3	10
3	0	3
4	0	4
5	0	11
6	0	6
7	3	10
8	0	11
9	7	9
10	0	10
11	0	11

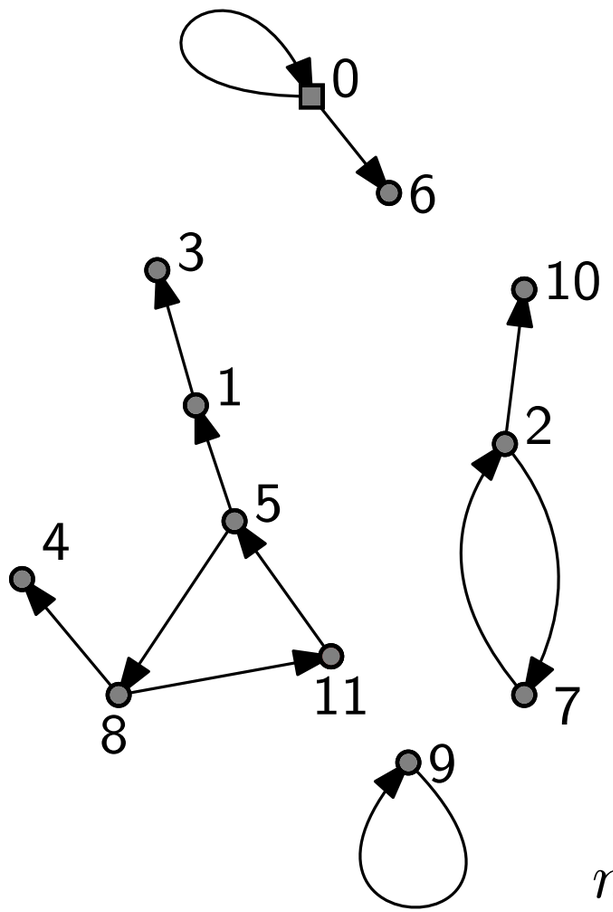
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**
    - **do**
      - $m \leftarrow \min\{m, a_{j,n}\}$
    - **while** $(j \leftarrow x_j) \neq i'$
    - **do**
      - $e_j \leftarrow m$
    - **while** $(j \leftarrow x_j) \neq i'$

Berechnung Hilfsvariablen



m:

$a_{\_,n}$:

5 3 20 8 15 4 16 13 7 2 17

Knoten i	e_i	g_i
0	0	6
1	0	3
2	3	10
3	0	3
4	0	4
5	0	11
6	0	6
7	3	10
8	0	11
9	7	9
10	0	10
11	0	11

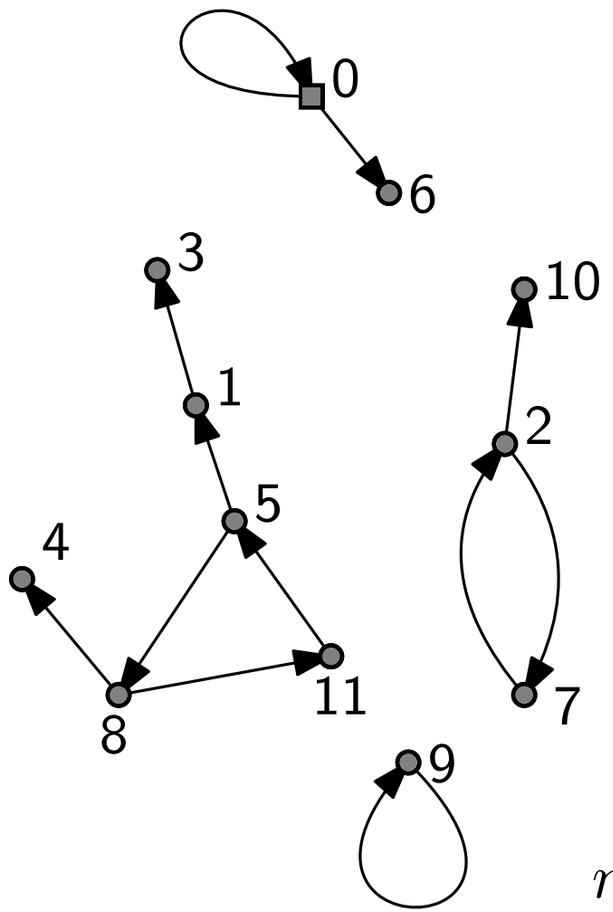
e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**
    - **do**
      - $m \leftarrow \min\{m, a_{j,n}\}$
    - **while** $(j \leftarrow x_j) \neq i'$
    - **do**
      - $e_j \leftarrow m$
    - **while** $(j \leftarrow x_j) \neq i'$

Berechnung Hilfsvariablen



m:

$a_{\_,n}$:

5 3 20 8 15 4 16 13 7 2 17

Knoten i	e_i	g_i
0	0	6
1	0	3
2	3	10
3	0	3
4	0	4
5	0	11
6	0	6
7	3	10
8	0	11
9	7	9
10	0	10
11	0	11

e_i – Ersparnis

g_i – größtes Kind

initialisiereHilfsvariablen()

for $i = n$ **downto** 1 **do**
- $j \leftarrow i$
- **while** $g_j = 0$ **do**
  - $g_j \leftarrow i$
  - $j \leftarrow x_j$
- **if** $g_j = i$ **then**
  - $i', m \leftarrow (j, a_{j,n})$
  - **if** $i' \neq n$ **then**
    - **do**
      - $m \leftarrow \min\{m, a_{j,n}\}$
    - **while** $(j \leftarrow x_j) \neq i'$
    - **do**
      - $e_j \leftarrow m$
    - **while** $(j \leftarrow x_j) \neq i'$

$O(n)$

Pseudocode

schnellsterSpeedrun(s, x, a)

```
normalisiereAbkürzungen(s, x, a)
initialisiereHilfsvariablen(s,x,a)
O ← [∞] * (n + 1)
O[g_0] ← 0
for i = g_0 to n do
    if i = g_i then
        i' ← i
        besucht ← [false] * (n + 1)
        while g_{i'} = i und besucht[i'] = false do
            besucht[i'] ← true
            for j = g_0 to n − 1 do
                O[i] ← min{O[i], O[j] + a_{i',j} − e_{i'}}
            i' ← x_{i'}
return O[n]
```

e_i – Ersparnis

g_i – größtes Kind

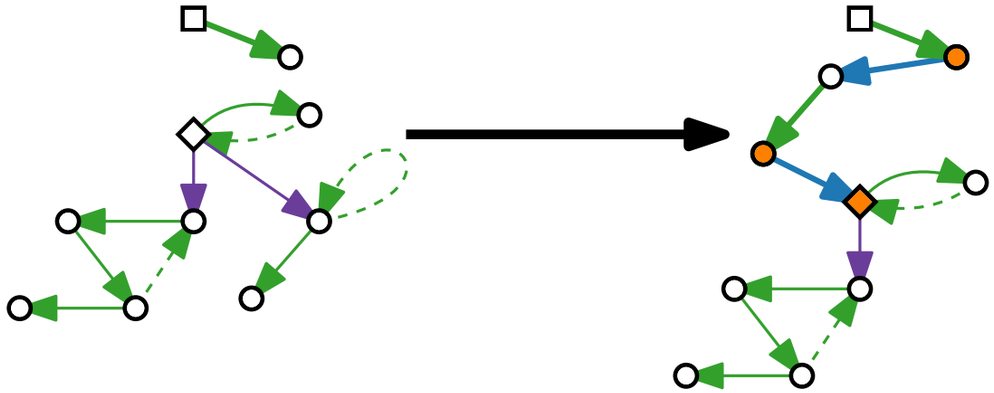
Pseudocode

schnellsterSpeedrun(s, x, a)

```
normalisiereAbkürzungen(s, x, a)
initialisiereHilfsvariablen(s,x,a)
O ← [∞] * (n + 1)
O[g_0] ← 0
for i = g_0 to n do
    if i = g_i then
        i' ← i
        besucht ← [false] * (n + 1)
        while g_{i'} = i und besucht[i'] = false do
            besucht[i'] ← true
            for j = g_0 to n − 1 do
                O[i] ← min{O[i], O[j] + a_{i',j} − e_{i'}}
            i' ← x_{i'}
return O[n]
```

e_i – Ersparnis

g_i – größtes Kind



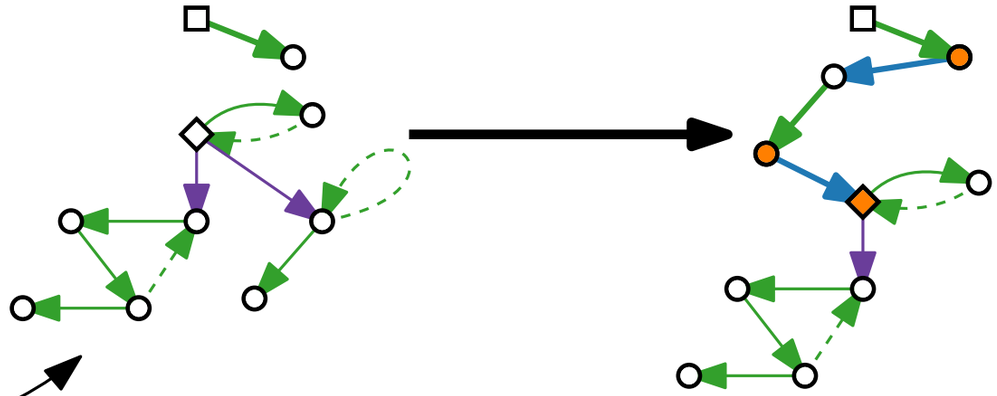
Pseudocode

schnellsterSpeedrun(s, x, a)

normalisiereAbkürzungen(s, x, a)
initialisiereHilfsvariablen(s,x,a)
$O \leftarrow [\infty] * (n+1)$
$O[g_0] \leftarrow 0$
for $i = g_0$ **to** n **do**
  if $i = g_i$ **then**
    $i' \leftarrow i$
    besucht $\leftarrow [false] * (n+1)$
    while $g_{i'} = i$ und besucht$[i'] = false$ **do**
      besucht$[i'] \leftarrow true$
      for $j = g_0$ **to** $n-1$ **do**
        $O[i] \leftarrow \min\{O[i], O[j] + a_{i',j} - e_{i'}\}$
      $i' \leftarrow x_{i'}$
return $O[n]$ +?

e_i – Ersparnis

g_i – größtes Kind



Pseudocode

schnellsterSpeedrun(s, x, a)

normalisiereAbkürzungen(s, x, a)
initialisiereHilfsvariablen(s,x,a)
$O \leftarrow [\infty] * (n+1)$
$O[g_0] \leftarrow 0$
for $i = g_0$ **to** n **do**
  if $i = g_i$ **then**
    $i' \leftarrow i$
    besucht $\leftarrow [false] * (n+1)$
    while $g_{i'} = i$ und besucht$[i'] = false$ **do**
      besucht$[i'] \leftarrow true$
      for $j = g_0$ **to** $n-1$ **do**
        $O[i] \leftarrow \min\{O[i], O[j] + a_{i',j} - e_{i'}\}$
      $i' \leftarrow x_{i'}$
return $O[n] + ?$

} $O(n^2)$

e_i – Ersparnis

g_i – größtes Kind

Pseudocode

schnellsterSpeedrun(s, x, a)

normalisiereAbkürzungen(s, x, a) } $O(n^2)$
initialisiereHilfsvariablen(s,x,a) } $O(n)$
$O \leftarrow [\infty] * (n+1)$
$O[g_0] \leftarrow 0$
for $i = g_0$ **to** n **do**
  if $i = g_i$ **then**
    $i' \leftarrow i$
    besucht $\leftarrow [false] * (n+1)$
    while $g_{i'} = i$ und besucht$[i'] = false$ **do**
      besucht$[i'] \leftarrow true$
      for $j = g_0$ **to** $n-1$ **do**
        $O[i] \leftarrow \min\{O[i], O[j] + a_{i',j} - e_{i'}\}$
      $i' \leftarrow x_{i'}$
return $O[n]$ +?

e_i – Ersparnis

g_i – größtes Kind

Pseudocode

schnellsterSpeedrun(s, x, a)

normalisiereAbkürzungen(s, x, a) } $O(n^2)$
initialisiereHilfsvariablen(s,x,a) } $O(n)$
$O \leftarrow [\infty] * (n+1)$
$O[g_0] \leftarrow 0$
for $i = g_0$ **to** n **do**
  if $i = g_i$ **then**
    $i' \leftarrow i$
    besucht $\leftarrow [false] * (n+1)$ } $O(n)$
    while $g_{i'} = i$ und besucht$[i'] = false$ **do**
      besucht$[i'] \leftarrow true$
      for $j = g_0$ **to** $n-1$ **do**
        $O[i] \leftarrow \min\{O[i], O[j] + a_{i',j} - e_{i'}\}$
      $i' \leftarrow x_{i'}$
return $O[n]$ +?

e_i – Ersparnis

g_i – größtes Kind

Pseudocode

schnellsterSpeedrun(s, x, a)

normalisiereAbkürzungen(s, x, a) } $O(n^2)$
initialisiereHilfsvariablen(s,x,a) } $O(n)$
$O \leftarrow [\infty] * (n+1)$
$O[g_0] \leftarrow 0$
for $i = g_0$ **to** n **do**
  if $i = g_i$ **then**
    $i' \leftarrow i$
    besucht $\leftarrow [false] * (n+1)$ } $O(n)$
    while $g_{i'} = i$ und besucht$[i'] = false$ **do**
      besucht$[i'] \leftarrow true$ } $O(1)$
      for $j = g_0$ **to** $n-1$ **do**
        $O[i] \leftarrow \min\{O[i], O[j] + a_{i',j} - e_{i'}\}$
      $i' \leftarrow x_{i'}$
return $O[n] + ?$

e_i – Ersparnis

g_i – größtes Kind

Pseudocode

schnellsterSpeedrun(s, x, a)

normalisiereAbkürzungen(s, x, a) } $O(n^2)$
initialisiereHilfsvariablen(s,x,a) } $O(n)$
$O \leftarrow [\infty] * (n+1)$
$O[g_0] \leftarrow 0$
for $i = g_0$ **to** n **do**
  if $i = g_i$ **then**
    $i' \leftarrow i$
    besucht $\leftarrow [false] * (n+1)$ } $O(n)$
    while $g_{i'} = i$ und besucht$[i'] = false$ **do**
      besucht$[i'] \leftarrow true$ } $O(1)$
      for $j = g_0$ **to** $n-1$ **do**
        $O[i] \leftarrow \min\{O[i], O[j] + a_{i',j} - e_{i'}\}$ } $O(1)$
      $i' \leftarrow x_{i'}$
return $O[n] + ?$

e_i – Ersparnis

g_i – größtes Kind

Pseudocode

schnellsterSpeedrun(s, x, a)

normalisiereAbkürzungen(s, x, a)	} $O(n^2)$
initialisiereHilfsvariablen(s,x,a)	} $O(n)$
$O \leftarrow [\infty] * (n+1)$	
$O[g_0] \leftarrow 0$	
for $i = g_0$ **to** n **do**	
if $i = g_i$ **then**	
$i' \leftarrow i$	
besucht $\leftarrow [false] * (n+1)$	} $O(n)$
while $g_{i'} = i$ und besucht$[i'] = false$ **do**	
besucht$[i'] \leftarrow true$	} $O(1)$
for $j = g_0$ **to** $n-1$ **do**	
$O[i] \leftarrow \min\{O[i], O[j] + a_{i',j} - e_{i'}\}$	} $O(1)$
$i' \leftarrow x_{i'}$	} $O(1)$
return $O[n] + ?$	

e_i – Ersparnis

g_i – größtes Kind

Pseudocode

schnellsterSpeedrun(s, x, a)

normalisiereAbkürzungen(s, x, a) } $O(n^2)$
initialisiereHilfsvariablen(s,x,a) } $O(n)$
$O \leftarrow [\infty] * (n+1)$
$O[g_0] \leftarrow 0$
for $i = g_0$ **to** n **do**
  if $i = g_i$ **then**
    $i' \leftarrow i$
    besucht $\leftarrow [false] * (n+1)$ } $O(n)$
    while $g_{i'} = i$ und besucht$[i'] = false$ **do**
      besucht$[i'] \leftarrow true$ } $O(1)$
      for $j = g_0$ **to** $n-1$ **do**
        $O[i] \leftarrow \min\{O[i], O[j] + a_{i',j} - e_{i'}\}$ } $O(1)$ } $O(n)$
      $i' \leftarrow x_{i'}$ } $O(1)$
return $O[n] + ?$

e_i – Ersparnis

g_i – größtes Kind

Pseudocode

schnellsterSpeedrun(s, x, a)

normalisiereAbkürzungen(s, x, a)	$\} O(n^2)$
initialisiereHilfsvariablen(s,x,a)	$\} O(n)$
$O \leftarrow [\infty] * (n+1)$	
$O[g_0] \leftarrow 0$	
for $i = g_0$ **to** n **do**	
if $i = g_i$ **then**	
$i' \leftarrow i$	
besucht $\leftarrow [false] * (n+1)$	$\} O(n)$
while $g_{i'} = i$ und besucht$[i'] = false$ **do**	
besucht$[i'] \leftarrow true$	$\} O(1)$
for $j = g_0$ **to** $n-1$ **do**	$\Big\}$
$O[i] \leftarrow \min\{O[i], O[j] + a_{i',j} - e_{i'}\}$	$\} O(1)$ $\Big\} O(n)$
$i' \leftarrow x_{i'}$	$\} O(1)$
return $O[n] + ?$	

e_i – Ersparnis

g_i – größtes Kind

Pseudocode

schnellsterSpeedrun(s, x, a)

normalisiereAbkürzungen(s, x, a) } $O(n^2)$
initialisiereHilfsvariablen(s,x,a) } $O(n)$
$O \leftarrow [\infty] * (n+1)$
$O[g_0] \leftarrow 0$
for $i = g_0$ **to** n **do**
  if $i = g_i$ **then**
    $i' \leftarrow i$
    besucht $\leftarrow [false] * (n+1)$ } $O(n)$
    while $g_{i'} = i$ und besucht$[i'] = false$ **do**
      besucht$[i'] \leftarrow true$ } $O(1)$
      for $j = g_0$ **to** $n-1$ **do**
        $O[i] \leftarrow \min\{O[i], O[j] + a_{i',j} - e_{i'}\}$ } $O(1)$ } $O(n)$
      $i' \leftarrow x_{i'}$ } $O(1)$
} $O(n)$
return $O[n] + ?$

e_i – Ersparnis

g_i – größtes Kind

Pseudocode

schnellsterSpeedrun(s, x, a)

normalisiereAbkürzungen(s, x, a) } $O(n^2)$
initialisiereHilfsvariablen(s,x,a) } $O(n)$
$O \leftarrow [\infty] * (n+1)$
$O[g_0] \leftarrow 0$
for $i = g_0$ **to** n **do**
  if $i = g_i$ **then**
    $i' \leftarrow i$
    besucht $\leftarrow [false] * (n+1)$ } $O(n)$
    while $g_{i'} = i$ und besucht$[i'] = false$ **do**
      besucht$[i'] \leftarrow true$ } $O(1)$
      for $j = g_0$ **to** $n-1$ **do**
        $O[i] \leftarrow \min\{O[i], O[j] + a_{i',j} - e_{i'}\}$ } $O(1)$ } $O(n)$
      $i' \leftarrow x_{i'}$ } $O(1)$
} $O(n)$
return $O[n]$ +?

$\Rightarrow$ Laufzeit $O(n^2)$

e_i – Ersparnis

g_i – größtes Kind

Pseudocode

schnellsterSpeedrun(s, x, a)

normalisiereAbkürzungen(s, x, a) } $O(n^2)$
initialisiereHilfsvariablen(s,x,a) } $O(n)$
$O \leftarrow [\infty] * (n+1)$
$O[g_0] \leftarrow 0$
for $i = g_0$ **to** n **do**
  if $i = g_i$ **then**
    $i' \leftarrow i$
    besucht $\leftarrow [false] * (n+1)$ } $O(n)$
    while $g_{i'} = i$ und besucht$[i'] = false$ **do**
      besucht$[i'] \leftarrow true$ } $O(1)$
      for $j = g_0$ **to** $n-1$ **do**
        $O[i] \leftarrow \min\{O[i], O[j] + a_{i',j} - e_{i'}\}$ } $O(1)$ } $O(n)$
      $i' \leftarrow x_{i'}$ } $O(1)$
} $O(n)$
return $O[n]$ +?

e_i – Ersparnis

g_i – größtes Kind

$\Rightarrow$ Laufzeit $O(n^2)$ ✓

Pseudocode

schnellsterSpeedrun(s, x, a)

normalisiereAbkürzungen(s, x, a) } $O(n^2)$
initialisiereHilfsvariablen(s,x,a) } $O(n)$
$O \leftarrow [\infty] * (n+1)$
$O[g_0] \leftarrow 0$
for $i = g_0$ **to** n **do**
  if $i = g_i$ **then**
    $i' \leftarrow i$
    besucht $\leftarrow [false] * (n+1)$ } $O(n)$
    while $g_{i'} = i$ und besucht$[i'] = false$ **do**
      besucht$[i'] \leftarrow true$ } $O(1)$
      for $j = g_0$ **to** $n-1$ **do**
        $O[i] \leftarrow \min\{O[i], O[j] + a_{i',j} - e_{i'}\}$ } $O(1)$ } $O(n)$
      $i' \leftarrow x_{i'}$ } $O(1)$
} $O(n)$
return $O[n]$ +?

e_i – Ersparnis

g_i – größtes Kind

$\Rightarrow$ Laufzeit $O(n^2)$ ✓