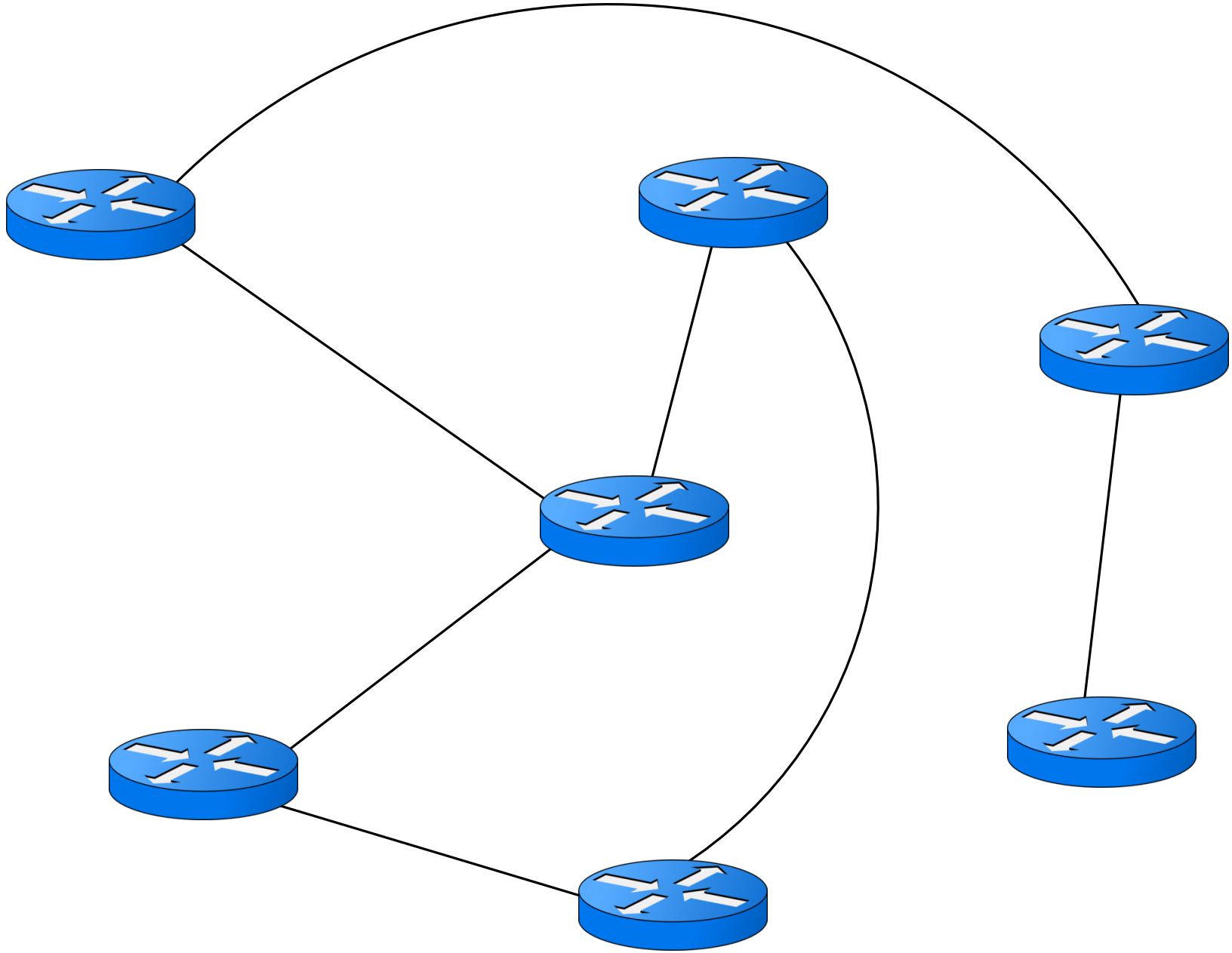
Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Seminar
Algorithmen für Programmierwettbewerbe

Problem C
A Series of Tubes

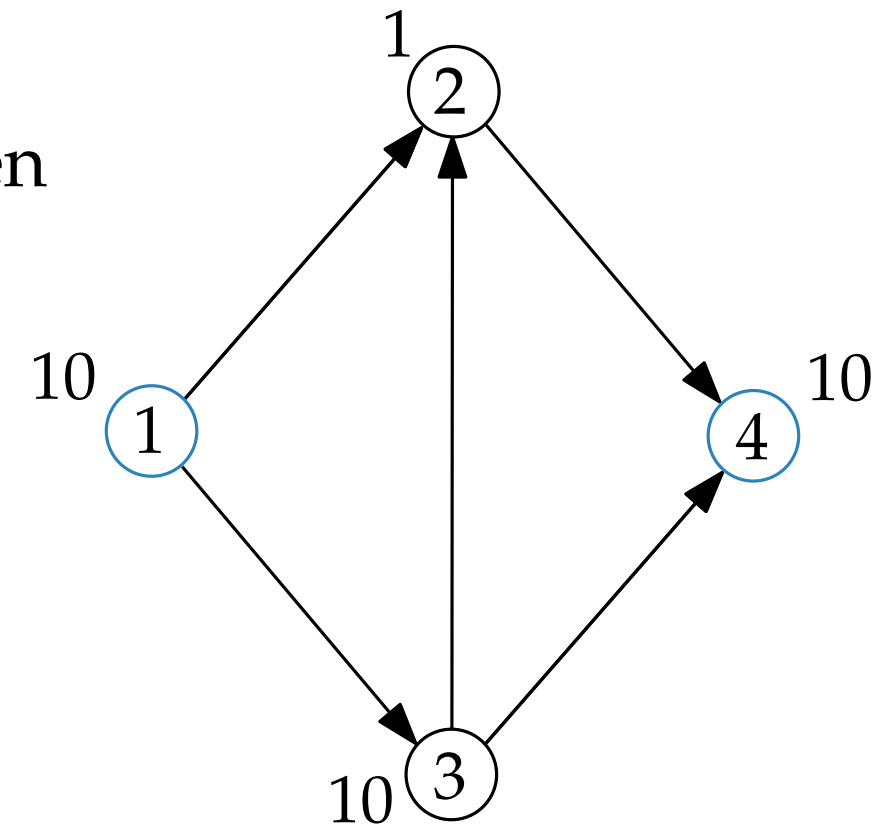
Martin Klüpfel

Ferdinand Zink



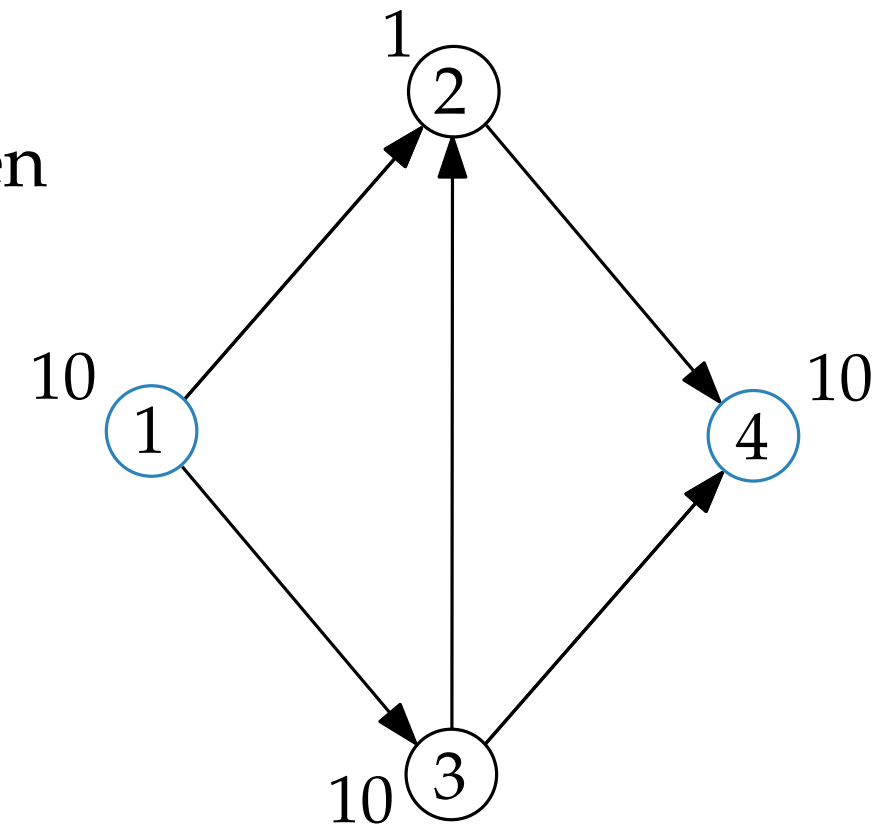
Problemstellung

Gegeben:
- Routernetzwerk
- dazugehörige Bearbeitungszeiten
- Start- und Endknoten



Problemstellung

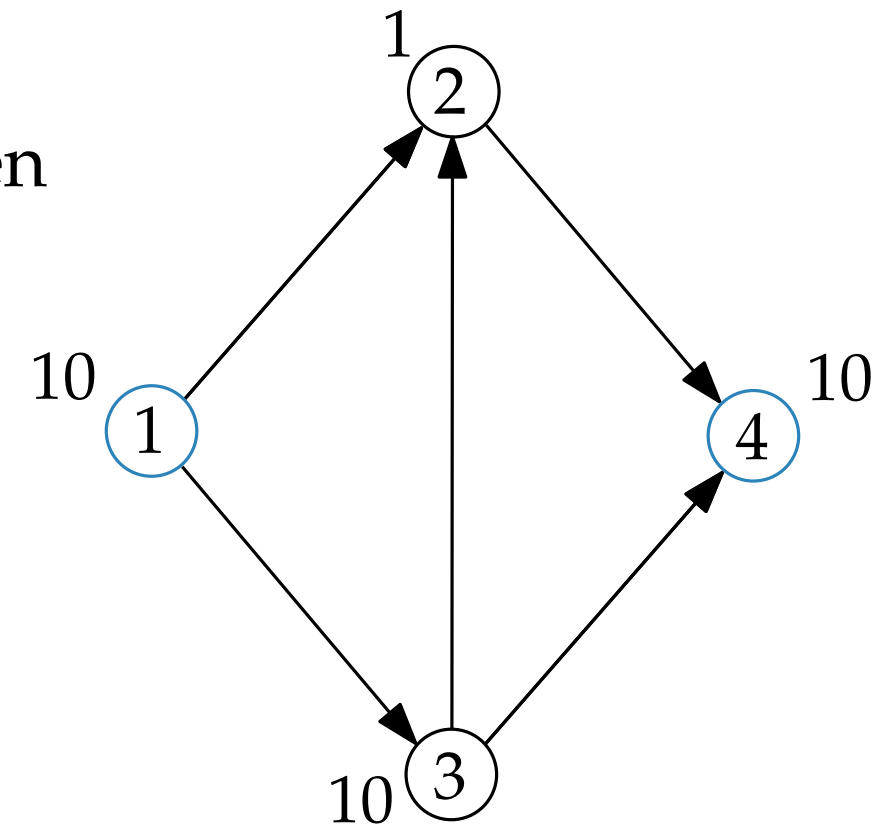
Gegeben:
- Routernetzwerk
- dazugehörige Bearbeitungszeiten
- Start- und Endknoten



Ziel:
Bestimmung der Länge eines kürzesten Pfades

Problemstellung

Gegeben:
- Routernetzwerk
- dazugehörige Bearbeitungszeiten
- Start- und Endknoten

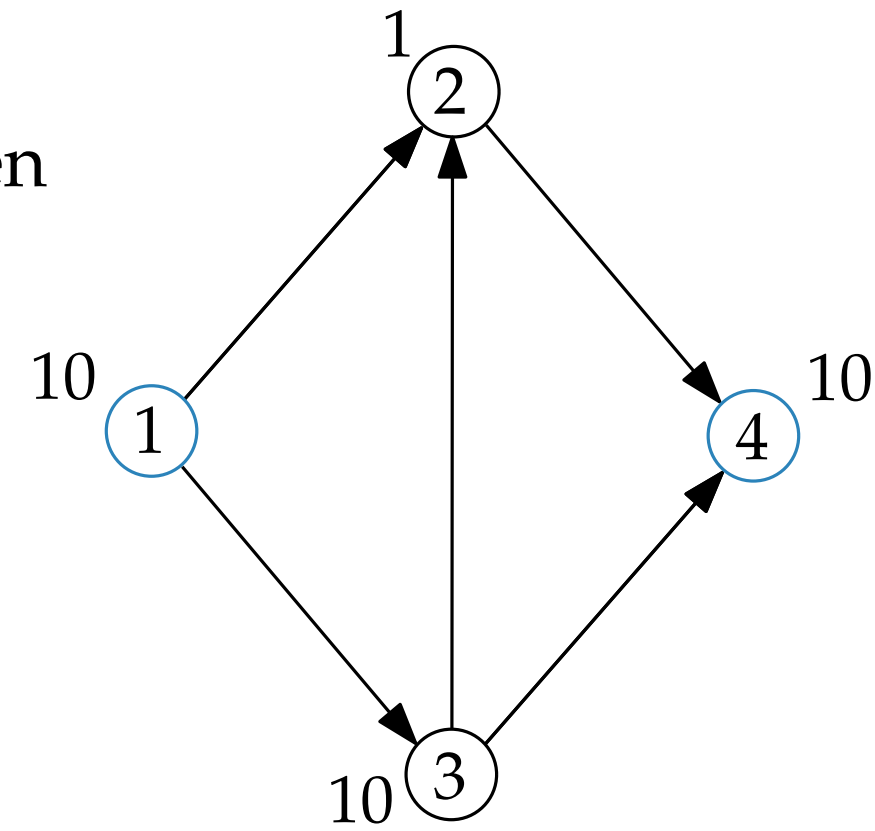


Ziel:
Bestimmung der Länge eines kürzesten Pfades

Aber das war noch nicht das ganze Problem...

Problemstellung

Gegeben:
- Routernetzwerk
- dazugehörige Bearbeitungszeiten
- Start- und Endknoten
- Routingregeln



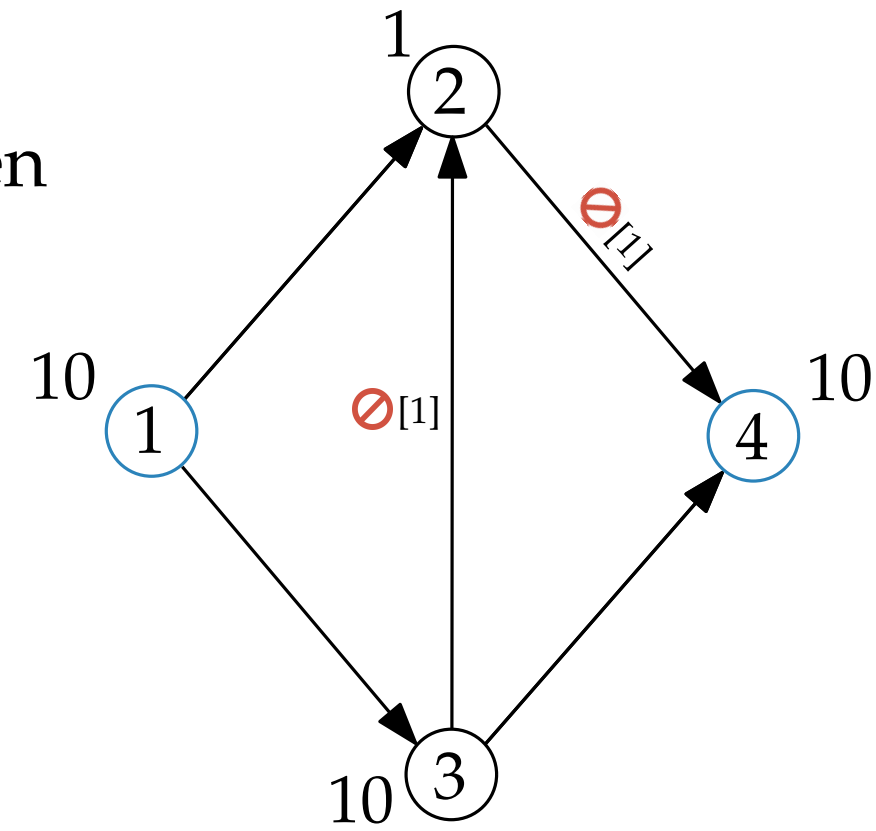
Ziel:
Bestimmung der Länge eines kürzesten Pfades

Aber das war noch nicht das ganze Problem...

Problemstellung

Gegeben:
- Routernetzwerk
- dazugehörige Bearbeitungszeiten
- Start- und Endknoten
- Routingregeln

Ziel:
Bestimmung der Länge eines kürzesten Pfades



Aber das war noch nicht das ganze Problem...

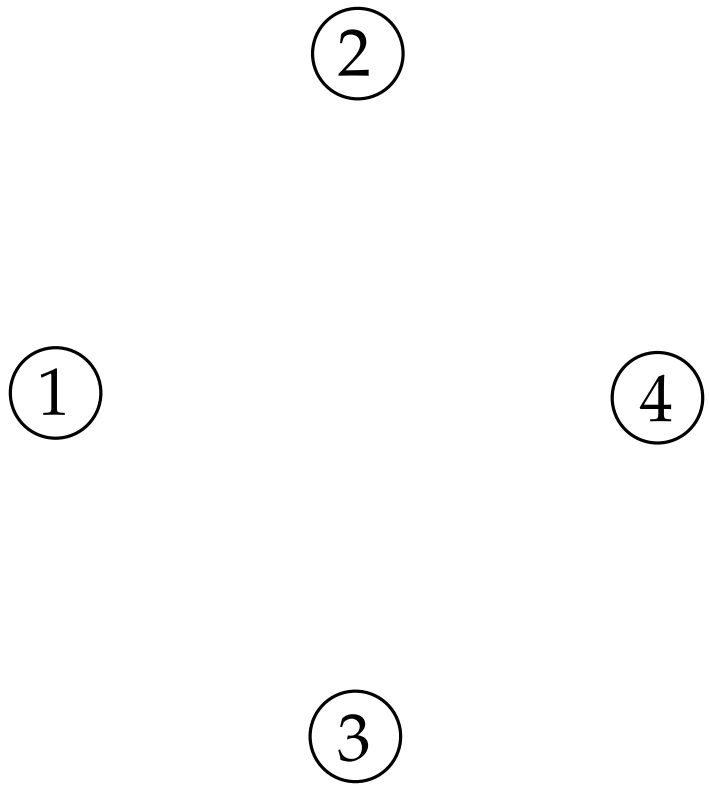
Problemstellung

Eingabe:

Problemstellung

Eingabe:

4

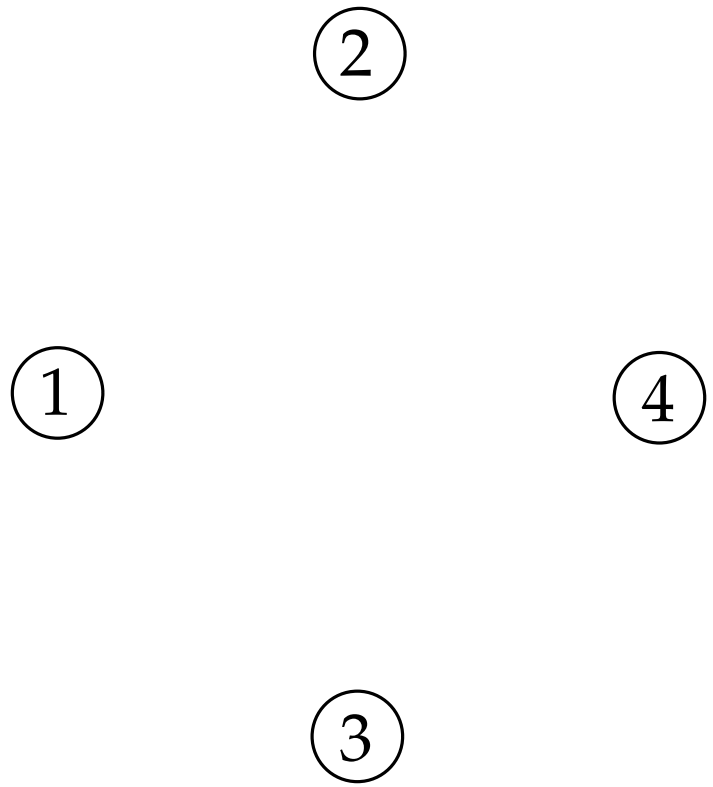


Problemstellung

Eingabe:

4

2 10

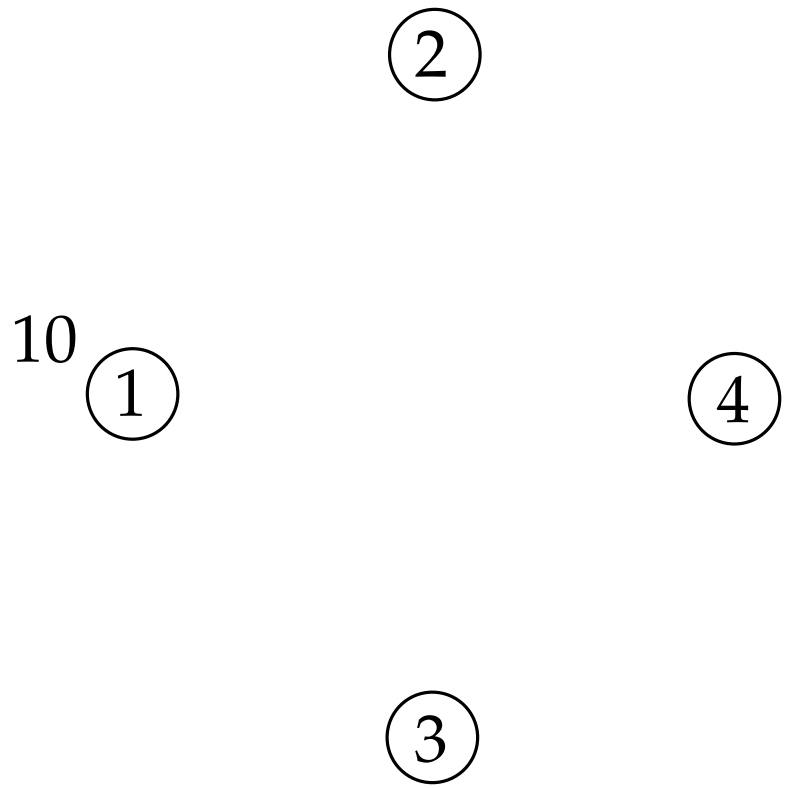


Problemstellung

Eingabe:

4

2 10

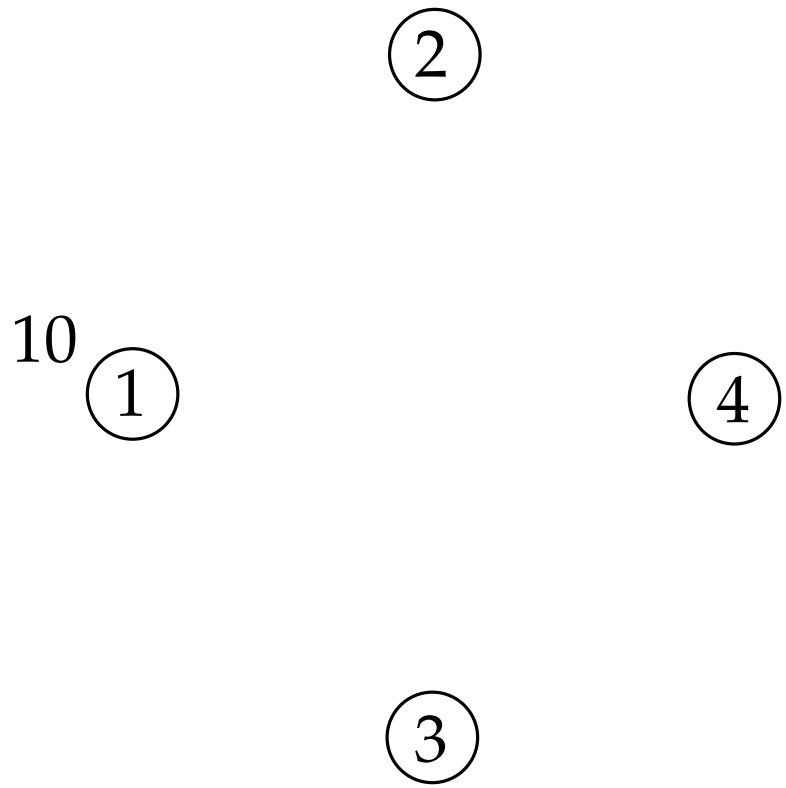


Problemstellung

Eingabe:

4

```
2 10
0 2
0 3
```

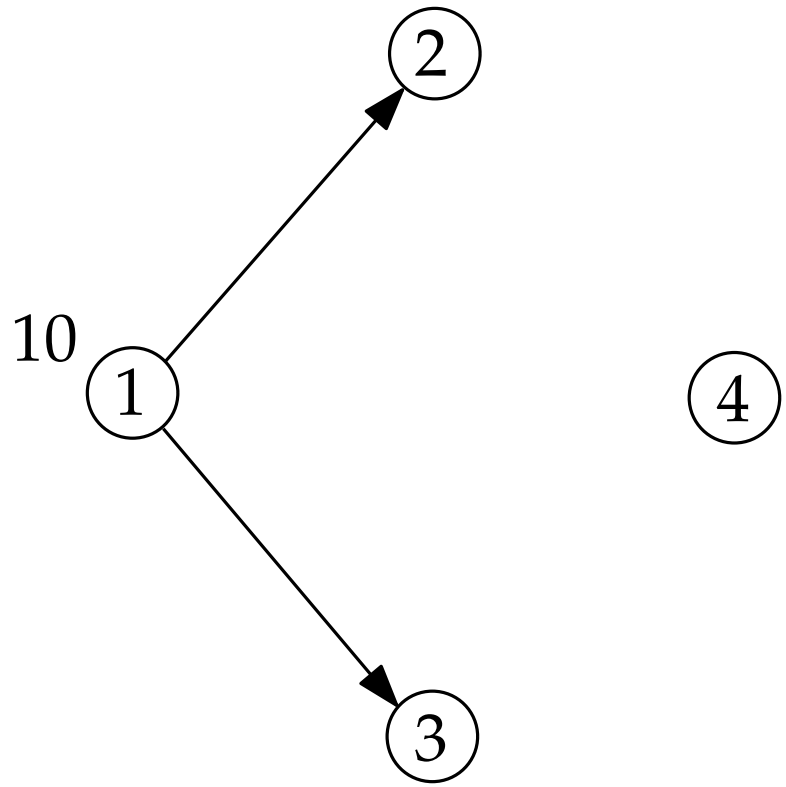


Problemstellung

Eingabe:

4

```
2 10
0 2
0 3
```



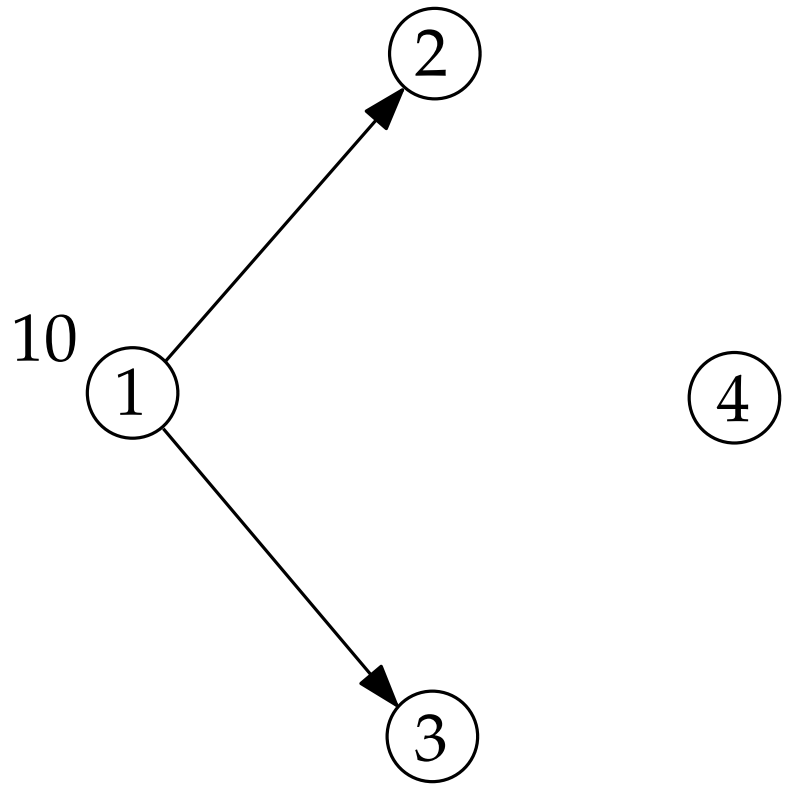
Problemstellung

Eingabe:

4

2 10
0 2
0 3

1 1



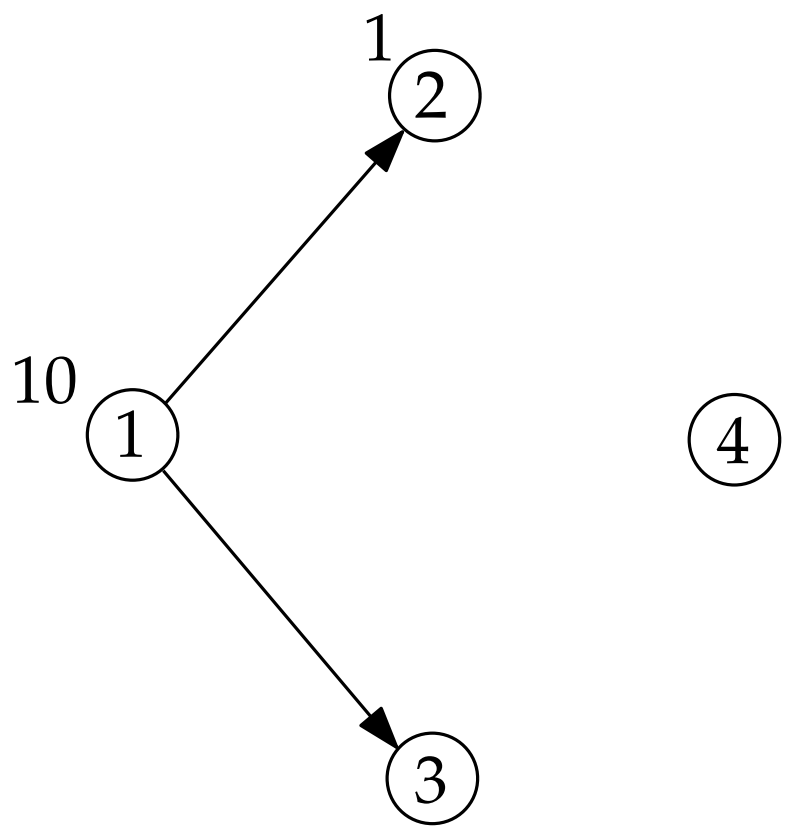
Problemstellung

Eingabe:

4

2 10
0 2
0 3

1 1



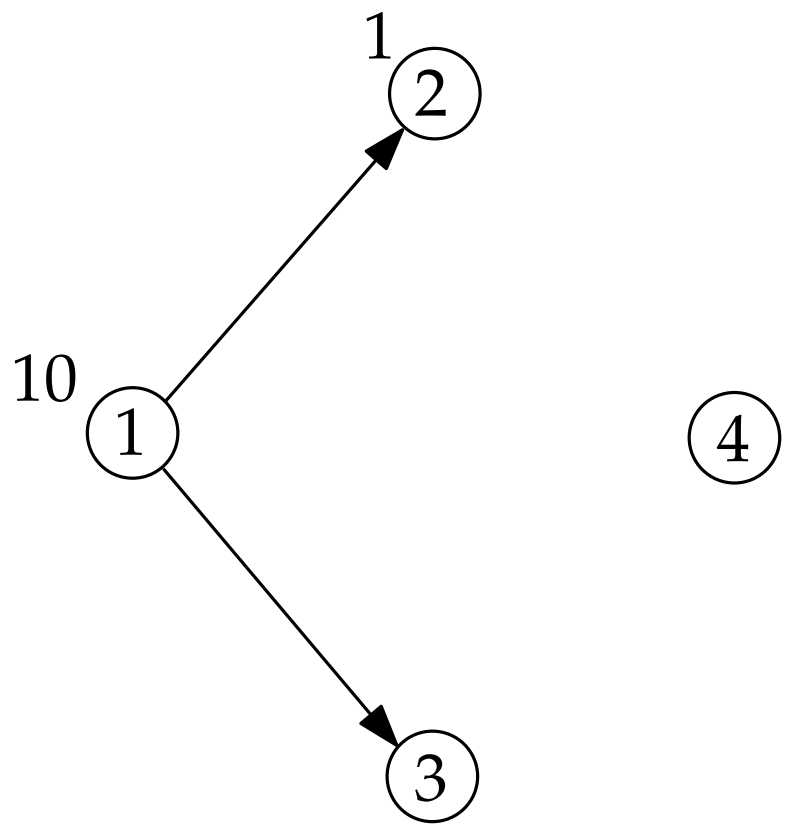
Problemstellung

Eingabe:

4

```
2 10
0 2
0 3
```

```
1 1
1 4 1
```



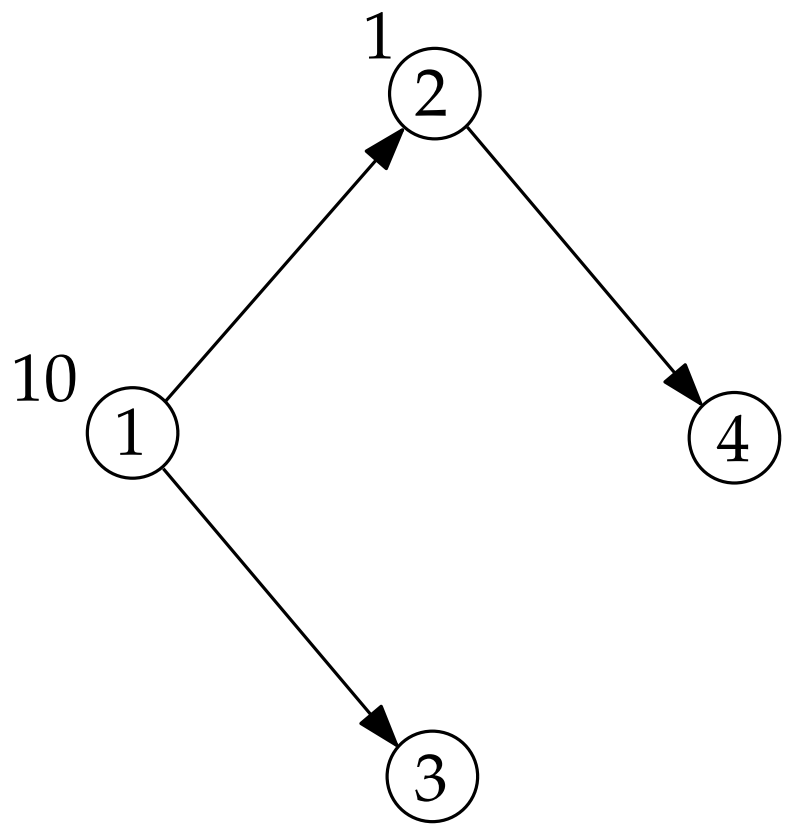
Problemstellung

Eingabe:

4

```
2 10
0 2
0 3
```

```
1 1
1 4 1
```



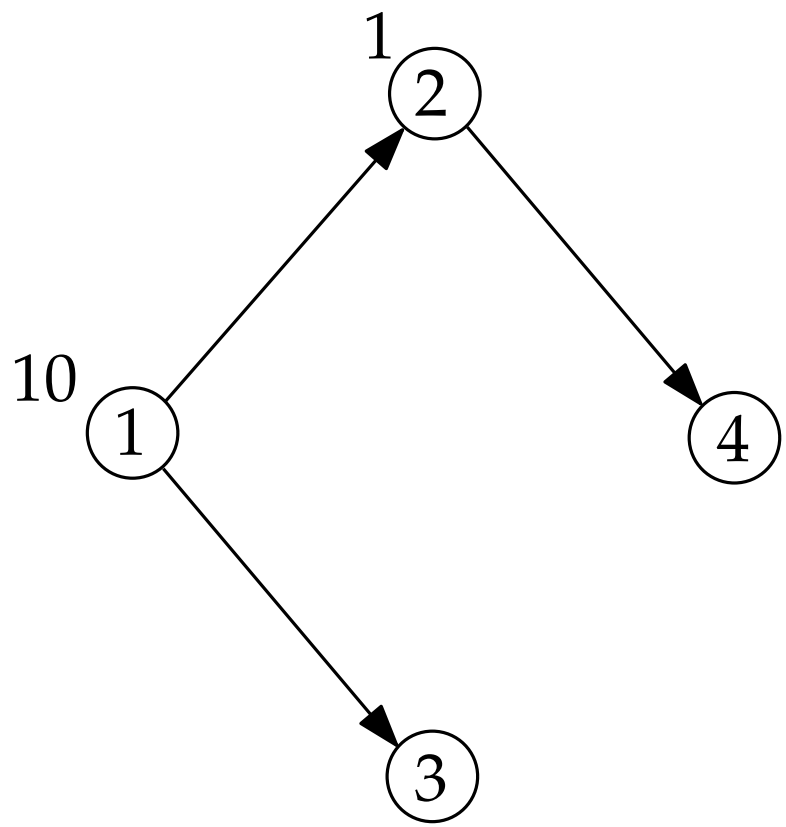
Problemstellung

Eingabe:

4

```
2 10
0 2
0 3
```

```
1 1
1 4 1
```



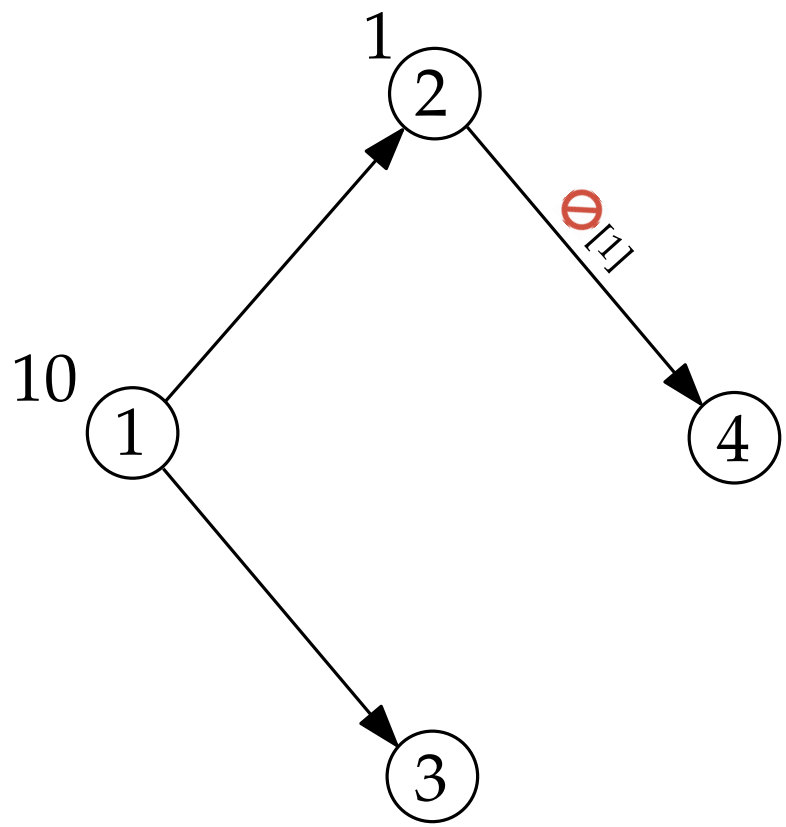
Problemstellung

Eingabe:

4

```
2 10
0 2
0 3
```

```
1 1
1 4 1
```



Problemstellung

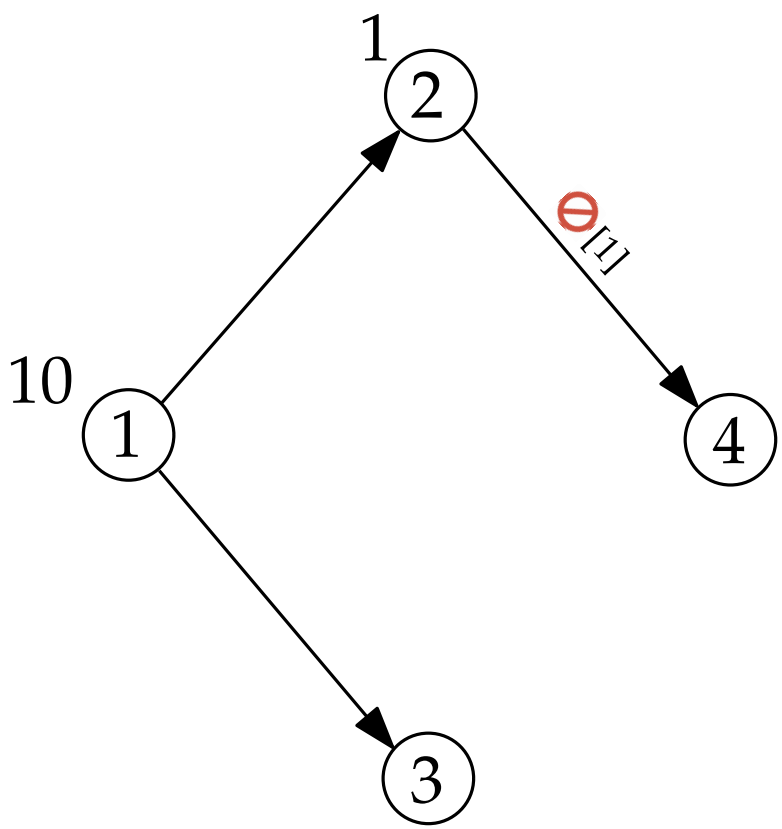
Eingabe:

4

```
2 10
0 2
0 3
```

```
1 1
1 4 1
```

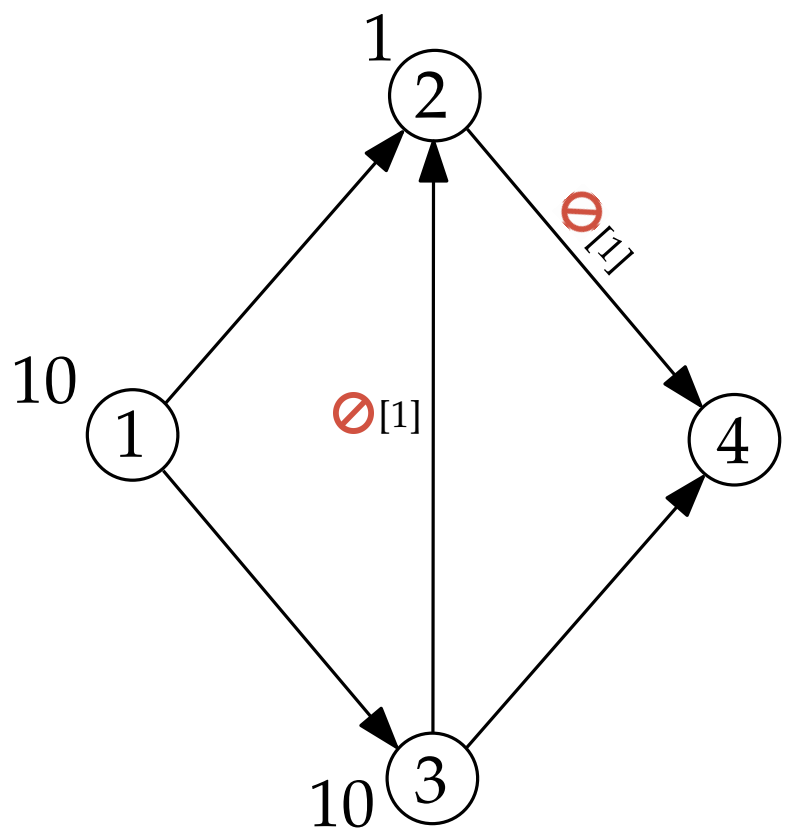
```
2 10
1 2 1
0 4
```



Problemstellung

Eingabe:

```
4
2 10
0 2
0 3
1 1
1 4 1
2 10
1 2 1
0 4
```



Problemstellung

Eingabe:

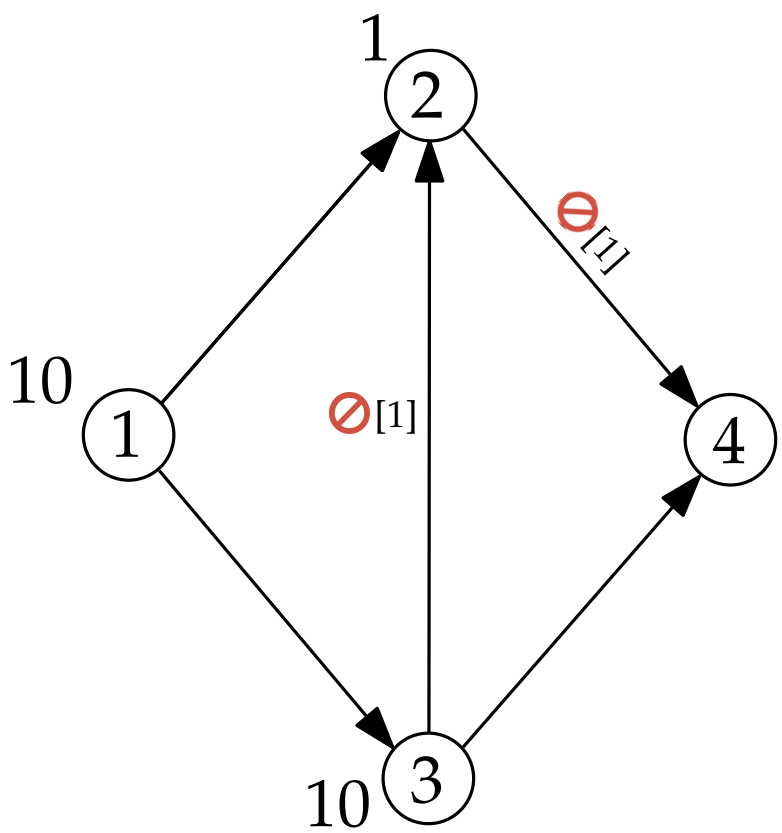
4

```
2 10
0 2
0 3
```

```
1 1
1 4 1
```

```
2 10
1 2 1
0 4
```

```
0 10
```



Problemstellung

Eingabe:

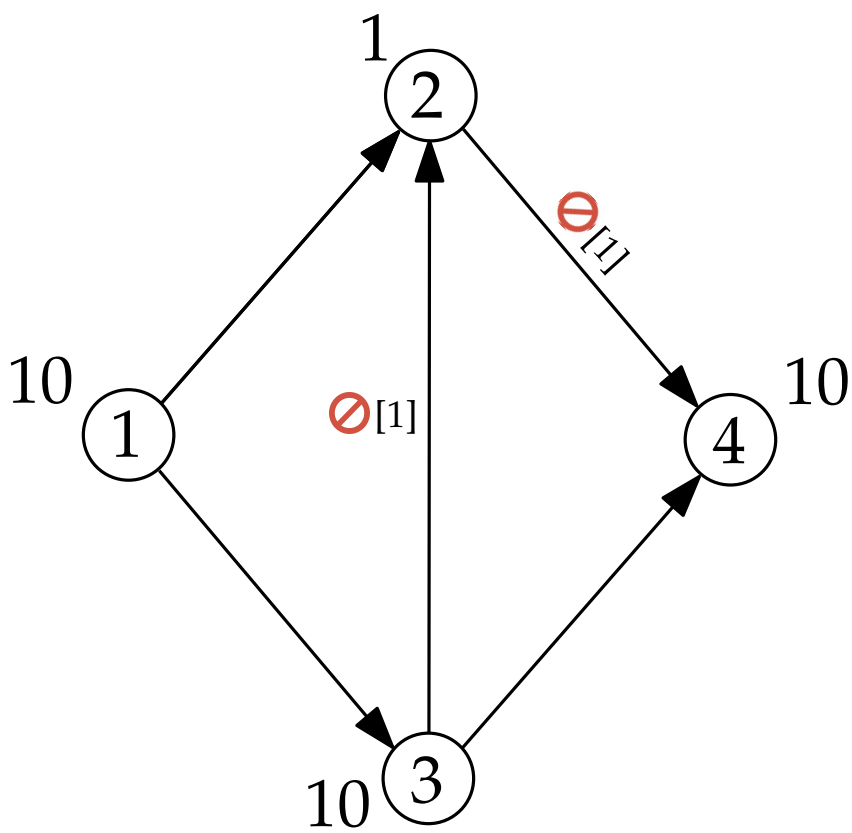
4

```
2 10
0 2
0 3
```

```
1 1
1 4 1
```

```
2 10
1 2 1
0 4
```

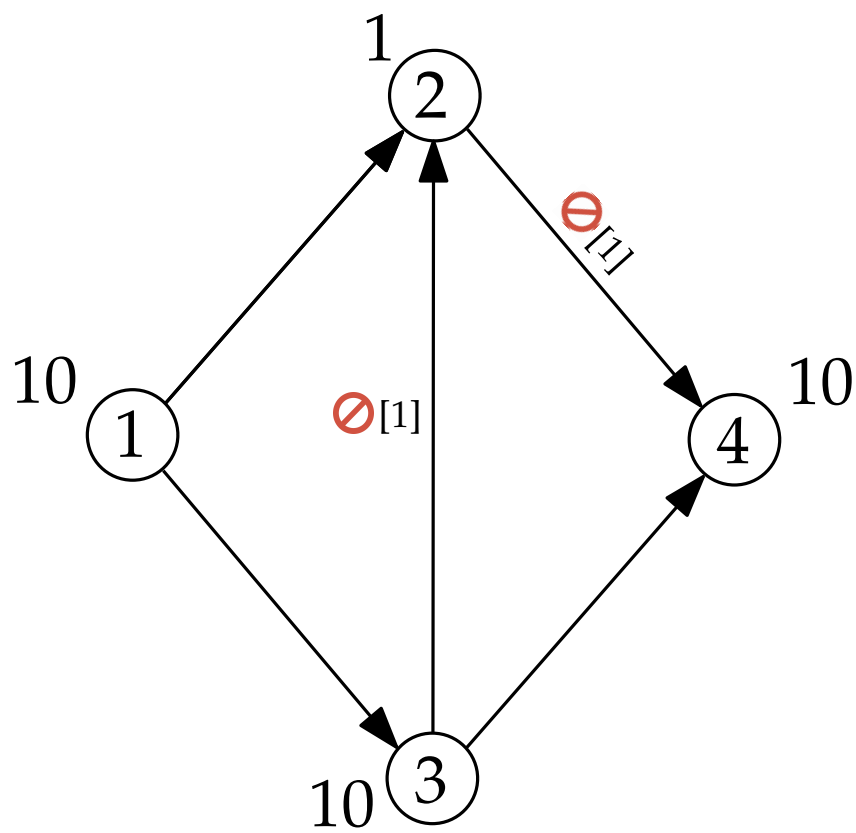
```
0 10
```



Beispiel

Eingabe:

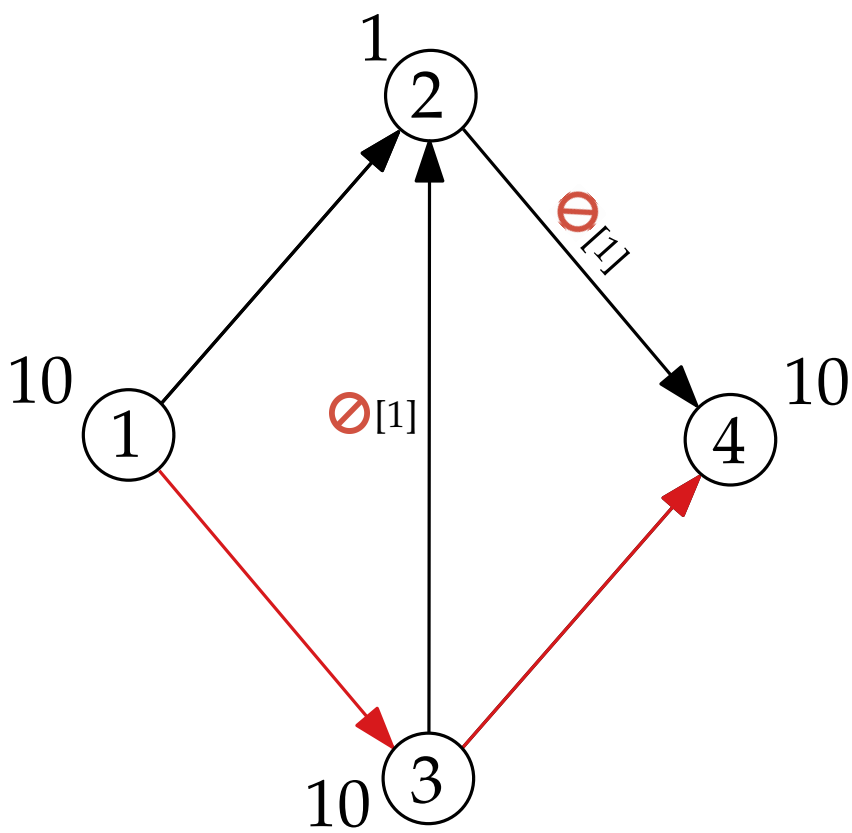
```
4
2 10
0 2
0 3
1 1
1 4 1
2 10
1 2 1
0 4
0 10
```



Beispiel

Eingabe:

```
4
2 10
0 2
0 3
1 1
1 4 1
2 10
1 2 1
0 4
0 10
```



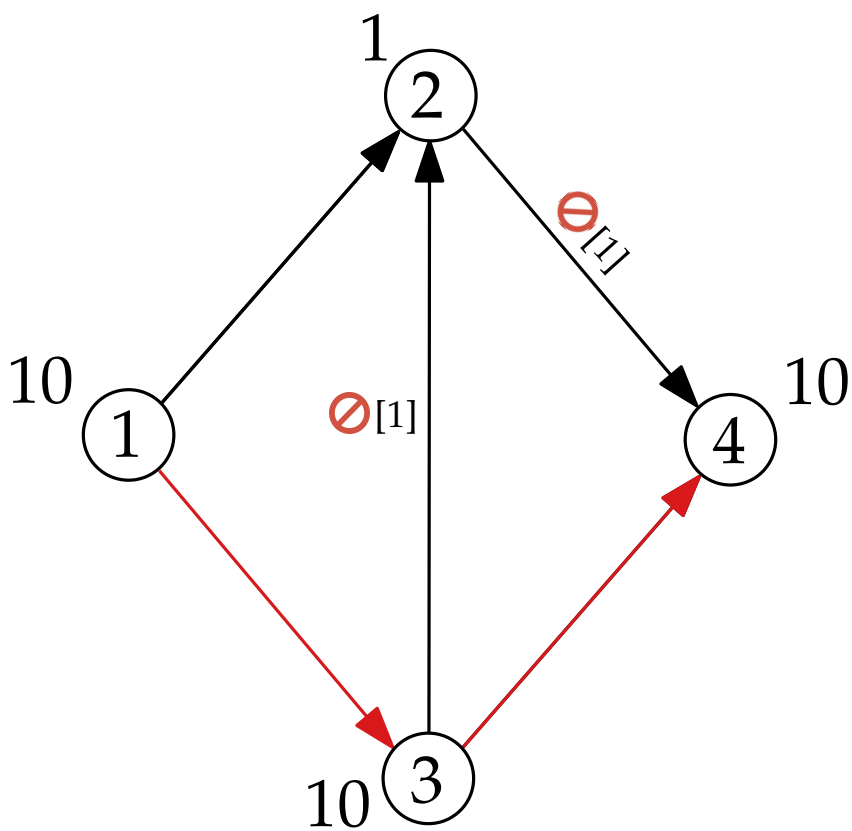
Beispiel

Eingabe:

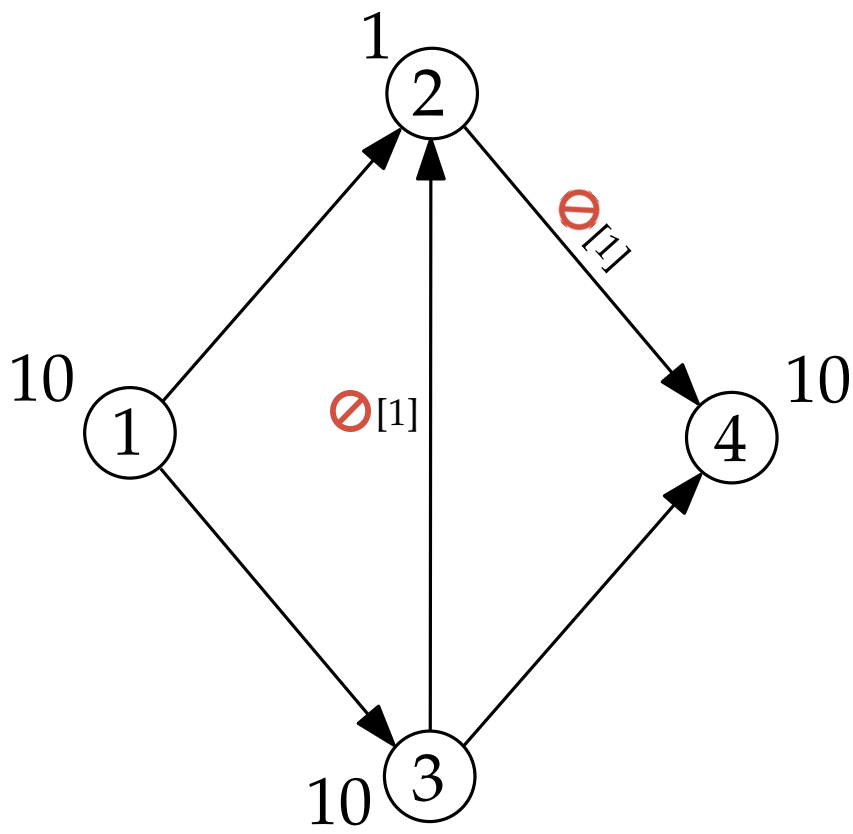
```
4
2 10
0 2
0 3
1 1
1 4 1
2 10
1 2 1
0 4
0 10
```

Ausgabe:

30

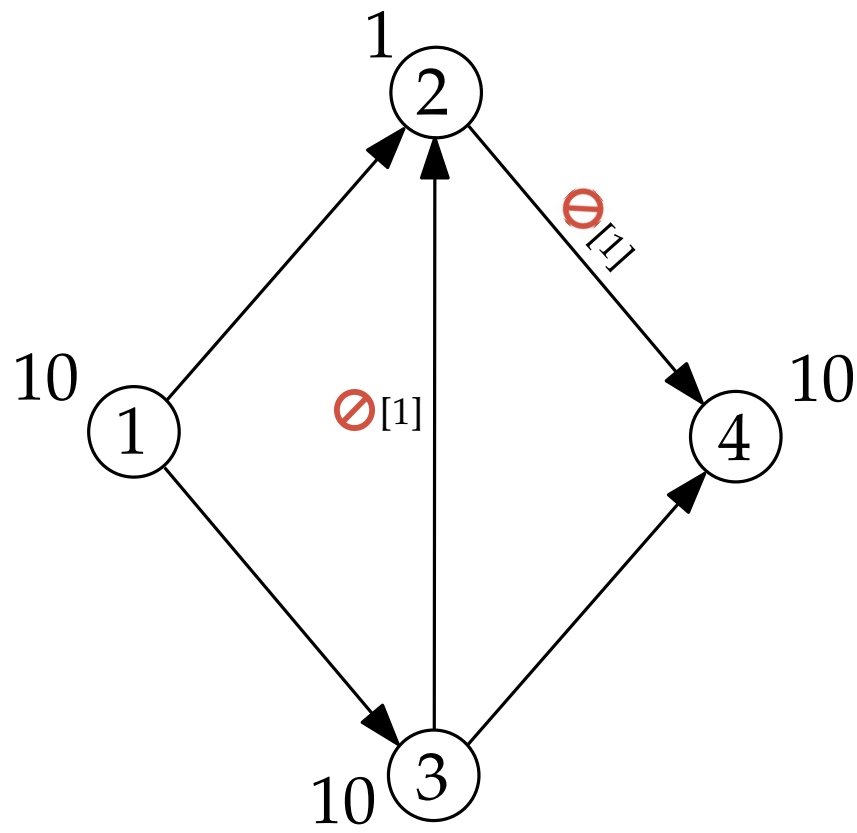


Brute-Force?



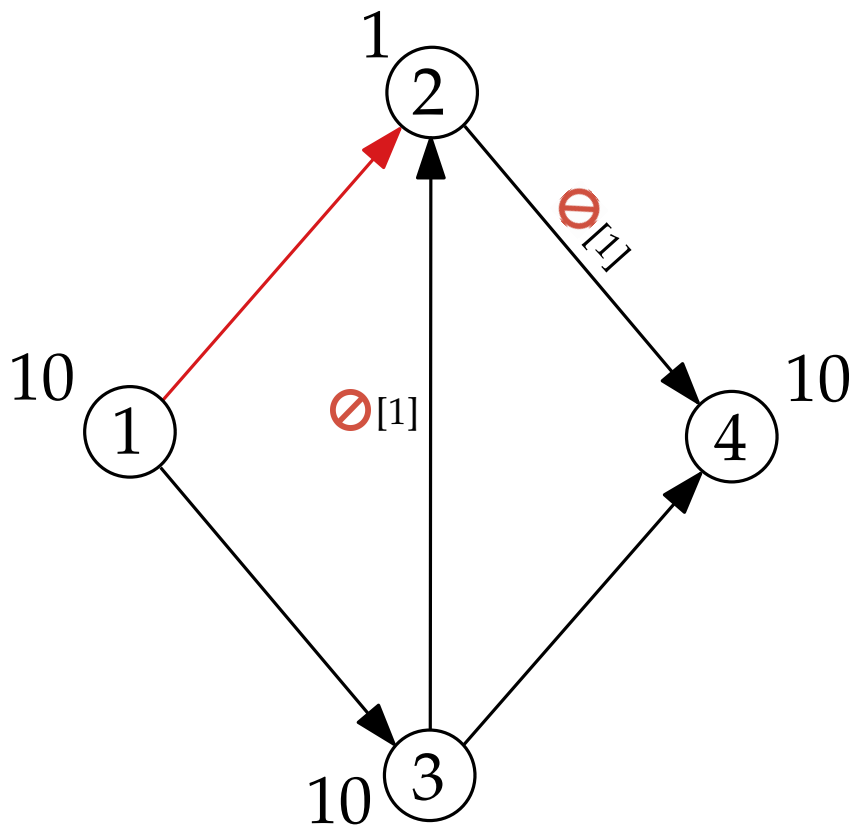
Brute-Force?

Rekursives Durchlaufen aller Pfade unter Berücksichtigung der Regeln



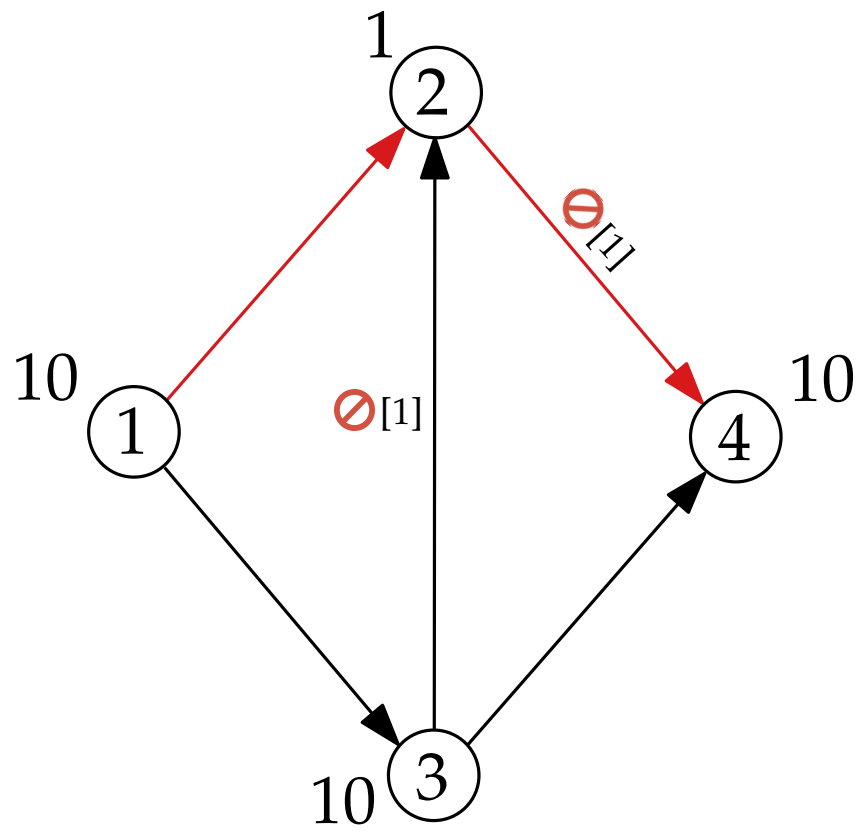
Brute-Force?

Rekursives Durchlaufen aller Pfade unter Berücksichtigung der Regeln



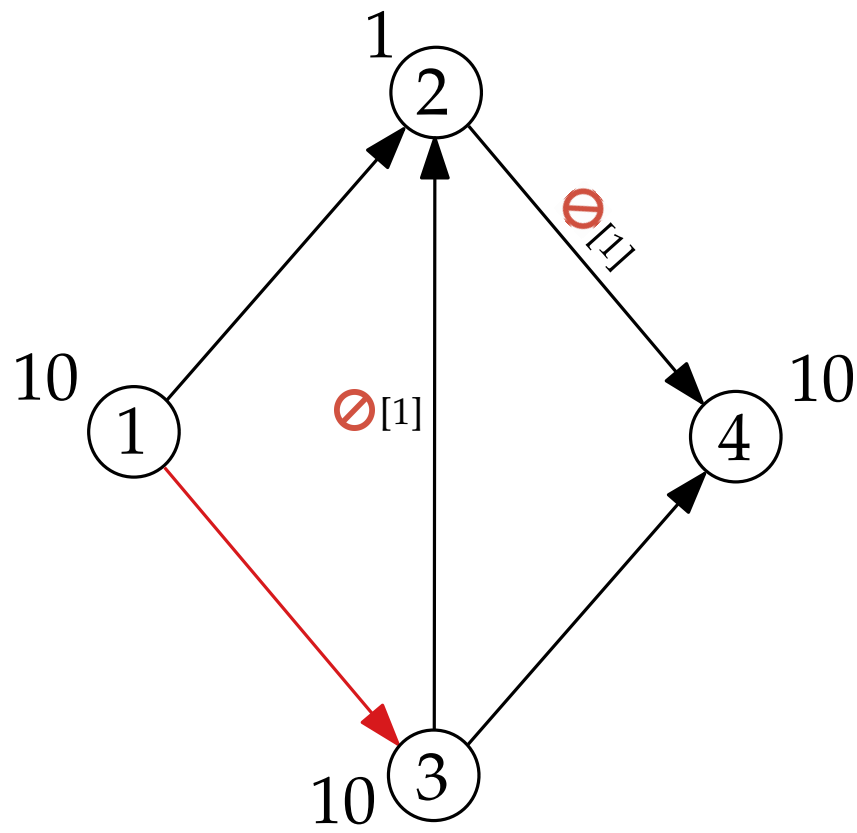
Brute-Force?

Rekursives Durchlaufen aller Pfade unter Berücksichtigung der Regeln



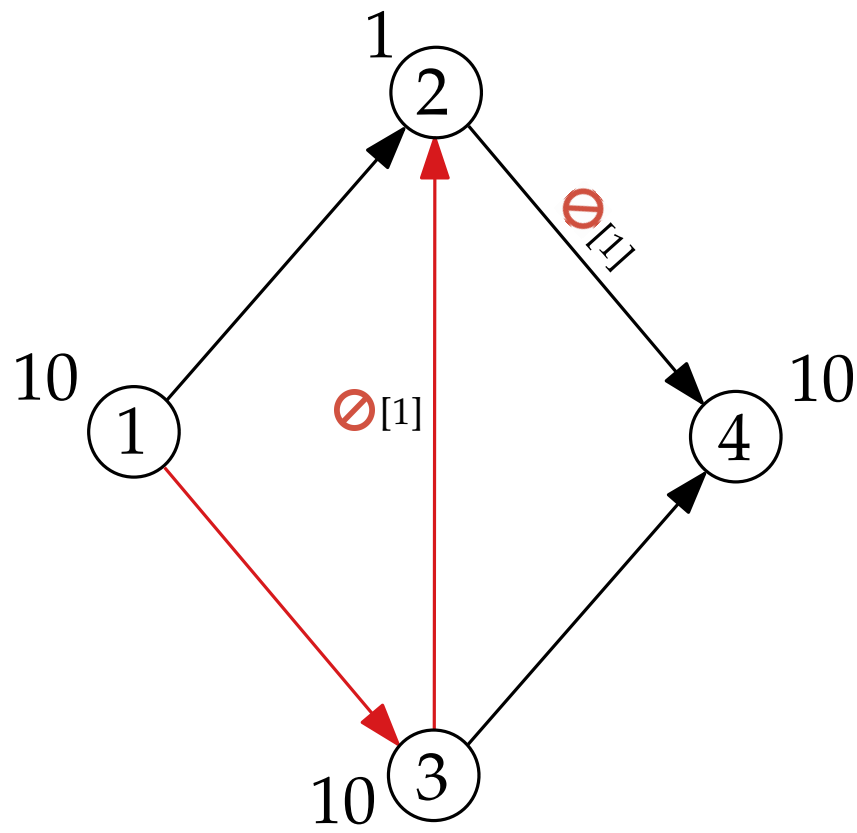
Brute-Force?

Rekursives Durchlaufen aller Pfade unter Berücksichtigung der Regeln



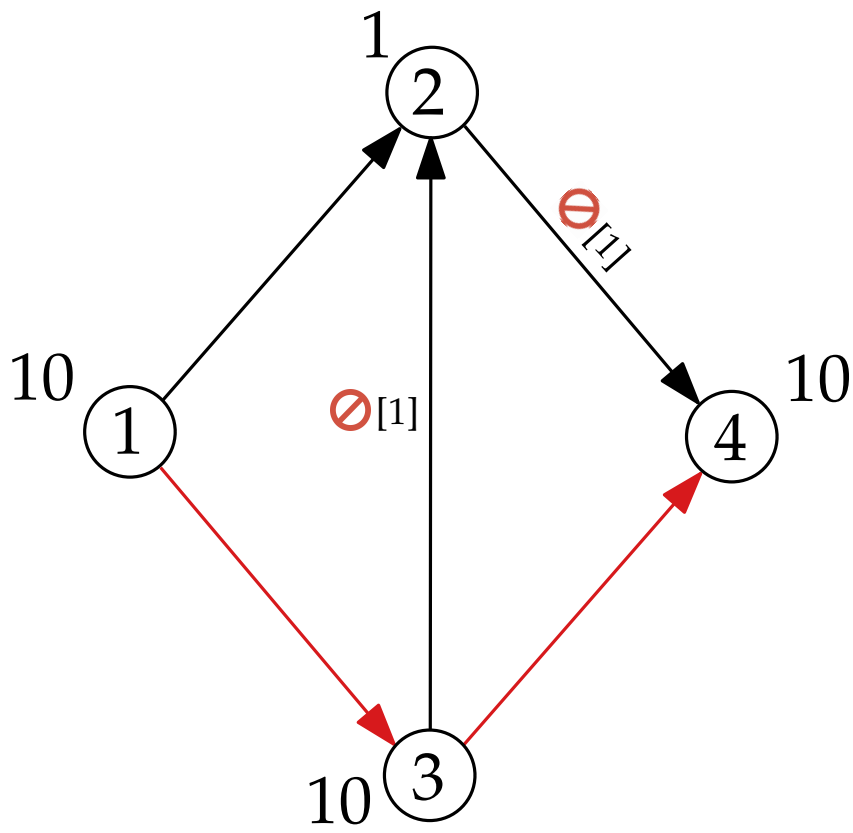
Brute-Force?

Rekursives Durchlaufen aller Pfade unter Berücksichtigung der Regeln



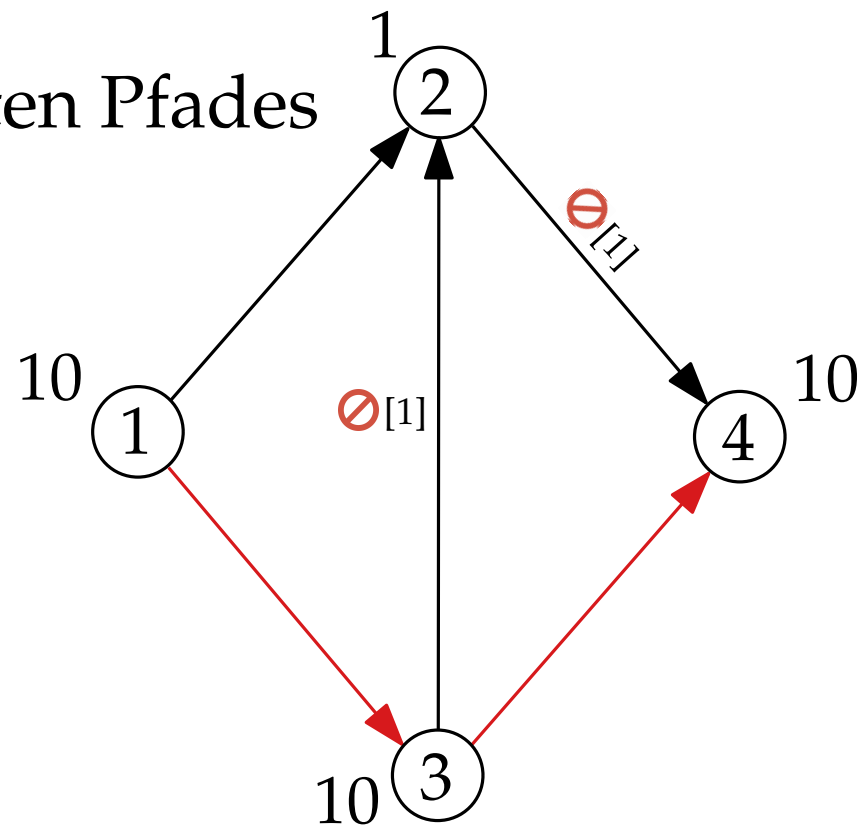
Brute-Force?

Rekursives Durchlaufen aller Pfade unter Berücksichtigung der Regeln



Brute-Force?

Rekursives Durchlaufen aller Pfade unter Berücksichtigung der Regeln
→ Zurückgeben des kürzesten Pfades

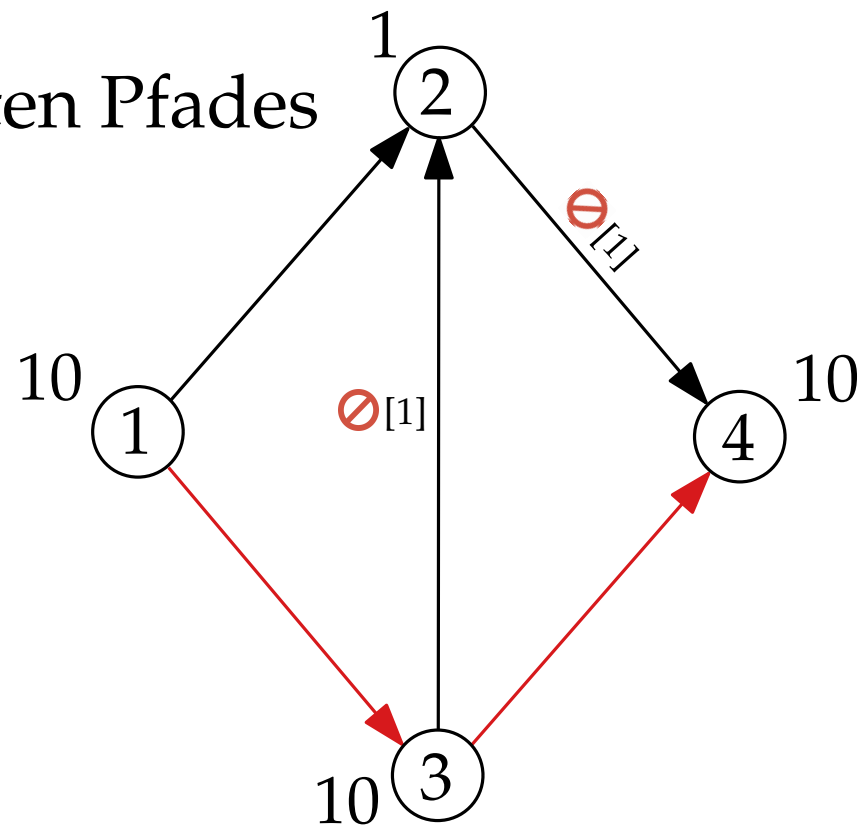


Brute-Force?

Rekursives Durchlaufen aller Pfade unter Berücksichtigung der Regeln

→ Zurückgeben des kürzesten Pfades

→ Laufzeit: $O(n!) = O(n^n)$

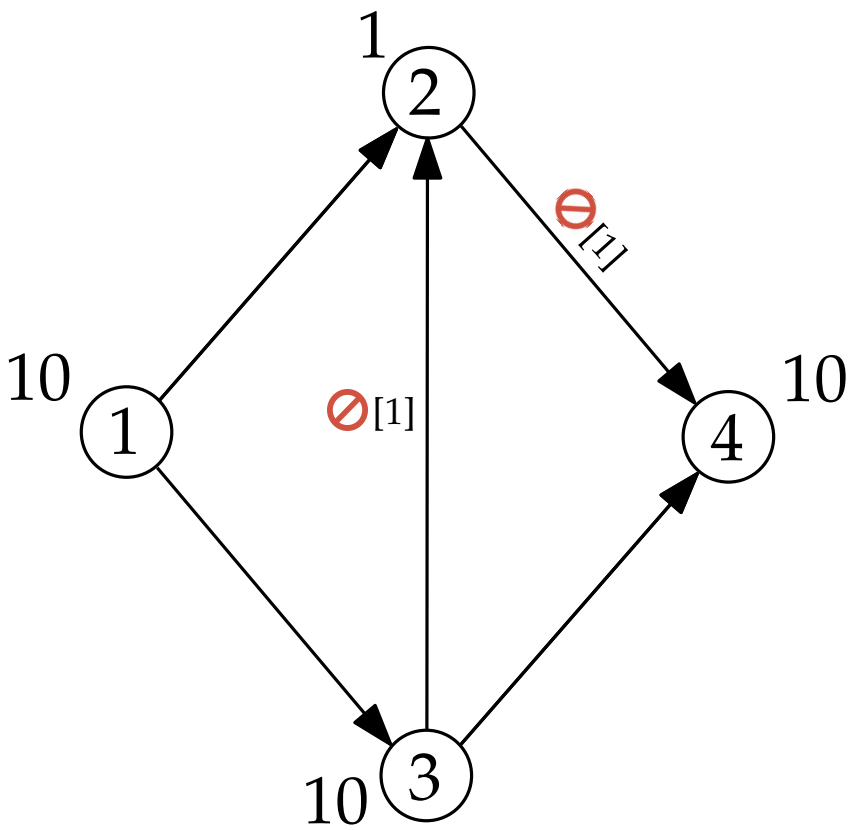


Ansatz: Kantengraph

Konstruieren des Kantengraphen G' aus dem gegebenen Graphen G.

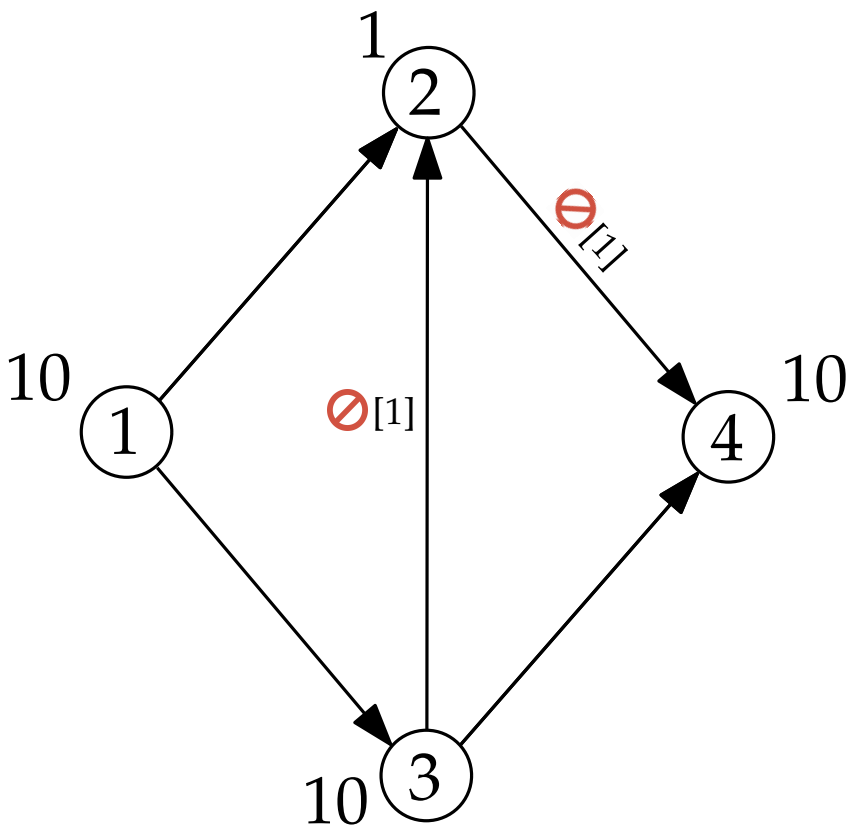
Ansatz: Kantengraph

Konstruieren des Kantengraphen G' aus dem gegebenen Graphen G.



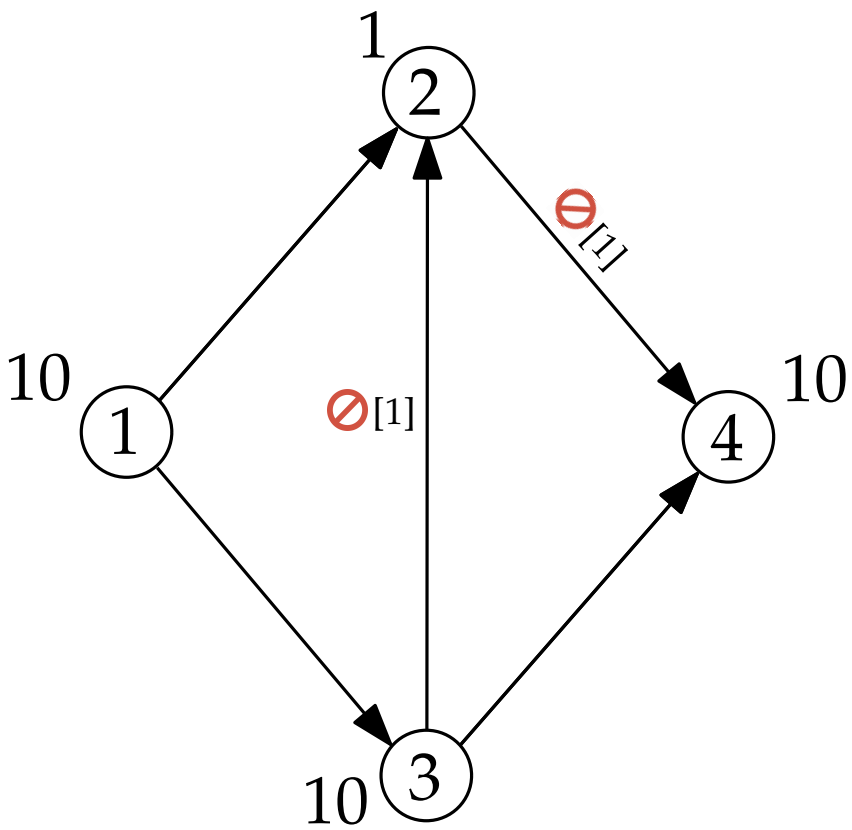
Ansatz: Kantengraph

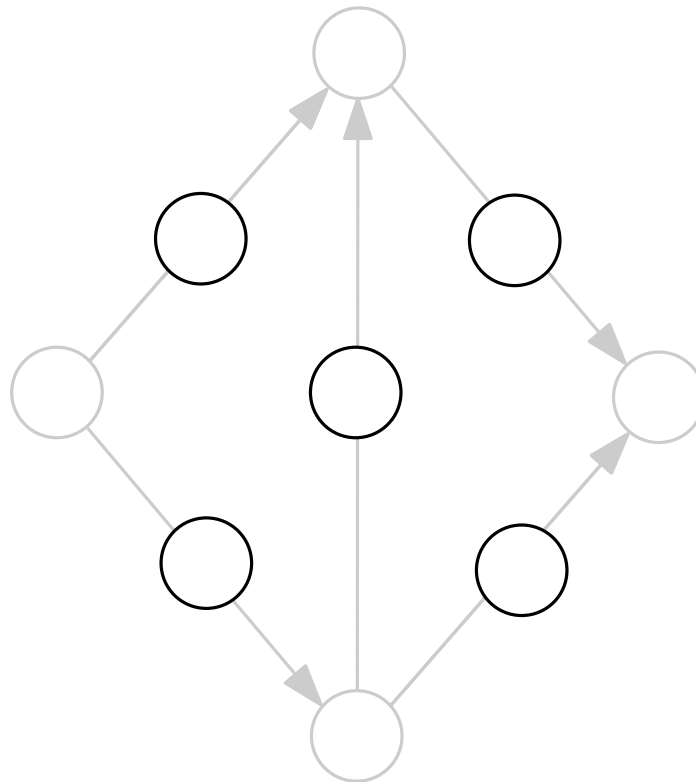
Konstruieren des Kantengraphen G' aus dem gegebenen Graphen G.



Ansatz: Kantengraph

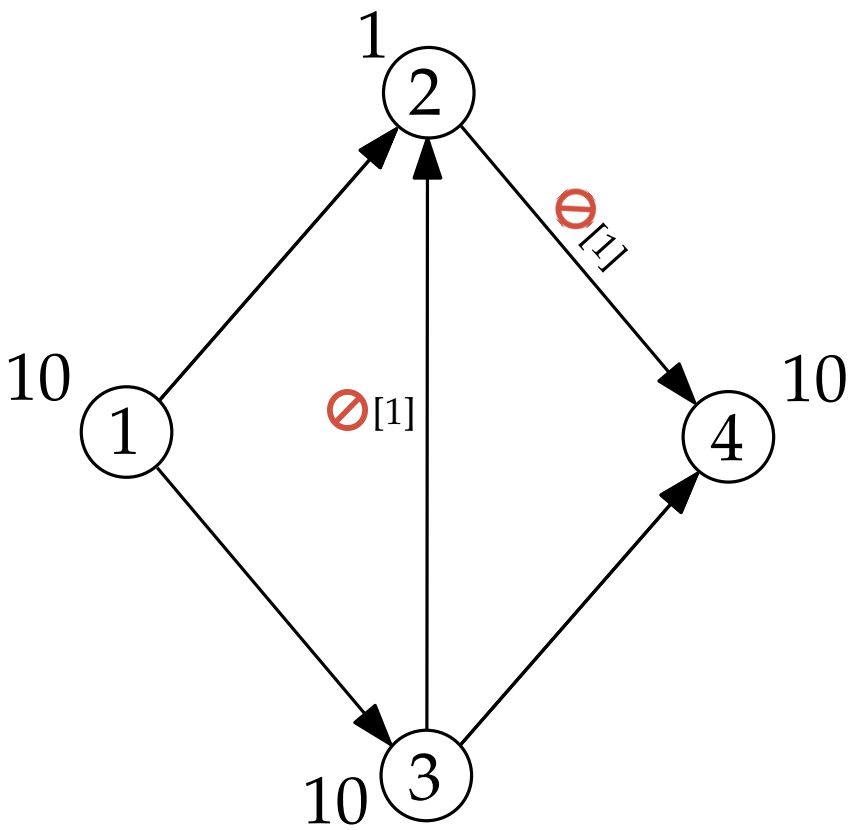
Konstruieren des Kantengraphen G' aus dem gegebenen Graphen G.

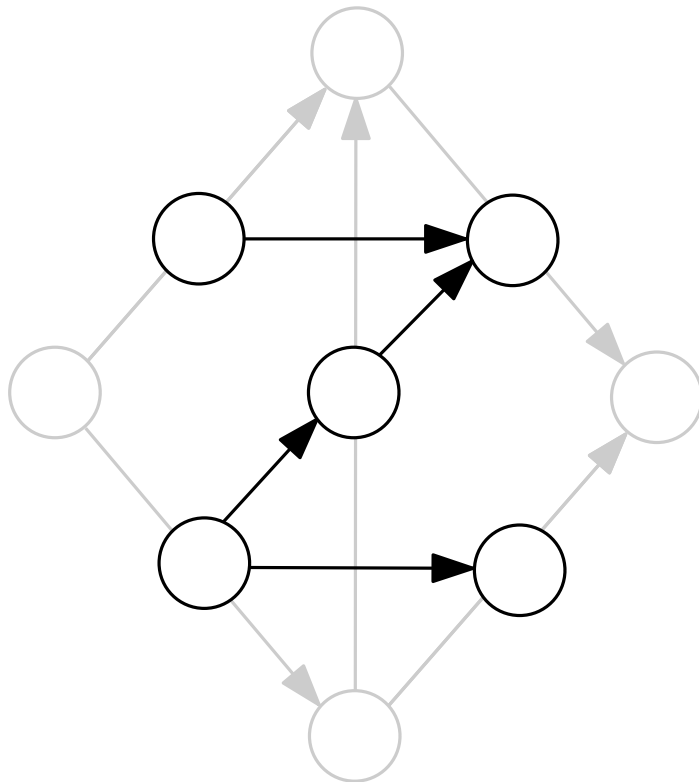




Ansatz: Kantengraph

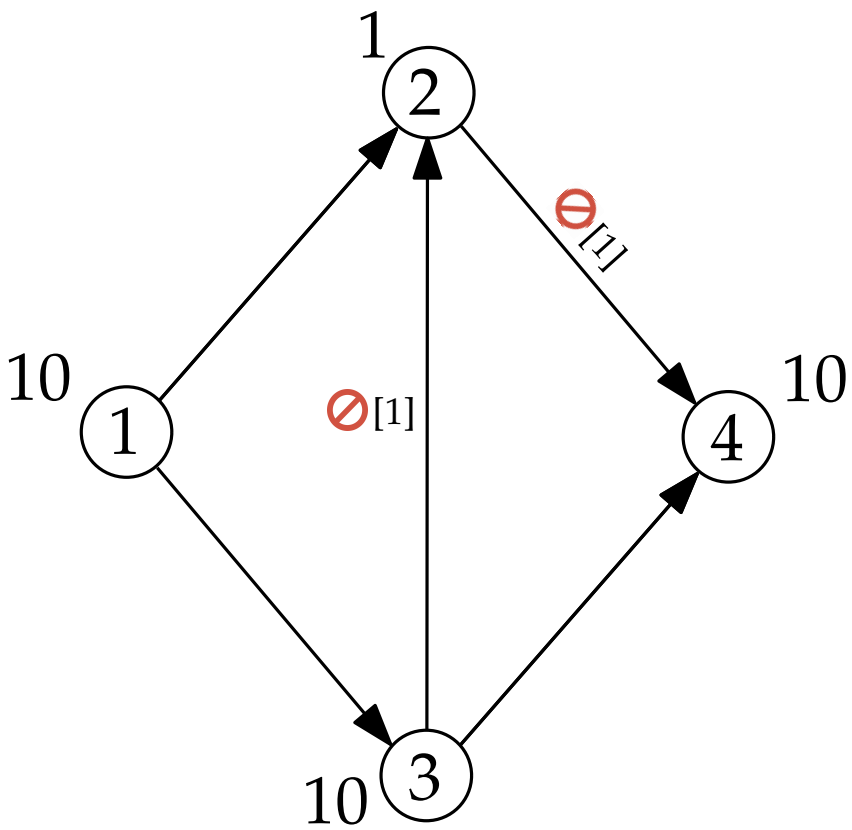
Konstruieren des Kantengraphen G' aus dem gegebenen Graphen G.

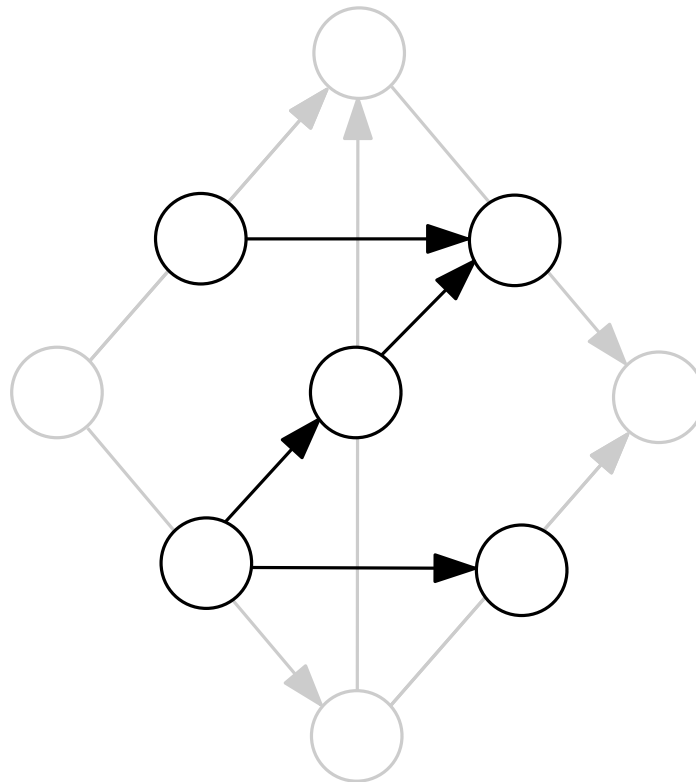




Ansatz: Kantengraph

Konstruieren des Kantengraphen G' aus dem gegebenen Graphen G.

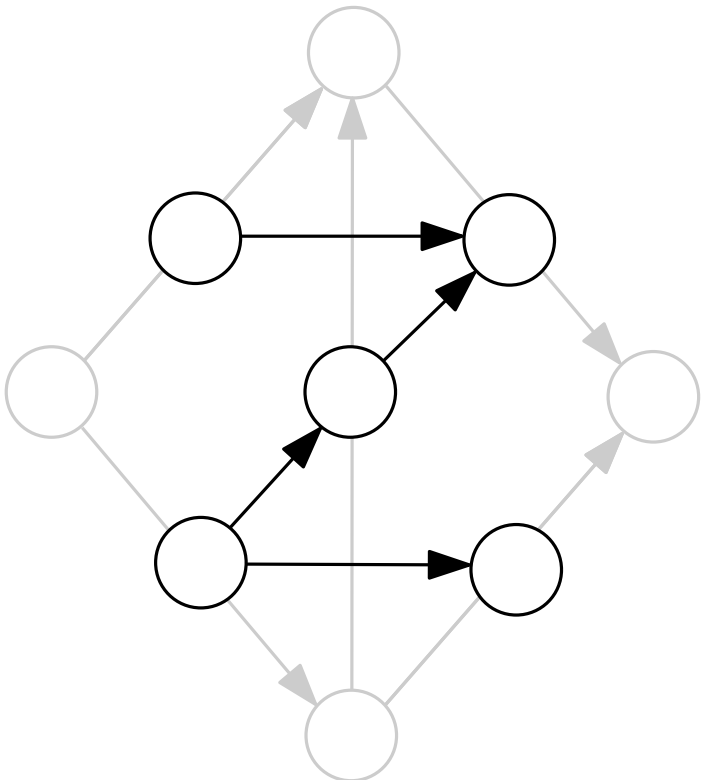




G' muss noch weiter modifiziert werden...

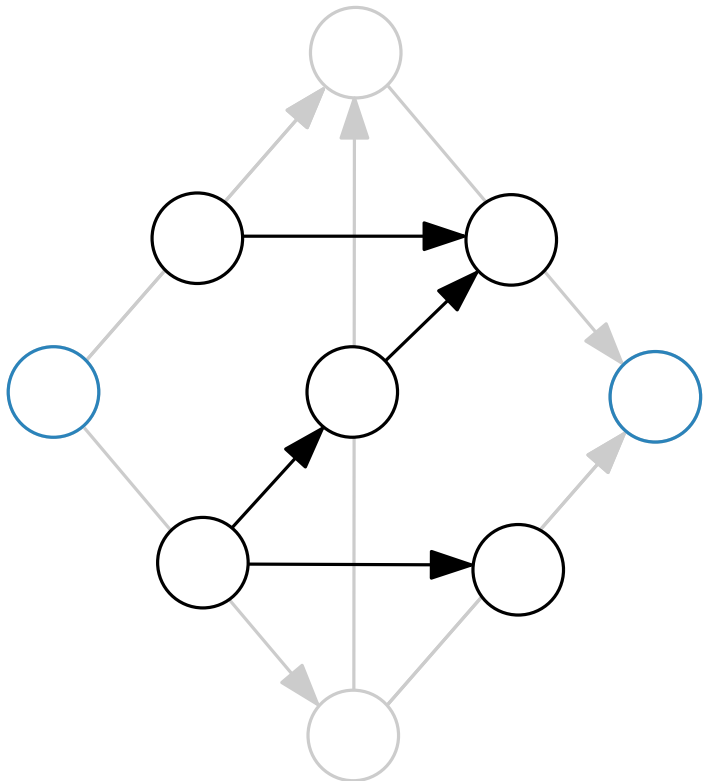
Ansatz: Kantengraph

Modifikation 1: Wiederherstellung von Start- und Endknoten



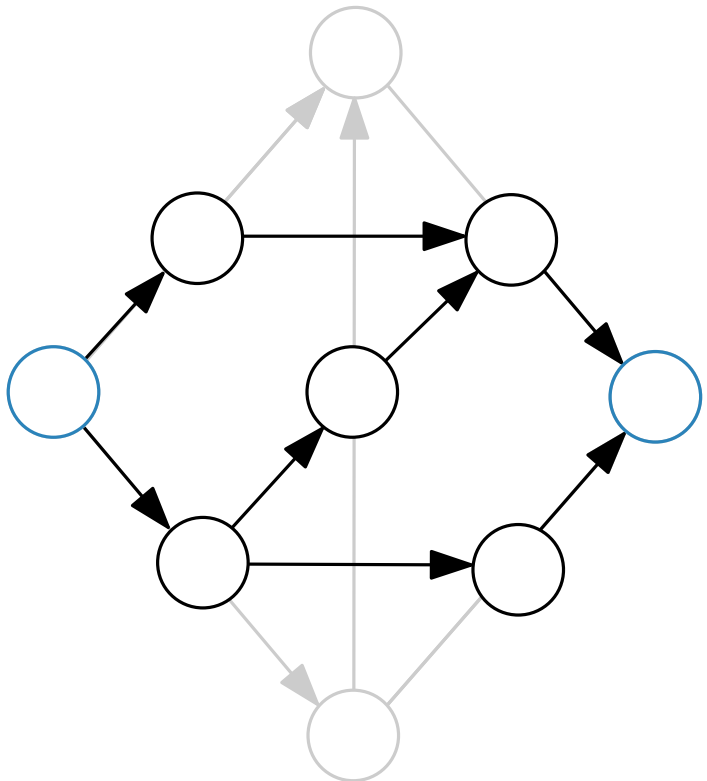
Ansatz: Kantengraph

Modifikation 1: Wiederherstellung von Start- und Endknoten



Ansatz: Kantengraph

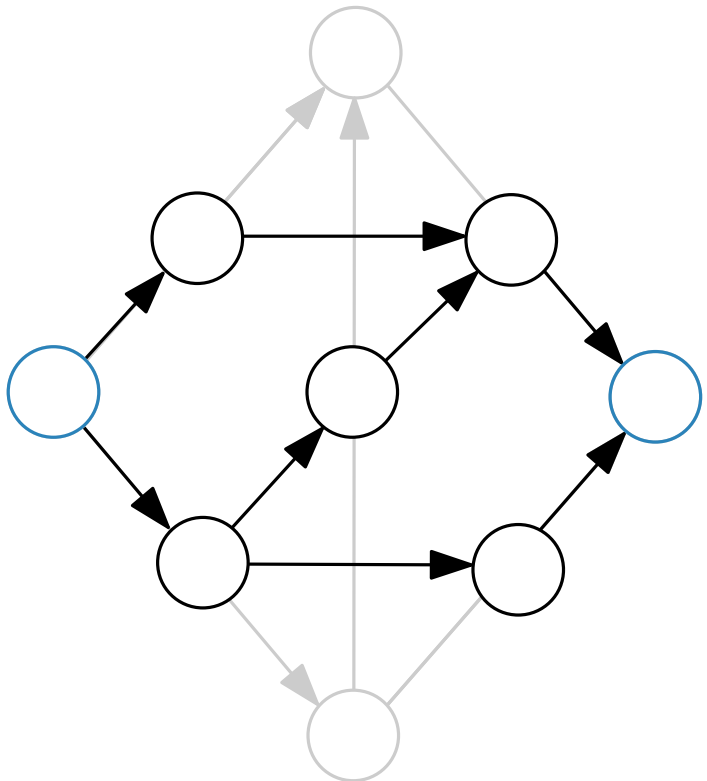
Modifikation 1: Wiederherstellung von Start- und Endknoten



Ansatz: Kantengraph

Modifikation 1: Wiederherstellung von Start- und Endknoten

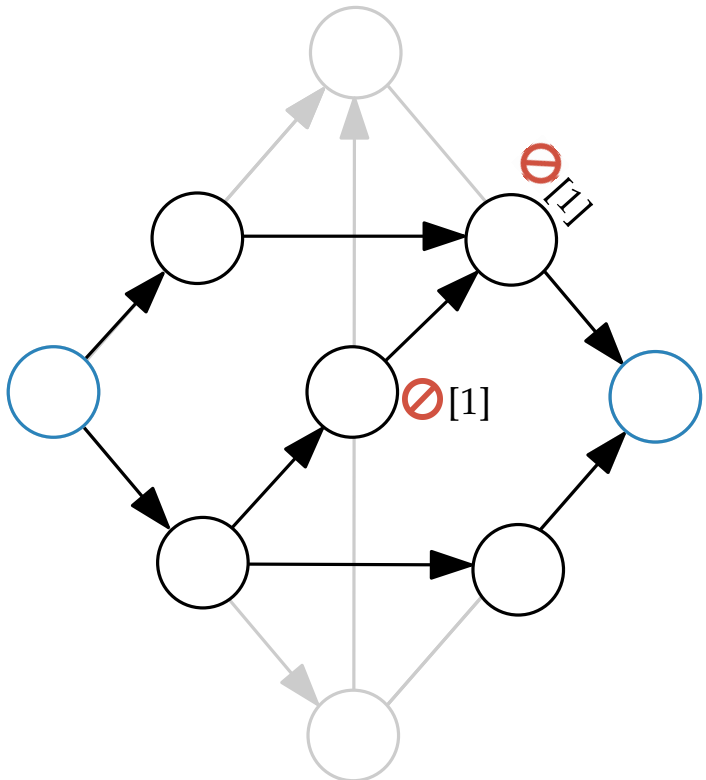
Modifikation 2: Entfernen der durch Regeln verbotenen Kanten



Ansatz: Kantengraph

Modifikation 1: Wiederherstellung von Start- und Endknoten

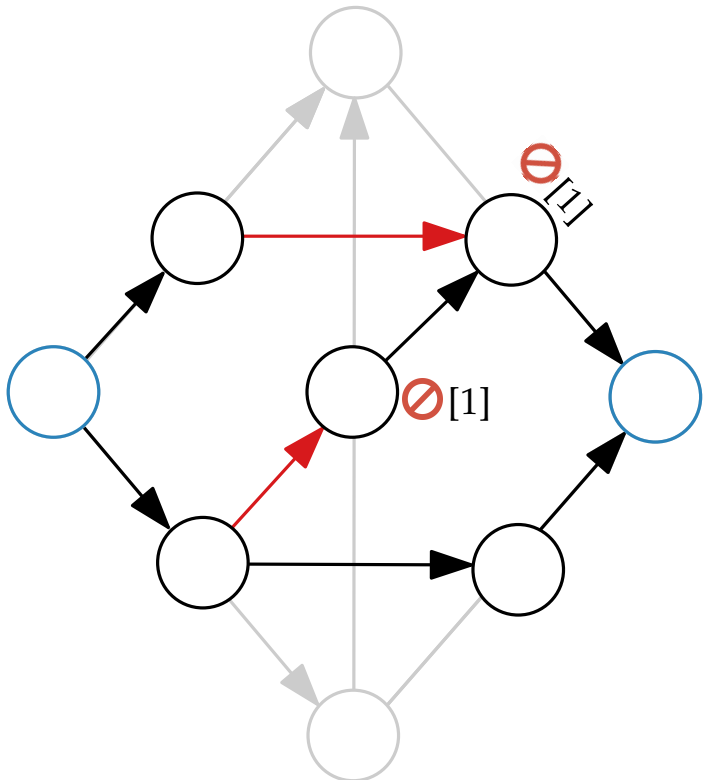
Modifikation 2: Entfernen der durch Regeln verbotenen Kanten



Ansatz: Kantengraph

Modifikation 1: Wiederherstellung von Start- und Endknoten

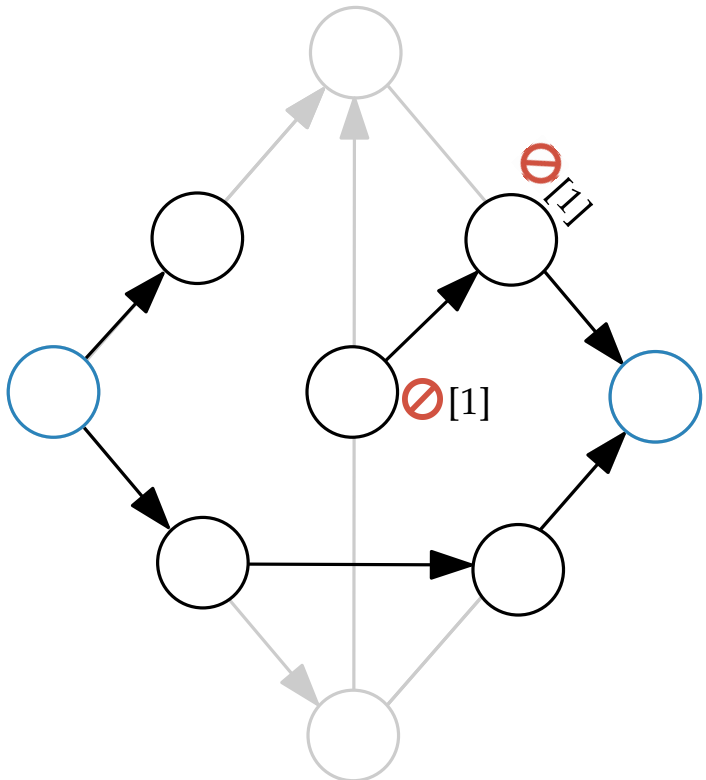
Modifikation 2: Entfernen der durch Regeln verbotenen Kanten



Ansatz: Kantengraph

Modifikation 1: Wiederherstellung von Start- und Endknoten

Modifikation 2: Entfernen der durch Regeln verbotenen Kanten

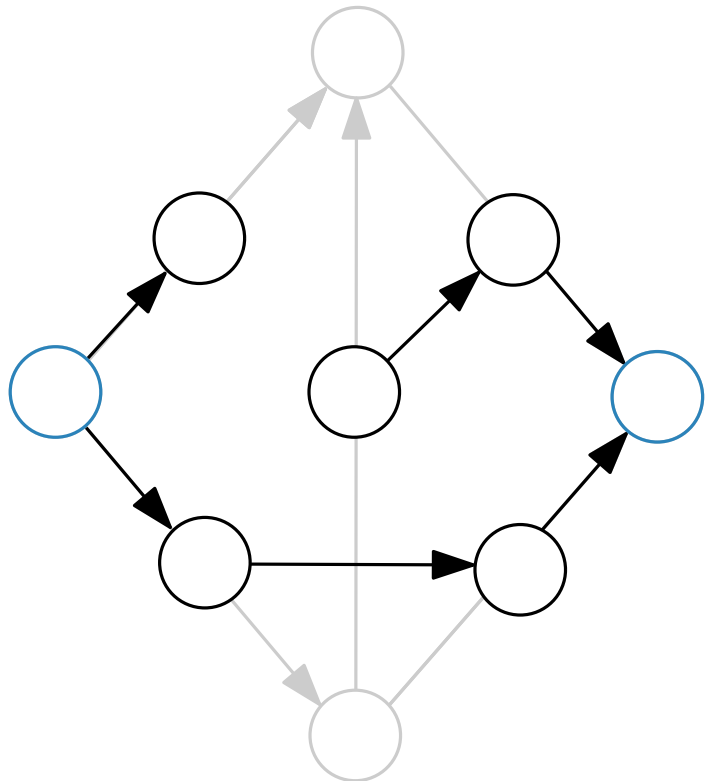


Ansatz: Kantengraph

Modifikation 1: Wiederherstellung von Start- und Endknoten

Modifikation 2: Entfernen der durch Regeln verbotenen Kanten

Modifikation 3: Setzen der Kantengewichte entsprechend ihrer ehemaligen Knoten

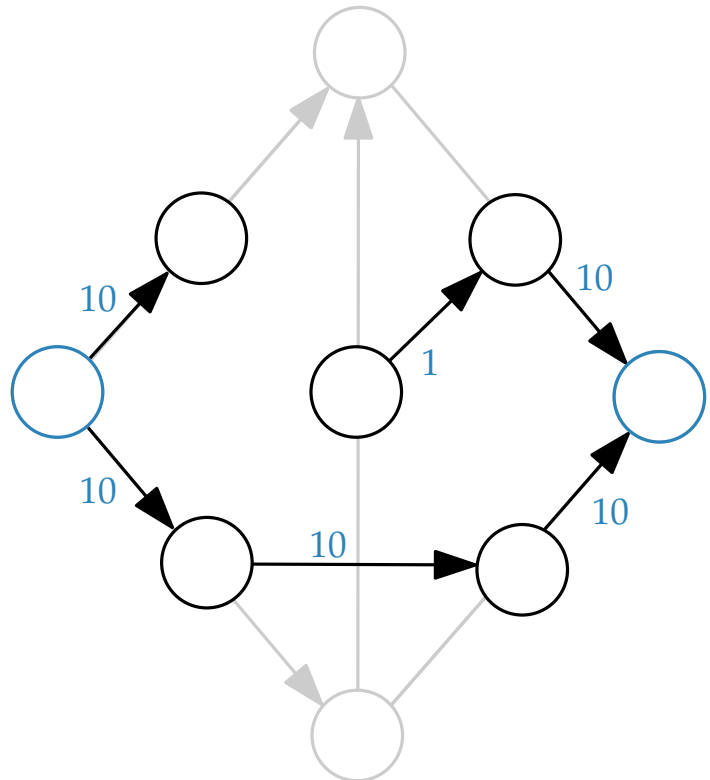


Ansatz: Kantengraph

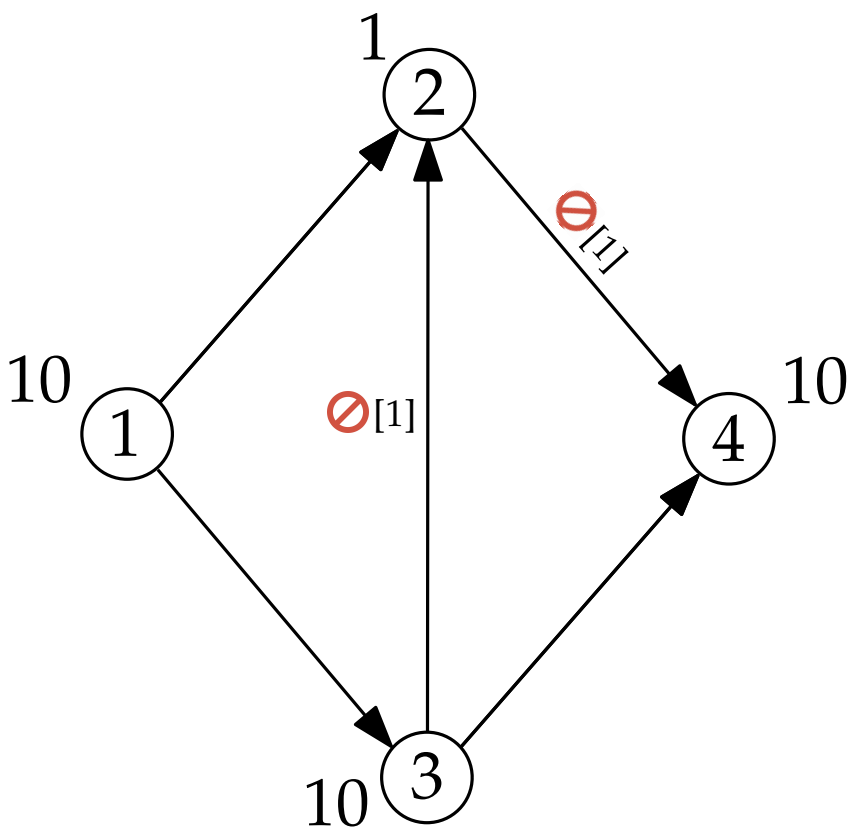
Modifikation 1: Wiederherstellung von Start- und Endknoten

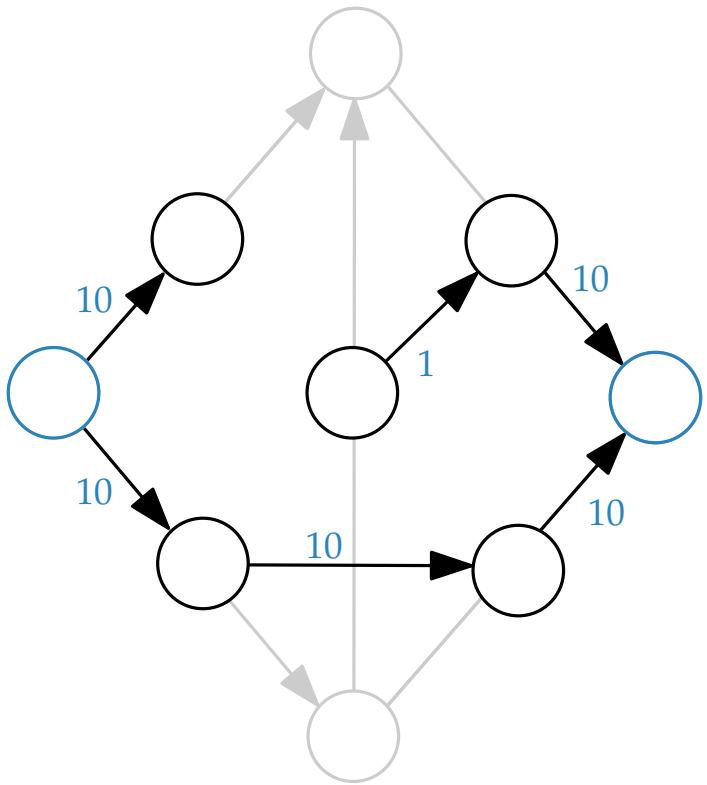
Modifikation 2: Entfernen der durch Regeln verbotenen Kanten

Modifikation 3: Setzen der Kantengewichte entsprechend ihrer ehemaligen Knoten

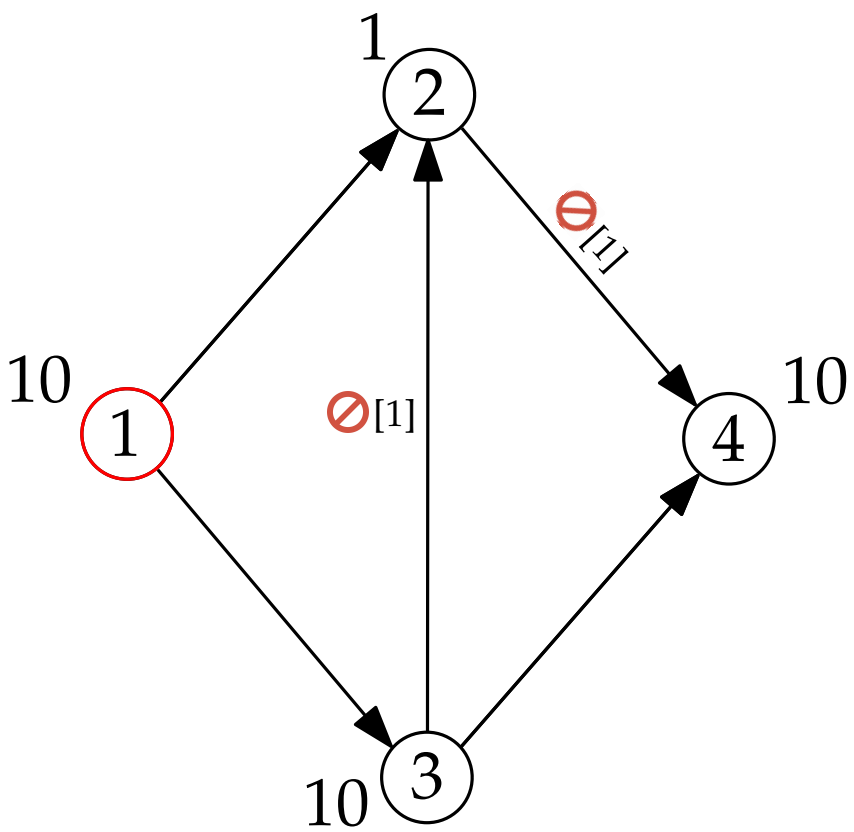


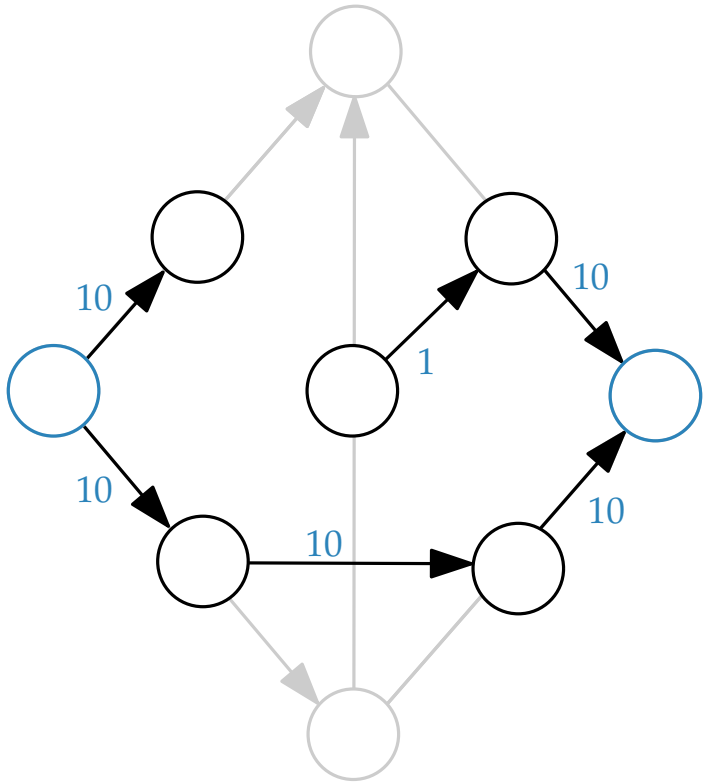
Ansatz: Kantengraph



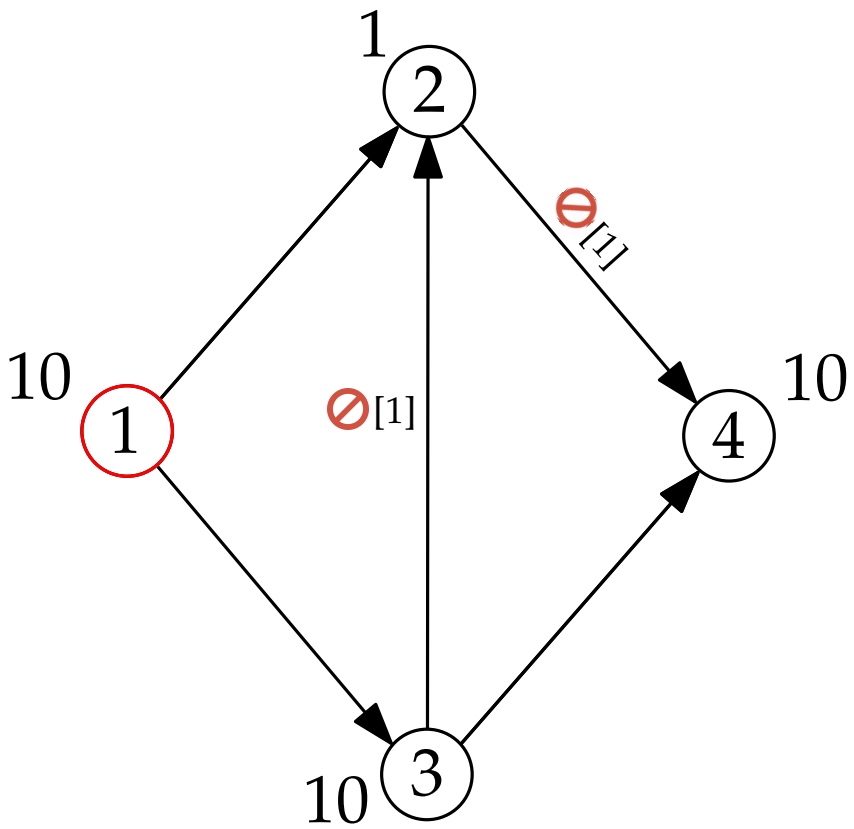


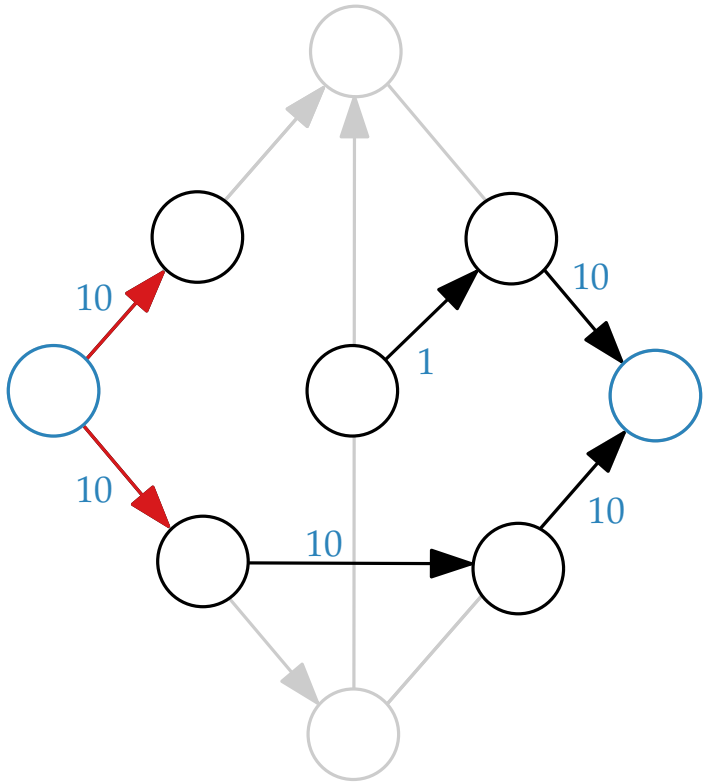
Ansatz: Kantengraph



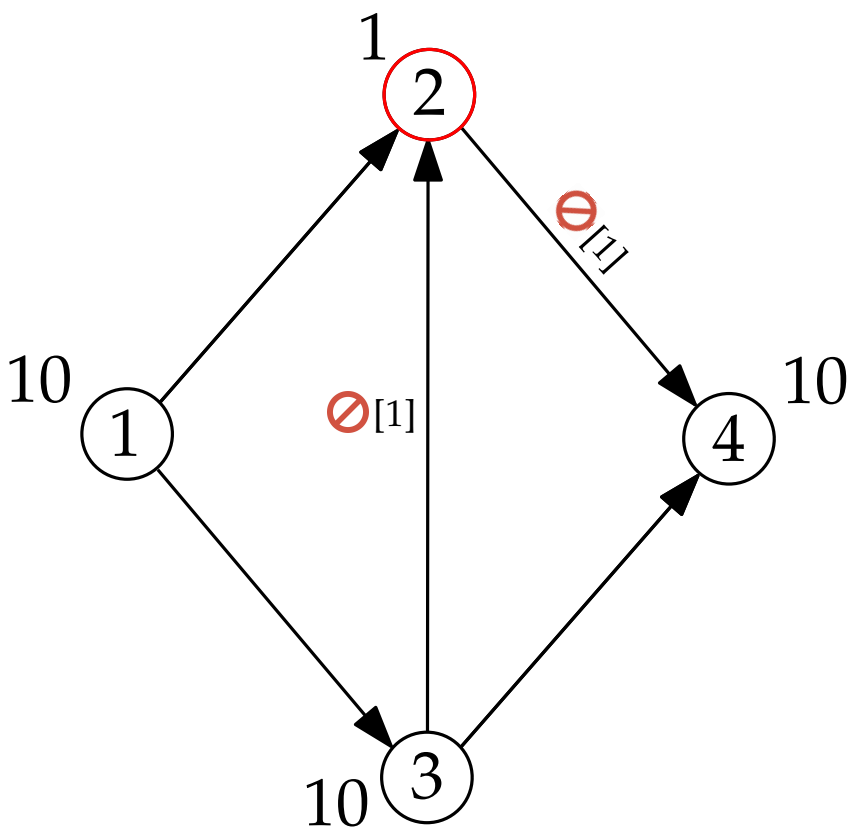


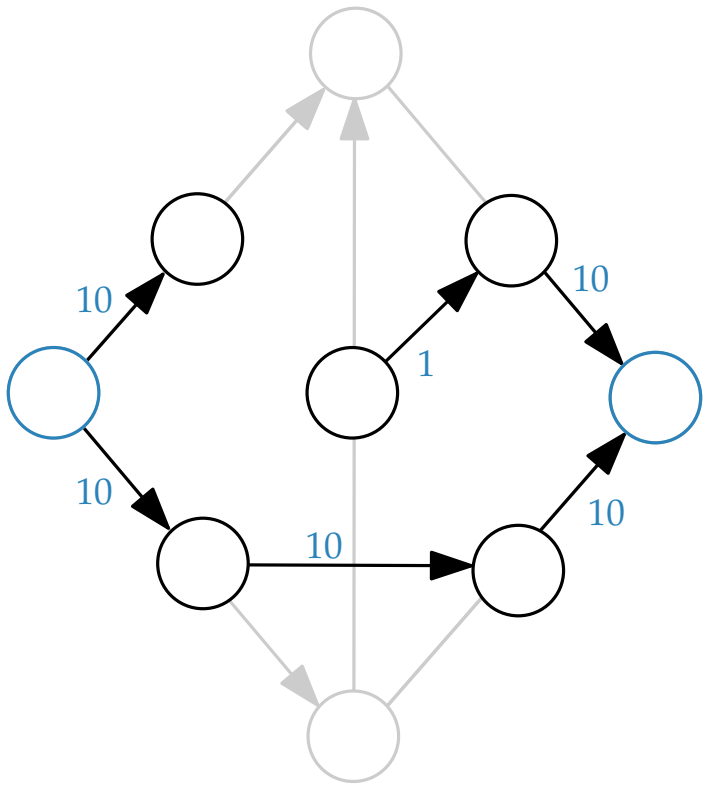
Ansatz: Kantengraph



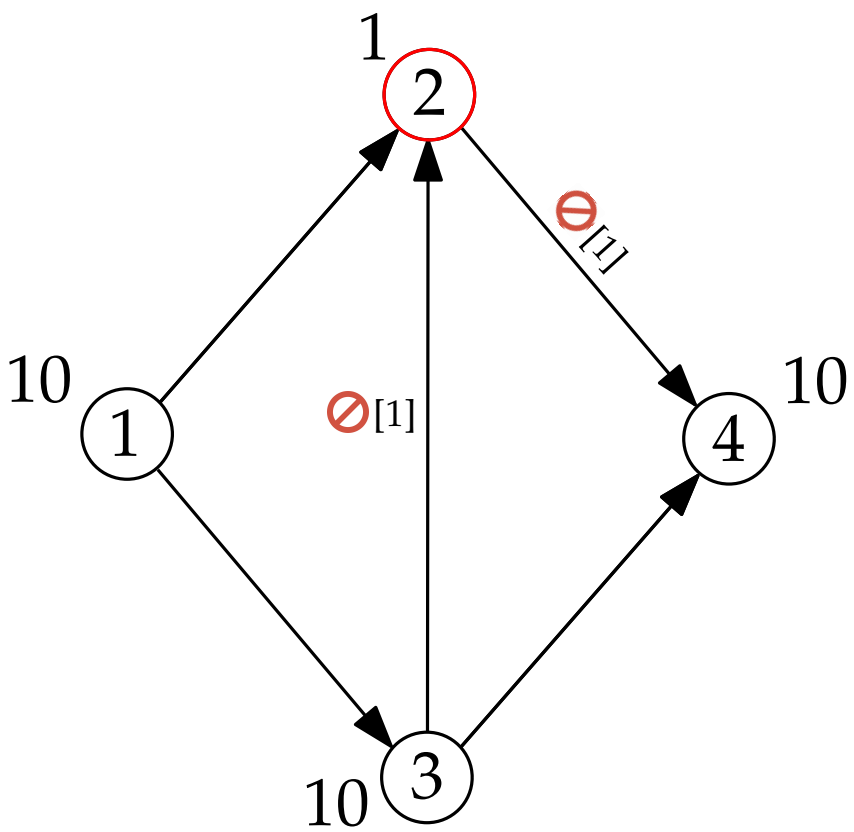


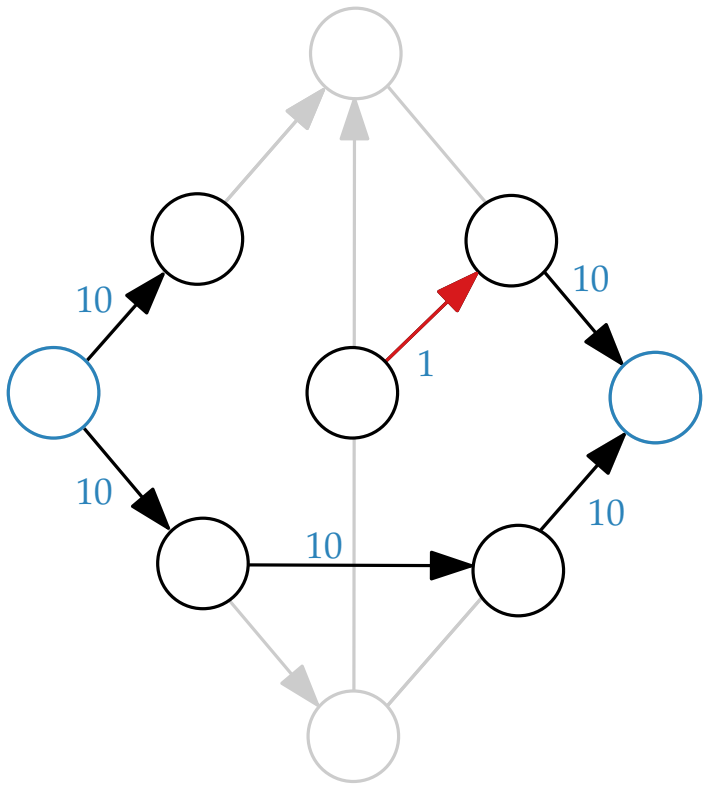
Ansatz: Kantengraph



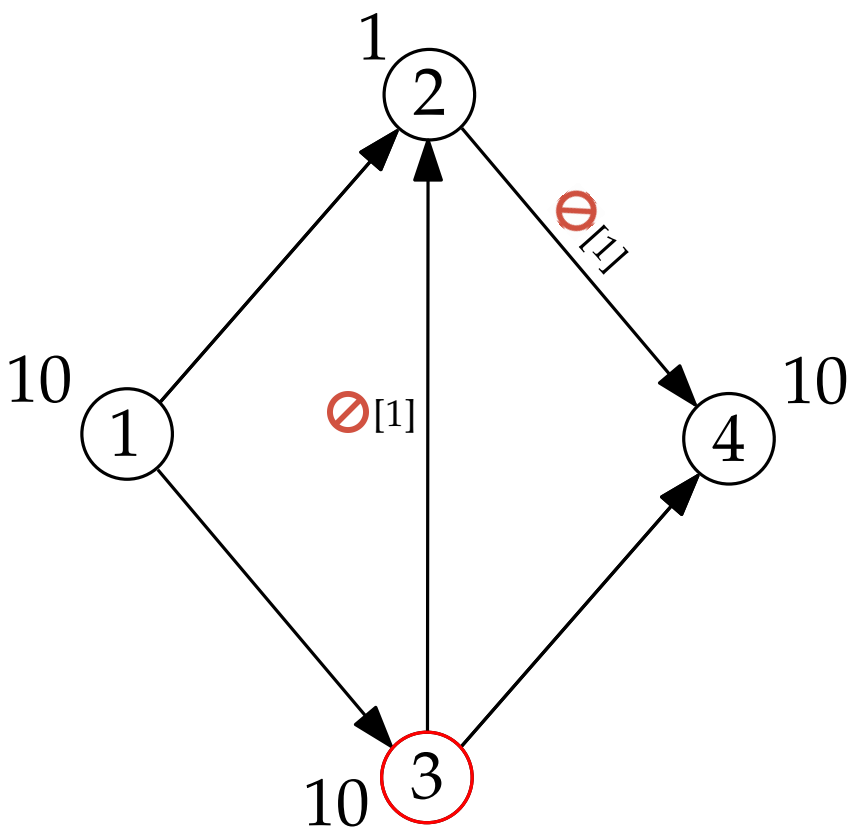


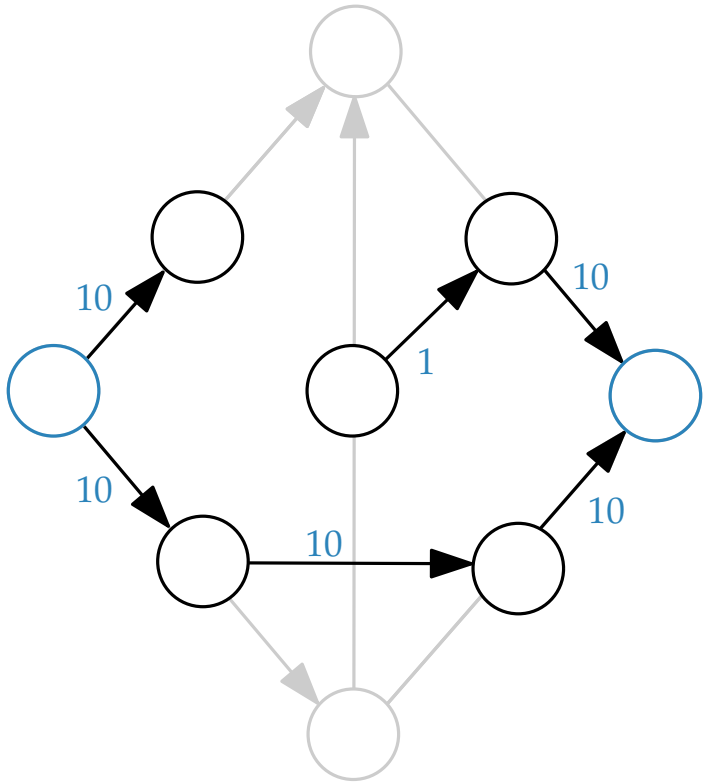
Ansatz: Kantengraph



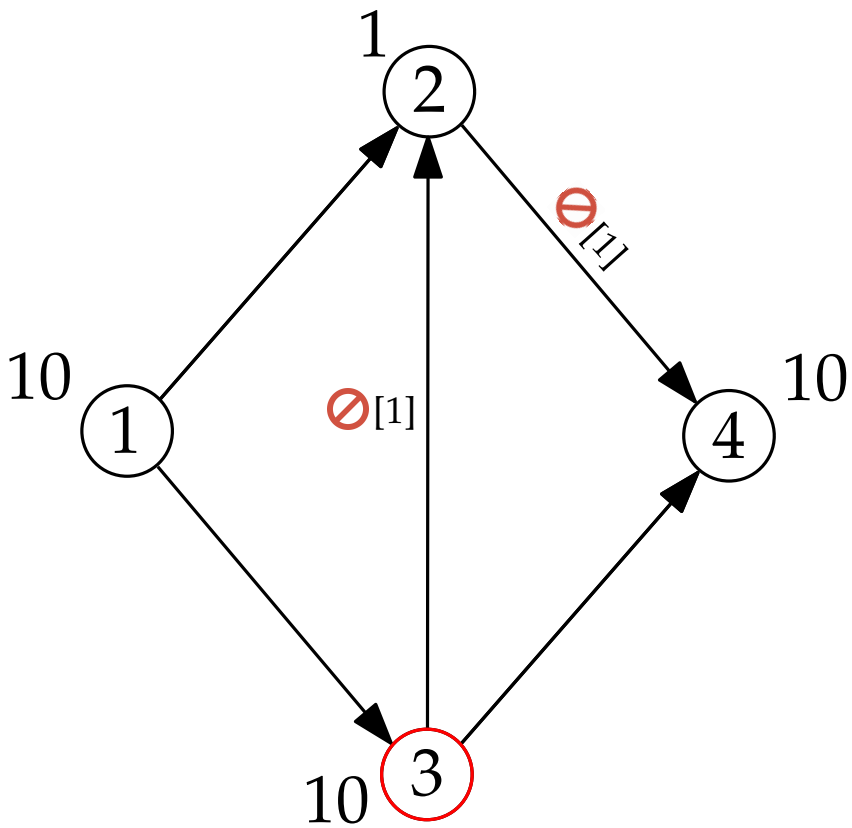


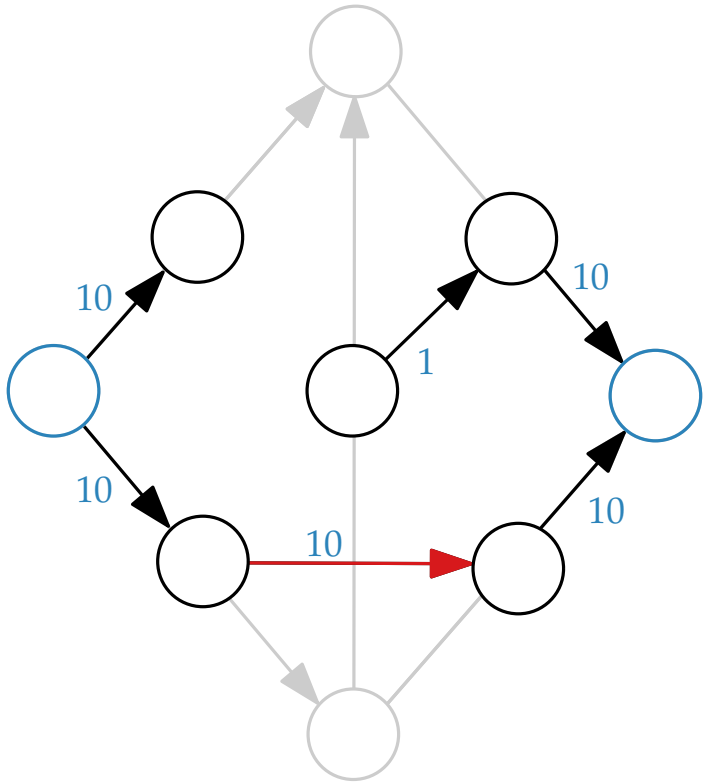
Ansatz: Kantengraph



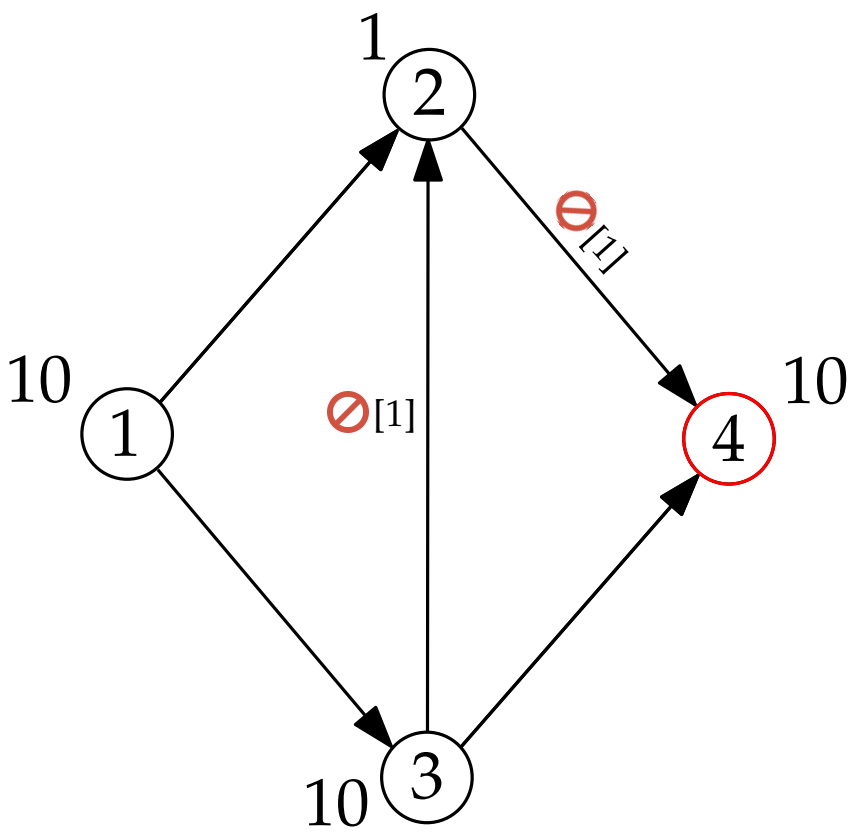


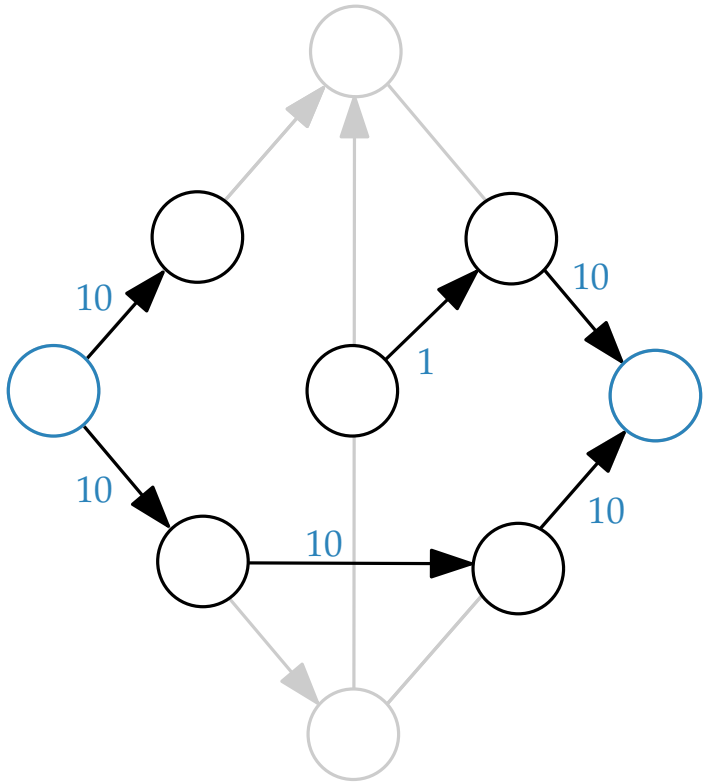
Ansatz: Kantengraph



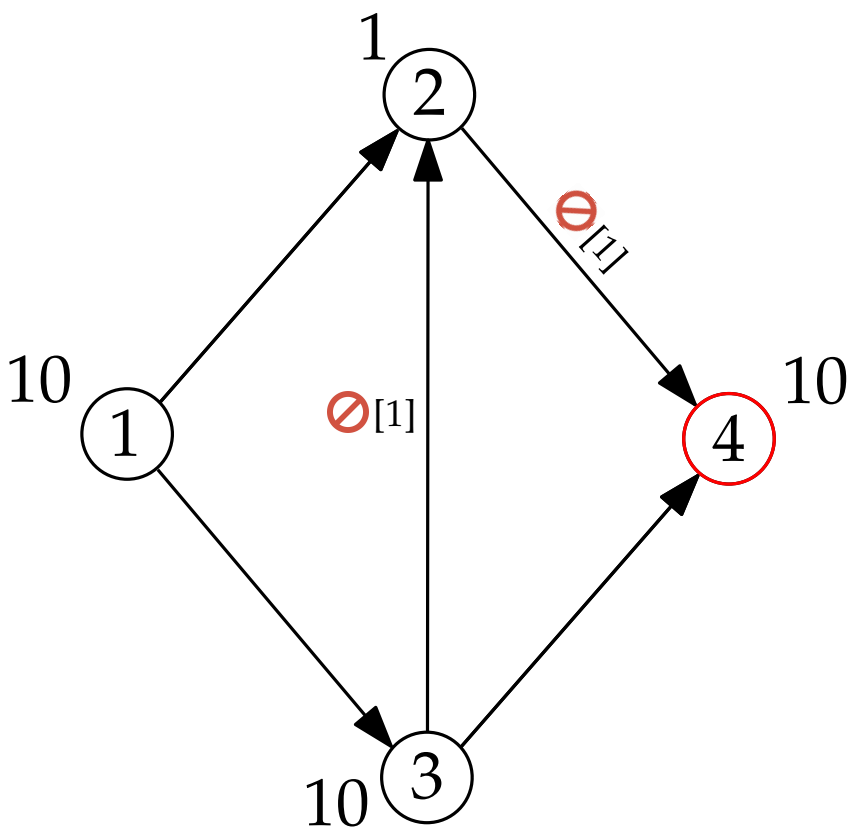


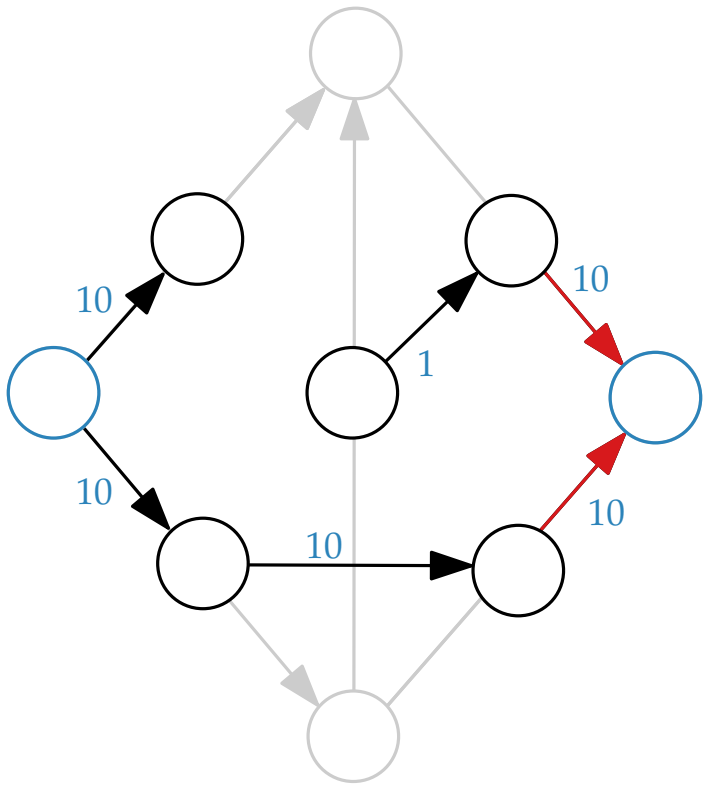
Ansatz: Kantengraph



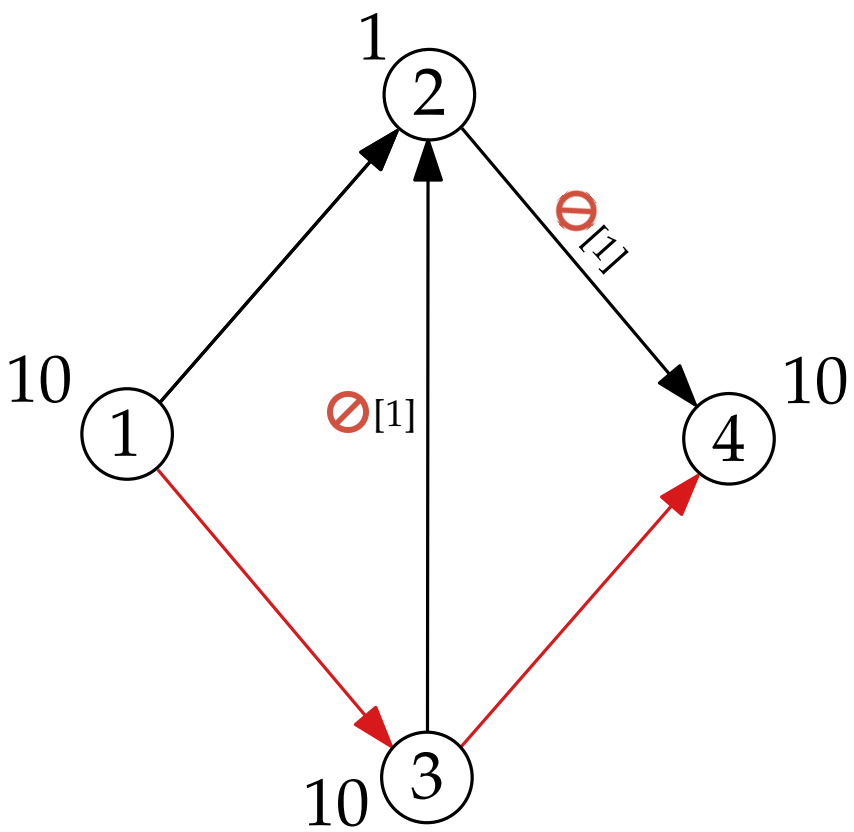


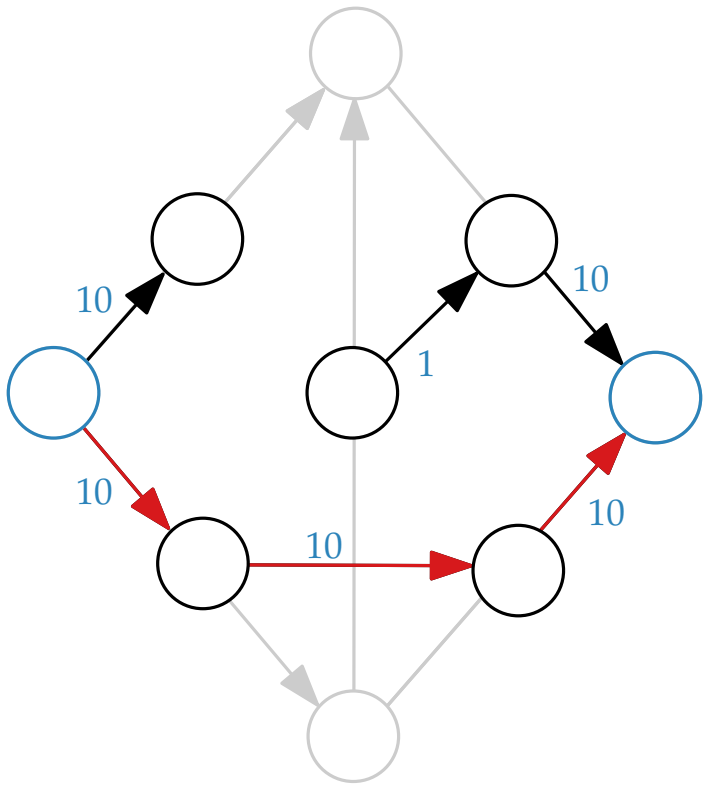
Ansatz: Kantengraph





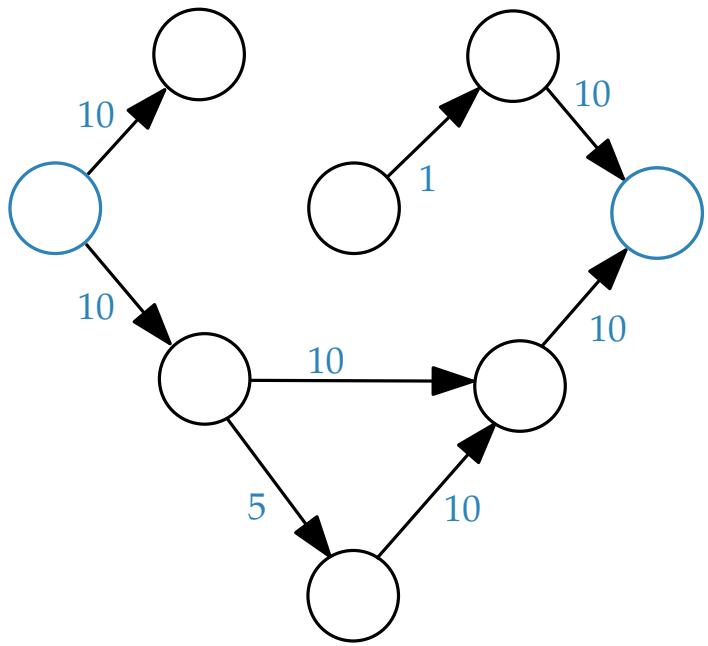
Ansatz: Kantengraph





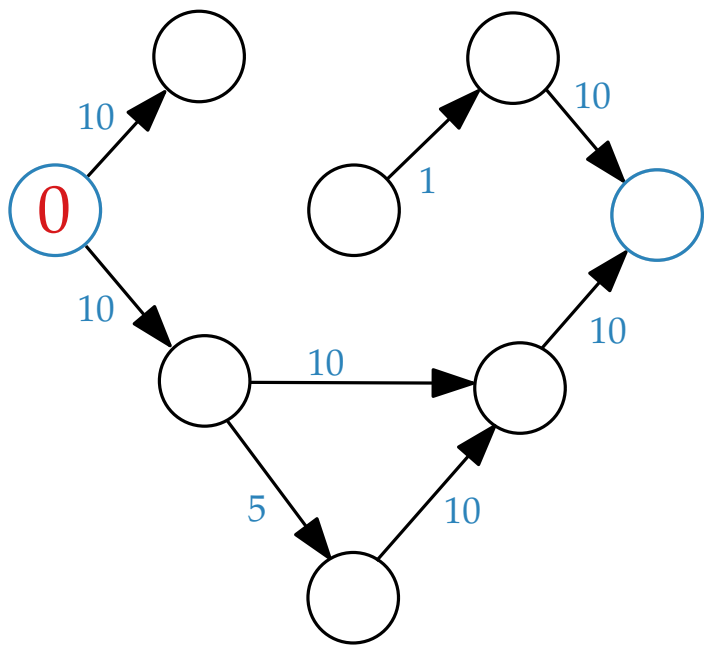
Ansatz: Kürzester Weg (Dijkstra)

→ Anwendung des Dijkstra-Algorithmus zur Bestimmung des kürzesten Weges



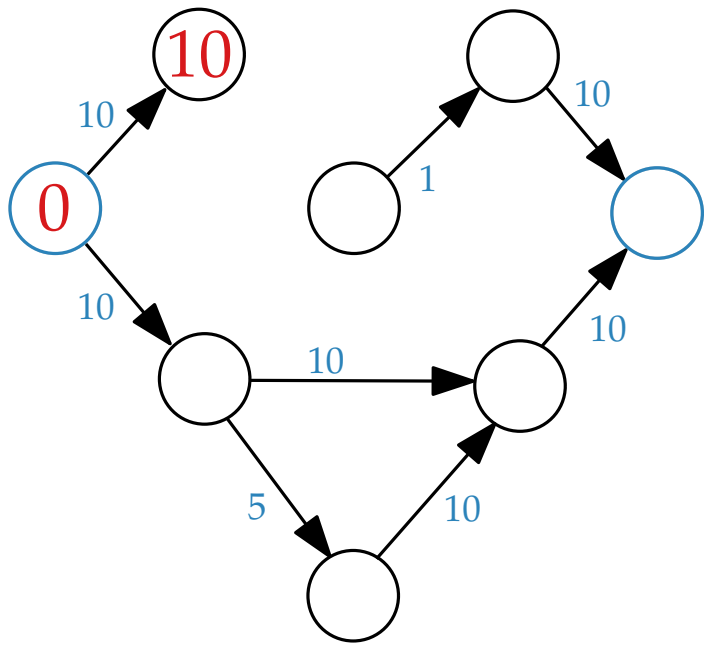
Ansatz: Kürzester Weg (Dijkstra)

→ Anwendung des Dijkstra-Algorithmus zur Bestimmung des kürzesten Weges



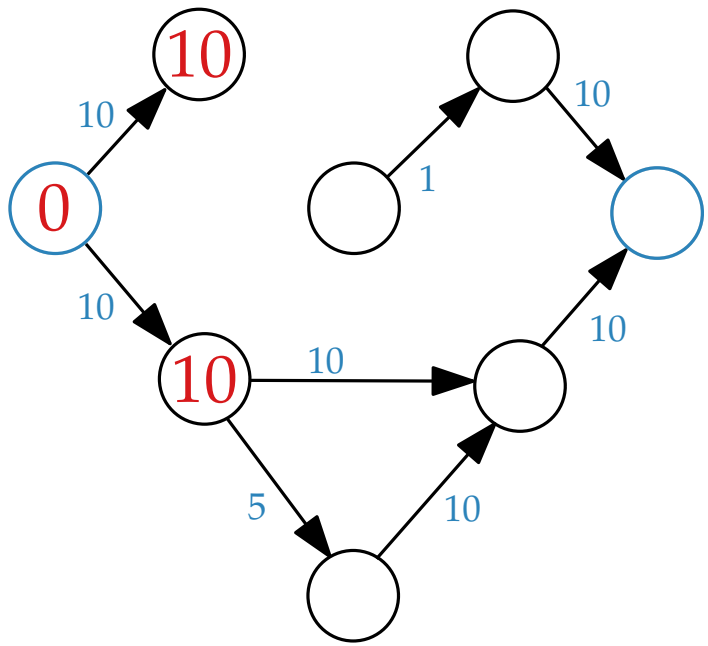
Ansatz: Kürzester Weg (Dijkstra)

→ Anwendung des Dijkstra-Algorithmus zur Bestimmung des kürzesten Weges



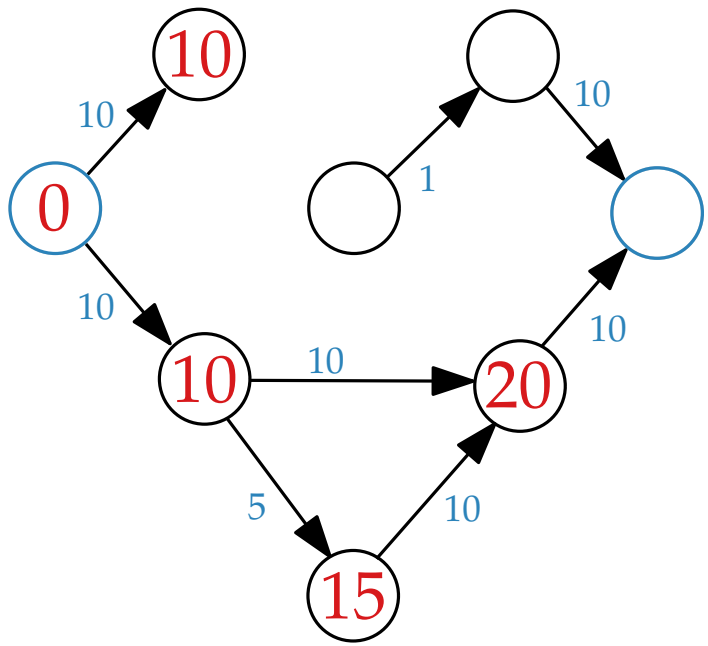
Ansatz: Kürzester Weg (Dijkstra)

→ Anwendung des Dijkstra-Algorithmus zur Bestimmung des kürzesten Weges



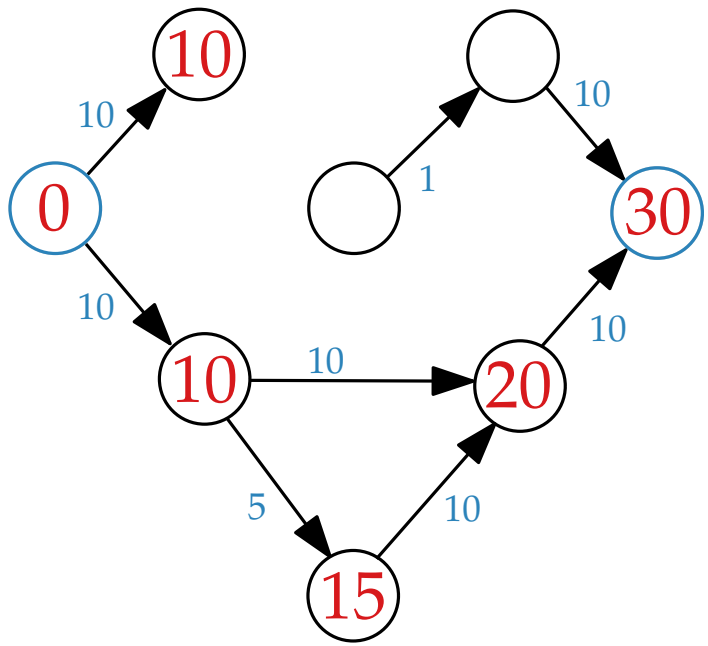
Ansatz: Kürzester Weg (Dijkstra)

→ Anwendung des Dijkstra-Algorithmus zur Bestimmung des kürzesten Weges



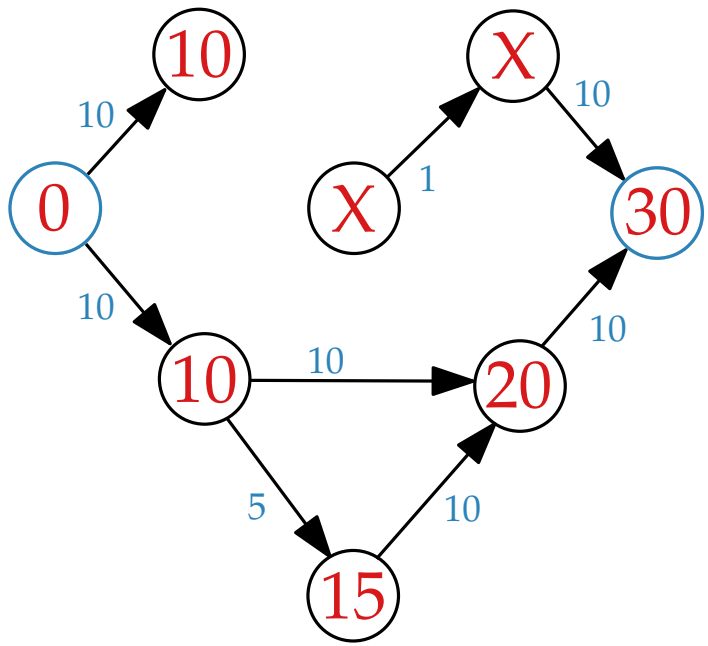
Ansatz: Kürzester Weg (Dijkstra)

→ Anwendung des Dijkstra-Algorithmus zur Bestimmung des kürzesten Weges

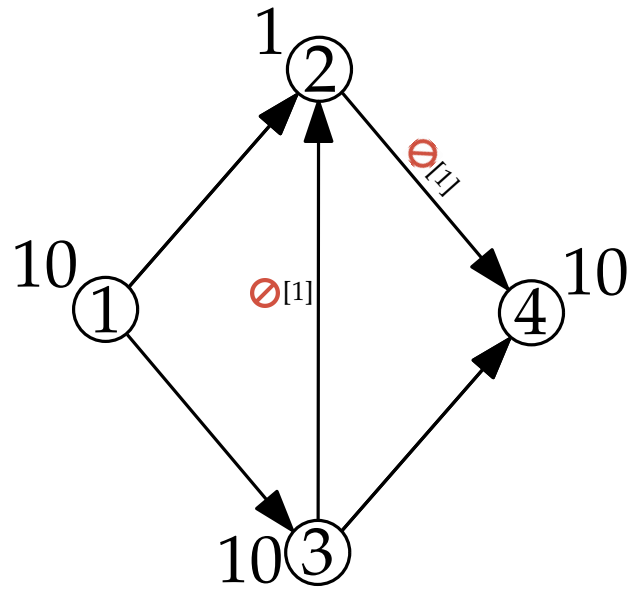


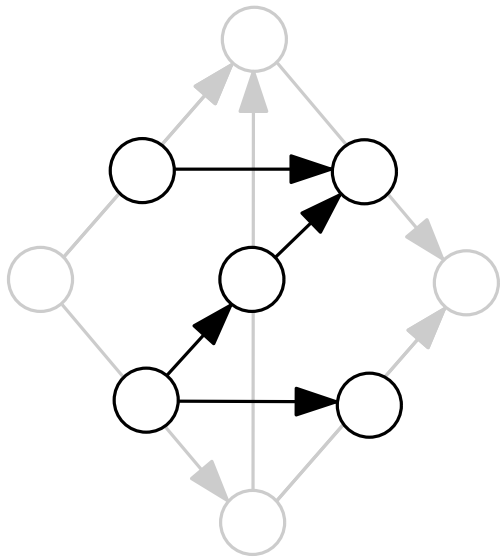
Ansatz: Kürzester Weg (Dijkstra)

→ Anwendung des Dijkstra-Algorithmus zur Bestimmung des kürzesten Weges



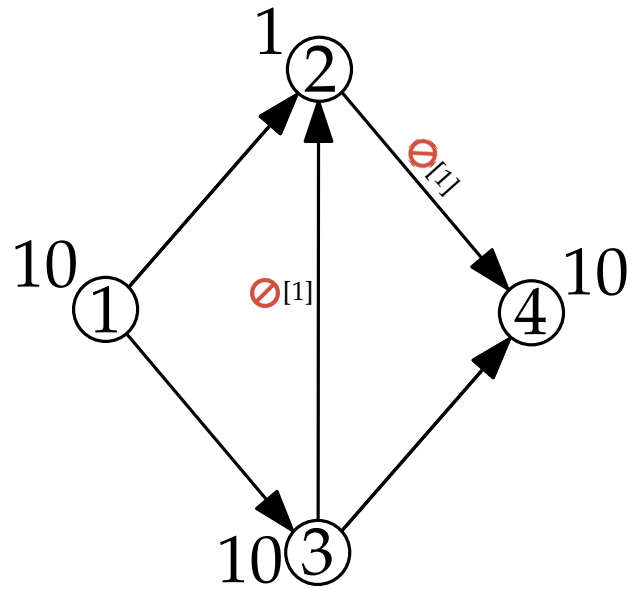
Ansatz: Algorithmus

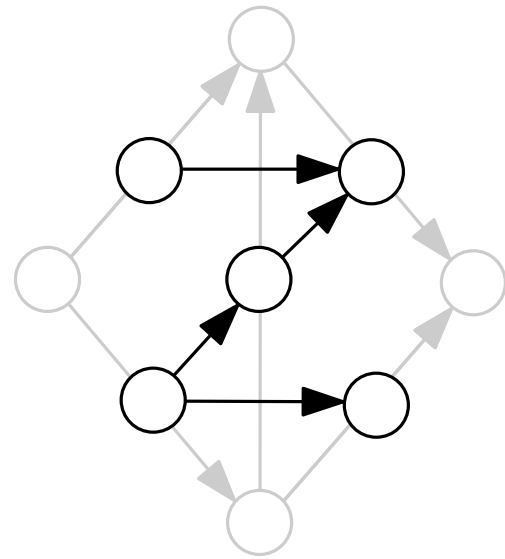




Eingaben: Graph $G = (V, E)$, Startknoten s, Zielknoten t

Ansatz: Algorithmus

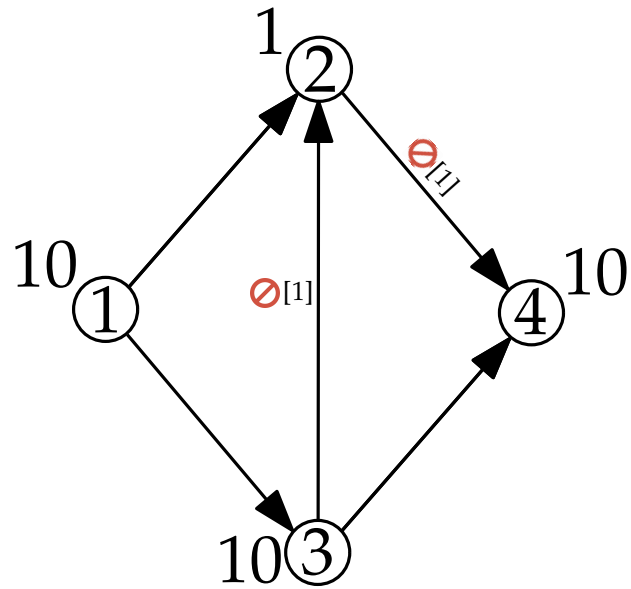


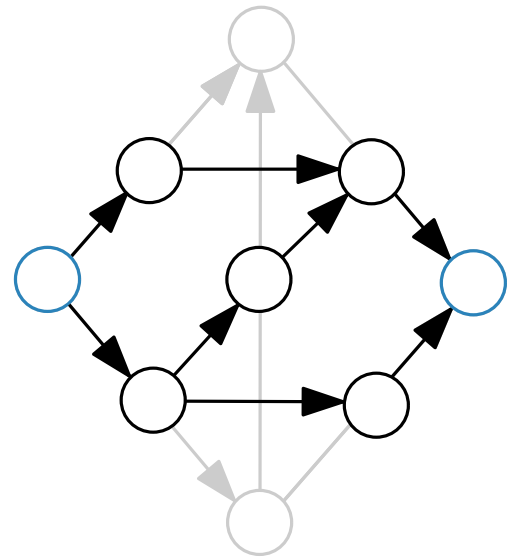


Eingaben: Graph $G = (V, E)$, Startknoten s, Zielknoten t

$G' = (V', E')$
for e in E **do**
  $V' = V' \cup \{v_e\}$
for (v,u) in E **do**
  for x in Adj[u] **do**
    $E' = E' \cup \{(v_{(v,u)}, v_{(u,x)})\}$

Ansatz: Algorithmus

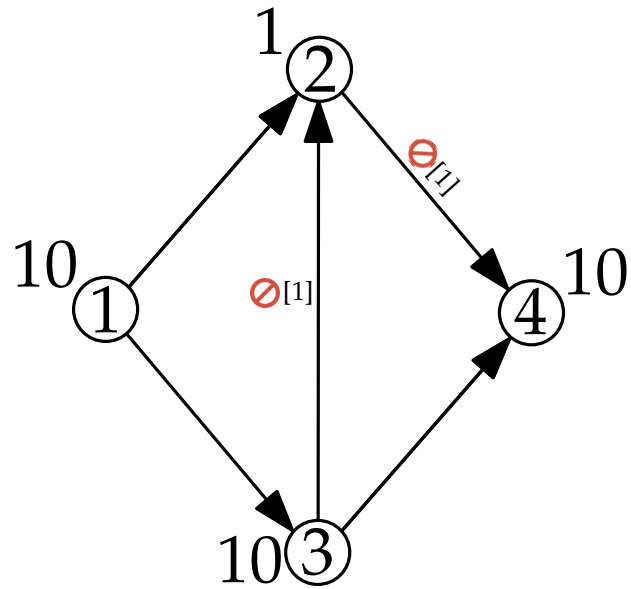


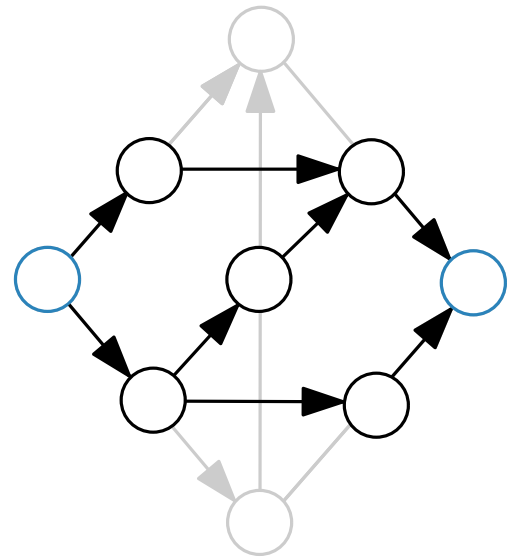


Eingaben: Graph $G = (V, E)$, Startknoten s, Zielknoten t

$G' = (V', E')$
for e in E **do**
  $V' = V' \cup \{v_e\}$
for (v,u) in E **do**
  for x in Adj[u] **do**
    $E' = E' \cup \{(v_{(v,u)}, v_{(u,x)})\}$

Ansatz: Algorithmus



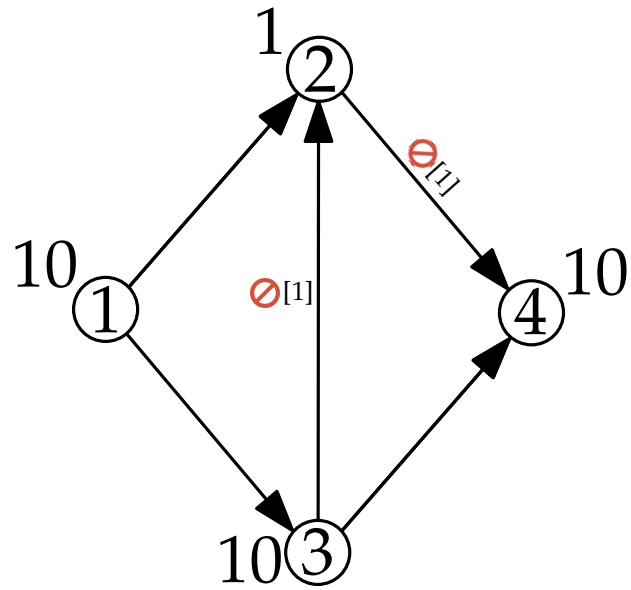


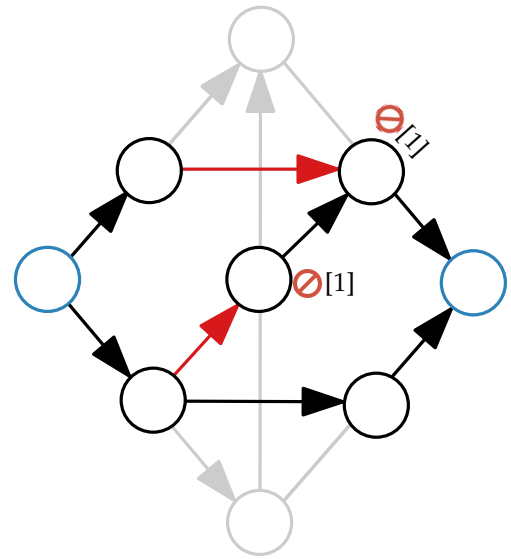
Eingaben: Graph $G = (V, E)$, Startknoten s, Zielknoten t

$G' = (V', E')$
for e in E **do**
  $V' = V' \cup \{v_e\}$
for (v,u) in E **do**
  for x in Adj[u] **do**
    $E' = E' \cup \{(v_{(v,u)}, v_{(u,x)})\}$

$V' = V' \cup \{s', t'\}$
for v in Adj[s] **do**
  $E' = E' \cup \{(s', v_{(s,v)})\}$
for (v, t) in E **do**
  $E' = E' \cup \{(v_{(v,t)}, t')\}$

Ansatz: Algorithmus



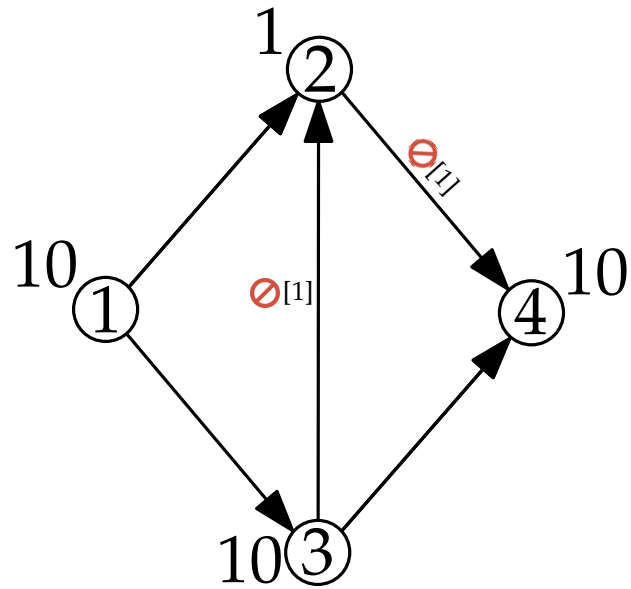


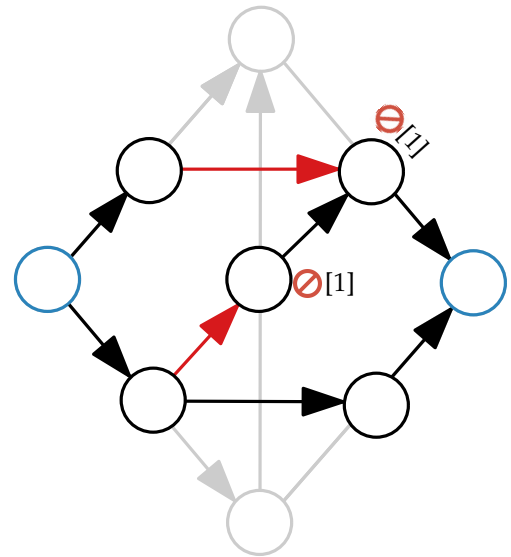
Eingaben: Graph $G = (V, E)$, Startknoten s, Zielknoten t

$G' = (V', E')$
for e in E **do**
  $V' = V' \cup \{v_e\}$
for (v,u) in E **do**
  for x in Adj[u] **do**
    $E' = E' \cup \{(v_{(v,u)}, v_{(u,x)})\}$

$V' = V' \cup \{s', t'\}$
for v in Adj[s] **do**
  $E' = E' \cup \{(s', v_{(s,v)})\}$
for (v, t) in E **do**
  $E' = E' \cup \{(v_{(v,t)}, t')\}$

Ansatz: Algorithmus



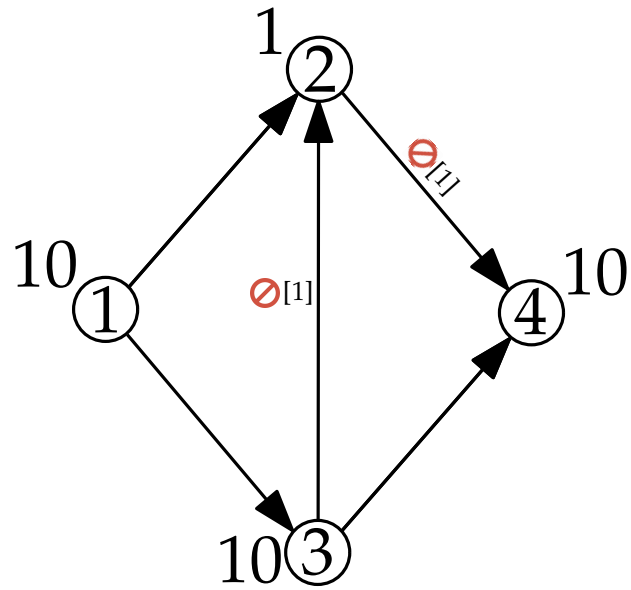


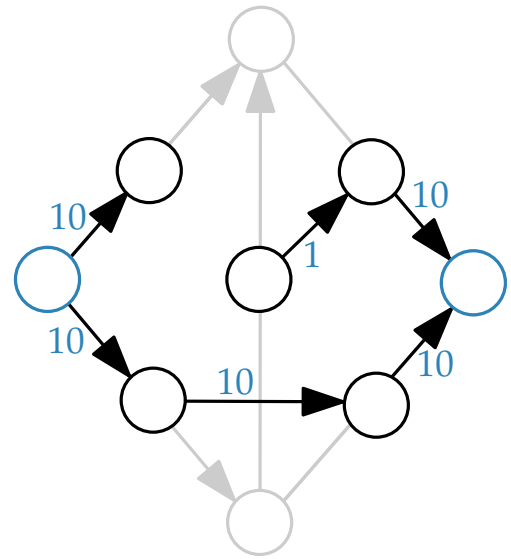
Eingaben: Graph $G = (V, E)$, Startknoten s, Zielknoten t

$G' = (V', E')$
for e in E **do**
  $V' = V' \cup \{v_e\}$
for (v,u) in E **do**
  for x in Adj[u] **do**
    if v nicht blockiert von (u,x) **then**
      $E' = E' \cup \{(v_{(v,u)}, v_{(u,x)})\}$

$V' = V' \cup \{s', t'\}$
for v in Adj[s] **do**
  $E' = E' \cup \{(s', v_{(s,v)})\}$
for (v, t) in E **do**
  $E' = E' \cup \{(v_{(v,t)}, t')\}$

Ansatz: Algorithmus



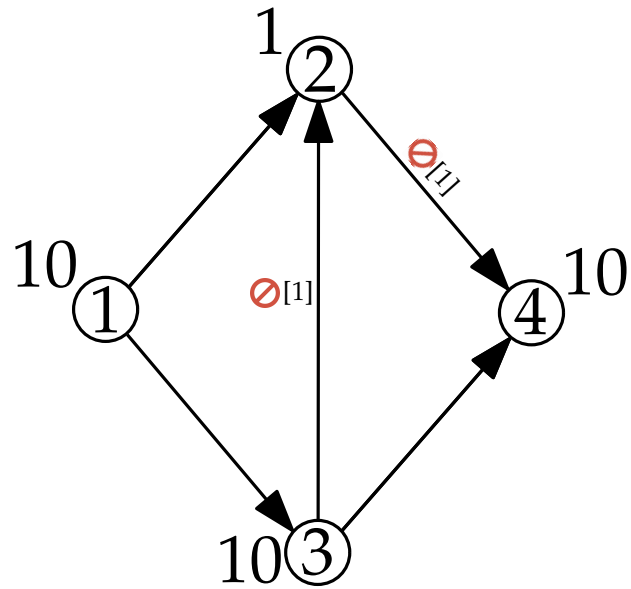


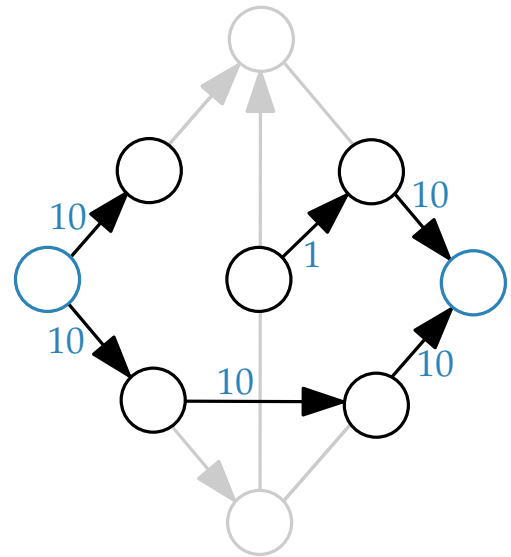
Eingaben: Graph $G = (V, E)$, Startknoten s, Zielknoten t

$G' = (V', E')$
for e in E **do**
  $V' = V' \cup \{v_e\}$
for (v,u) in E **do**
  for x in Adj[u] **do**
    if v nicht blockiert von (u,x) **then**
      $E' = E' \cup \{(v_{(v,u)}, v_{(u,x)})\}$

$V' = V' \cup \{s', t'\}$
for v in Adj[s] **do**
  $E' = E' \cup \{(s', v_{(s,v)})\}$
for (v, t) in E **do**
  $E' = E' \cup \{(v_{(v,t)}, t')\}$

Ansatz: Algorithmus





Eingaben: Graph $G = (V, E)$, Startknoten s, Zielknoten t

$G' = (V', E')$
for e in E **do**
  $V' = V' \cup \{v_e\}$
for (v,u) in E **do**
  for x in Adj[u] **do**
    if v nicht blockiert von (u,x) **then**
      $E' = E' \cup \{(v_{(v,u)}, v_{(u,x)})\}$
      $(v_{(v,u)}, v_{(u,x)}).weight := u.p$

$V' = V' \cup \{s', t'\}$
for v in Adj[s] **do**
  $E' = E' \cup \{(s', v_{(s,v)})\}$
  $(s', v_{(s,v)}).weight := s.p$
for (v, t) in E **do**
  $E' = E' \cup \{(v_{(v,t)}, t')\}$
  $(v_{(v,t)}, t').weight := t.p$

Ansatz: Laufzeit

Eingabelänge: $O(|E| \cdot |V|)$

Ansatz: Laufzeit

Eingabelänge: $O(|E| \cdot |V|)$

$G' = (V', E')$
for e in E **do**
  $V' = V' \cup \{v_e\}$
for (v,u) in E **do**
  for x in Adj[u] **do**
    if v nicht geblockt von (u,x) **then**
      $E' = E' \cup \{(v_{(v,u)}, v_{(u,x)})\}$
      $(v_{(v,u)}, v_{(u,x)}).weight := u.processingTime$
$V' = V' \cup \{s', t'\}$
for v in Adj[s] **do**
  $E' = E' \cup \{(s', v_{s,v})\}$
for (v, t) in E **do**
  $E' = E' \cup \{(v_{(v,t)}, t')\}$

Ansatz: Laufzeit

Eingabelänge: $O(|E| \cdot |V|)$

$G' = (V', E')$
for e in E **do**
  $V' = V' \cup \{v_e\}$ $O(|E|)$
for (v,u) in E **do**
  for x in Adj[u] **do**
    if v nicht geblockt von (u,x) **then**
      $E' = E' \cup \{(v_{(v,u)}, v_{(u,x)})\}$
      $(v_{(v,u)}, v_{(u,x)}).weight := u.processingTime$
$V' = V' \cup \{s', t'\}$
for v in Adj[s] **do**
  $E' = E' \cup \{(s', v_{s,v})\}$
for (v, t) in E **do**
  $E' = E' \cup \{(v_{(v,t)}, t')\}$

Ansatz: Laufzeit

Eingabelänge: $O(|E| \cdot |V|)$

$G' = (V', E')$
for e in E **do**
  $V' = V' \cup \{v_e\}$ $O(|E|)$
for (v,u) in E **do** $O(|E| \cdot |V|)$
  for x in Adj[u] **do**
    if v nicht geblockt von (u,x) **then**
      $E' = E' \cup \{(v_{(v,u)}, v_{(u,x)})\}$
      $(v_{(v,u)}, v_{(u,x)}).weight := u.processingTime$
$V' = V' \cup \{s', t'\}$
for v in Adj[s] **do**
  $E' = E' \cup \{(s', v_{s,v})\}$
for (v, t) in E **do**
  $E' = E' \cup \{(v_{(v,t)}, t')\}$

Ansatz: Laufzeit

Eingabelänge: $O(|E| \cdot |V|)$

$G' = (V', E')$
for e in E **do**
  $V' = V' \cup \{v_e\}$ $O(|E|)$
for (v,u) in E **do** $O(|E| \cdot |V|)$
  for x in Adj[u] **do**
    if v nicht geblockt von (u,x) **then**
      $E' = E' \cup \{(v_{(v,u)}, v_{(u,x)})\}$
      $(v_{(v,u)}, v_{(u,x)}).weight := u.processingTime$
$V' = V' \cup \{s', t'\}$
for v in Adj[s] **do**
  $E' = E' \cup \{(s', v_{s,v})\}$ $O(|V|)$
for (v, t) in E **do**
  $E' = E' \cup \{(v_{(v,t)}, t')\}$

Ansatz: Laufzeit

Eingabelänge: $O(|E| \cdot |V|)$

$G' = (V', E')$
for e in E **do**
  $V' = V' \cup \{v_e\}$ $O(|E|)$

for (v,u) in E **do** $O(|E| \cdot |V|)$
  for x in Adj[u] **do**
    if v nicht geblockt von (u,x) **then**
      $E' = E' \cup \{(v_{(v,u)}, v_{(u,x)})\}$
      $(v_{(v,u)}, v_{(u,x)}).weight := u.processingTime$

$V' = V' \cup \{s', t'\}$
for v in Adj[s] **do**
  $E' = E' \cup \{(s', v_{s,v})\}$ $O(|V|)$

for (v, t) in E **do**
  $E' = E' \cup \{(v_{(v,t)}, t')\}$ $O(|V|)$

Ansatz: Laufzeit

Eingabelänge: $O(|E| \cdot |V|)$

$G' = (V', E')$
for e in E **do**
  $V' = V' \cup \{v_e\}$ $O(|E|)$
for (v,u) in E **do** $O(|E| \cdot |V|)$
  for x in Adj[u] **do**
    if v nicht geblockt von (u,x) **then**
      $E' = E' \cup \{(v_{(v,u)}, v_{(u,x)})\}$
      $(v_{(v,u)}, v_{(u,x)}).weight := u.processingTime$
$V' = V' \cup \{s', t'\}$
for v in Adj[s] **do**
  $E' = E' \cup \{(s', v_{s,v})\}$ $O(|V|)$
for (v, t) in E **do**
  $E' = E' \cup \{(v_{(v,t)}, t')\}$ $O(|V|)$

Dijkstra: $O(|E|^2)$ (eigentlich: $O(|V'|^2)$)

Ansatz: Laufzeit

Eingabelänge: $O(|E| \cdot |V|)$

$G' = (V', E')$
for e in E **do**
  $V' = V' \cup \{v_e\}$ $O(|E|)$

for (v,u) in E **do** $O(|E| \cdot |V|)$
  for x in Adj[u] **do**
    if v nicht geblockt von (u,x) **then**
      $E' = E' \cup \{(v_{(v,u)}, v_{(u,x)})\}$
      $(v_{(v,u)}, v_{(u,x)}).weight := u.processingTime$

$V' = V' \cup \{s', t'\}$
for v in Adj[s] **do**
  $E' = E' \cup \{(s', v_{s,v})\}$ $O(|V|)$

for (v, t) in E **do**
  $E' = E' \cup \{(v_{(v,t)}, t')\}$ $O(|V|)$

Dijkstra: $O(|E|^2)$ (eigentlich: $O(|V'|^2)$)

Gesamtlaufzeit: $O(|E|^2) \cup O(|E| \cdot |V|)$