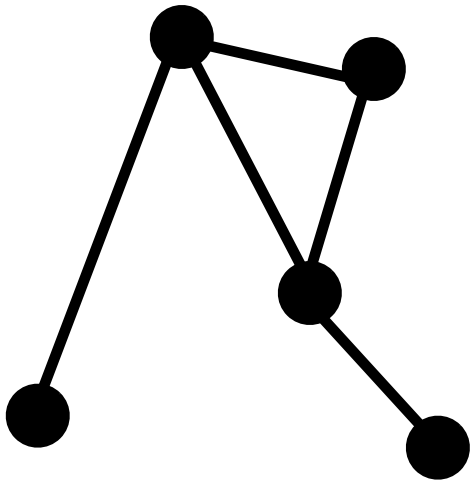
Kapitel 8
Graphenalgorithmen

- Minimaler Spannbaum
- Union-find-Problem
- Kürzeste Wege

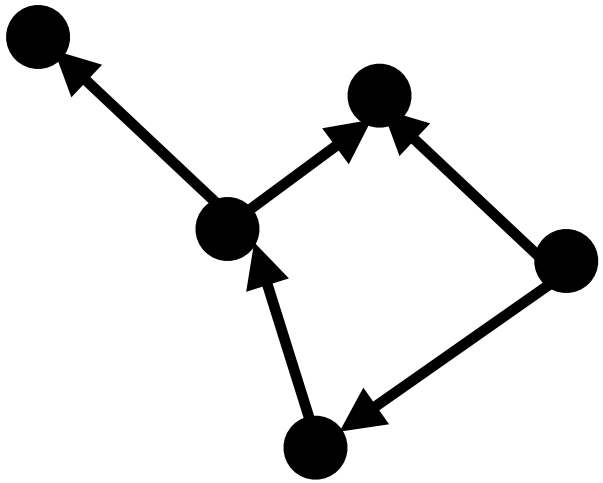
Graphenalgorithmen
Rückblick

- Graphen
  - Modelle für Objekte und Beziehungen untereinander
    - Personen - Bekanntschaften
    - Bahnhöfe - Zugverbindungen
    - Ereignisse - Abhängigkeiten
    - Schachstellungen - Erreichbarkeit

Graphenalgorithmen
Rückblick



ungerichteter Graph



gerichteter Graph

Graphenalgorithmen
Minimaler Spannbaum

- Anwendung: Bestimmung möglichst kostengünstiger Verkehrs- oder Versorgungsnetze

Graphenalgorithmen
Minimaler Spannbaum - Definitionen

Definition

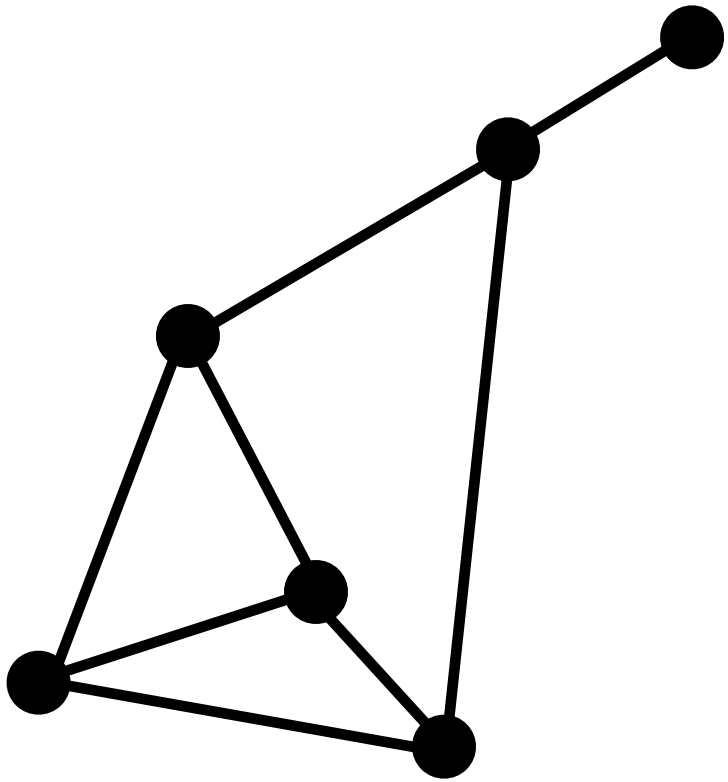
Sei G=(V,E,c) ein ungerichteter zusammenhängender Graph mit Kantenbewertung, also (V,E) ungerichteter zusammenhängender Graph und c: $E \rightarrow \mathbb{N}$ eine Kostenfunktion. Ein **Spannbaum** B für G ist ein Baum, der alle Knoten von G enthält, zusammenhängend ist und dessen Kantenmenge $E' \subseteq E$ ist. Die **Kosten** c(B) des Spannbaums sind die Summe der Kosten seiner Kanten.

Genauer: B=(V,E',c) ist Spannbaum für G=(V,E,c), falls $E' \subseteq E$ und B zusammenhängend ist: $c(B)=\sum_{e \in E'} c(e)$.

Gesucht: Spannbaum zu G mit minimalen Kosten.

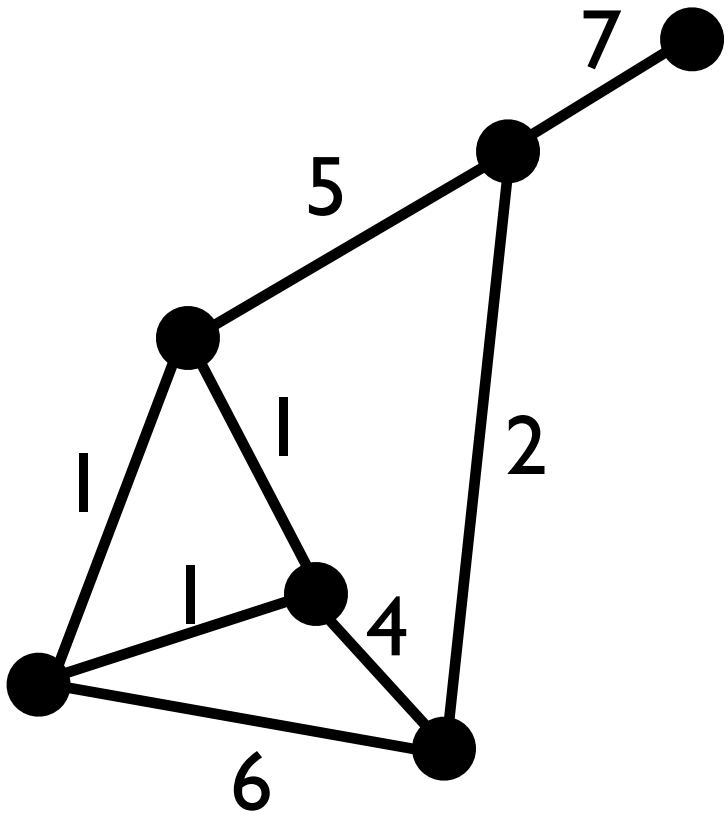
Graphenalgorithmen

Minimaler Spannbaum - Beispiel



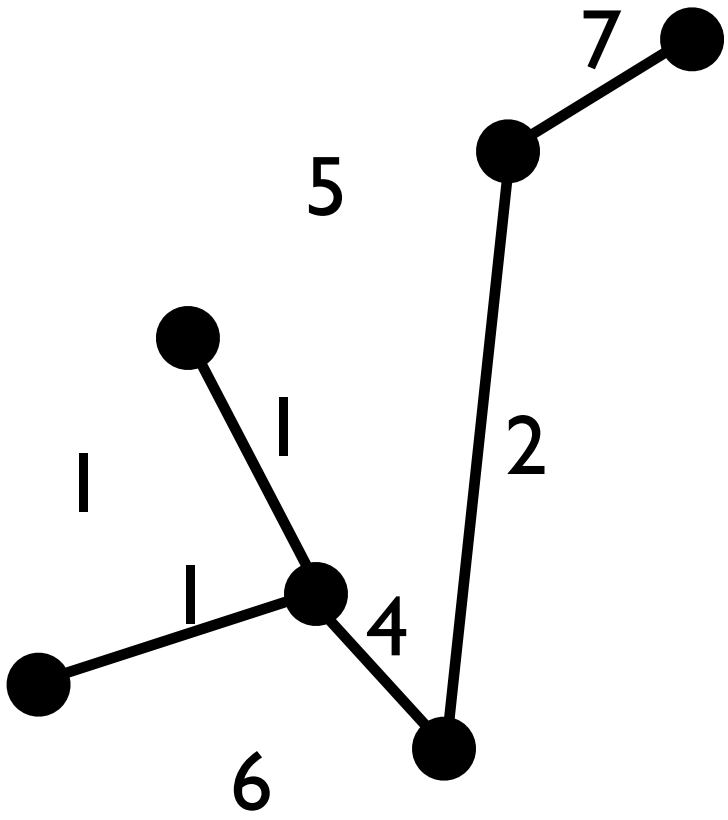
Graphenalgorithmen

Minimaler Spannbaum - Beispiel



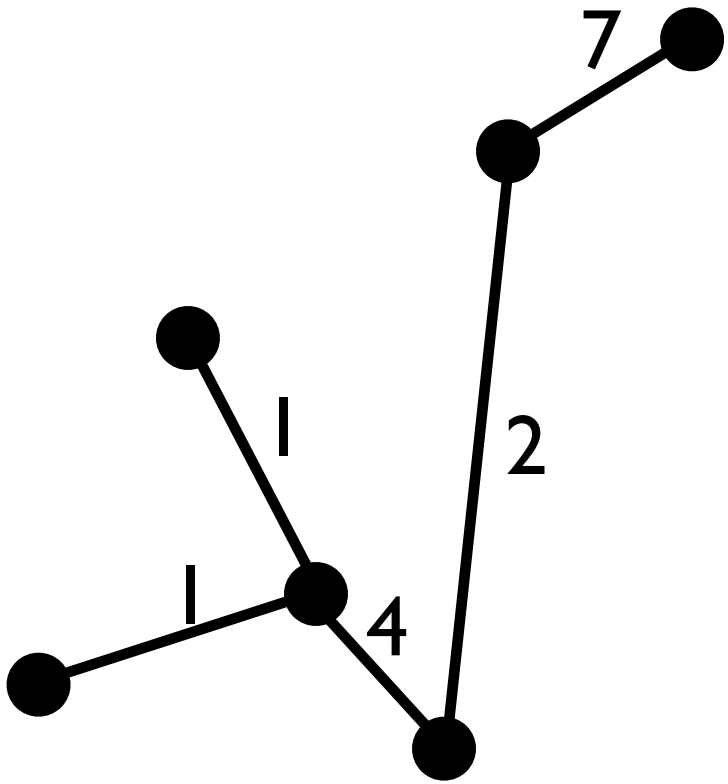
Graphenalgorithmen

Minimaler Spannbaum - Beispiel



Graphenalgorithmen

Minimaler Spannbaum - Beispiel



Kosten: 1+1+4+2+7=15

Graphenalgorithmen
Minimaler Spannbaum - Vorgehen

- Zentrales Algorithmenmuster: **greedy-Verfahren** (greedy=gierig):
  - Wähle Lösungsschritte so, daß sie sich mit jedem Schritt der Lösung näherbringen und nimm nie einen Schritt zurück.
  - Hier: ein Rechenschritt=Wahl einer Kante und Hinzufügen zum Spannbaum(-fragment)
  - Kante wird nie wieder entfernt
  - Sobald Baum Spannbaum, fertig.

Graphenalgorithmen
Minimaler Spannbaum - Korrektheit

Korrektheit des Verfahrens beruht auf

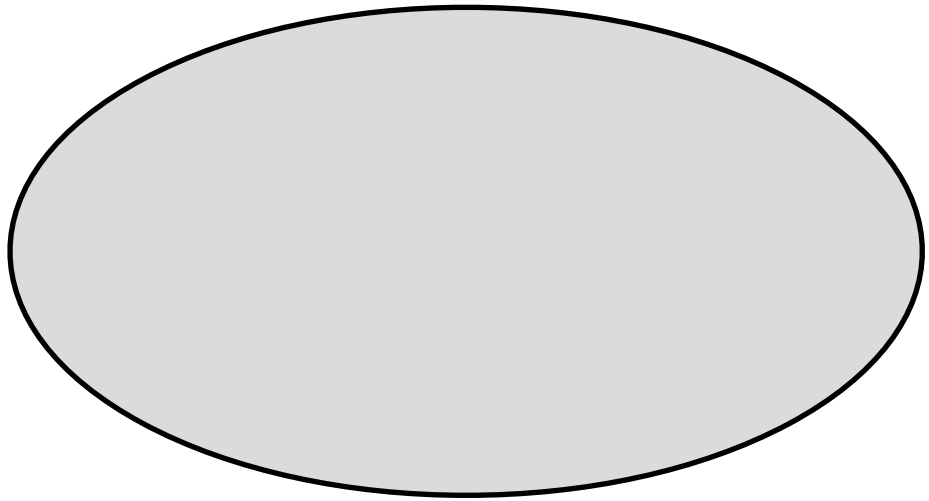
Lemma

Sei G=(V,E) ein Graph und {U,W} eine Zerlegung der Knotenmenge V. Sei {u,w} eine Kante in G mit minimalen Kosten unter allen Kanten {{u',w'} | u'∈U, w'∈W}. Dann gibt es einen minimalen Spannbaum für G, der {u,w} enthält.

Graphenalgorithmen
Minimaler Spannbaum - Korrektheit

Lemma

Sei G=(V,E) ein Graph und {U,W} eine Zerlegung der Knotenmenge V. Sei {u,w} eine Kante in G mit minimalen Kosten unter allen Kanten {{u',w'} | u'∈U, w'∈W}.

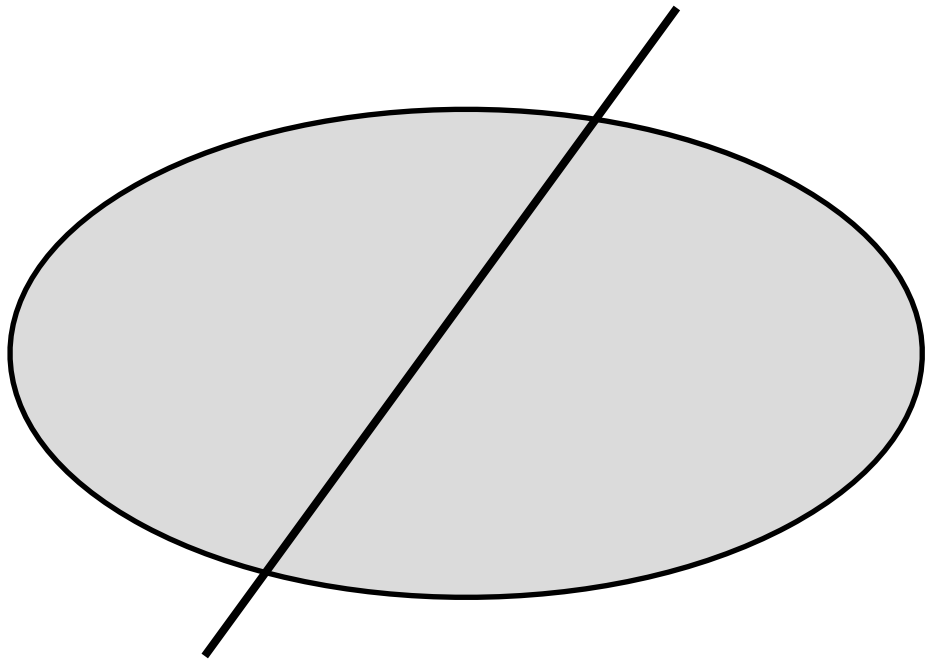


Dann gibt es einen minimalen Spannbaum für G, der {u,w} enthält.

Graphenalgorithmen
Minimaler Spannbaum - Korrektheit

Lemma

Sei G=(V,E) ein Graph und {U,W} eine Zerlegung der Knotenmenge V. Sei {u,w} eine Kante in G mit minimalen Kosten unter allen Kanten {{u',w'} | u'∈U, w'∈W}.

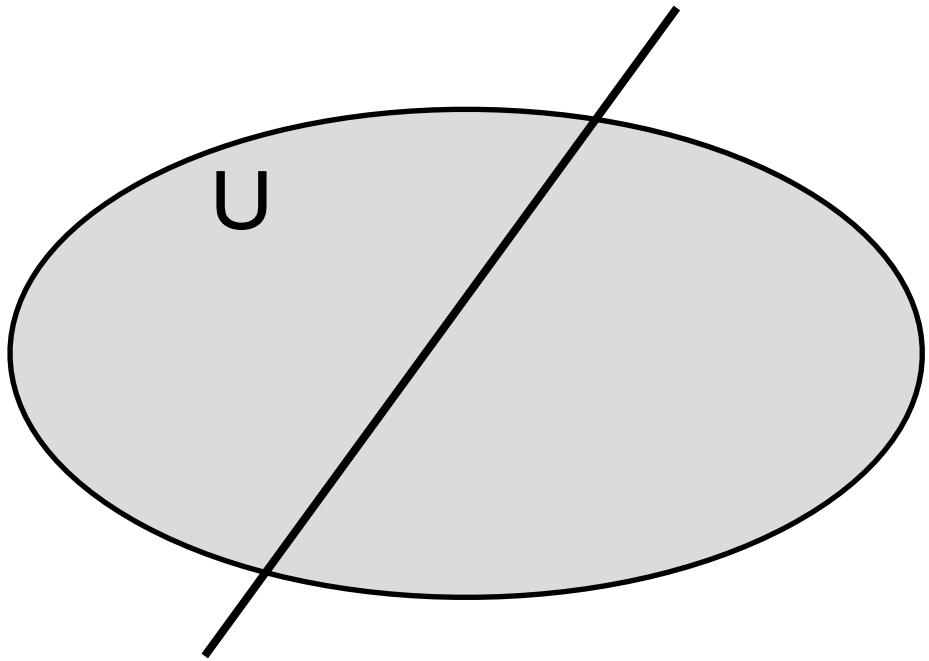


Dann gibt es einen minimalen Spannbaum für G, der {u,w} enthält.

Graphenalgorithmen
Minimaler Spannbaum - Korrektheit

Lemma

Sei G=(V,E) ein Graph und {U,W} eine Zerlegung der Knotenmenge V. Sei {u,w} eine Kante in G mit minimalen Kosten unter allen Kanten {{u',w'} | u'∈U, w'∈W}.



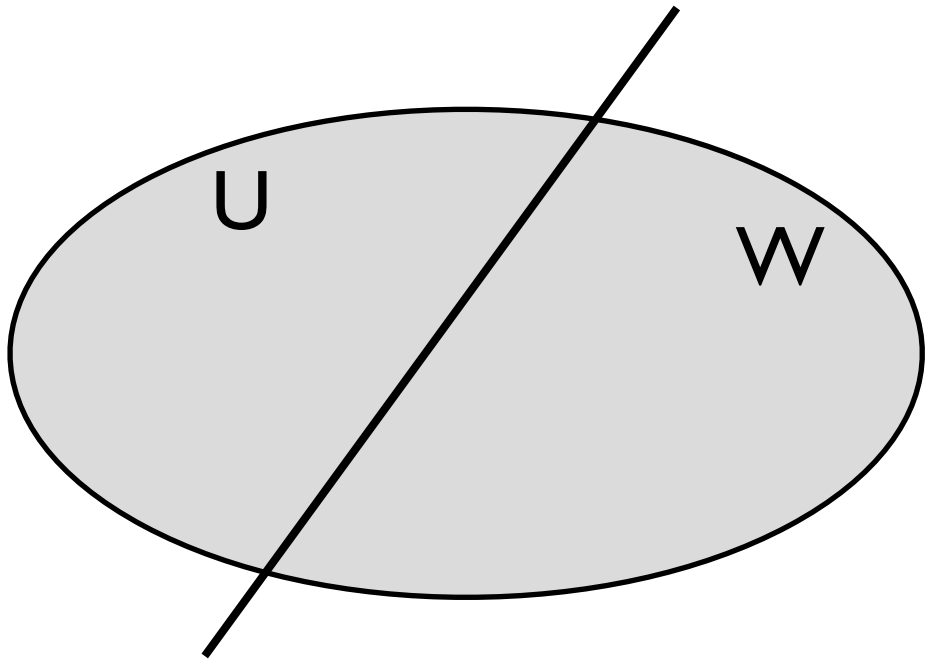
Dann gibt es einen minimalen Spannbaum für G, der {u,w} enthält.

Graphenalgorithmen

Minimaler Spannbaum - Korrektheit

Lemma

Sei G=(V,E) ein Graph und {U,W} eine Zerlegung der Knotenmenge V. Sei {u,w} eine Kante in G mit minimalen Kosten unter allen Kanten {{u',w'} | u'∈U, w'∈W}.



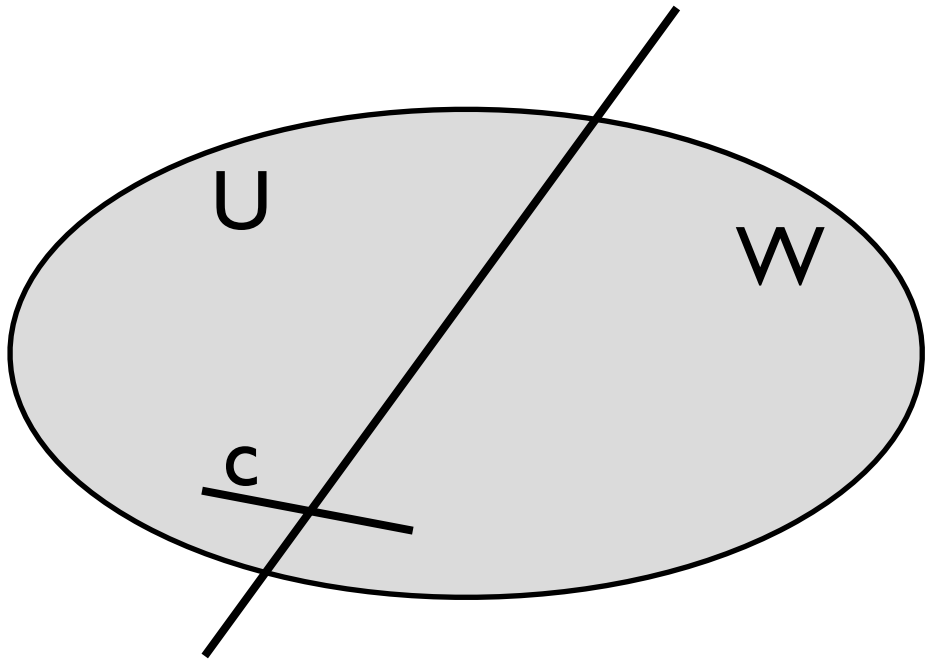
Dann gibt es einen minimalen Spannbaum für G, der {u,w} enthält.

Graphenalgorithmen

Minimaler Spannbaum - Korrektheit

Lemma

Sei G=(V,E) ein Graph und {U,W} eine Zerlegung der Knotenmenge V. Sei {u,w} eine Kante in G mit minimalen Kosten unter allen Kanten {{u',w'} | u'∈U, w'∈W}.

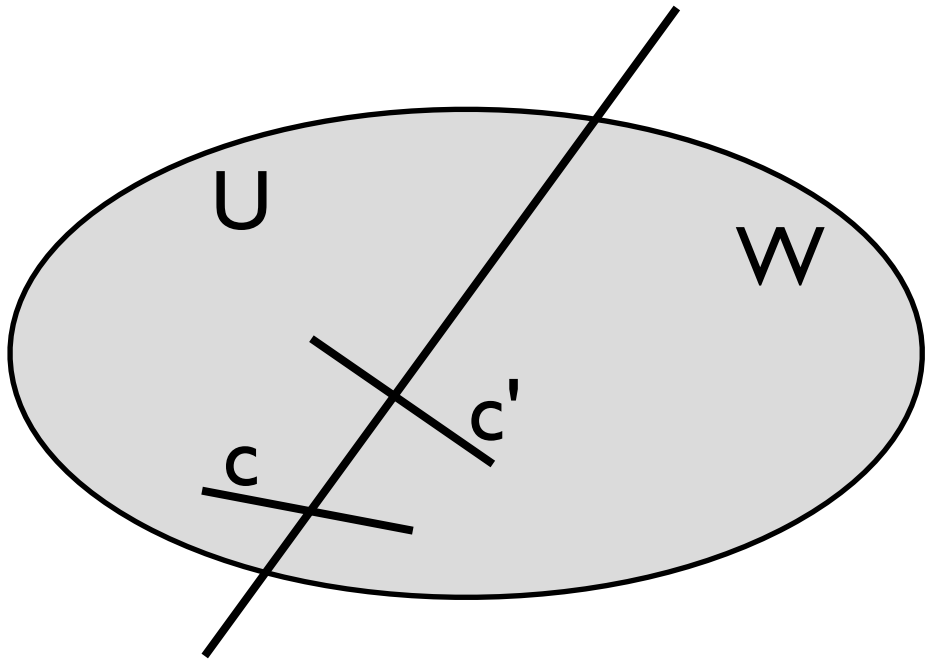


Dann gibt es einen minimalen Spannbaum für G, der {u,w} enthält.

Graphenalgorithmen
Minimaler Spannbaum - Korrektheit

Lemma

Sei G=(V,E) ein Graph und {U,W} eine Zerlegung der Knotenmenge V. Sei {u,w} eine Kante in G mit minimalen Kosten unter allen Kanten {{u',w'} | u'∈U, w'∈W}.



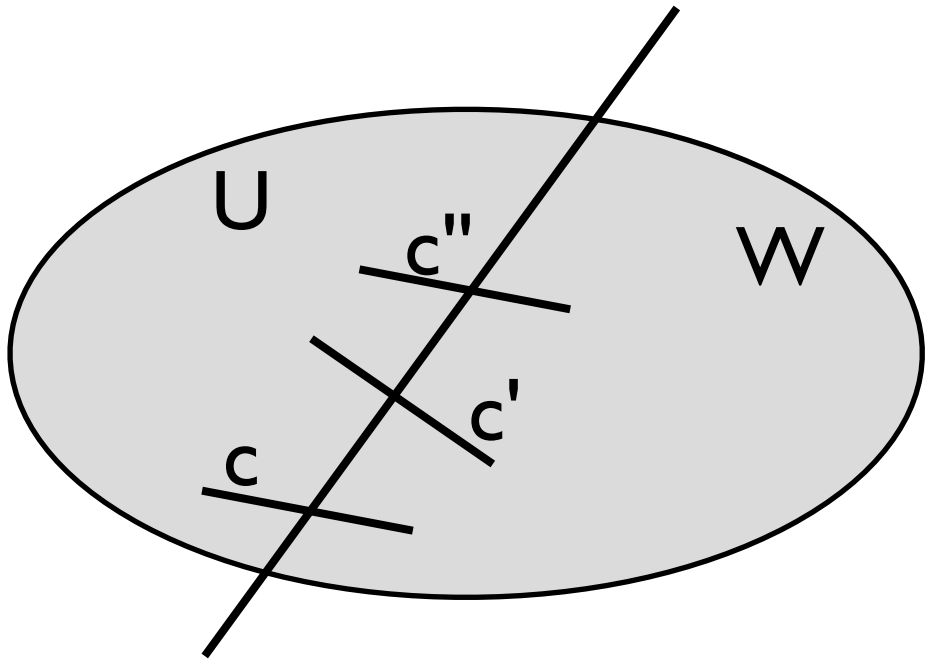
Dann gibt es einen minimalen Spannbaum für G, der {u,w} enthält.

Graphenalgorithmen

Minimaler Spannbaum - Korrektheit

Lemma

Sei G=(V,E) ein Graph und {U,W} eine Zerlegung der Knotenmenge V. Sei {u,w} eine Kante in G mit minimalen Kosten unter allen Kanten {{u',w'} | u'∈U, w'∈W}.

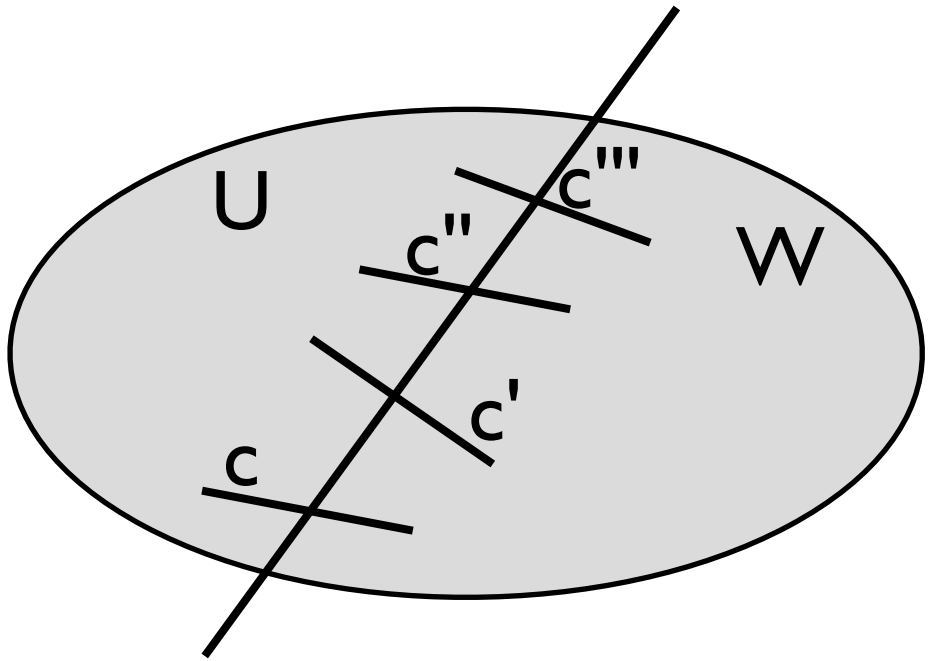


Dann gibt es einen minimalen Spannbaum für G, der {u,w} enthält.

Graphenalgorithmen
Minimaler Spannbaum - Korrektheit

Lemma

Sei G=(V,E) ein Graph und {U,W} eine Zerlegung der Knotenmenge V. Sei {u,w} eine Kante in G mit minimalen Kosten unter allen Kanten {{u',w'} | u'∈U, w'∈W}.

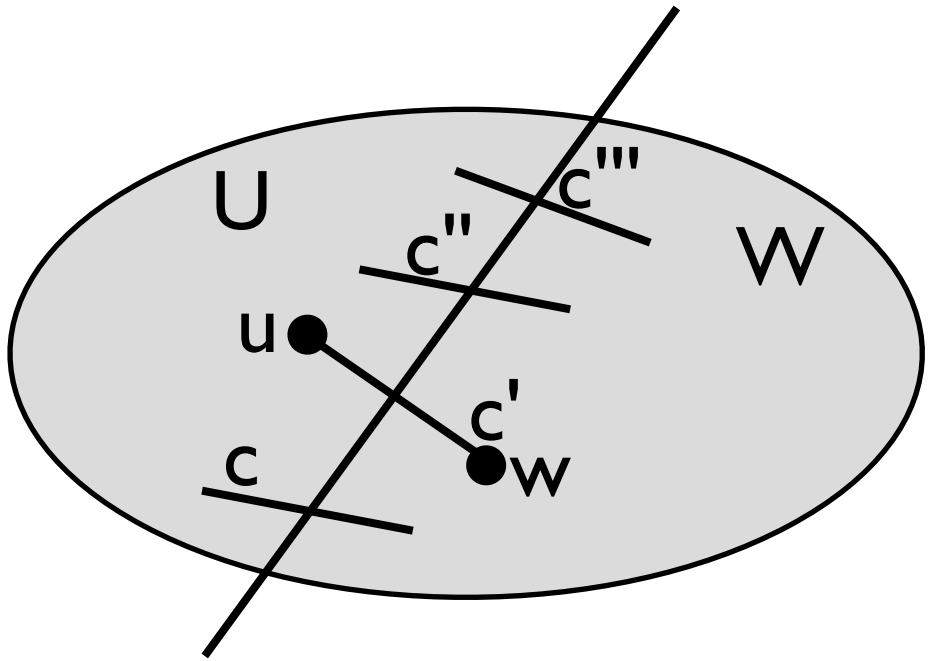


Dann gibt es einen minimalen Spannbaum für G, der {u,w} enthält.

Graphenalgorithmen
Minimaler Spannbaum - Korrektheit

Lemma

Sei G=(V,E) ein Graph und {U,W} eine Zerlegung der Knotenmenge V. Sei {u,w} eine Kante in G mit minimalen Kosten unter allen Kanten {{u',w'} | u'∈U, w'∈W}.

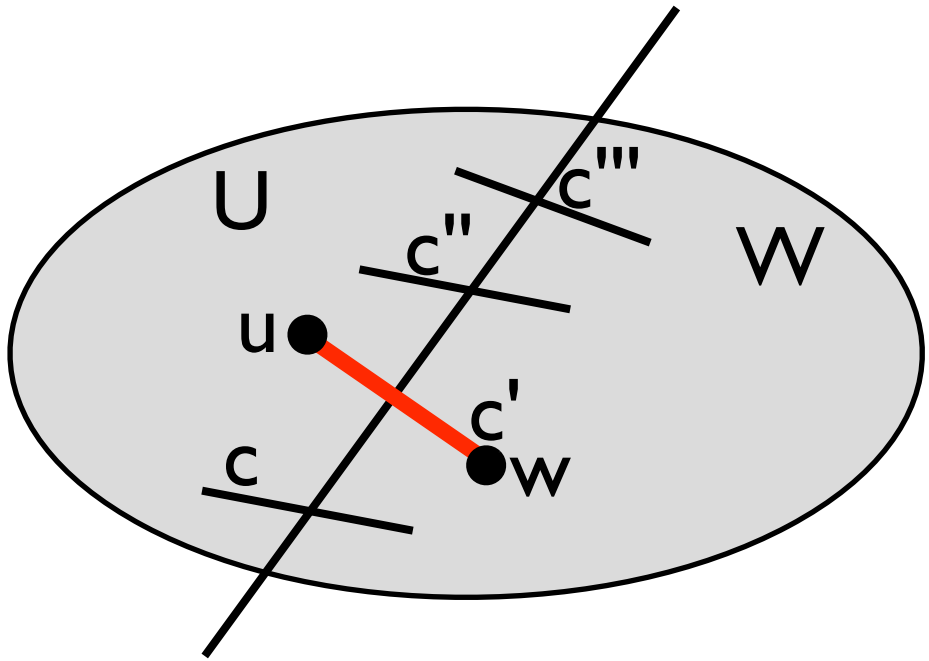


Dann gibt es einen minimalen Spannbaum für G, der {u,w} enthält.

Graphenalgorithmen
Minimaler Spannbaum - Korrektheit

Lemma

Sei G=(V,E) ein Graph und {U,W} eine Zerlegung der Knotenmenge V. Sei {u,w} eine Kante in G mit minimalen Kosten unter allen Kanten {{u',w'} | u'∈U, w'∈W}.

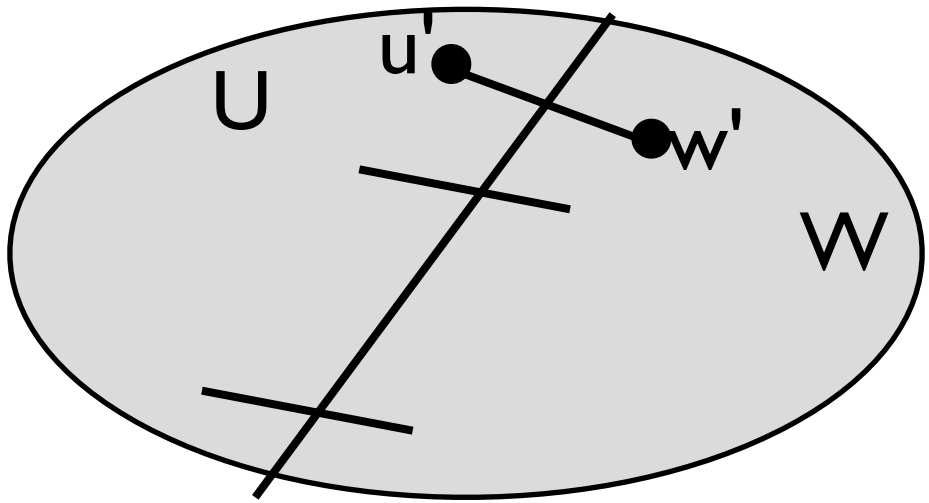


Dann gibt es einen minimalen Spannbaum für G, der {u,w} enthält.

Graphenalgorithmen
Minimaler Spannbaum - Korrektheit

Beweis: Seien U, W, {u,w} wie im Lemma und T=(V,$\bar{E}$) ein minimaler Spannbaum für G, der {u,w} *nicht* enthält. Dann gibt es mindestens eine Kante, die U mit W verbindet.

Fügt man {u,w} hinzu, so entsteht ein Graph T' mit einem Zyklus, der über {u,w} läuft.

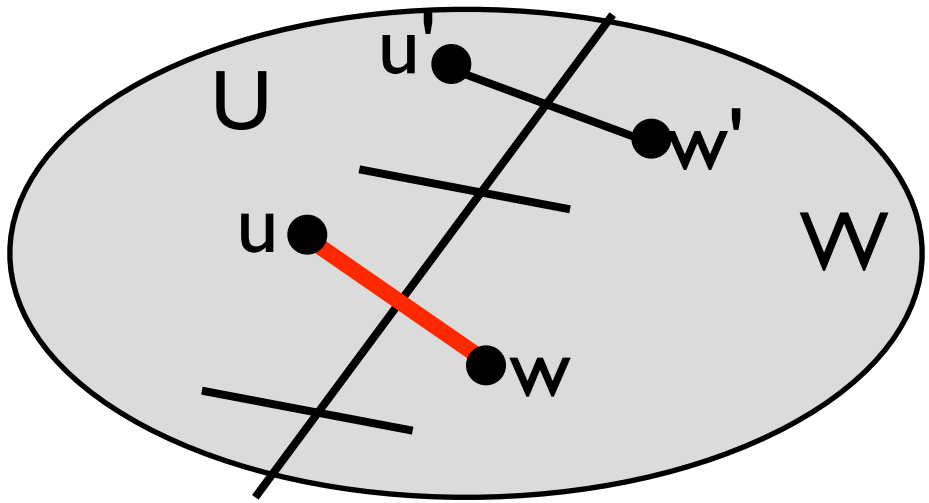


Graphenalgorithmen

Minimaler Spannbaum - Korrektheit

Beweis: Seien U, W, {u,w} wie im Lemma und T=(V,$\bar{E}$) ein minimaler Spannbaum für G, der {u,w} *nicht* enthält. Dann gibt es mindestens eine Kante, die U mit W verbindet.

Fügt man {u,w} hinzu, so entsteht ein Graph T' mit einem Zyklus, der über {u,w} läuft.

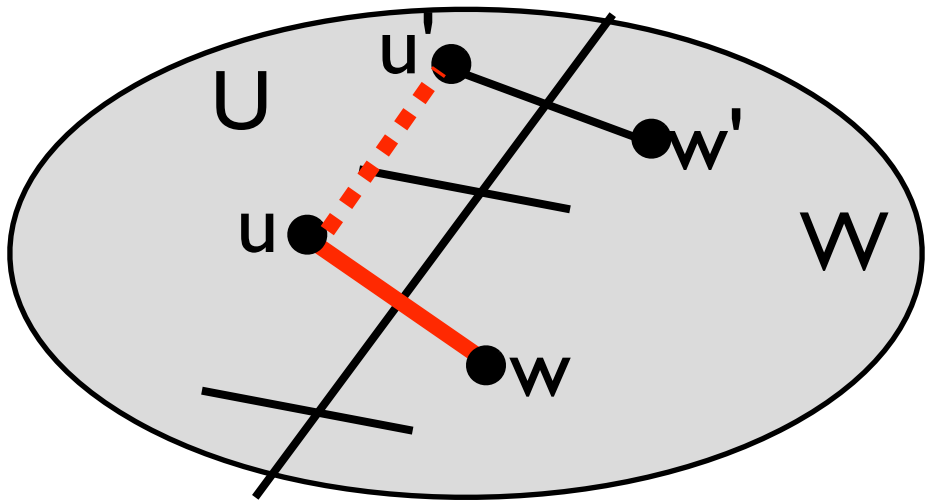


Graphenalgorithmen

Minimaler Spannbaum - Korrektheit

Beweis: Seien U, W, {u,w} wie im Lemma und T=(V,$\bar{E}$) ein minimaler Spannbaum für G, der {u,w} *nicht* enthält. Dann gibt es mindestens eine Kante, die U mit W verbindet.

Fügt man {u,w} hinzu, so entsteht ein Graph T' mit einem Zyklus, der über {u,w} läuft.

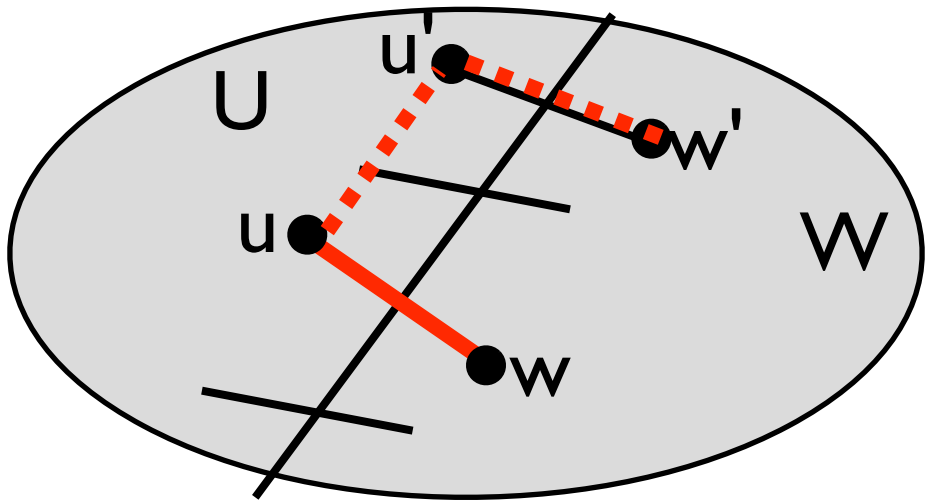


Graphenalgorithmen

Minimaler Spannbaum - Korrektheit

Beweis: Seien U, W, {u,w} wie im Lemma und T=(V,$\bar{E}$) ein minimaler Spannbaum für G, der {u,w} *nicht* enthält. Dann gibt es mindestens eine Kante, die U mit W verbindet.

Fügt man {u,w} hinzu, so entsteht ein Graph T' mit einem Zyklus, der über {u,w} läuft.

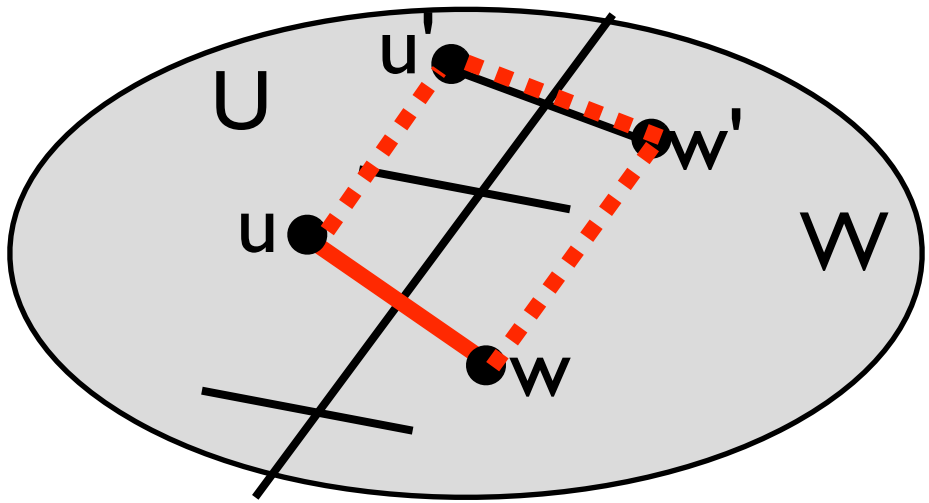


Graphenalgorithmen

Minimaler Spannbaum - Korrektheit

Beweis: Seien U, W, {u,w} wie im Lemma und T=(V,$\bar{E}$) ein minimaler Spannbaum für G, der {u,w} *nicht* enthält. Dann gibt es mindestens eine Kante, die U mit W verbindet.

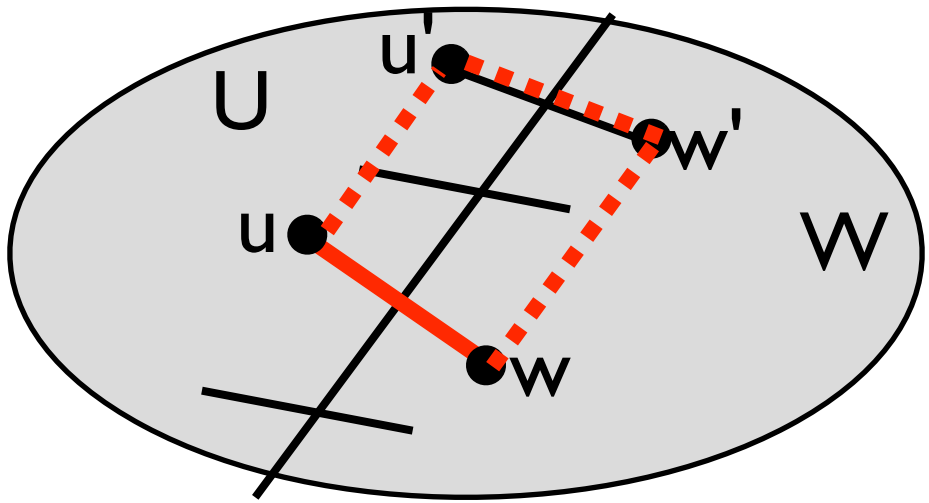
Fügt man {u,w} hinzu, so entsteht ein Graph T' mit einem Zyklus, der über {u,w} läuft.



Graphenalgorithmen

Minimaler Spannbaum - Korrektheit

u muß also noch auf einem anderen Weg mit w verbunden sein, der mind. eine weitere Kante {u',w'} enthält, die U mit W verbindet. Löscht man {u',w'}, so entsteht ein Spannbaum T", dessen Kosten höchstens so hoch sind wie die von T, da {u,w} minimale Kosten hat. Also ist T" minimaler Spannbaum, der {u,w} enthält.



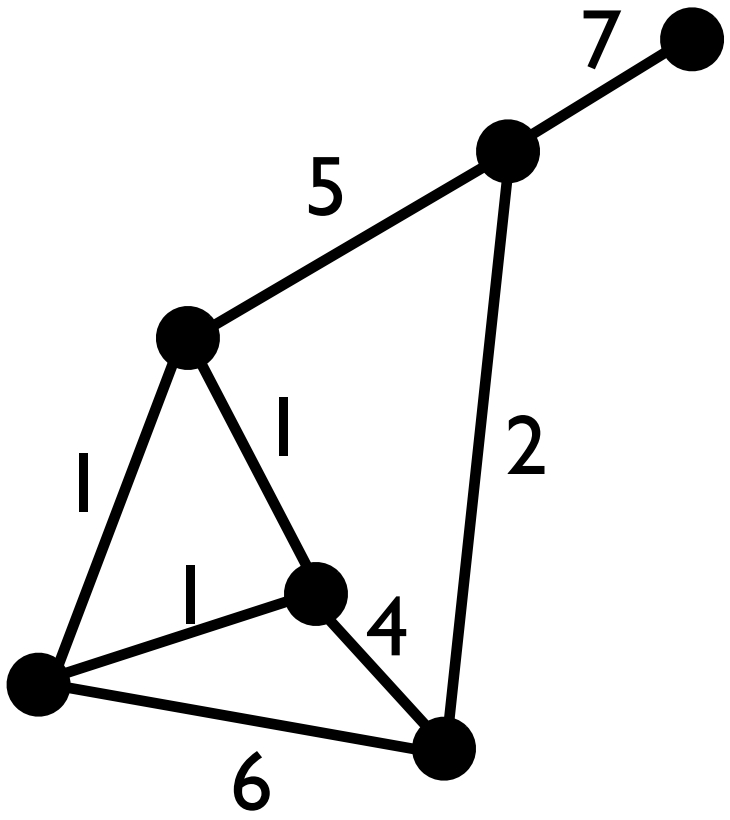
Graphenalgorithmen
Spannbaum - Kruskal-Algorithmus

Algorithmus von Kruskal:

- Beginne mit $T=(V,\varnothing)$
- Sortiere Kanten aufsteigend nach Gewichten
- Nimm fortlaufend billigste Kante, die noch verfügbar ist und die zwei Komponenten von T verbindet, zu hinzu, bis T nur noch eine Komponente hat

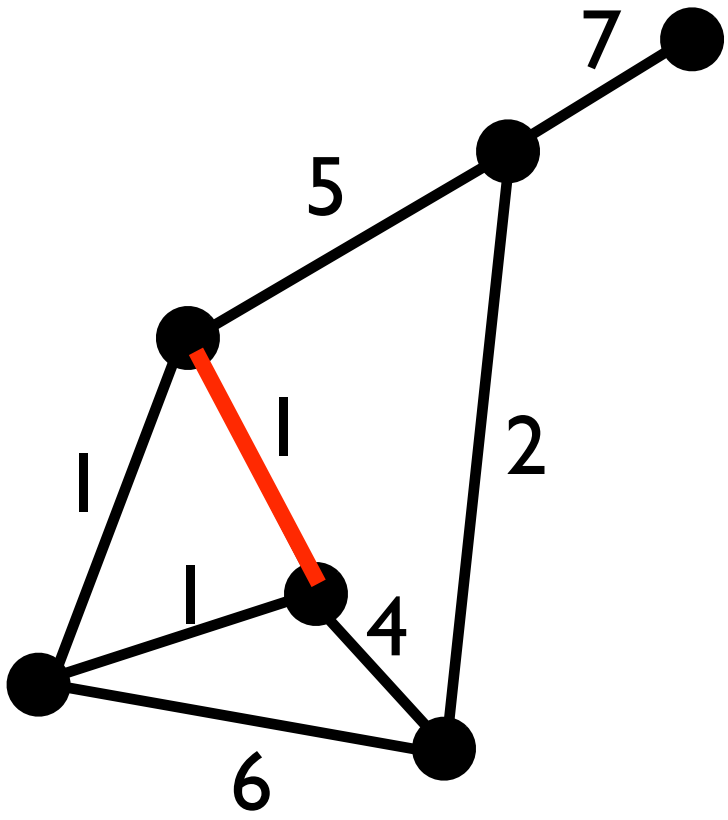
Graphenalgorithmen

Kruskal-Algorithmus - Beispiel



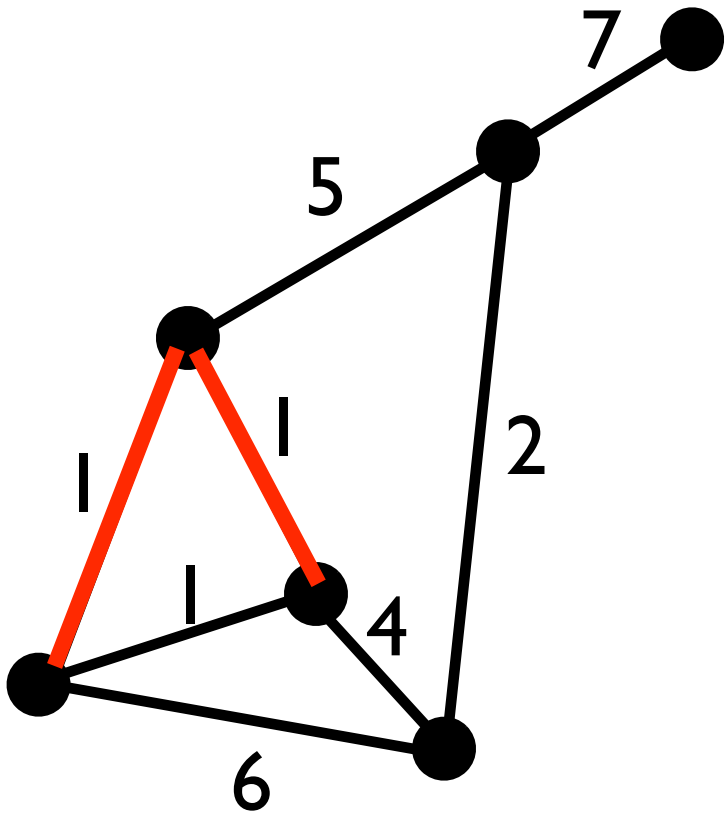
Graphenalgorithmen

Kruskal-Algorithmus - Beispiel



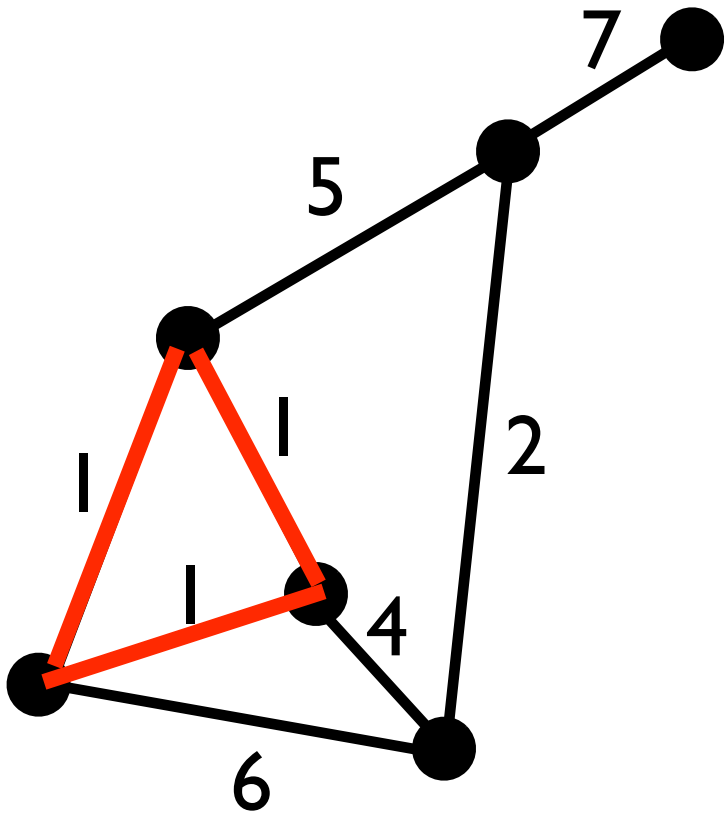
Graphenalgorithmen

Kruskal-Algorithmus - Beispiel



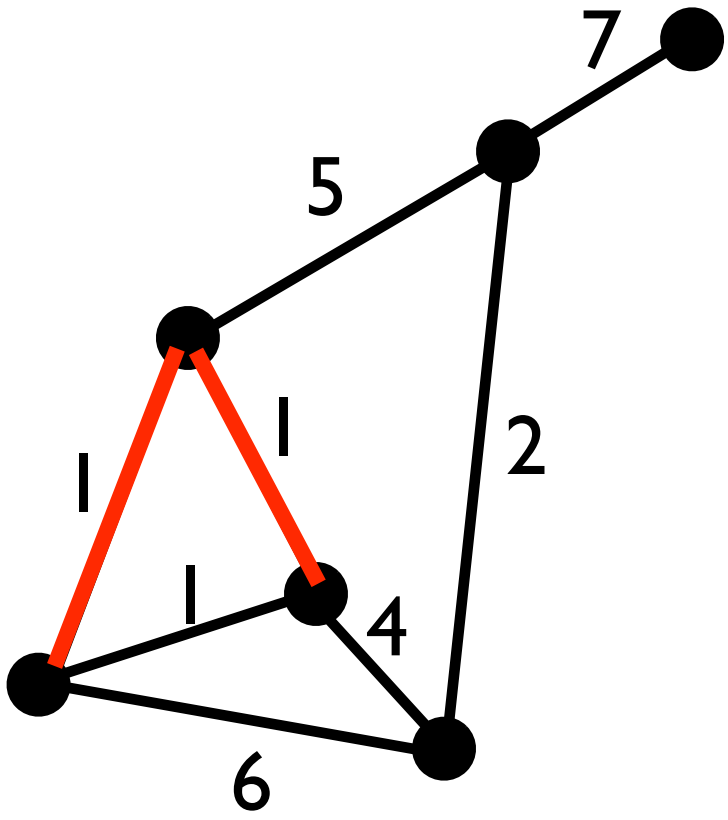
Graphenalgorithmen

Kruskal-Algorithmus - Beispiel



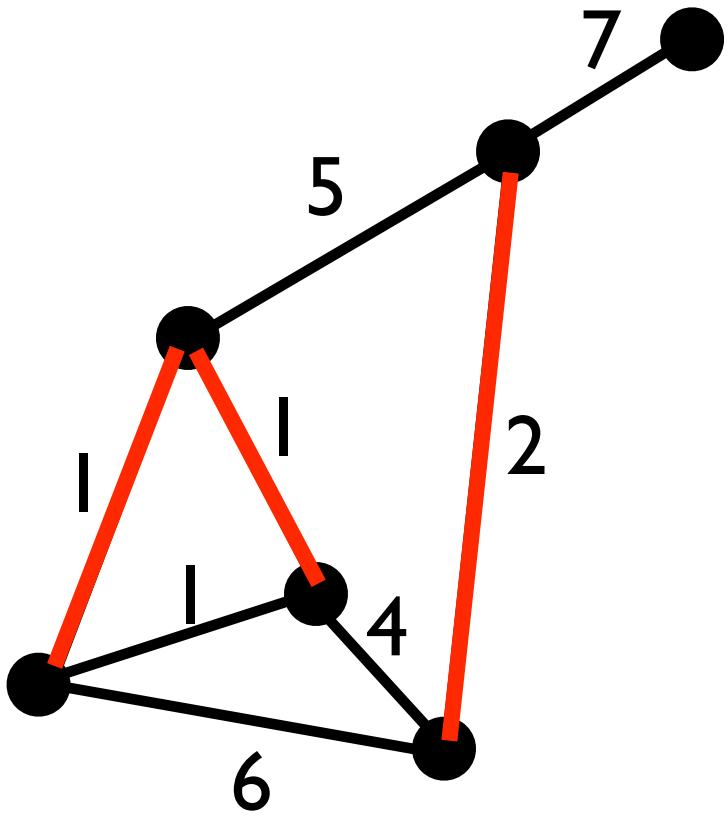
Graphenalgorithmen

Kruskal-Algorithmus - Beispiel



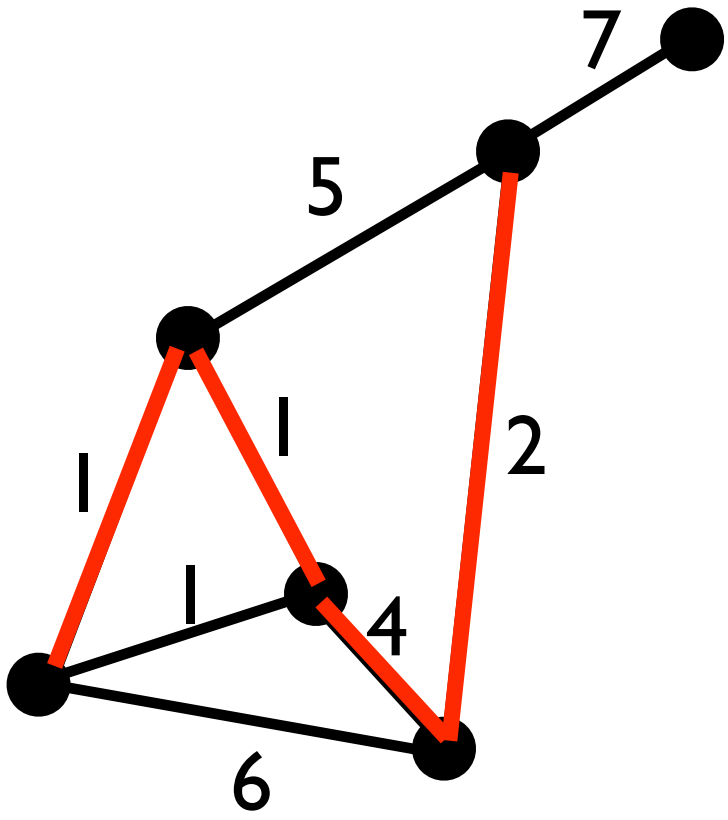
Graphenalgorithmen

Kruskal-Algorithmus - Beispiel



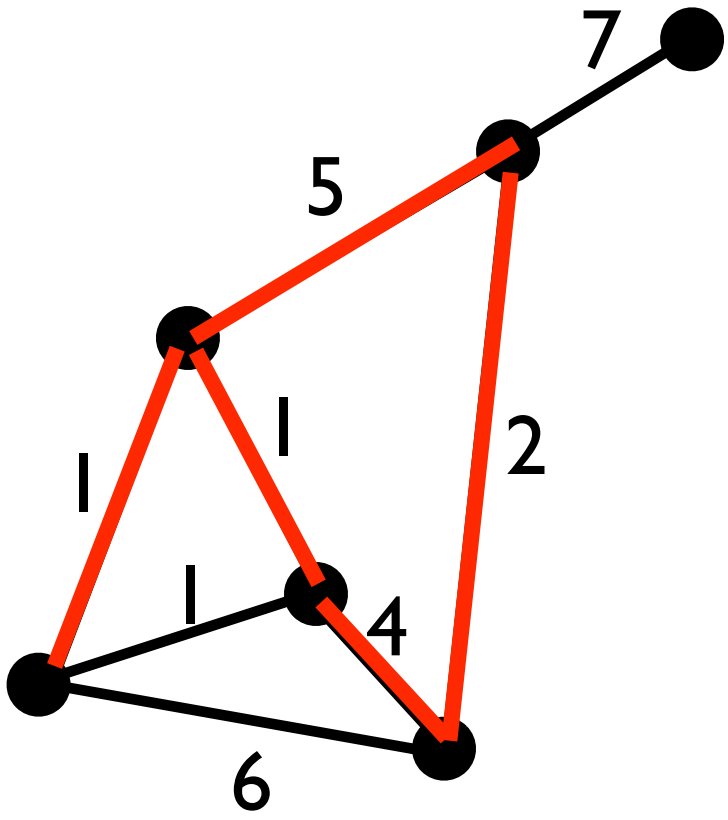
Graphenalgorithmen

Kruskal-Algorithmus - Beispiel



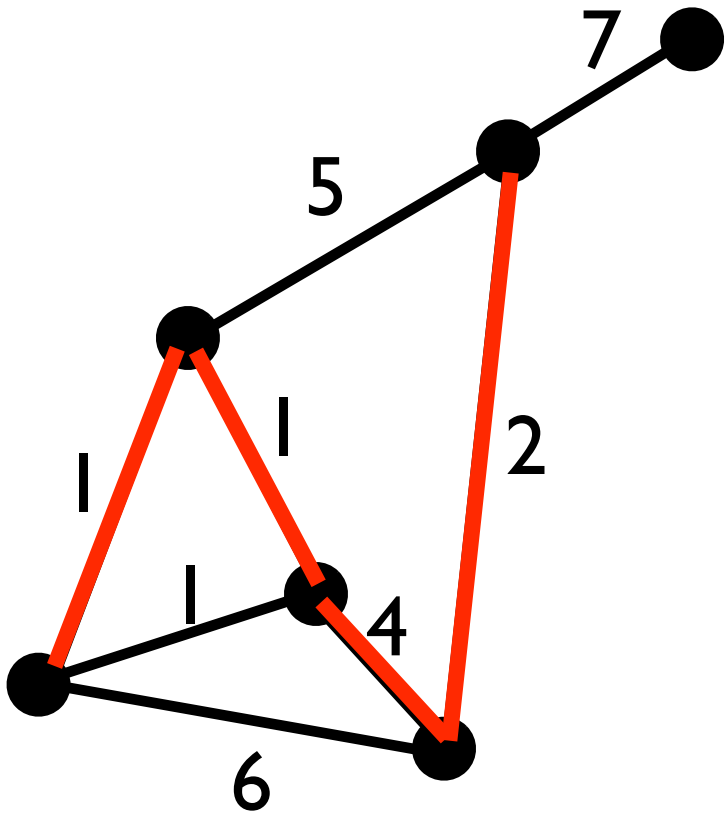
Graphenalgorithmen

Kruskal-Algorithmus - Beispiel



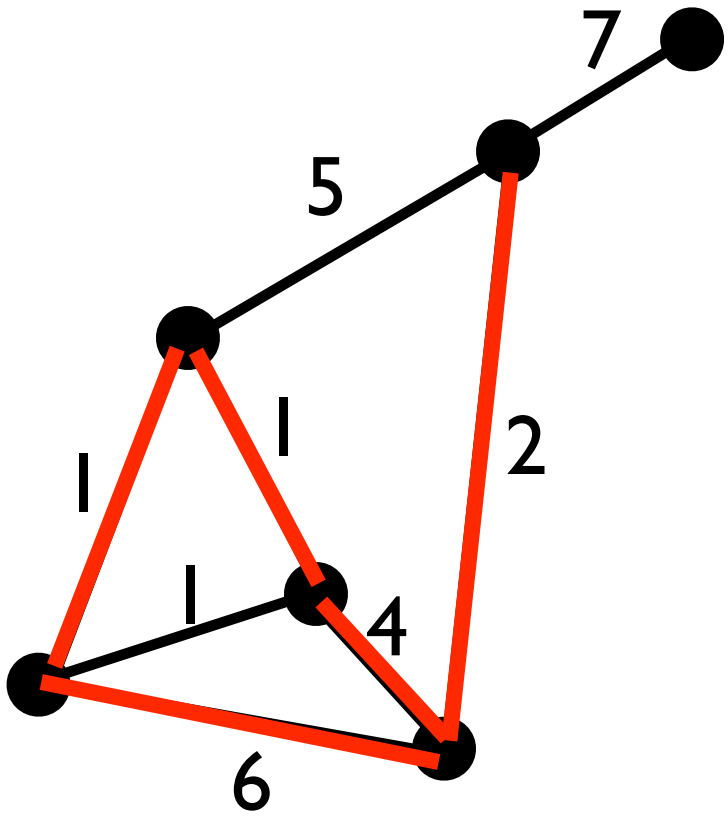
Graphenalgorithmen

Kruskal-Algorithmus - Beispiel



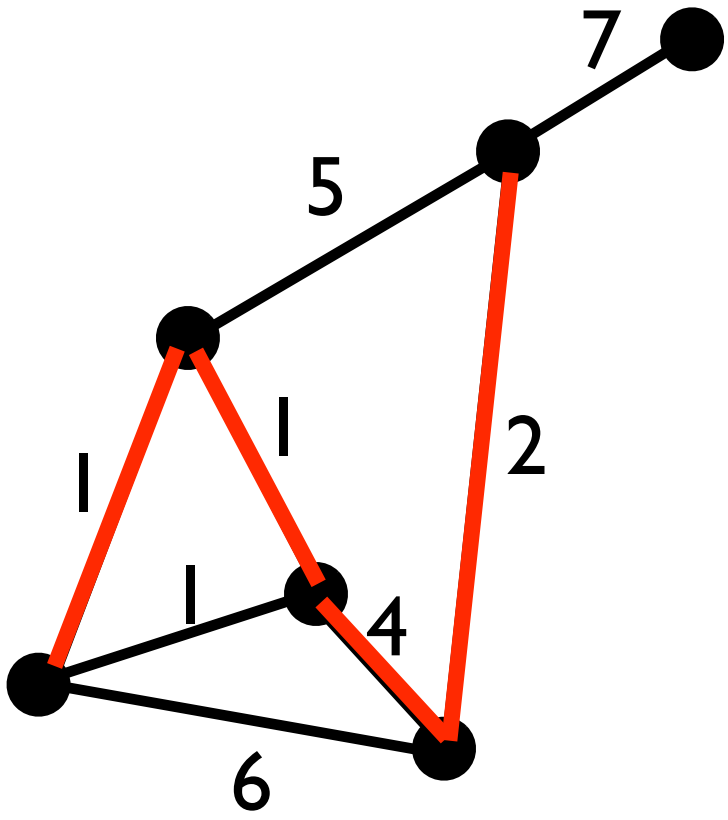
Graphenalgorithmen

Kruskal-Algorithmus - Beispiel



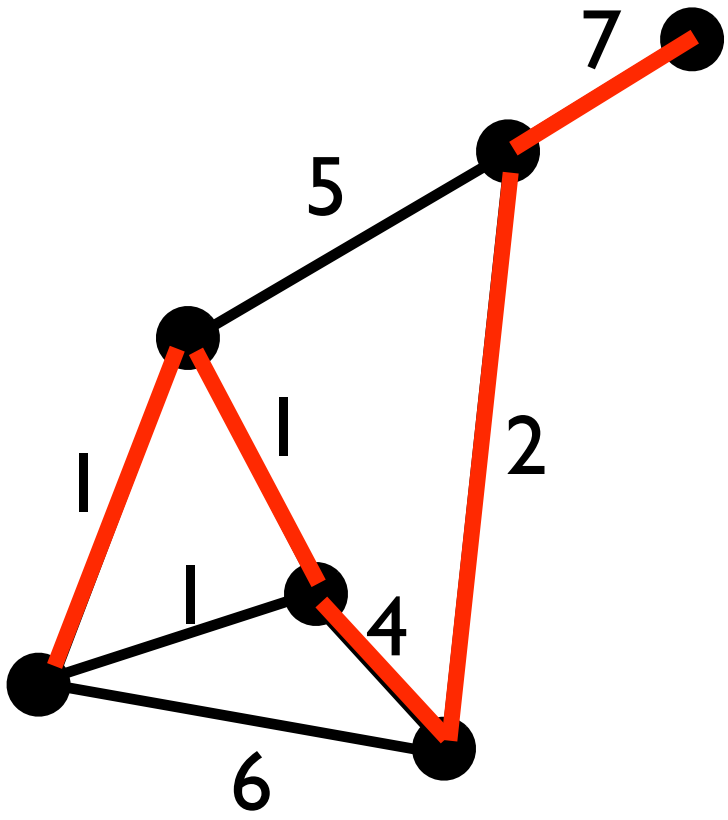
Graphenalgorithmen

Kruskal-Algorithmus - Beispiel



Graphenalgorithmen

Kruskal-Algorithmus - Beispiel



Graphenalgorithmen
Spannbaum - Kruskal-Algorithmus

```
var   P: Partition (Zerlegung) der Knoten von V
      Q: Heap von Kanten geordnet nach
            Gewichten, Minimum an der Wurzel
begin
   P:={{x} | x∈V}; T:=(V,∅); n:=|V|;
   initialisiere Q mit allen Kanten nach
      Kosten aufsteigend sortiert;
```

Graphenalgorithmen
Spannbaum - Kruskal-Algorithmus

```
while n>1 do
begin
    (Q,{v,w}):=deletemin(Q);
    a:=find(P,v); b:=find(P,w);
    if a≠b then
    begin
        T:=T+{v,w};
        P:=union(P,a,b);
        n:=n-1
    end
end
```

finde Komponenten, in denen v und w liegen

nimm {v,w} zu Spannbaum

vereinige Komponenten, bilde neue Partition

Graphenalgorithmen
Kruskal-Algorithmus - Laufzeiten

```
var   P: Partition (Zerlegung) der Knoten von V
      Q: Heap von Kanten geordnet nach
            Gewichten, Minimum an der Wurzel
begin
   P:={{x} | x∈V}; T:=(V,∅); n:=|V|;
   initialisiere Q mit allen Kanten nach
      Kosten aufsteigend sortiert;
```

Graphenalgorithmen
Kruskal-Algorithmus - Laufzeiten

var P: Partition (Zerlegung) der Knoten von V

Q: Heap von Kanten geordnet nach

Gewichten, Minimum an der Wurzel

begin

P:={{x} | x∈V}; T:=(V,∅); n:=|V|; O(n)

initialisiere Q mit allen Kanten nach

Kosten aufsteigend sortiert;

Graphenalgorithmen
Kruskal-Algorithmus - Laufzeiten

```
var   P: Partition (Zerlegung) der Knoten von V
      Q: Heap von Kanten geordnet nach
            Gewichten, Minimum an der Wurzel
begin
   P:={{x} | x∈V}; T:=(V,∅); n:=|V|;
   initialisiere Q mit allen Kanten nach
      Kosten aufsteigend sortiert;
```

Graphenalgorithmen
Kruskal-Algorithmus - Laufzeiten

var P: Partition (Zerlegung) der Knoten von V

Q: Heap von Kanten geordnet nach

Gewichten, Minimum an der Wurzel

begin

P:={{x} | x∈V}; T:=(V,∅); n:=|V|;

initialisiere Q mit allen Kanten nach Kosten aufsteigend sortiert;

Aufbau eines Heaps mit |E| Elementen: O(|E|log|E|)

Graphenalgorithmen
Kruskal-Algorithmus - Laufzeiten

```
while n>1 do
begin
    (Q,{v,w}):=deletemin(Q);
    a:=find(P,v); b:=find(P,w);
    if a≠b then
    begin
        T:=T+{v,w};
        P:=union(P,a,b);
        n:=n-1
    end
end
```

Graphenalgorithmen

Kruskal-Algorithmus - Laufzeiten

```
while n>1 do
begin
    (Q,{v,w}):=deletemin(Q);
    a:=find(P,v); b:=find(P,w);
    if a≠b then
    begin
        T:=T+{v,w};
        P:=union(P,a,b);
        n:=n-1
    end
end
```

Löschen der Wurzel, Heap umorganisieren: O(log|E|)

Graphenalgorithmen
Kruskal-Algorithmus - Laufzeiten

```
while n>1 do
begin
    (Q,{v,w}):=deletemin(Q);
    a:=find(P,v); b:=find(P,w);
    if a≠b then
    begin
        T:=T+{v,w};
        P:=union(P,a,b);
        n:=n-1
    end
end
```

Graphenalgorithmen
Kruskal-Algorithmus - Laufzeiten

```
while n>1 do
begin
    (Q,{v,w}):=deletemin(Q);
    a:=find(P,v); b:=find(P,w);
    if a≠b then
    begin
        T:=T+{v,w};
        P:=union(P,a,b);
        n:=n-1
    end
end
```

union-find-Problem:
union: O(1)
find: O(log |E|)

Graphenalgorithmen
Kruskal-Algorithmus - Laufzeiten

```
while n>1 do
begin
    (Q,{v,w}):=deletemin(Q);
    a:=find(P,v); b:=find(P,w);
    if a≠b then
    begin
        T:=T+{v,w};
        P:=union(P,a,b);
        n:=n-1
    end
end
```

Graphenalgorithmen

Kruskal-Algorithmus - Laufzeiten

|V| Schleifendurchläufe

```
while n>1 do
begin
    (Q,{v,w}):=deletemin(Q);
    a:=find(P,v); b:=find(P,w);
    if a≠b then
    begin
        T:=T+{v,w};
        P:=union(P,a,b);
        n:=n-1
    end
end
```

Graphenalgorithmen
Kruskal-Algorithmus - Laufzeiten

```
while n>1 do
begin
    (Q,{v,w}):=deletemin(Q);
    a:=find(P,v); b:=find(P,w);
    if a≠b then
    begin
        T:=T+{v,w};
        P:=union(P,a,b);
        n:=n-1
    end
end
```

Graphenalgorithmen
Kruskal-Algorithmus - Laufzeiten

```
while n>1 do
begin
    (Q,{v,w}):=deletemin(Q);
    a:=find(P,v); b:=find(P,w);
    if a≠b then
    begin
        T:=T+{v,w};
        P:=union(P,a,b);
        n:=n-1
    end
end
```

Gesamt:
$O(|E| \log |E|)=O(|E| \log |V|)$

Graphenalgorithmen
Kruskal-Algorithmus - Laufzeiten

Satz

Unter der Annahme der Laufzeiten für union und find benötigt der Kruskal-Algorithmus zur Bestimmung eines minimalen Spannbaums für einen Graphen G=(V,E)

$$O(|E| \log |V|)$$

Zeit.

Graphenalgorithmen
Union-find-Problem

- Klassisches und in vielen Zusammenhängen auftretendes Problem
- Gesucht: effiziente Implementierung von Folgen von Operationen bestehend aus Vereinigung und Element-Abfrage auf Mengen

Graphenalgorithmen
Union-find-Problem - Grundlagen

Sei S eine Menge und P eine Partition (disjunkte Zerlegung) von S in Teilmengen:

$$P=\{p_1,...,p_n\} \text{ mit } p_i \cap p_j = \emptyset,\ i \neq j,\ \cup p_i = S.$$

Wichtige Operationen:

- union(P,p_i,p_j):=$p_i \cup p_j$ und Erzeugung von P'
- find(P,x):=i für $x \in S$ und $x \in p_i$.

Ziel: effiziente Implementierung von Folgen von union-/find-Operationen

Graphenalgorithmen
Union-find-Problem - Idee: Wurzelgraph

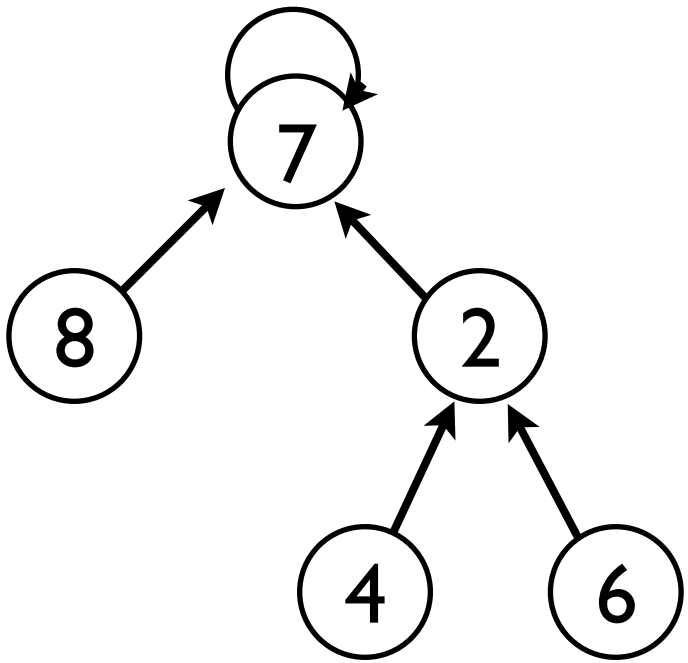
- Idee: Implementiere Mengen durch sog. Wurzelgraphen

Definition

Ein **Wurzelgraph** ist ein gerichteter Graph, dessen ungerichtete Version ein Baum ist. Alle (gerichteten) Kanten sind von den Söhnen zu den Vätern gerichtet. Die Wurzel besitzt eine (gerichtete) Schlinge.

Graphenalgorithmen

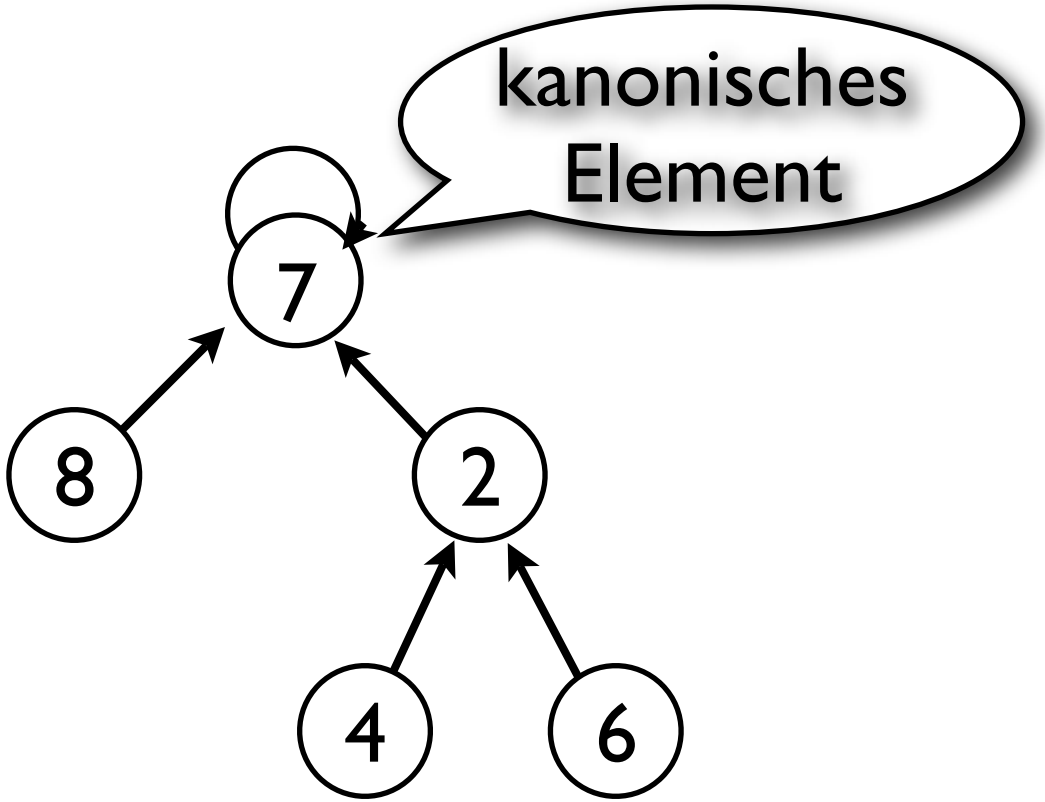
Union-find-Problem - Wurzelgraph



{2,7,4,8,6}

Graphenalgorithmen

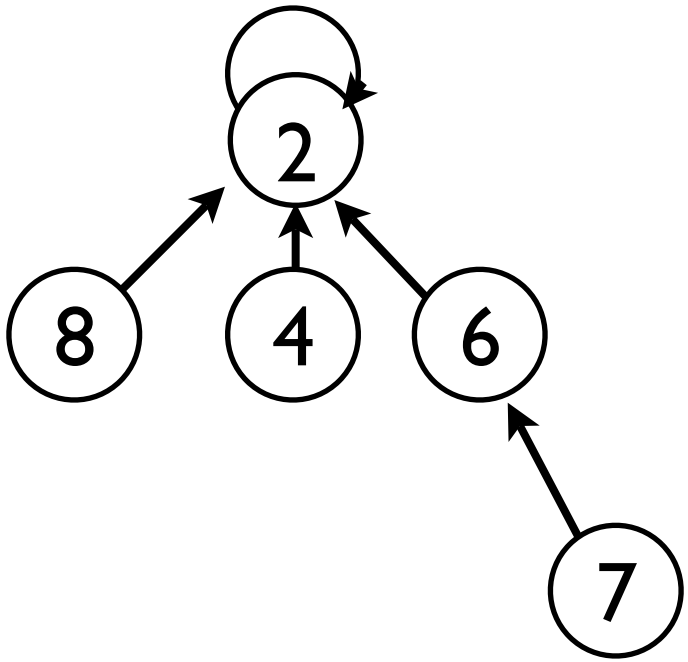
Union-find-Problem - Wurzelgraph



{2,7,4,8,6}

Graphenalgorithmen

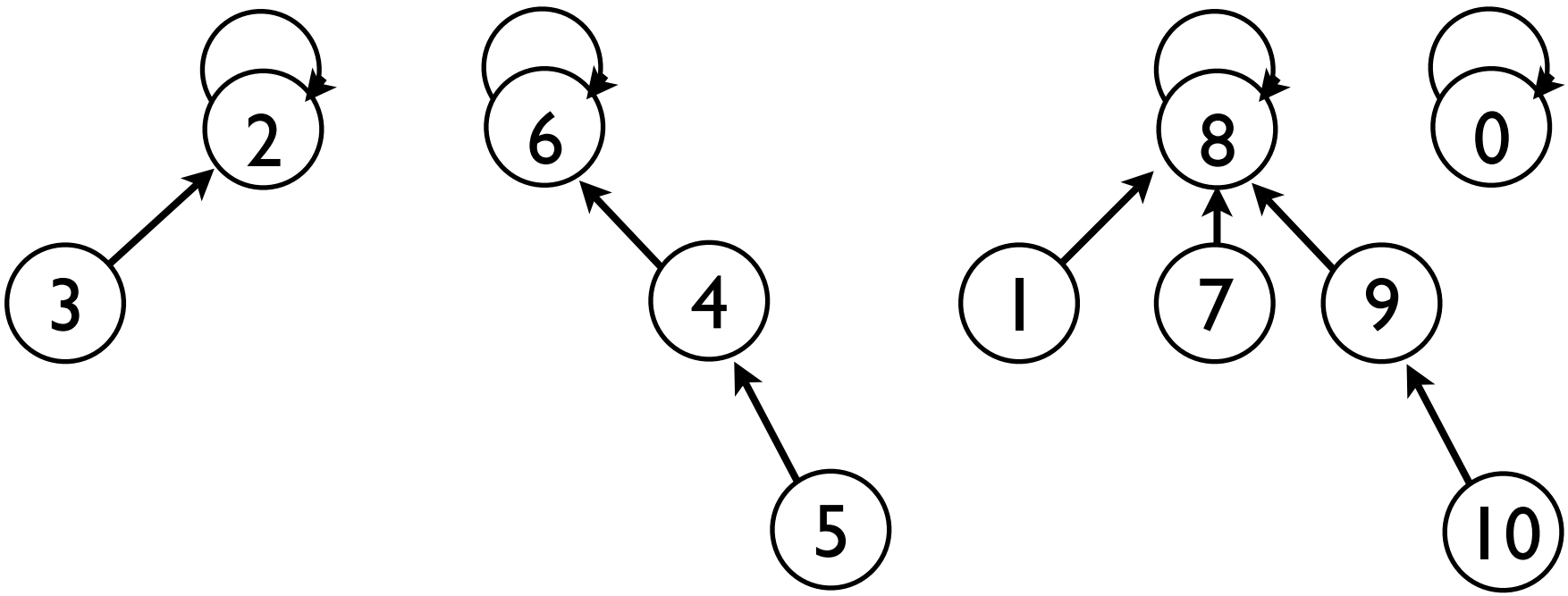
Union-find-Problem - Wurzelgraph



{2,7,4,8,6}

Graphenalgorithmen

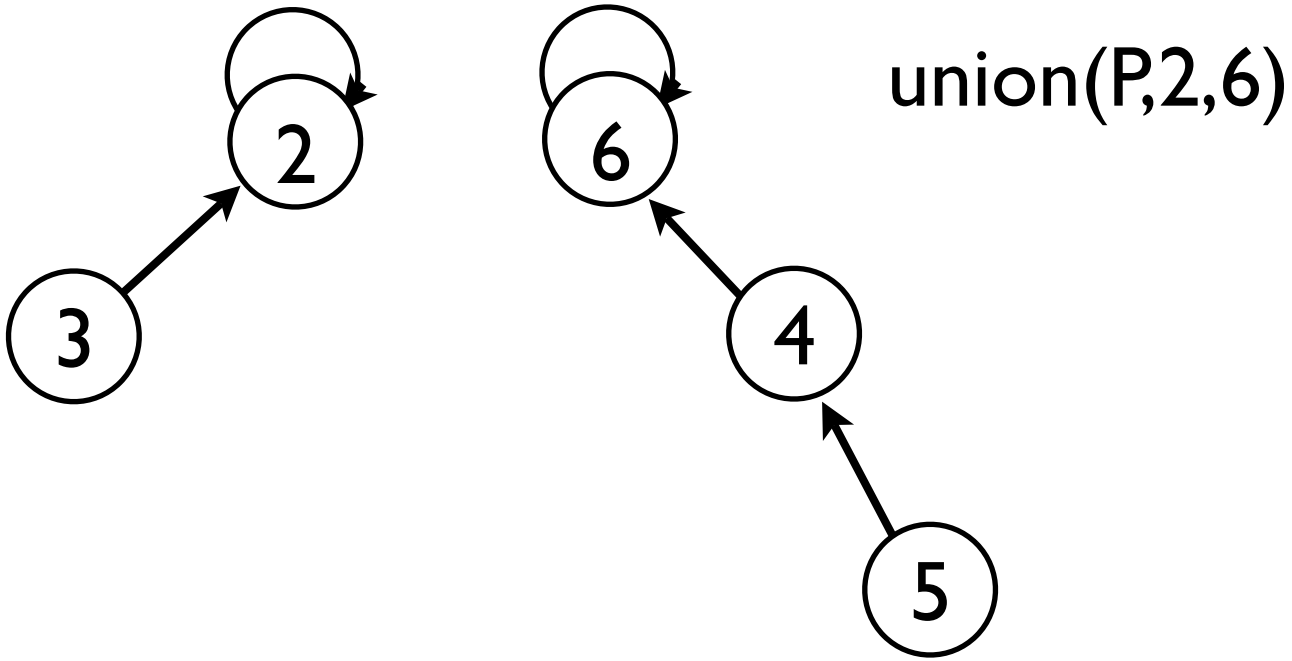
Union-find-Problem - Wurzelgraph



Partition: {{2,3},{4,6,5},{1,8,7,9,10},{0}}

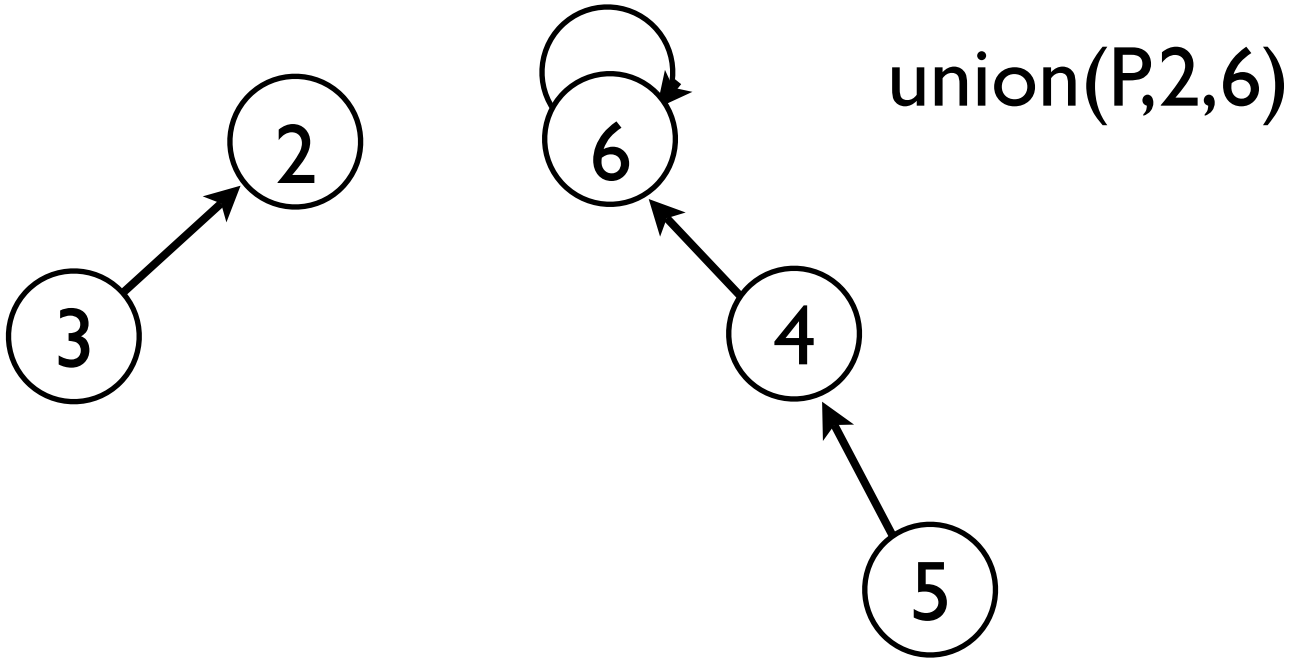
Graphenalgorithmen

Union-find-Problem - Union



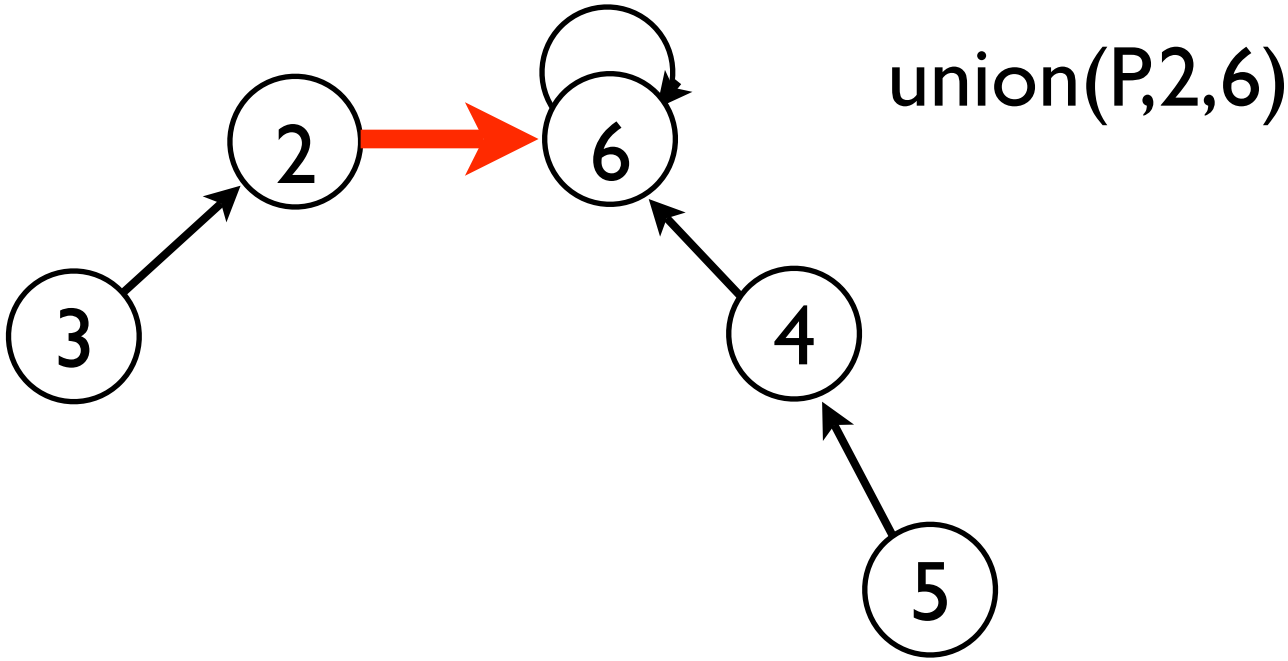
Graphenalgorithmen

Union-find-Problem - Union



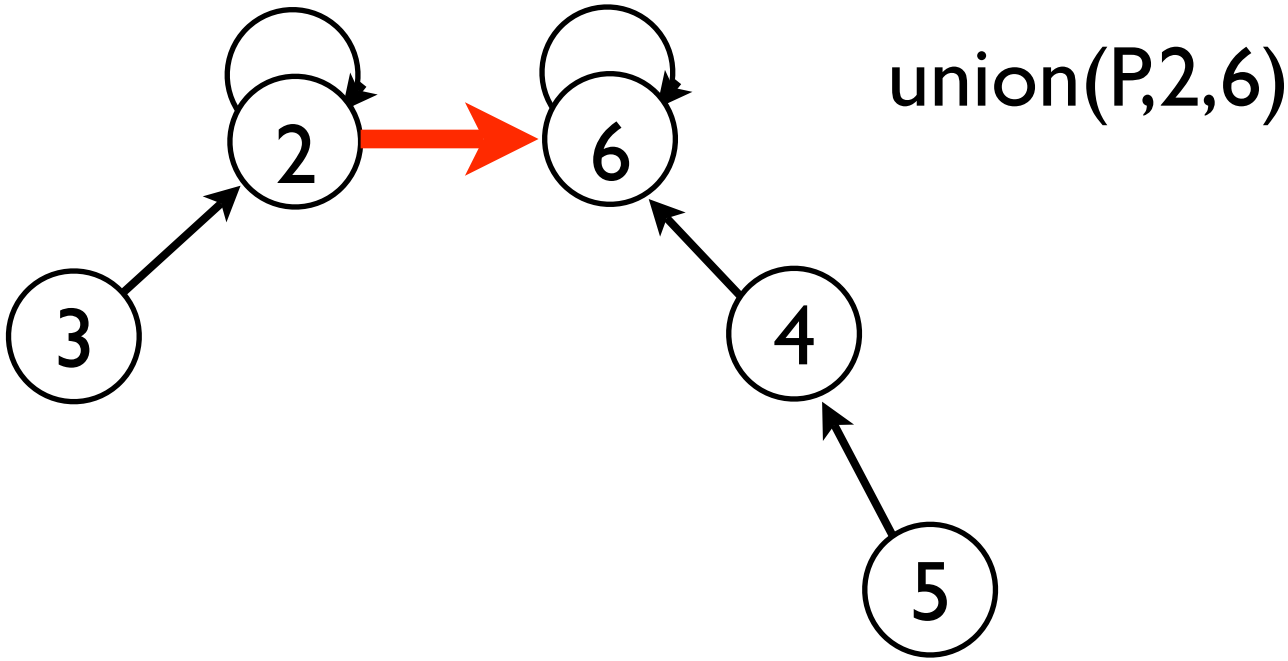
Graphenalgorithmen

Union-find-Problem - Union



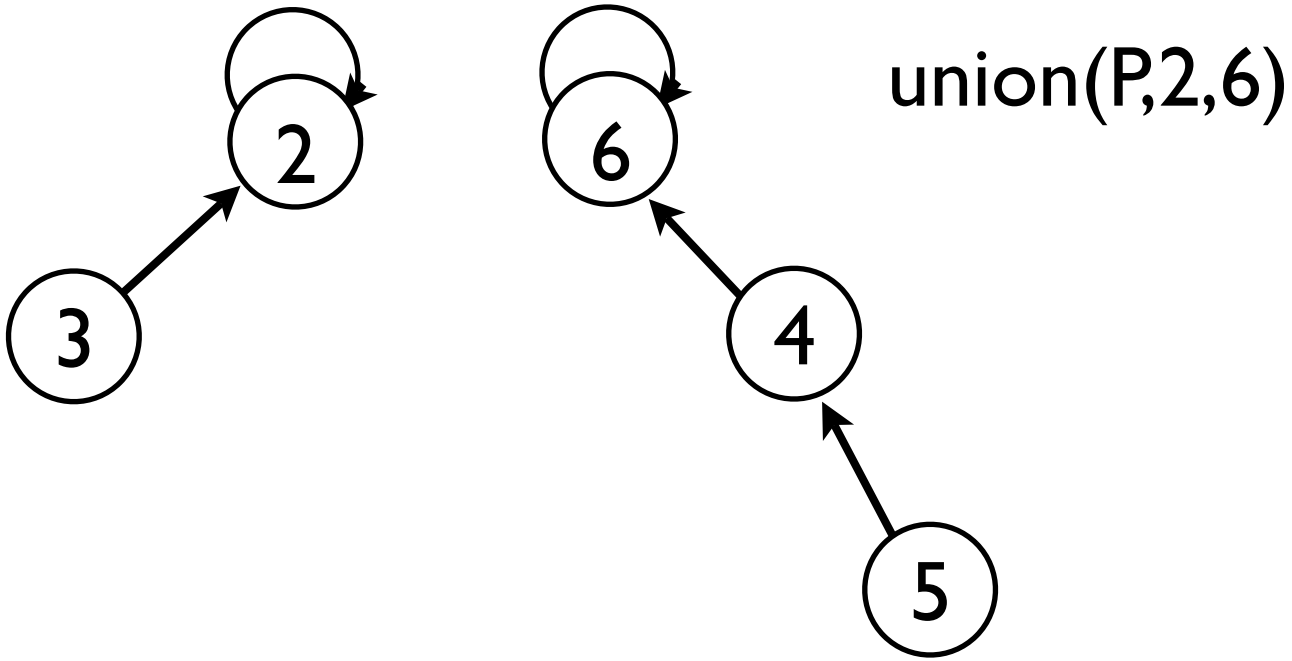
Graphenalgorithmen

Union-find-Problem - Union



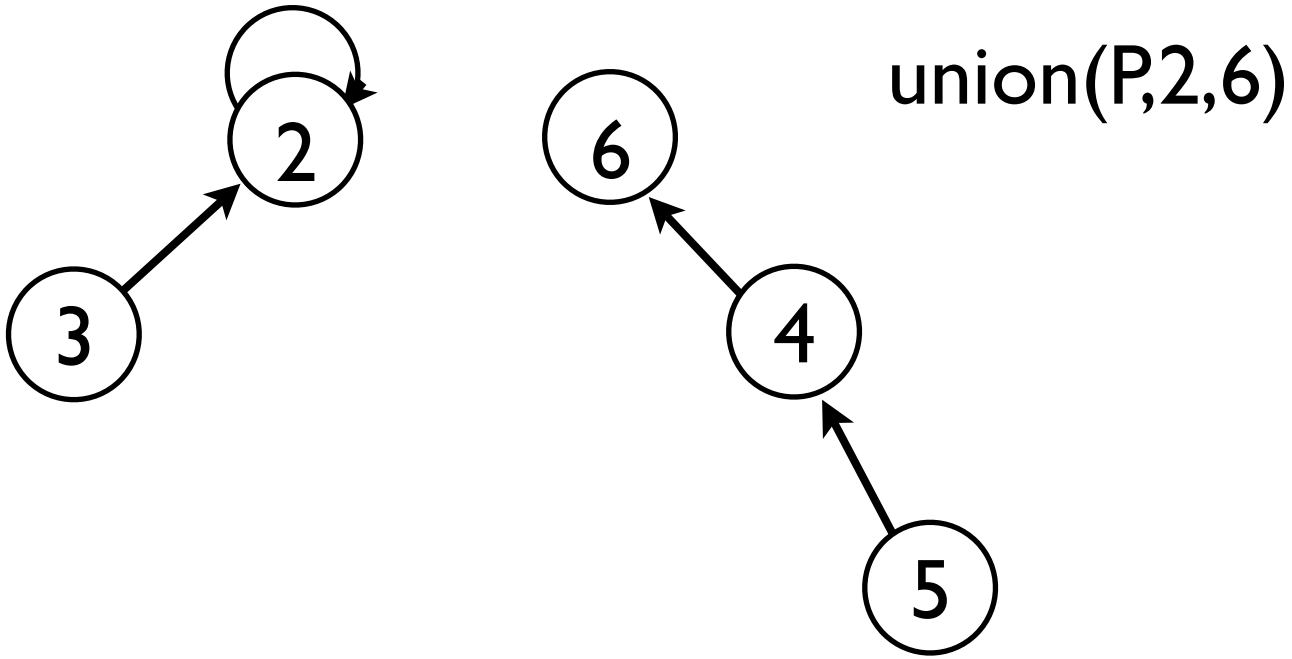
Graphenalgorithmen

Union-find-Problem - Union



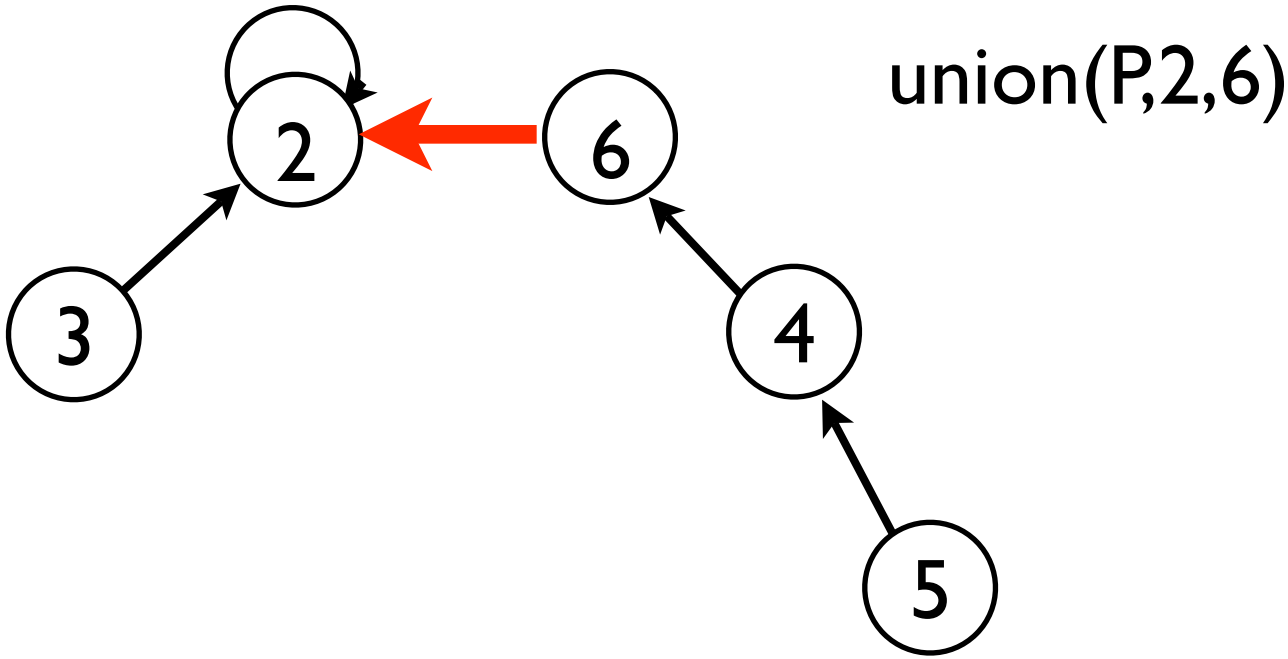
Graphenalgorithmen

Union-find-Problem - Union



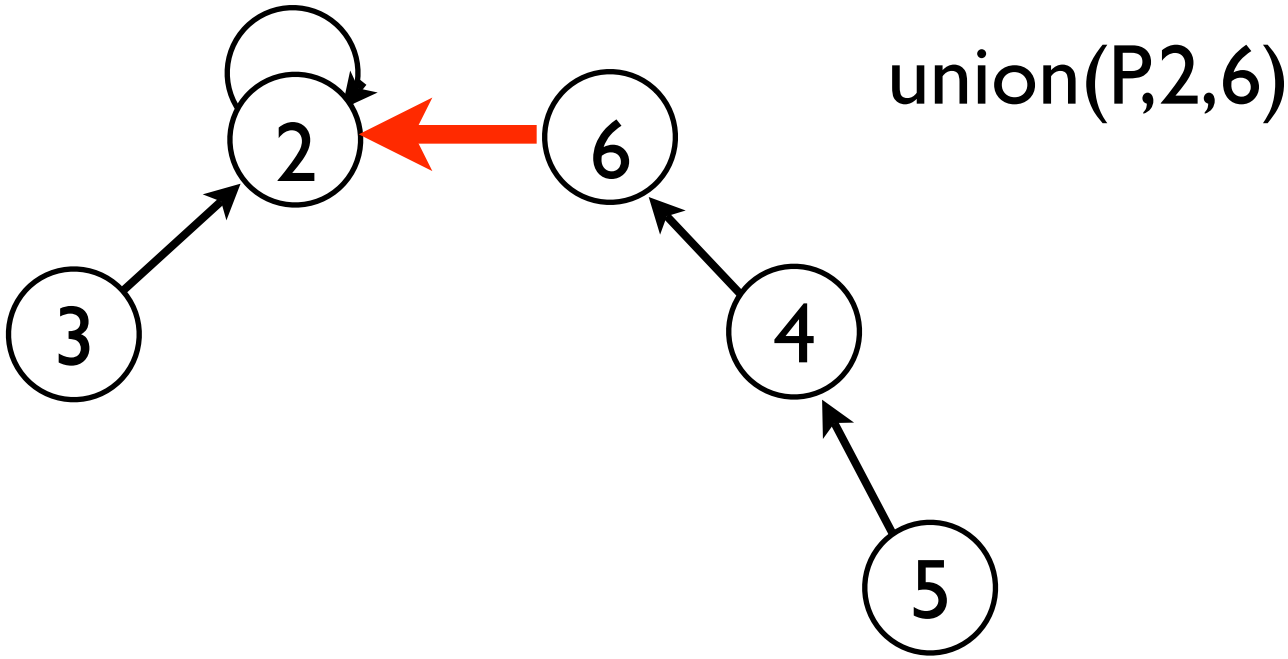
Graphenalgorithmen

Union-find-Problem - Union



Graphenalgorithmen

Union-find-Problem - Union

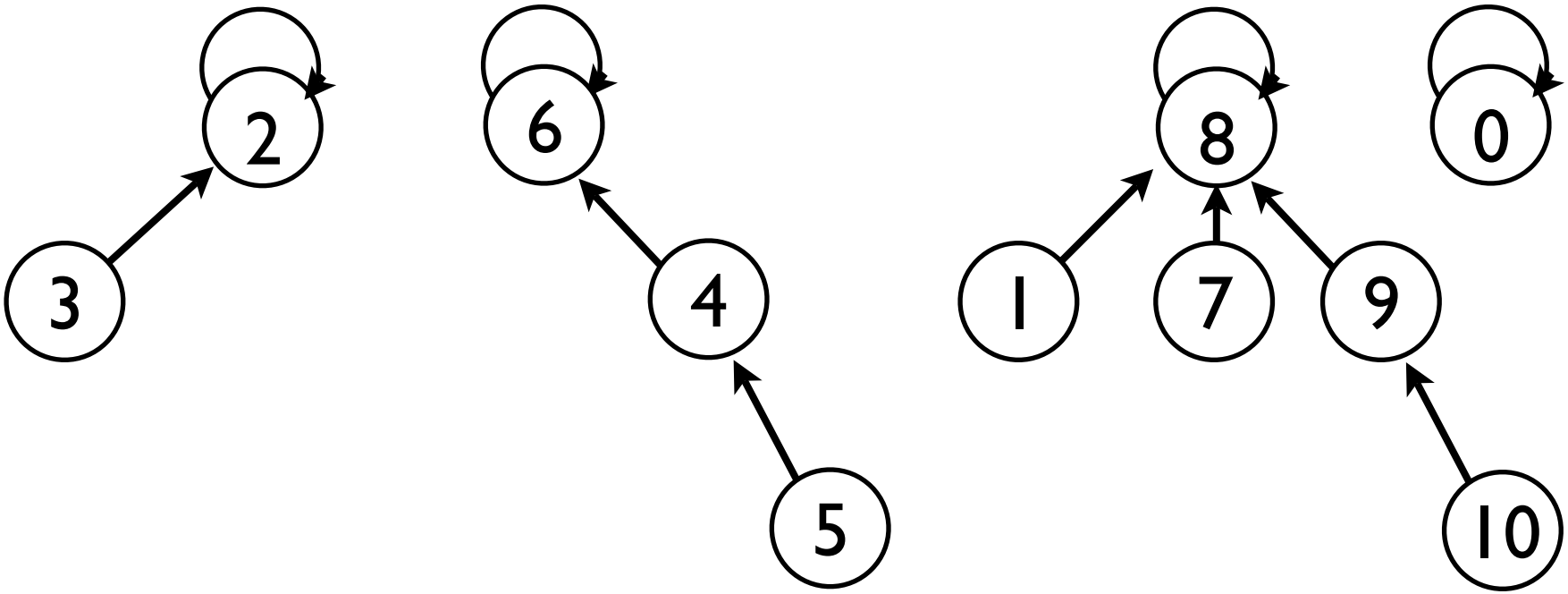


Laufzeit: O(1)

Graphenalgorithmen

Union-find-Problem - find

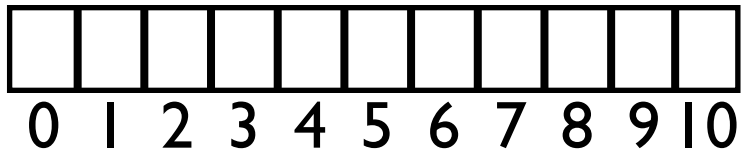
find(P,x)

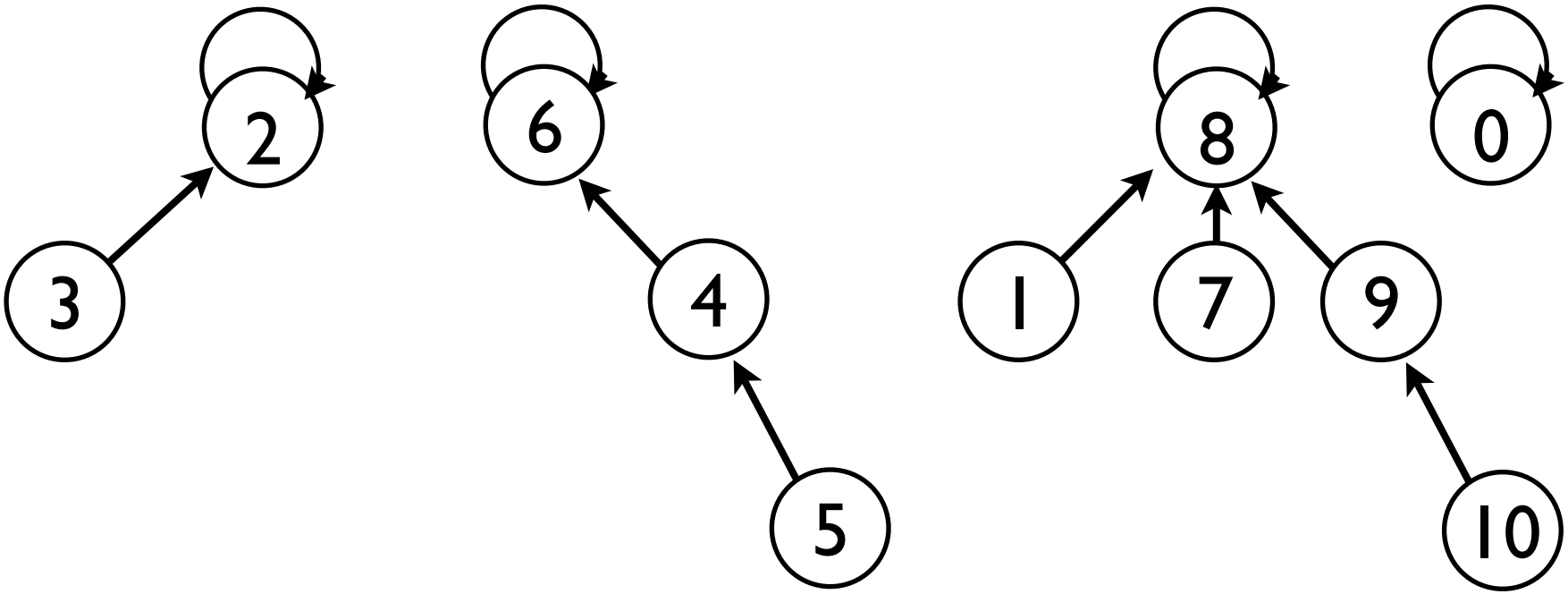


Graphenalgorithmen

Union-find-Problem - find

find(P,x)

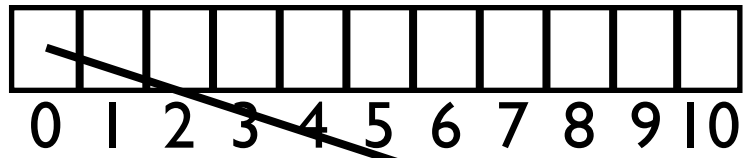


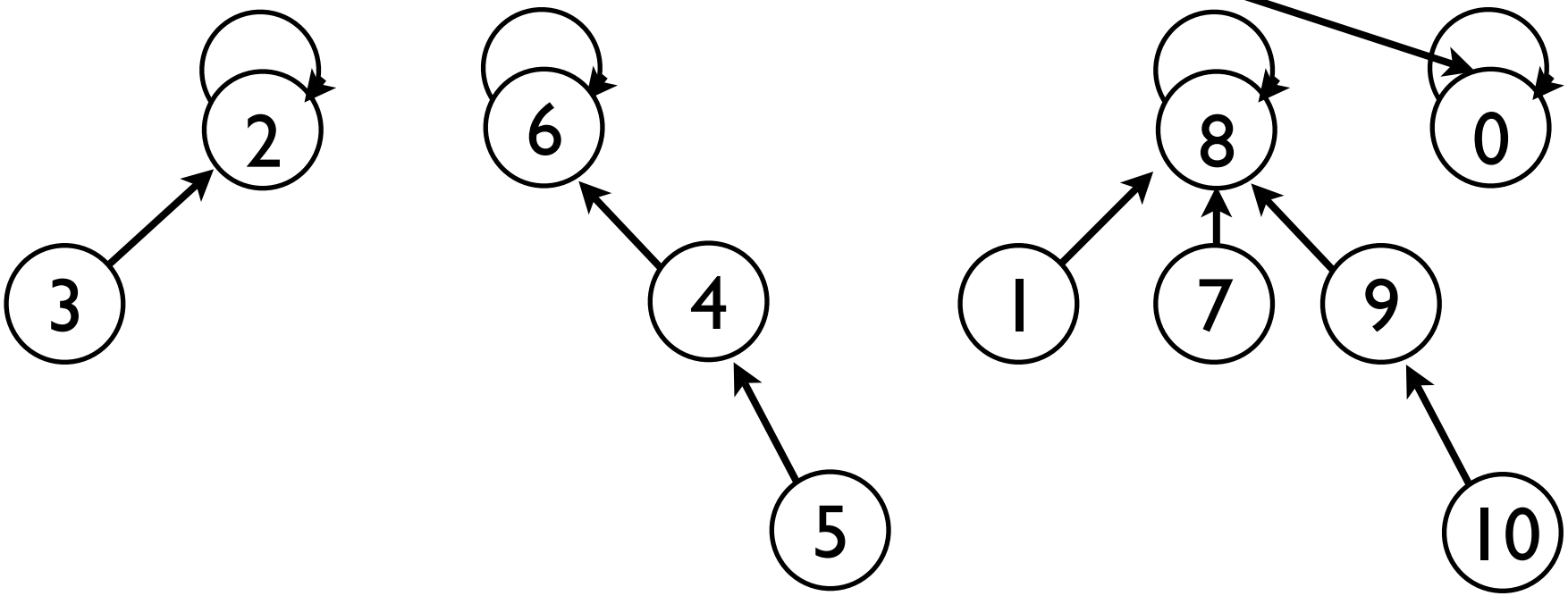


Graphenalgorithmen

Union-find-Problem - find

find(P,x)

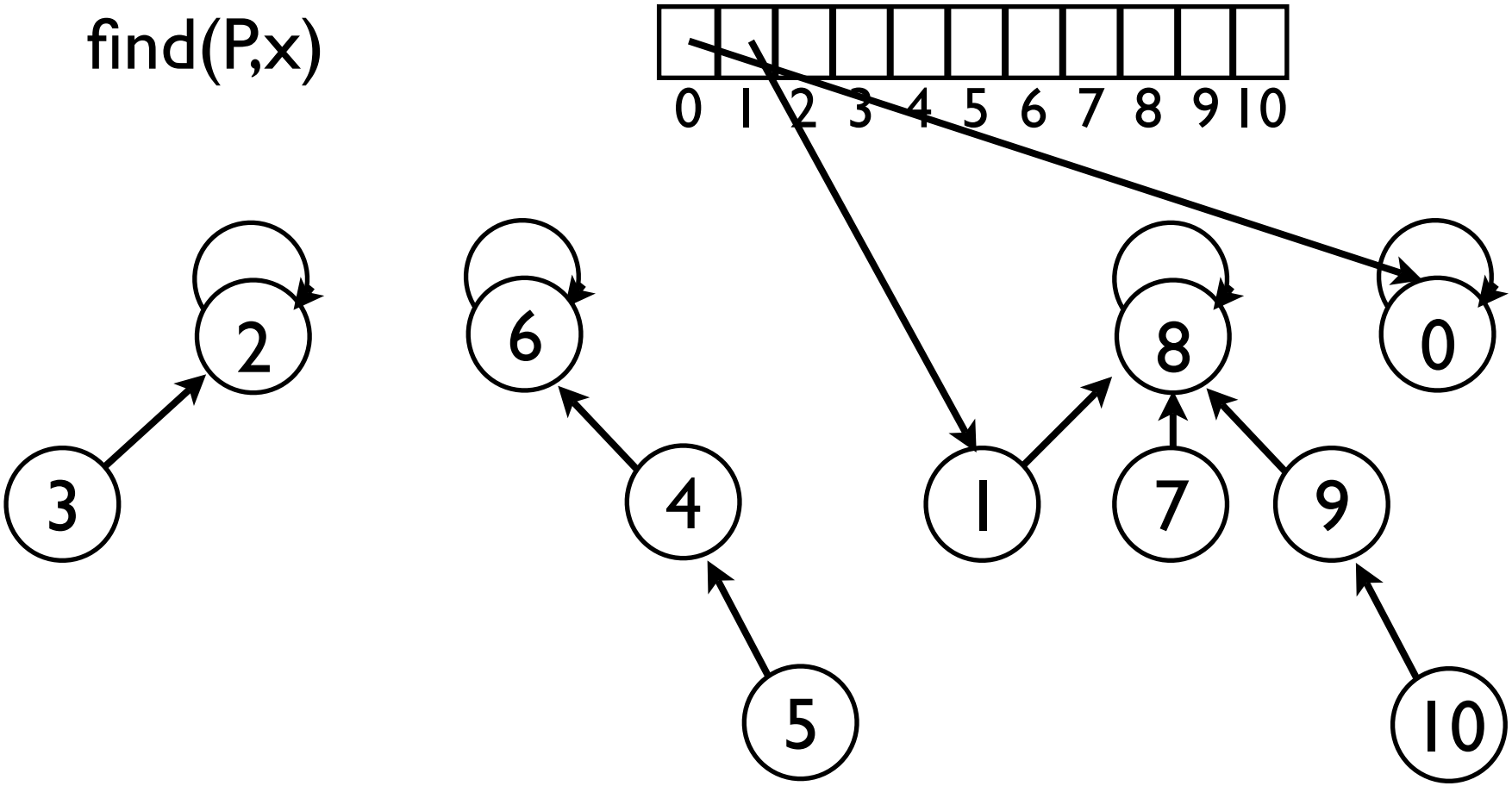




Graphenalgorithmen

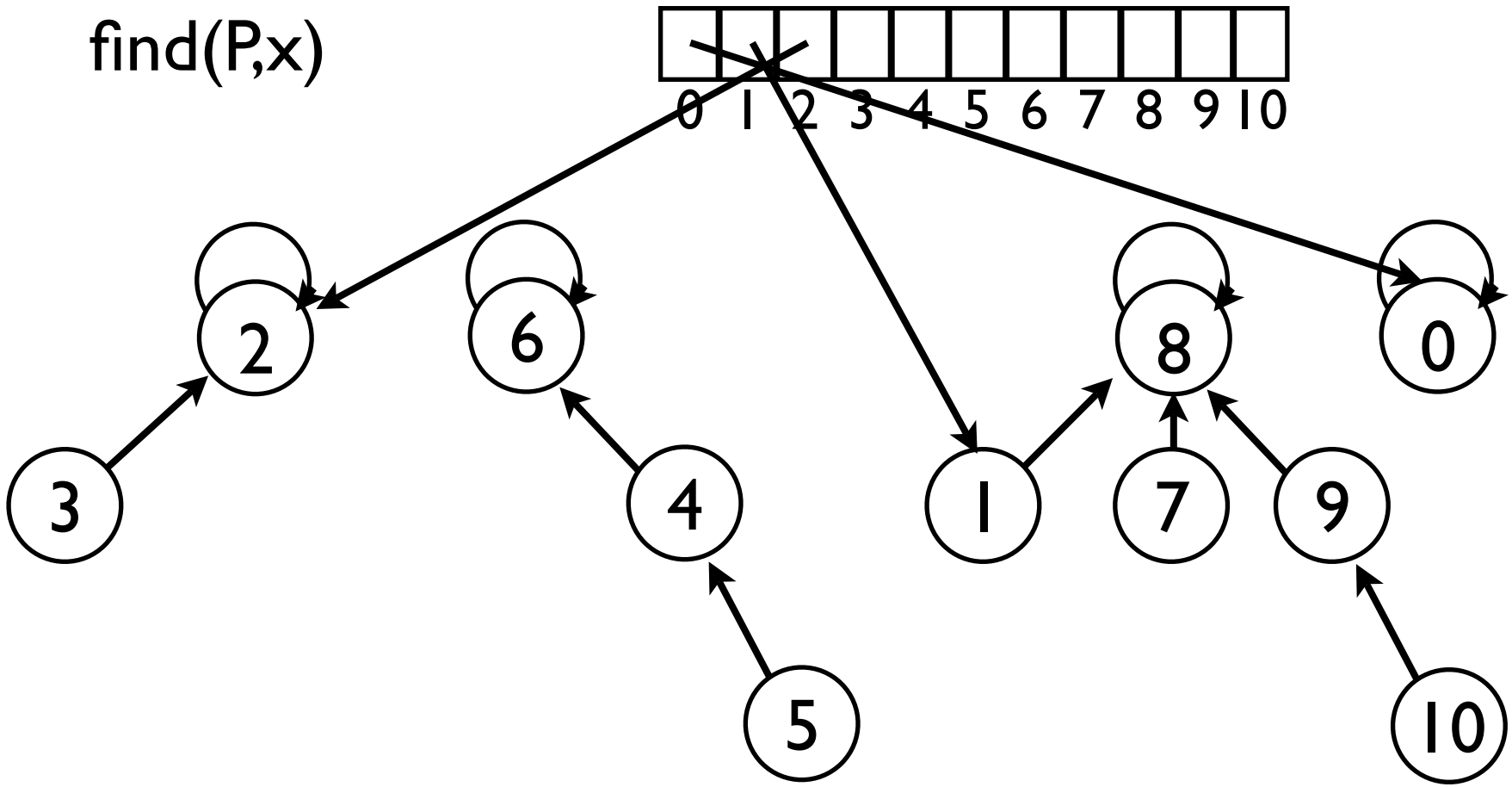
Union-find-Problem - find

find(P,x)



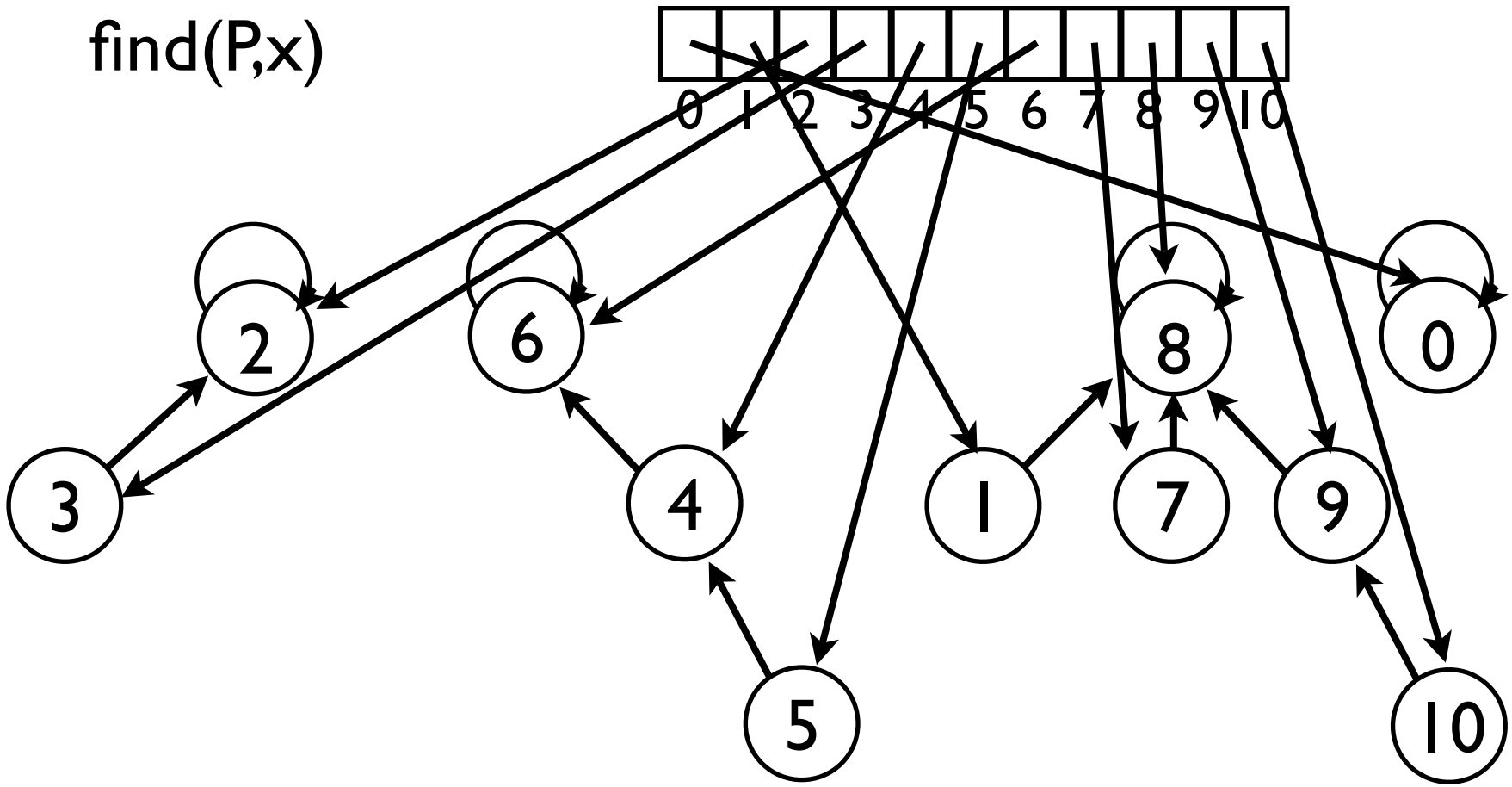
Graphenalgorithmen

Union-find-Problem - find



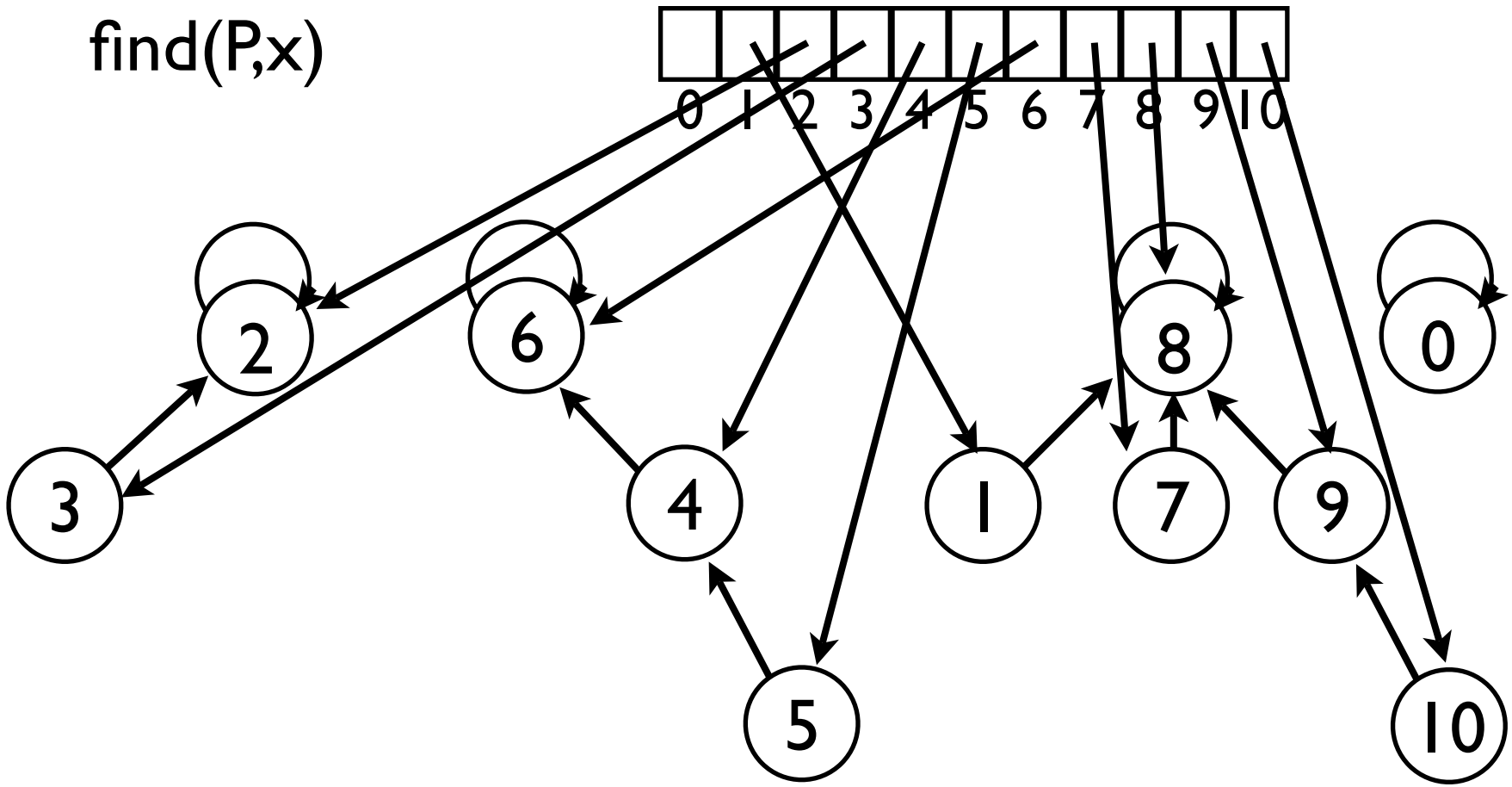
Graphenalgorithmen

Union-find-Problem - find



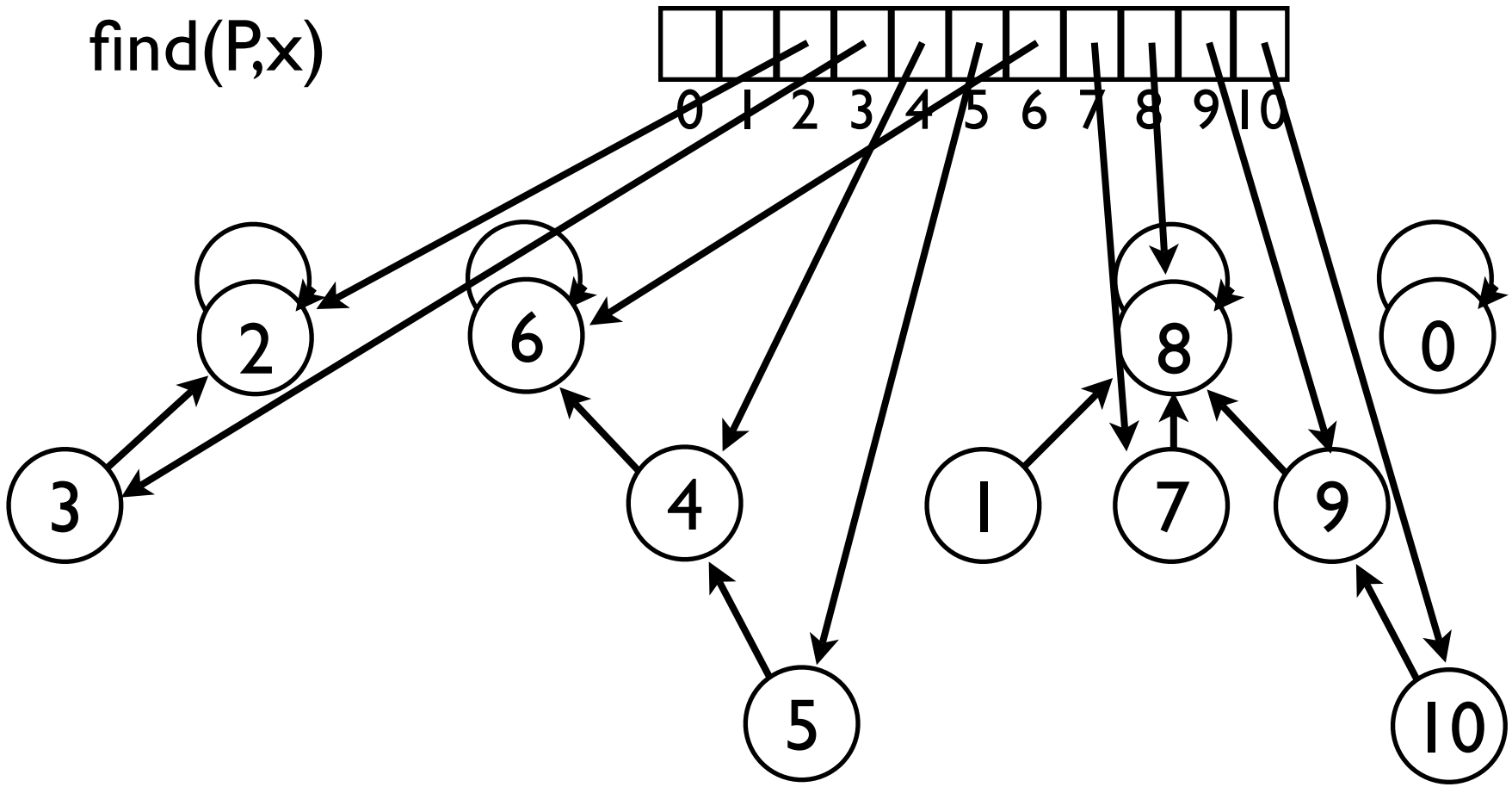
Graphenalgorithmen

Union-find-Problem - find



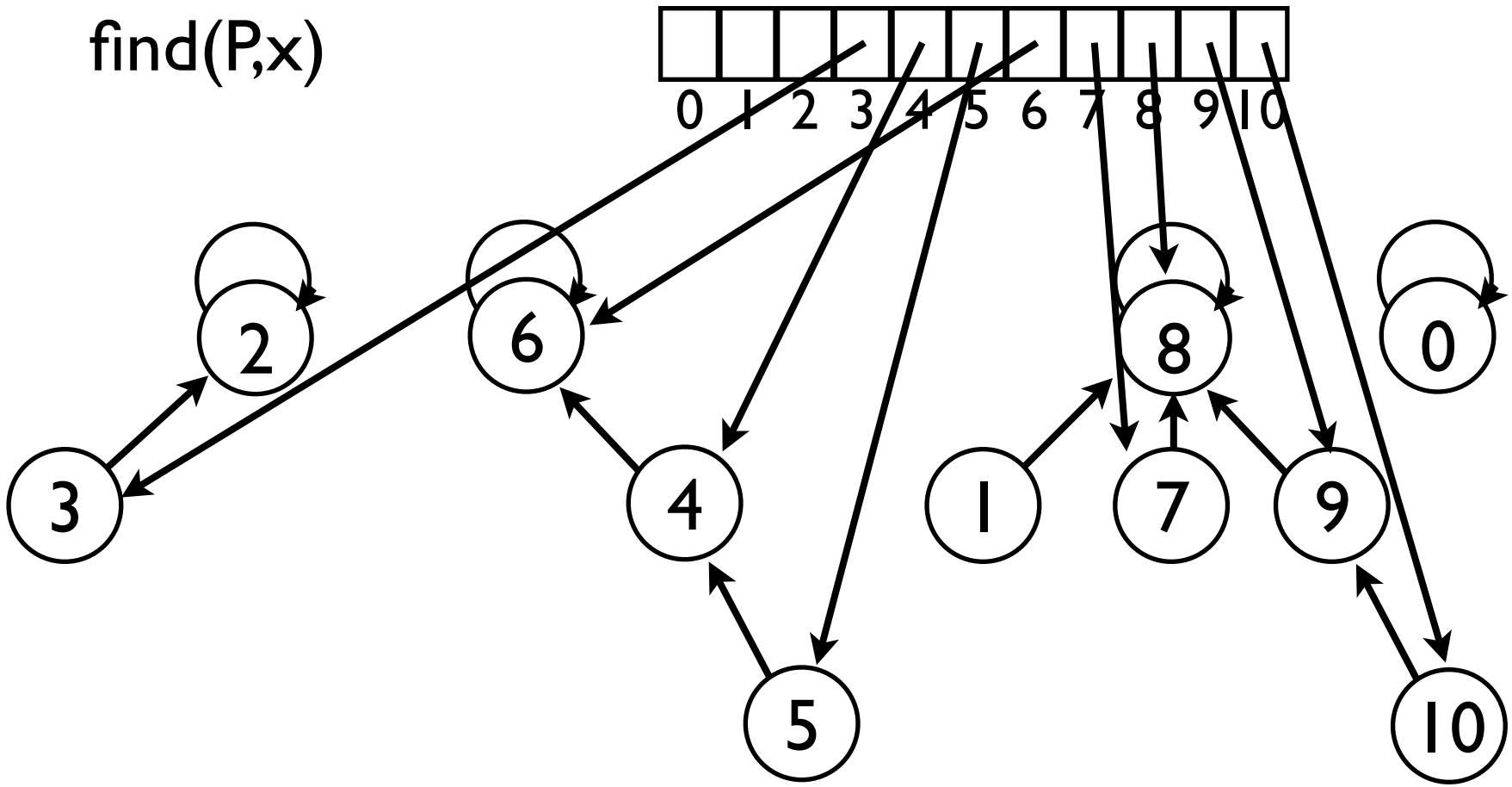
Graphenalgorithmen

Union-find-Problem - find



Graphenalgorithmen

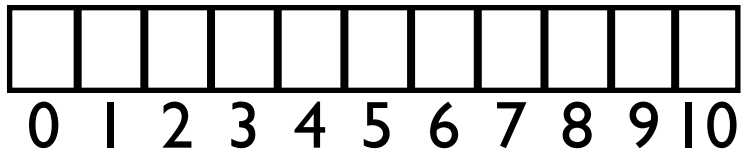
Union-find-Problem - find

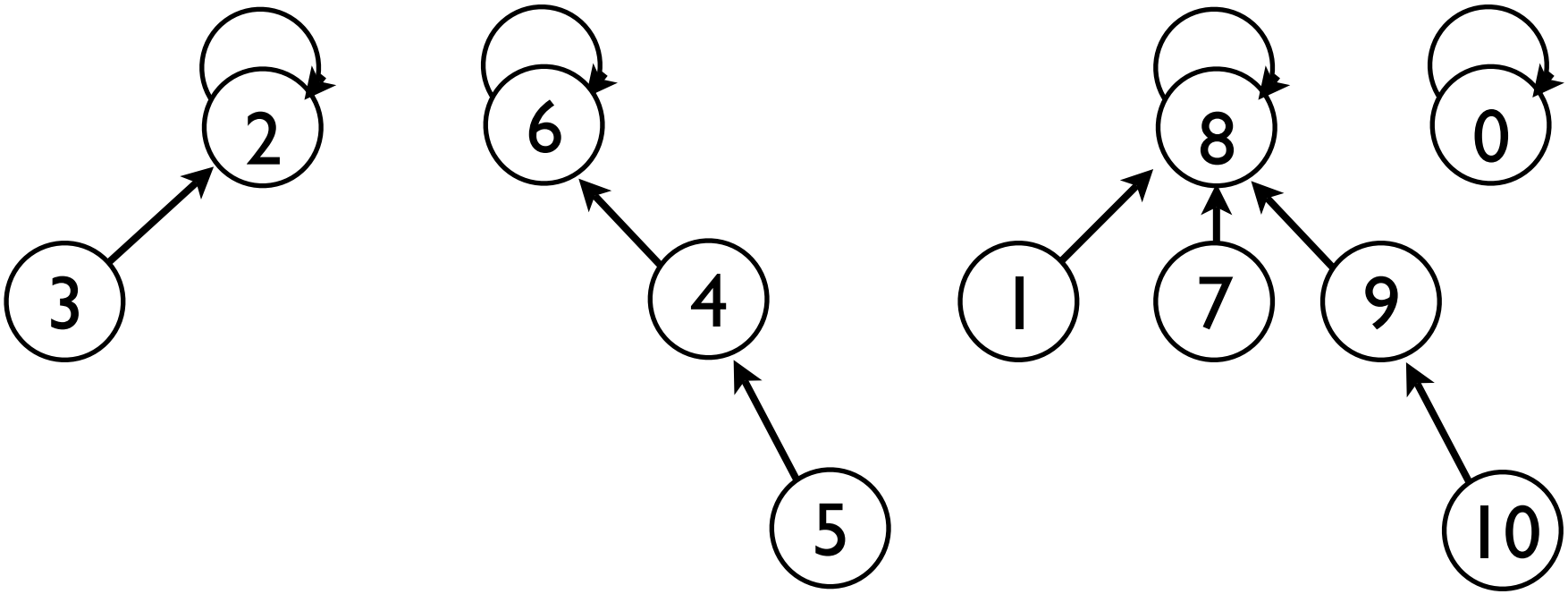


Graphenalgorithmen

Union-find-Problem - find

find(P,x)

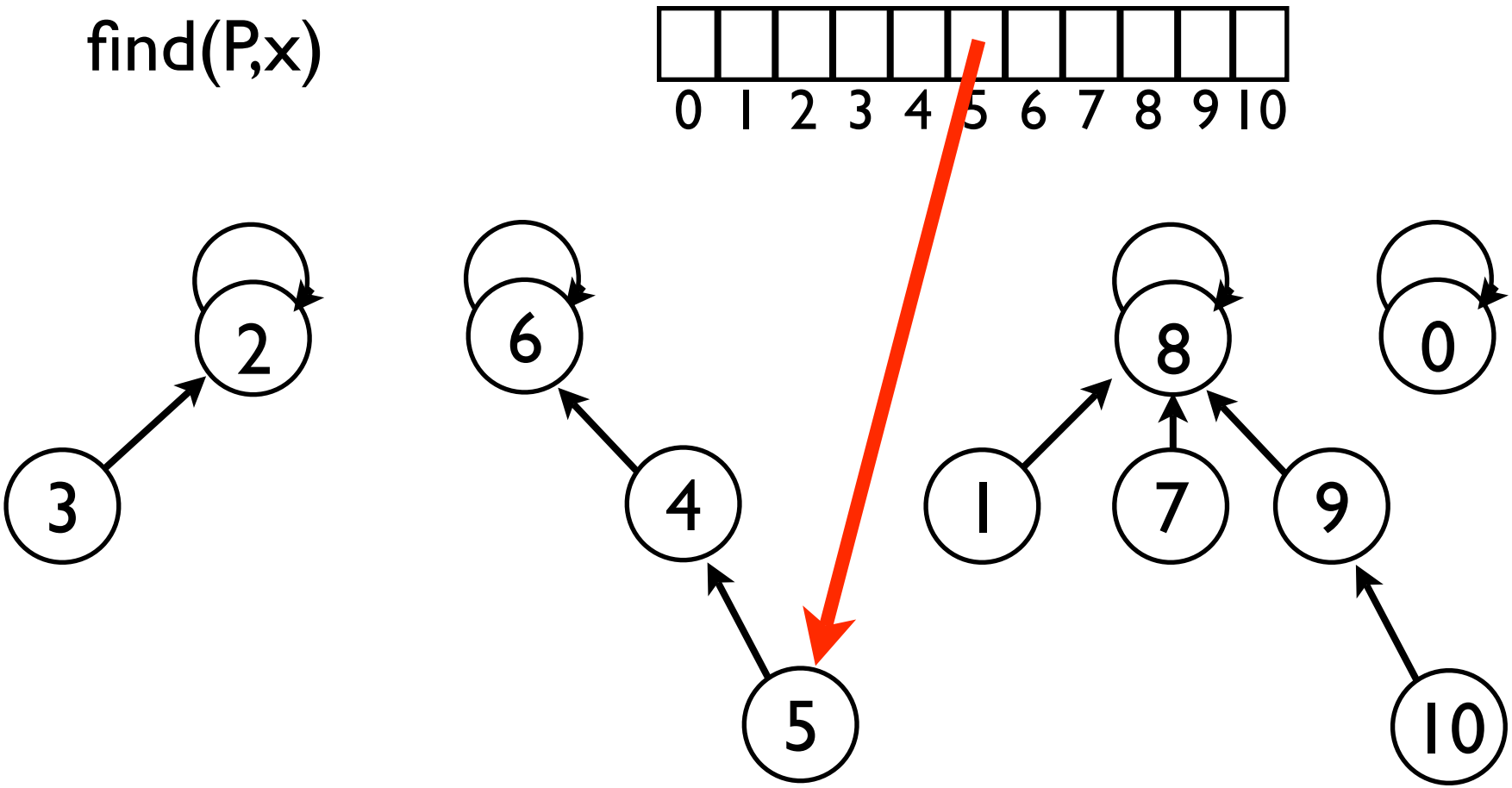




Graphenalgorithmen

Union-find-Problem - find

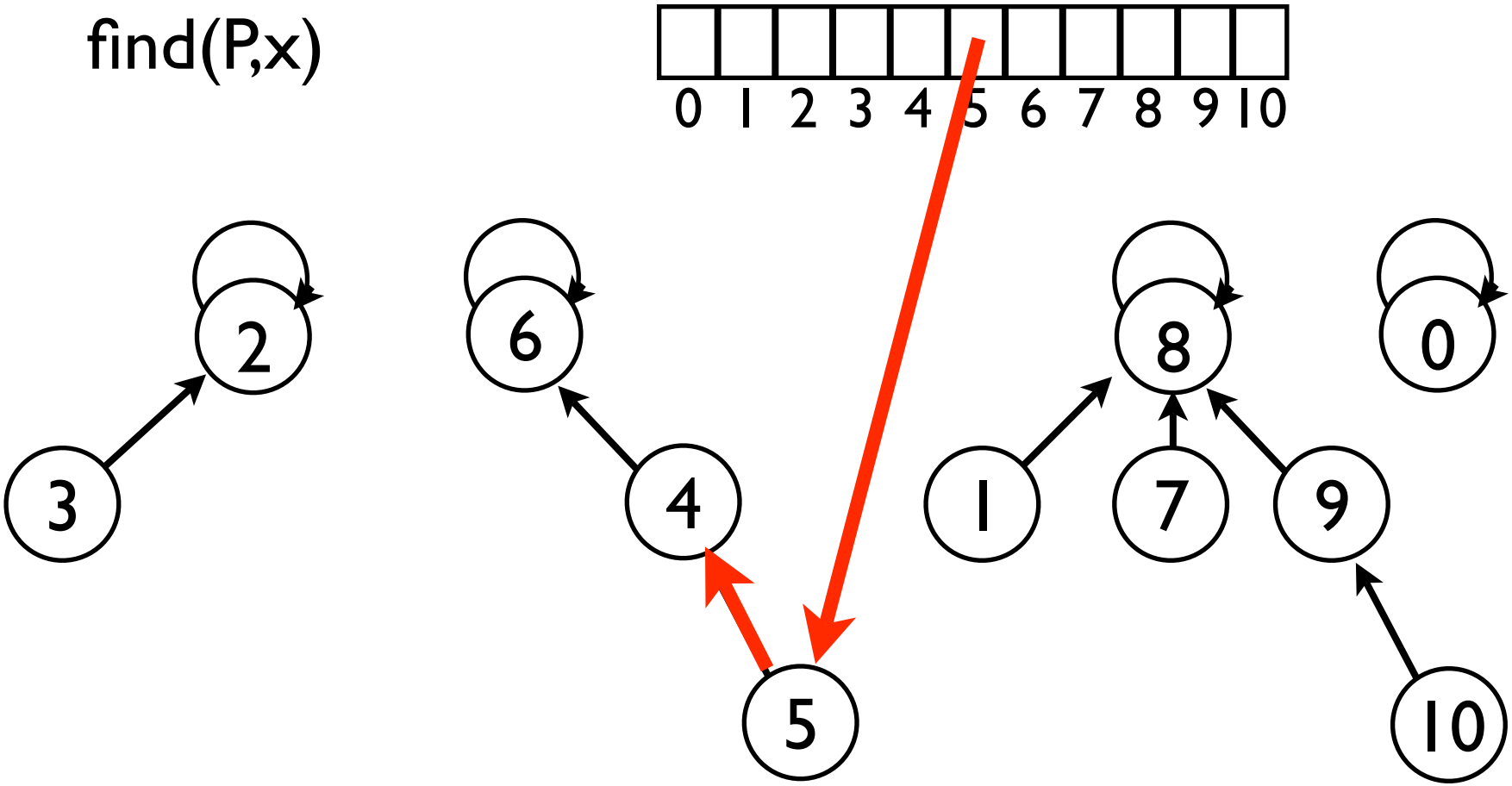
find(P,x)



Graphenalgorithmen

Union-find-Problem - find

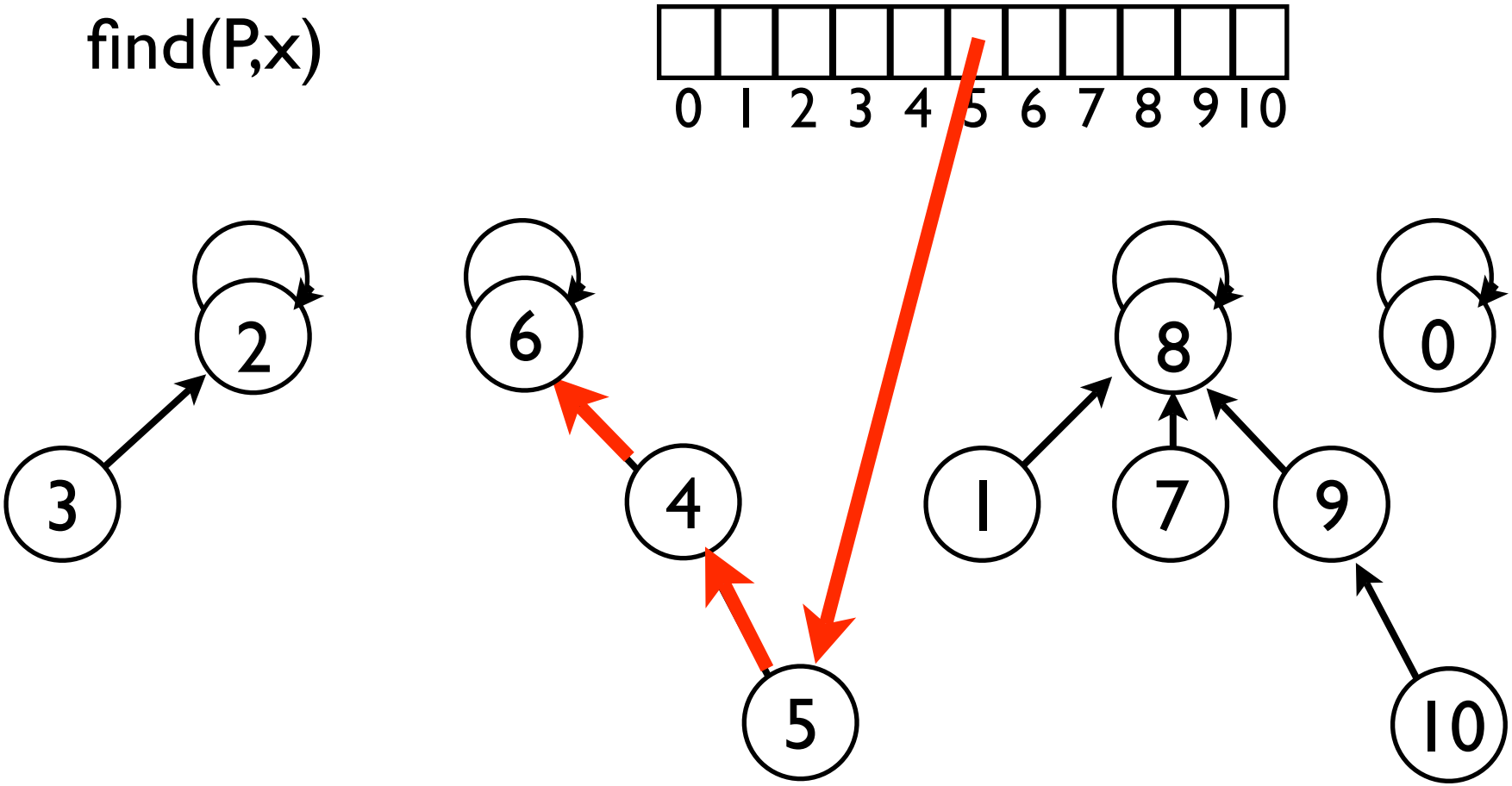
find(P,x)



Graphenalgorithmen

Union-find-Problem - find

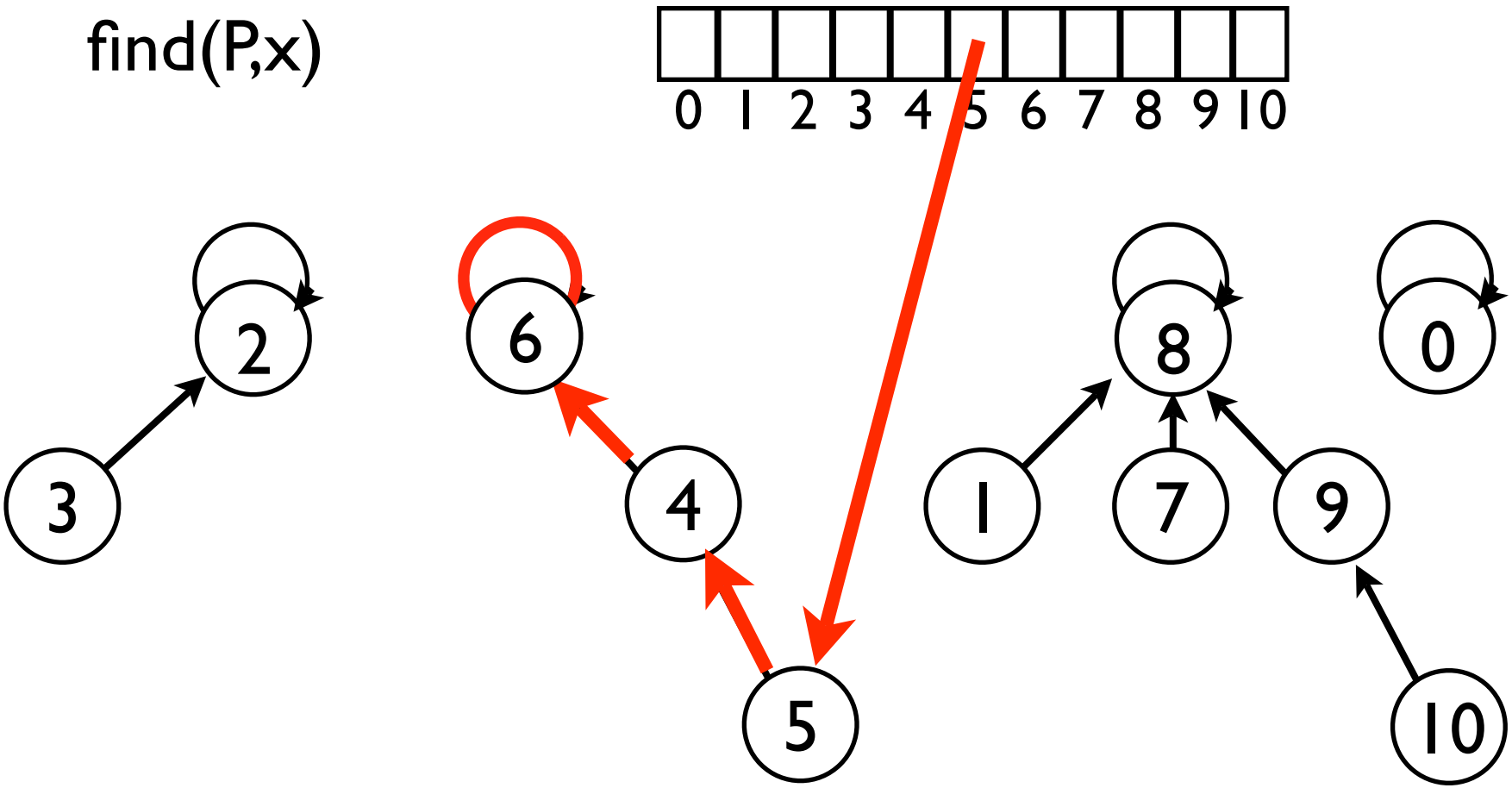
find(P,x)



Graphenalgorithmen

Union-find-Problem - find

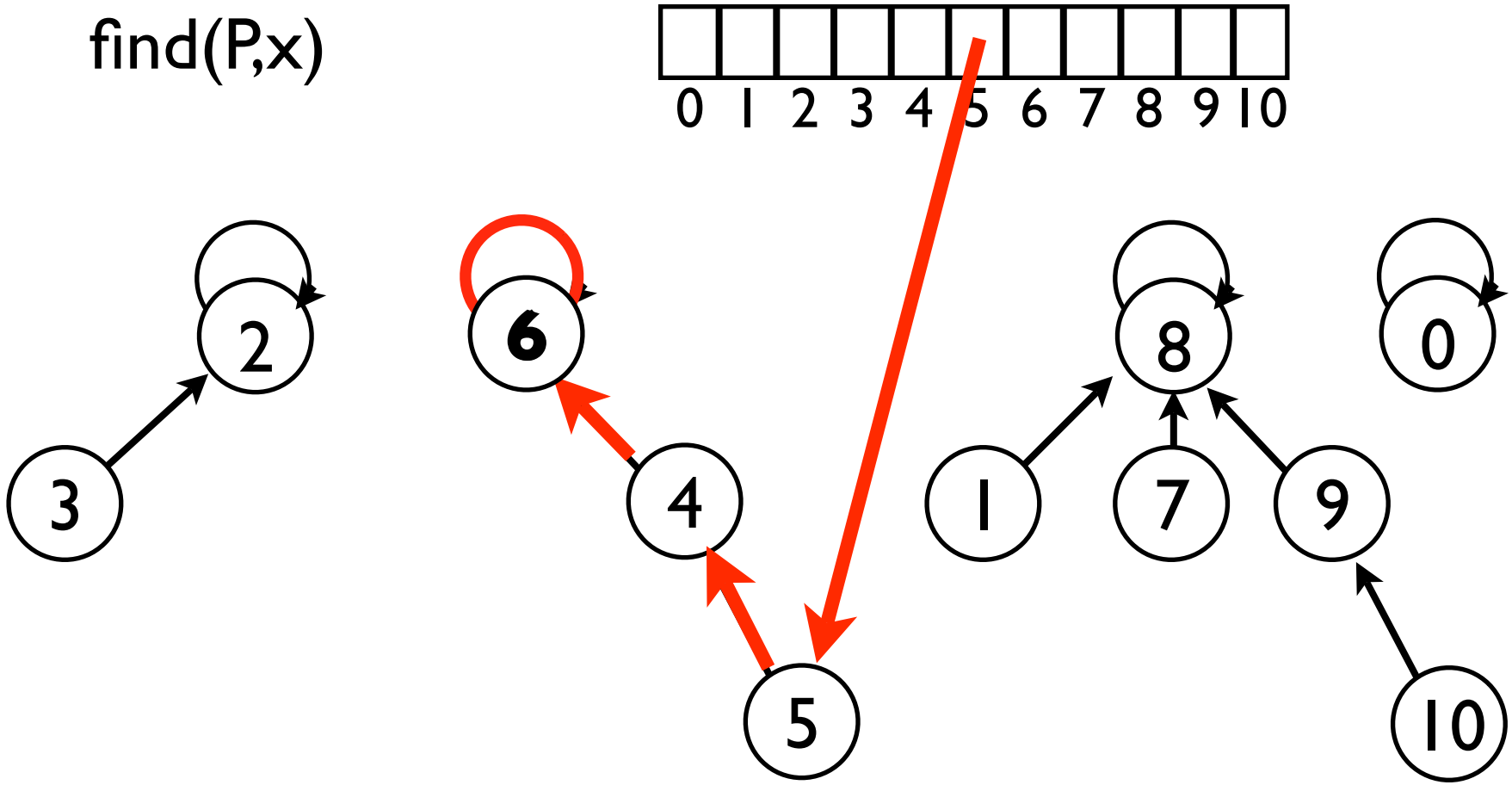
find(P,x)



Graphenalgorithmen

Union-find-Problem - find

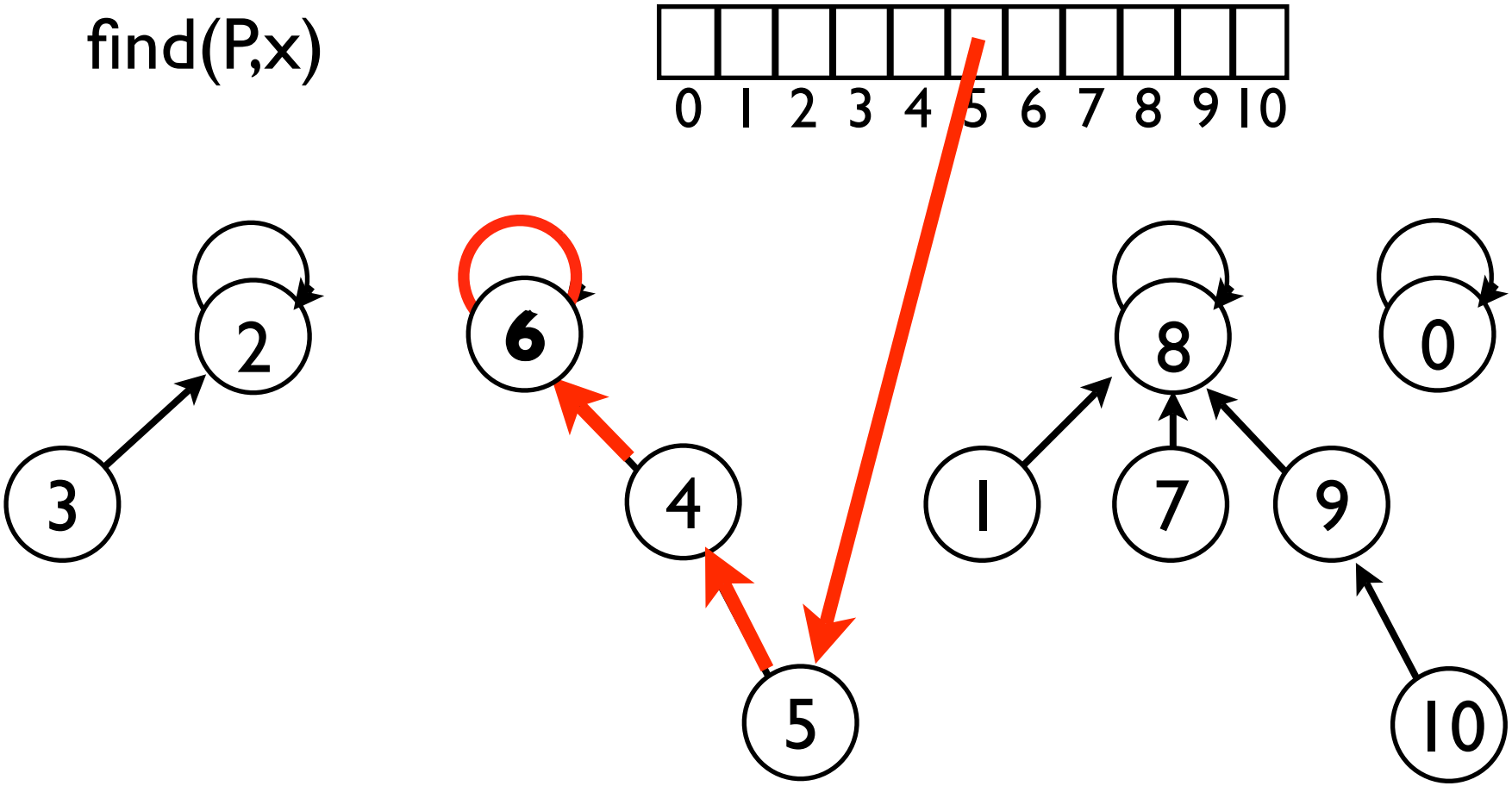
find(P,x)



Graphenalgorithmen

Union-find-Problem - find

find(P,x)

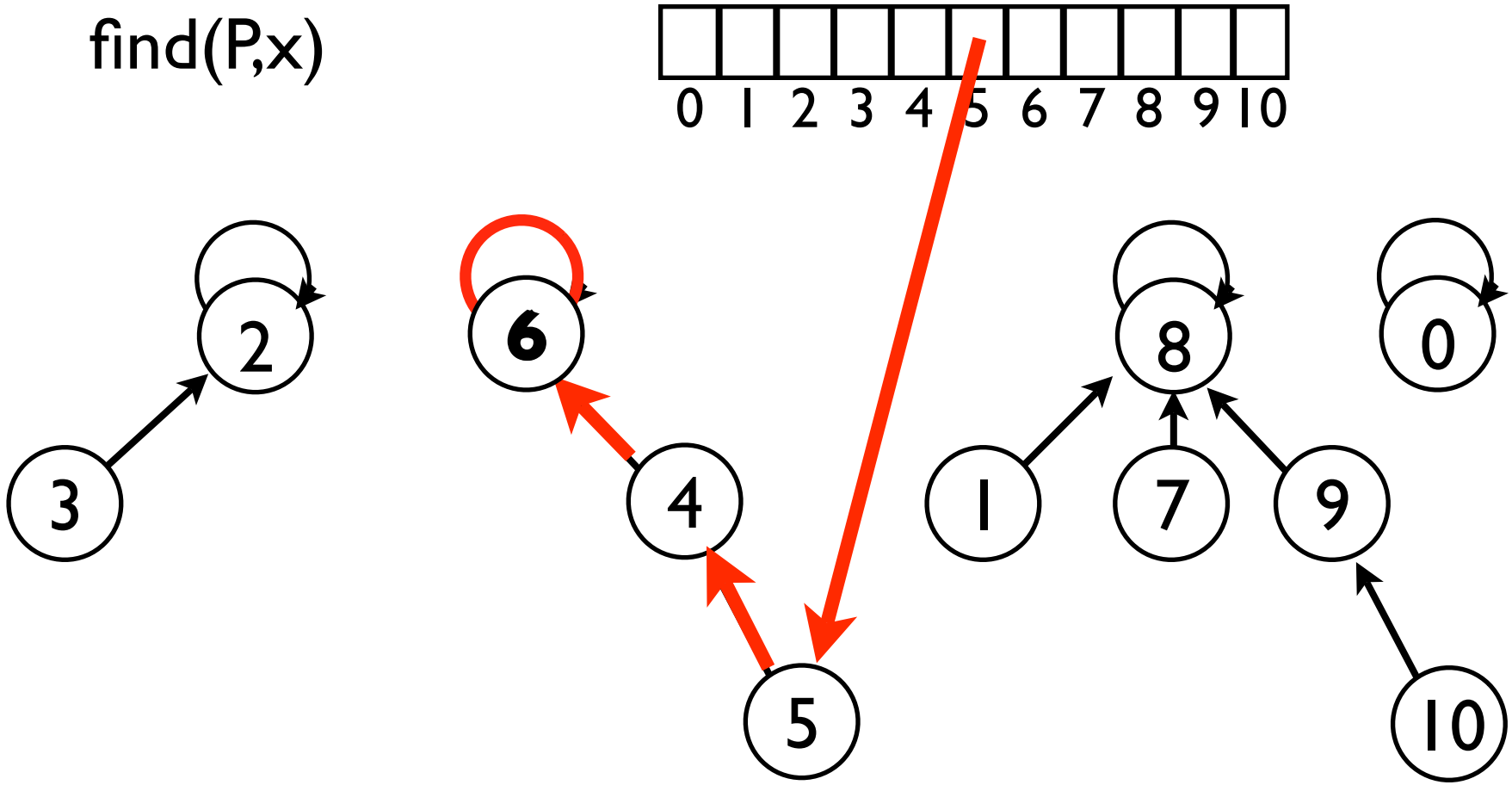


Laufzeit: O(h); h: Höhe des Baums, in dem x liegt

Graphenalgorithmen

Union-find-Problem - find

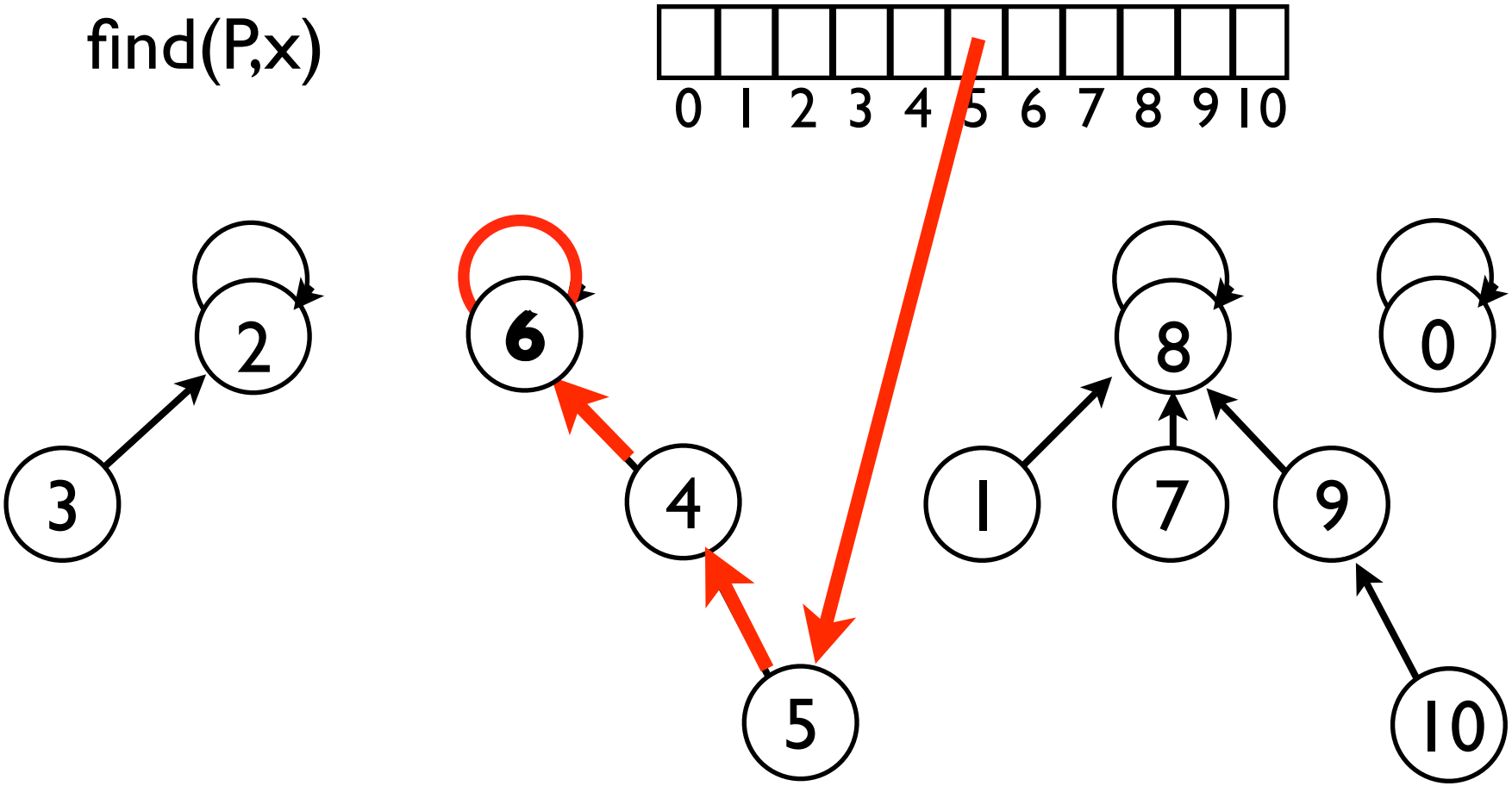
find(P,x)



Graphenalgorithmen

Union-find-Problem - find

find(P,x)



Wie hoch können die Bäume werden?

Graphenalgorithmen
Union-find-Problem - Algorithmen

```
var p: array[element] of element;
procedure union(p,a,b);
begin
    p[b]:=a
end;
function find(p,x): element;
var y: element;
begin
    y:=x;
    while p[y]≠y do y:=p[y];
    find:=y
end.
```

Graphenalgorithmen
Union-find-Problem - Baumhöhe

- Wie hoch können die Wurzelgraphen werden?
- Trick:
  - Mache bei einer Vereinigung zweier Bäume a und b die Wurzel des größeren (höheren) Baums zur Wurzel des gesamten Baums (bei gleichen Größen bel. wählen)
  - Konsequenz: Bäume wachsen nicht.
  - Notwendig: Höhenvariable in jedem (Wurzel-)Knoten

Graphenalgorithmen
Union-find-Problem - Baumhöhe

Lemma:

Die Operation union garantiert die folgende Eigenschaft aller Wurzelgraphen: Ein Baum der Höhe $h \geq 1$ hat wenigstens 2^{h-1} Knoten.

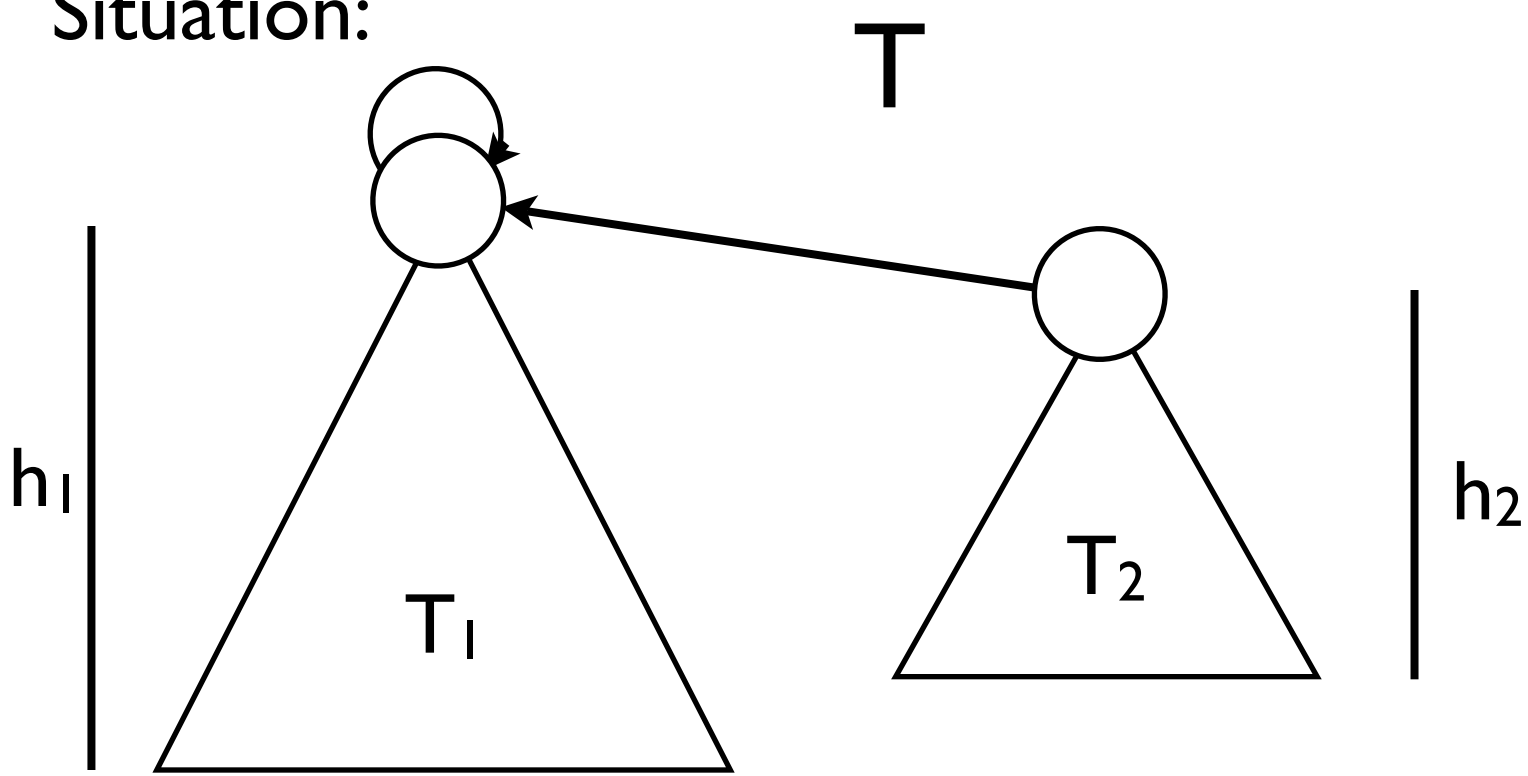
Beweis:

Seien T_1, T_2 zwei Bäume mit den Größen $g(T_1)$ und $g(T_2)$, die vereinigt werden sollen; h_1, h_2 seien Höhen von T_1 und T_2.

O.B.d.A. $g(T_1) \geq g(T_2)$.

Graphenalgorithmen
Union-find-Problem - Baumhöhe

Situation:

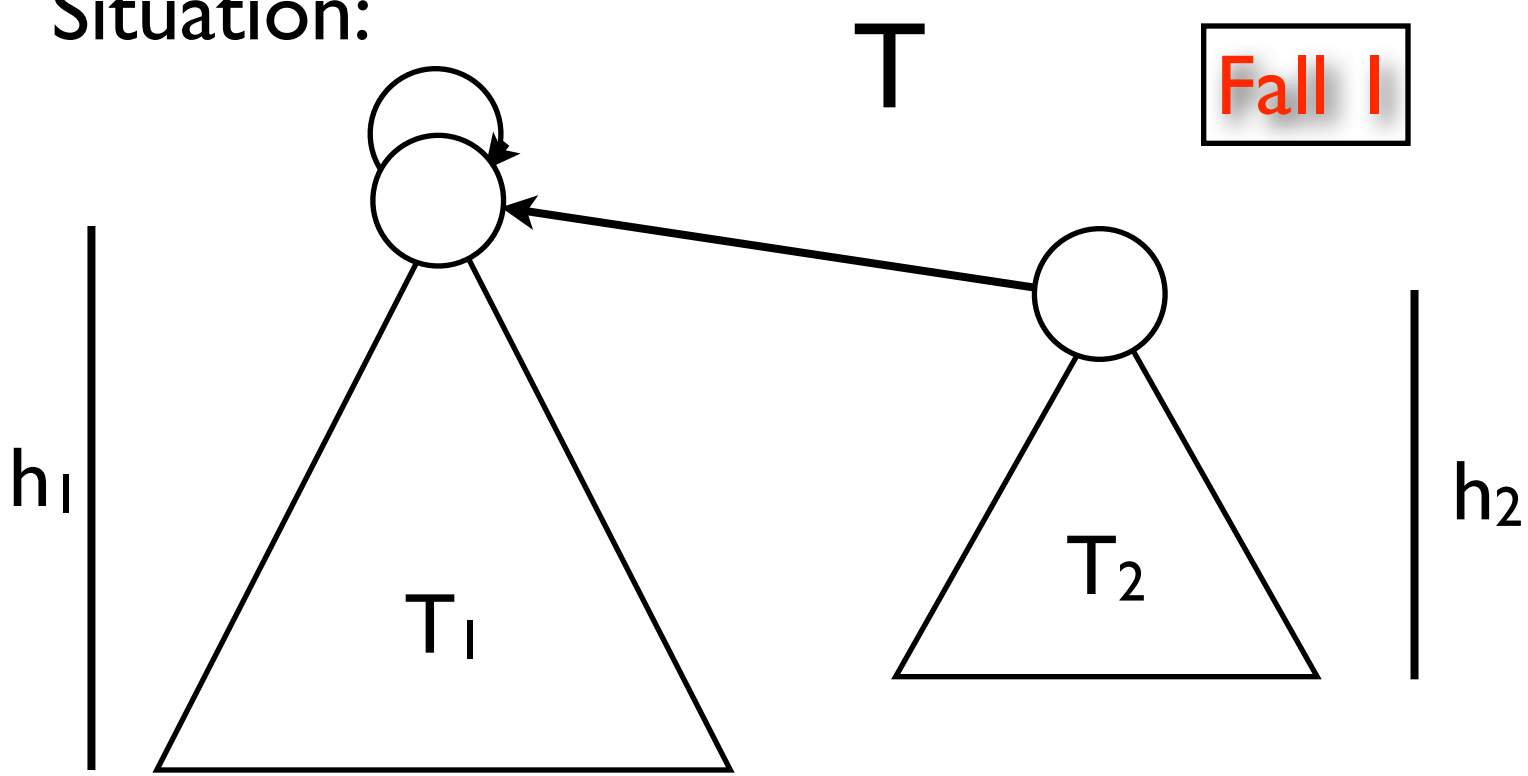


Nach Vor. gilt: T_i hat mind. 2^{h_i-1} Knoten, i=1,2.

Graphenalgorithmen

Union-find-Problem - Baumhöhe

Situation:

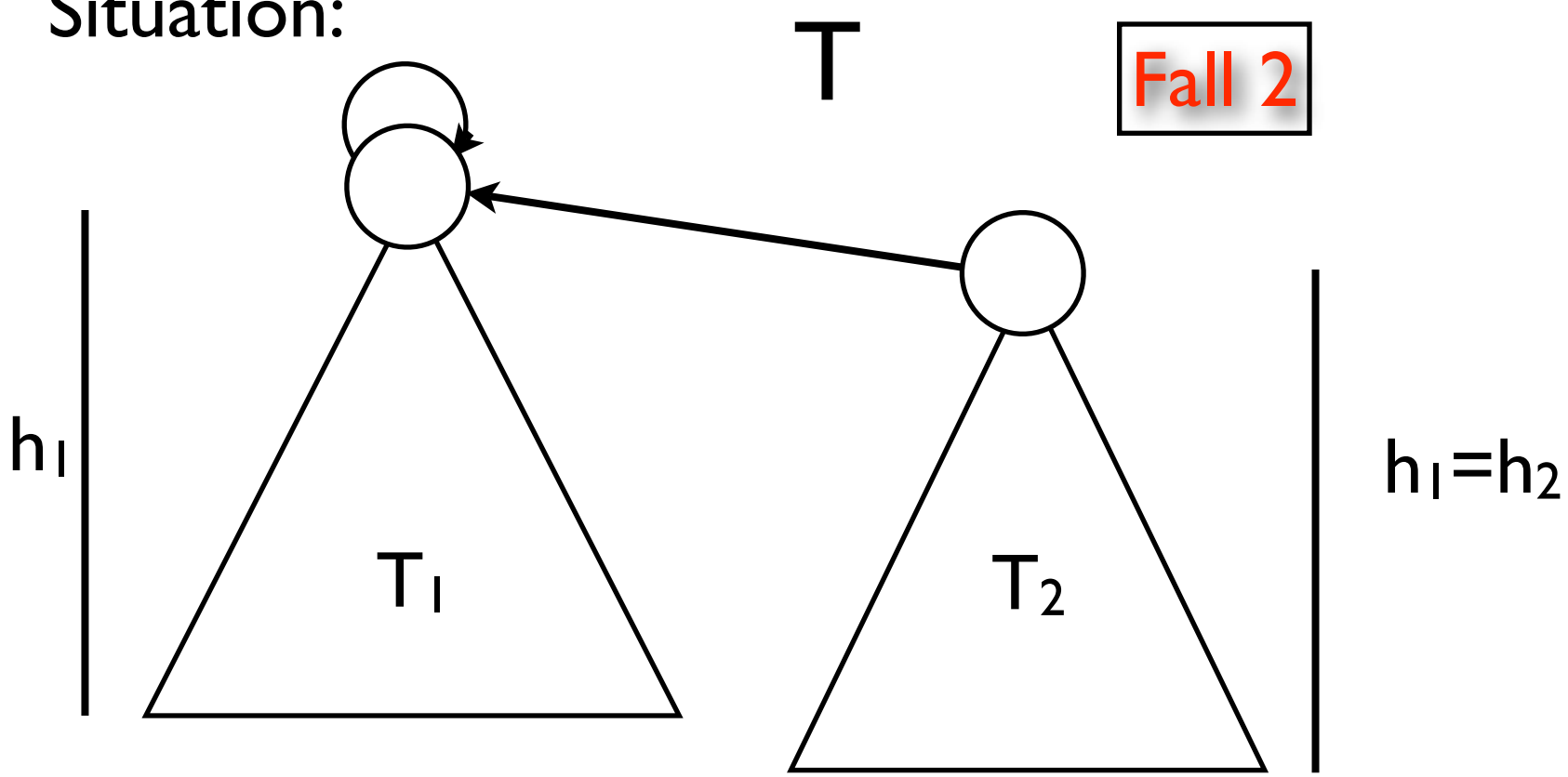


Nach Vor. gilt: T_i hat mind. 2^{h_i-1} Knoten, i=1,2.

Graphenalgorithmen

Union-find-Problem - Baumhöhe

Situation:



Nach Vor. gilt: T_i hat mind. 2^{h_i-1} Knoten, i=1,2.

Graphenalgorithmen
Union-find-Problem - Baumhöhe

Fall 1: $h(T)=\max\{h_1,h_2\}=h_1$

Dann: $g(T)=g(T_1)+g(T_2)\geq 2^{h_1-1}+2^{h_2-1}\geq 2^{h-1}$

Fall 2: $h(T)=\max\{h_1,h_2\}+1$

Dann gilt wegen der Annahmen $h(T)=h_2+1$.

Ferner: $g(T_1)\geq g(T_2)\geq 2^{h_2-1}$. Also

$g(T)=g(T1)+g(T2)\geq 2\cdot 2^{h_2-1}=2^{h_2}$.

Graphenalgorithmen
Union-find-Problem - Baumhöhe

Folgerung:

Durch iterierte Vereinigung gilt bei einer Anfangsmenge von n Objekten in der Grundmenge S, also n einzelnen Mengen (Wurzelgraphen) mit je einem Objekt (Knoten): Alle entstehenden Bäume haben Höhen $h \leq \log_2 n$.

Folgerung: find(P,x) läuft in O(log n) Operationen.

Graphenalgorithmen
Union-find-Problem - Speicher

Bemerkung zum Speicher:

O(n log n) zusätzliche Bits für die Größe jedes Teilbaums.

Optimierung:

Speichere nicht Größen, sondern Höhen im Wurzelknoten: O(n log log n) zusätzliche Bits.

Graphenalgorithmen
Union-find-Problem - Pfadkompression

Beobachtung:

Folgen von n find-Operationen benötigen Aufwand O(n log n).

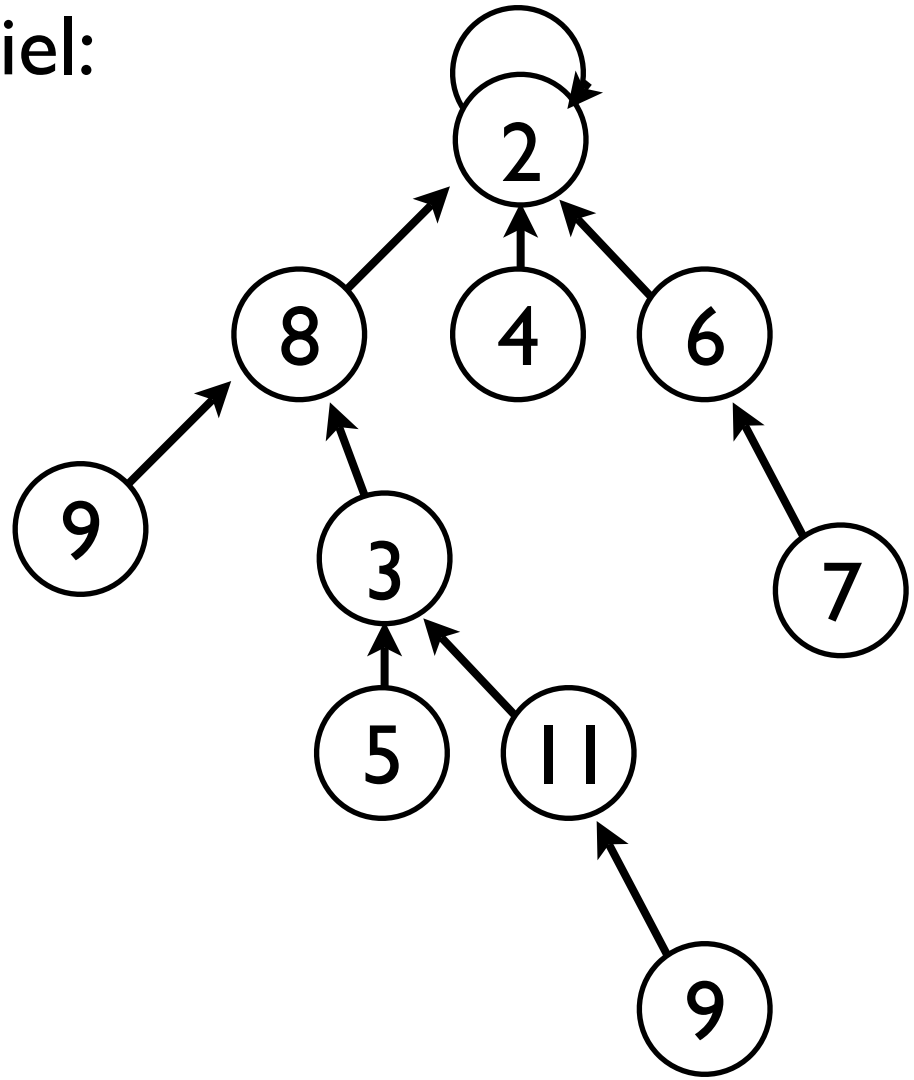
Idee:

Verkürze bei jedem find die Länge der durchlaufenen Wege auf 1, indem alle Knoten zu direkten Söhnen der Wurzel gemacht werden.

Graphenalgorithmen

Union-find-Problem - Pfadkompression

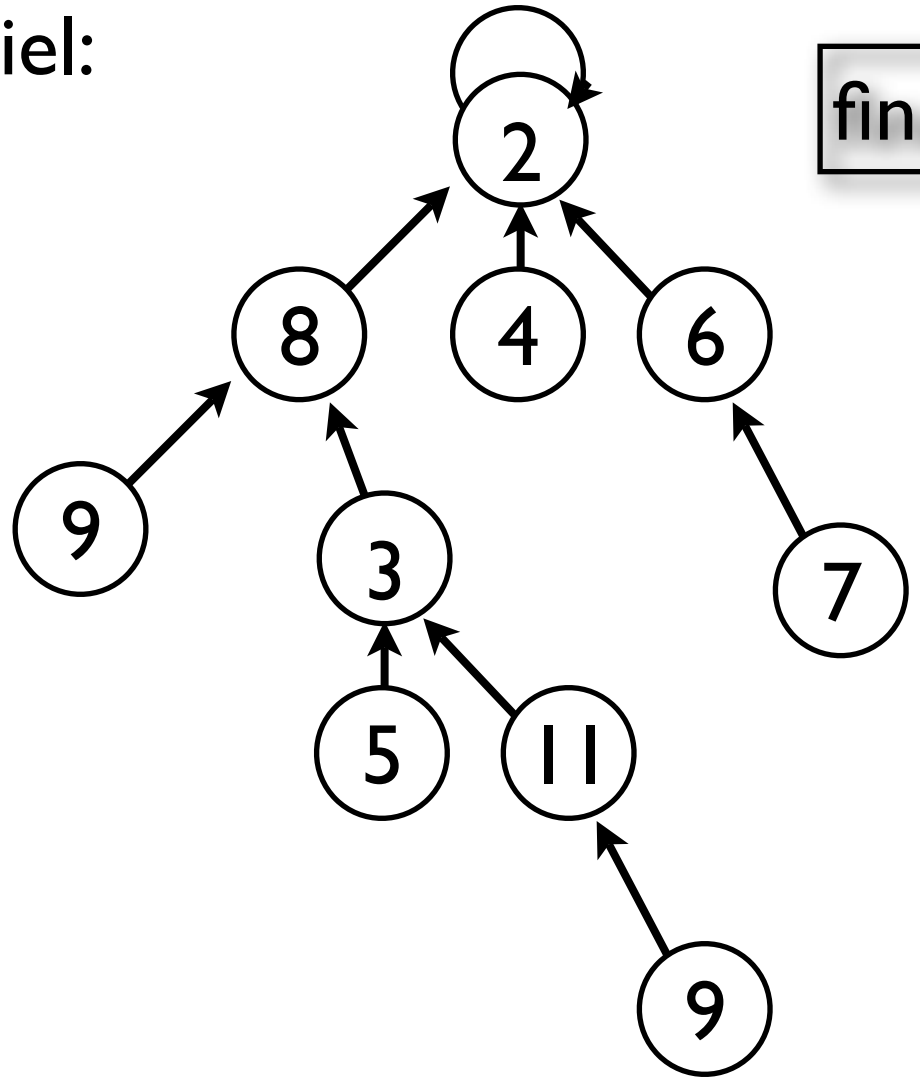
Beispiel:



Graphenalgorithmen

Union-find-Problem - Pfadkompression

Beispiel:

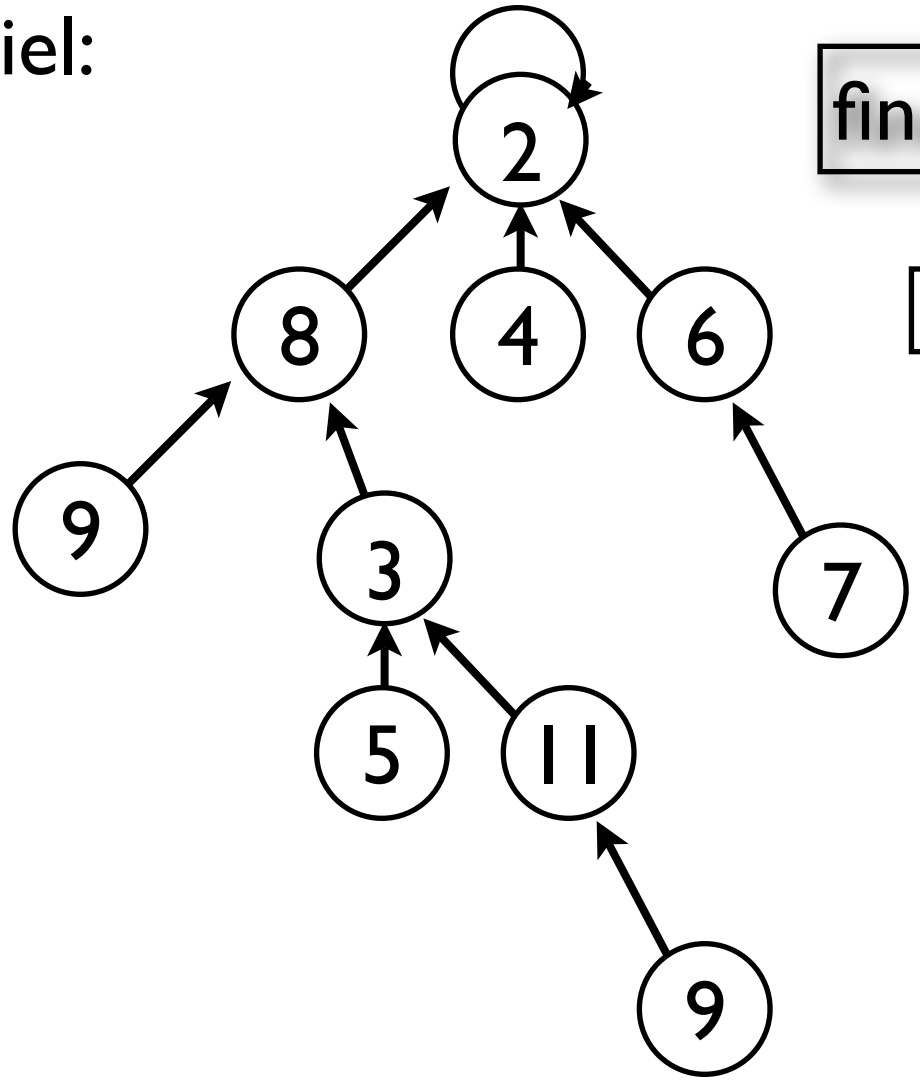


find(11)→2

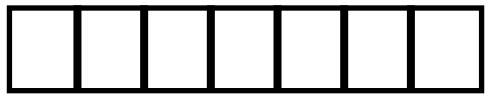
Graphenalgorithmen

Union-find-Problem - Pfadkompression

Beispiel:



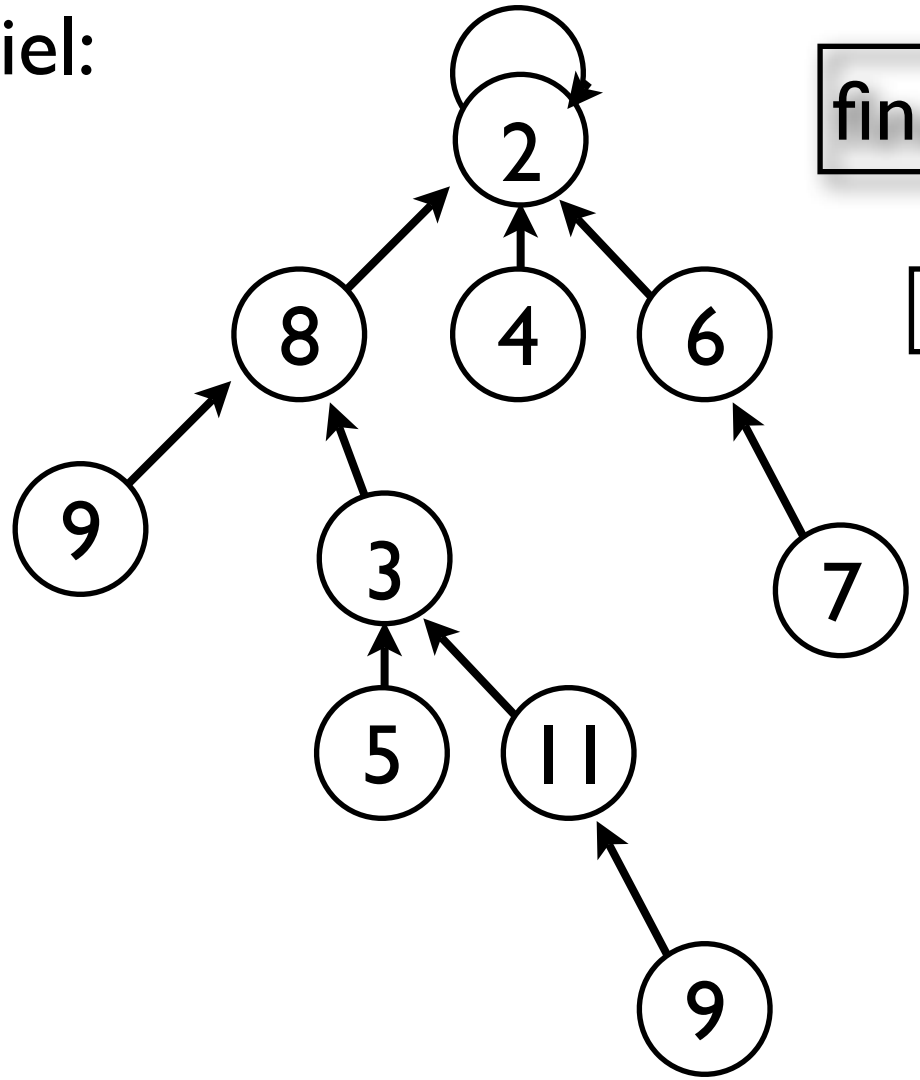
find(11)→2



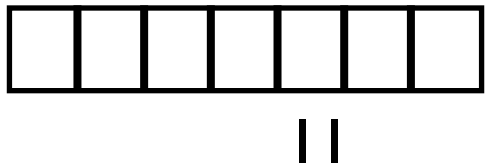
Graphenalgorithmen

Union-find-Problem - Pfadkompression

Beispiel:



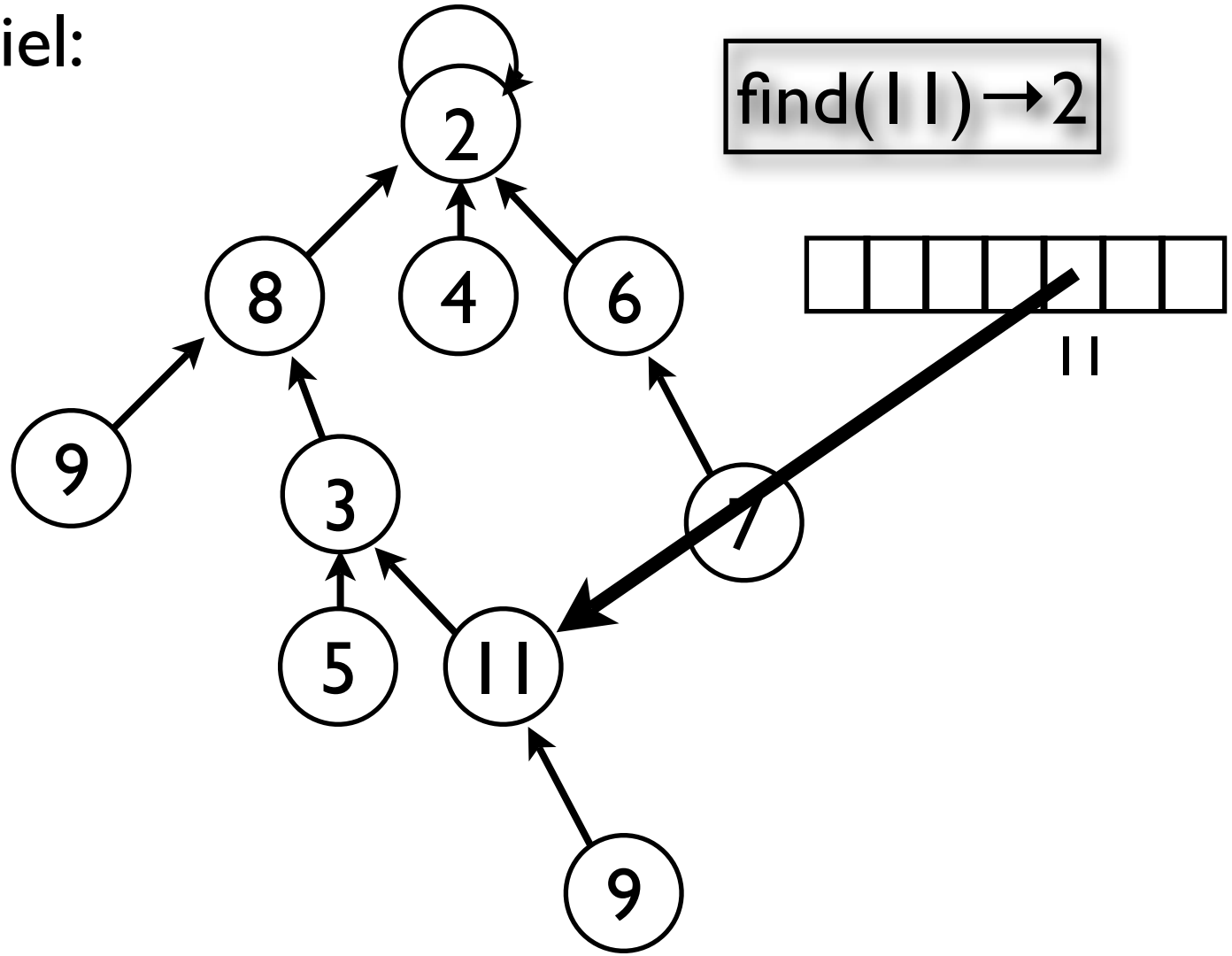
find(11)→2



Graphenalgorithmen

Union-find-Problem - Pfadkompression

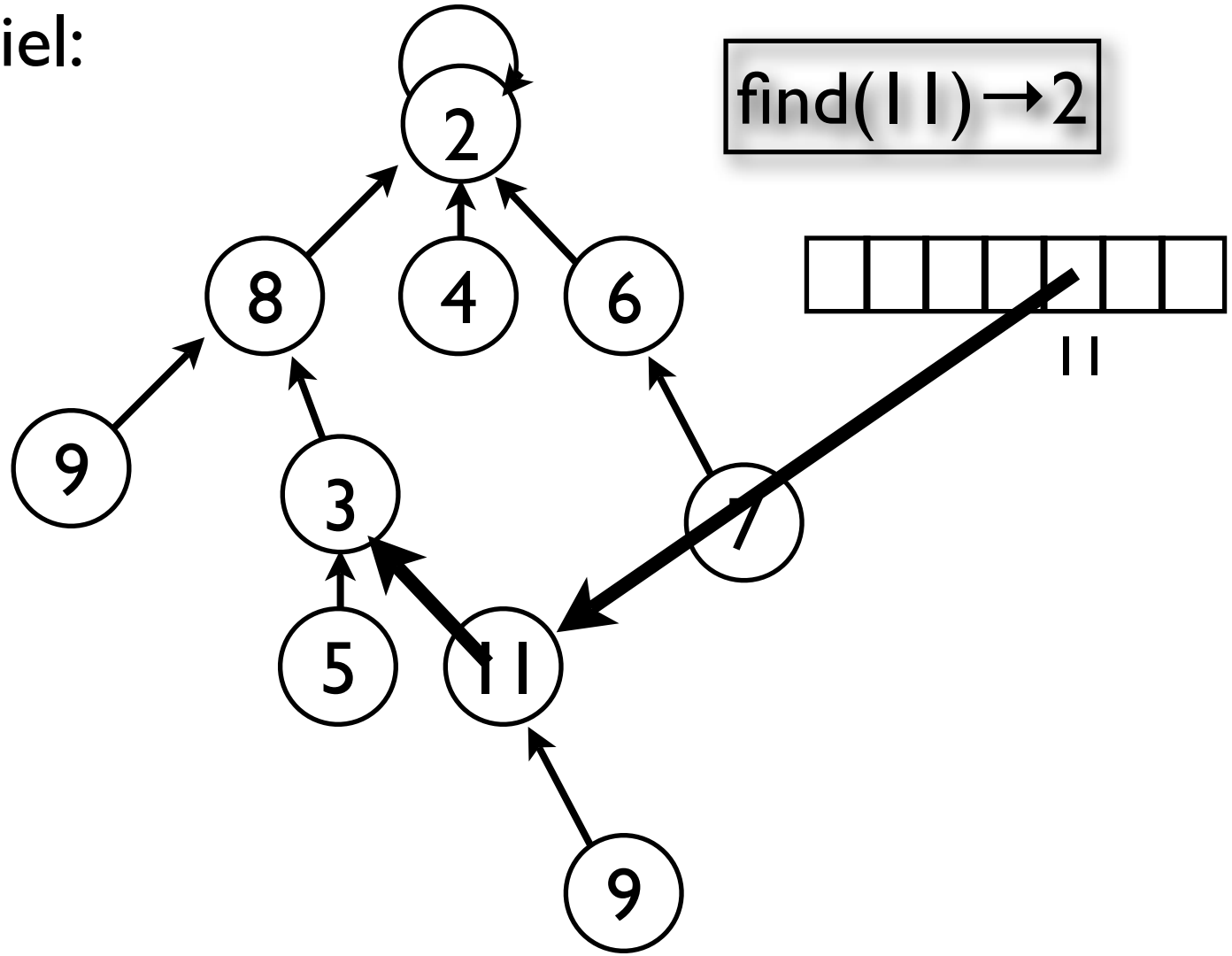
Beispiel:



Graphenalgorithmen

Union-find-Problem - Pfadkompression

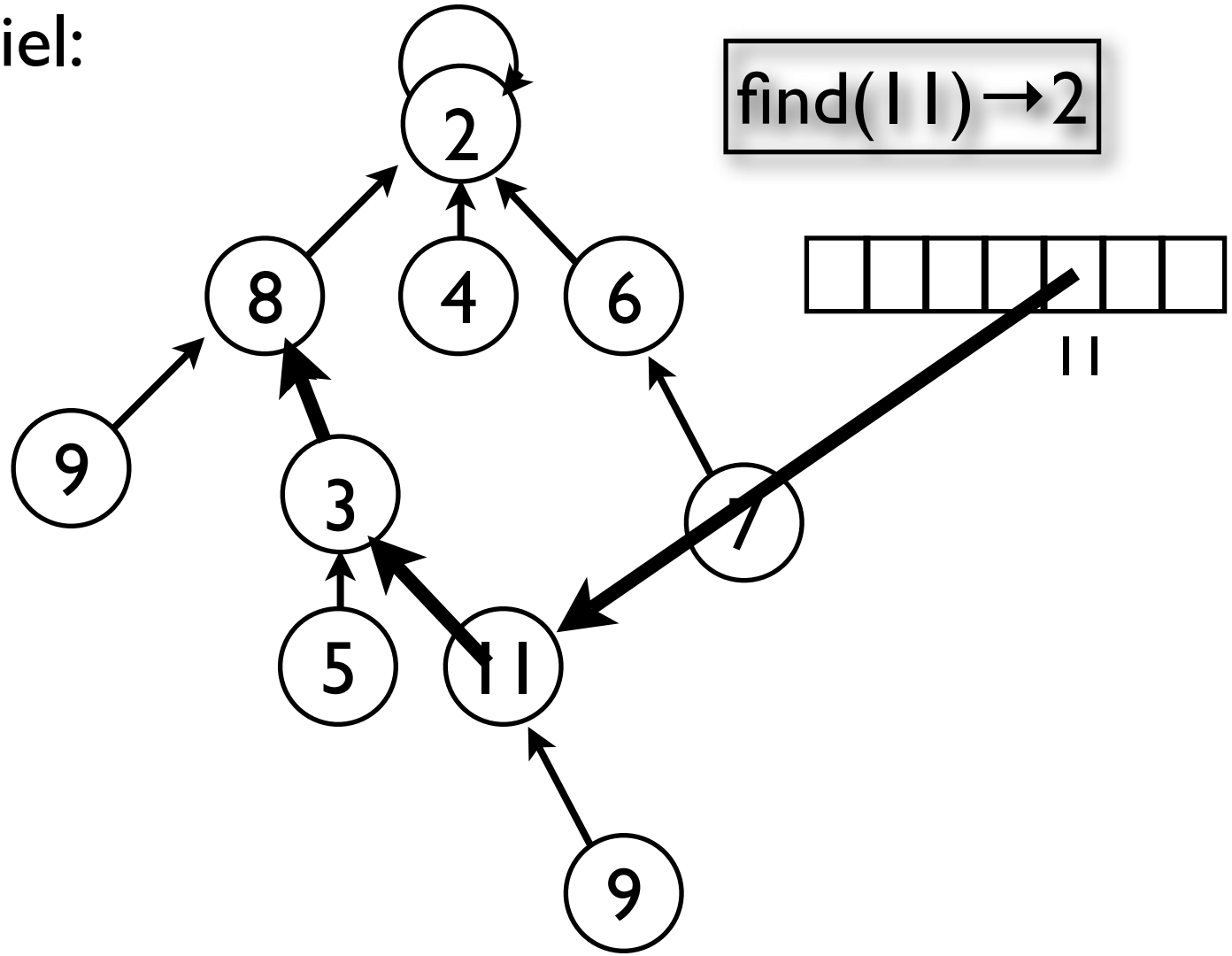
Beispiel:



Graphenalgorithmen

Union-find-Problem - Pfadkompression

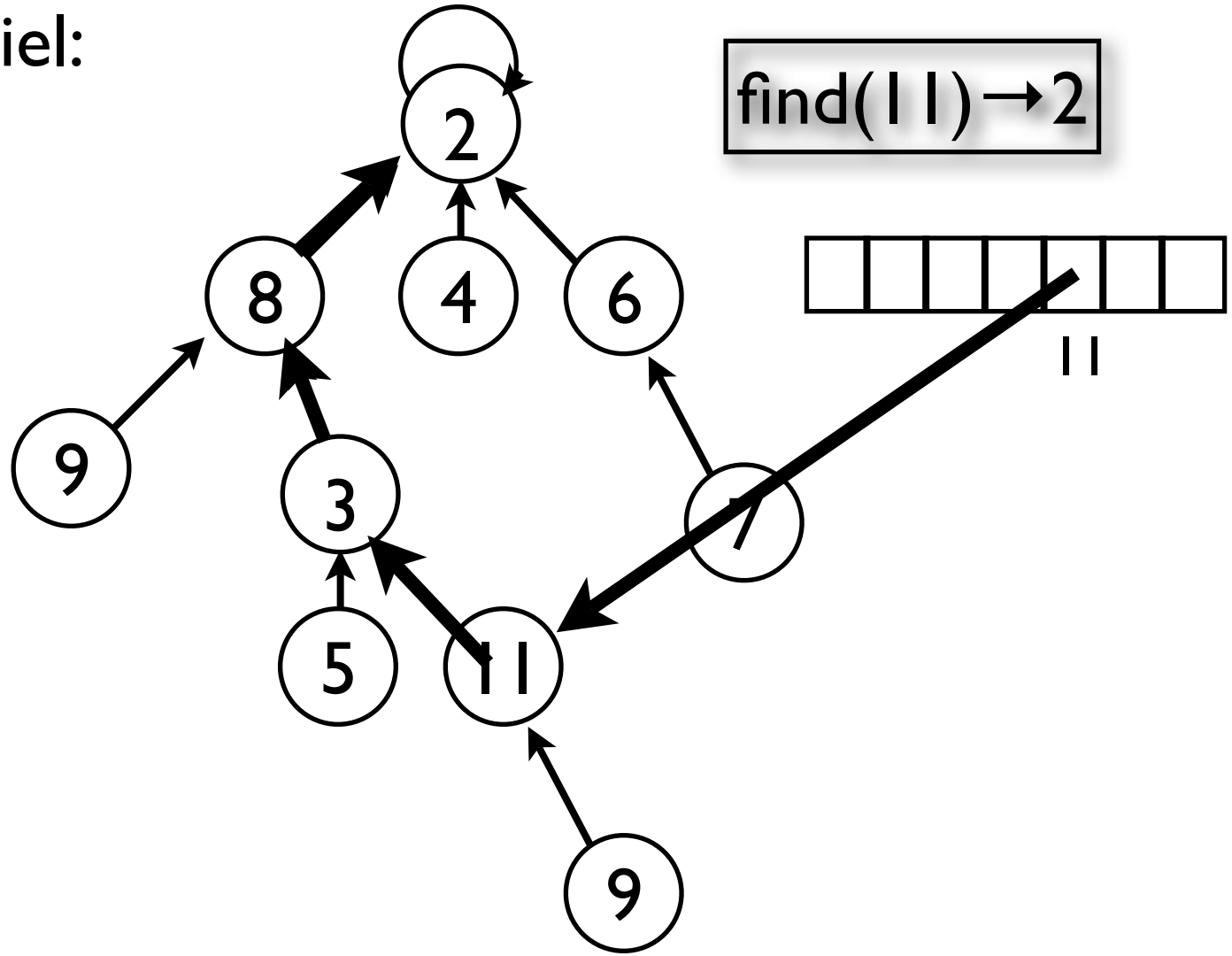
Beispiel:



Graphenalgorithmen

Union-find-Problem - Pfadkompression

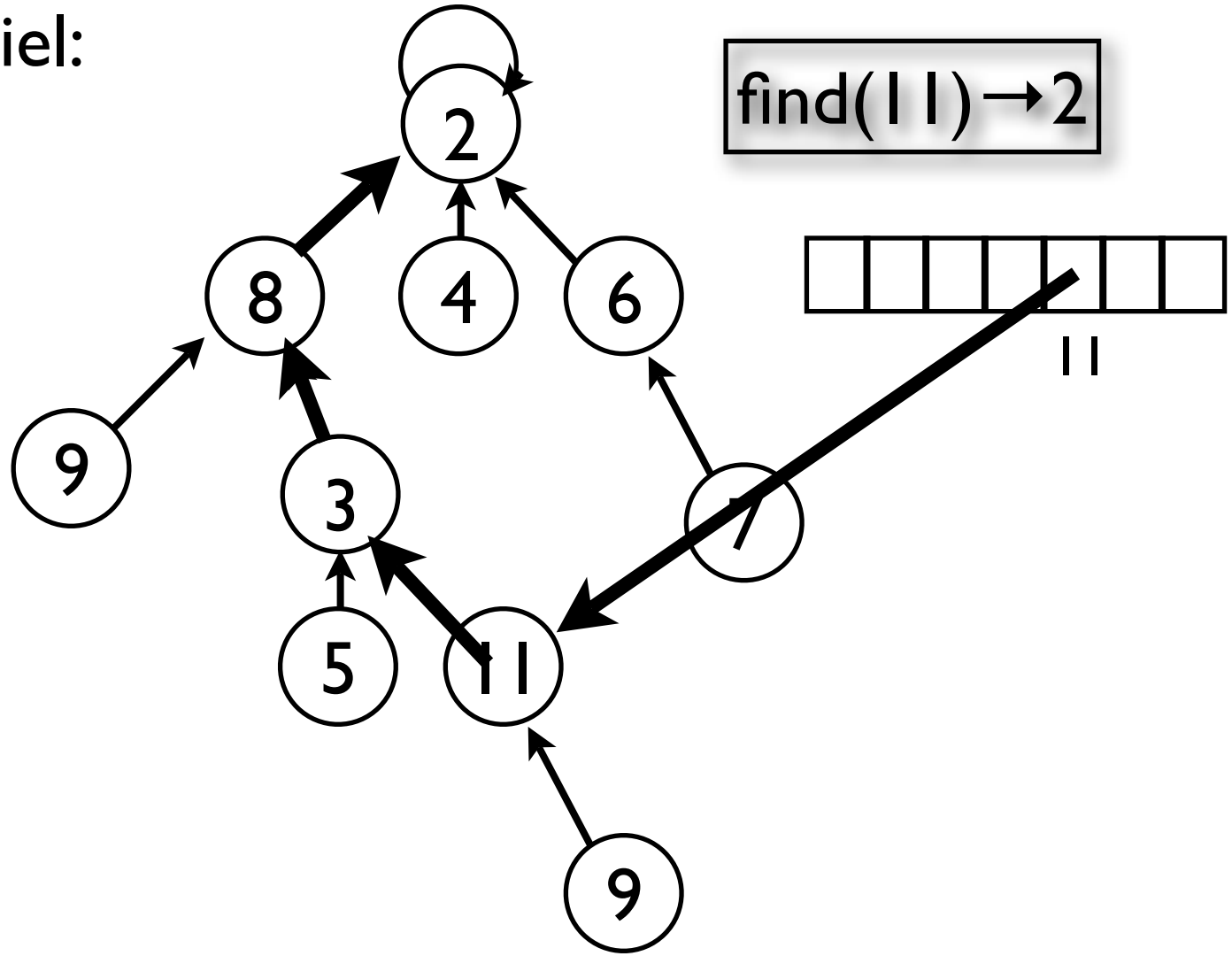
Beispiel:



Graphenalgorithmen

Union-find-Problem - Pfadkompression

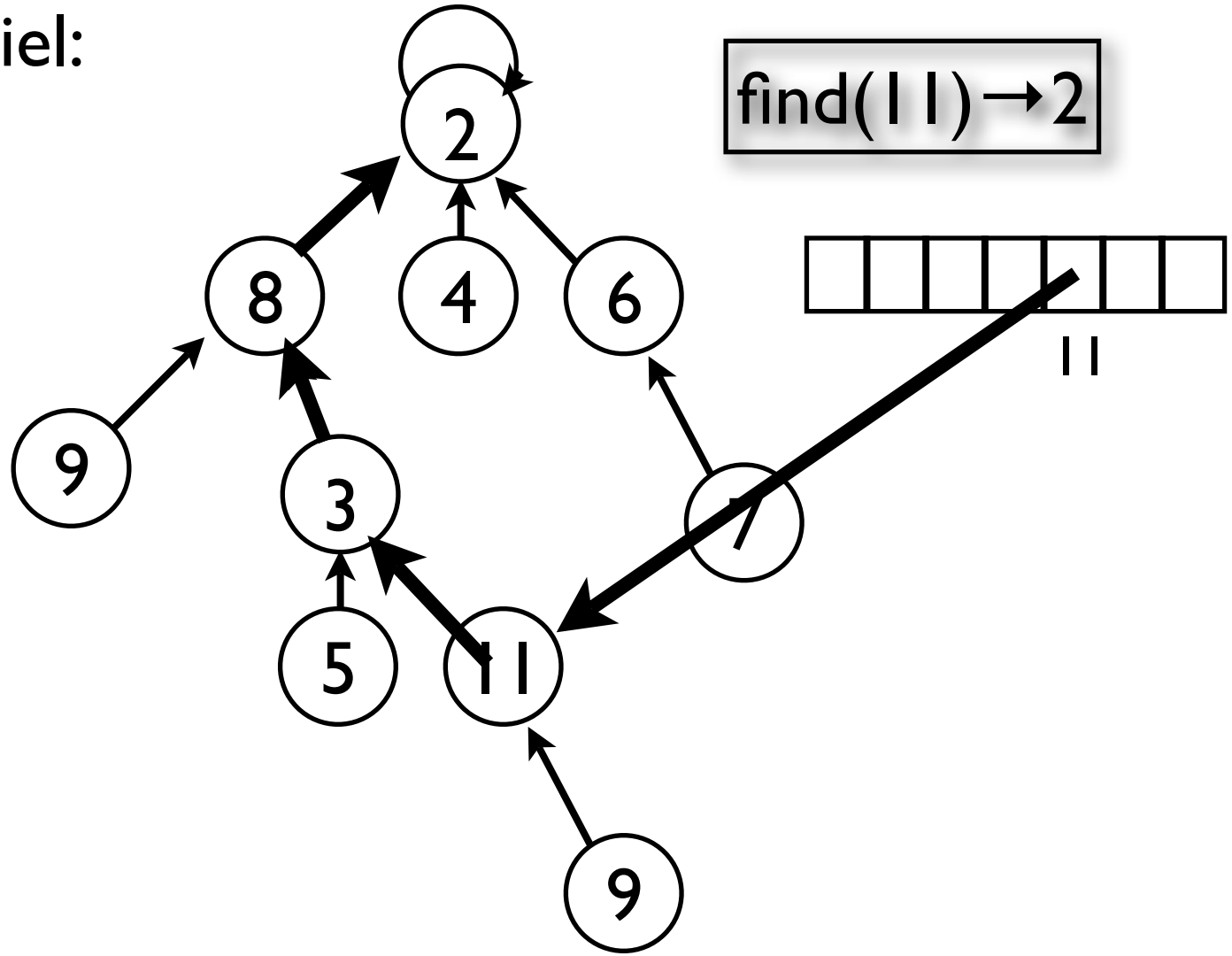
Beispiel:



Graphenalgorithmen

Union-find-Problem - Pfadkompression

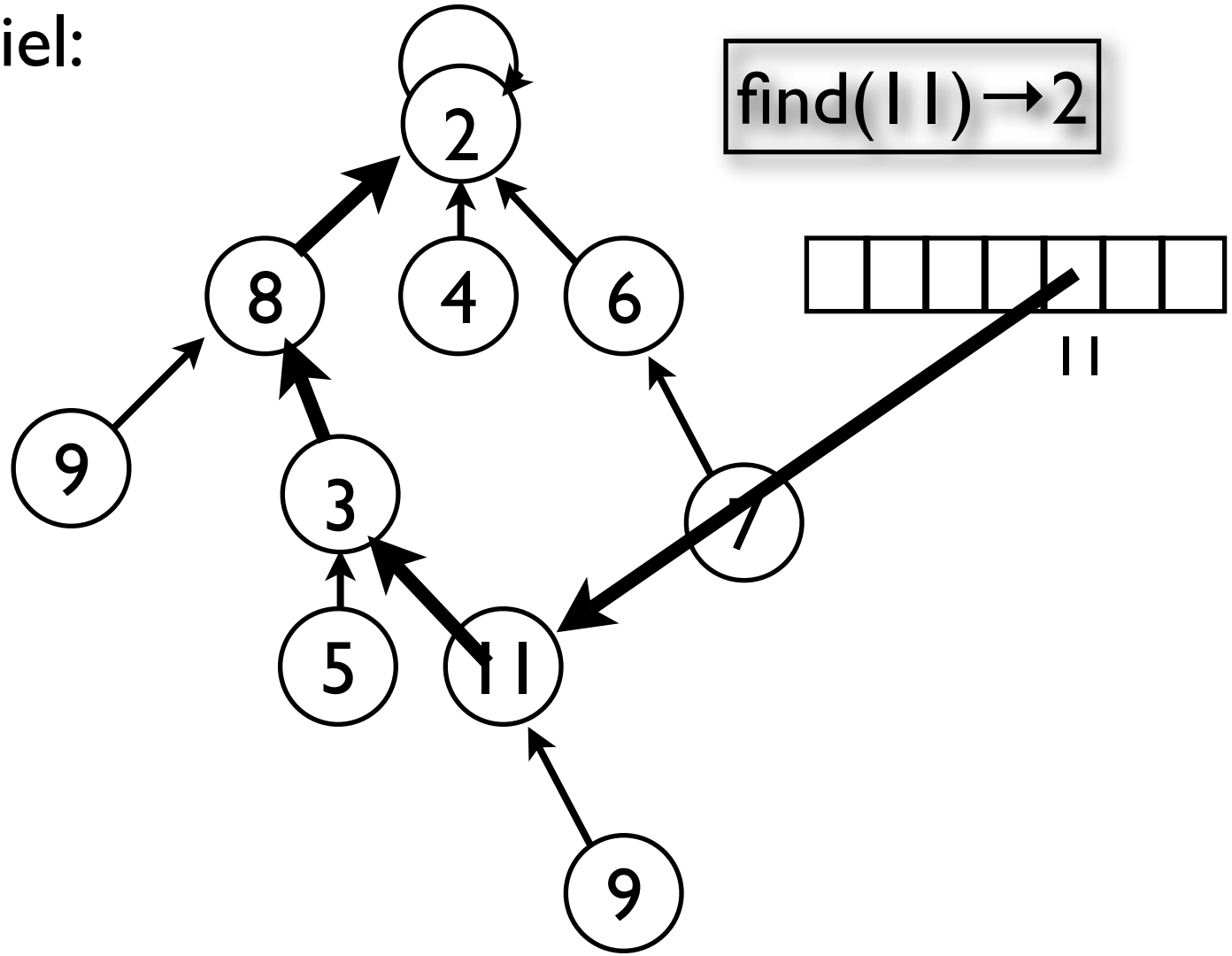
Beispiel:



Graphenalgorithmen

Union-find-Problem - Pfadkompression

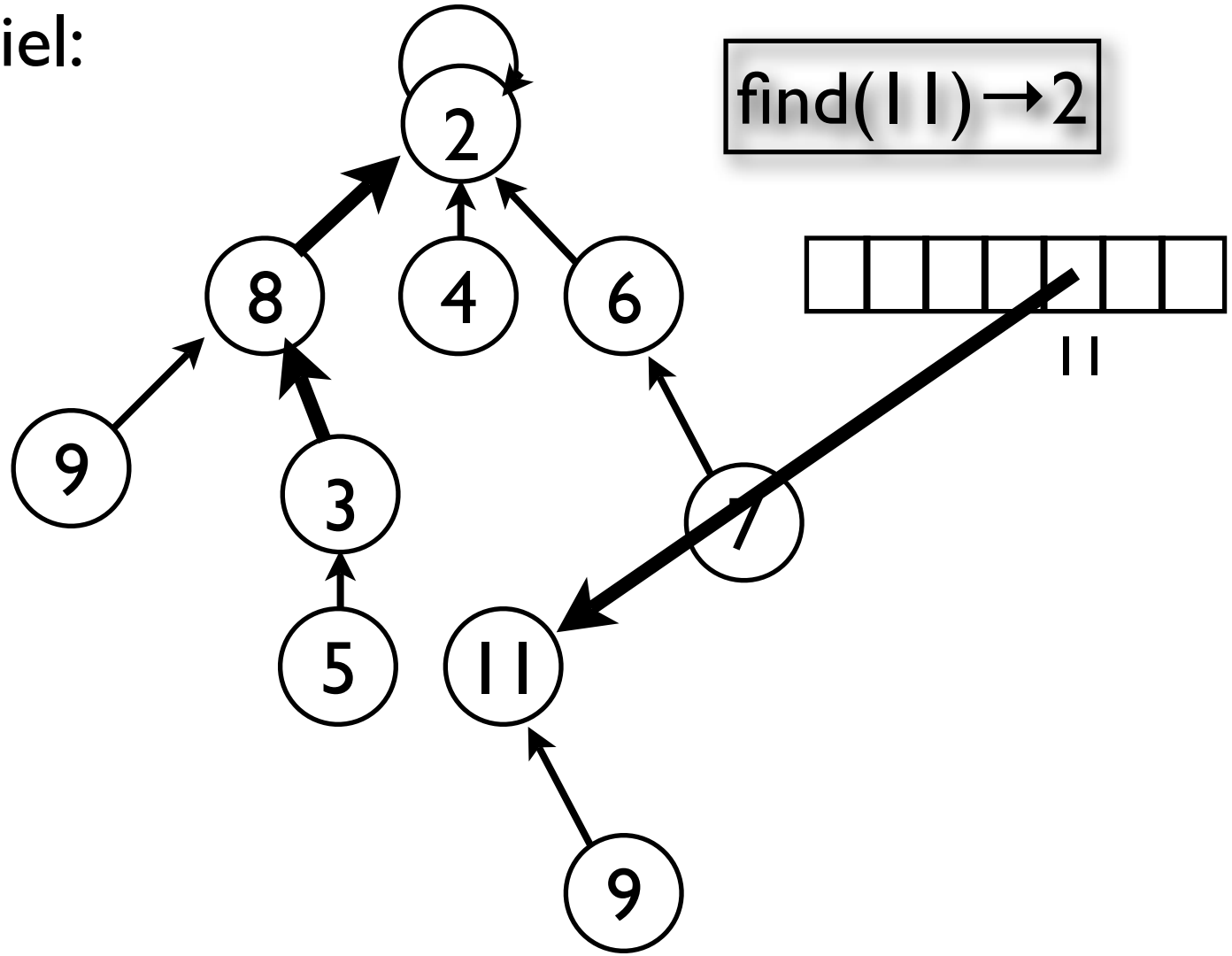
Beispiel:



Graphenalgorithmen

Union-find-Problem - Pfadkompression

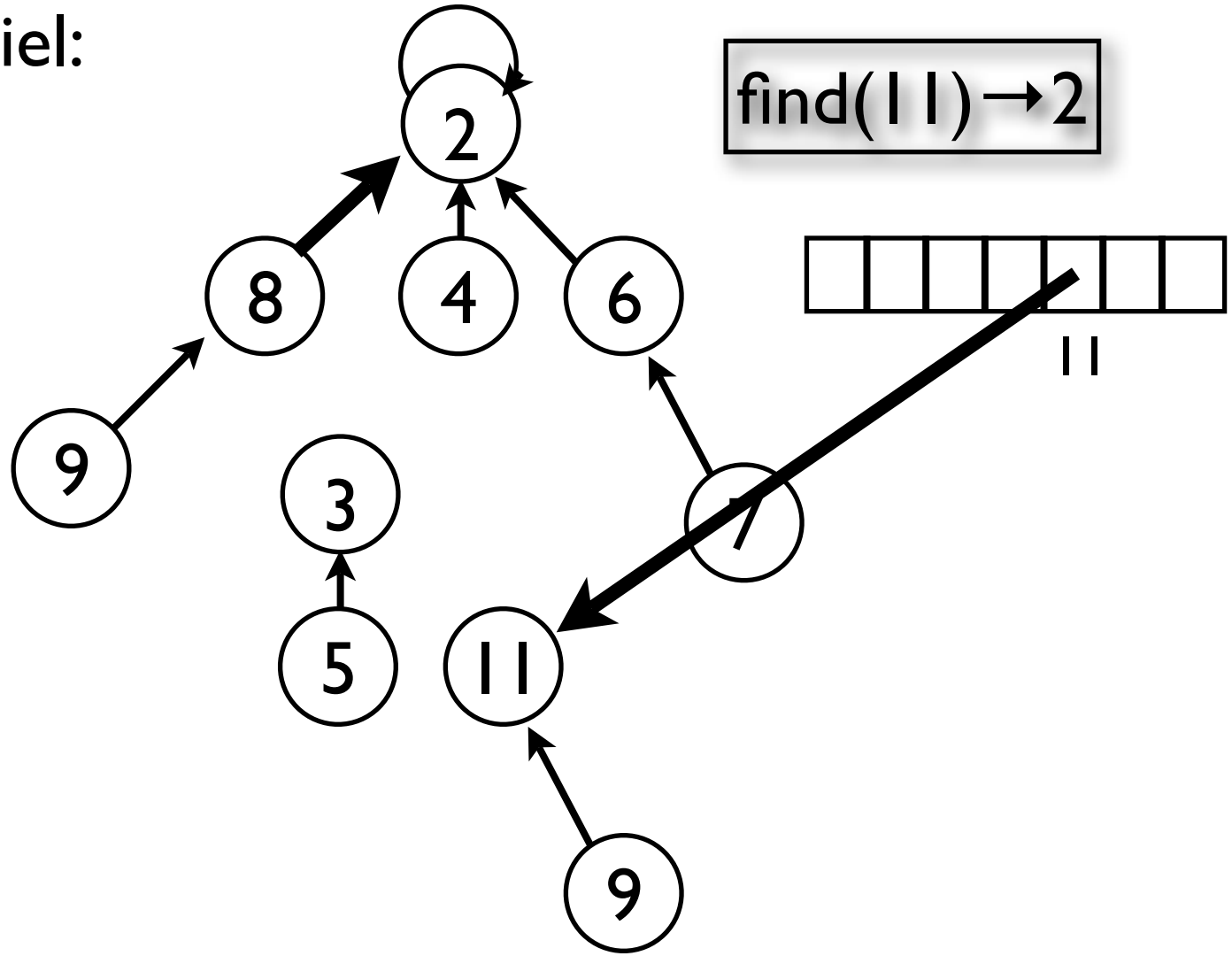
Beispiel:



Graphenalgorithmen

Union-find-Problem - Pfadkompression

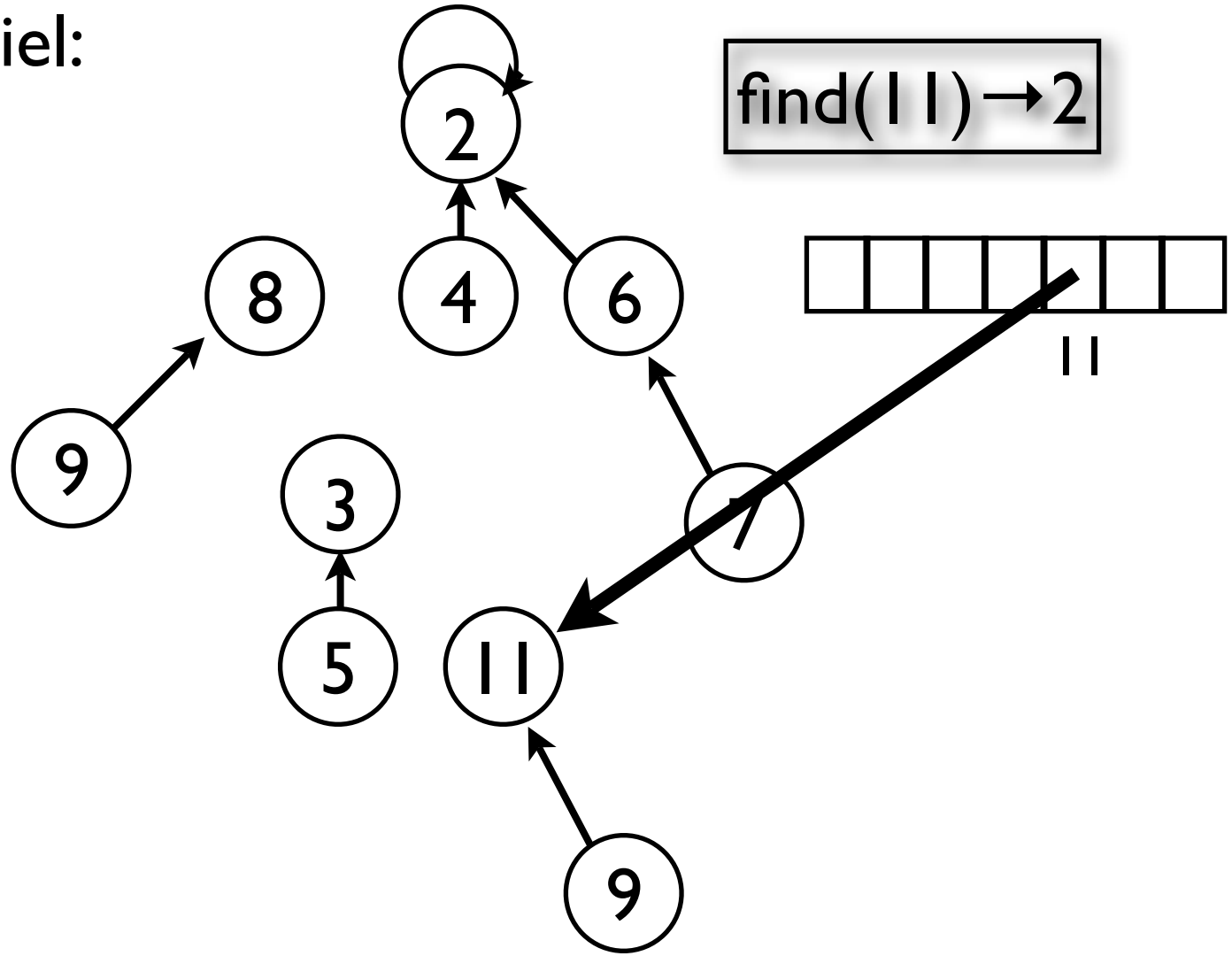
Beispiel:



Graphenalgorithmen

Union-find-Problem - Pfadkompression

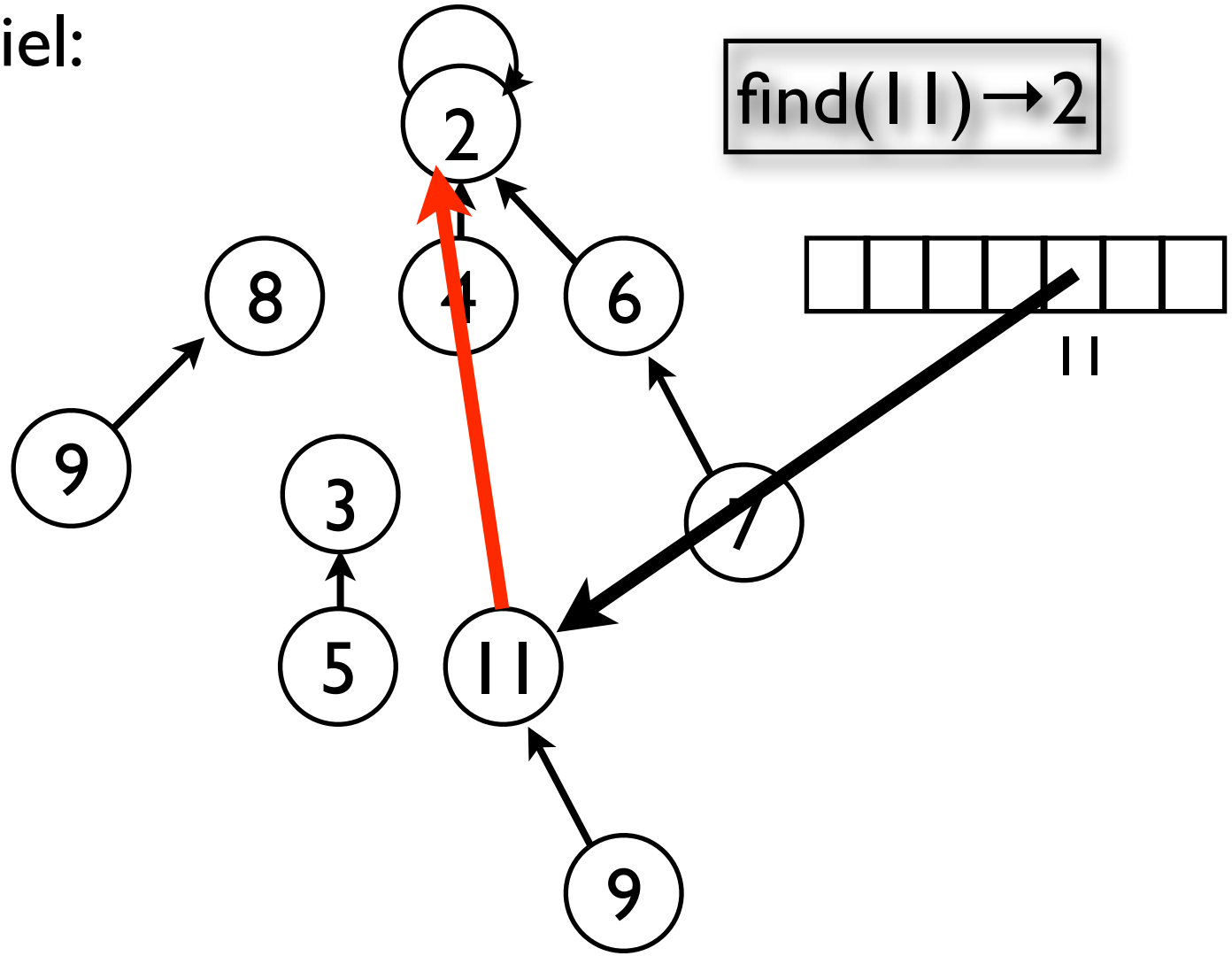
Beispiel:



Graphenalgorithmen

Union-find-Problem - Pfadkompression

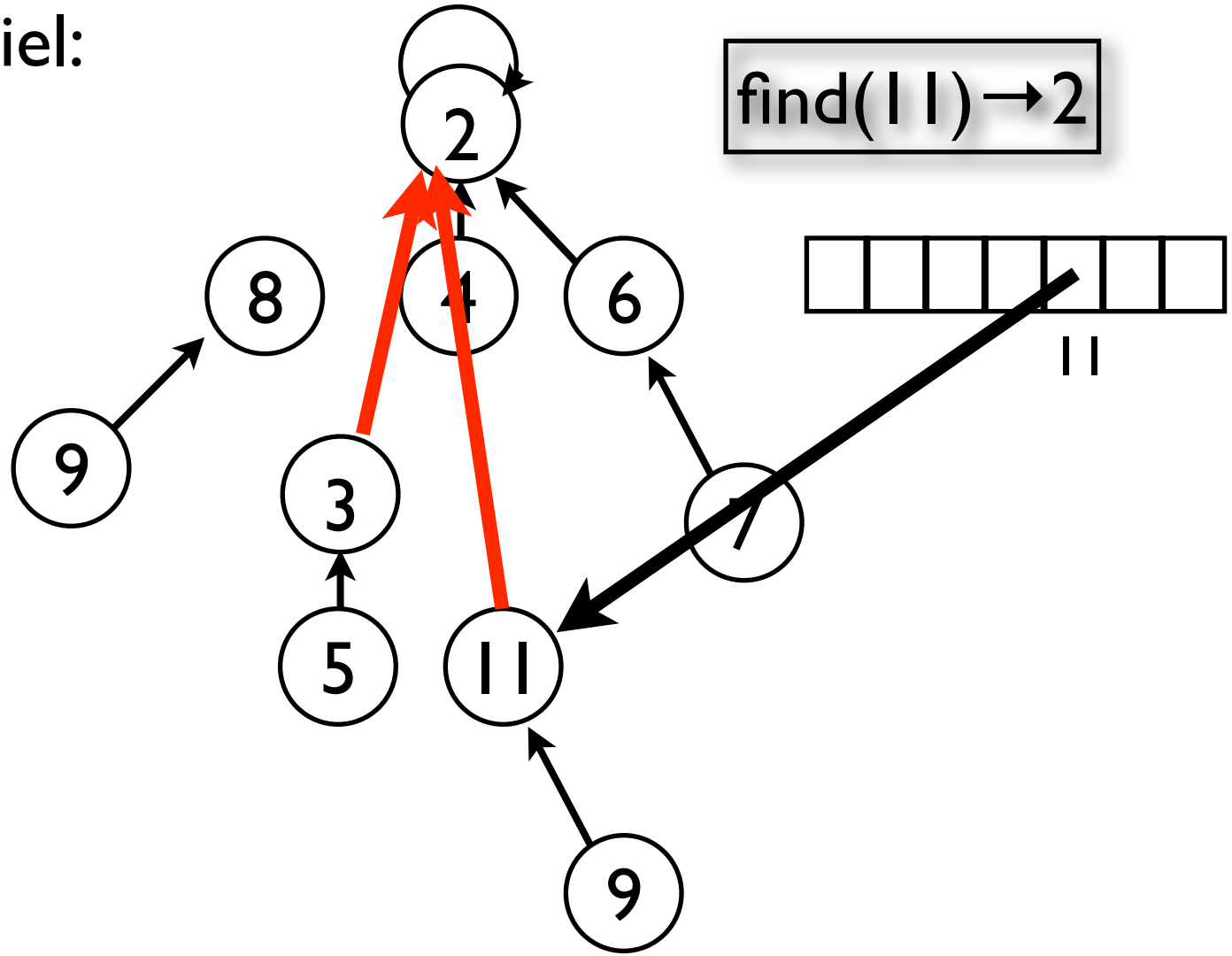
Beispiel:



Graphenalgorithmen

Union-find-Problem - Pfadkompression

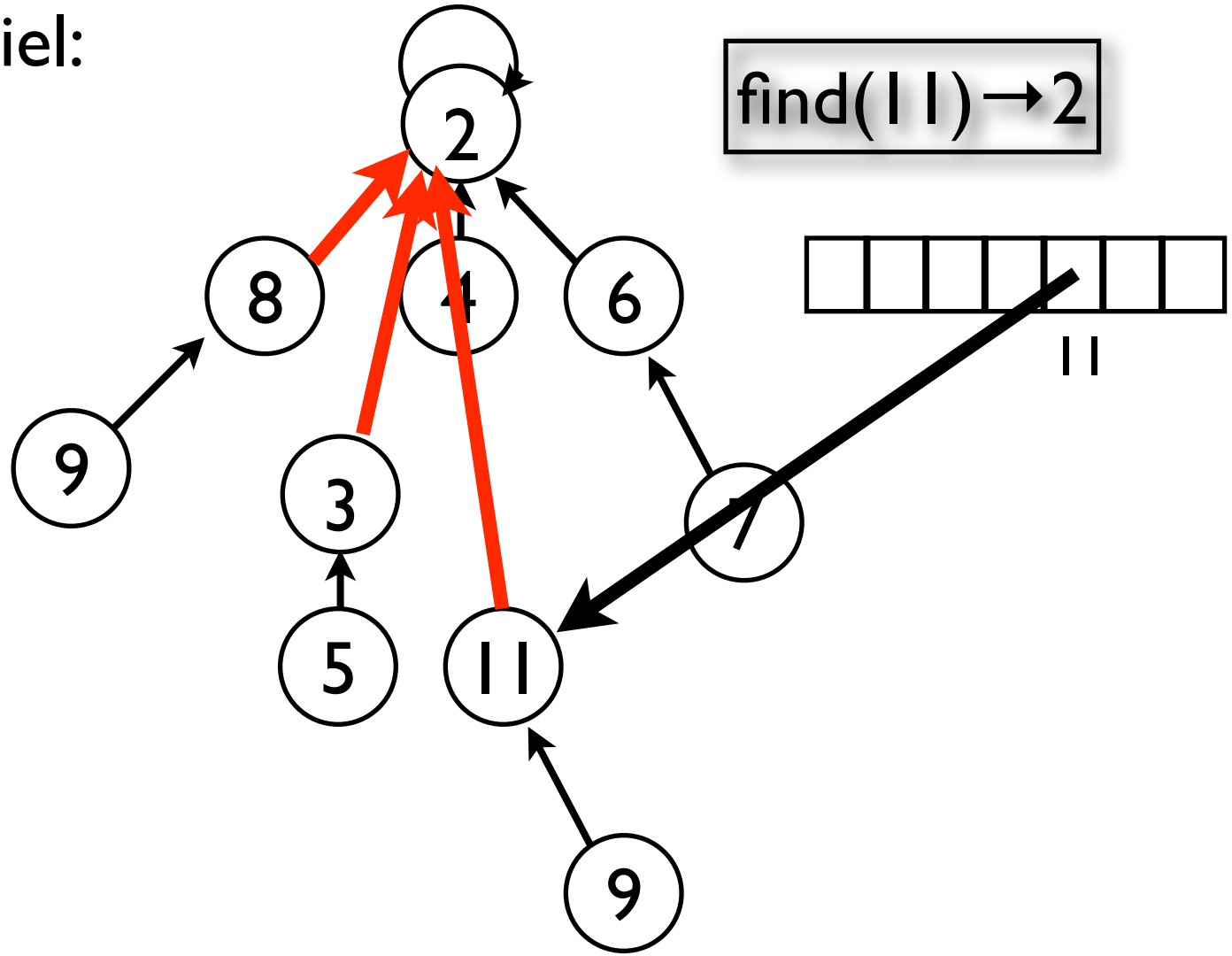
Beispiel:



Graphenalgorithmen

Union-find-Problem - Pfadkompression

Beispiel:



Graphenalgorithmen
Union-find-Problem - Pfadkompression

Ergebnis [Tarjan/van Leeuwen 1975,1984]:

Der Aufwand für n find-Operationen reduziert sich auf

$$O(n \cdot \alpha(n)).$$

α ist die inverse Funktion der Ackermann-Funktion.

($\alpha(n)=4$ für alle praktisch auftretenden Werte n.)

Graphenalgorithmen
Union-find-Problem - Pfadkompression

α ist die inverse Funktion der Ackermann-Funktion A:

$A(1,j)=2^j$, $j \geq 1$

$A(i,1)=A(i-1,2)$, $i \geq 2$

$A(i,j)=A(i-1,A(i,j-1))$, $i,j \geq 2$.

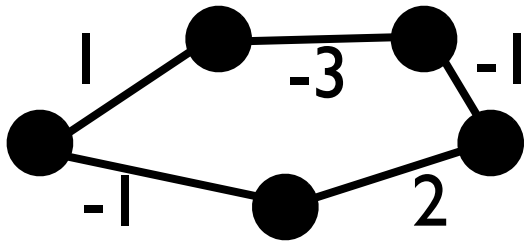
$\alpha(n)=\min \{i \geq 1 \mid A(i,1)>\log n\}$,

$\alpha(n) \leq 3$ für $n<216$

$\alpha(n)=4$ für alle praktisch auftretenden Werte n.

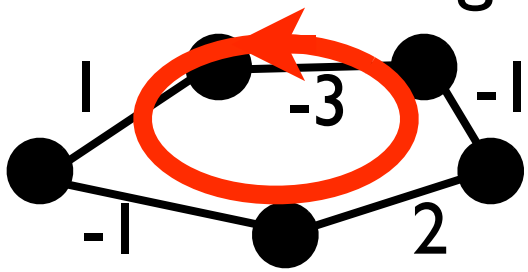
Graphenalgorithmen
Kürzeste Wege

- Aufgabe: Bestimmung aller kürzesten Wege von einem festen Knoten s (source) zu allen anderen Knoten eines ungerichteten Graphen G=(V,E,c) mit Kostenfunktion c: $E \rightarrow IR^+$.
  - single-source shortest paths problem
  - all-pairs shortest paths problem
- Beachte: "IR^+". Nimmt man negative Kanten hinzu, können kürzeste Wege $-\infty$ lang sein.



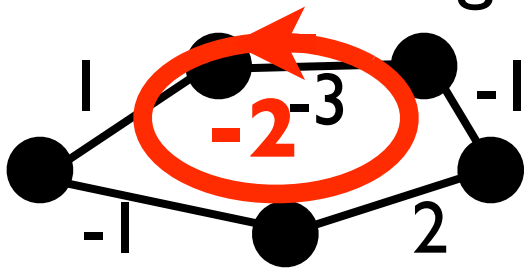
Graphenalgorithmen
Kürzeste Wege

- Aufgabe: Bestimmung aller kürzesten Wege von einem festen Knoten s (source) zu allen anderen Knoten eines ungerichteten Graphen G=(V,E,c) mit Kostenfunktion c: $E \rightarrow \mathbb{R}^+$.
  - single-source shortest paths problem
  - all-pairs shortest paths problem
- Beachte: "$\mathbb{R}^+$". Nimmt man negative Kanten hinzu, können kürzeste Wege $-\infty$ lang sein.



Graphenalgorithmen
Kürzeste Wege

- Aufgabe: Bestimmung aller kürzesten Wege von einem festen Knoten s (source) zu allen anderen Knoten eines ungerichteten Graphen G=(V,E,c) mit Kostenfunktion c: $E \rightarrow \mathbb{R}^+$.
  - single-source shortest paths problem
  - all-pairs shortest paths problem
- Beachte: "$\mathbb{R}^+$". Nimmt man negative Kanten hinzu, können kürzeste Wege $-\infty$ lang sein.



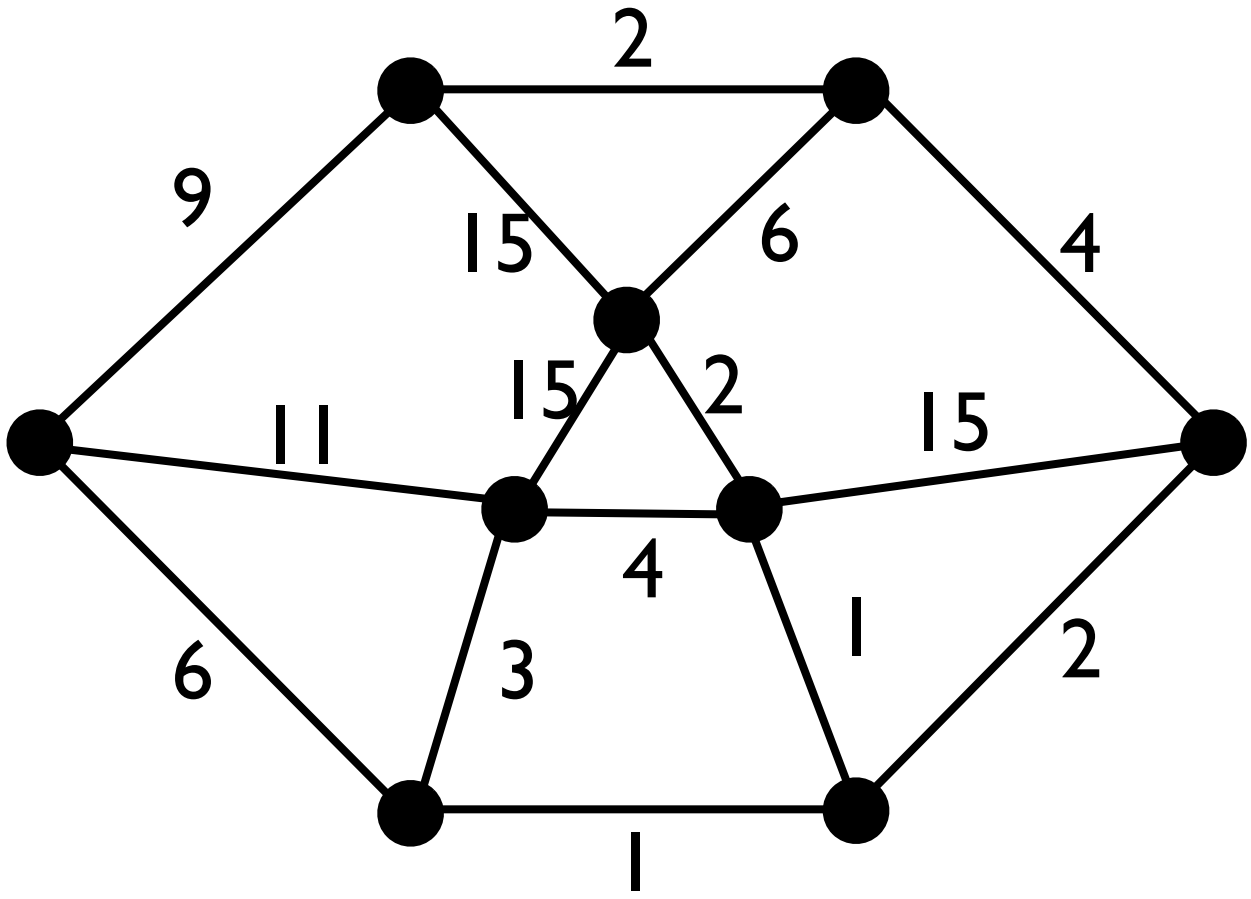
Graphenalgorithmen
Kürzeste Wege - Begriffe

Begriffe:

- **Weg**: $p=(v_0,v_1,...,v_k)$, falls $\{v_i,v_{i+1}\}\in E$, $i=0,...,k-1$
- **Kosten**: $c(p)=\sum_{i=0}^{k-1} c(\{v_i,v_{i+1}\})$
- **Abstand**: $d(v,v')=\min\{c(p) \mid p \text{ ist Weg von } v \text{ nach } v'\}$, falls es einen Weg von v nach v' gibt, sonst $d(v,v')=\infty$
- **kürzester Weg** sp(v,v'): Weg von v nach v' mit $c(p)=d(v,v')$

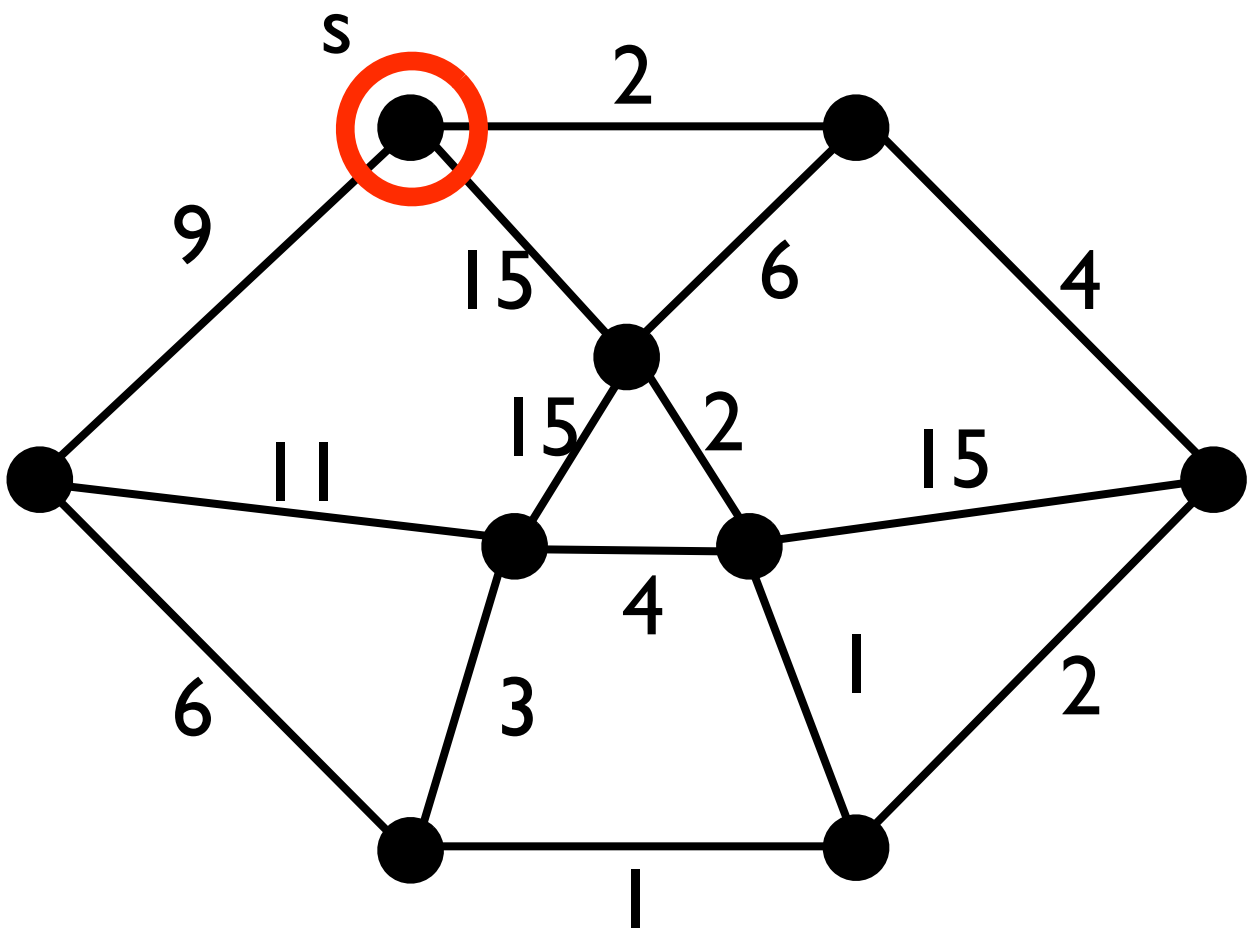
Graphenalgorithmen

Kürzeste Wege - Beispiel



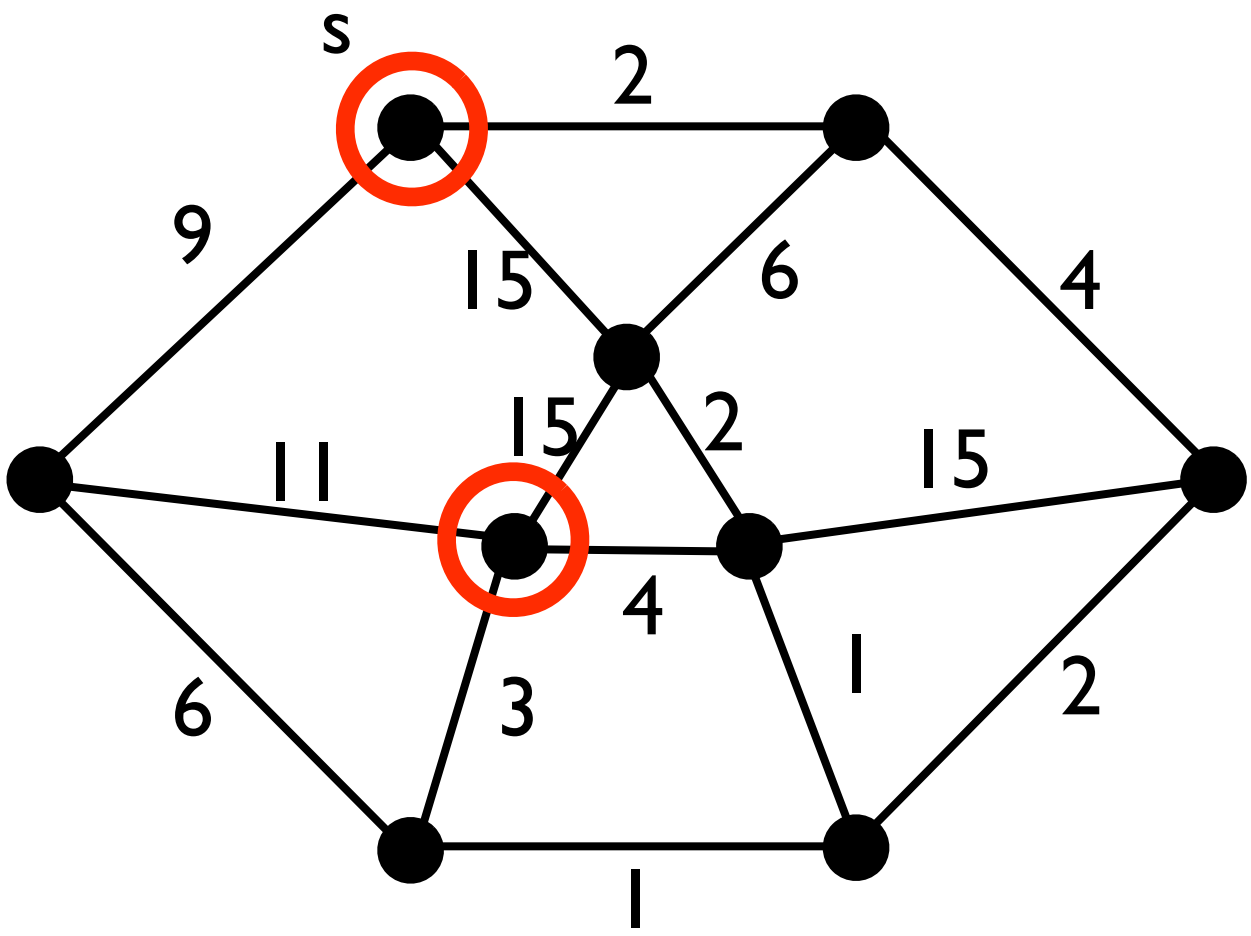
Graphenalgorithmen

Kürzeste Wege - Beispiel



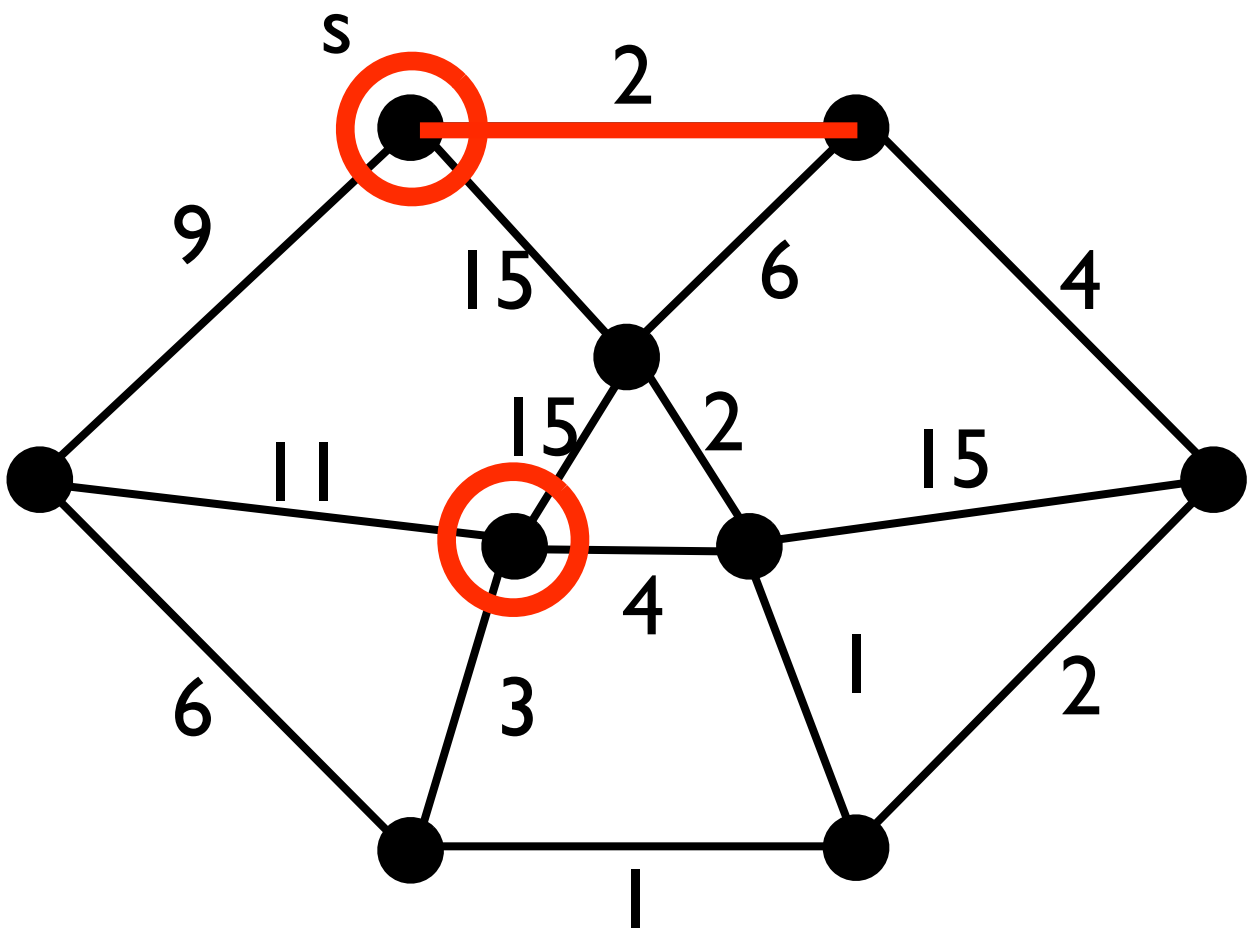
Graphenalgorithmen

Kürzeste Wege - Beispiel



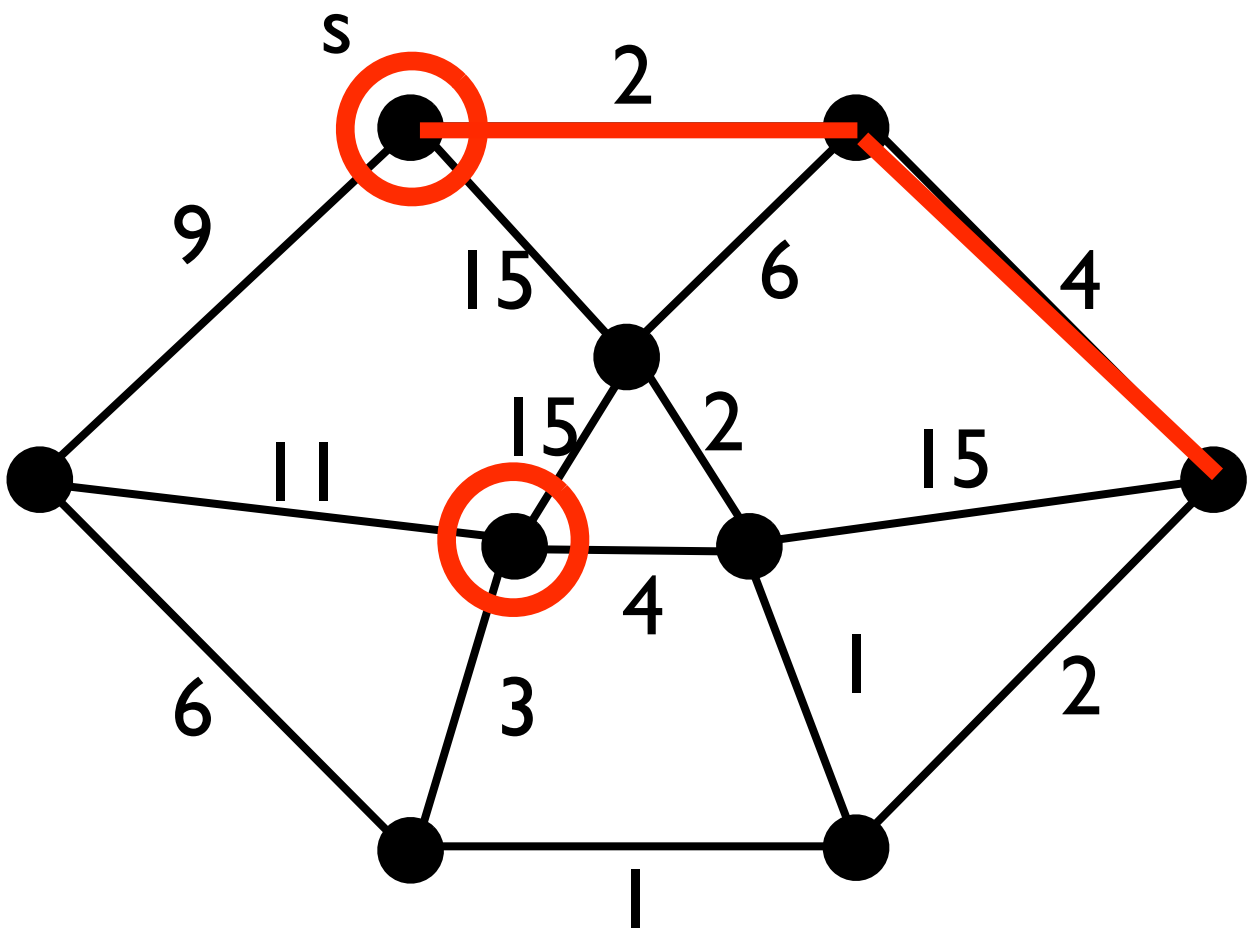
Graphenalgorithmen

Kürzeste Wege - Beispiel



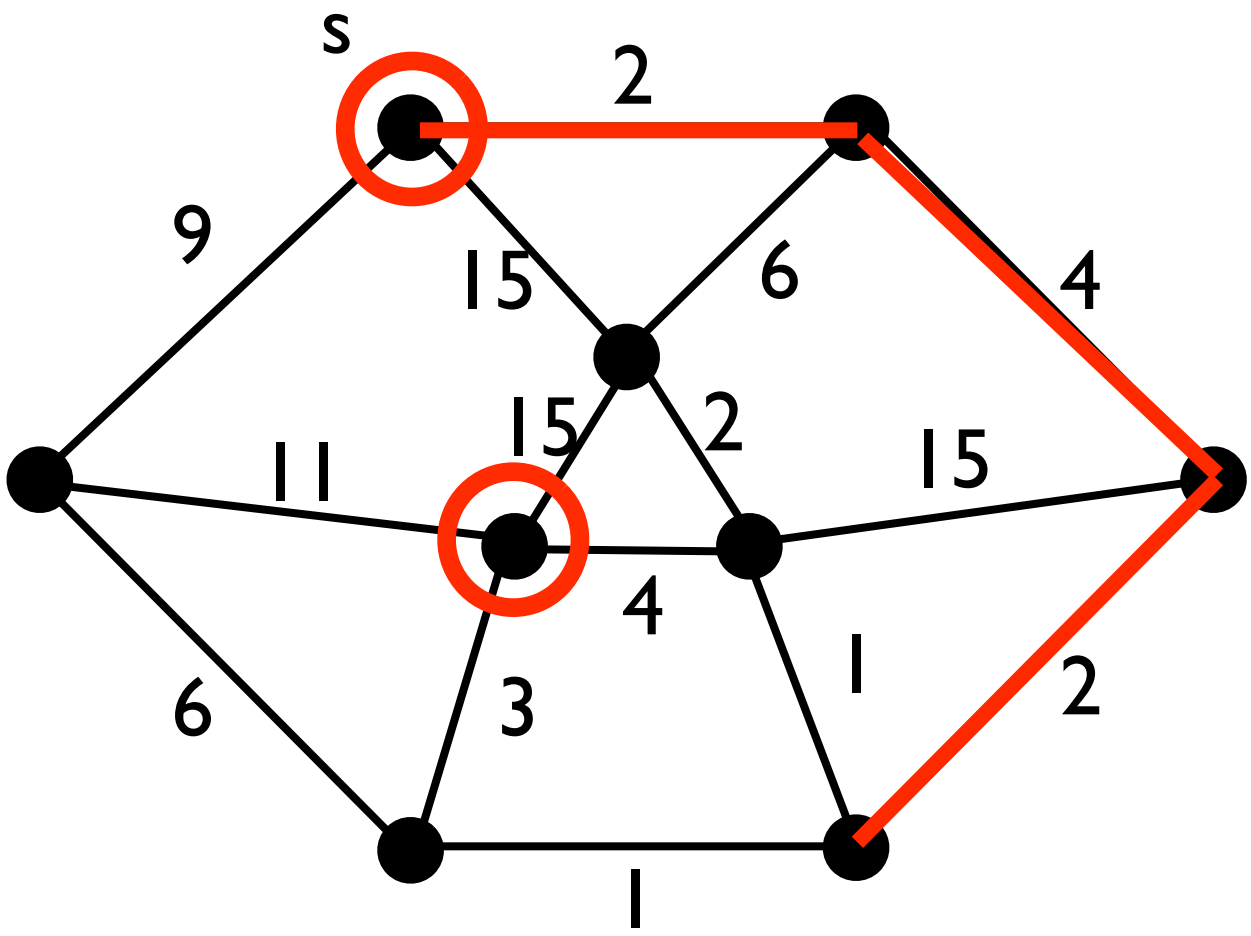
Graphenalgorithmen

Kürzeste Wege - Beispiel



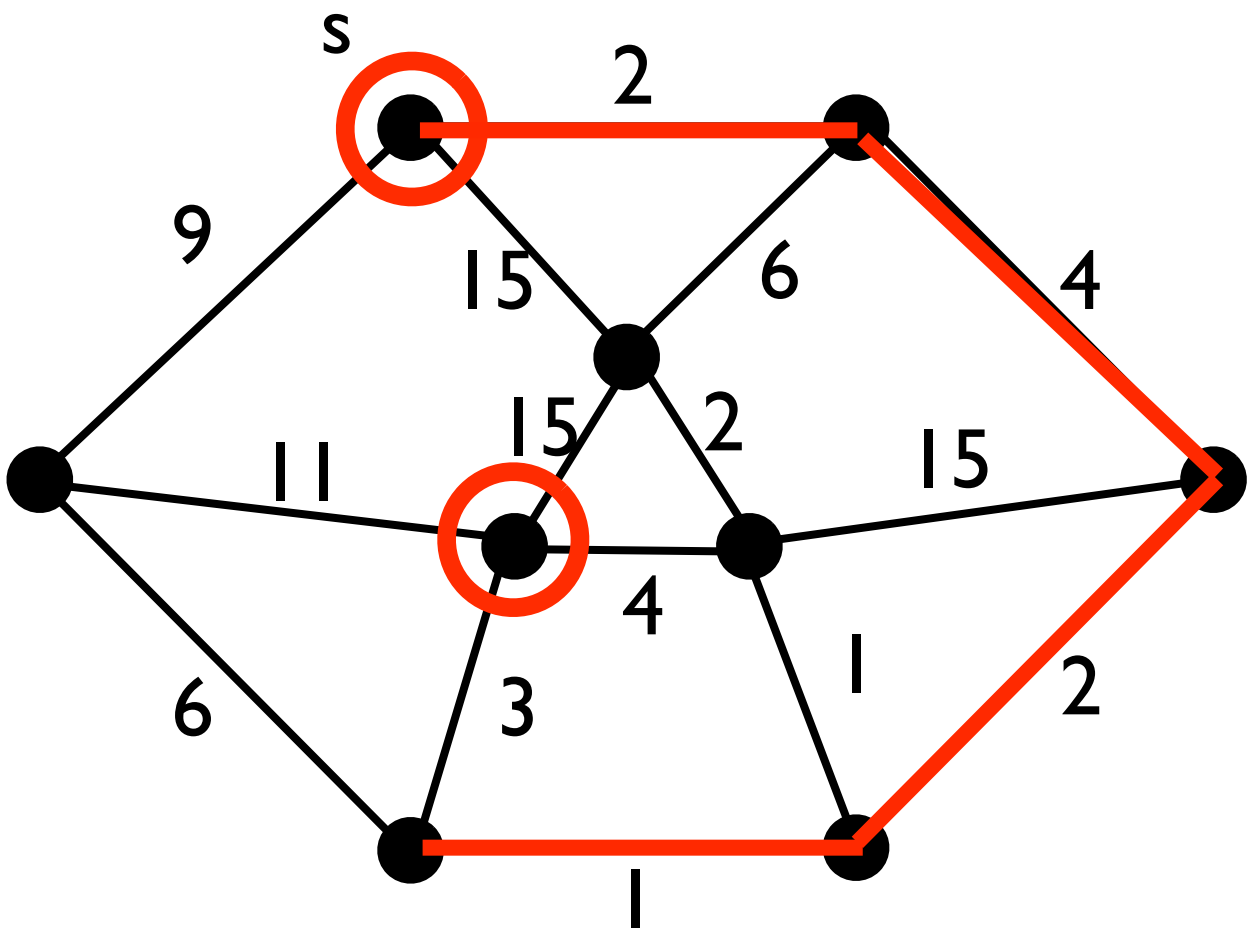
Graphenalgorithmen

Kürzeste Wege - Beispiel



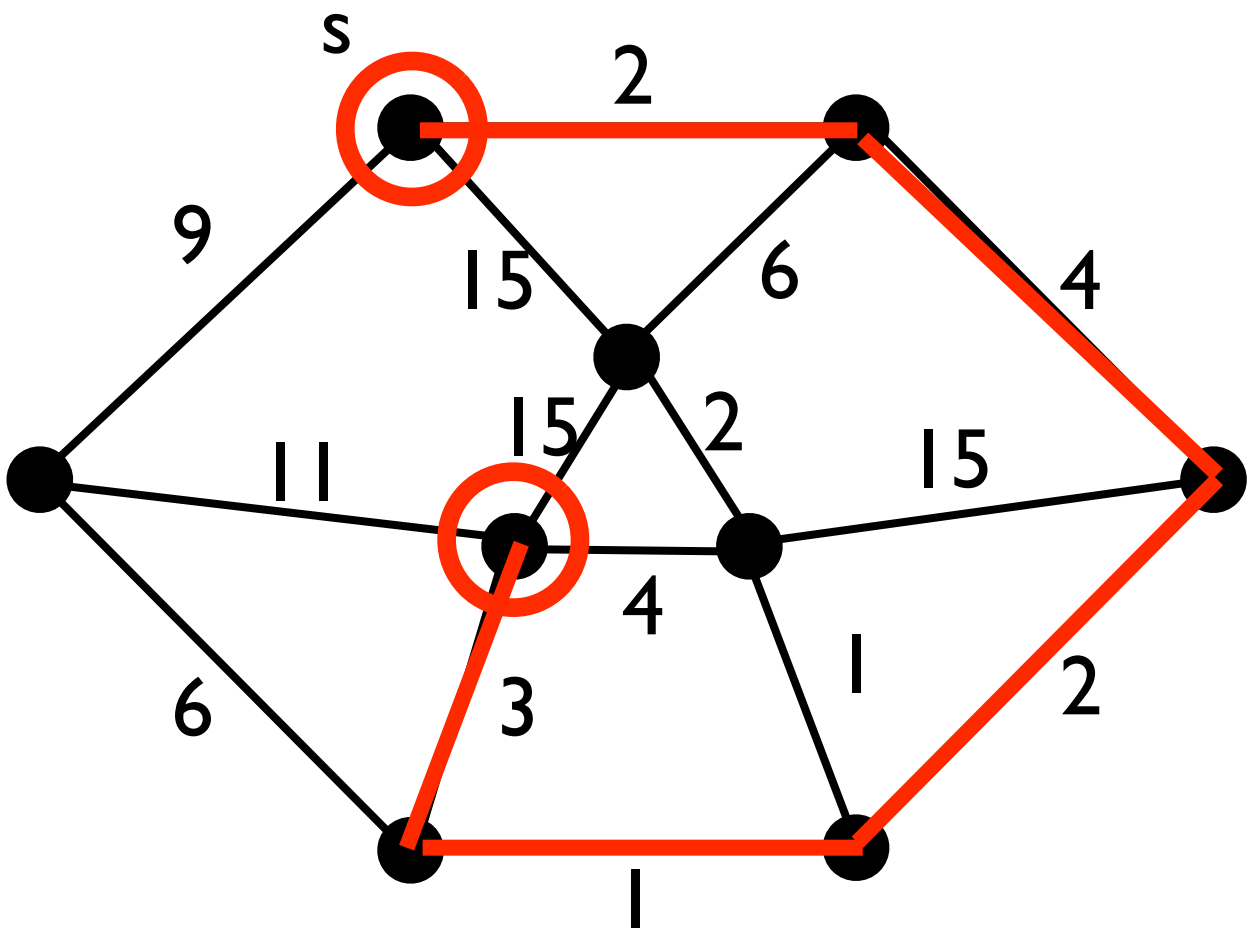
Graphenalgorithmen

Kürzeste Wege - Beispiel



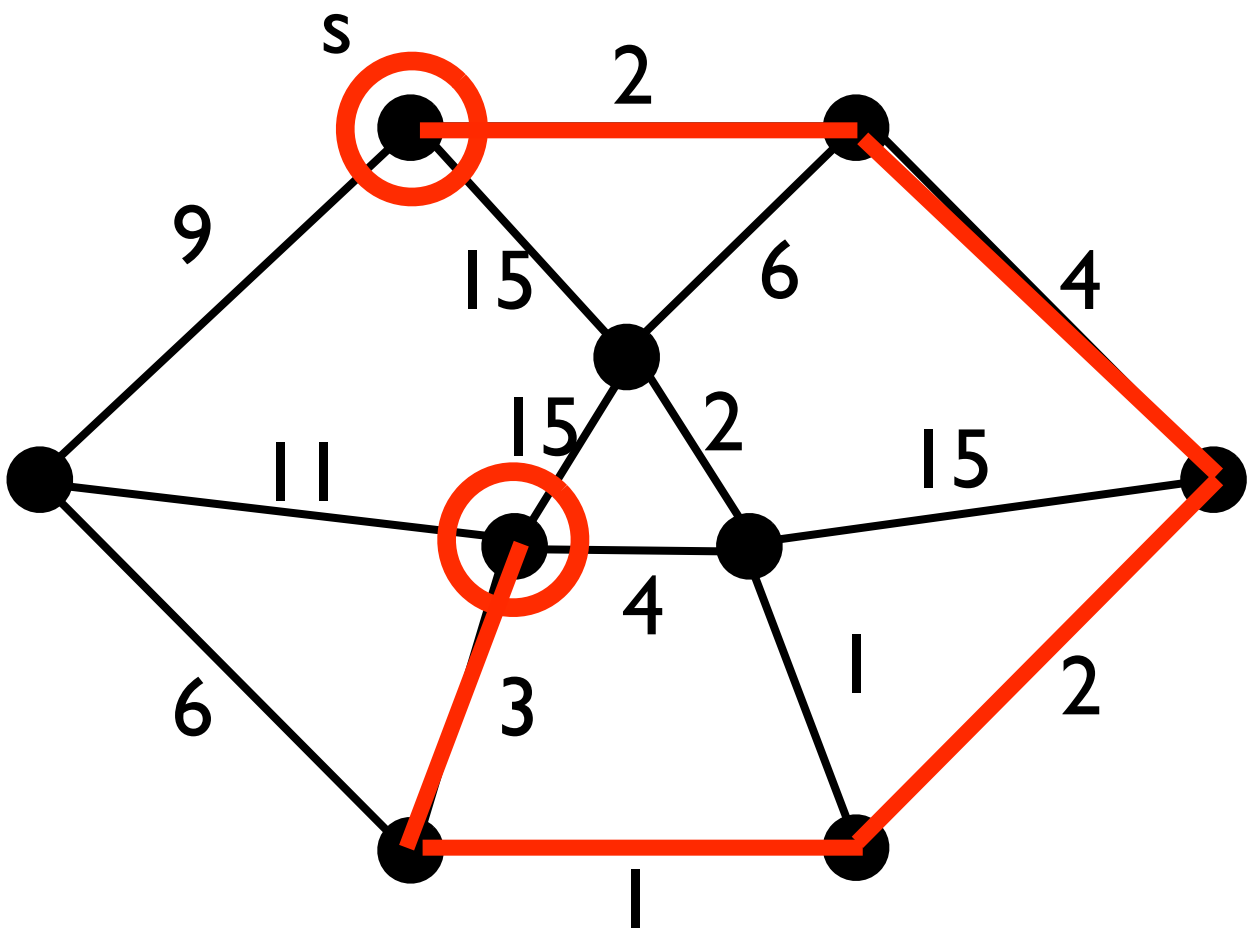
Graphenalgorithmen

Kürzeste Wege - Beispiel



Graphenalgorithmen

Kürzeste Wege - Beispiel



Kosten c(p)=12

Graphenalgorithmen
Kürzeste Wege - Lösungsidee

Idee:

Starte bei Knoten s eine kreisförmige (äquidistante) Welle und halte fest, wann sie welchen Knoten erreicht.

Graphenalgorithmen
Kürzeste Wege - Optimalitätsprinzip

Optimalitätsprinzip:

Für jeden kürzesten Weg $p=(v_0,v_1,...,v_k)$ ist jeder Teilweg $p'= (v_i,...,v_j)$, $0 \leq i < j \leq k$, kürzester Weg von v_i nach v_j.

Nutzung für einen **Greedy**-Algorithmus (E.W. Dijkstra, 1959):

Baue kürzeste Teilwege durch Hinzunahme von Kanten zu kürzesten Wegen aus.

Graphenalgorithmen
Kürzeste Wege - Greedy-Beziehung

Greedy-Beziehung: Es gilt:

1. Für alle kürzesten Wege sp(s,v) und Kanten {v,v'} gilt:

$$c(sp(s,v))+c(\{v,v'\}) \geq c(sp(s,v'))$$

2. Für wenigstens einen kürzesten Weg sp(s,v) und eine Kante {v,v'} gilt:

$$c(sp(s,v))+c(\{v,v'\}) = c(sp(s,v'))$$

Graphenalgorithmen
Kürzeste Wege - Algorithmus

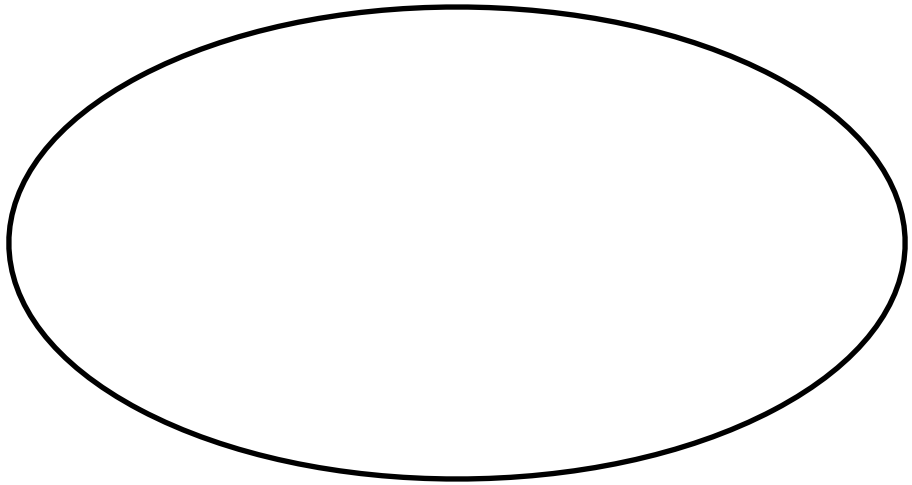
Unterteile Knoten des Graphen in drei Gruppen:

- *gewählt*: ein kürzester Weg von s ist bekannt
- *Rand R*: ein Weg von s ist bekannt
- *unerreicht*: klar.

Graphenalgorithmen
Kürzeste Wege - Algorithmus

Unterteile Knoten des Graphen in drei Gruppen:

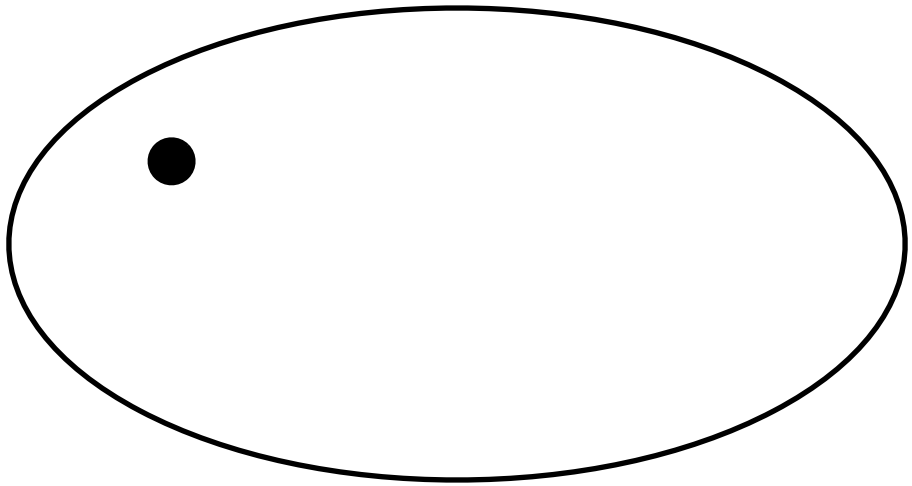
- *gewählt*: ein kürzester Weg von s ist bekannt
- *Rand R*: ein Weg von s ist bekannt
- *unerreicht*: klar.



Graphenalgorithmen
Kürzeste Wege - Algorithmus

Unterteile Knoten des Graphen in drei Gruppen:

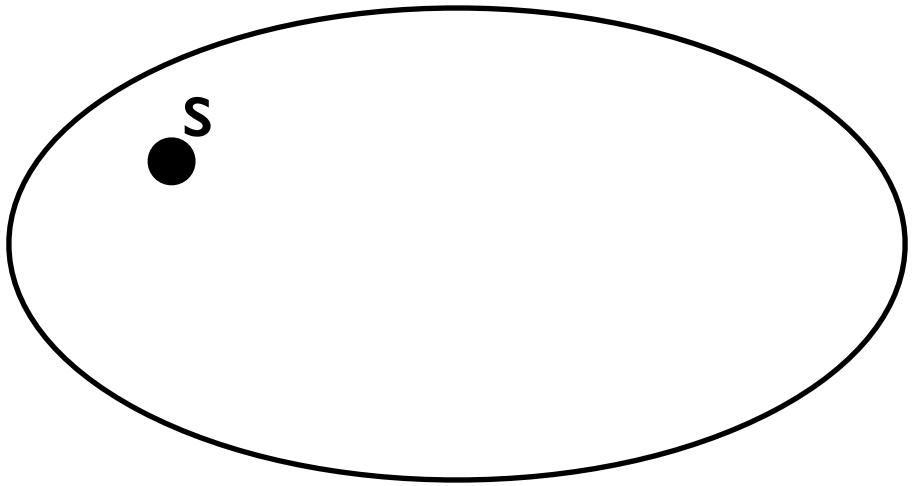
- *gewählt*: ein kürzester Weg von s ist bekannt
- *Rand R*: ein Weg von s ist bekannt
- *unerreicht*: klar.



Graphenalgorithmen
Kürzeste Wege - Algorithmus

Unterteile Knoten des Graphen in drei Gruppen:

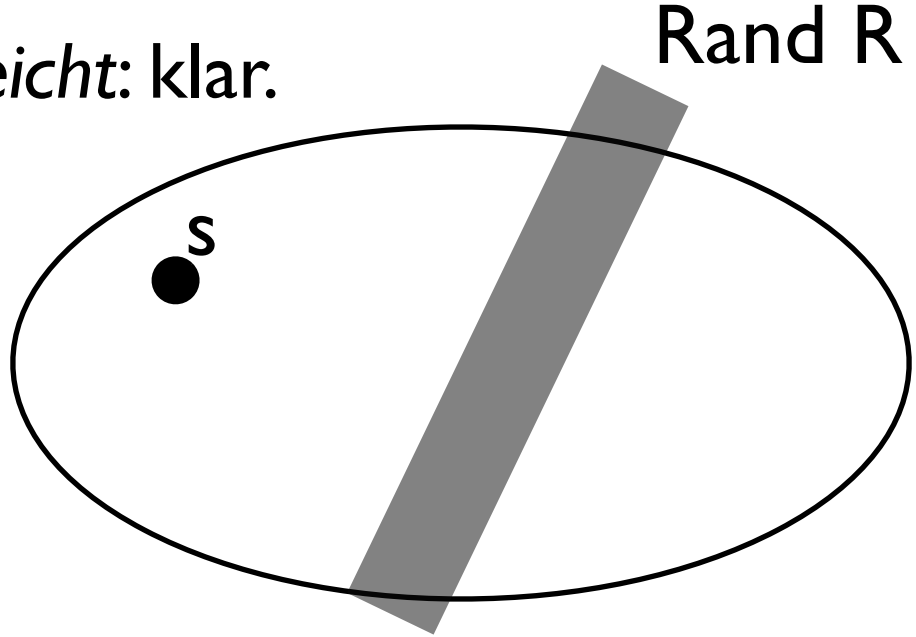
- *gewählt*: ein kürzester Weg von s ist bekannt
- *Rand R*: ein Weg von s ist bekannt
- *unerreicht*: klar.



Graphenalgorithmen
Kürzeste Wege - Algorithmus

Unterteile Knoten des Graphen in drei Gruppen:

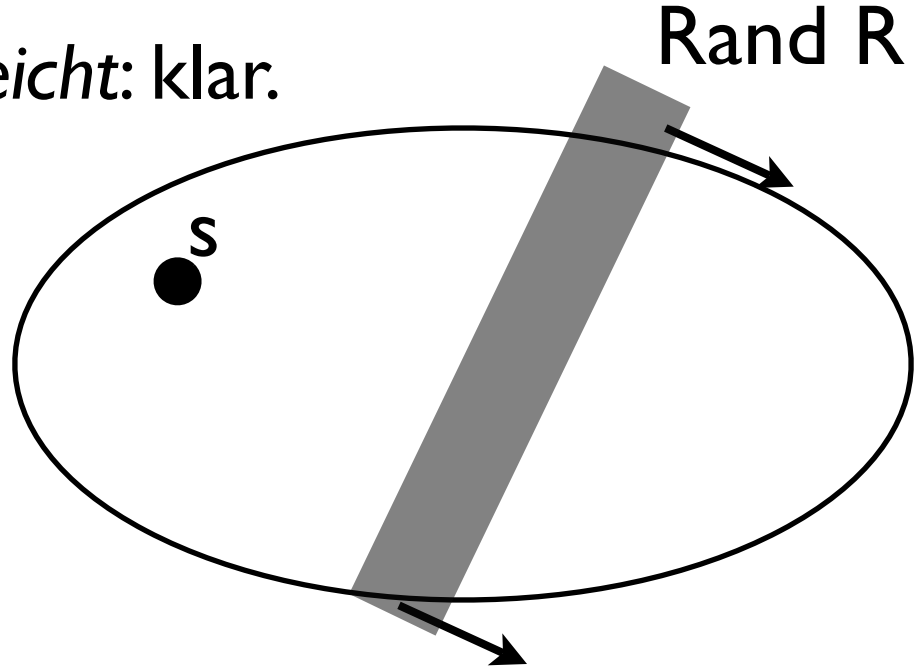
- *gewählt*: ein kürzester Weg von s ist bekannt
- *Rand R*: ein Weg von s ist bekannt
- *unerreicht*: klar.



Graphenalgorithmen
Kürzeste Wege - Algorithmus

Unterteile Knoten des Graphen in drei Gruppen:

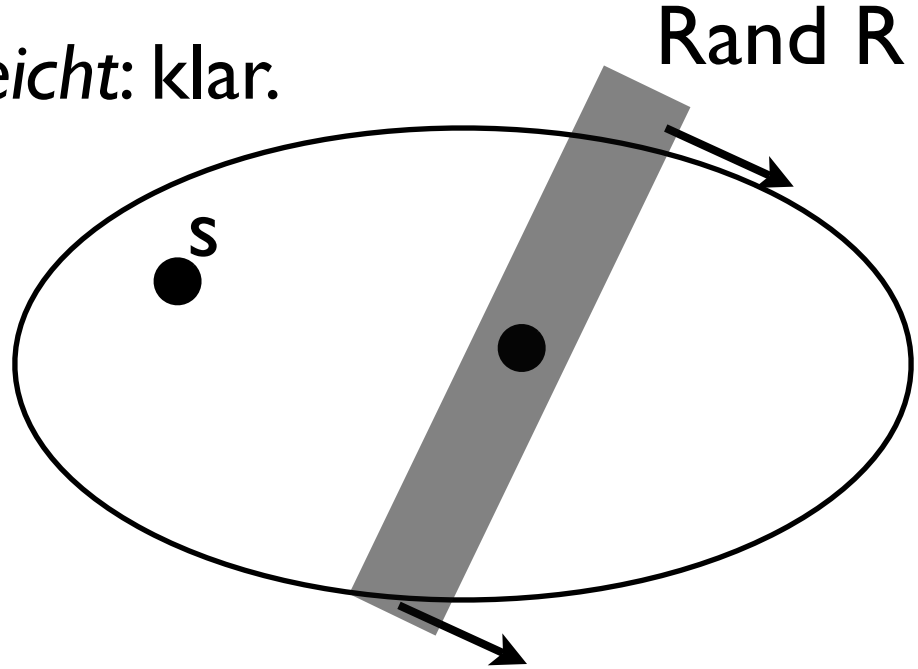
- *gewählt*: ein kürzester Weg von s ist bekannt
- *Rand R*: ein Weg von s ist bekannt
- *unerreicht*: klar.



Graphenalgorithmen
Kürzeste Wege - Algorithmus

Unterteile Knoten des Graphen in drei Gruppen:

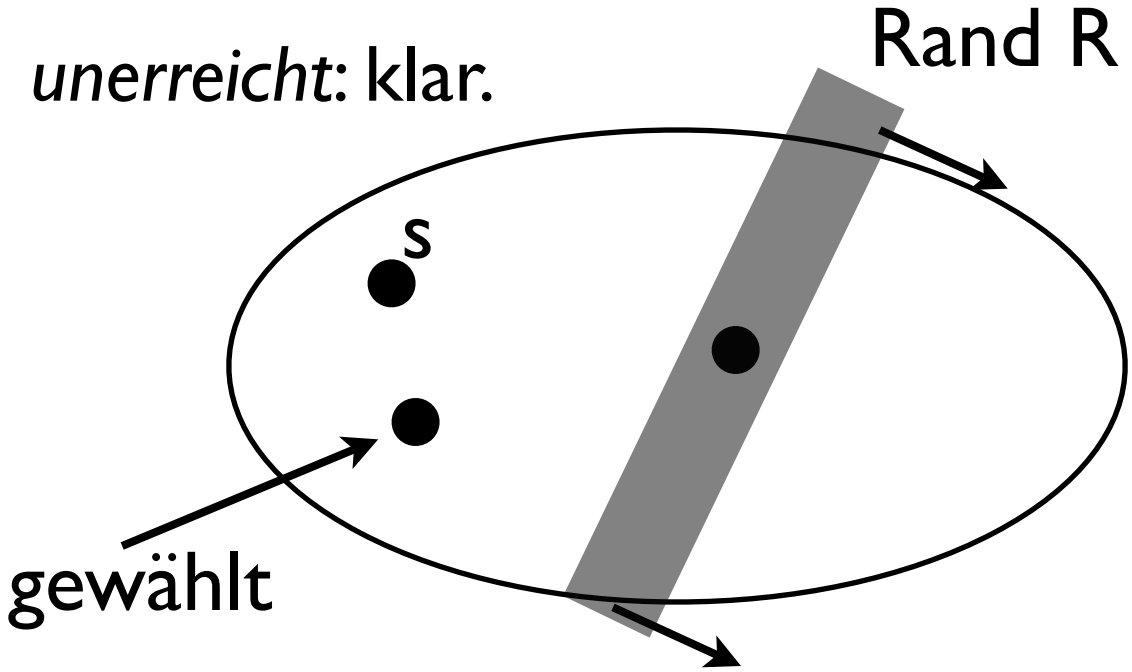
- *gewählt*: ein kürzester Weg von s ist bekannt
- *Rand R*: ein Weg von s ist bekannt
- *unerreicht*: klar.



Graphenalgorithmen
Kürzeste Wege - Algorithmus

Unterteile Knoten des Graphen in drei Gruppen:

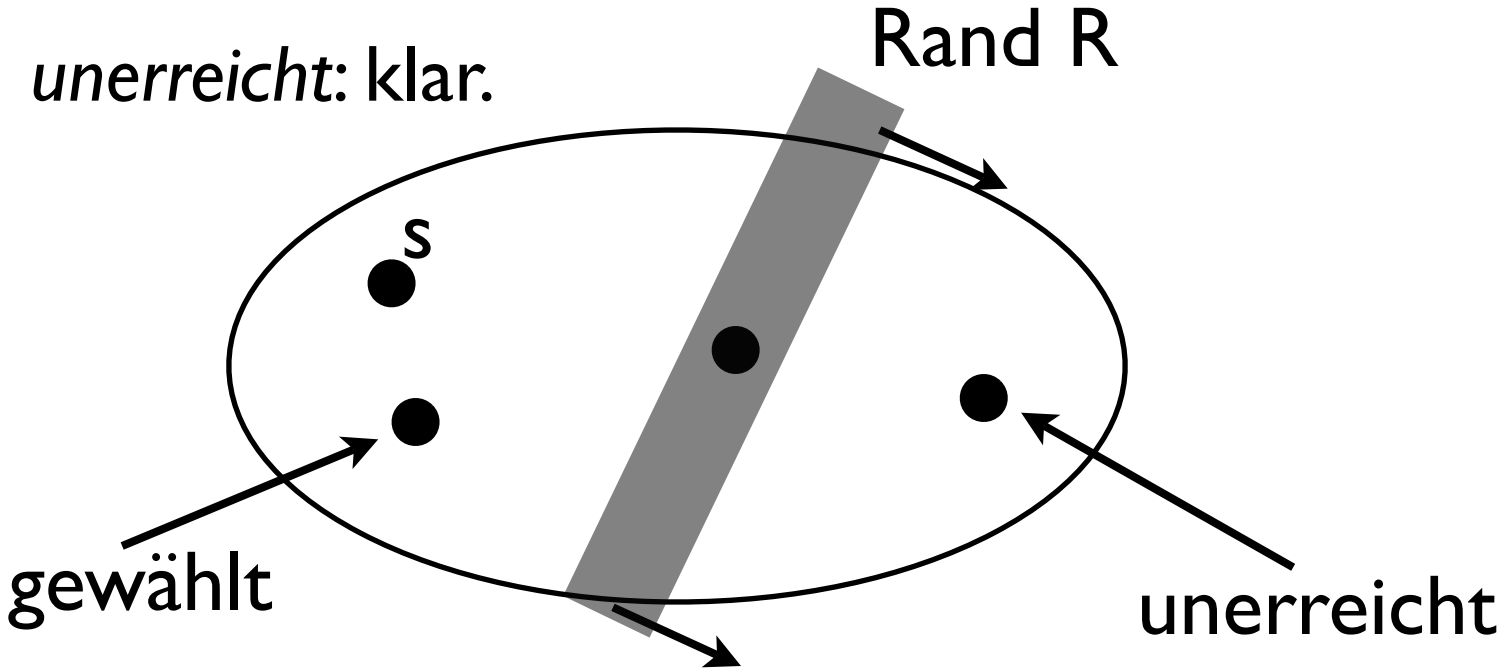
- *gewählt*: ein kürzester Weg von s ist bekannt
- *Rand R*: ein Weg von s ist bekannt
- *unerreicht*: klar.



Graphenalgorithmen
Kürzeste Wege - Algorithmus

Unterteile Knoten des Graphen in drei Gruppen:

- *gewählt*: ein kürzester Weg von s ist bekannt
- *Rand R*: ein Weg von s ist bekannt
- *unerreicht*: klar.



Graphenalgorithmen
Kürzeste Wege - Algorithmus

Eingabe: $G=(V,E,c)$ mit $c: E \rightarrow \mathbb{R}^+$, $s \in V$

Ausgabe: alle kürzesten Wege von s zu allen $v \in V$

```
for all v∈V\{s} do
begin
    v.Vorgänger:=undefiniert;
    v.Abstand:=∞;
    v.gewählt:=false
end;
```

Graphenalgorithmen
Kürzeste Wege - Algorithmus

s.Vorgänger:=s;

s.Abstand:=0;

s.gewählt:=true;

R:=∅;

add(R,s);

Graphenalgorithmen
Kürzeste Wege - Algorithmus

```
while R≠∅ do
begin
    wähle v∈R mit v.Abstand minimal und
    entferne v aus R;
    v.gewählt:=true;
    add(R,v)
end;
```

Graphenalgorithmen
Kürzeste Wege - Algorithmus

```
procedure add(R,v);
for all {v,v'}∈E do
    if not v'.gewählt and
        (v.Abstand+c({v,v'})<v'.Abstand)
    then
        v'.Vorgänger:=v;
        v'.Abstand:=v.Abstand+c({v,v'});
        R:=R+{v'}
    end
```

Graphenalgorithmen
Kürzeste Wege - Implementierung

- Schlüsselidee: Rand R als **Heap** implementieren
  - Einfügen: O(log |E|)=O(log |V|)
  - Minimum auswählen: O(1)+O(log |V|) für Umordnen des Heap
  - R:=∅: O(1)
  - Abfrage R=∅: O(1)
  - Sondersituation: Aktualisieren des Distanzwerts innerhalb add(R,v)

Graphenalgorithmen
Kürzeste Wege - Implementierung

- Aktualisieren des Distanzwerts in R erlaubt der Heap nicht effizient
- Ausweg: neues Tripel mit aktualisierten Werten einfach zusätzlich in Heap einfügen; da dessen Distanzwert kleiner ist als der alte, wird das korrekte Tripel ausgewählt
- Achtung: Knoten aus veraltetem Tripel darf nicht noch einmal entfernt werden

Graphenalgorithmen
Kürzeste Wege - Laufzeit

Eingabe: $G=(V,E,c)$ mit $c: E \rightarrow \mathbb{R}^+$, $s \in V$

Ausgabe: alle kürzesten Wege von s zu allen $v \in V$

```
for all v∈V\{s} do
begin
    v.Vorgänger:=undefiniert;
    v.Abstand:=∞;
    v.gewählt:=false
end;
```

$O(|V|)$

Graphenalgorithmen
Kürzeste Wege - Laufzeit

```
s.Vorgänger:=s;
s.Abstand:=0;
s.gewählt:=true;
R:=∅;
add(R,s);
```

O(1)

Graphenalgorithmen
Kürzeste Wege - Laufzeit

```
while R≠∅ do
begin
    wähle v∈R mit v.Abstand minimal und
    entferne v aus R;
    v.gewählt:=true;
    add(R,v)
end;
```

Graphenalgorithmen
Kürzeste Wege - Laufzeit

```
while R≠∅ do
begin
    wähle v∈R mit v.Abstand minimal und
    entferne v aus R;
    v.gewählt:=true;
    add(R,v)
end;
```

O(1)

Graphenalgorithmen
Kürzeste Wege - Laufzeit

```
while R≠∅ do
begin
    wähle v∈R mit v.Abstand minimal und
    entferne v aus R;
    v.gewählt:=true;
    add(R,v)
end;
```

Graphenalgorithmen
Kürzeste Wege - Laufzeit

```
while R≠∅ do
begin
    wähle v∈R mit v.Abstand minimal und
    entferne v aus R;
    v.gewählt:=true;
    add(R,v)
end;
```

O(1)

Graphenalgorithmen
Kürzeste Wege - Laufzeit

```
while R≠∅ do
begin
    wähle v∈R mit v.Abstand minimal und
    entferne v aus R;
    v.gewählt:=true;
    add(R,v)
end;
```

Graphenalgorithmen
Kürzeste Wege - Laufzeit

```
while R≠∅ do
begin
    wähle v∈R mit v.Abstand minimal und
    entferne v aus R;
    v.gewählt:=true;
    add(R,v)
end;
```

O(log |V|)

Graphenalgorithmen
Kürzeste Wege - Laufzeit

```
while R≠∅ do
begin
    wähle v∈R mit v.Abstand minimal und
    entferne v aus R;
    v.gewählt:=true;
    add(R,v)
end;
```

Graphenalgorithmen
Kürzeste Wege - Laufzeit

```
while R≠∅ do
begin
    wähle v∈R mit v.Abstand minimal und
    entferne v aus R;
    v.gewählt:=true;
    add(R,v)
end;
```

|E| mal

Graphenalgorithmen
Kürzeste Wege - Laufzeit

```
procedure add(R,v);
for all {v,v'}∈E do
    if not v'.gewählt and
        (v.Abstand+c({v,v'})<v'.Abstand)
    then
        v'.Vorgänger:=v;
        v'.Abstand:=v.Abstand+c({v,v'});
        R:=R+{v'}
    end
```

Graphenalgorithmen
Kürzeste Wege - Laufzeit

```
procedure add(R,v);
for all {v,v'}∈E do
    if not v'.gewählt and
        (v.Abstand+c({v,v'})<v'.Abstand)
    then
        v'.Vorgänger:=v;
        v'.Abstand:=v.Abstand+c({v,v'});
        R:=R+{v'}
    end
```

O(log |V|)

Graphenalgorithmen
Kürzeste Wege - Laufzeit

```
procedure add(R,v);
for all {v,v'}∈E do
    if not v'.gewählt and
        (v.Abstand+c({v,v'})<v'.Abstand)
    then
        v'.Vorgänger:=v;
        v'.Abstand:=v.Abstand+c({v,v'});
        R:=R+{v'}
    end
```

Graphenalgorithmen
Kürzeste Wege - Laufzeit

```
procedure add(R,v);
for all {v,v'}∈E do
    if not v'.gewählt and
        (v.Abstand+c({v,v'})<v'.Abstand)
    then
        v'.Vorgänger:=v;
        v'.Abstand:=v.Abstand+c({v,v'});
        R:=R+{v'}
    end
```

wird in der Summe nur für |E| Kanten ausgeführt

Graphenalgorithmen
Kürzeste Wege - Laufzeit

```
procedure add(R,v);
for all {v,v'}∈E do
    if not v'.gewählt and
        (v.Abstand+c({v,v'})<v'.Abstand)
    then
        v'.Vorgänger:=v;
        v'.Abstand:=v.Abstand+c({v,v'});
        R:=R+{v'}
    end
```

Graphenalgorithmen
Kürzeste Wege - Laufzeit

- Obiger Algorithmus läuft in Zeit

$$O(|E| \log |V|)$$

- gut geeignet für dünne Graphen $|E| \leq c|V|$

- Alternativalgorithmus (s. Buch von Ottmann/ Widmayer) für dichte Graphen ($|E|=O(|V|^2)$ benötigt Zeit

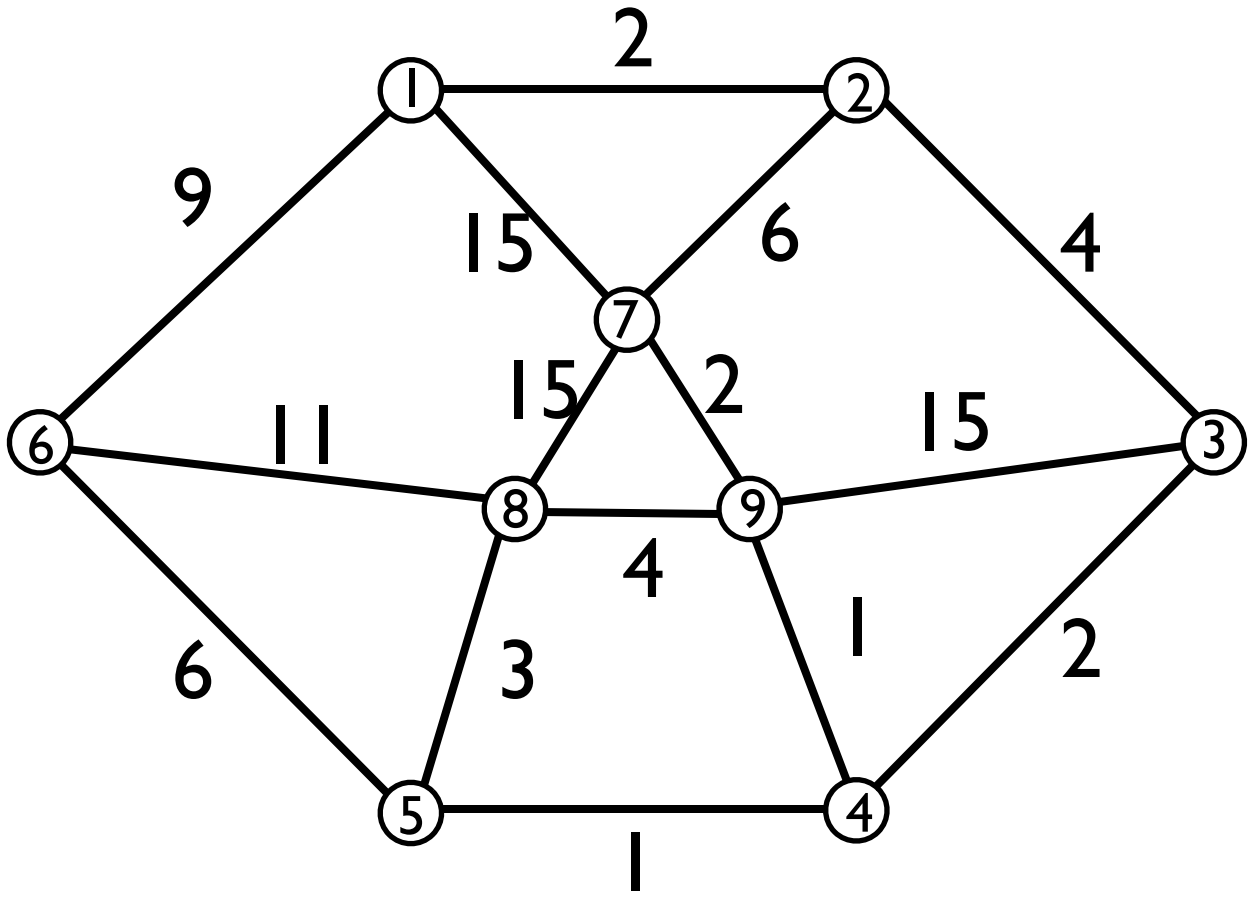
$$O(|V|^2)$$

Graphenalgorithmen

Kürzeste Wege - Beispiel

Darstellung:
(Knotennr, Distanz, Vorgänger)

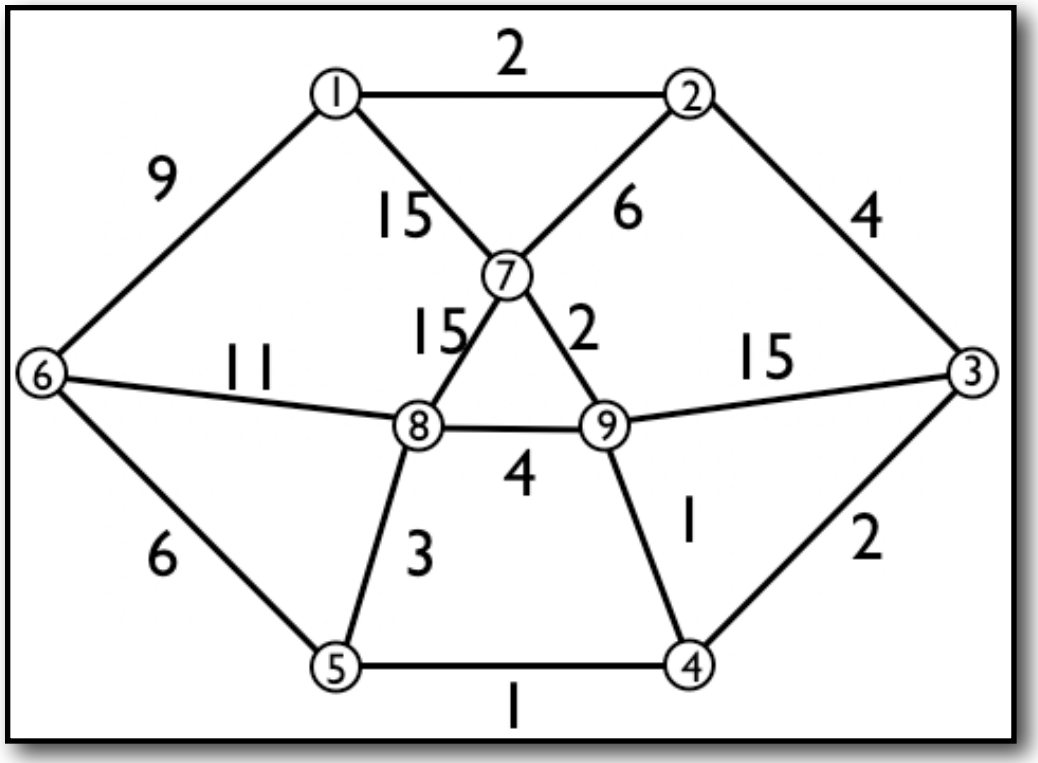
gewählt Rand R



Graphenalgorithmen
Kürzeste Wege - Beispiel

Darstellung:
(Knotennr, Distanz, Vorgänger)

gewählt Rand R

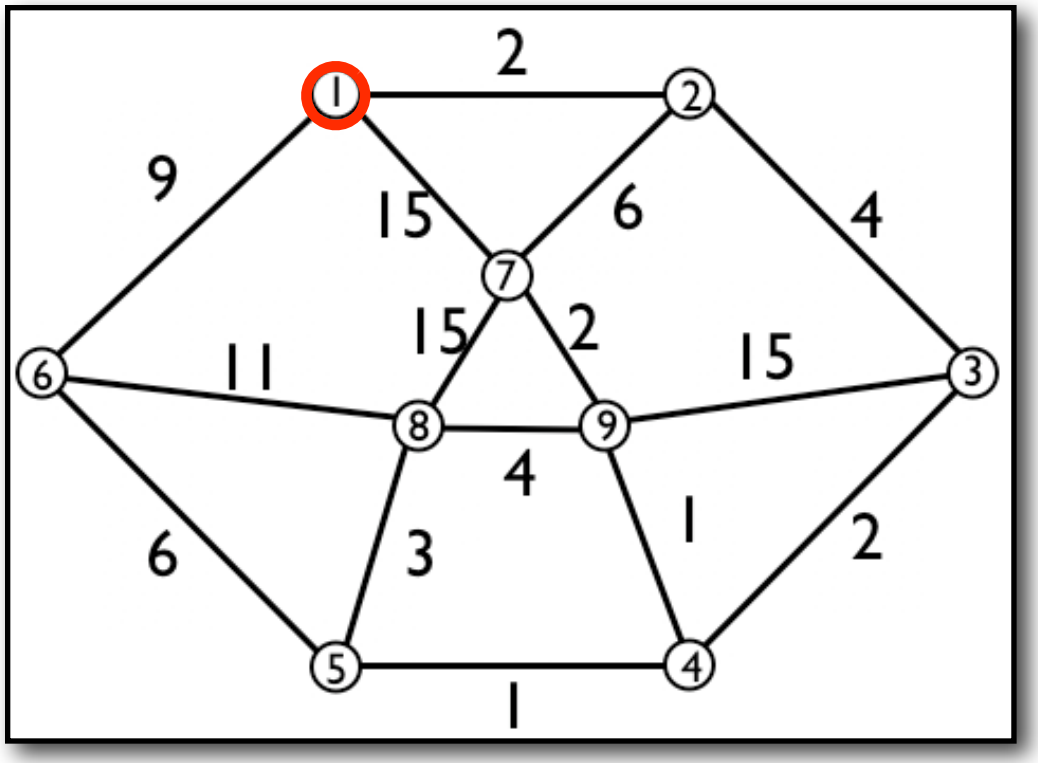


Graphenalgorithmen

Kürzeste Wege - Beispiel

Darstellung:
(Knotennr, Distanz, Vorgänger)

gewählt Rand R

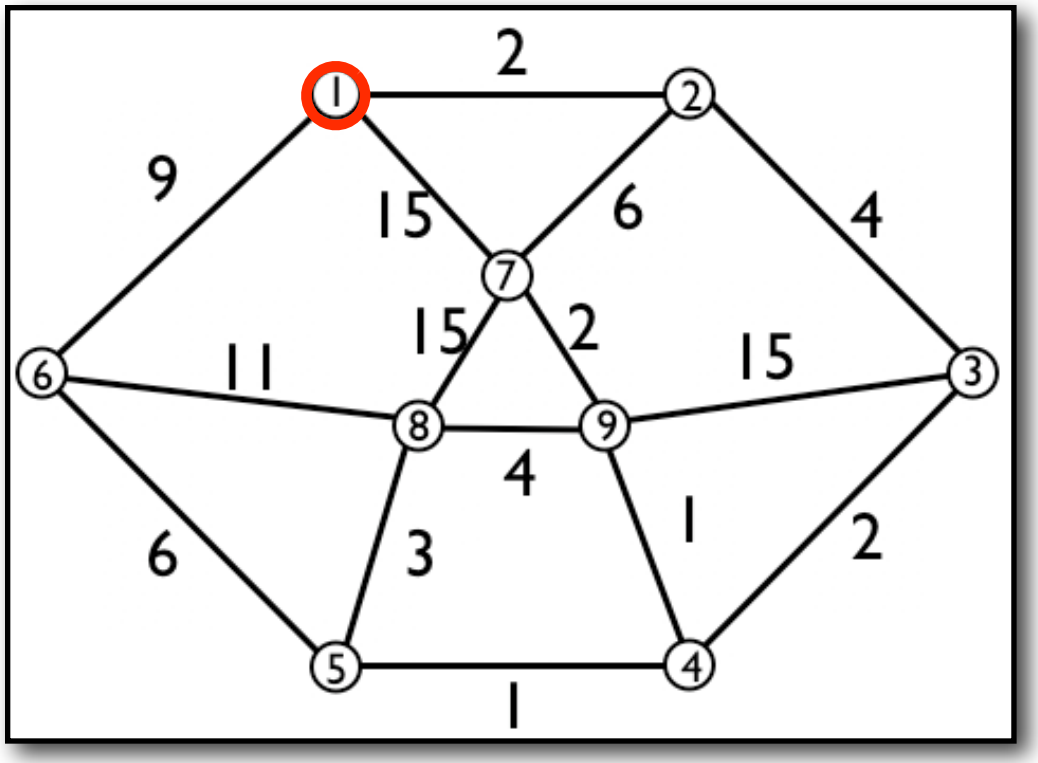


Graphenalgorithmen
Kürzeste Wege - Beispiel

Darstellung:
(Knotennr, Distanz, Vorgänger)

gewählt Rand R

(1,0,1)

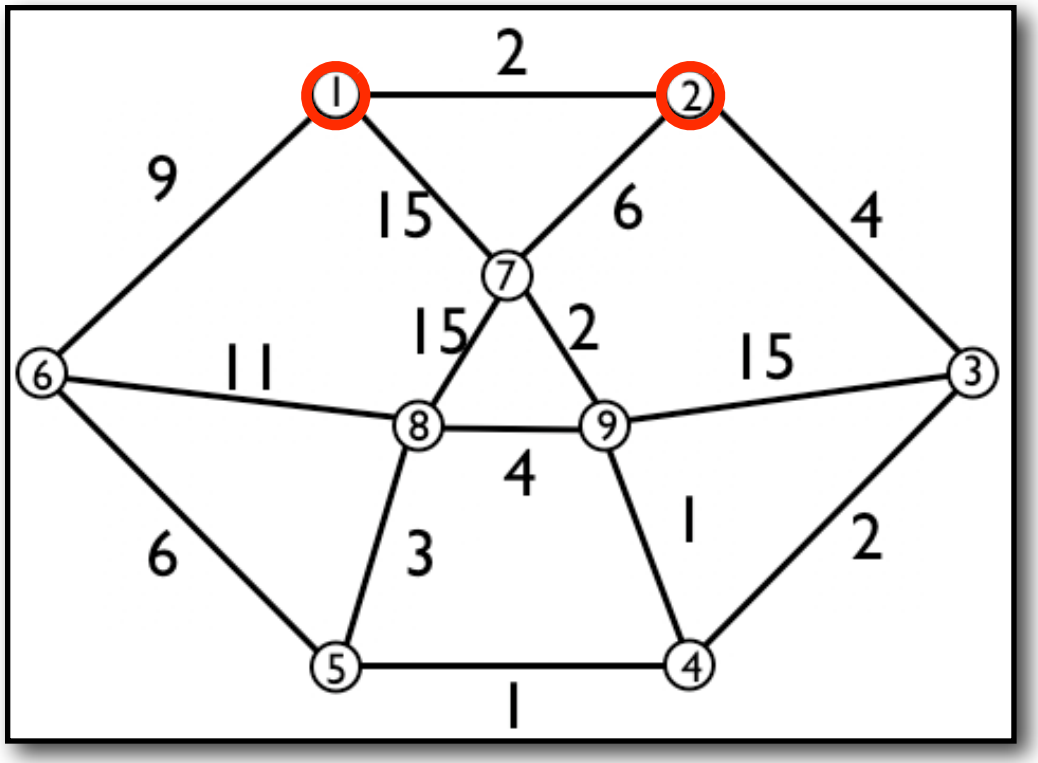


Graphenalgorithmen
Kürzeste Wege - Beispiel

Darstellung:
(Knotennr, Distanz, Vorgänger)

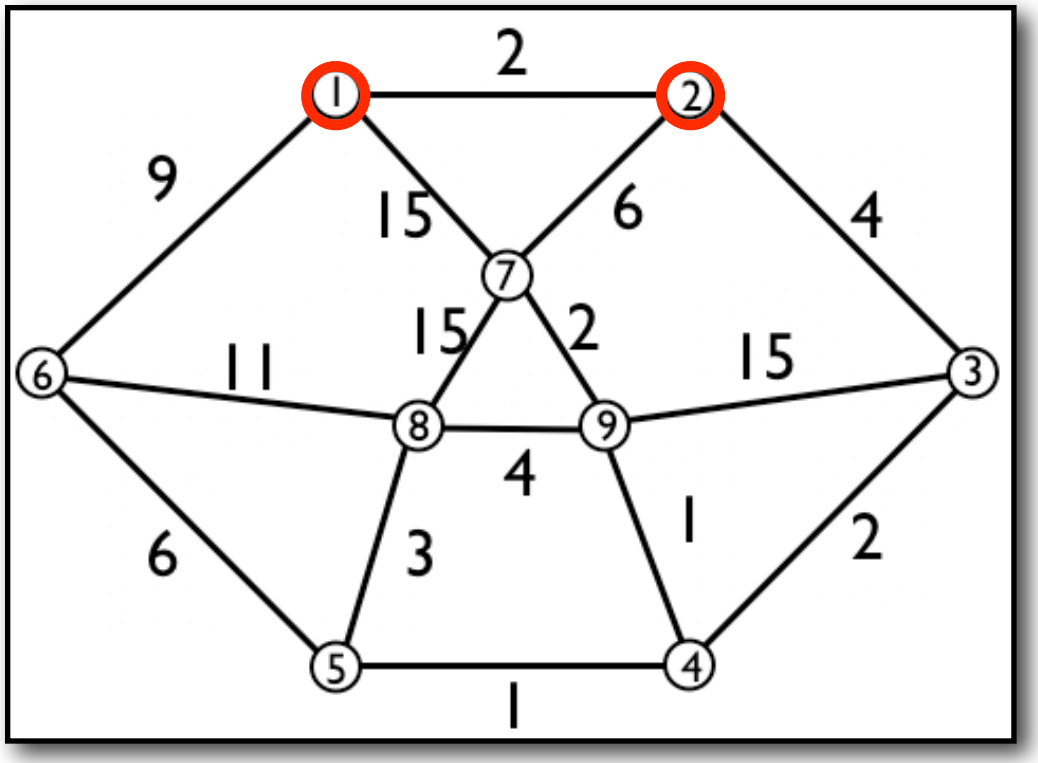
gewählt Rand R

(1,0,1)



Graphenalgorithmen
Kürzeste Wege - Beispiel

Darstellung:
(Knotennr, Distanz, Vorgänger)

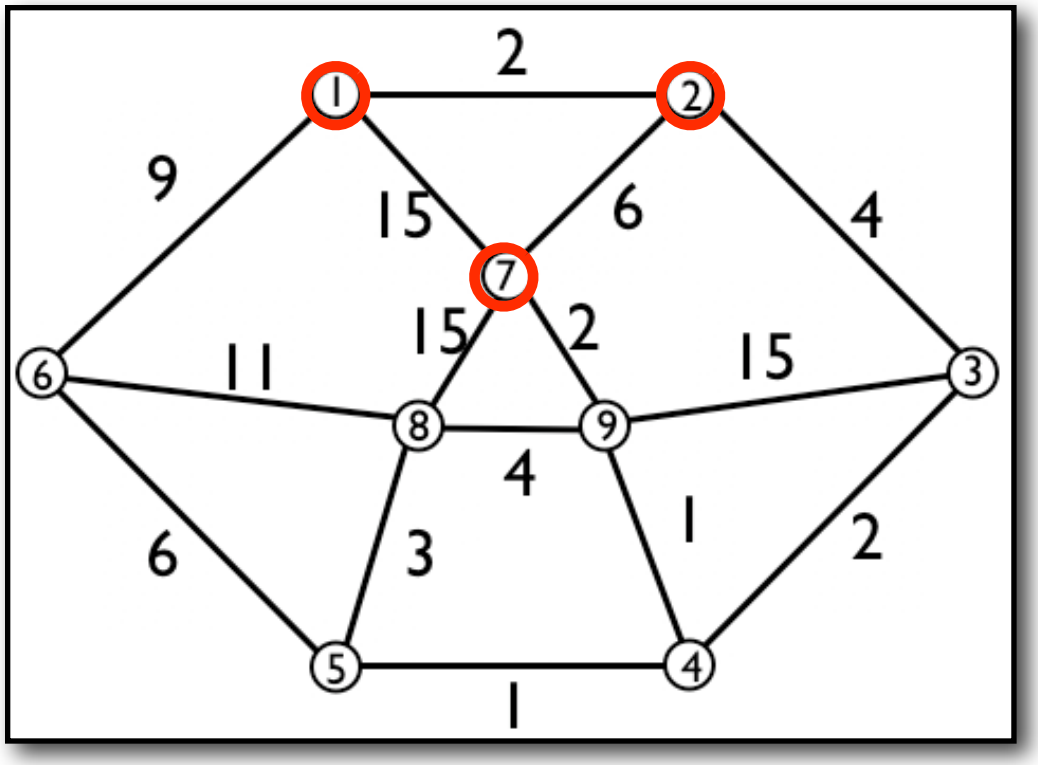


gewählt	Rand R
(1,0,1)	(2,2,1)

Graphenalgorithmen
Kürzeste Wege - Beispiel

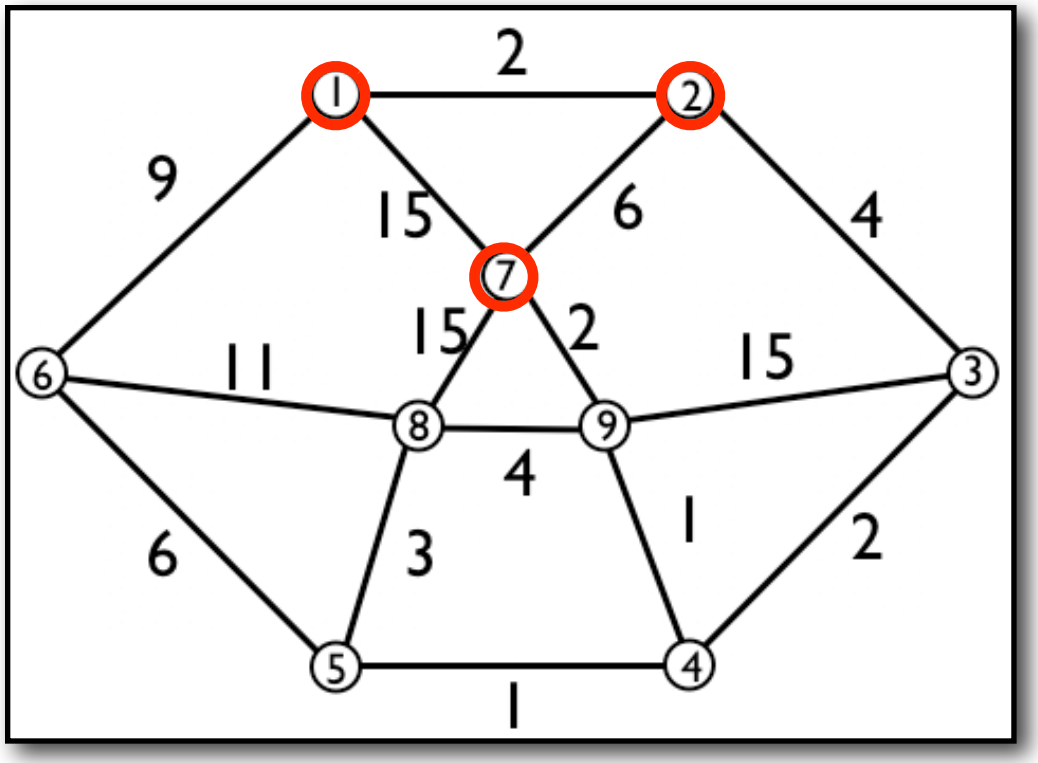
Darstellung:
(Knotennr, Distanz, Vorgänger)

gewählt	Rand R
(1,0,1)	(2,2,1)



Graphenalgorithmen
Kürzeste Wege - Beispiel

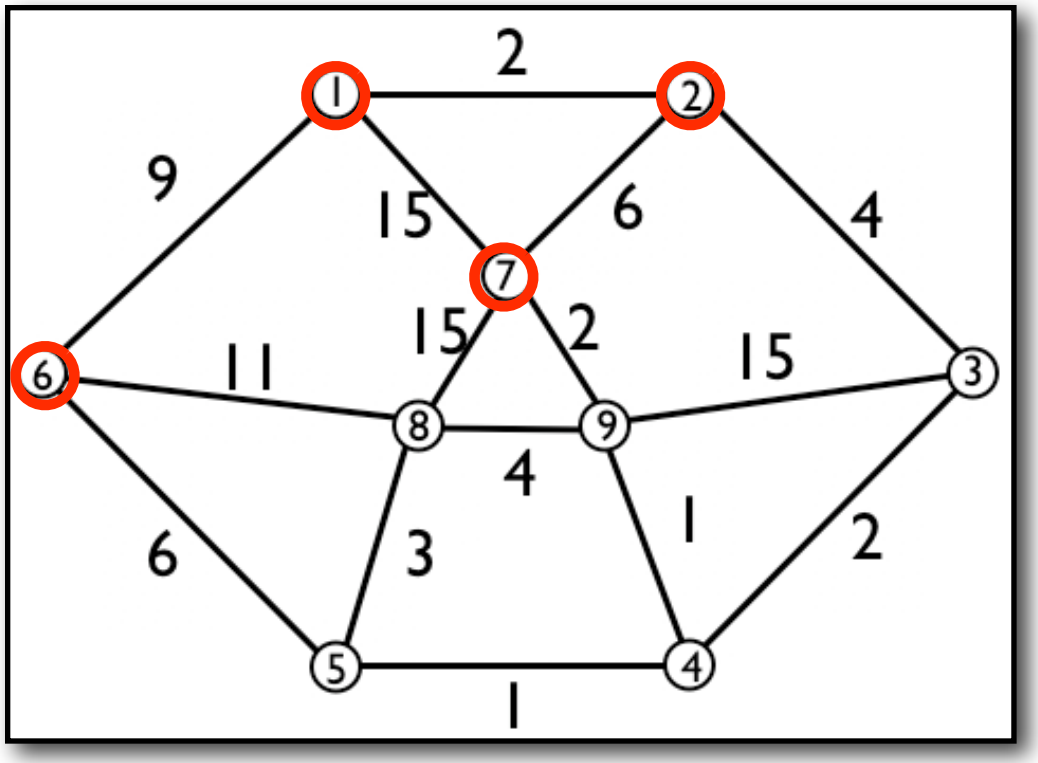
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R
(1,0,1)	(2,2,1) (7,15,1)

Graphenalgorithmen
Kürzeste Wege - Beispiel

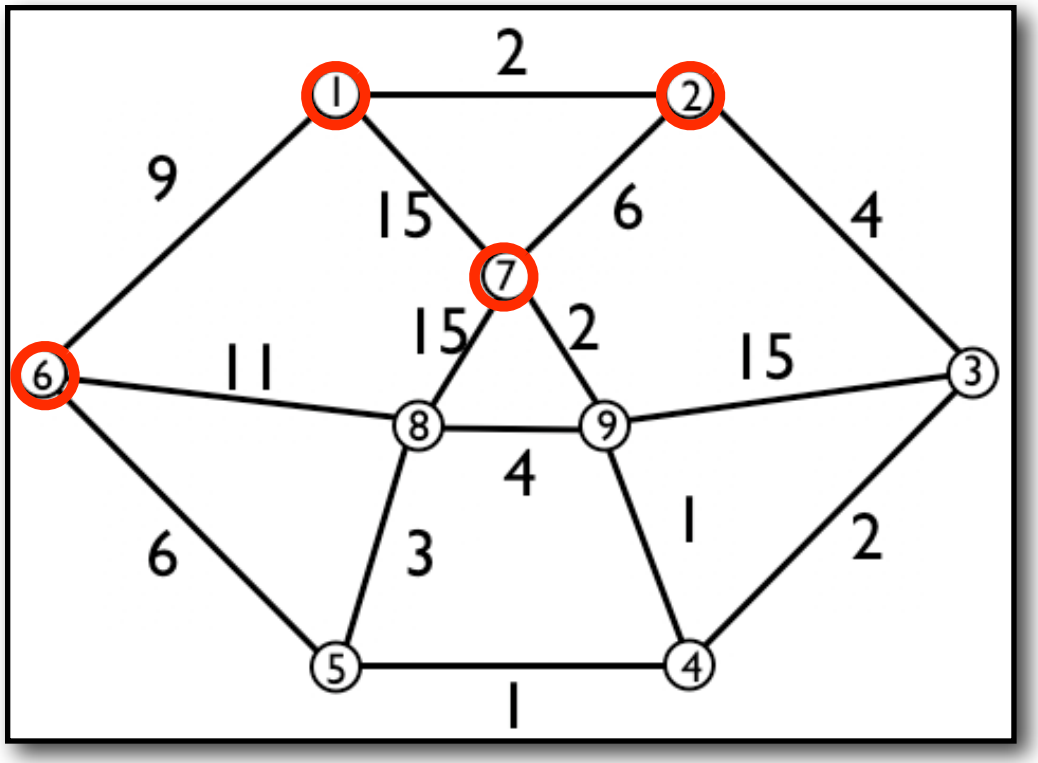
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R
(1,0,1)	(2,2,1)(7,15,1)

Graphenalgorithmen
Kürzeste Wege - Beispiel

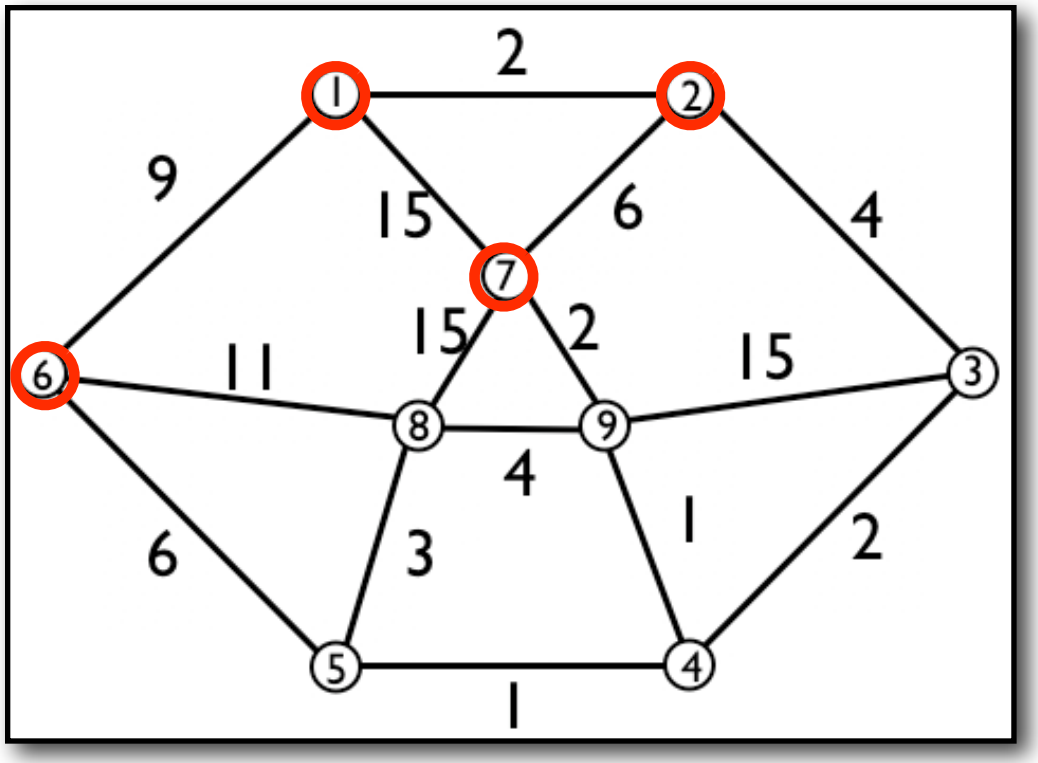
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R
(1,0,1)	(2,2,1) (7,15,1) (6,9,1)

Graphenalgorithmen
Kürzeste Wege - Beispiel

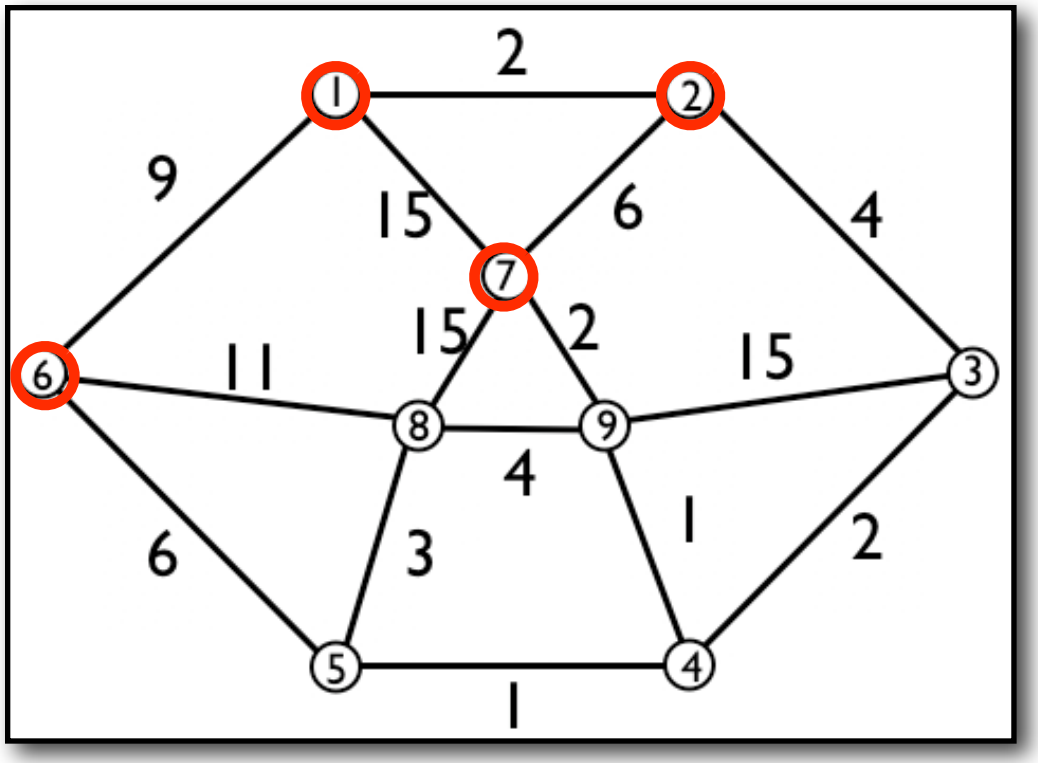
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R
(1,0,1)	(2,2,1) (7,15,1) (6,9,1)

Graphenalgorithmen
Kürzeste Wege - Beispiel

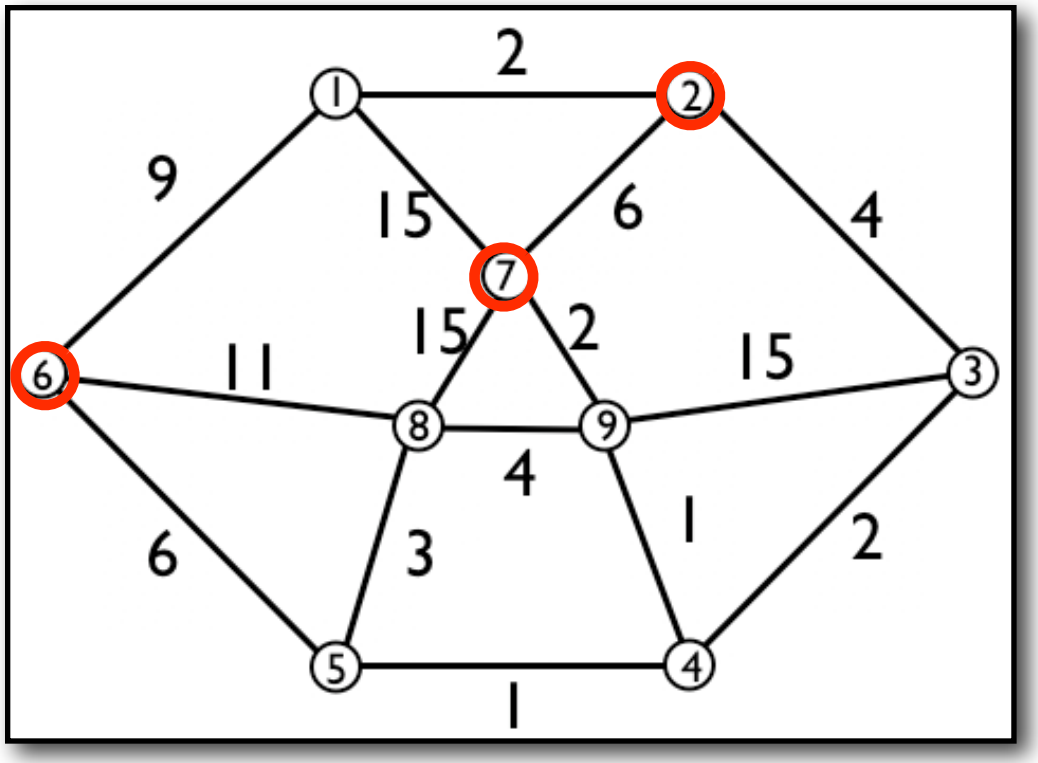
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R
(1,0,1)	(2,2,1) (7,15,1) (6,9,1)
(2,2,1)	

Graphenalgorithmen
Kürzeste Wege - Beispiel

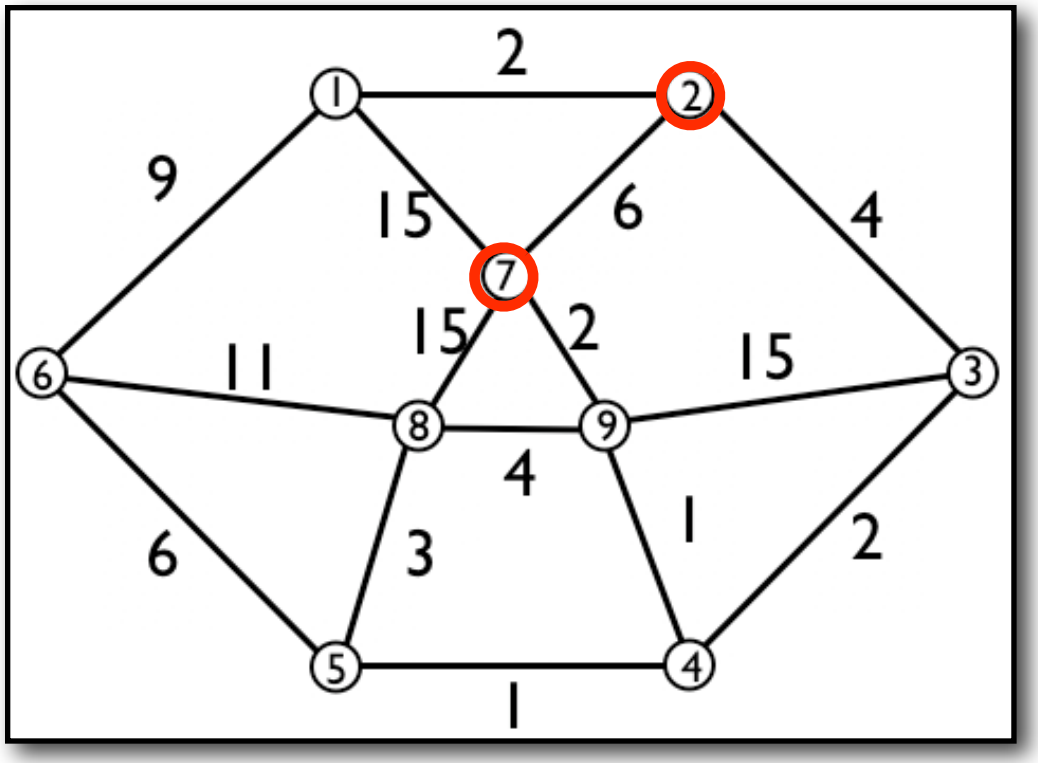
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R
(1,0,1)	(2,2,1) (7,15,1) (6,9,1)
(2,2,1)	

Graphenalgorithmen
Kürzeste Wege - Beispiel

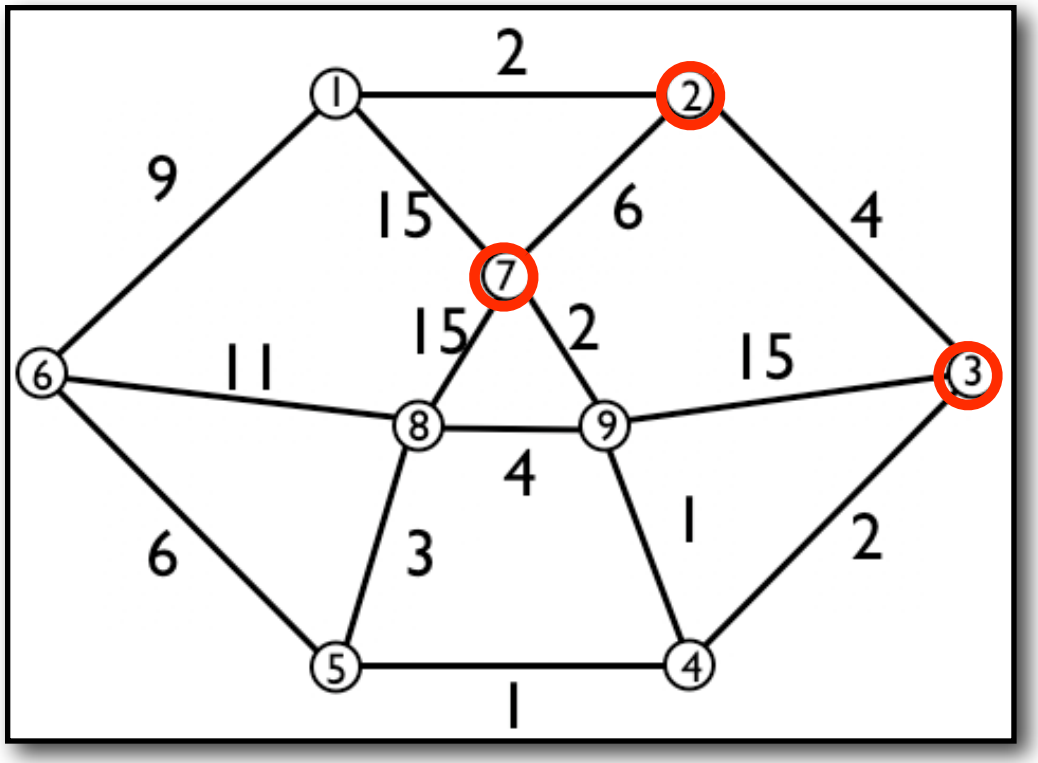
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R
(1,0,1)	(2,2,1) (7,15,1) (6,9,1)
(2,2,1)	

Graphenalgorithmen
Kürzeste Wege - Beispiel

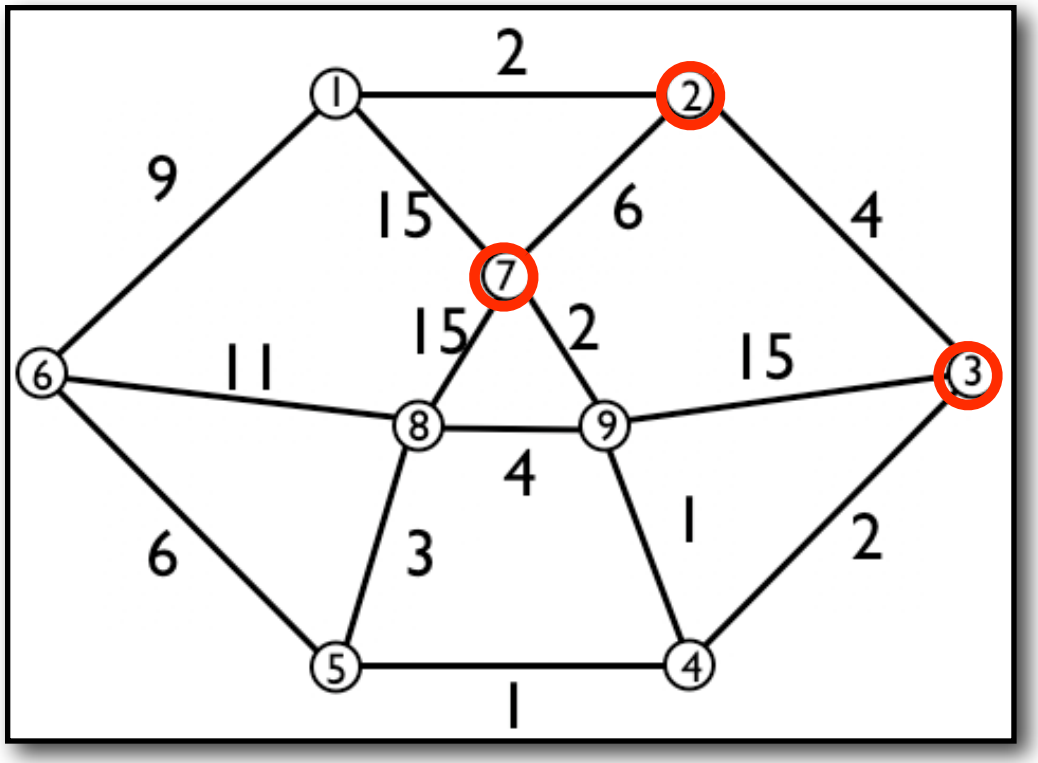
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)			

Graphenalgorithmen
Kürzeste Wege - Beispiel

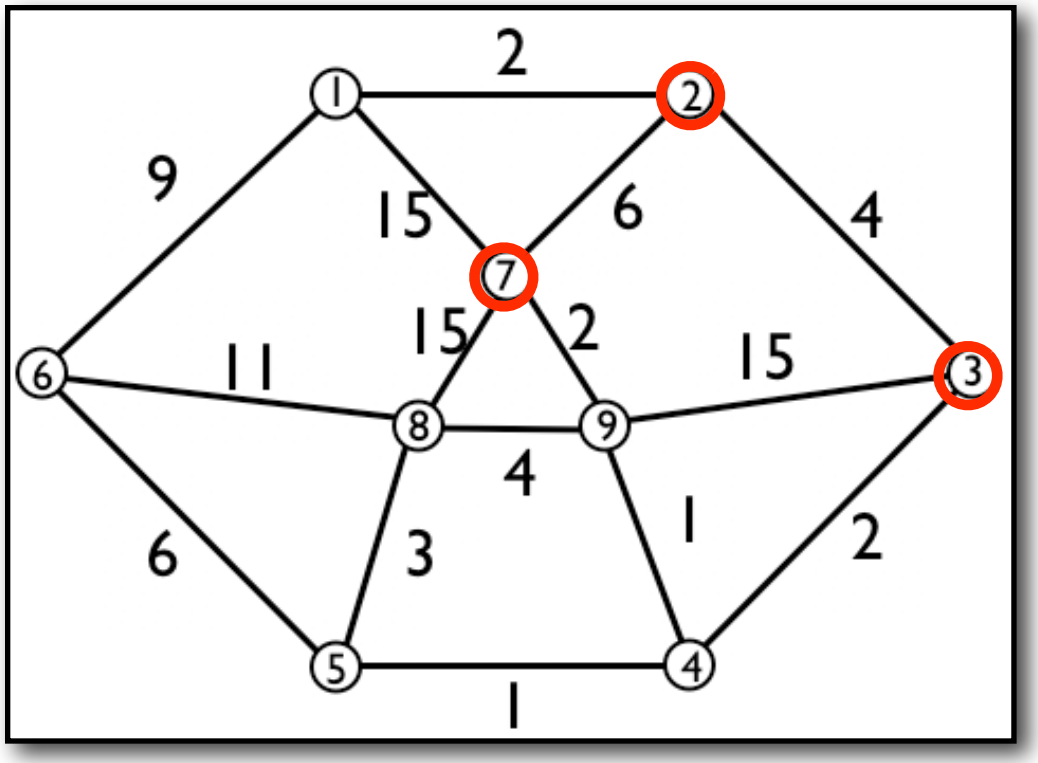
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R
(1,0,1)	(2,2,1) (7,15,1) (6,9,1)
(2,2,1)	(7,8,2)

Graphenalgorithmen
Kürzeste Wege - Beispiel

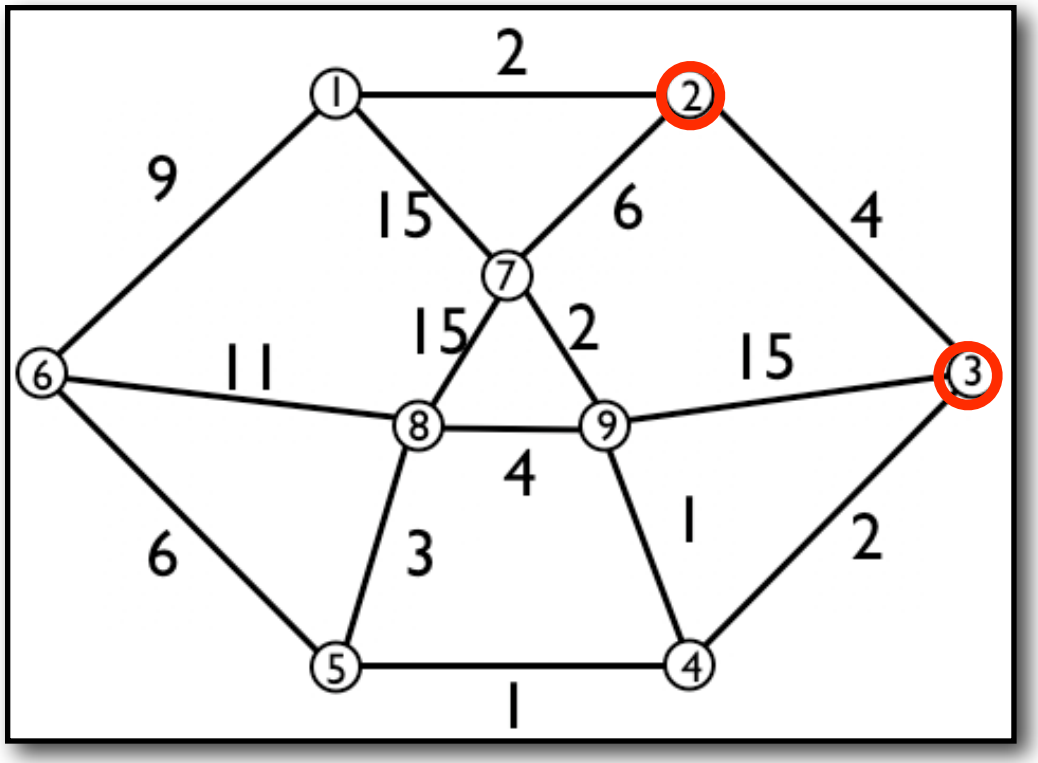
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R
(1,0,1)	(2,2,1) (7,15,1) (6,9,1)
(2,2,1)	(7,8,2) (3,6,2)

Graphenalgorithmen
Kürzeste Wege - Beispiel

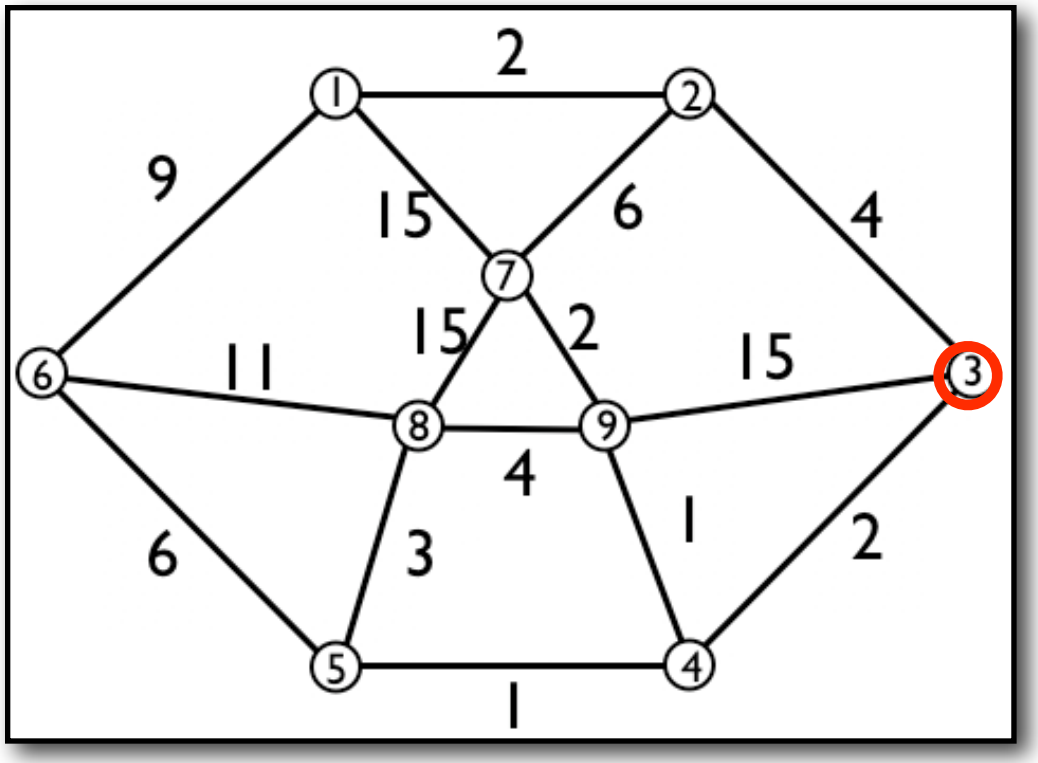
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R
(1,0,1)	(2,2,1) (7,15,1) (6,9,1)
(2,2,1)	(7,8,2) (3,6,2)

Graphenalgorithmen
Kürzeste Wege - Beispiel

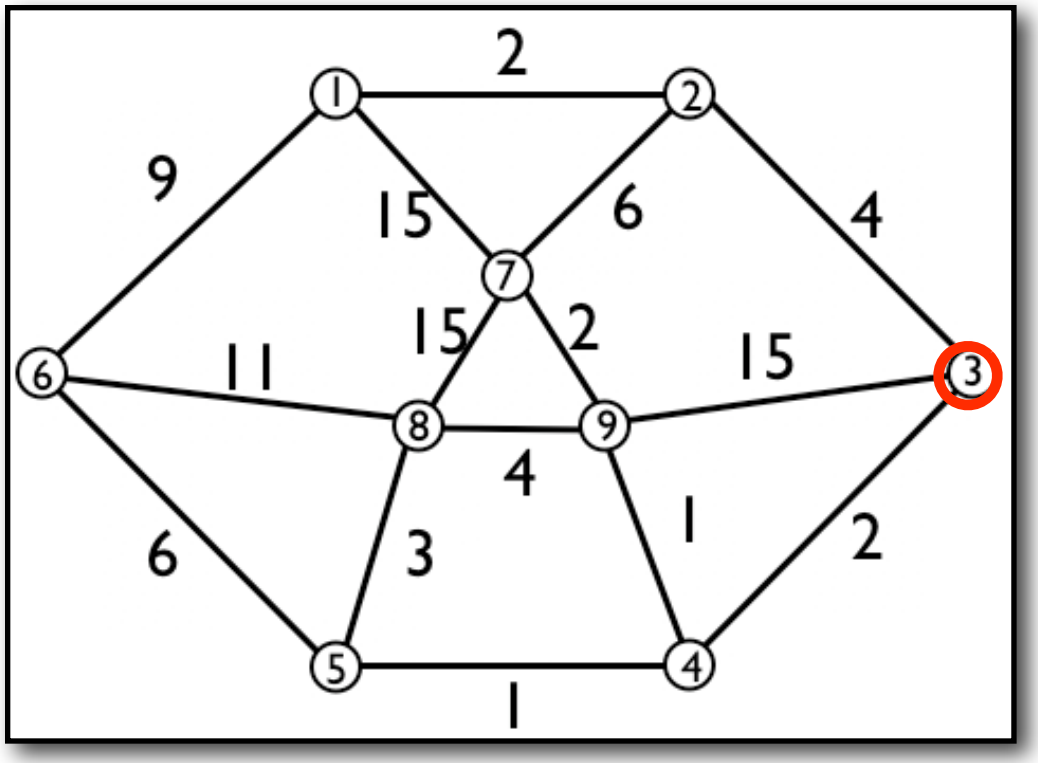
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R
(1,0,1)	(2,2,1) (7,15,1) (6,9,1)
(2,2,1)	(7,8,2) (3,6,2)

Graphenalgorithmen
Kürzeste Wege - Beispiel

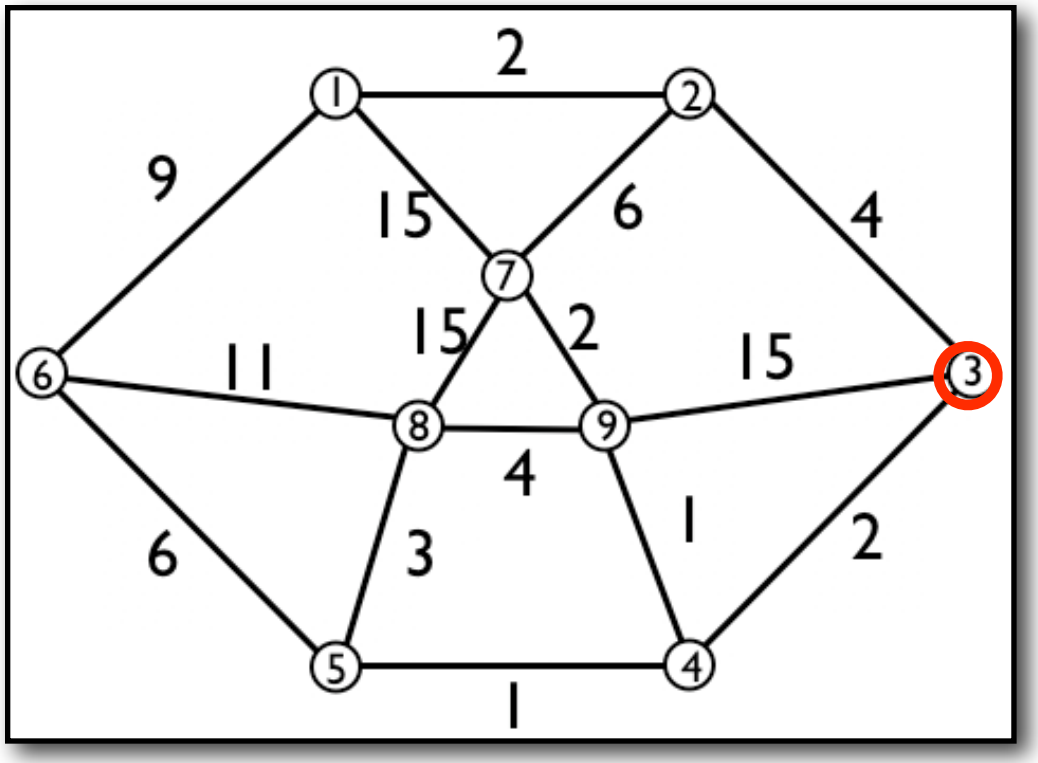
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	

Graphenalgorithmen
Kürzeste Wege - Beispiel

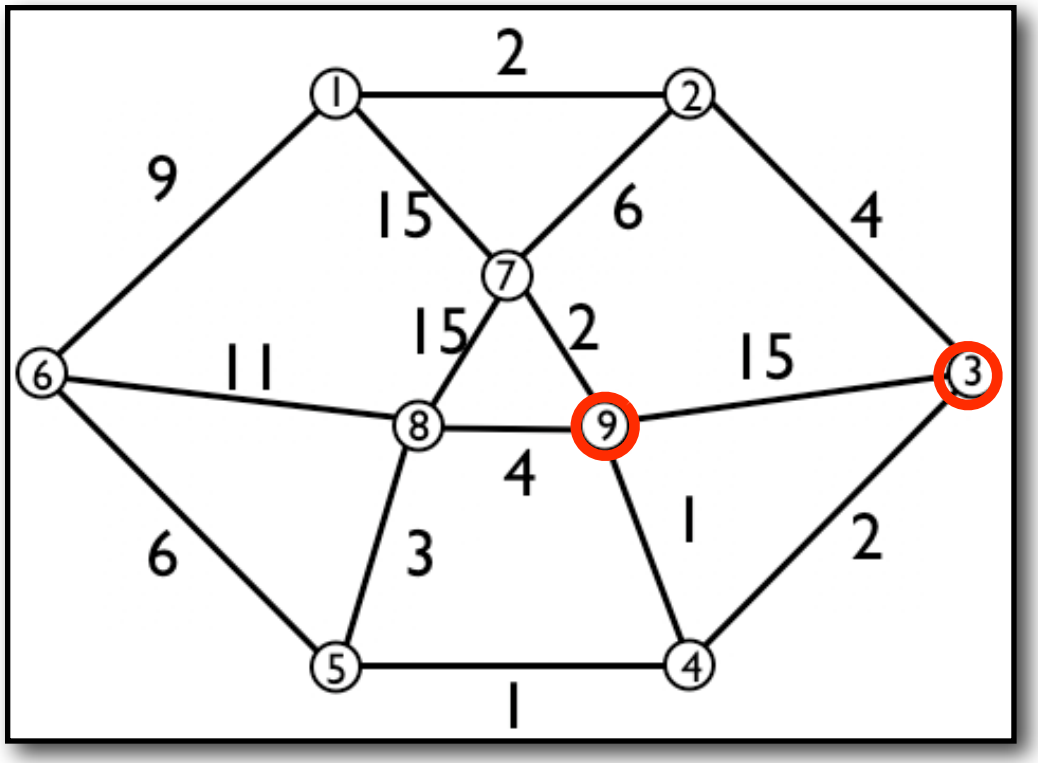
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)			

Graphenalgorithmen
Kürzeste Wege - Beispiel

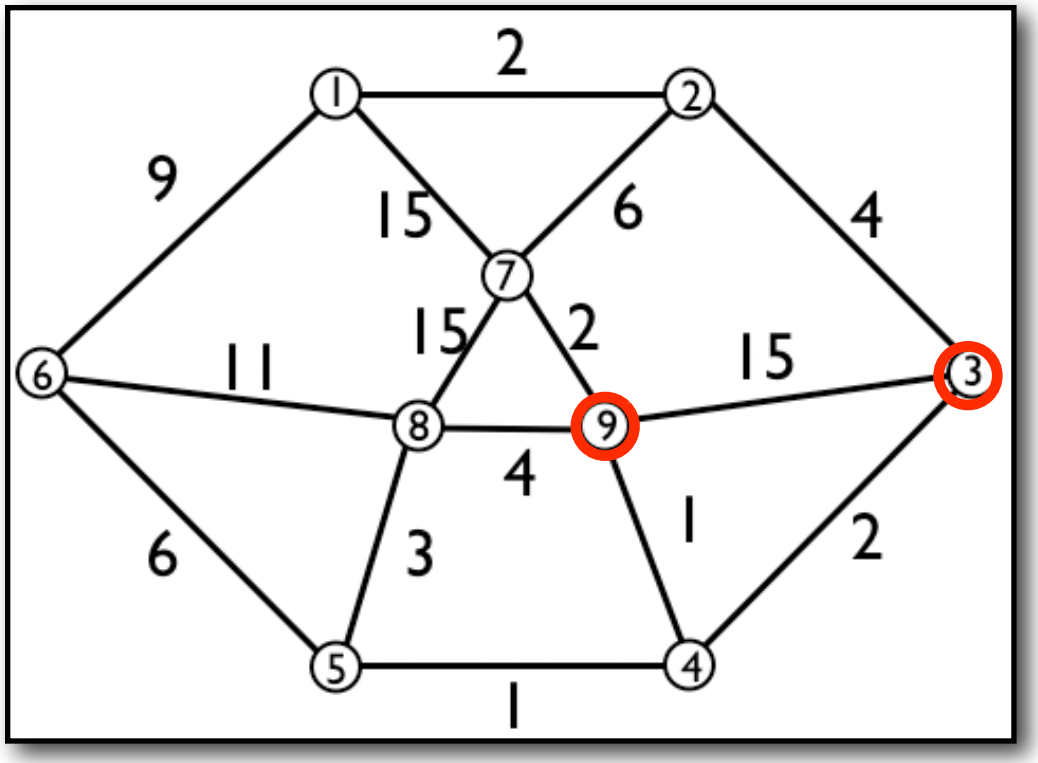
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)			

Graphenalgorithmen
Kürzeste Wege - Beispiel

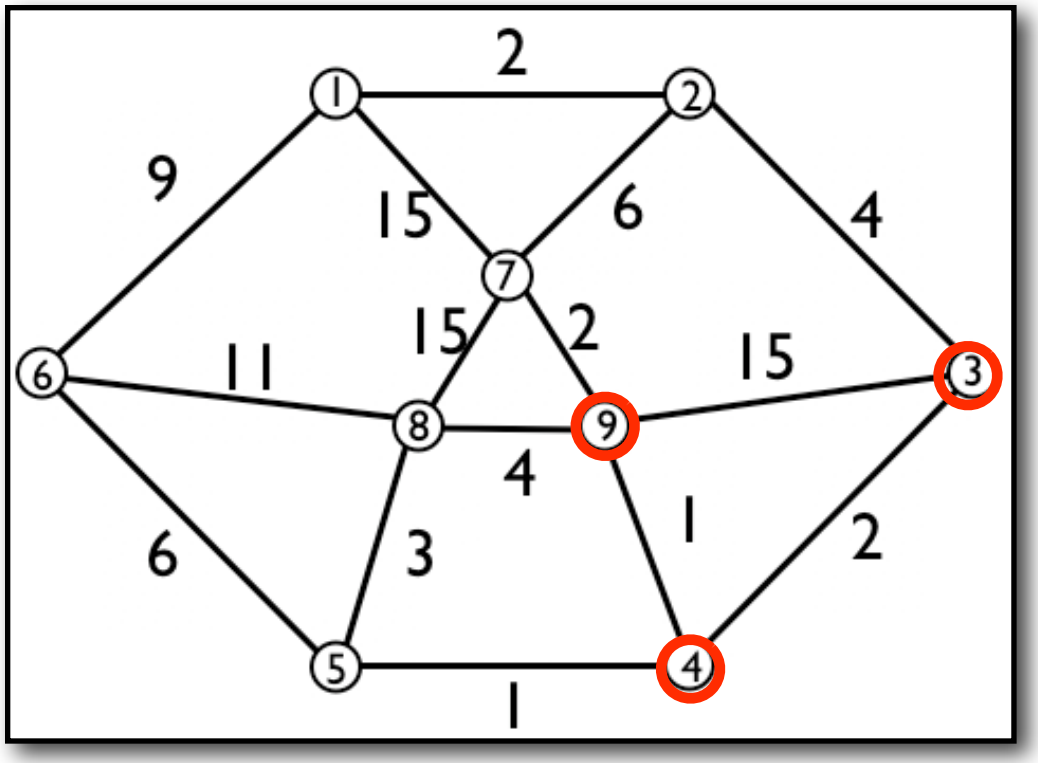
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)		

Graphenalgorithmen
Kürzeste Wege - Beispiel

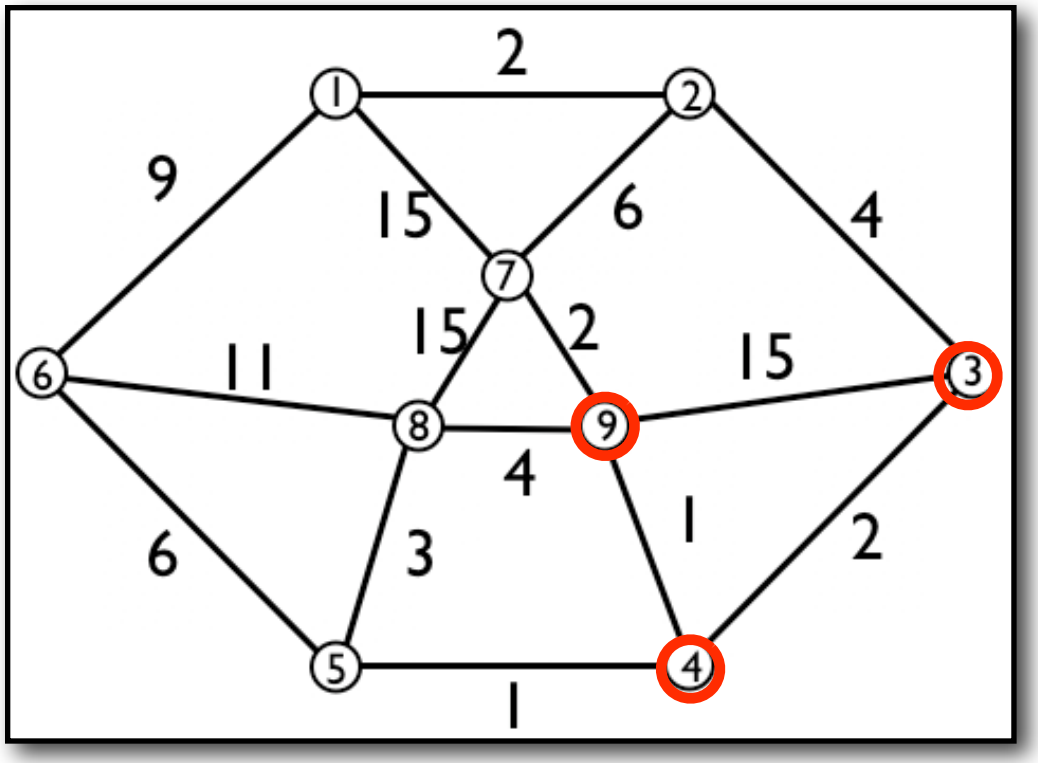
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)		

Graphenalgorithmen
Kürzeste Wege - Beispiel

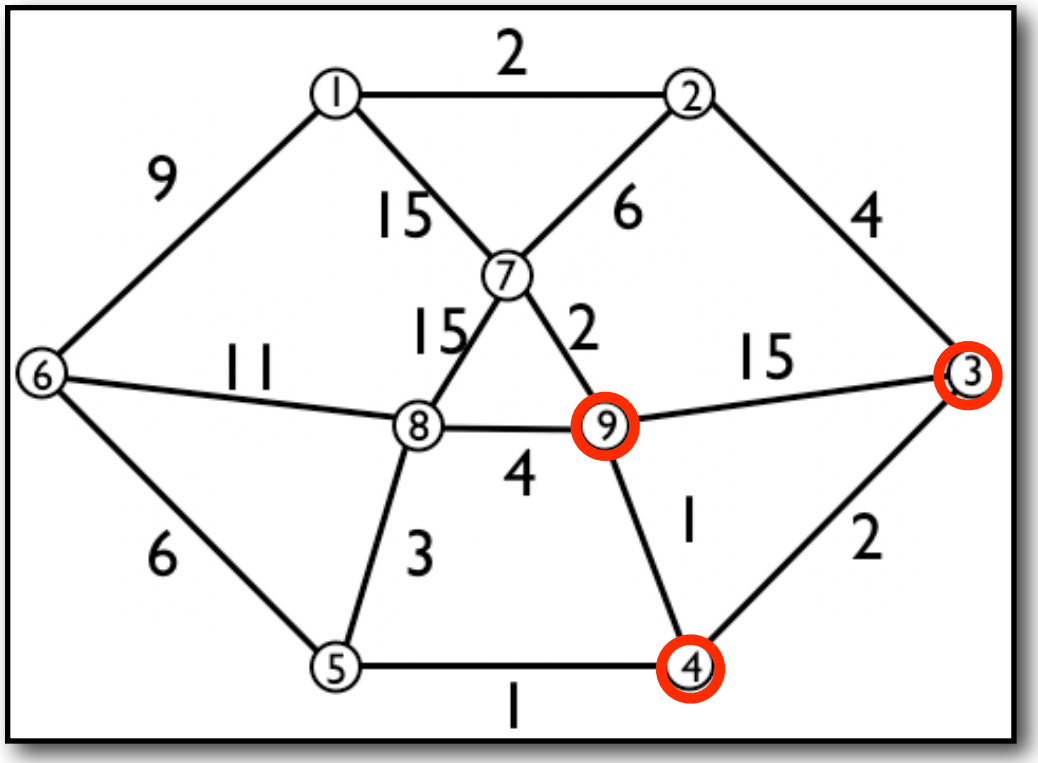
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	

Graphenalgorithmen
Kürzeste Wege - Beispiel

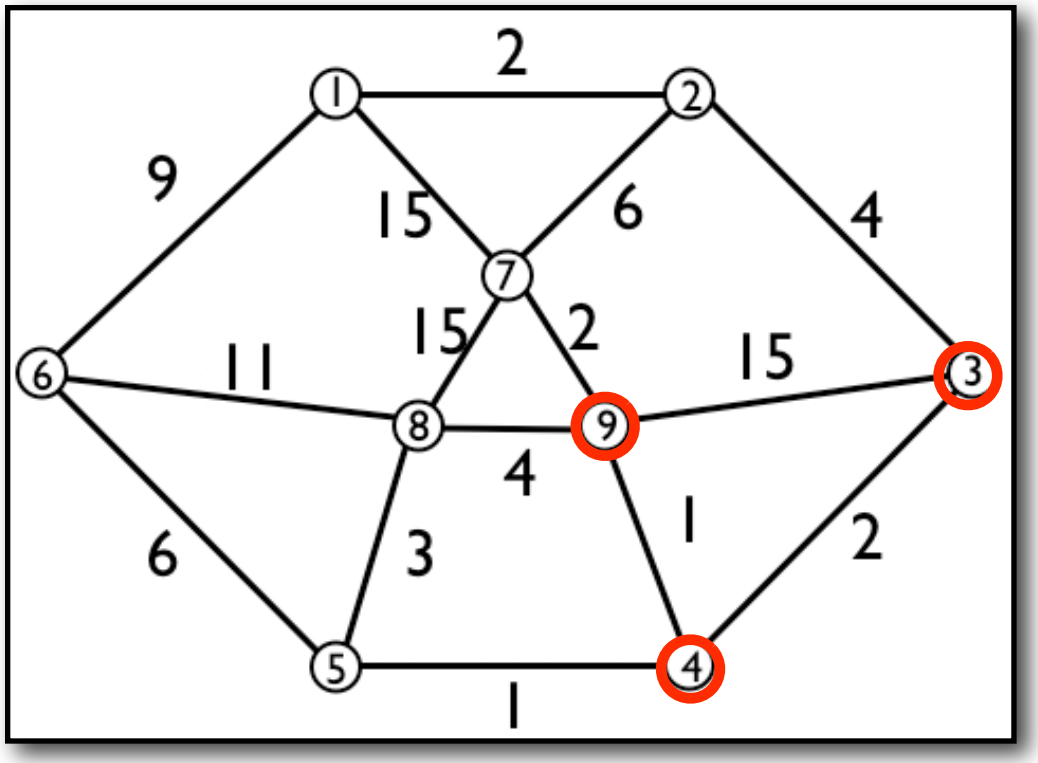
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	

Graphenalgorithmen
Kürzeste Wege - Beispiel

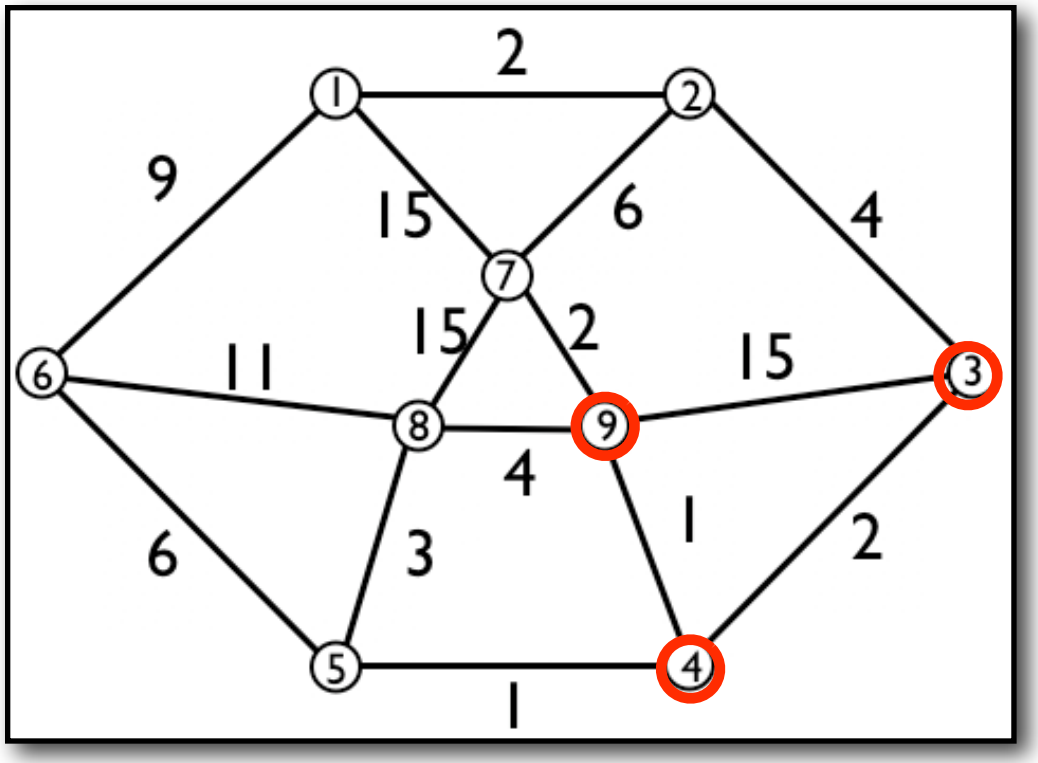
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	

Graphenalgorithmen
Kürzeste Wege - Beispiel

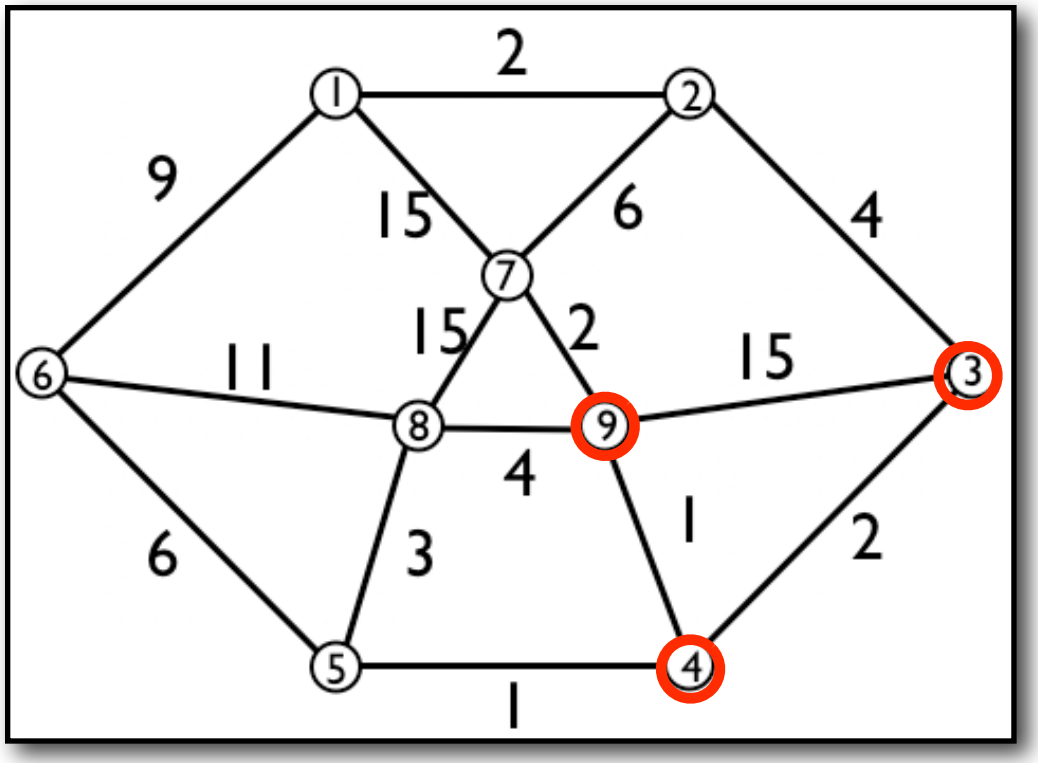
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	
(7,8,2)			

Graphenalgorithmen
Kürzeste Wege - Beispiel

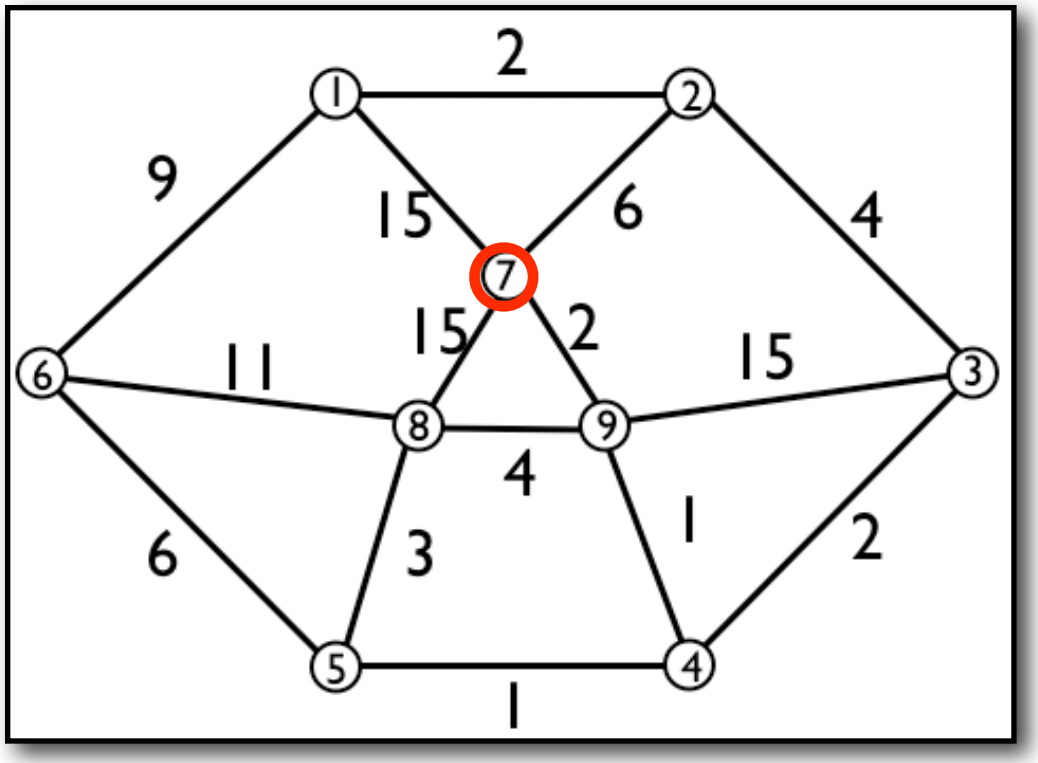
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	
(7,8,2)			

Graphenalgorithmen
Kürzeste Wege - Beispiel

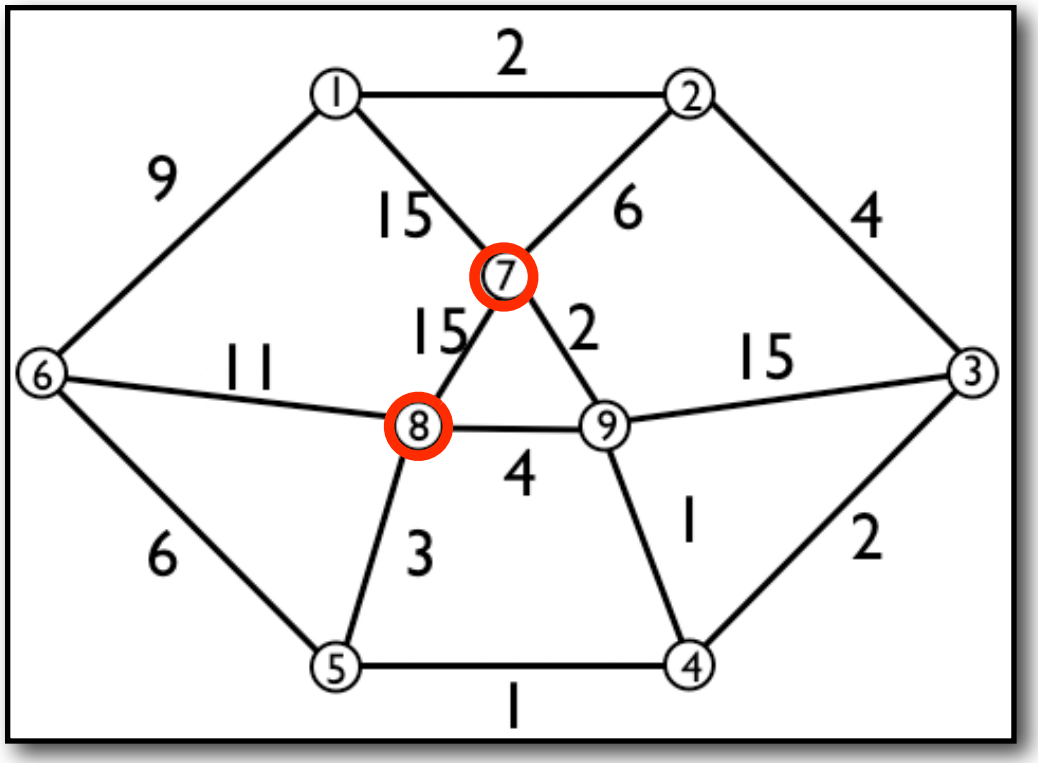
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	
(7,8,2)			

Graphenalgorithmen
Kürzeste Wege - Beispiel

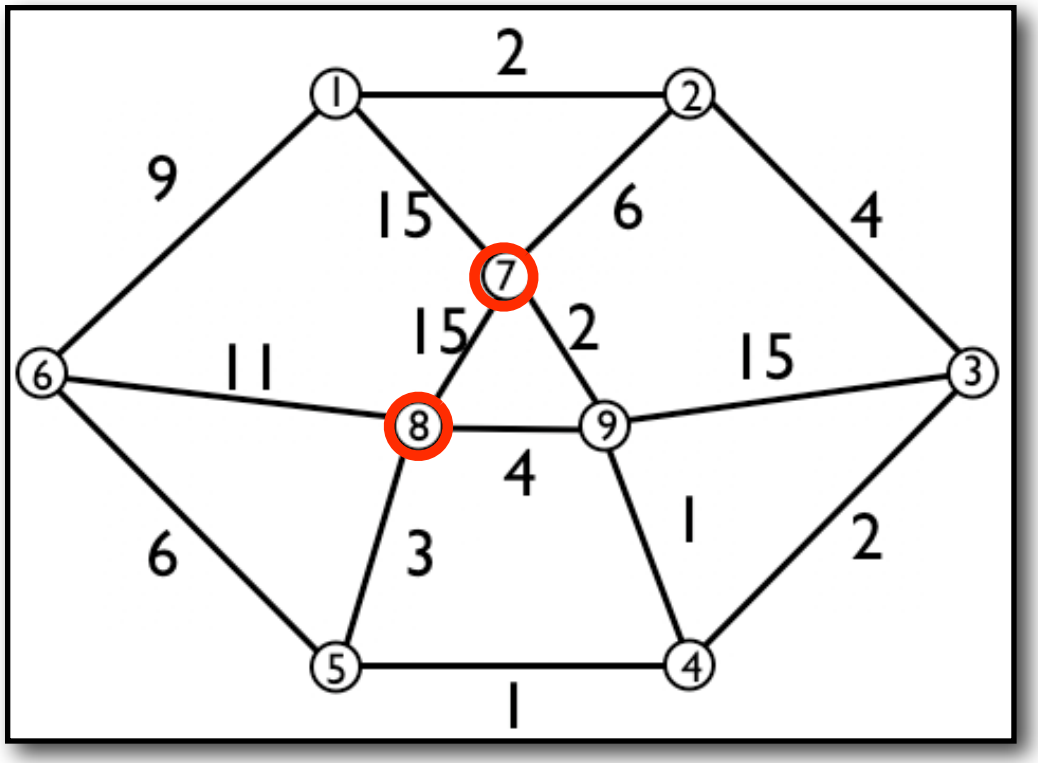
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	
(7,8,2)			

Graphenalgorithmen
Kürzeste Wege - Beispiel

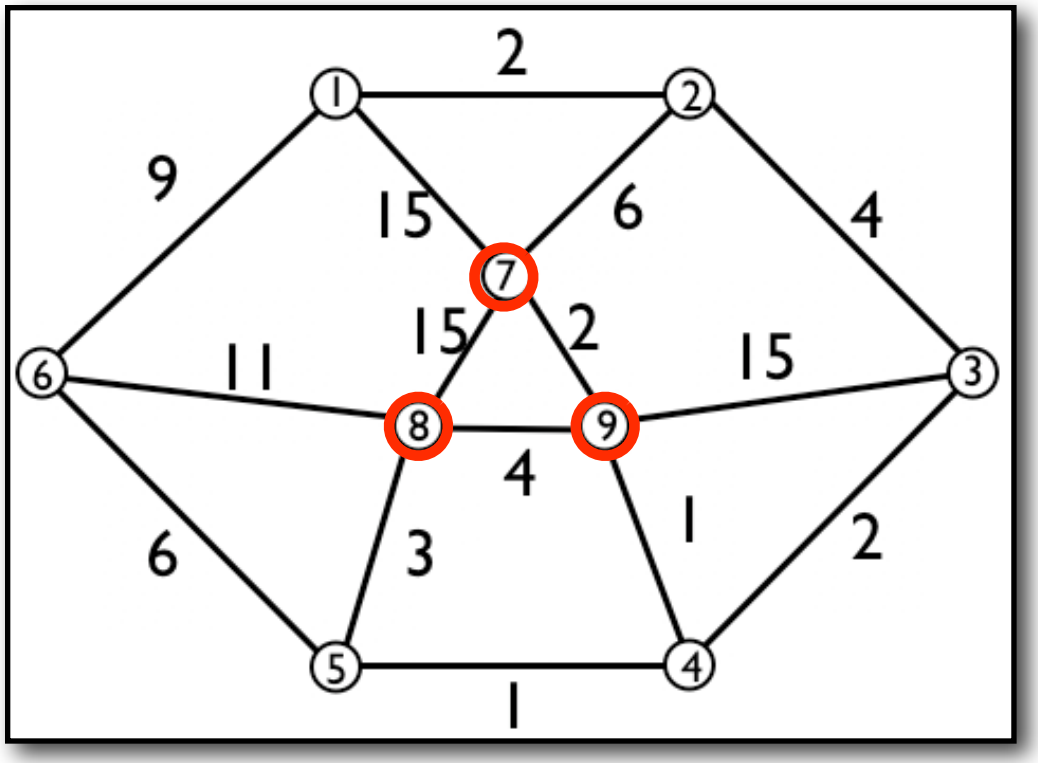
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	
(7,8,2)	(8,23,7)		

Graphenalgorithmen
Kürzeste Wege - Beispiel

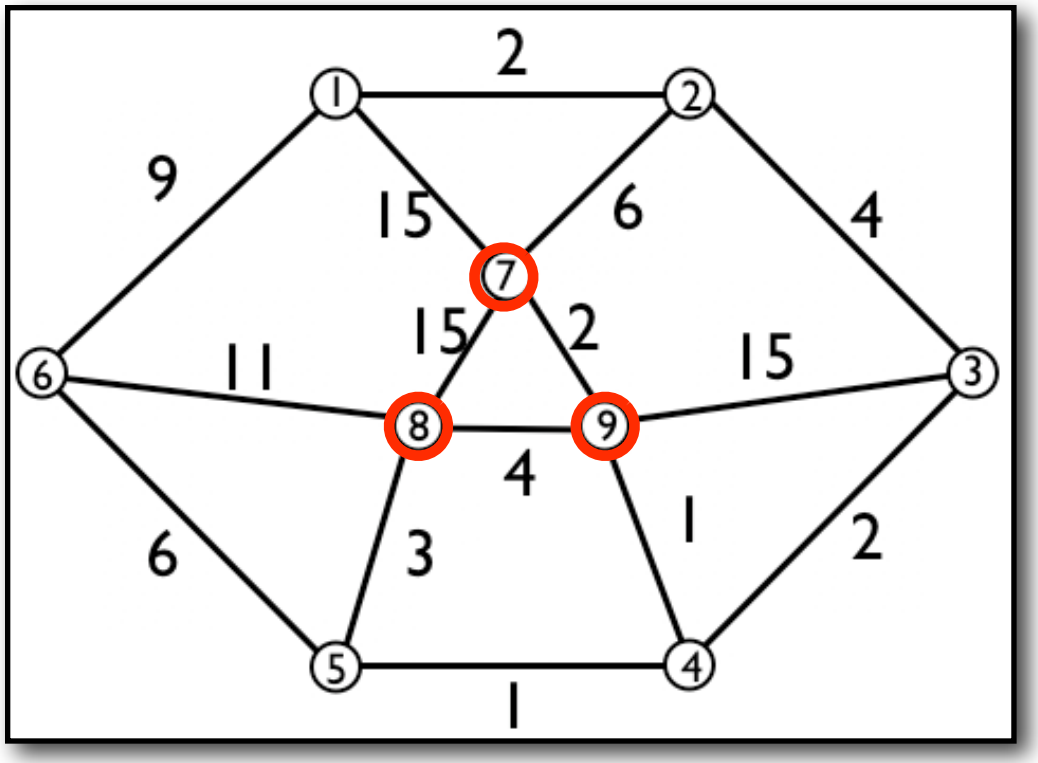
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	
(7,8,2)	(8,23,7)		

Graphenalgorithmen
Kürzeste Wege - Beispiel

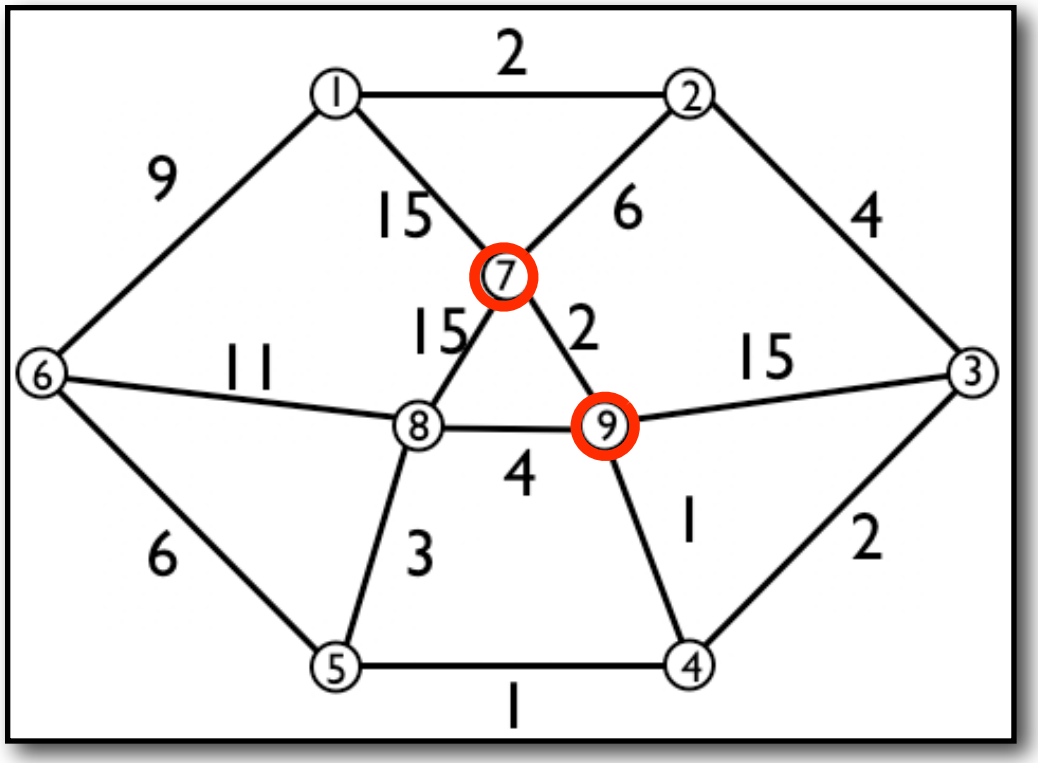
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	
(7,8,2)	(8,23,7)	(9,10,7)	

Graphenalgorithmen
Kürzeste Wege - Beispiel

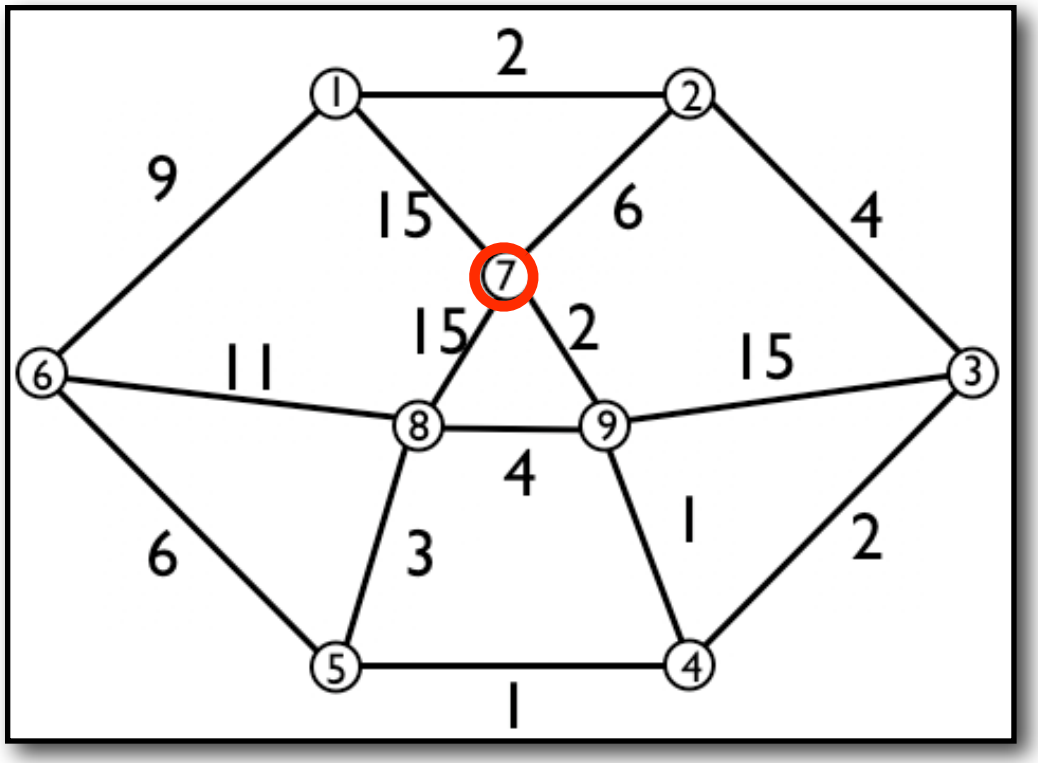
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	
(7,8,2)	(8,23,7)	(9,10,7)	

Graphenalgorithmen
Kürzeste Wege - Beispiel

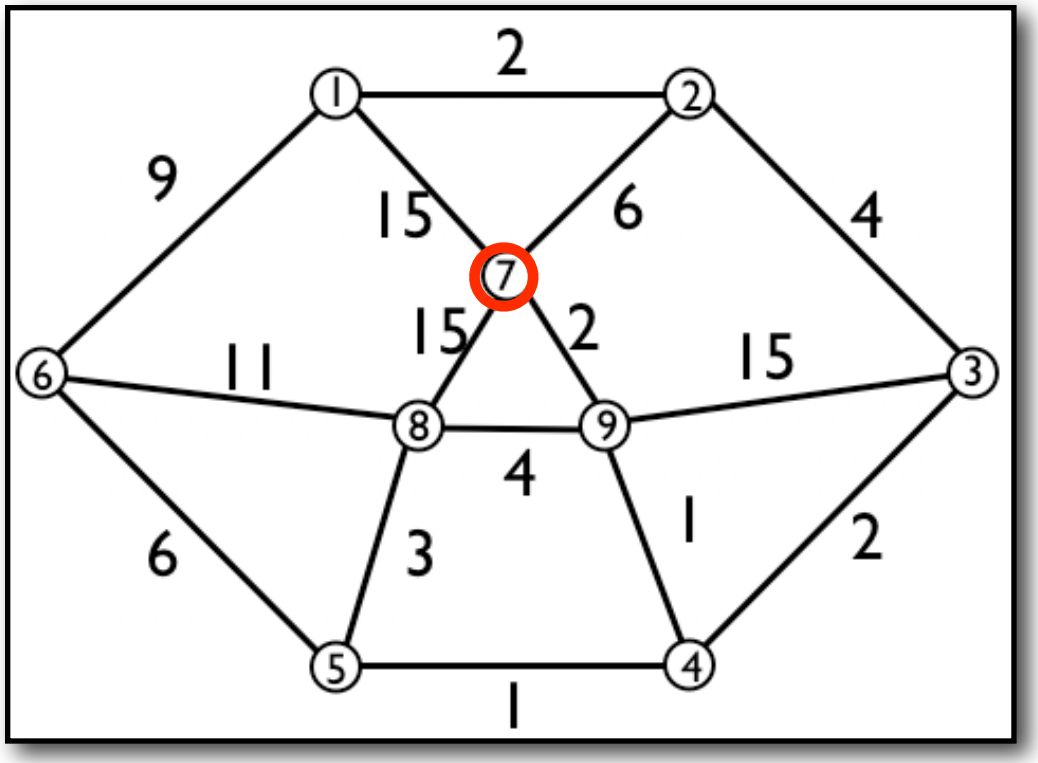
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	
(7,8,2)	(8,23,7)	(9,10,7)	

Graphenalgorithmen
Kürzeste Wege - Beispiel

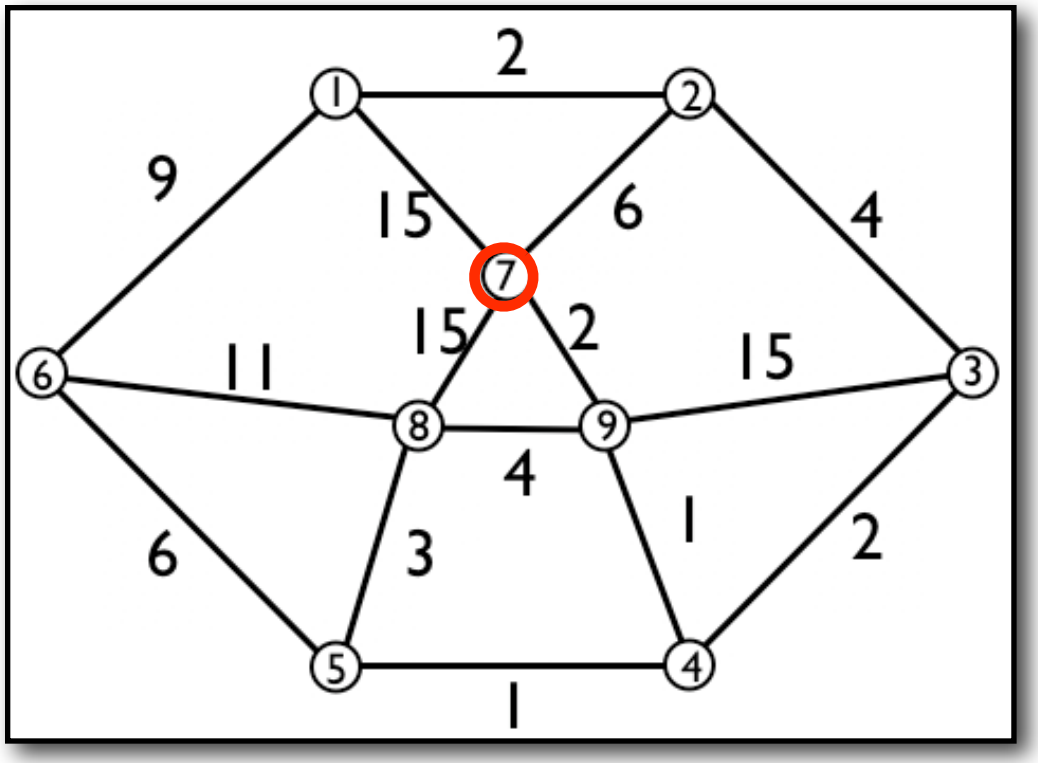
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	
(7,8,2)	(8,23,7)	(9,10,7)	

Graphenalgorithmen
Kürzeste Wege - Beispiel

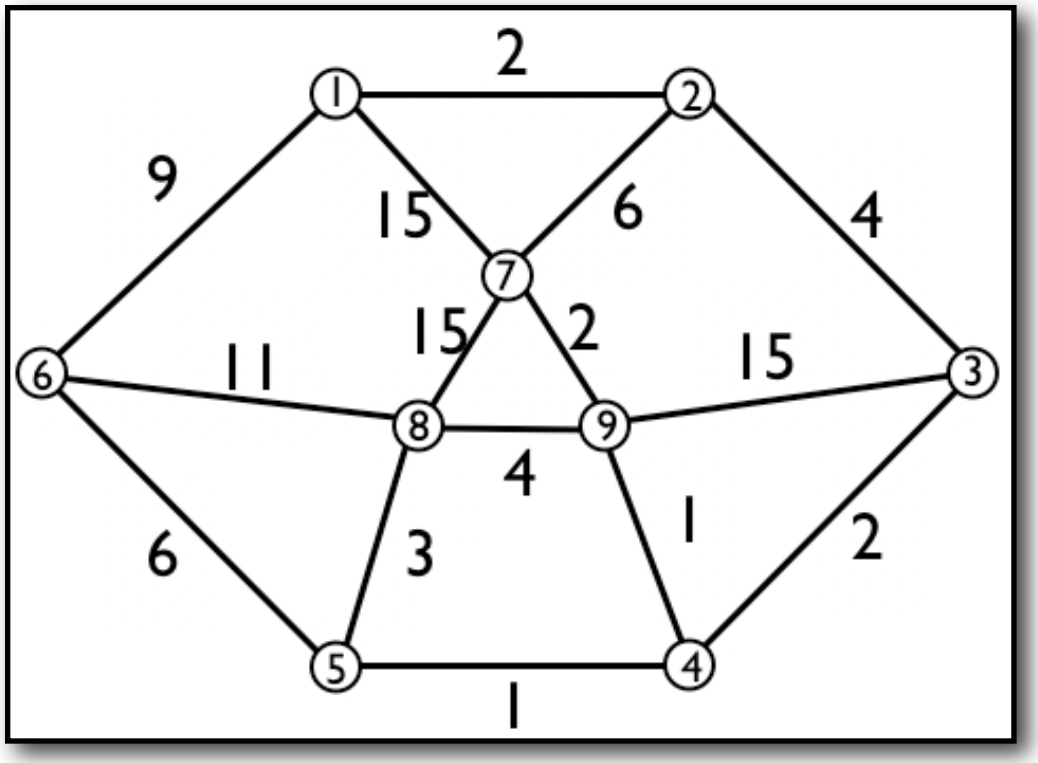
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	
(7,8,2)	(8,23,7)	(9,10,7)	
(4,8,3)			

Graphenalgorithmen
Kürzeste Wege - Beispiel

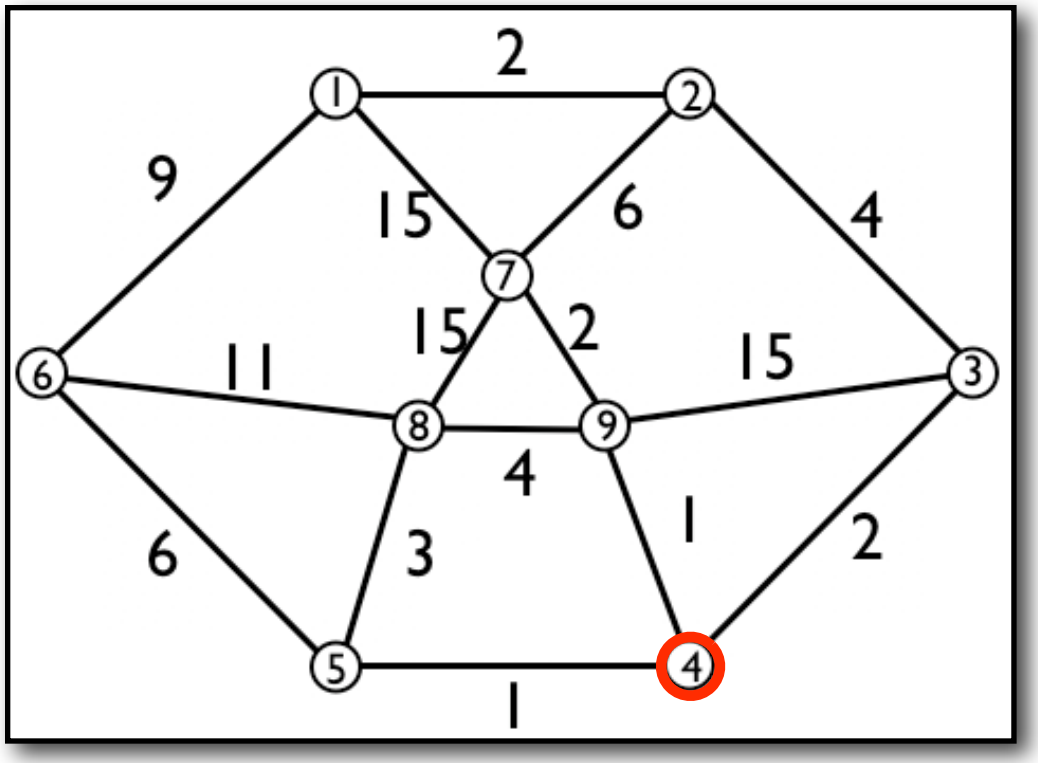
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	
(7,8,2)	(8,23,7)	(9,10,7)	
(4,8,3)			

Graphenalgorithmen
Kürzeste Wege - Beispiel

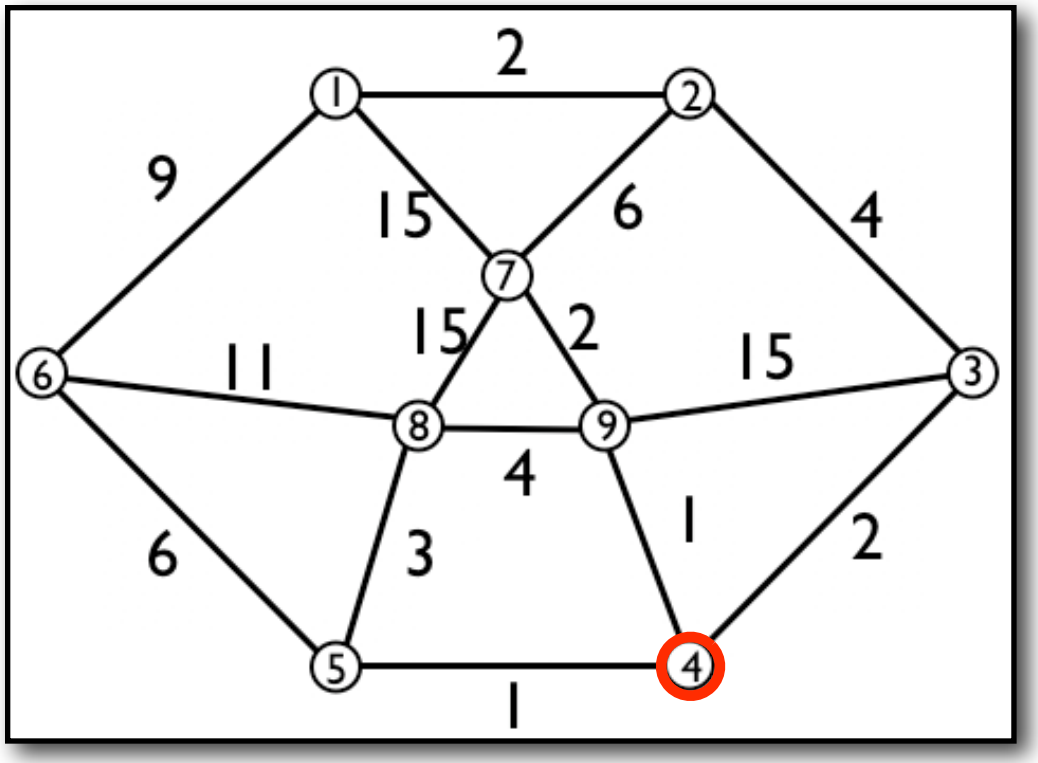
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	
(7,8,2)	(8,23,7)	(9,10,7)	
(4,8,3)			

Graphenalgorithmen
Kürzeste Wege - Beispiel

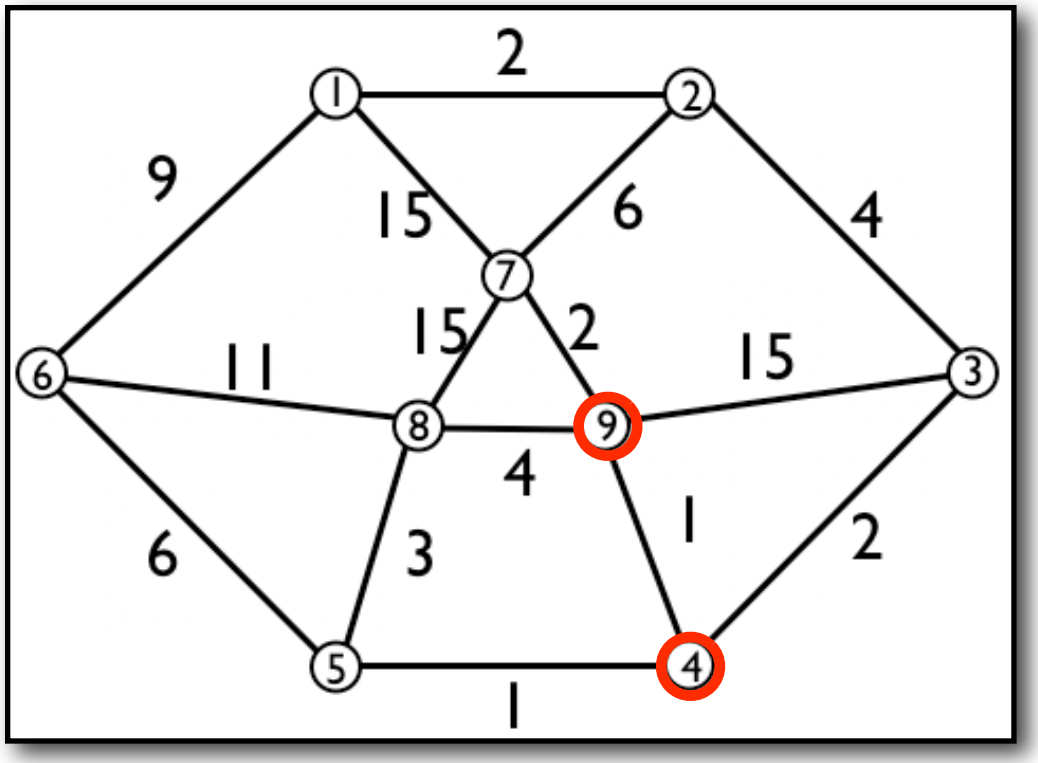
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	
(7,8,2)	(8,23,7)	(9,10,7)	
(4,8,3)			

Graphenalgorithmen
Kürzeste Wege - Beispiel

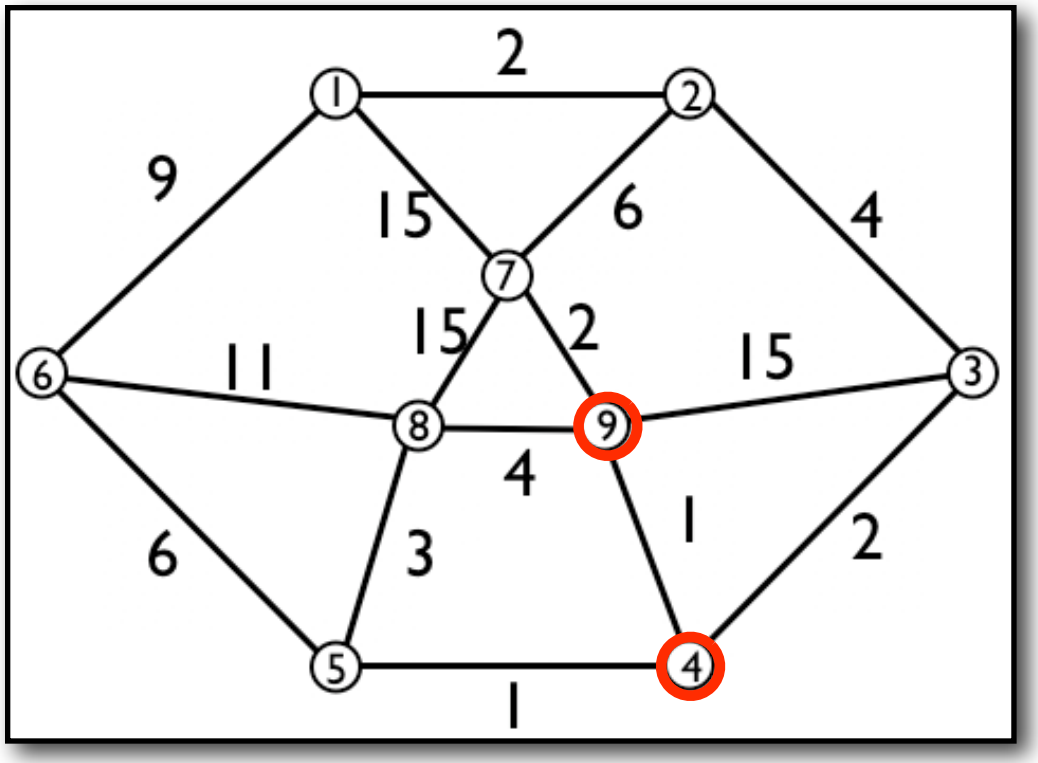
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	
(7,8,2)	(8,23,7)	(9,10,7)	
(4,8,3)			

Graphenalgorithmen
Kürzeste Wege - Beispiel

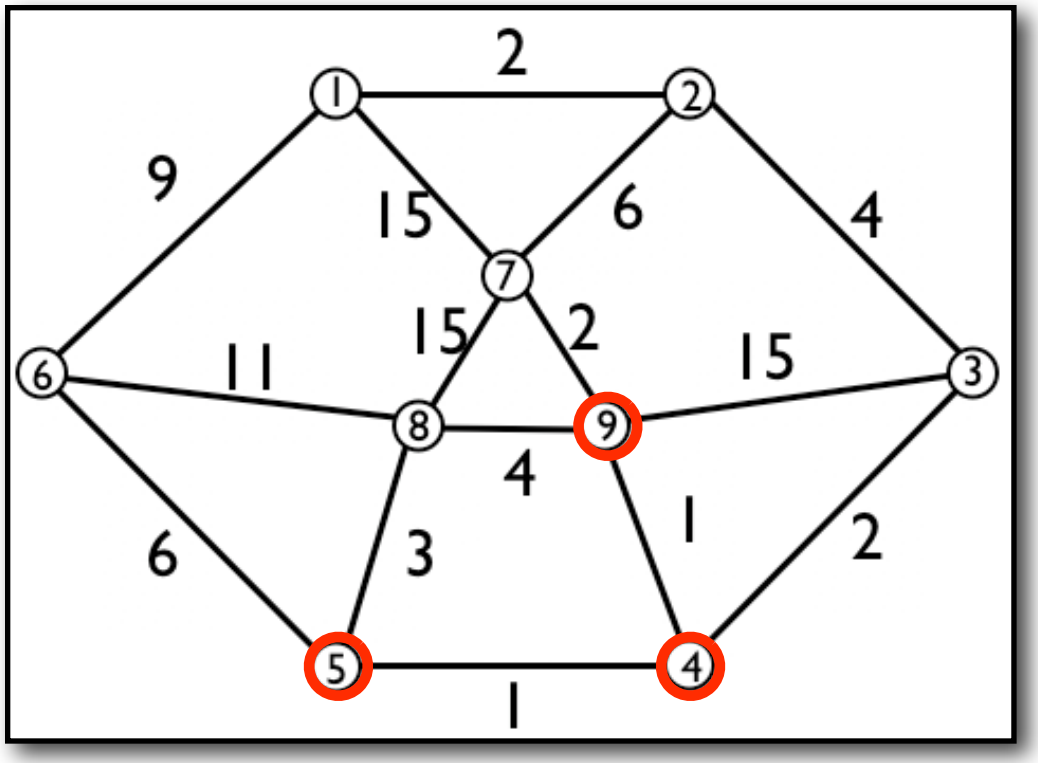
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	
(7,8,2)	(8,23,7)	(9,10,7)	
(4,8,3)	(9,9,4)		

Graphenalgorithmen
Kürzeste Wege - Beispiel

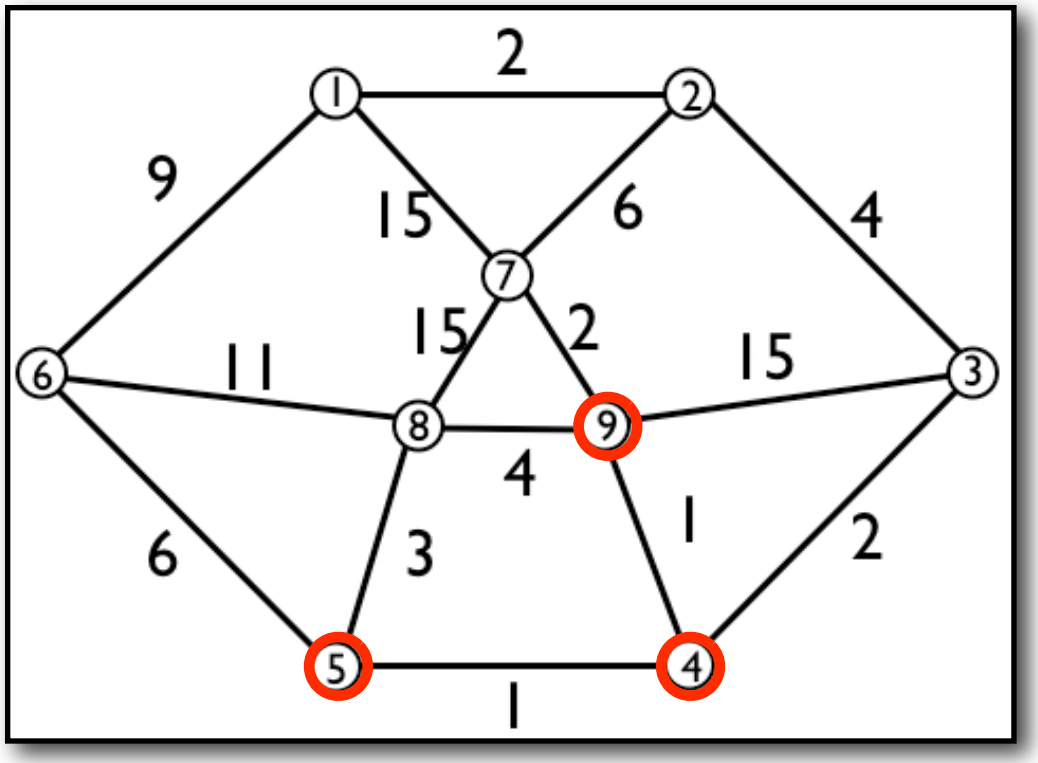
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	
(7,8,2)	(8,23,7)	(9,10,7)	
(4,8,3)	(9,9,4)		

Graphenalgorithmen
Kürzeste Wege - Beispiel

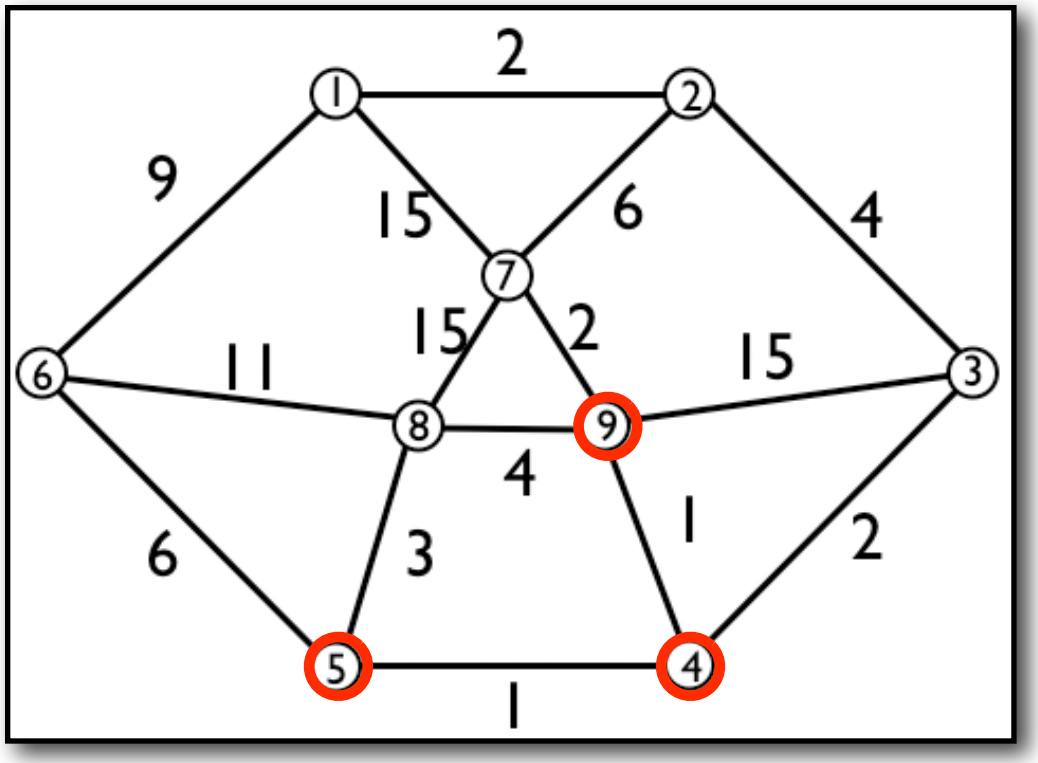
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	
(7,8,2)	(8,23,7)	(9,10,7)	
(4,8,3)	(9,9,4)	(5,9,4)	

Graphenalgorithmen
Kürzeste Wege - Beispiel

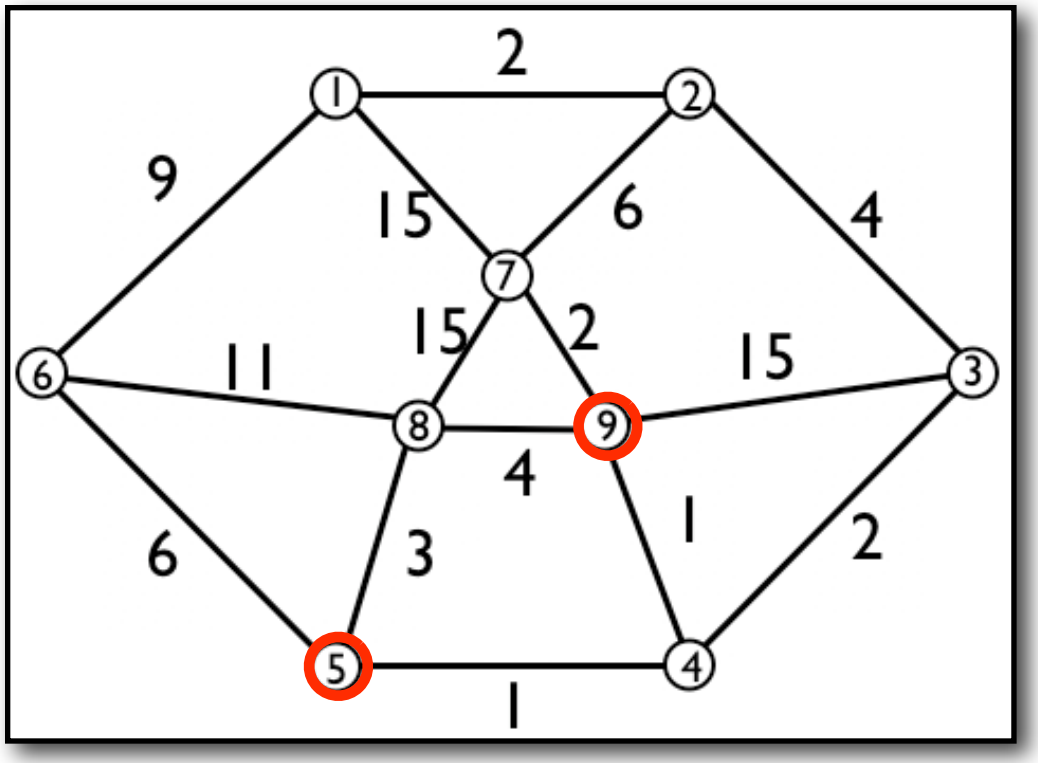
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	
(7,8,2)	(8,23,7)	(9,10,7)	
(4,8,3)	(9,9,4)	(5,9,4)	

Graphenalgorithmen
Kürzeste Wege - Beispiel

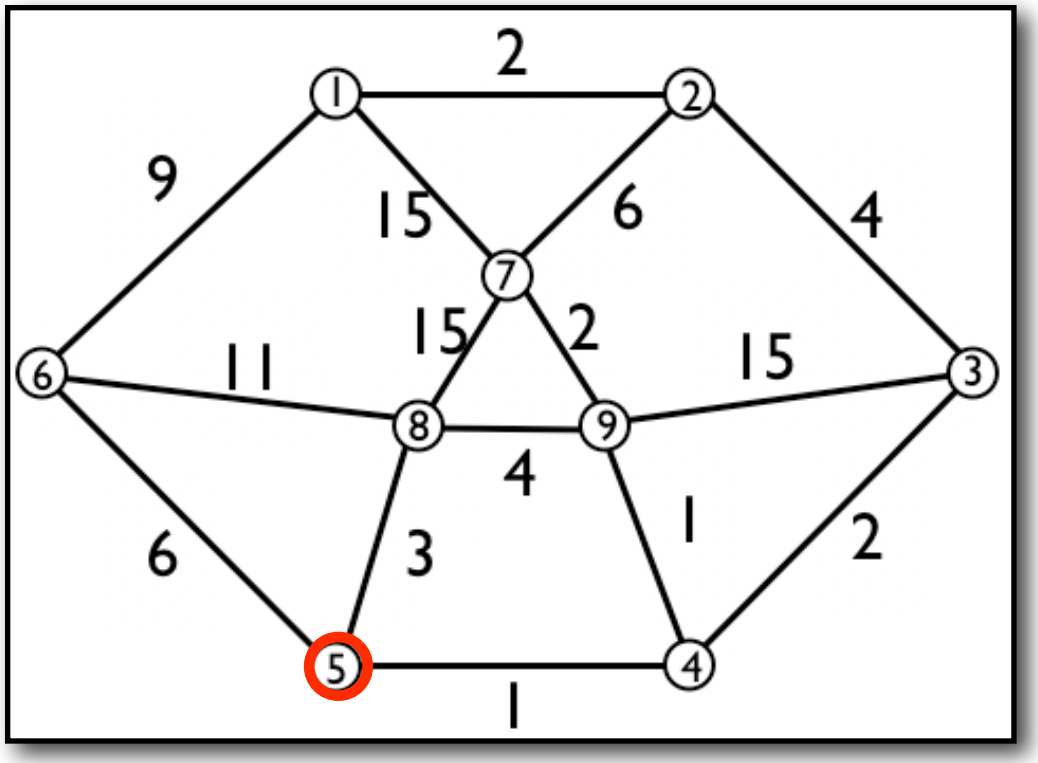
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	
(7,8,2)	(8,23,7)	(9,10,7)	
(4,8,3)	(9,9,4)	(5,9,4)	

Graphenalgorithmen
Kürzeste Wege - Beispiel

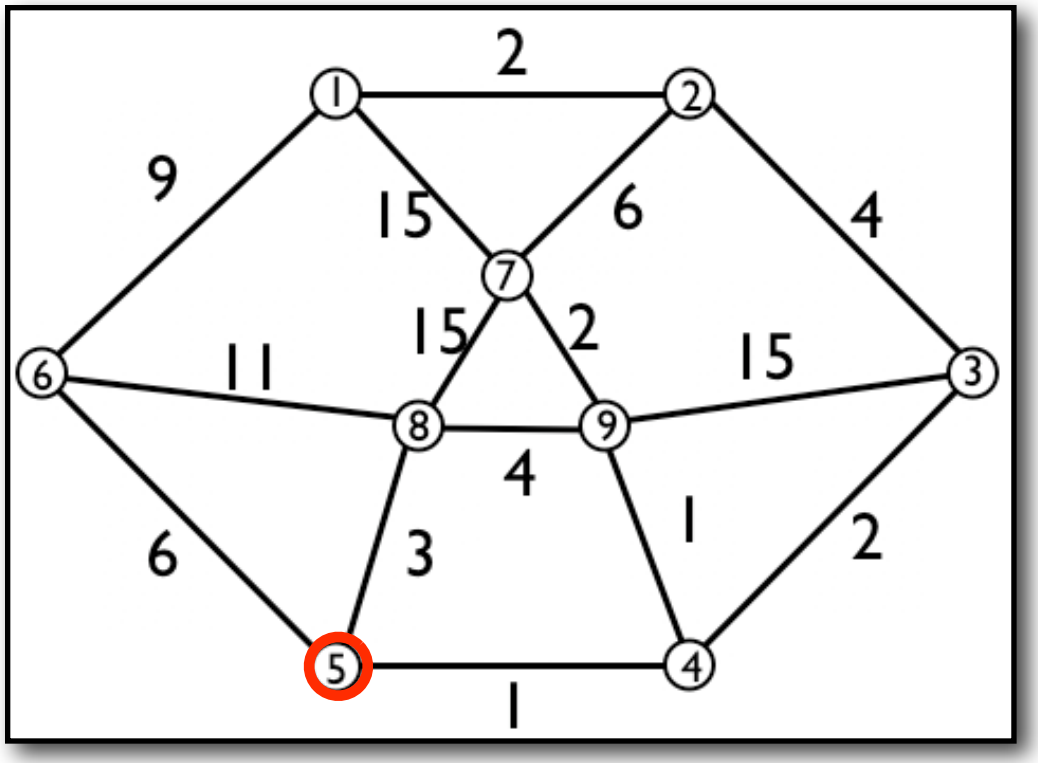
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	
(7,8,2)	(8,23,7)	(9,10,7)	
(4,8,3)	(9,9,4)	(5,9,4)	

Graphenalgorithmen
Kürzeste Wege - Beispiel

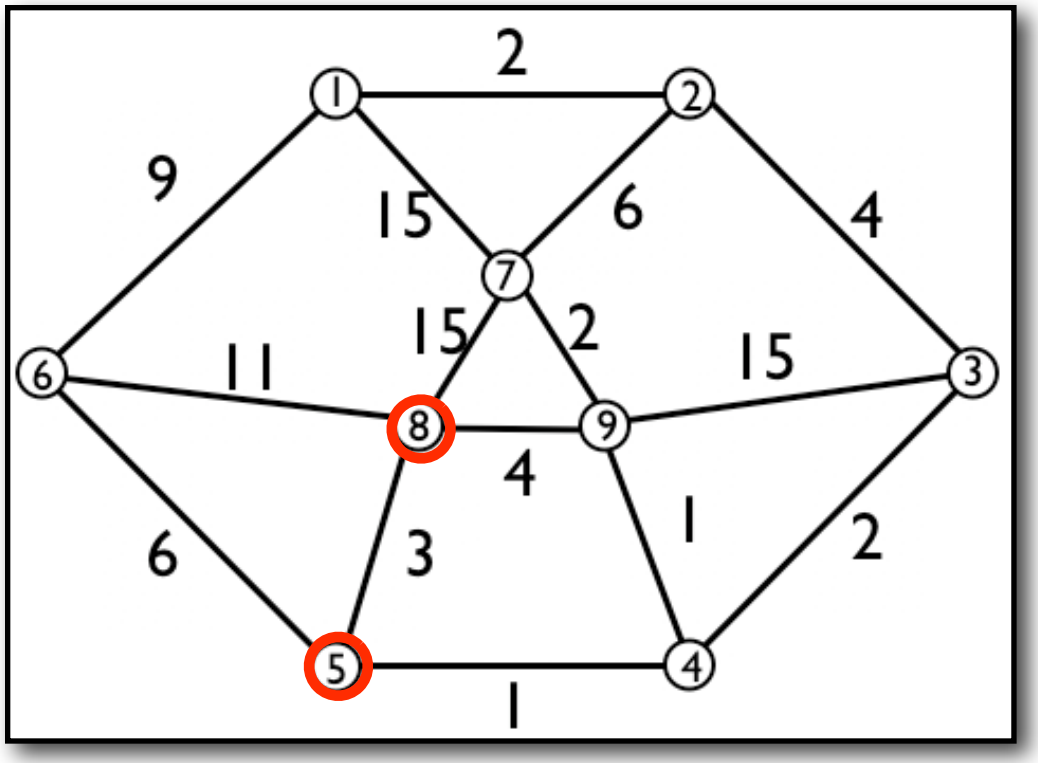
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	
(7,8,2)	(8,23,7)	(9,10,7)	
(4,8,3)	(9,9,4)	(5,9,4)	
(5,9,4)			

Graphenalgorithmen
Kürzeste Wege - Beispiel

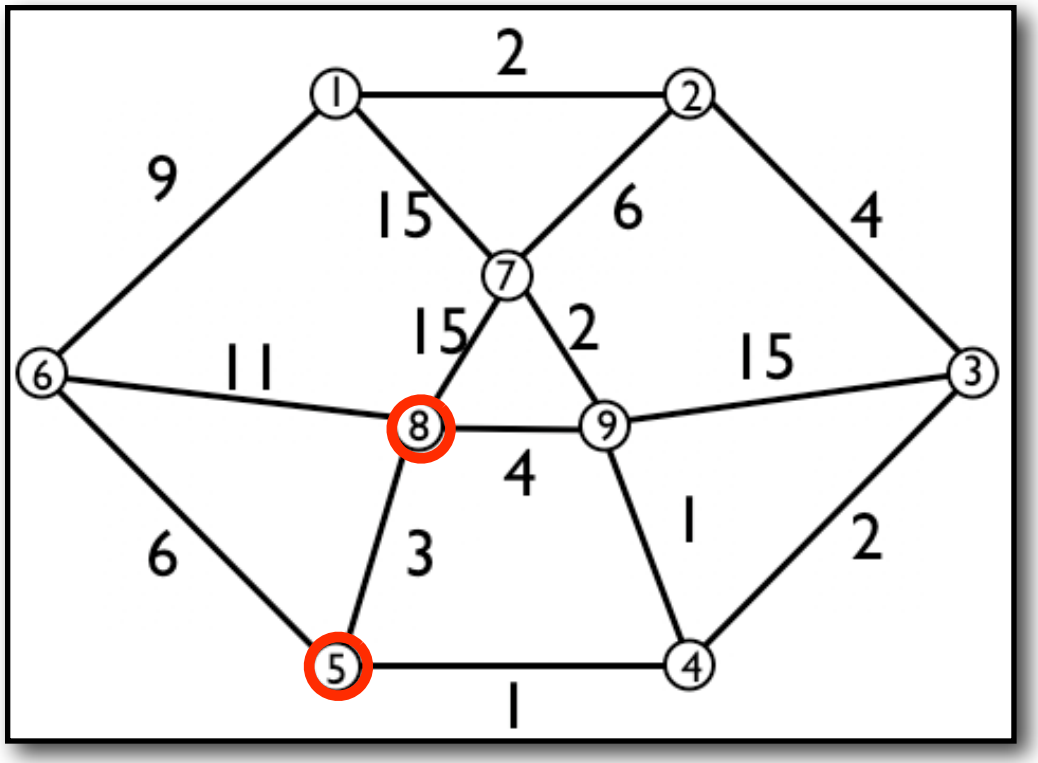
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	
(7,8,2)	(8,23,7)	(9,10,7)	
(4,8,3)	(9,9,4)	(5,9,4)	
(5,9,4)			

Graphenalgorithmen
Kürzeste Wege - Beispiel

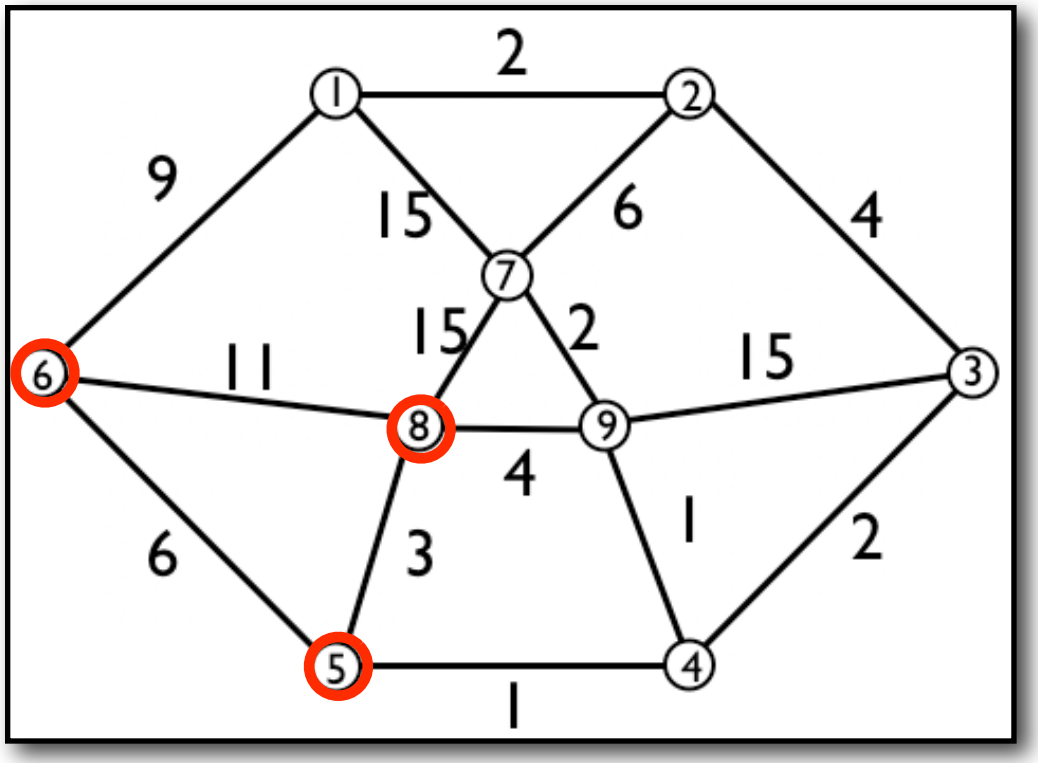
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	
(7,8,2)	(8,23,7)	(9,10,7)	
(4,8,3)	(9,9,4)	(5,9,4)	
(5,9,4)	(8,12,5)		

Graphenalgorithmen
Kürzeste Wege - Beispiel

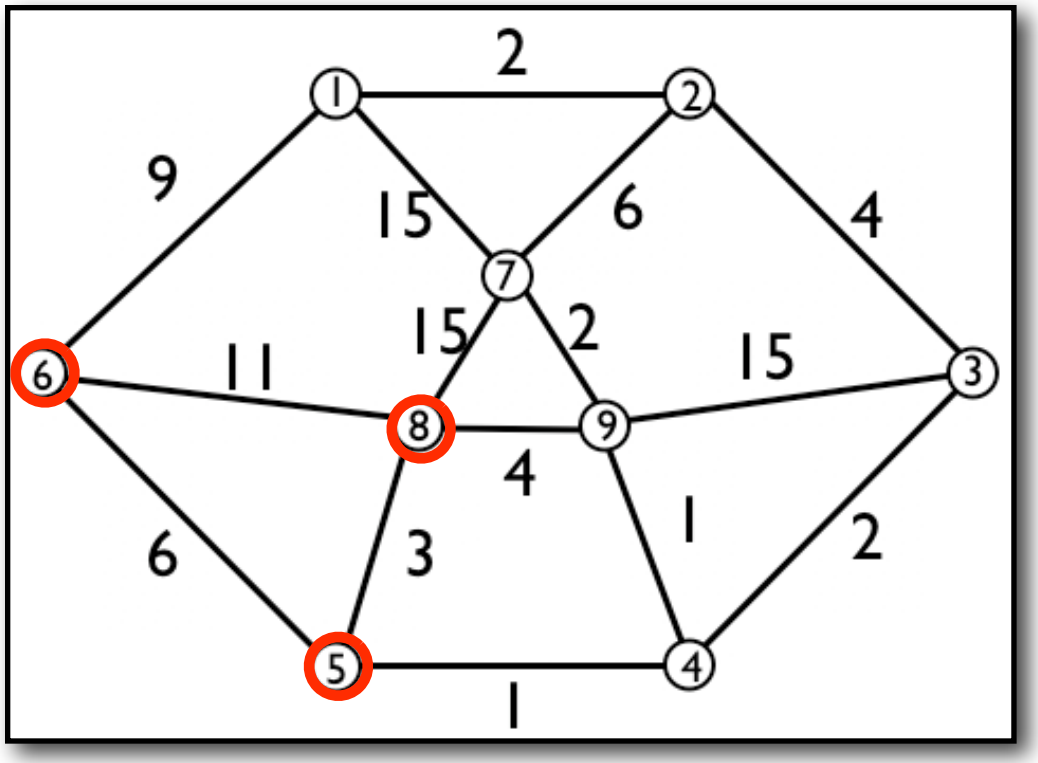
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	
(7,8,2)	(8,23,7)	(9,10,7)	
(4,8,3)	(9,9,4)	(5,9,4)	
(5,9,4)	(8,12,5)		

Graphenalgorithmen
Kürzeste Wege - Beispiel

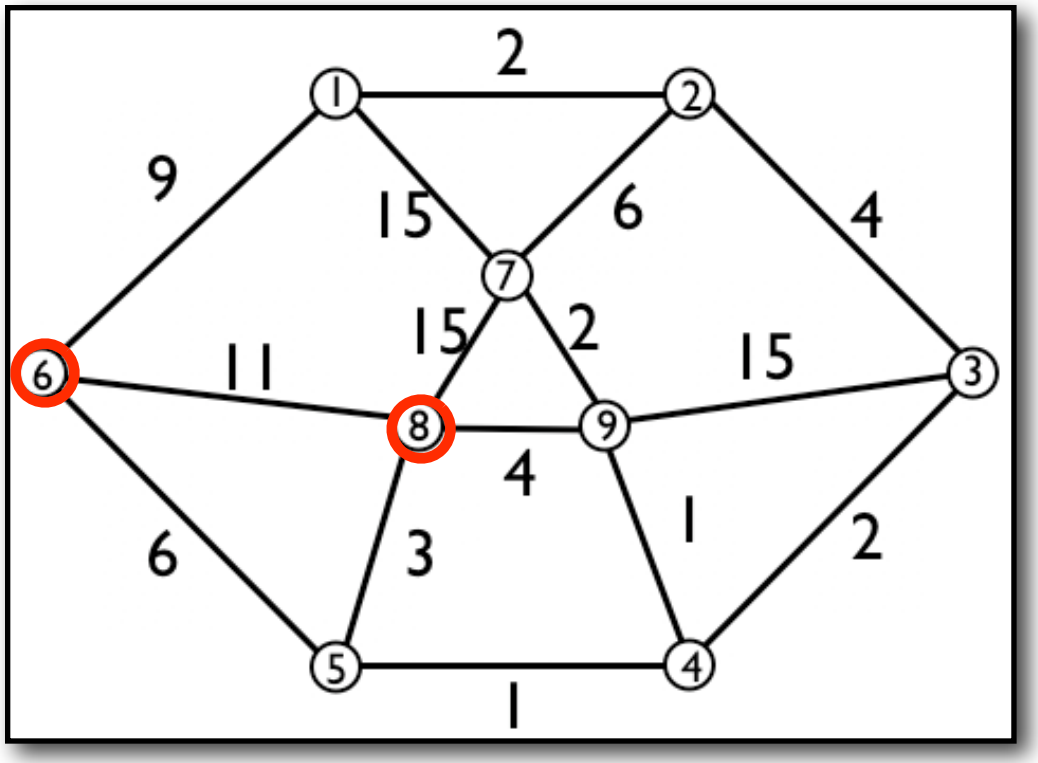
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	
(7,8,2)	(8,23,7)	(9,10,7)	
(4,8,3)	(9,9,4)	(5,9,4)	
(5,9,4)	(8,12,5)	(6,15,5)	

Graphenalgorithmen
Kürzeste Wege - Beispiel

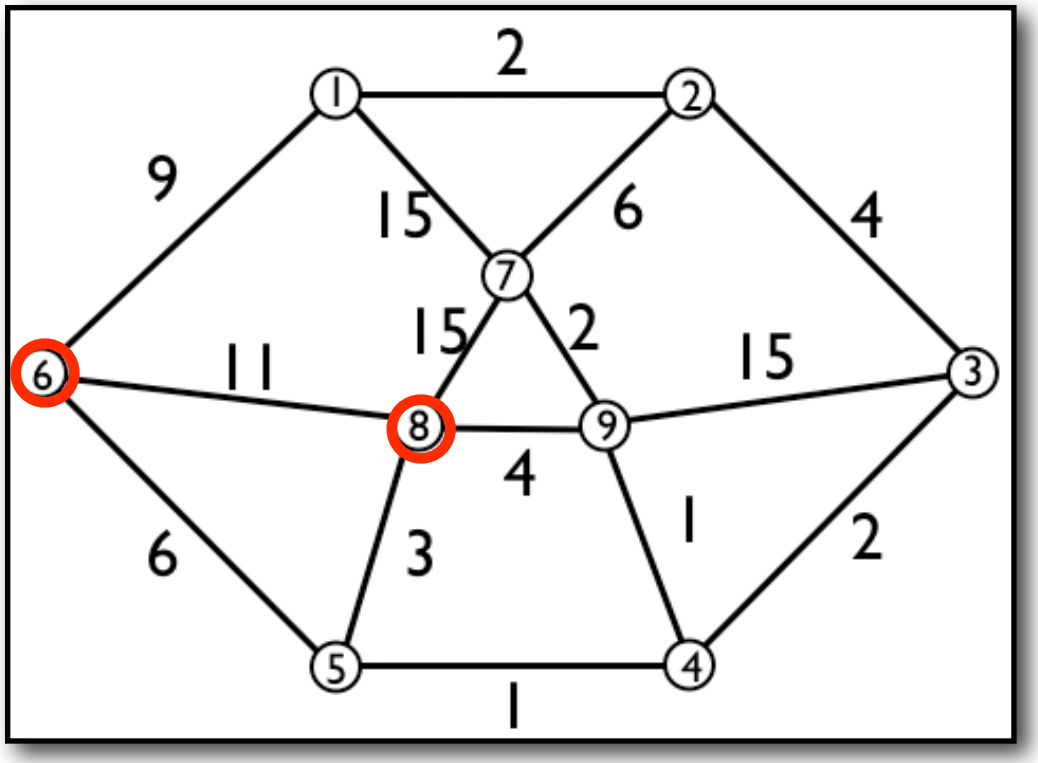
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	
(7,8,2)	(8,23,7)	(9,10,7)	
(4,8,3)	(9,9,4)	(5,9,4)	
(5,9,4)	(8,12,5)	(6,15,5)	

Graphenalgorithmen
Kürzeste Wege - Beispiel

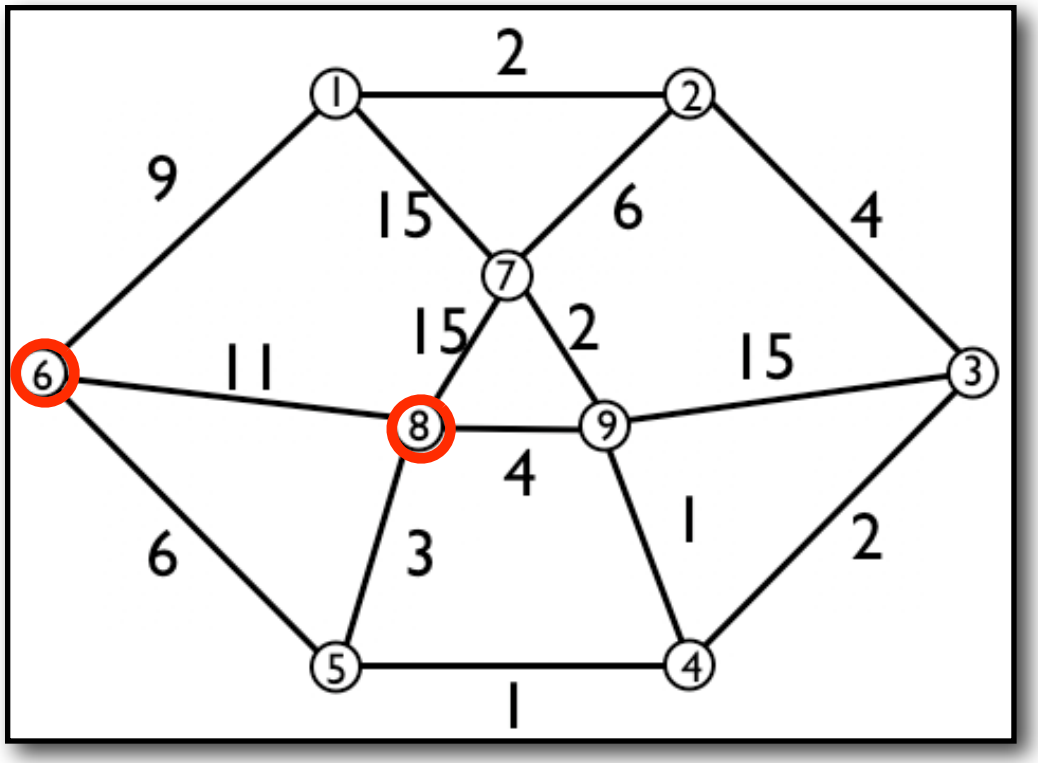
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	
(7,8,2)	(8,23,7)	(9,10,7)	
(4,8,3)	(9,9,4)	(5,9,4)	
(5,9,4)	(8,12,5)	(6,15,5)	

Graphenalgorithmen
Kürzeste Wege - Beispiel

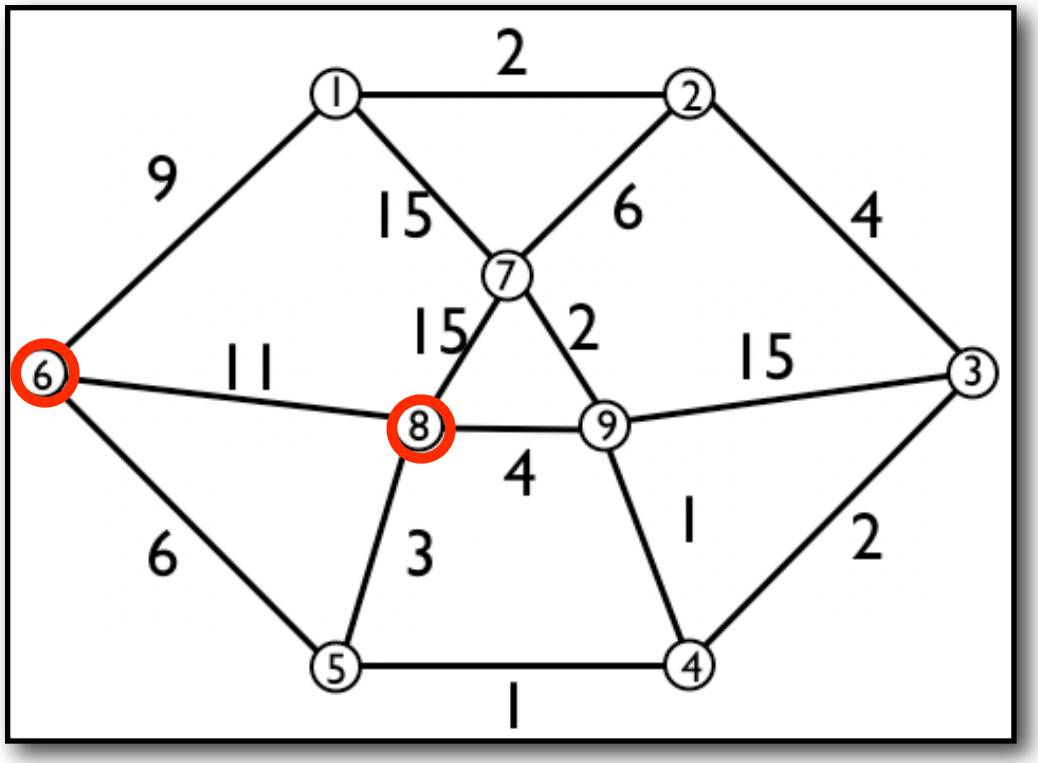
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	
(7,8,2)	(8,23,7)	(9,10,7)	
(4,8,3)	(9,9,4)	(5,9,4)	
(5,9,4)	(8,12,5)	(6,15,5)	
(6,9,1)			

Graphenalgorithmen
Kürzeste Wege - Beispiel

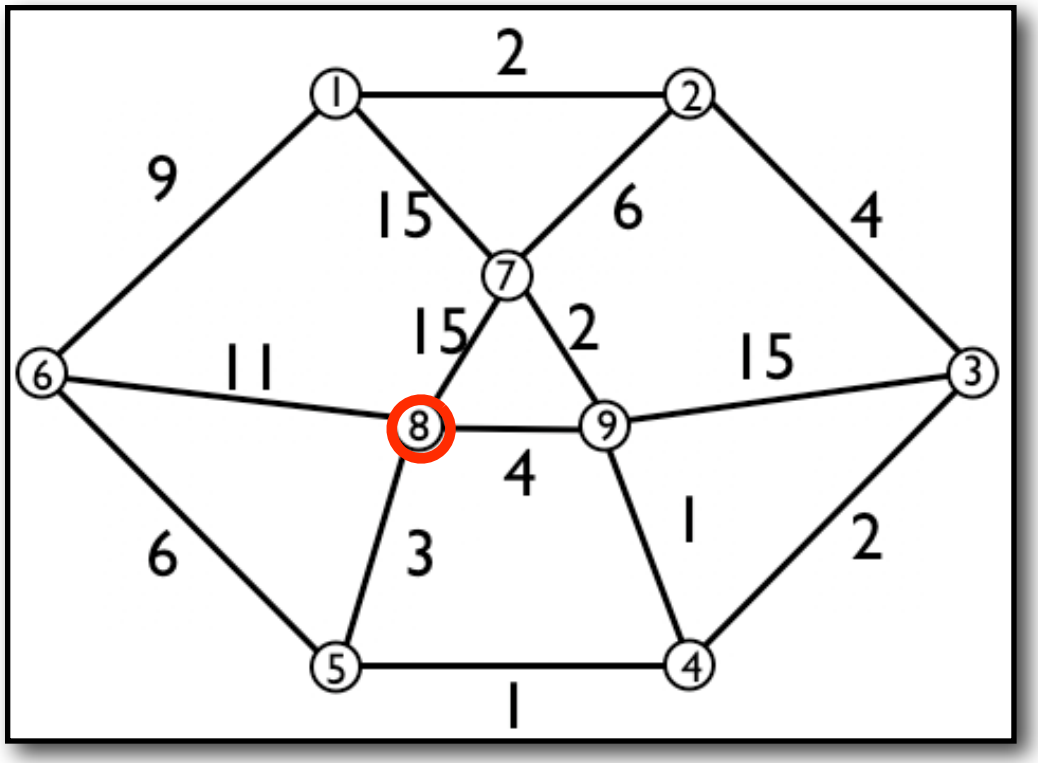
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	
(7,8,2)	(8,23,7)	(9,10,7)	
(4,8,3)	(9,9,4)	(5,9,4)	
(5,9,4)	(8,12,5)	(6,15,5)	
(6,9,1)	(8,20,6)		

Graphenalgorithmen
Kürzeste Wege - Beispiel

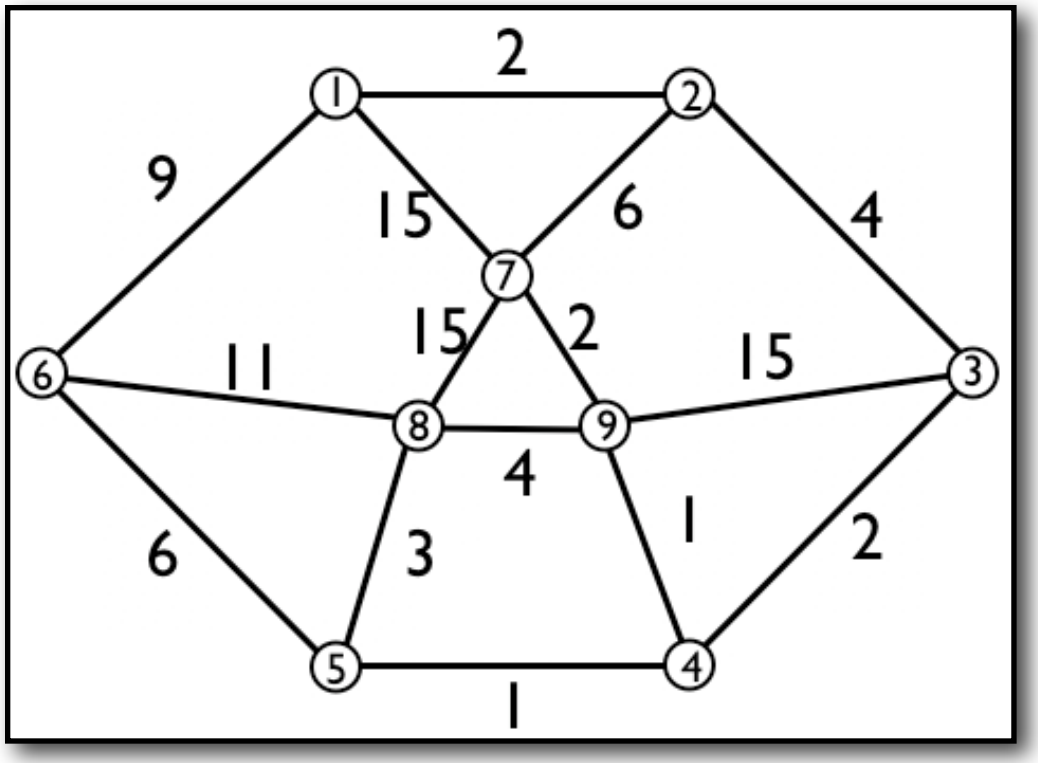
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	
(7,8,2)	(8,23,7)	(9,10,7)	
(4,8,3)	(9,9,4)	(5,9,4)	
(5,9,4)	(8,12,5)	(6,15,5)	
(6,9,1)	(8,20,6)		

Graphenalgorithmen
Kürzeste Wege - Beispiel

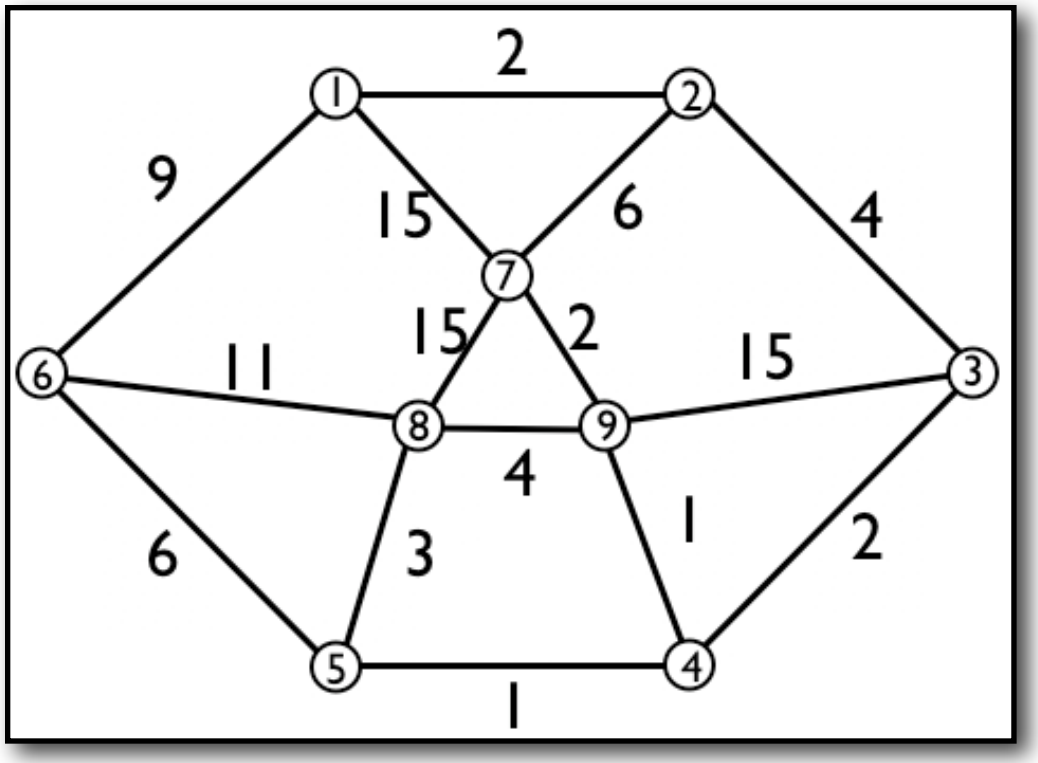
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	
(7,8,2)	(8,23,7)	(9,10,7)	
(4,8,3)	(9,9,4)	(5,9,4)	
(5,9,4)	(8,12,5)	(6,15,5)	
(6,9,1)	(8,20,6)		

Graphenalgorithmen
Kürzeste Wege - Beispiel

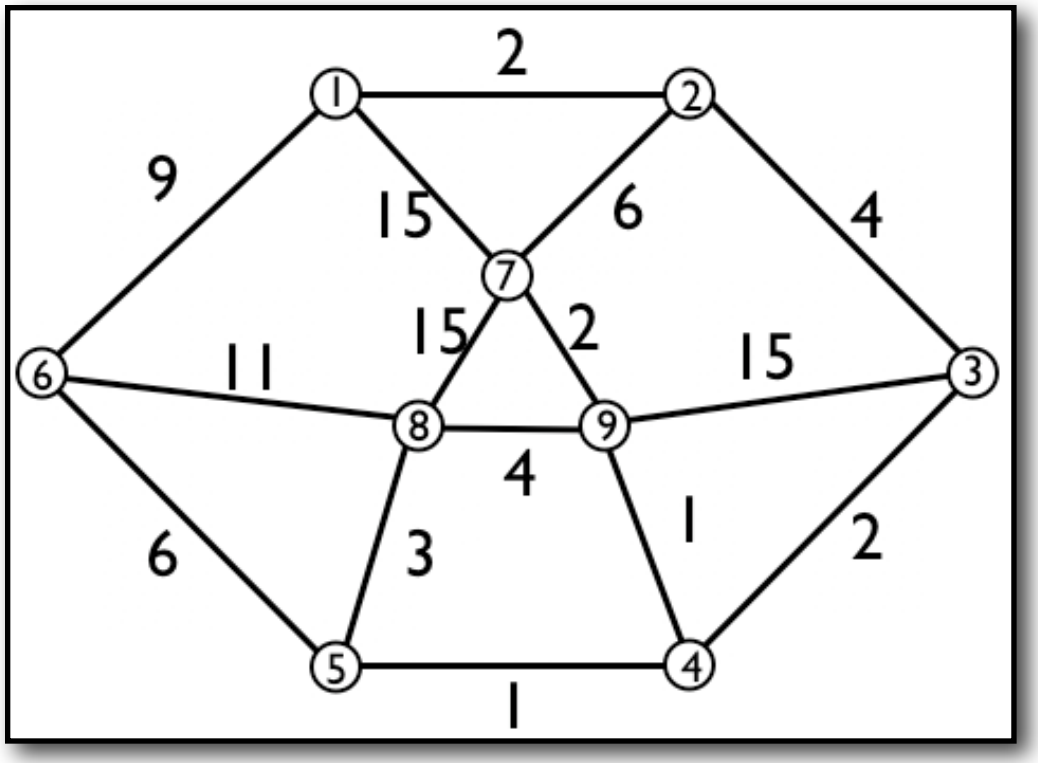
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	
(7,8,2)	(8,23,7)	(9,10,7)	
(4,8,3)	(9,9,4)	(5,9,4)	
(5,9,4)	(8,12,5)	(6,15,5)	
(6,9,1)	(8,20,6)		

Graphenalgorithmen
Kürzeste Wege - Beispiel

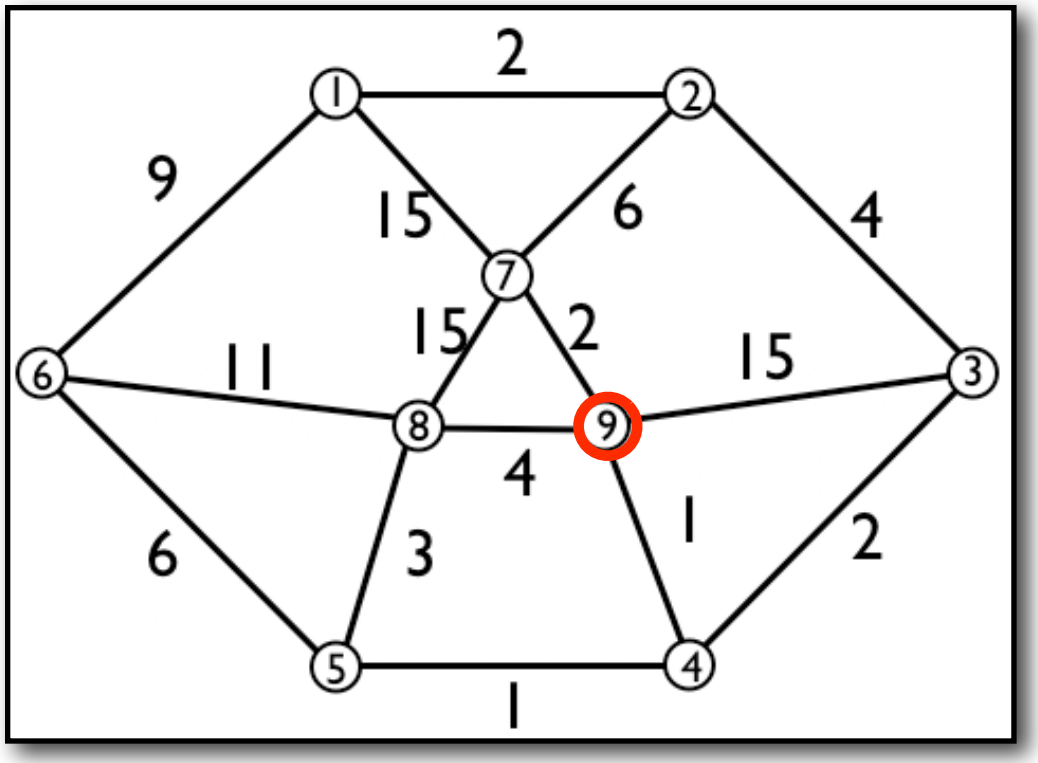
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	
(7,8,2)	(8,23,7)	(9,10,7)	
(4,8,3)	(9,9,4)	(5,9,4)	
(5,9,4)	(8,12,5)	(6,15,5)	
(6,9,1)	(8,20,6)		
(9,9,4)			

Graphenalgorithmen
Kürzeste Wege - Beispiel

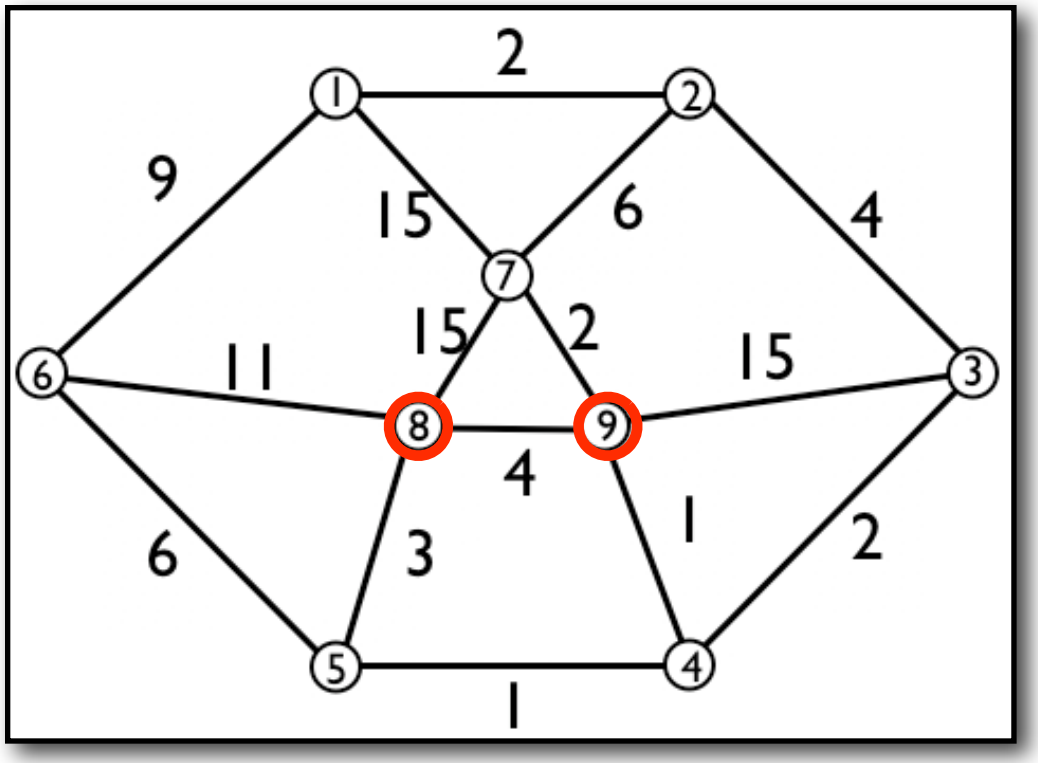
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	
(7,8,2)	(8,23,7)	(9,10,7)	
(4,8,3)	(9,9,4)	(5,9,4)	
(5,9,4)	(8,12,5)	(6,15,5)	
(6,9,1)	(8,20,6)		
(9,9,4)			

Graphenalgorithmen
Kürzeste Wege - Beispiel

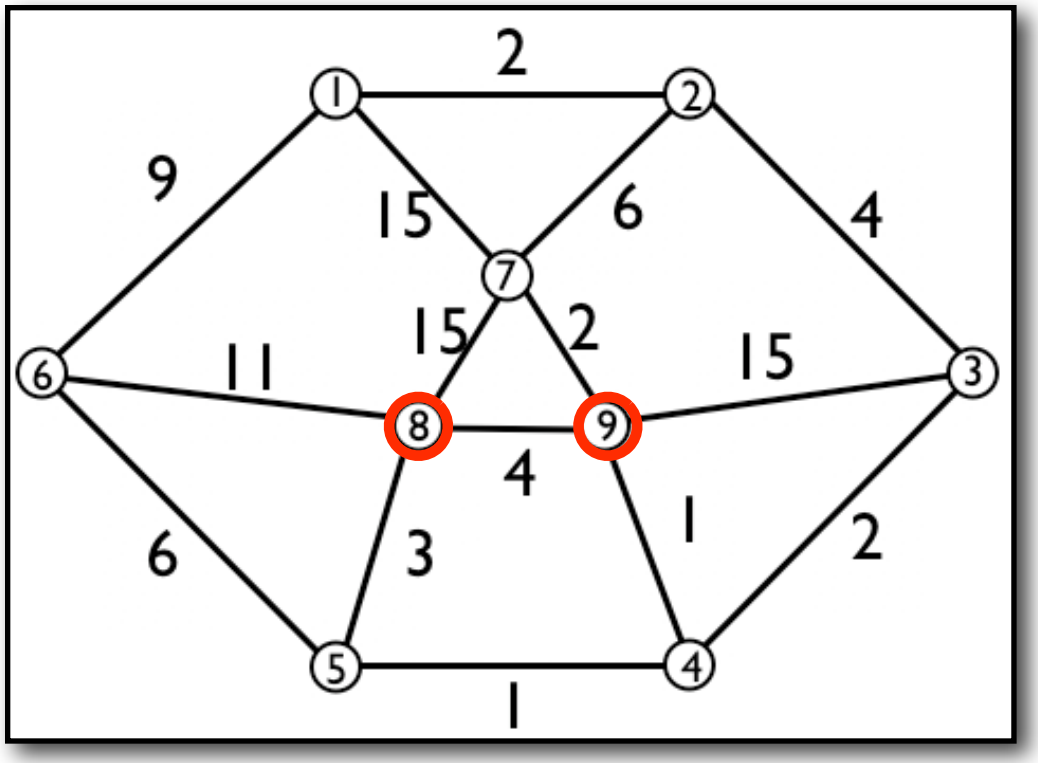
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	
(7,8,2)	(8,23,7)	(9,10,7)	
(4,8,3)	(9,9,4)	(5,9,4)	
(5,9,4)	(8,12,5)	(6,15,5)	
(6,9,1)	(8,20,6)		
(9,9,4)			

Graphenalgorithmen
Kürzeste Wege - Beispiel

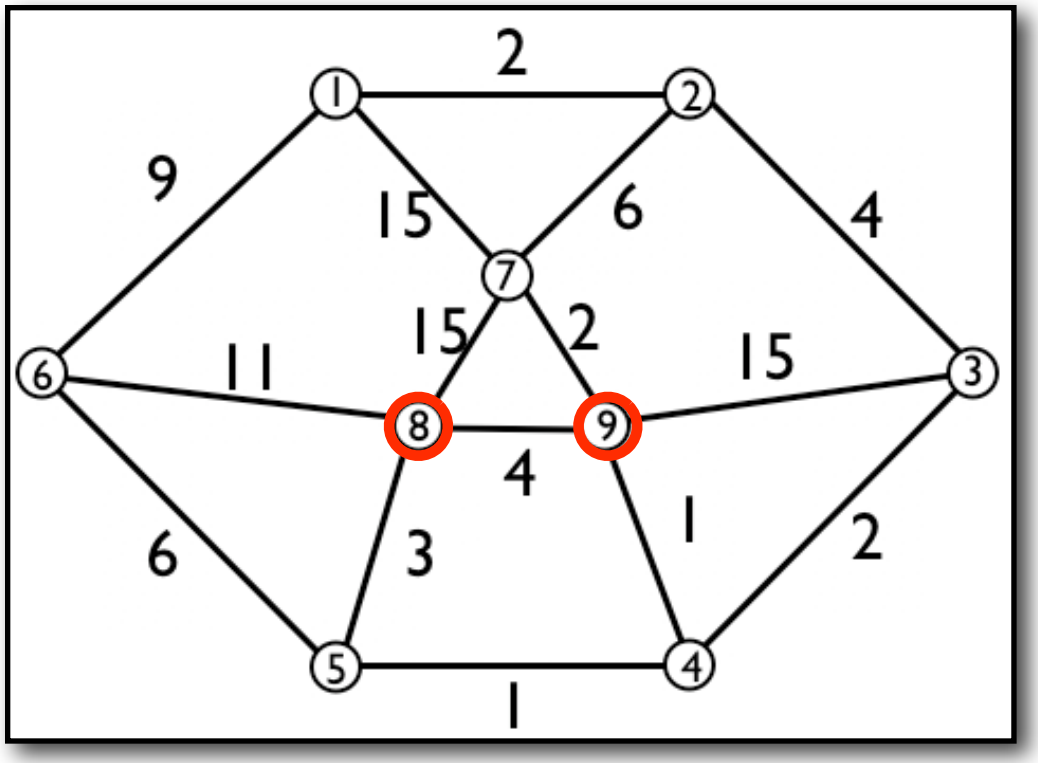
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	
(7,8,2)	(8,23,7)	(9,10,7)	
(4,8,3)	(9,9,4)	(5,9,4)	
(5,9,4)	(8,12,5)	(6,15,5)	
(6,9,1)	(8,20,6)		
(9,9,4)	(8,13,9)		

Graphenalgorithmen
Kürzeste Wege - Beispiel

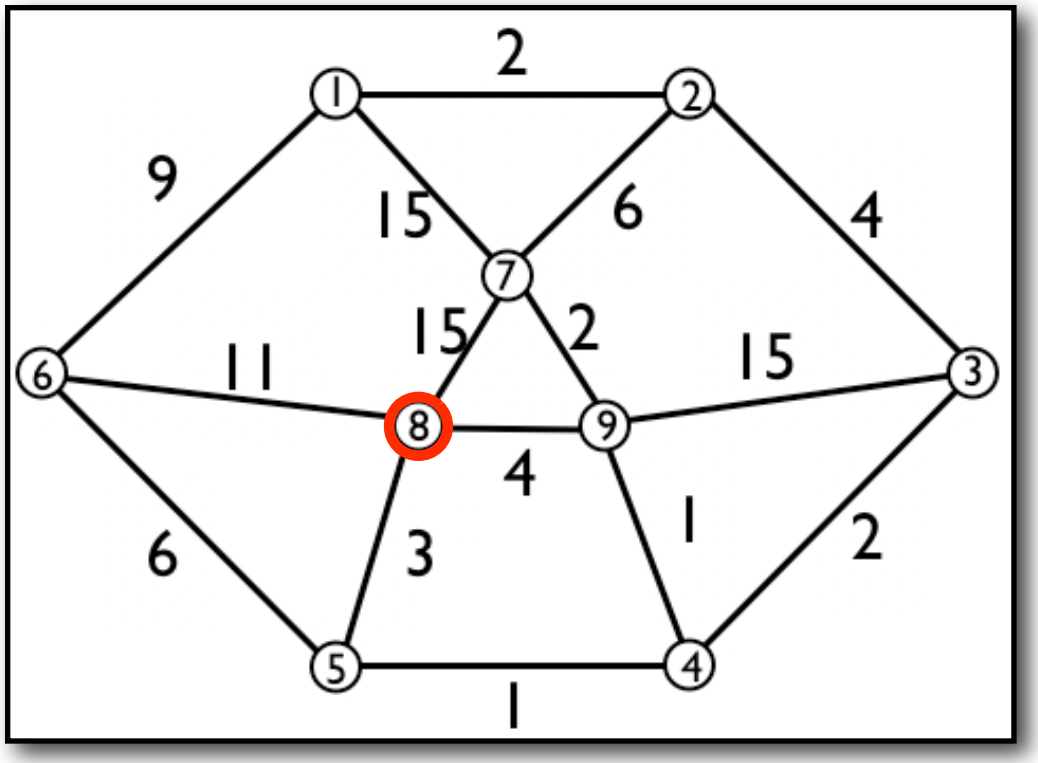
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	
(7,8,2)	(8,23,7)	(9,10,7)	
(4,8,3)	(9,9,4)	(5,9,4)	
(5,9,4)	(8,12,5)	(6,15,5)	
(6,9,1)	(8,20,6)		
(9,9,4)	(8,13,9)		

Graphenalgorithmen
Kürzeste Wege - Beispiel

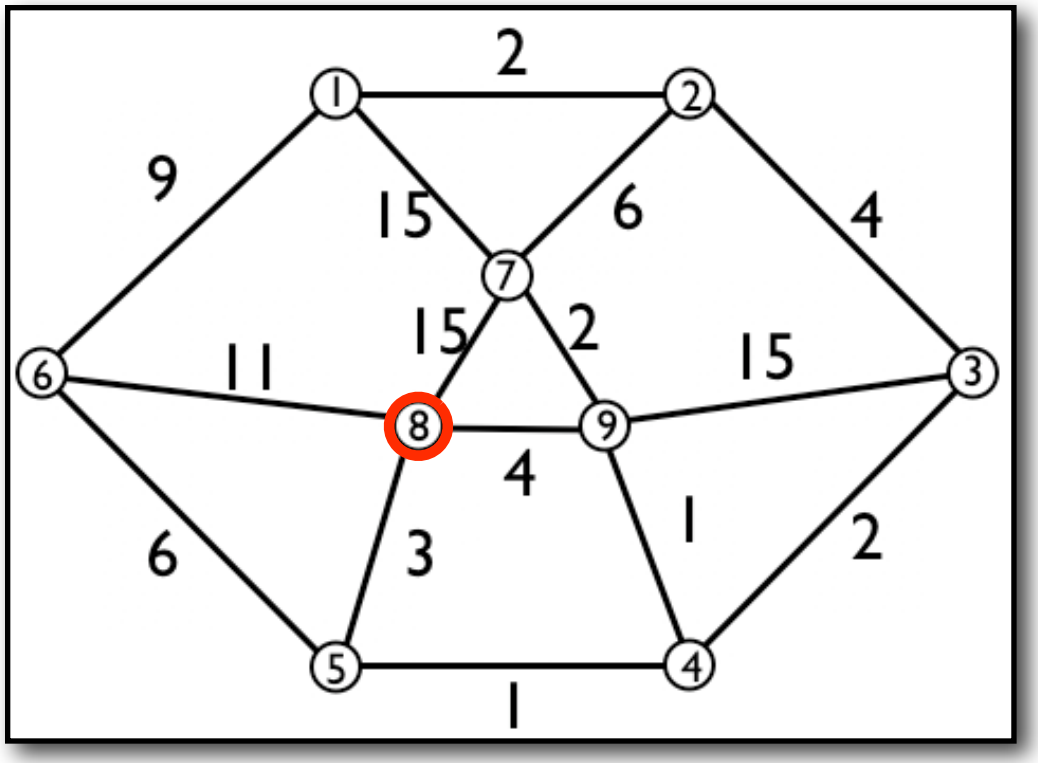
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	
(7,8,2)	(8,23,7)	(9,10,7)	
(4,8,3)	(9,9,4)	(5,9,4)	
(5,9,4)	(8,12,5)	(6,15,5)	
(6,9,1)	(8,20,6)		
(9,9,4)	(8,13,9)		

Graphenalgorithmen
Kürzeste Wege - Beispiel

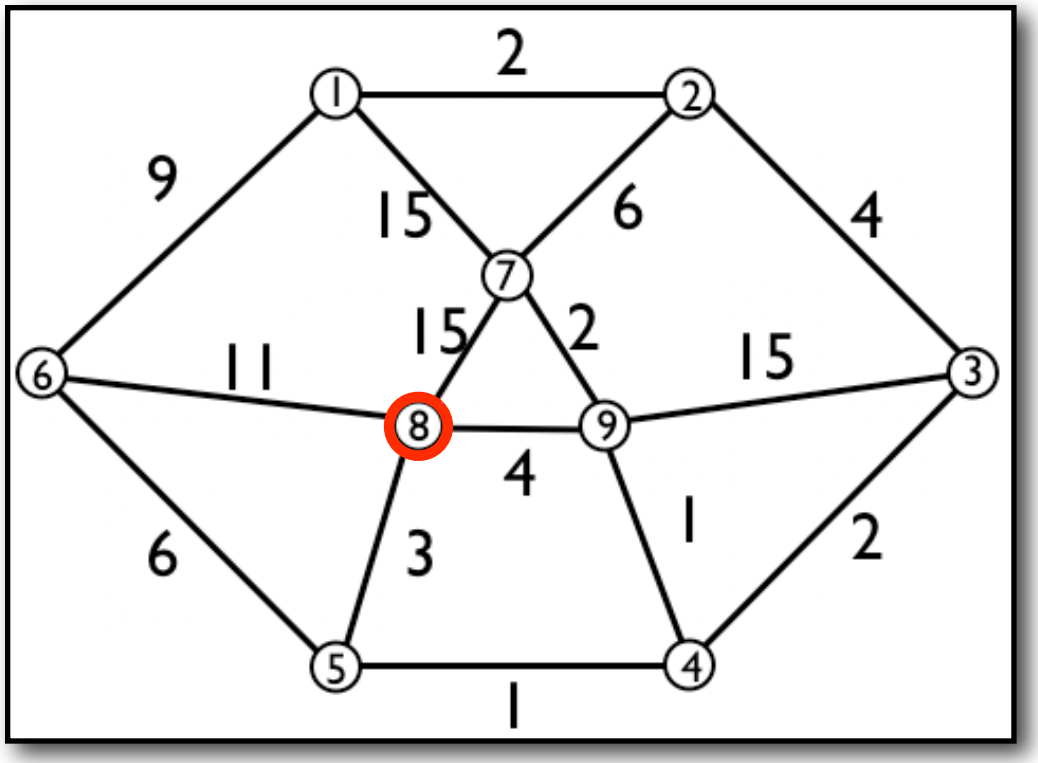
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	
(7,8,2)	(8,23,7)	(9,10,7)	
(4,8,3)	(9,9,4)	(5,9,4)	
(5,9,4)	(8,12,5)	(6,15,5)	
(6,9,1)	(8,20,6)		
(9,9,4)	(8,13,9)		
(8,12,5)			

Graphenalgorithmen
Kürzeste Wege - Beispiel

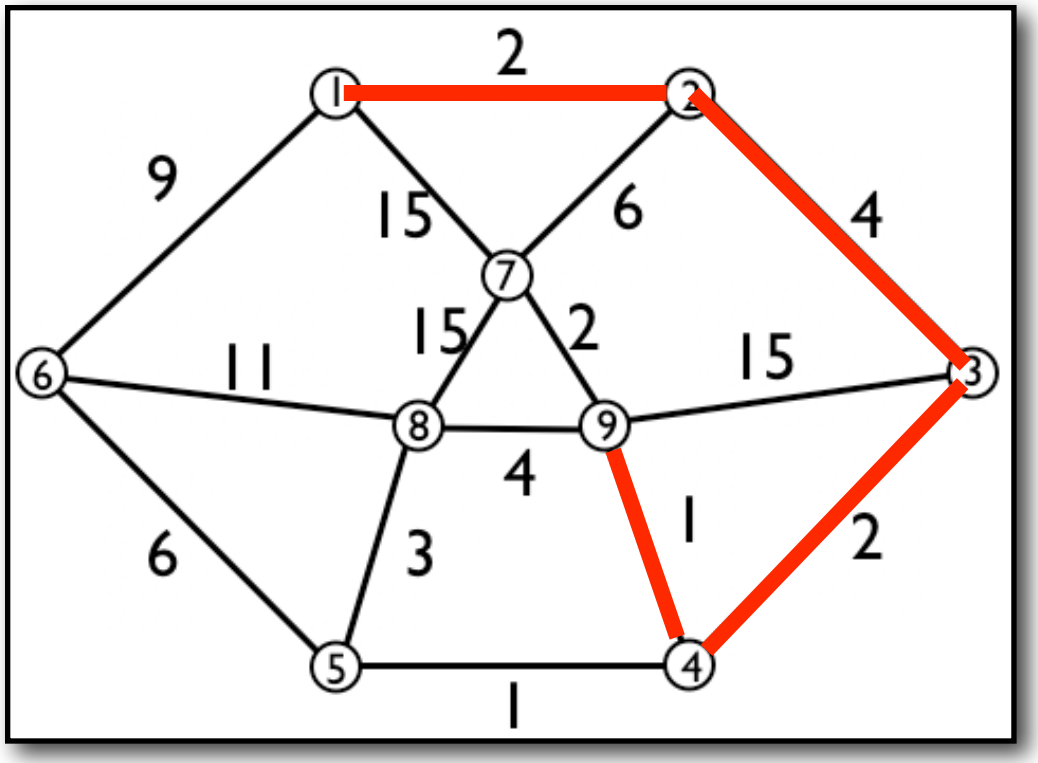
Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt	Rand R		
(1,0,1)	(2,2,1)	(7,15,1)	(6,9,1)
(2,2,1)	(7,8,2)	(3,6,2)	
(3,6,2)	(9,21,3)	(4,8,3)	
(7,8,2)	(8,23,7)	(9,10,7)	
(4,8,3)	(9,9,4)	(5,9,4)	
(5,9,4)	(8,12,5)	(6,15,5)	
(6,9,1)	(8,20,6)		
(9,9,4)	(8,13,9)		
(8,12,5)	Fertig		

Graphenalgorithmen
Kürzeste Wege - Beispiel

Darstellung:
(Knotennr, Distanz, Vorgänger)

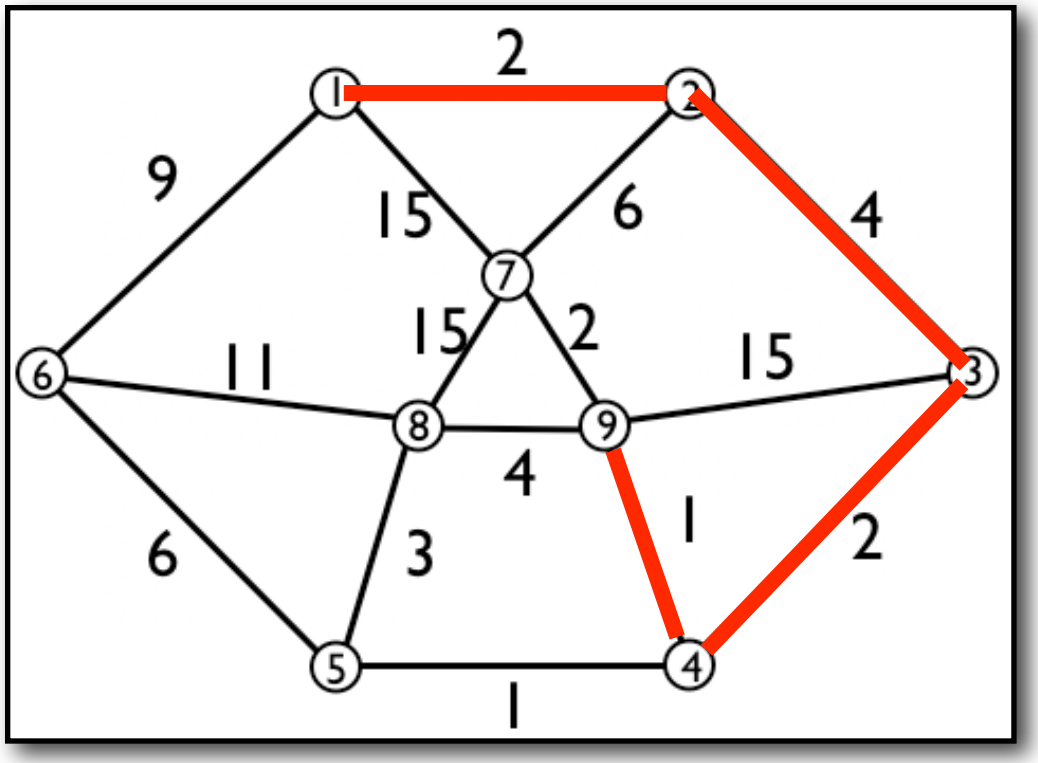


gewählt

(1,0,1)
(2,2,1)
(3,6,2)
(7,8,2)
(4,8,3)
(5,9,4)
(6,9,1)
(9,9,4)
(8,12,5)

Graphenalgorithmen
Kürzeste Wege - Beispiel

Darstellung:
(Knotennr, Distanz, Vorgänger)

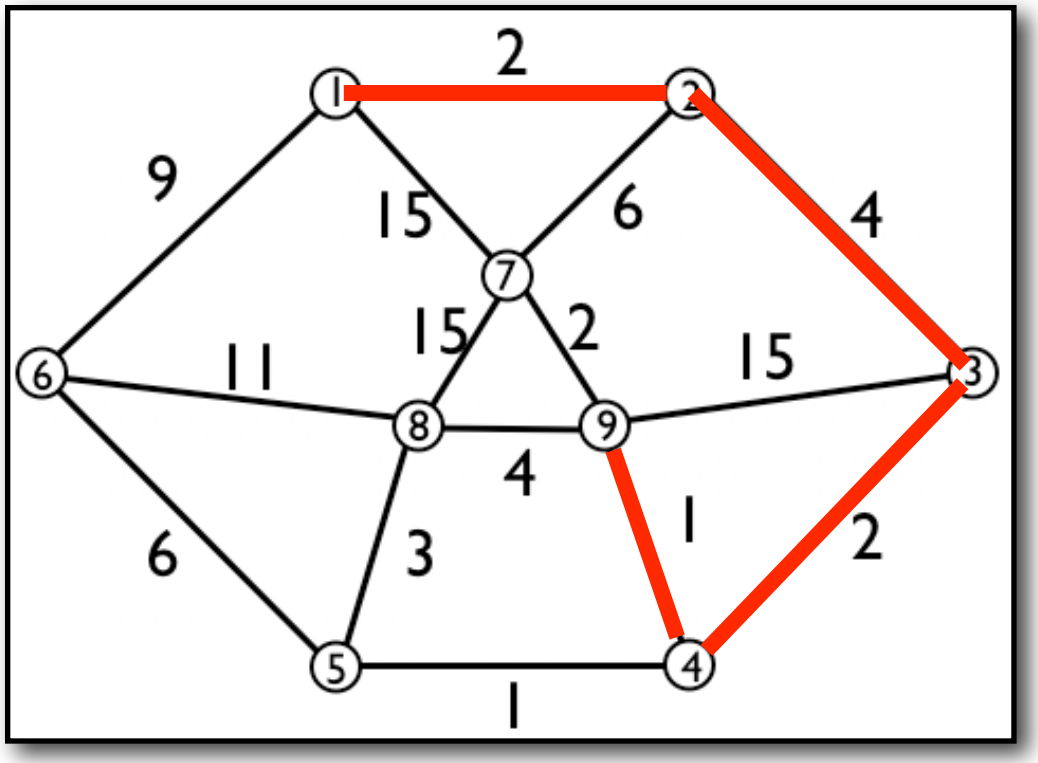


gewählt

(1,0,1)
(2,2,1)
(3,6,2)
(7,8,2)
(4,8,3)
(5,9,4)
(6,9,1)
(9,9,4)
(8,12,5)

Graphenalgorithmen
Kürzeste Wege - Beispiel

Darstellung:
(Knotennr, Distanz, Vorgänger)

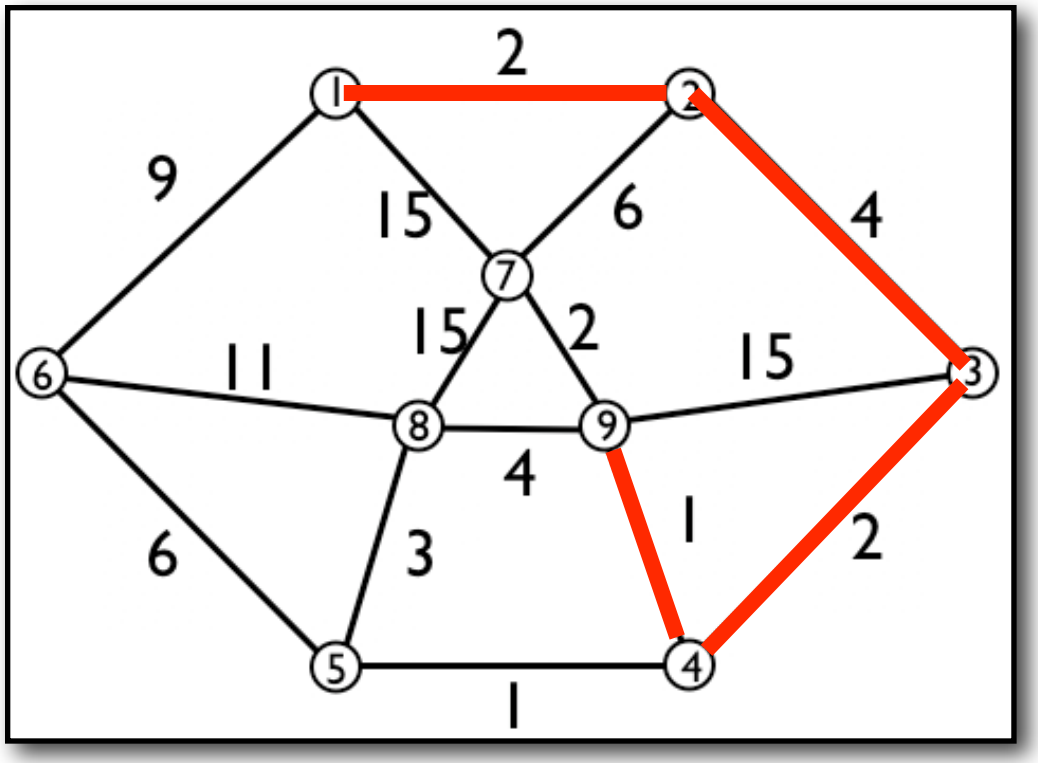


gewählt

(1,0,1)
(2,2,1)
(3,6,2)
(7,8,2)
(4,8,3)
(5,9,4)
(6,9,1)
(9,9,4)
(8,12,5)

Graphenalgorithmen
Kürzeste Wege - Beispiel

Darstellung:
(Knotennr, Distanz, Vorgänger)

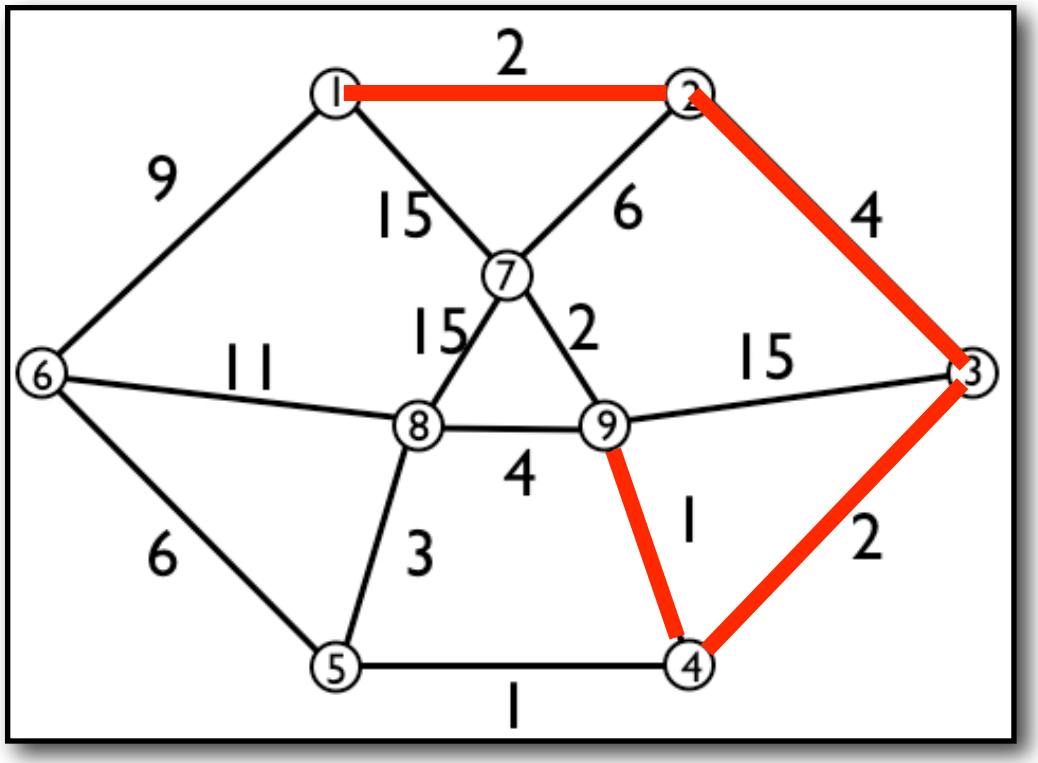


gewählt

(1,0,1)
(2,2,1)
(3,6,2)
(7,8,2)
(4,8,3)
(5,9,4)
(6,9,1)
(9,9,4)
(8,12,5)

Graphenalgorithmen
Kürzeste Wege - Beispiel

Darstellung:
(Knotennr, Distanz, Vorgänger)

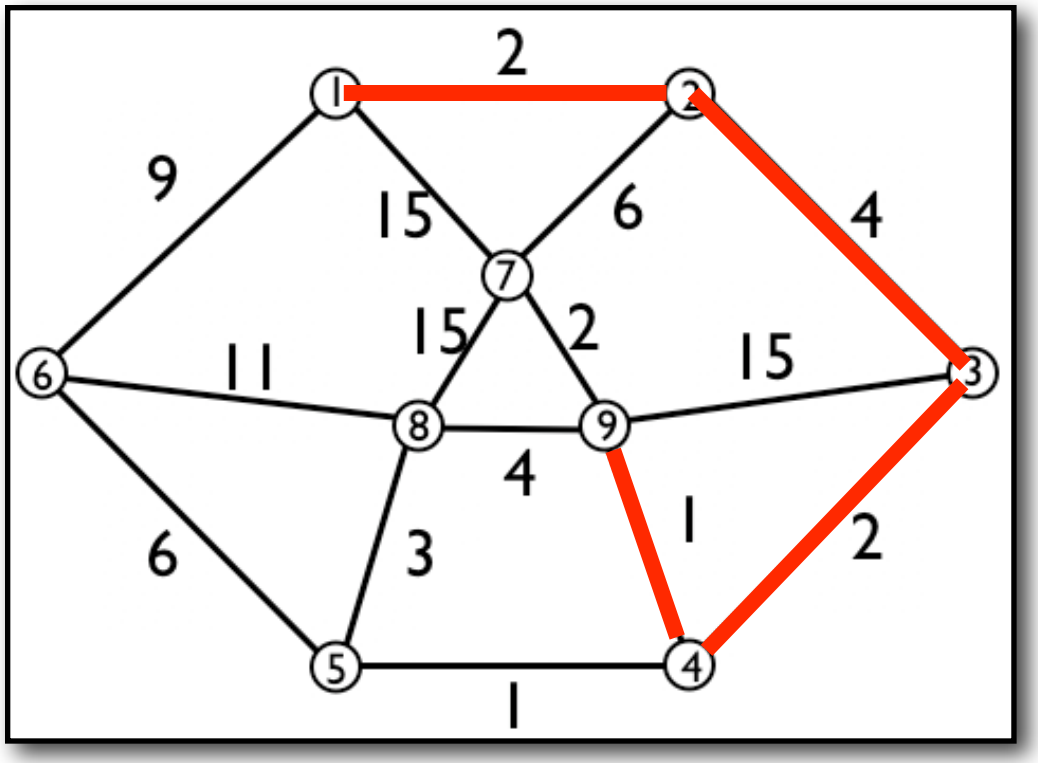


gewählt

(1,0,1)
(2,2,1)
(3,6,2)
(7,8,2)
(4,8,3)
(5,9,4)
(6,9,1)
(9,9,4)
(8,12,5)

Graphenalgorithmen
Kürzeste Wege - Beispiel

Darstellung:
(Knotennr, Distanz, Vorgänger)

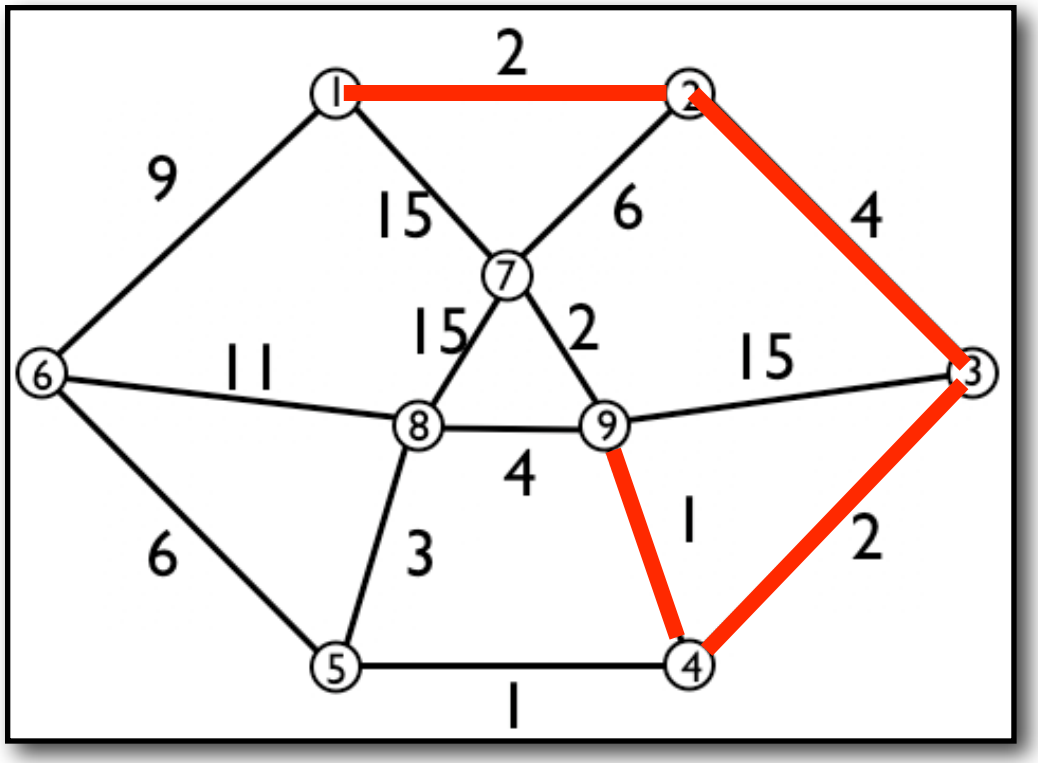


gewählt

(1,0,1)
(2,2,1)
(3,6,2)
(7,8,2)
(4,8,3)
(5,9,4)
(6,9,1)
(9,9,4)
(8,12,5)

Graphenalgorithmen
Kürzeste Wege - Beispiel

Darstellung:
(Knotennr, Distanz, Vorgänger)

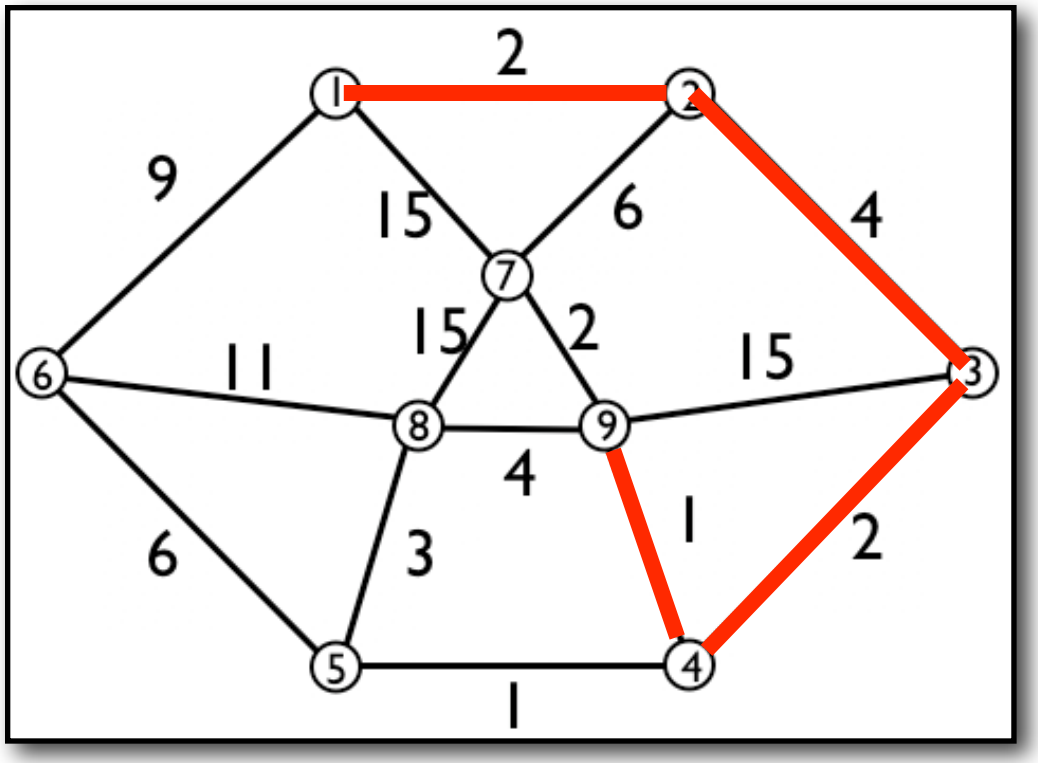


gewählt

(1,0,1)
(2,2,1)
(3,6,2)
(7,8,2)
(4,8,3)
(5,9,4)
(6,9,1)
(9,9,4)
(8,12,5)

Graphenalgorithmen
Kürzeste Wege - Beispiel

Darstellung:
(Knotennr, Distanz, Vorgänger)

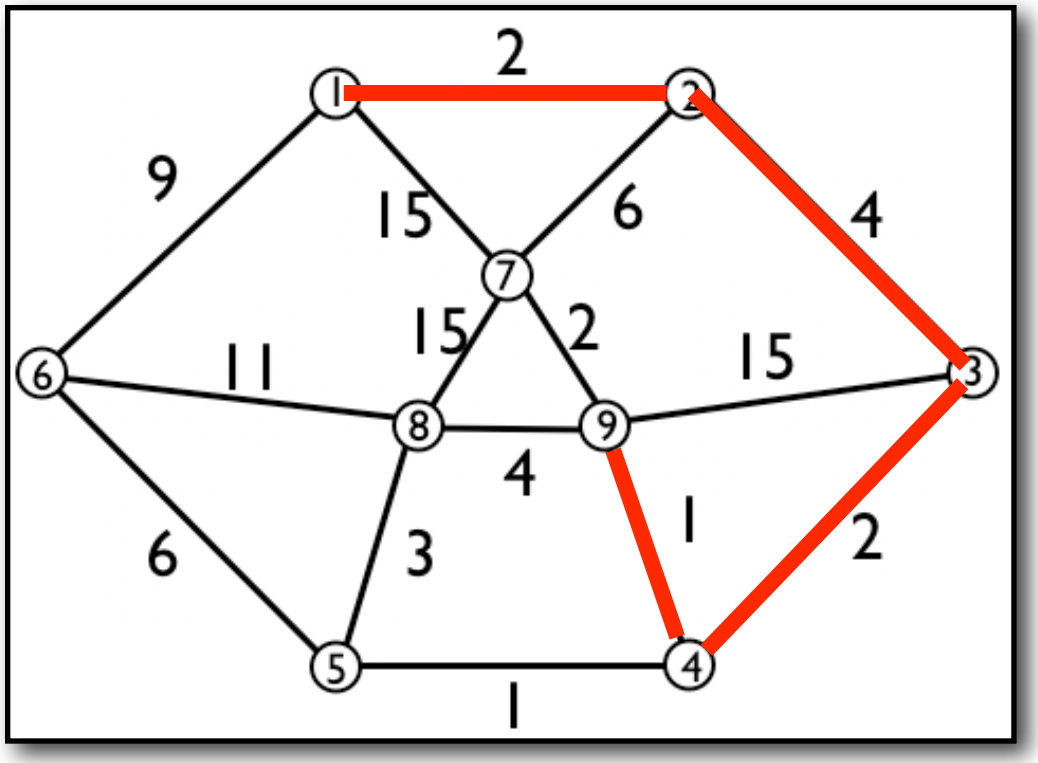


gewählt

(1,0,1)
(2,2,1)
(3,6,2)
(7,8,2)
(4,8,3)
(5,9,4)
(6,9,1)
(9,9,4)
(8,12,5)

Graphenalgorithmen
Kürzeste Wege - Beispiel

Darstellung:
(Knotennr, Distanz, Vorgänger)

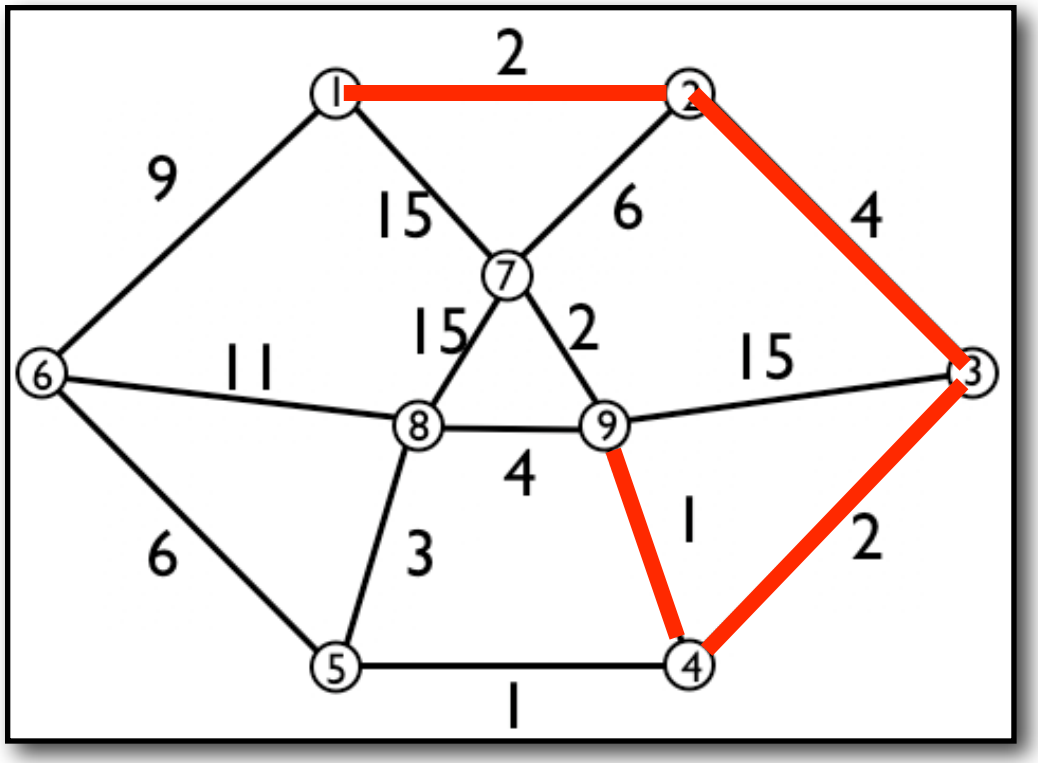


gewählt

(1,0,1)
(2,2,1)
(3,6,2)
(7,8,2)
(4,8,3)
(5,9,4)
(6,9,1)
(9,9,4)
(8,12,5)

Graphenalgorithmen
Kürzeste Wege - Beispiel

Darstellung:
(Knotennr, Distanz, Vorgänger)

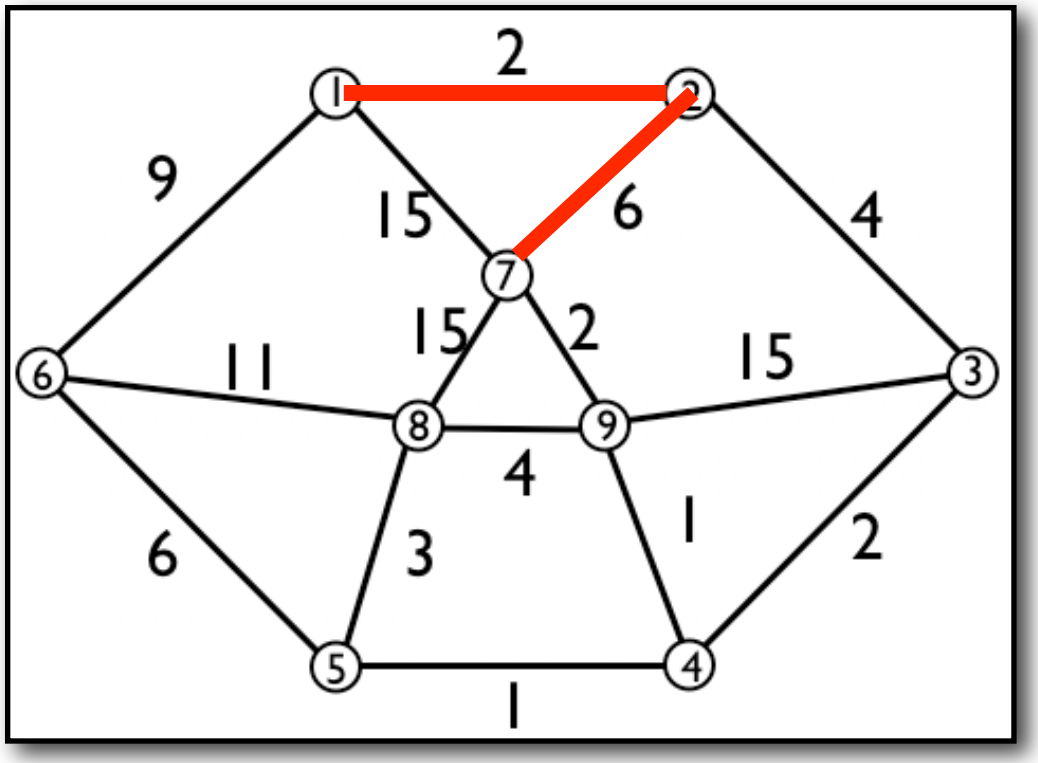


gewählt

(1,0,1)
(2,2,1)
(3,6,2)
(7,8,2)
(4,8,3)
(5,9,4)
(6,9,1)
(9,9,4)
(8,12,5)

Graphenalgorithmen
Kürzeste Wege - Beispiel

Darstellung:
(Knotennr, Distanz, Vorgänger)

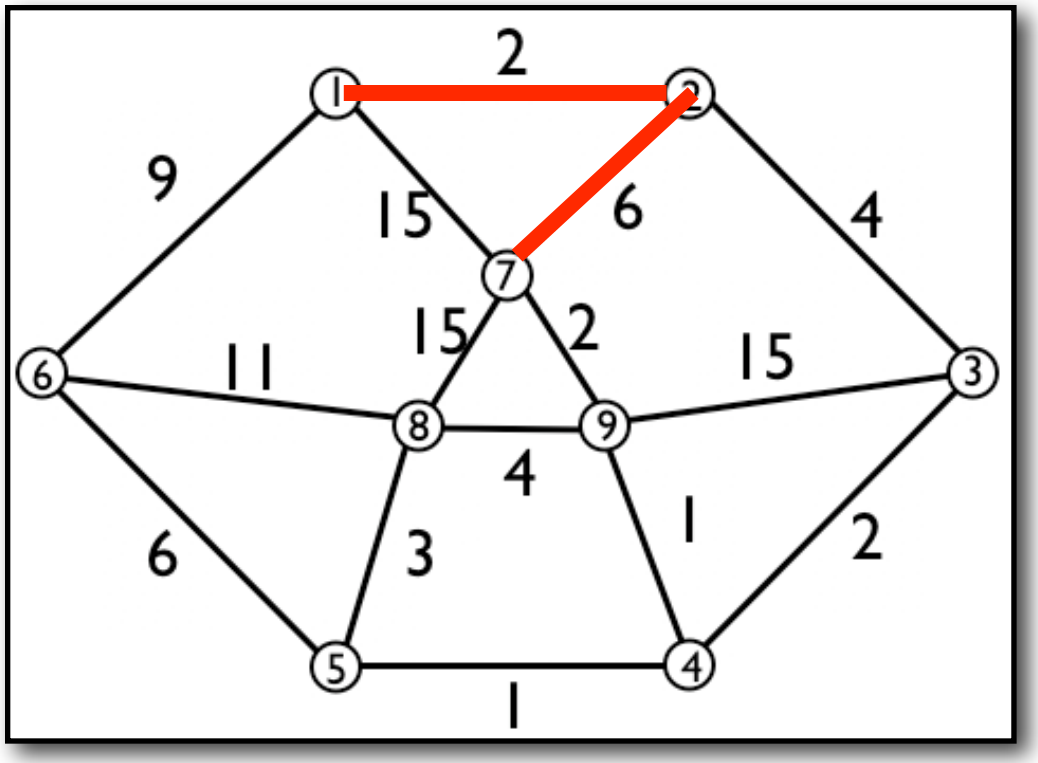


gewählt

(1,0,1)
(2,2,1)
(3,6,2)
(7,8,2)
(4,8,3)
(5,9,4)
(6,9,1)
(9,9,4)
(8,12,5)

Graphenalgorithmen
Kürzeste Wege - Beispiel

Darstellung:
(Knotennr, Distanz, Vorgänger)

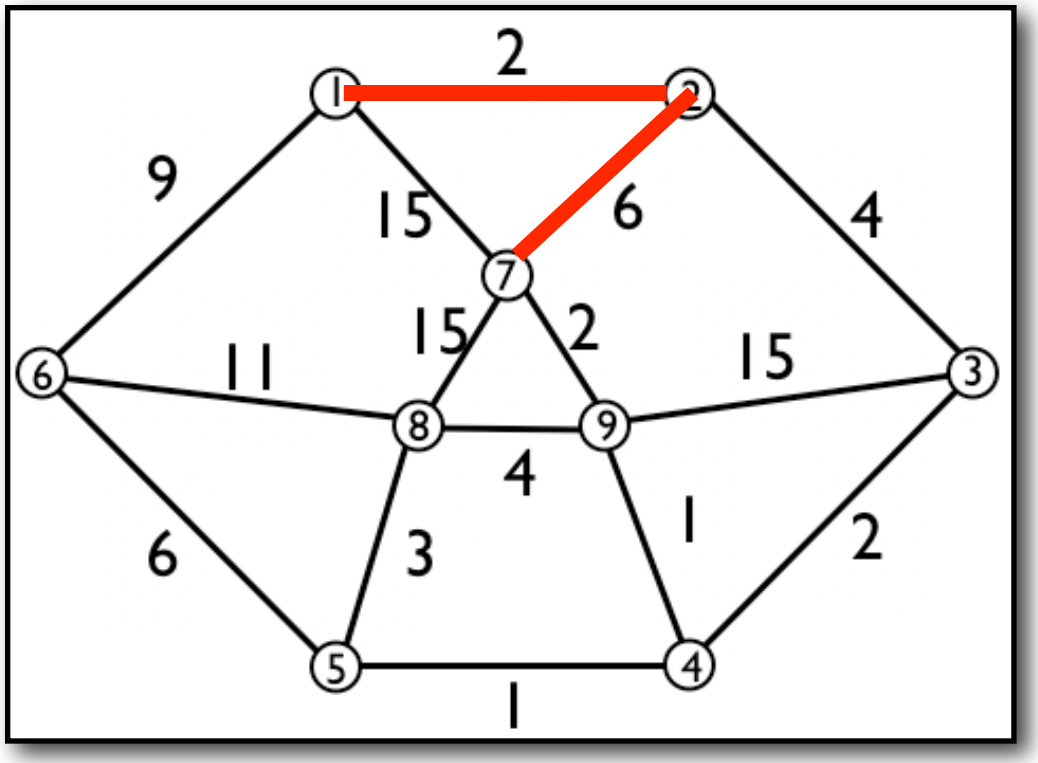


gewählt

(1,0,1)
(2,2,1)
(3,6,2)
(7,8,2)
(4,8,3)
(5,9,4)
(6,9,1)
(9,9,4)
(8,12,5)

Graphenalgorithmen
Kürzeste Wege - Beispiel

Darstellung:
(Knotennr, Distanz, Vorgänger)

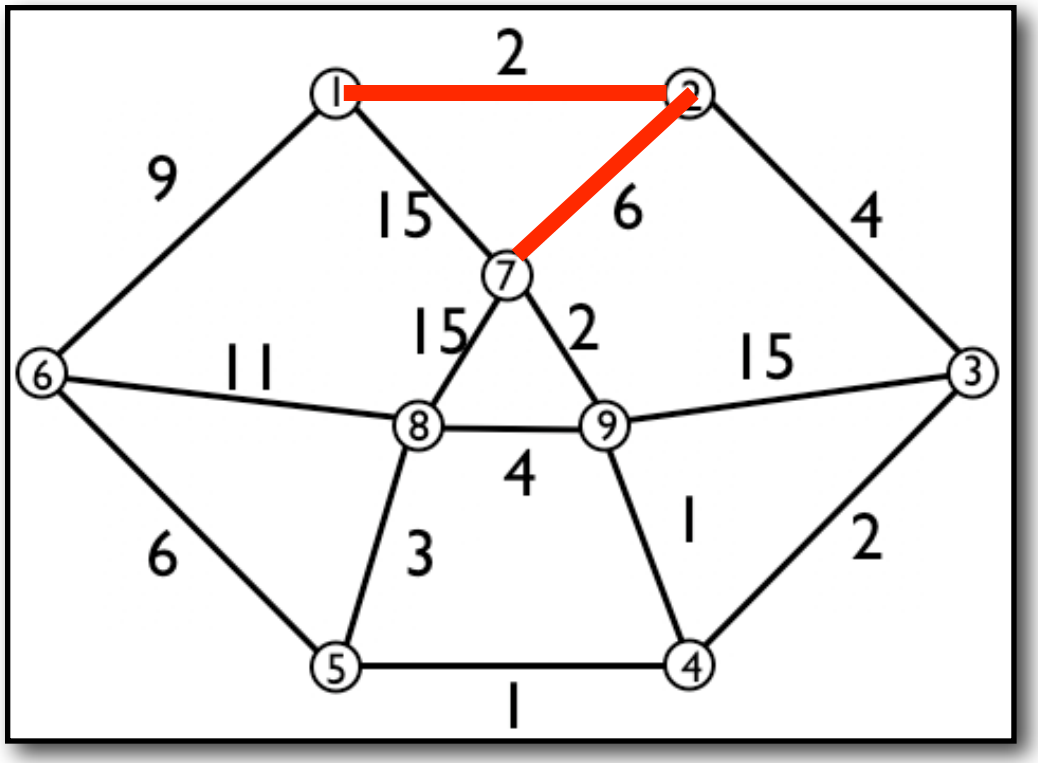


gewählt

(1,0,1)
(2,2,1)
(3,6,2)
(7,8,2)
(4,8,3)
(5,9,4)
(6,9,1)
(9,9,4)
(8,12,5)

Graphenalgorithmen
Kürzeste Wege - Beispiel

Darstellung:
(Knotennr, Distanz, Vorgänger)

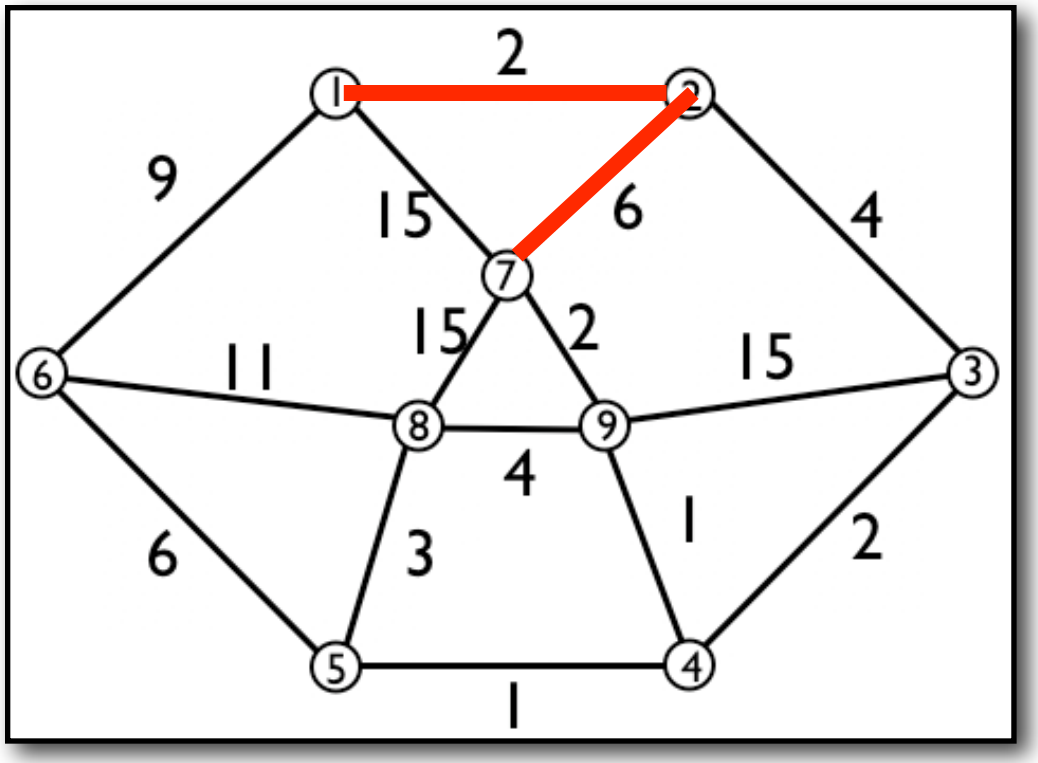


gewählt

(1,0,1)
(2,2,1)
(3,6,2)
(7,8,2)
(4,8,3)
(5,9,4)
(6,9,1)
(9,9,4)
(8,12,5)

Graphenalgorithmen
Kürzeste Wege - Beispiel

Darstellung:
(Knotennr, Distanz, Vorgänger)

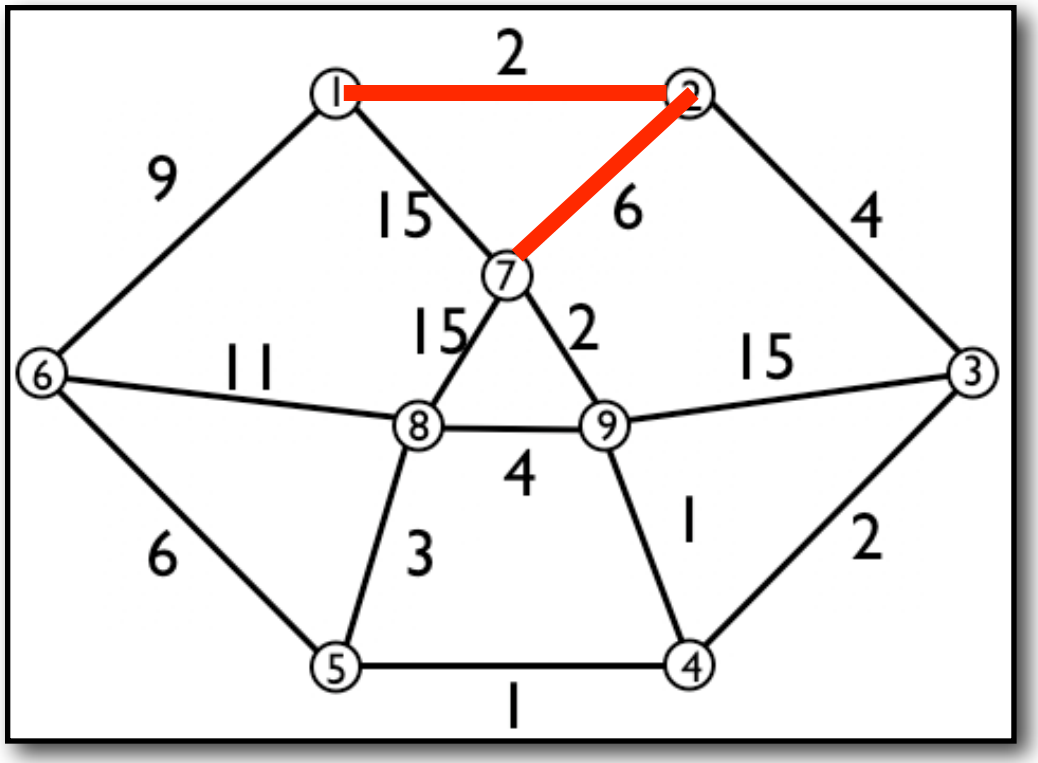


gewählt

(1,0,1)
(2,2,1)
(3,6,2)
(7,8,2)
(4,8,3)
(5,9,4)
(6,9,1)
(9,9,4)
(8,12,5)

Graphenalgorithmen
Kürzeste Wege - Beispiel

Darstellung:
(Knotennr, Distanz, Vorgänger)



gewählt

(1,0,1)
(2,2,1)
(3,6,2)
(7,8,2)
(4,8,3)
(5,9,4)
(6,9,1)
(9,9,4)
(8,12,5)