PROBLEM L: MOLVANIAN CASTLES

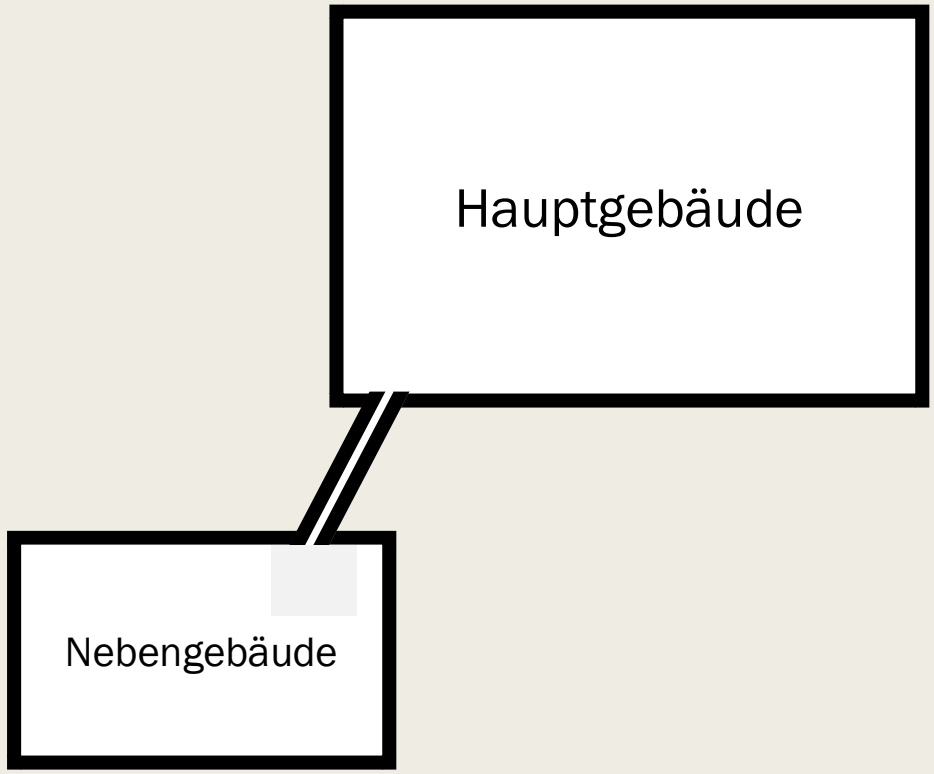
Bastian Häuser
Moritz Hering

Einführung

Vor langer Zeit ließ König Sane, der Erste seines Namens, im Königreich Molvanien prächtige Burgen errichten. Der Aufbau seiner Burgen folgte dabei seinem Regierungsmotto: „Teile und Herrsche“. Sie waren hierarchisch strukturiert und bestanden aus mehreren miteinander verbundenen Gebäuden. In den einzelnen Gebäuden gab es prunkvolle Hallen und Korridore, die die Hallen miteinander verbanden.

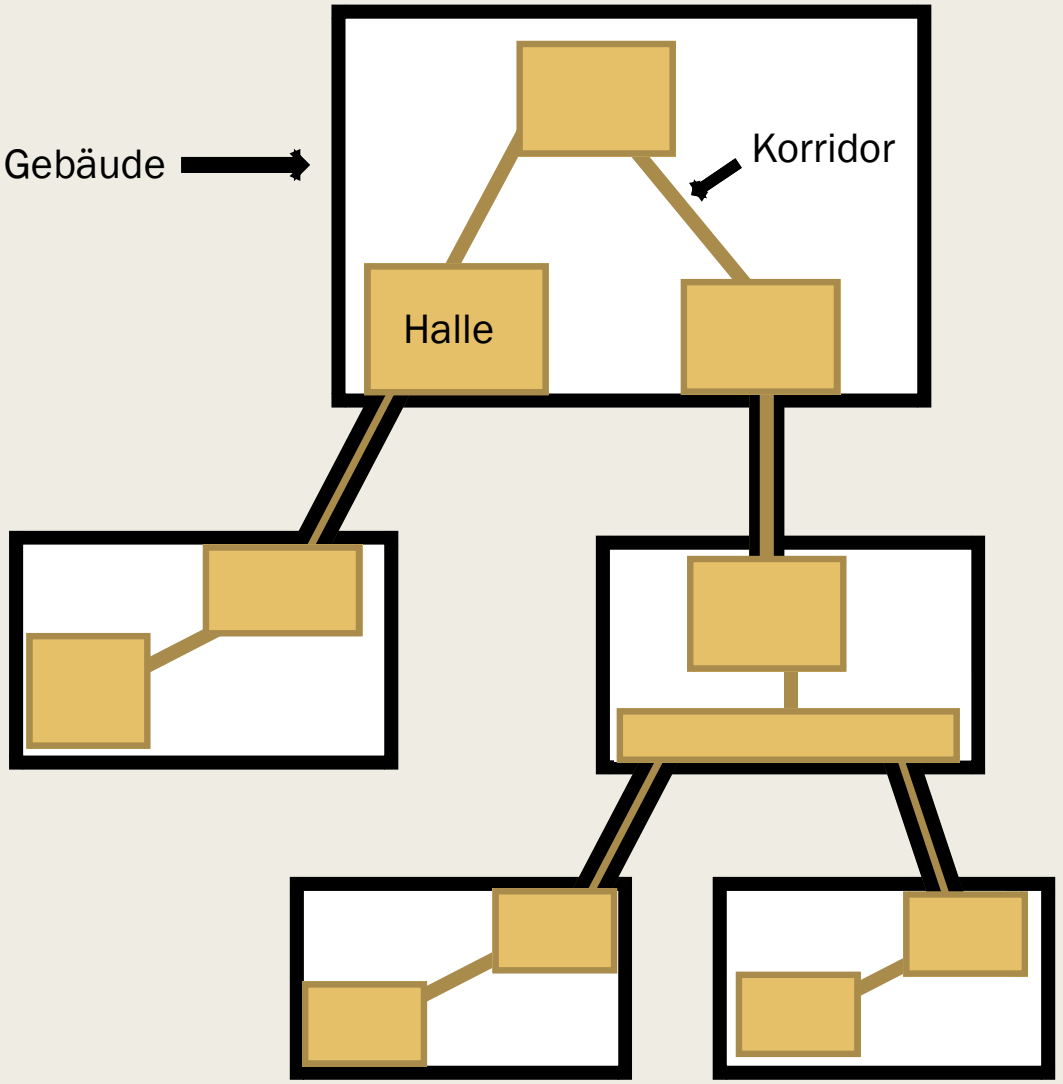
Aufbau der Burgen

Anfangs bestand eine Burg nur aus einem Hauptgebäude. Wenn die Bevölkerung des Landes wuchs wurde die Burg folgendermaßen erweitert:
Zunächst wurde ein neues Gebäude gebaut, welches anschließend mit einem der bereits existierenden über einen Korridor verbunden wurde.
Das neue Gebäude bestand selbst wieder aus Hallen und Korridoren.



Aufbau der Burg

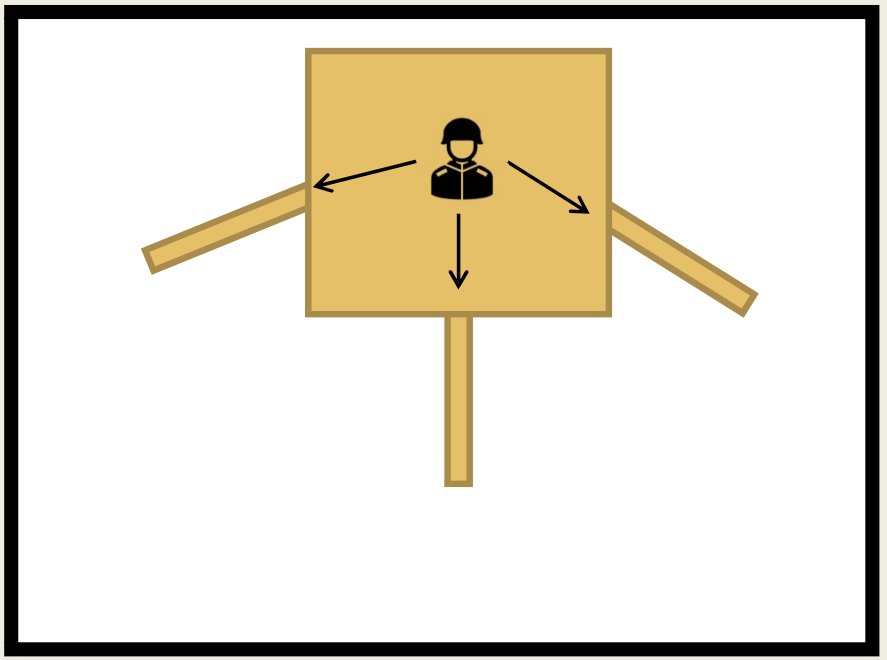
So entstand mit der Zeit eine Burg, die aus vielen Gebäuden, Hallen und Korridoren bestand.



Aufgabenstellung

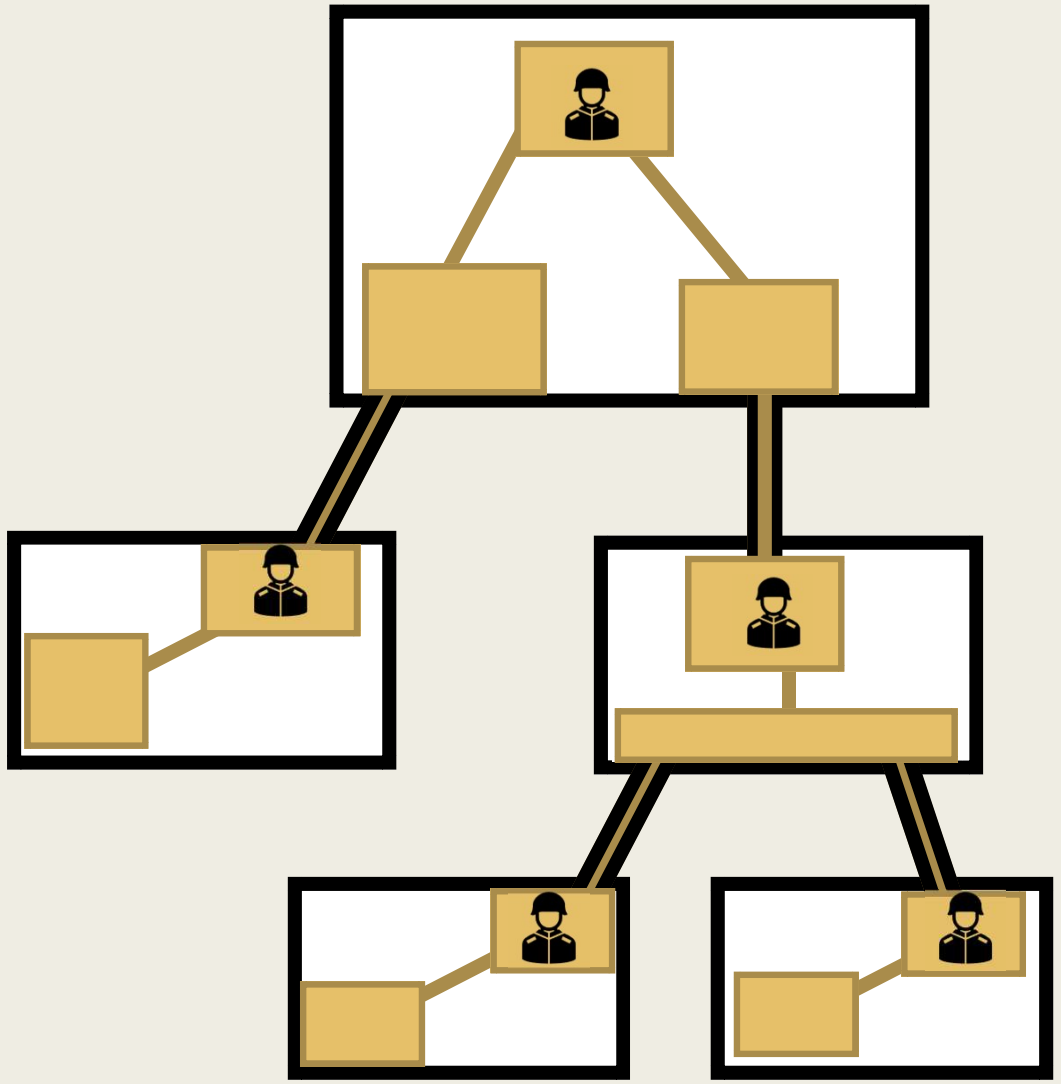
In Zeiten von Rebellion und Aufruhr ließ der König alle Korridore überwachen. Hierzu wurden Wachen in den Hallen aufgestellt, wobei eine Wache alle an die Halle angrenzende Korridore überblicken konnte.

Da er seine Soldaten aber für die Armee benötigte, wollte er natürlich nur so wenige wie möglich für den Wachdienst abstellen.



Aufgabenstellung

Deshalb verlangte er von seinem Hofmathematiker die geringste Anzahl an Wachen zu bestimmen, die nötig ist, um alle Korridore zu überwachen.



Die Burgen als Graph

- Wir können die Burgen natürlich als Graphen auffassen:

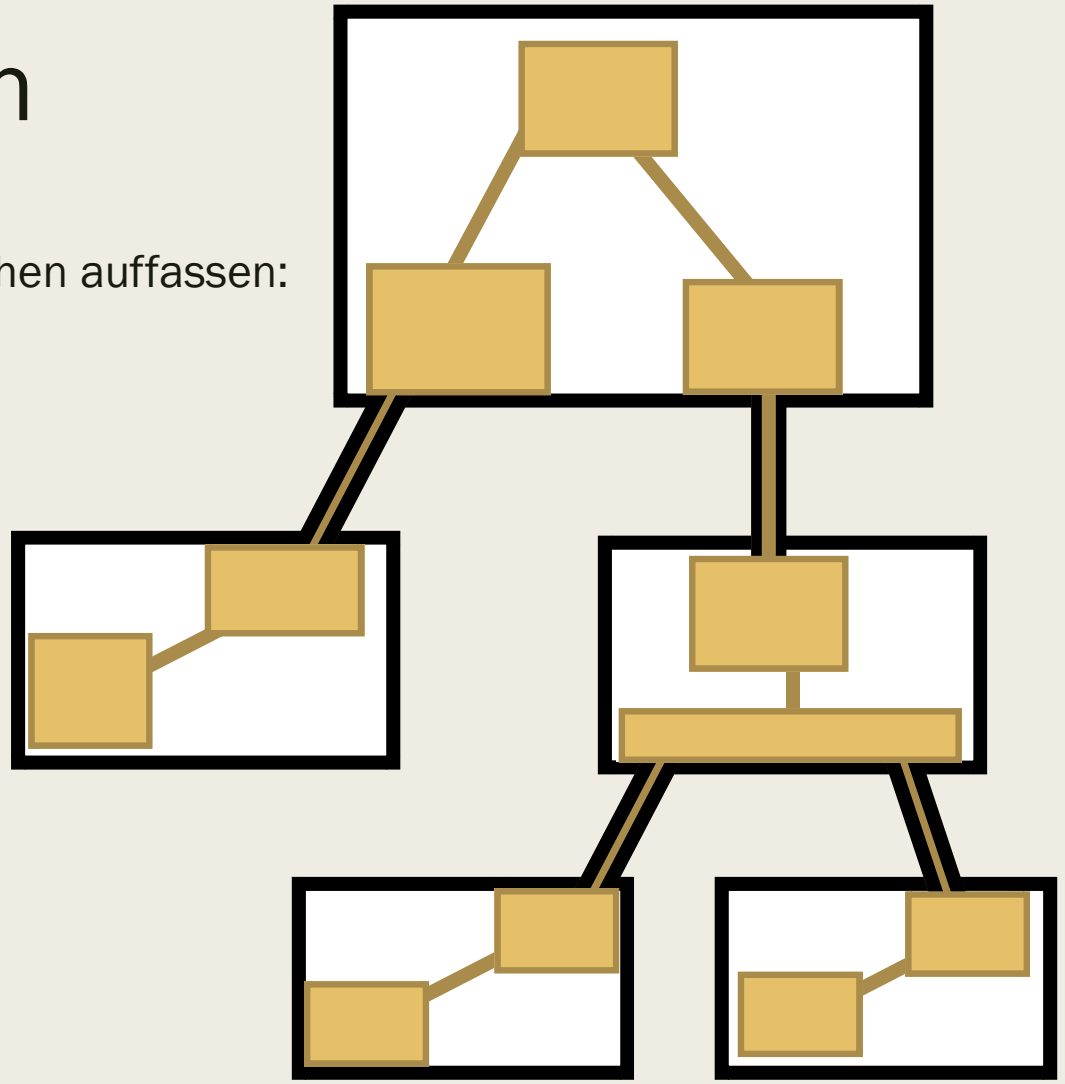
  Hallen → Knoten

  Korridore → Kanten

  Gebäude → Teilgraphen

  Korridore zwischen Gebäuden

  → Verbindungskanten



Die Burgen als Graph

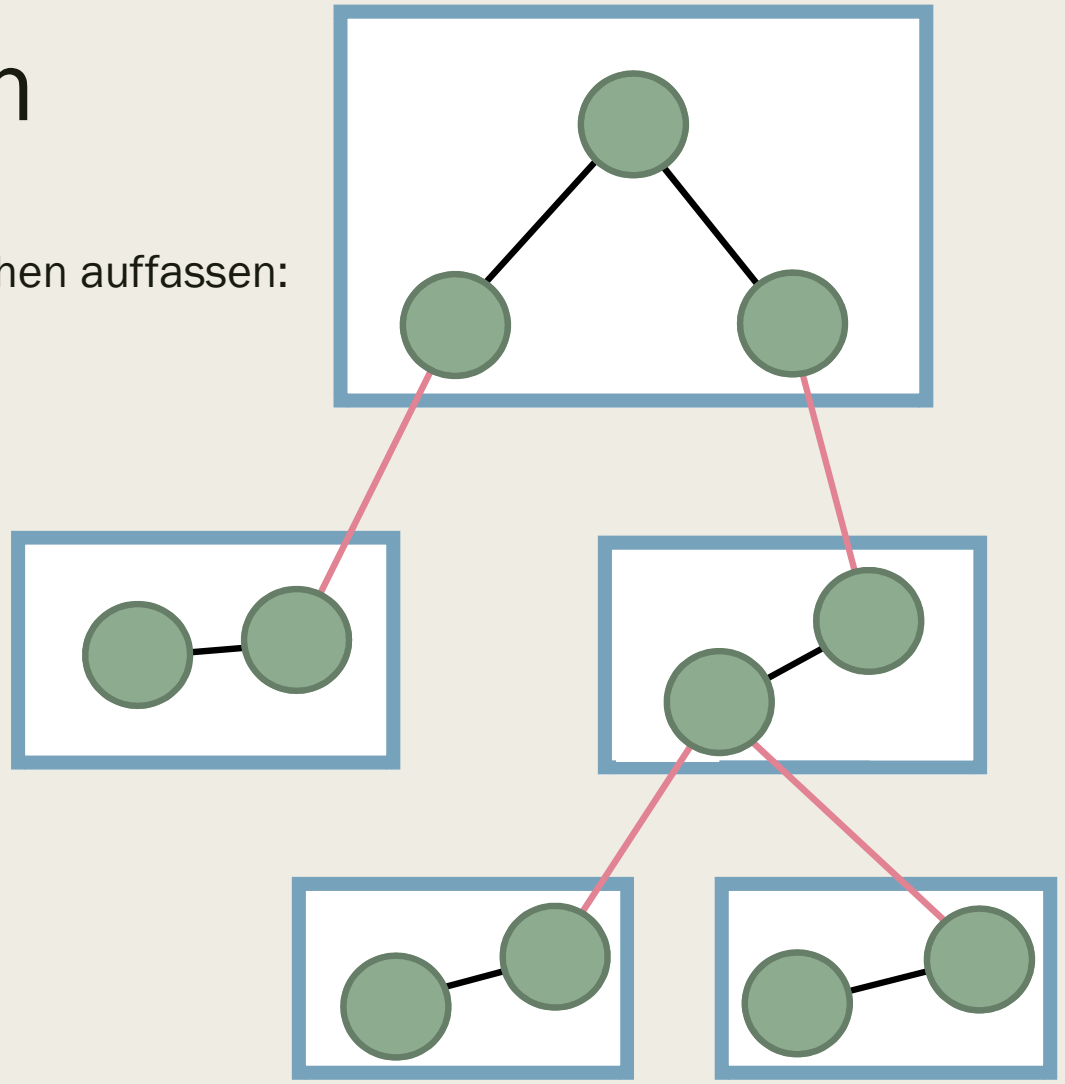
- Wir können die Burgen natürlich als Graphen auffassen:

Hallen → Knoten

Korridore → Kanten

Gebäude → Teilgraphen

Korridore zwischen Gebäuden
→ Verbindungskanten



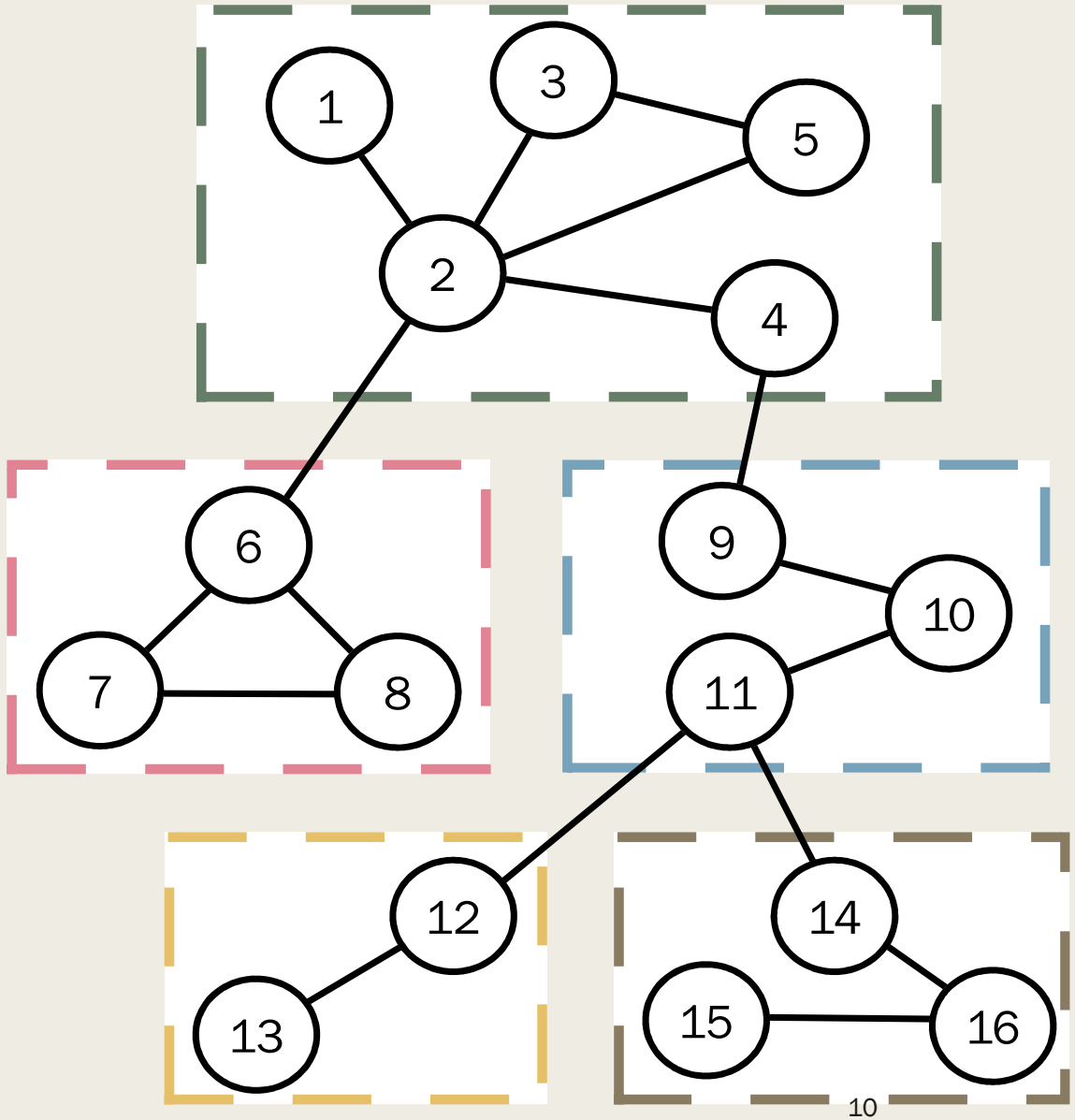
Der Input

- Ein Input steht jeweils für eine Burg, die angefangen beim Hauptgebäude rekursiv definiert ist:
  1. Zeile mit 3 integer Werten (*n m w*):
     - *n* : Anzahl an Hallen im Gebäude ($2 \leq n \leq 10$)
     - *m* : Anzahl an Korridoren im Gebäude ($1 \leq m \leq 45$)
     - *w* : Anzahl an angefügten Gebäuden ($0 \leq w \leq 10$)
  2. Eine Zeile für jeden der *m* Korridore: *x y* → Korridor zwischen Halle *x* und *y* des aktuellen Gebäudes.
  3. Für jedes der angefügten Gebäude:
     - Eine Zeile mit *v z* → Korridor zwischen Halle *v* (im aktuellen Gebäude) und Halle *z* (im angefügten Gebäude).
     - Die Struktur des angefügten Gebäudes (und möglicherweise an dieses wiederum angefügte Gebäude), beschrieben durch die Nutzung der Regeln 1 bis 3.

Input Beispiel

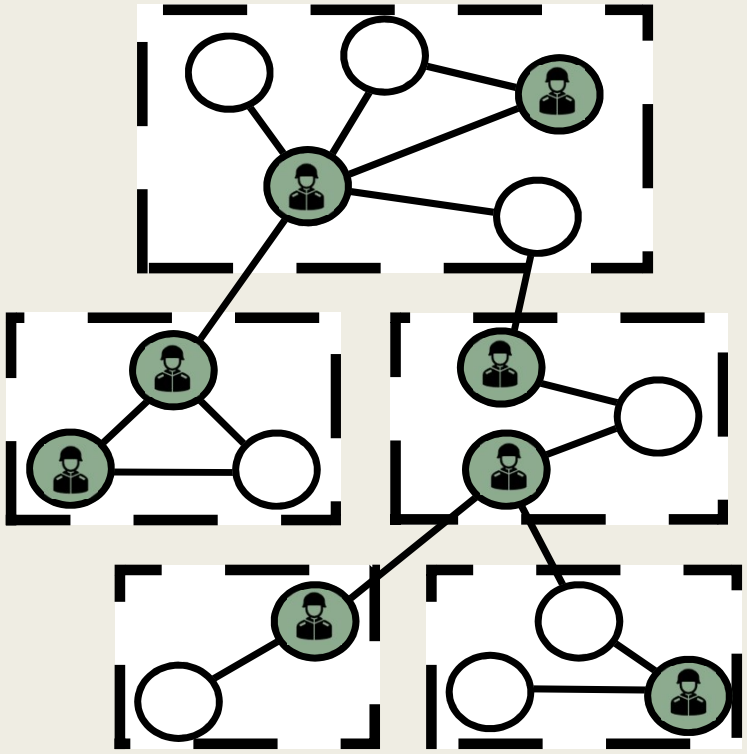
```
5 5 2
1 2
2 3
3 5
2 5
2 4
2 6
3 3 0
6 7
7 8
8 6
```

```
4 9
3 2 2
9 10
10 11
11 12
2 1 0
12 13
11 14
3 2 0
14 16
16 15
```



Output

- Minimale Anzahl nötiger Wachen um alle Korridore zu überwachen.



Min. Anzahl an Wachen: **8**

Input Verarbeiten

- Input könnte folgendermaßen eingelesen werden:

```
read_Building():
    read_Descripition_Line();
    for i=0 to number_Edges:
        read(internal_Edge);
    for i=0 to number_peripheral_Buildings:
        read(connecting_Edge);
        read_Building();
```

Minimale Knotenüberdeckung (Minimum Vertex Cover)

- Sei $G = (V, E)$ ein ungerichteter Graph.
- Eine Menge $U \subseteq V$ ist eine Knotenüberdeckung von G, wenn jede Kante $e \in E$ mindestens einen Knoten $v \in U$ enthält.
- Eine Knotenüberdeckung U von G ist minimal, wenn es **keine** andere Knotenüberdeckung W gibt mit $|W| < |U|$.
- Wir suchen nicht die konkreten minimalen Knotenüberdeckungen, sondern ihre Mächtigkeit.
- In der Literatur wird diese Zahl auch als $Minimum\ Vertex\ Covering\ Number$ bezeichnet.

Minimale Knotenüberdeckung Bestimmen

- Bilden aller Teilmengen X von V.
- Für alle Teilmengen, angefangen bei den kleinsten, überprüfen, ob sie eine Knotenüberdeckung sind.
- Wie bildetet man alle Teilmengen einer Menge?

Teilmengen Bilden

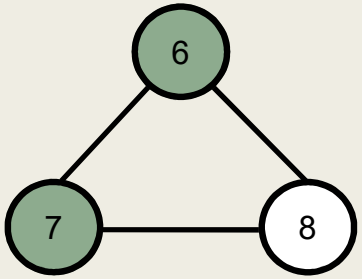
- Nutzung der Binärdarstellung eines integer Wertes
- Beispiel:

```
M = {a, b, c}
For i = 0 to (2^M.length-1)
```

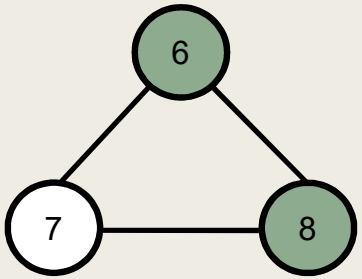
Binärdarstellung	Teilmenge
000	ø
001	{c}
010	{b}
011	{b,c}
100	{a}
101	{a,c}
110	{a,b}
111	{a,b,c}

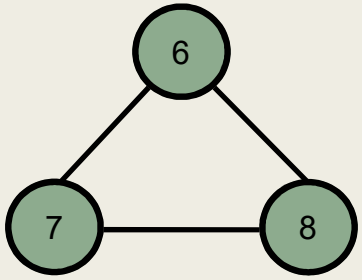
Beispiel

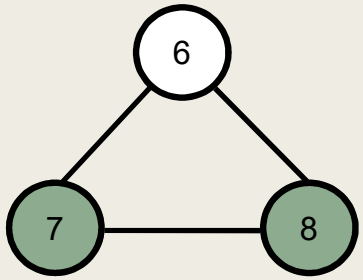
- Sei $G = (V, E)$ mit $V = \{6,7,8\}$ und $E = \{\{6,7\},\{7,8\},\{8,6\}\}$



Teilmenge	Knotenüberdeckung
{6}	Nein
{7}	Nein
{8}	Nein
{6,7}	Ja (minimal)
{6,8}	Ja (minimal)
{7,8}	Ja (minimal)
{6,7,8}	Ja

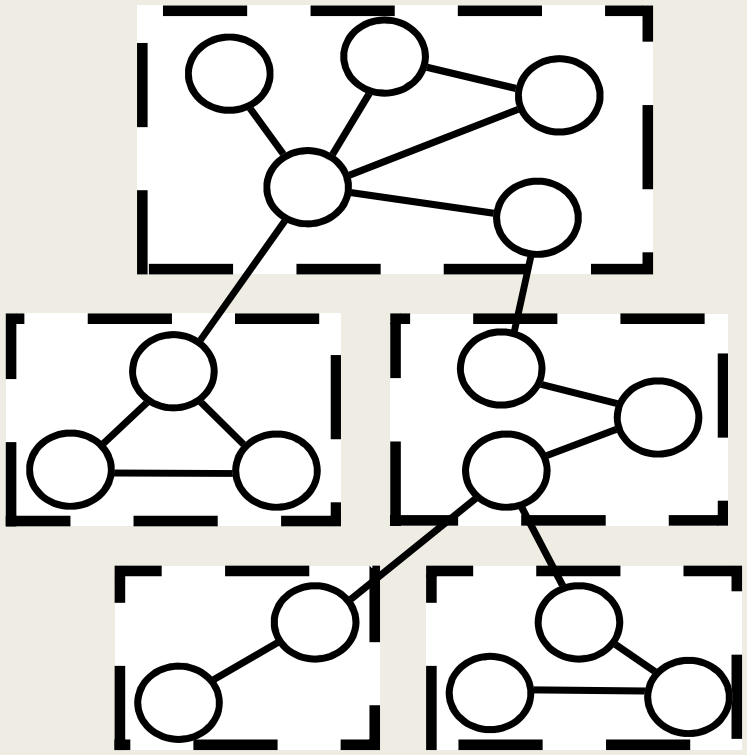


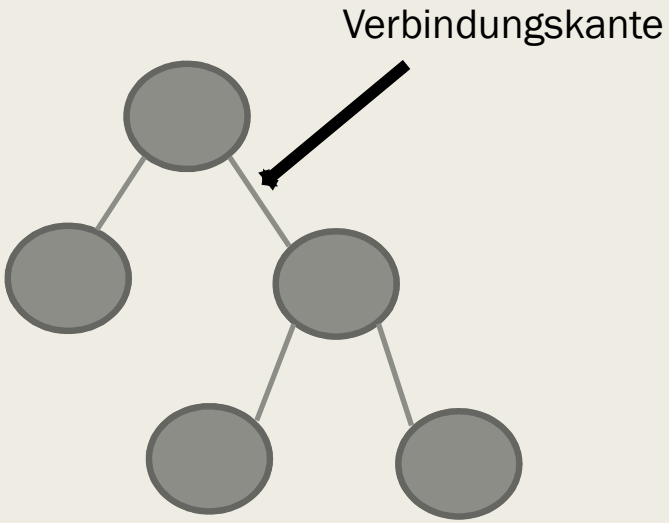




Burg als Baum

- Beobachtung: Teilgraphen bilden eine Baumstruktur





Lösungsansatz für unsere Burgen

Erinnerung: Maximal 10 Hallen und 45 Korridore in einem Gebäude!!!

- Minimale Knotenüberdeckung ist im ***allgemeinen*** NP-Vollständig.

→ nutze ***spezielle*** Form des Graphen, um Lösung doch in polynomieller Zeit zu finden:

- Auf kleinen Mengen ist die Laufzeit von nicht polynomiellen Algorithmen noch akzeptabel
- Wenn wir die Überdeckung für die limitierten Teilgraphen(Gebäude) unabhängig voneinander berechnen, wäre der Gesamtalgorithmus noch polynomiell
- Problem: Was machen wir mit den Verbindungskanten?

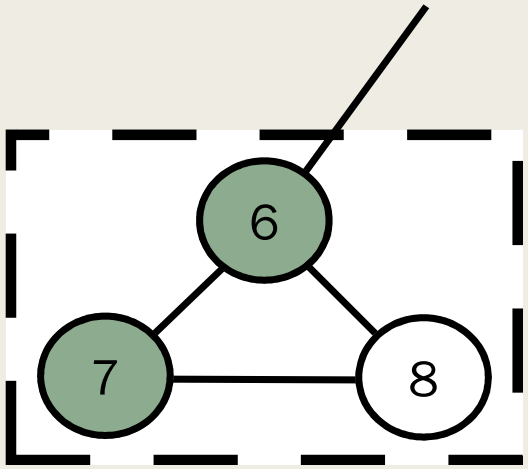
Verbindungskanten

- Betrachte „Kind“ – Teilgraphen
  - 1. Fall: Eine der minimalen Knotenüberdeckungen enthält den Endknoten der Verbindungskante.
    → wir merken uns die Kante als überdeckt.
  - 2. Fall: Keine der minimalen Knotenüberdeckungen enthält den Endknoten.
    → wir müssten den Endknoten zur Überdeckung hinzufügen.
    → Mächtigkeit der Menge stiege um 1 (nicht gut!!).
    besser: Kante dem „Eltern“ – Teilgraphen hinzufügen.

Verbindungskante Beispiel

- 1. Fall: Es gibt **eine** minimale Knotenüberdeckung die den Verbindungsknoten (hier 6) bereits enthält.

  → Nichts mehr zu erledigen



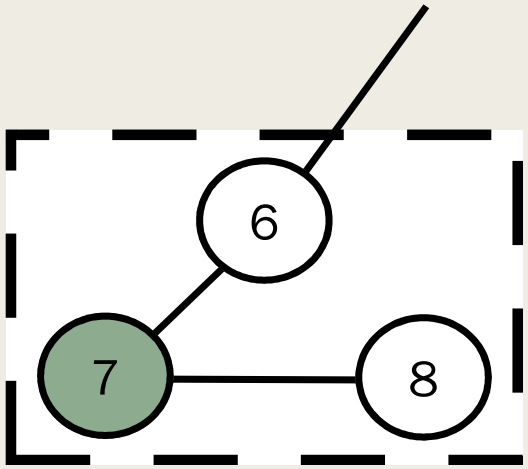
Verbindungskante Beispiel

- 1. Fall: Es gibt **keine** minimale Knotenüberdeckung die den Verbindungsknoten (hier 6) bereits enthält.

  → Der Verbindungsknoten müsste hinzugefügt werden.

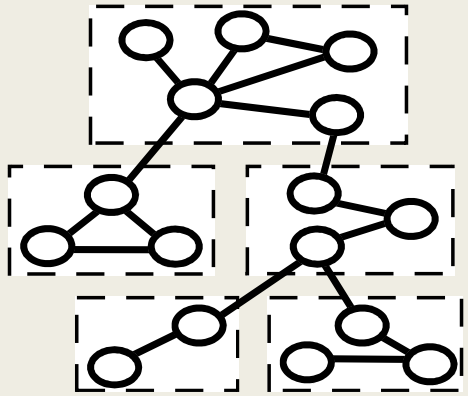
  → Mächtigkeit der Menge nähme um 1 zu.

- Lösung: Füge die Verbindungskante der Kantenmenge des „Eltern“ – Graphen hinzu.



Beweis folg später!

Skizze des Algorithmus



- Wir betrachten den Baum aus Teilgraphen und berechnen deren Lösung isoliert. Dabei gehen wir von den Blättern zur Wurzel hin vor.

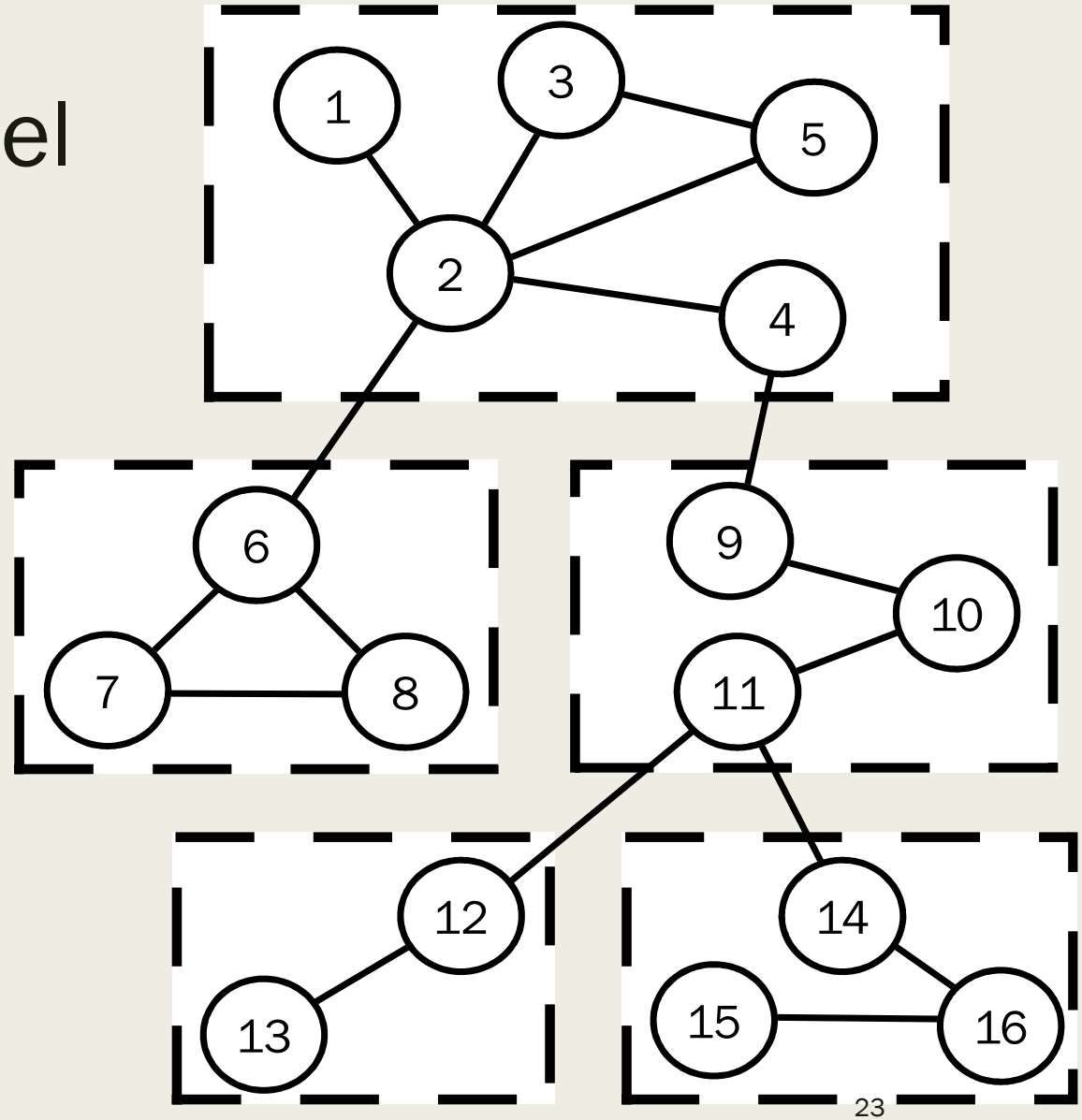
Berechnen der Knotenüberdeckung für einen Teilgraphen:

Voraussetzung: Die Knotenüberdeckung der „Kinder“ ist schon berechnet.

1. Füge nicht überdeckte Verbindungskanten der Kinder zur internen Kantenmenge hinzu
2. Finde per Bruteforce minimale Überdeckungen für die neue Kantenmenge
3. Enthält eine dieser Überdeckungen den Eingangsknoten, so notiere die eingehende Verbindungskante als überdeckt

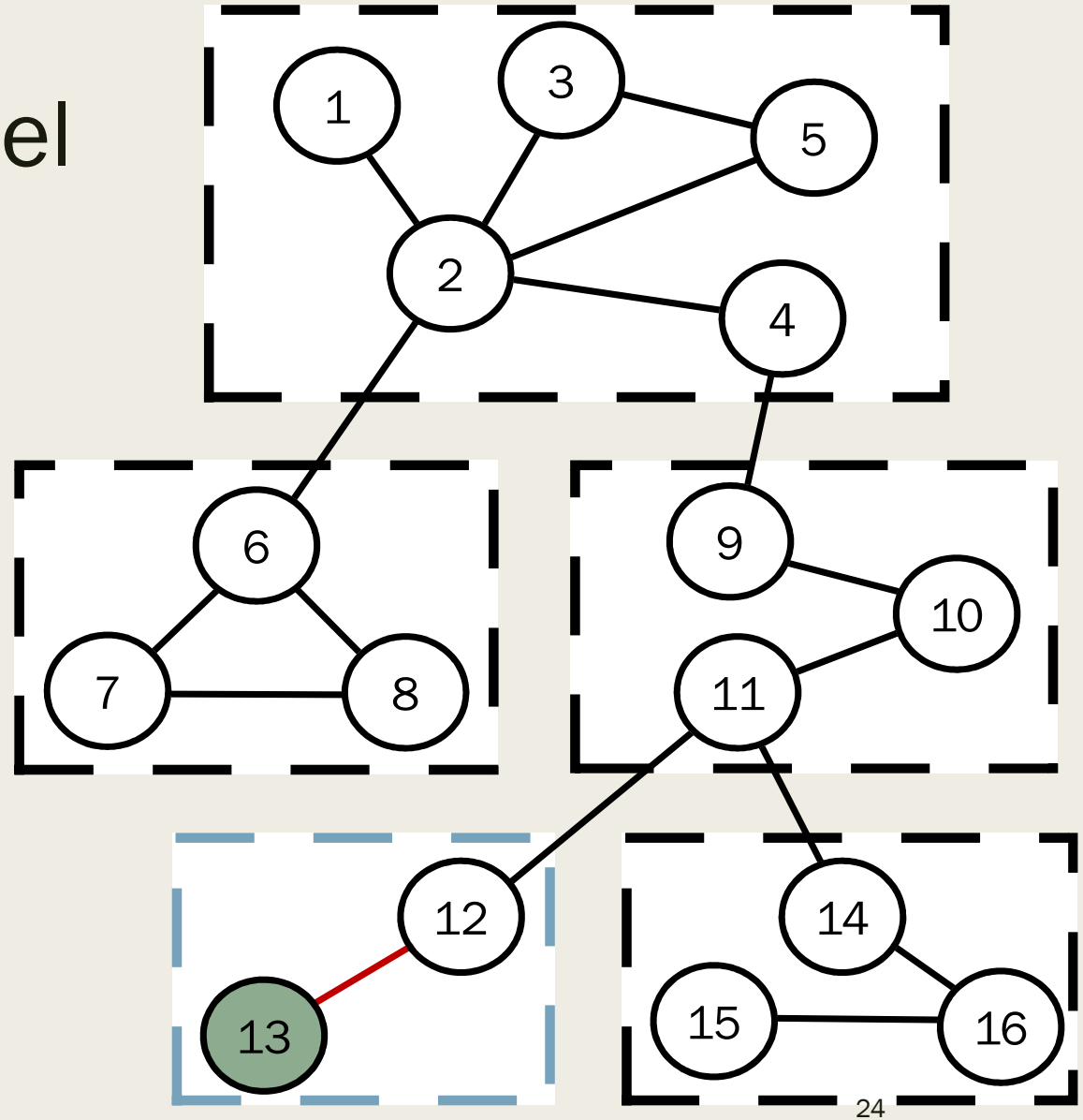
Algorithmus Beispiel

- Betrachte die einzelnen Teilgraphen:



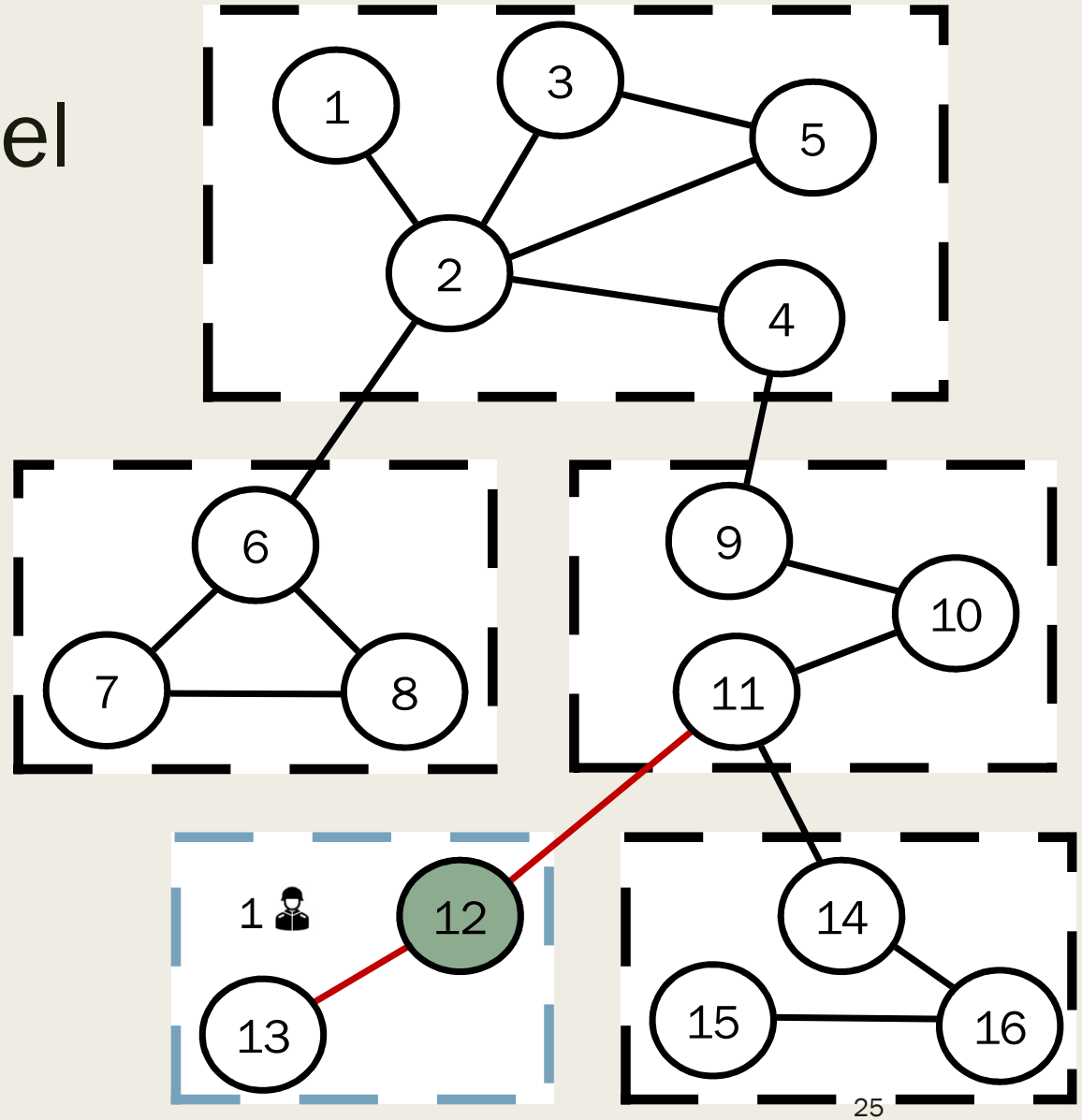
Algorithmus Beispiel

- Betrachte die einzelnen Teilgraphen:
  - {13} → Min Cover ohne Verbindungskante



Algorithmus Beispiel

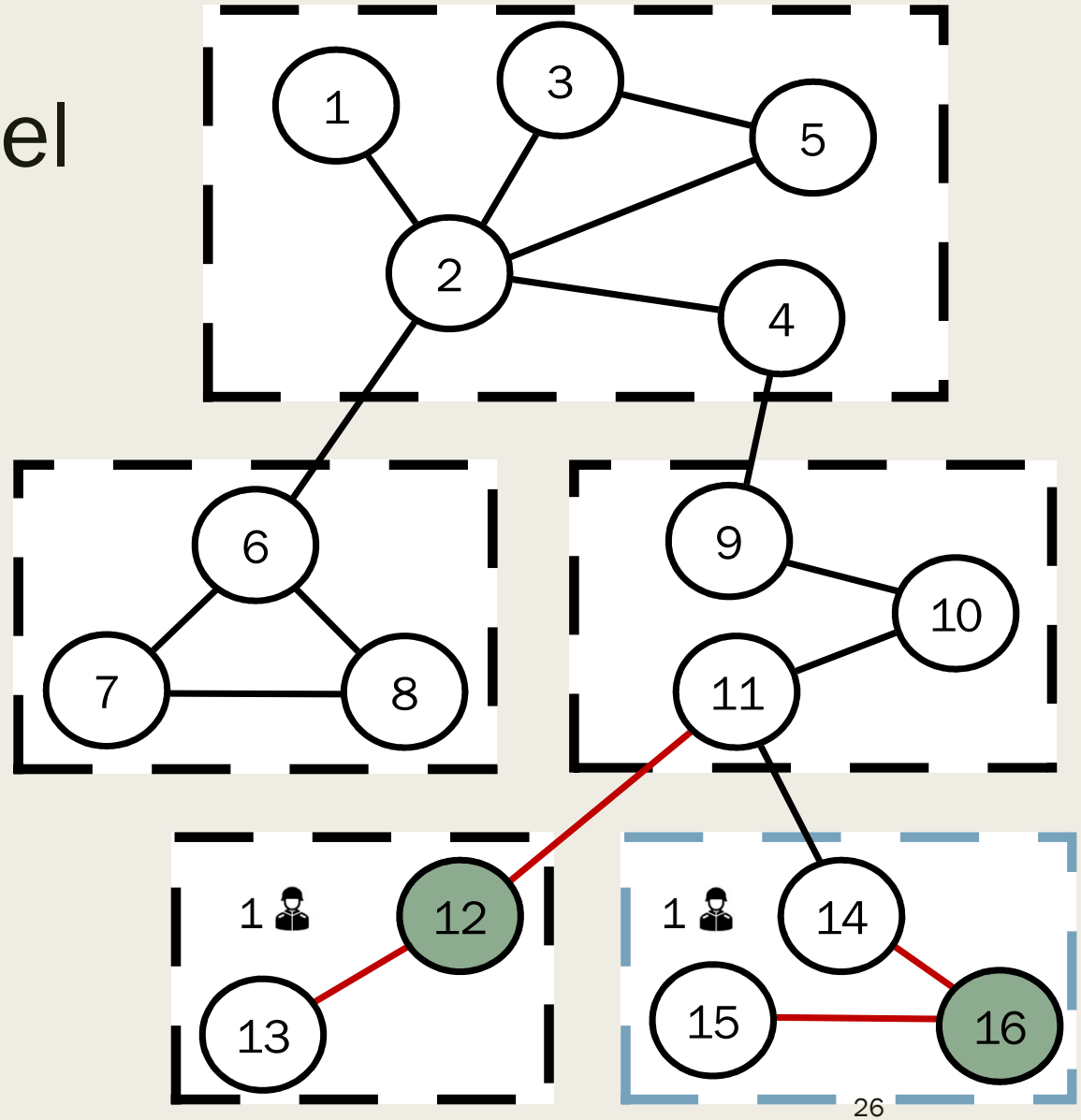
- Betrachte die einzelnen Teilgraphen:
  - {12} → Min Cover mit Verbindungskante
  - Wähle {12} als Min Cover



Anzahl 👤: 1

Algorithmus Beispiel

- Betrachte die einzelnen Teilgraphen:
  - {16} → Einziges Min Cover
  - Verbindungskante wird dem „Elterngraphen“ - hinzugefügt

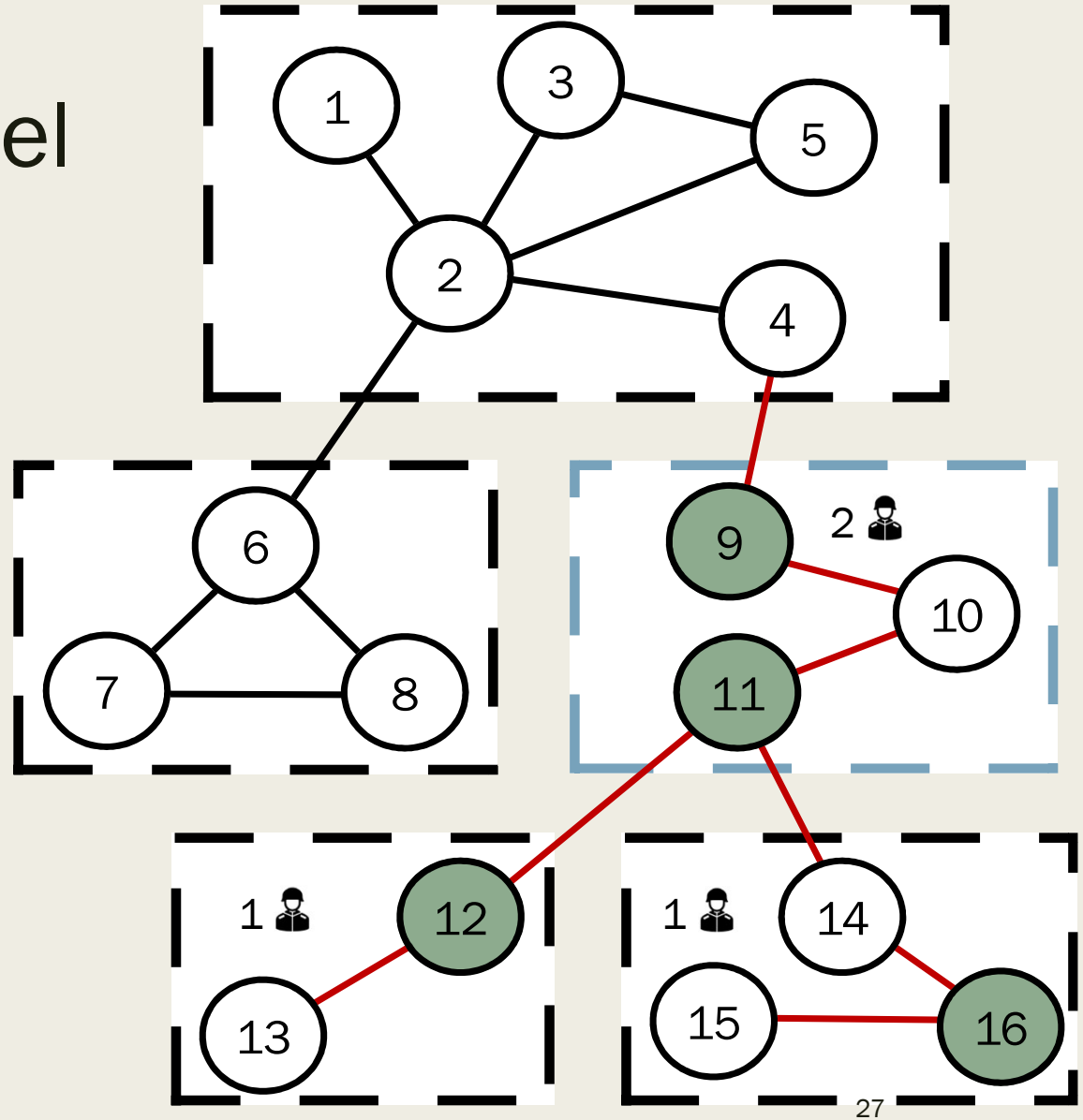


Anzahl 👤: 2

Algorithmus Beispiel

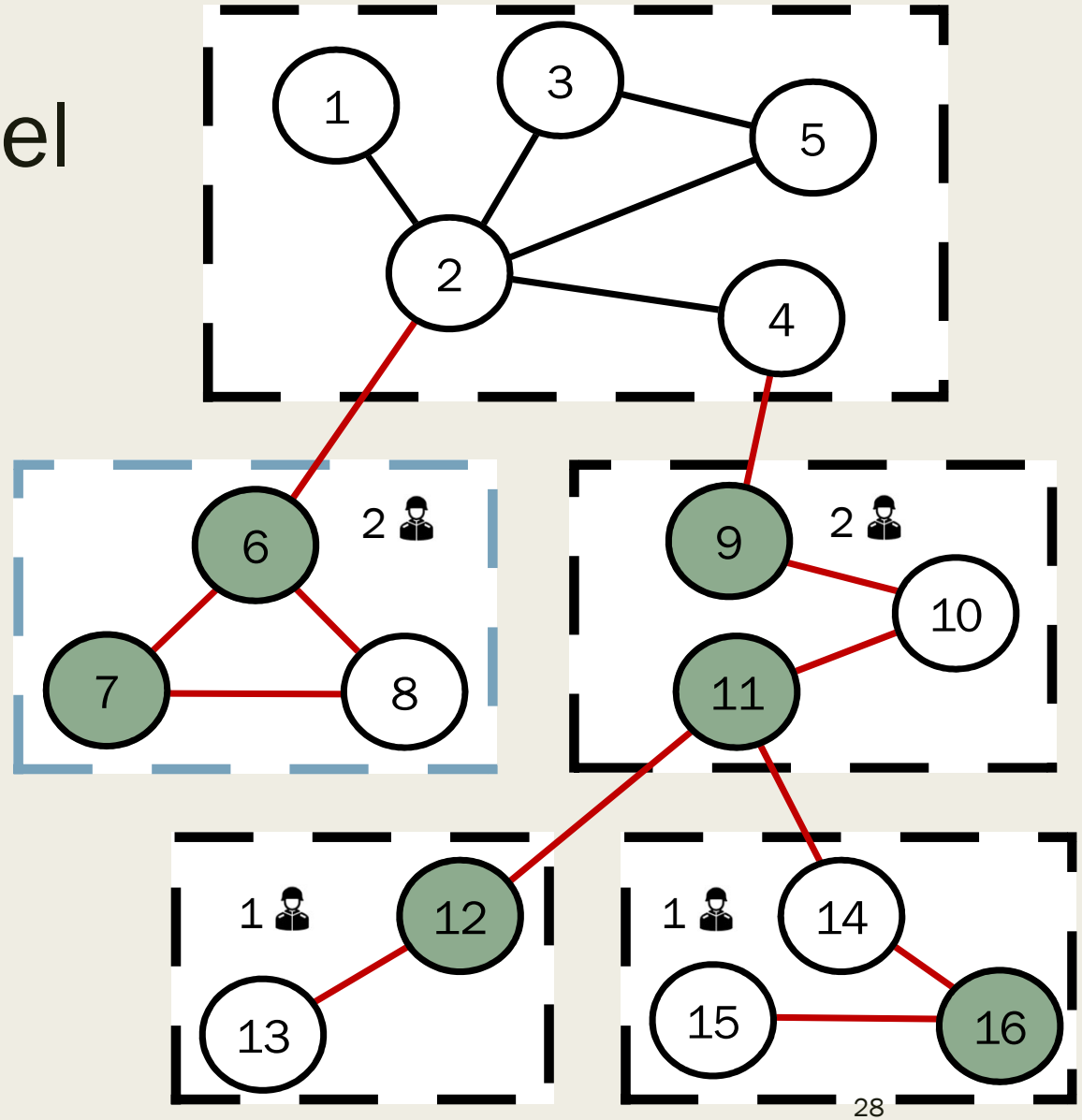
- Betrachte die einzelnen Teilgraphen:
  - Beachte: Kante (11, 14) wurde hinzugefügt
  - {9, 11} → Min Cover
  - {11, 10} → min Cover
  - Wähle {9, 11}, da es den Eingangsknoten enthält

Anzahl 👤: 4



Algorithmus Beispiel

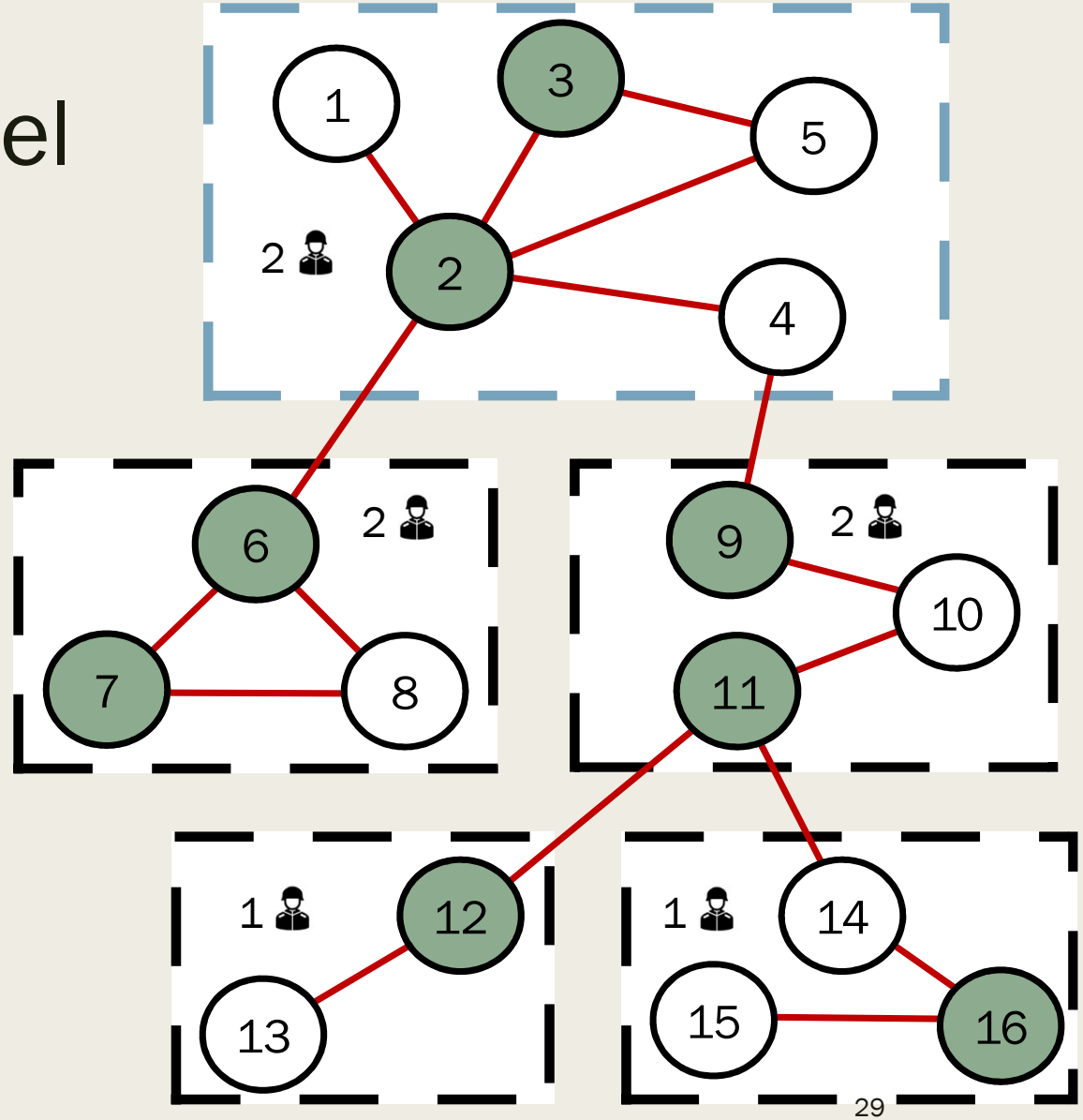
- Betrachte die einzelnen Teilgraphen:
  - {6, 7} → Min Cover
  - {6, 8} → Min Cover
  - {7, 8} → Min Cover
  - Wähle {6, 7} (oder {6, 8})



Anzahl 👤: 6

Algorithmus Beispiel

- Betrachte die einzelnen Teilgraphen:
  - {2, 3} → Min Cover
  - {2, 5} → Min Cover
  - Wähle {2, 3} (oder {2, 5})

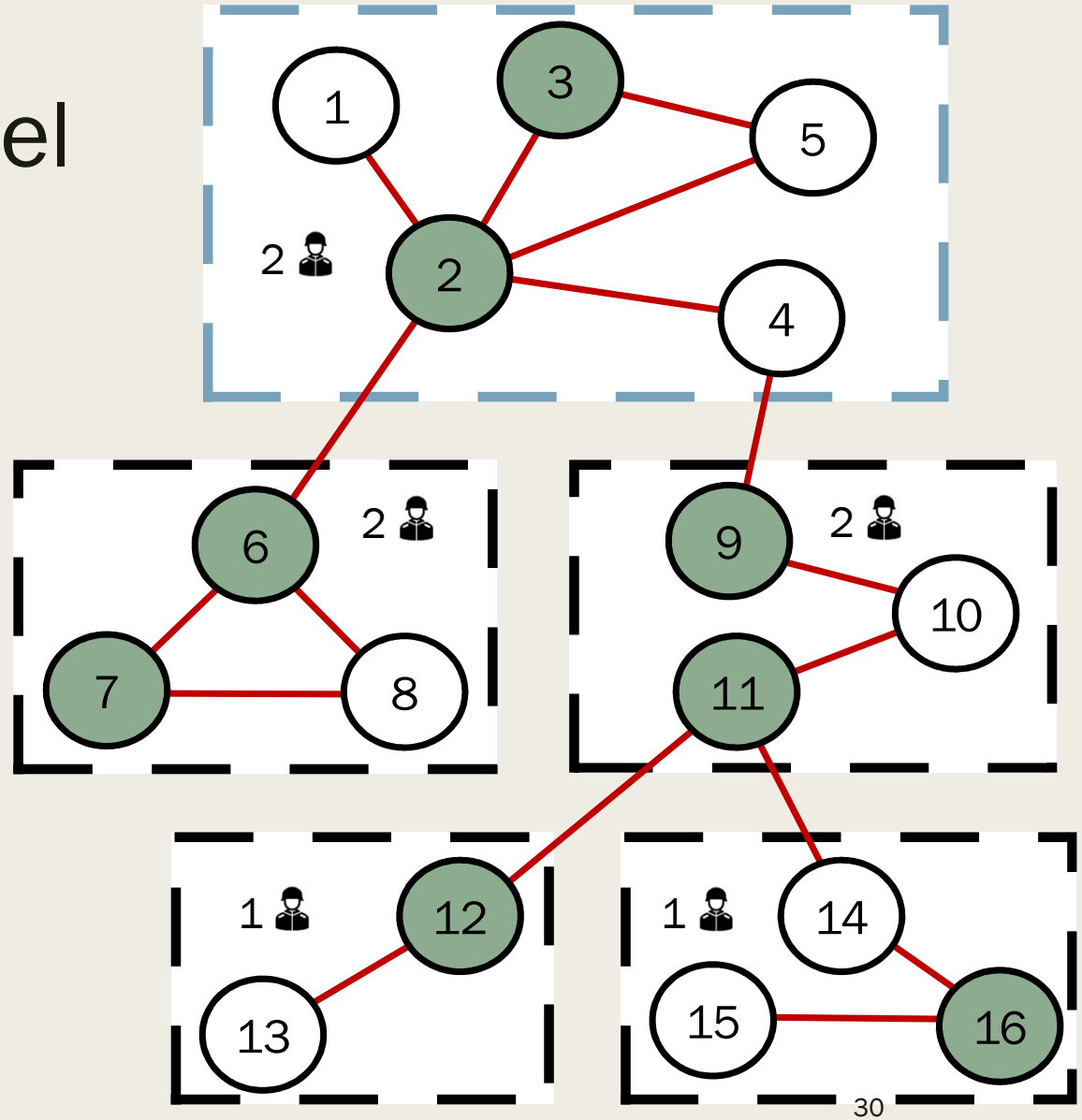


Anzahl 👤: 8

Algorithmus Beispiel

- Betrachte die einzelnen Teilgraphen:
  - {2, 3} → Min Cover
  - {2, 5} → Min Cover
  - Wähle {2, 3} (oder {2, 5})

- Output: 8



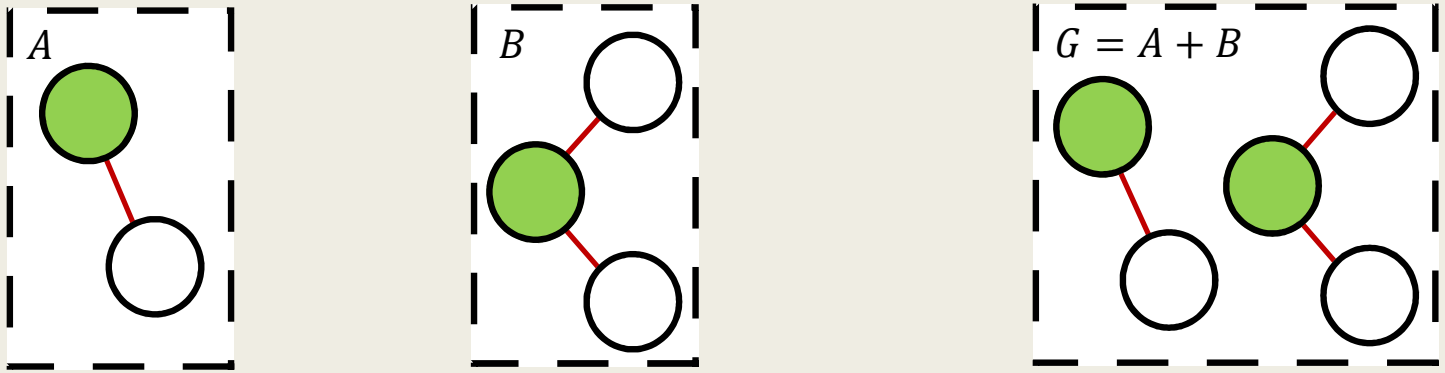
Anzahl : 8

Korrektheit

Sei $G = (V, E)$ ein Graph und $f: Graph \rightarrow \mathbb{N}$, $f(X) = |minVC(X)|$ die Funktion, die jedem Graphen die Anzahl an Knoten einer seiner minimalen Knotenüberdeckungen zuweist.

$Beob.1$: Besteht G aus zwei nicht miteinander verbundenen Teilgraphen A und B ("$G = A + B$") so gilt:

$$f(G) = f(A + B) = f(A) + f(B)$$



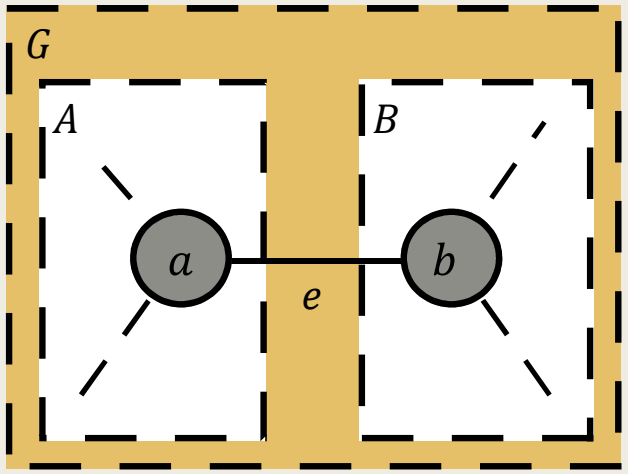
Korrektheit

$Beob.2$: Fügen wir Graph A Kante $e = \{u, v\}$ hinzu, so gilt für den resultierenden Graph $G = A + e$:

$$f(G) = f(A + e) = \begin{cases} f(A), & \text{wenn } u \text{ oder } v \text{ in einem } minVC \text{ von } G \\ f(A) + 1, & \text{sonst} \end{cases}$$

Korrektheit

Seien A und B zwei nicht miteinander verbundene Teilgraphen und e={a,b} eine Kante mit a aus A und b aus B.
Betrachten wir nun $f(G)$ für $G = (A + B) + e$:

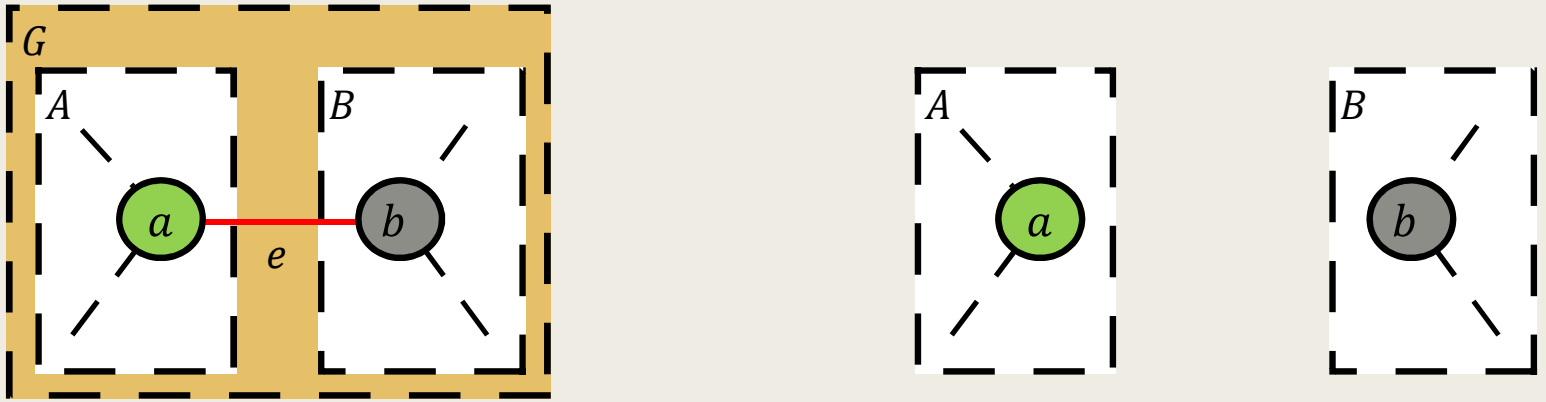


Korrektheit

Seien A und B zwei nicht miteinander verbundene Teilgraphen und e={a,b} eine Kante mit a aus A und b aus B.
Betrachten wir nun $f(G)$ für $G = (A + B) + e$:

$Folg.1a$: Liegt a in einem $minVC$ von A so gilt:

$$f(G) = f\big((A + B) + e\big) = f(A + B) = f(A) + f(B)$$

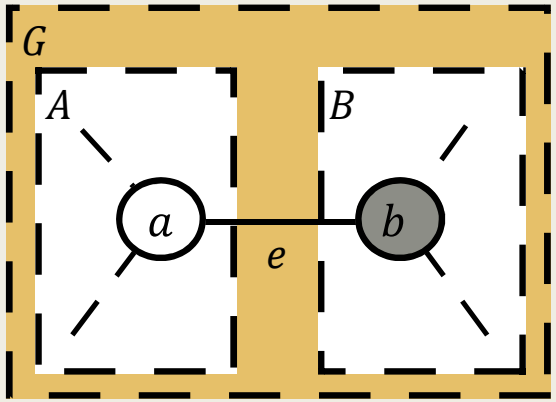


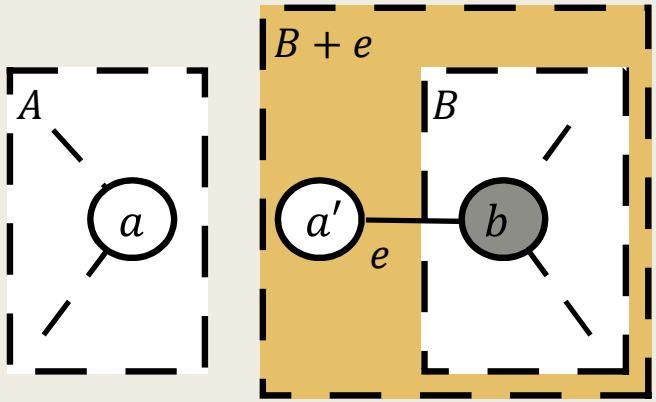
Korrektheit

Seien A und B zwei nicht miteinander verbundene Teilgraphen und e={a,b} eine Kante mit a aus A und b aus B.
Betrachten wir nun $f(G)$ für $G = (A + B) + e$:

$Folg.1b$: Liegt a in **keinem** $minVC$ von A so gilt:

$$f(G) = f\big((A + B) + e\big) = f(A) + f(B + e)$$

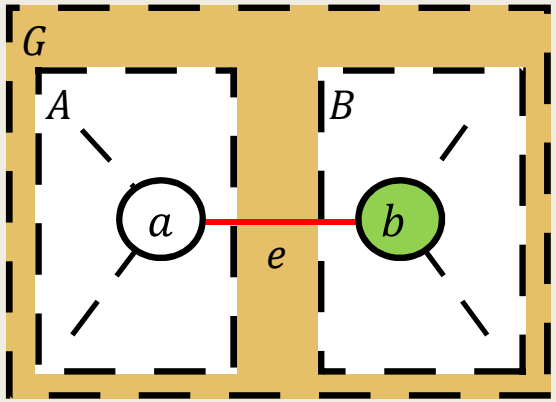


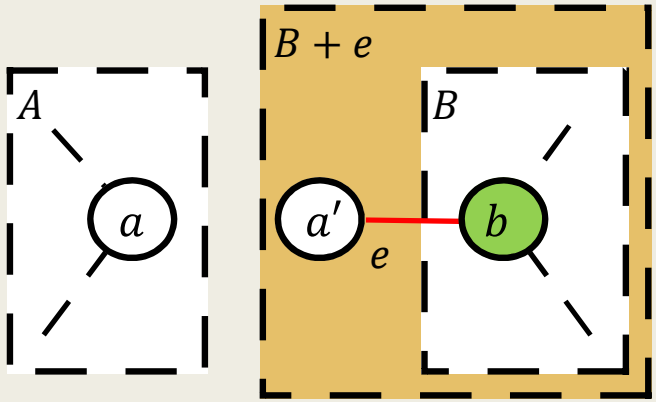


Korrektheit

a in **keinem** $minVC$ von $A \Rightarrow f(G) = f\big((A + B) + e\big) = f(A) + f(B + e)$:

- b liegt in einem $minVC$ von B:
  $$f(G) = f\big((A + B) + e\big) = f(A + B) = f(A) + f(B) = f(A) + f(B + e)$$

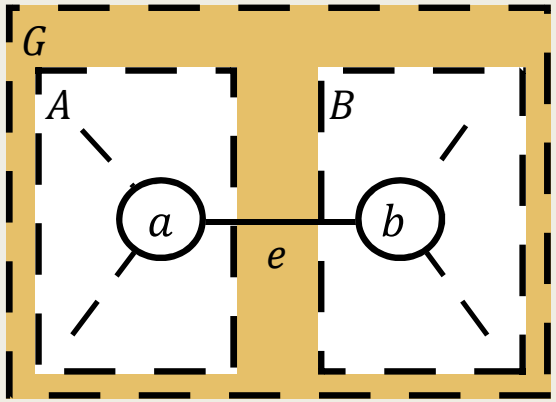


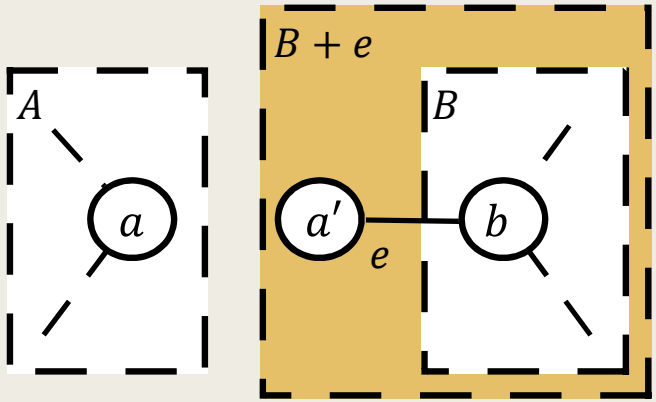


Korrektheit

a in **keinem** $minVC$ von $A \Rightarrow f(G) = f\big((A+B)+e\big) = f(A)+f(B+e)$:

- b liegt in einem $minVC$ von B:
  $$f(G) = f\big((A+B)+e\big) = f(A+B) = f(A)+f(B) = f(A)+f(B+e)$$
- b liegt in **keinem** $minVC$ von B:
  $$f(G) = f\big((A+B)+e\big) = f(A+B)+1 = \big(f(A)+f(B)\big)+1$$
  $$= f(A)+(f(B)+1) = f(A)+f(B+e)$$





Reminder Algorithmus

- Wir betrachten den Baum aus Teilgraphen und berechnen deren Lösung isoliert. Dabei gehen wir von den Blättern zur Wurzel hin vor.

Berechnen der Knotenüberdeckung für einen Teilgraphen:

Voraussetzung: Die Knotenüberdeckung der „Kinder“ ist schon berechnet.

1. Füge nicht überdeckte Verbindungskanten der Kinder zu internen Kantenmenge hinzu
2. Finde per Bruteforce minimale Überdeckungen für die neue Kantenmenge
3. Enthält eine dieser Überdeckungen den Eingangsknoten, so notiere die eingehende Verbindungskante als überdeckt

Korrektheit

Sei G der Graph einer (Unter−)Burg und M der Graph des Hauptgebäudes der Burg.
Seien $K_1 \dots K_n$ die Graphen der "Kindgebäude" von M, $v_1 \dots v_n$ deren Verbindungskanten zu M und K_i^{sub} jeweils der Graph der "Unterburg" von K_i.
Sei weiterhin M' der um die unbedeckten Kanten seiner Kind−Teilgraphen erweiterte Graph M.
Wir nennen $b(X)$ eine (per Bruteforce gewonnene) minimale Knotenüberdeckung von X, die wenn mgl. die Elternkante überdeckt und definieren:

$$u(M) \coloneqq \bigcup_{i=1\dots n} u(K_i) + b(M')$$

Korrektheit

Wir wollen zeigen, dass $u(M)$ eine minimale Knotenüberdeckung von G ist. Da u dem Vorgehen unseres Algorithmus$'$ entspricht (berechnet $|u(M)|$) liefert dieser somit auch die minimale Anzahl an Wachen.

Zu zeigen:
1. $u(M)$ ist Knotenüberdeckung
2. $u(M)$ ist minimal

Korrektheit

$$u(M) \coloneqq \bigcup_{i=1\ldots n} u(K_i) + b(M')$$

$u(M)$ist Knotenüberdeckung:
Sei $e = (a, b)$ beliebige Kante.
Liegt e in M, dann auch in M‘, also wird von $b(M$‘$)$ überdeckt.
Ist e eine Verbindungskante von M, so überdeckt sie entweder eines der $u(K_i)$ oder e wurde zu M‘ hinzugefügt und wird von $b(M$‘$)$ überdeckt.
Sonst überdeckt e eines der $u(K_i)$.

$\Rightarrow u(M)$ überdeckt jede Kante aus G.

Korrektheit

$$u(M) \coloneqq \bigcup_{i=1\dots n} u(K_i) + b(M')$$

$u(M)$ ist minimal:
Wir betrachten G_i, wobei G_i aus M und allen K_j^{sub} und v_j für $j \leq i$ besteht und berechnen:

$$f(G_i) = \left(f\left(G_{i-1} + K_i^{sub} + v_i\right)\right) = f\left(K_i^{sub}\right) + f(G_{i-1}(+v_i)^*)$$

somit gilt:

$$f(G) = f(G_n) = \sum_{i=1\dots n} f(K_i^{sub}) + f(G_0(=M)(+v_1)^* \dots (+v_n)^* =$$

$$= \sum_{i=1\dots n} |u(K_i)| + |b(M')| = |u(M)|$$

Laufzeitanalyse

Sei B eine Burg mit g Gebäuden und n Hallen. Wir bezeichnen $d(X)$ als die Dauer der Berechnung der Wachenzahl für Komponente X.

- Bestimmen der Wachenzahl eines Gebäudes G:
  Berechnungsdauer abhängig von der Anzahl der Hallen n, Korridore m und angehängten Gebäuden w. Jeder dieser Faktoren ist durch eine feste Ganzzahl beschränkt ($2 \leq n \leq 10{,}1 \leq m \leq 45{,}0 \leq w \leq 10$).

$$\Rightarrow d_{max} \coloneqq \max_{X \epsilon \{G | G\ ist\ Gebäude\}} d(X) \text{ ist Konstante}$$

$$\Rightarrow d(G) \leq d_{max} \in O(1)$$

Laufzeitanalyse

Sei B eine Burg mit g Gebäuden und n Hallen. Wir bezeichnen $d(X)$ als die Dauer der Berechnung der Wachenzahl für Komponente X.

- Pro Gebäude G: $d(G) \leq d_{max} \in O(1)$
- Anzahl der Gebäude:
  Pro Gebäude zwischen 2 und 10 Hallen.

$$\Rightarrow {}^{n}/_{10} \leq g \leq {}^{n}/_{2}$$

$$\Rightarrow g \leq {}^{n}/_{2}$$

Laufzeitanalyse

Sei B eine Burg mit g Gebäuden und n Hallen. Wir bezeichnen $d(X)$ als die Dauer der Berechnung der Wachenzahl für Komponente X.

- Pro Gebäude G: $d(G) \leq d_{max} \in O(1)$
- Anzahl der Gebäude: $g \leq {}^{n}/_{2}$
- Bestimmen der Wachenzahl der gesamten Burg:

$$d(B) = \sum d(G_i) \leq g \cdot d_{max} \leq {}^{n}/_{2} \cdot d_{max} \in O(n) \cdot O(1)$$

$$\Rightarrow d(B) \in O(n)$$