

# Algorithmus von Kruskal – Idee (1)

---

- Kruskals Algorithmus berechnet minimale Spannbäume, allerdings mit anderer Strategie als Prim's Algorithmus.
- Kruskals Algorithmus erweitert sukzessive Kantenmenge  $A$  zu einem Spannbaum. Zu jedem Zeitpunkt besteht  $G_A = (V, A)$  aus einer Menge von Bäumen.
- Eine Kante minimalen Gewichts, die in  $G_A$  keinen Kreis erzeugt, wird zu  $A$  in jedem Schritt hinzugefügt.

# Algorithmus von Kruskal - Datenstruktur

---

- Kruskals Algorithmus benötigt Datenstruktur mit deren Hilfe
  1. Für jede Kante  $(u,v) \in E$  effizient entschieden werden kann, ob  $u$  und  $v$  in derselben Zusammenhangskomponente von  $G_A$  liegen.
  2. Zusammenhangskomponenten effizient verschmolzen werden können.
- Solch eine Datenstruktur ist *Union-Find*-Datenstruktur für *disjunkte dynamische Mengen*.

# Disjunkte dynamische Mengen (2)

---

- Die folgenden Operationen sollen unterstützt werden:
  1. Make-Set( $x$ ). erzeugt Teilmenge  $\{x\}$  mit Repräsentant  $x$ .
  2. Union( $x,y$ ): vereinigt die beiden Teilmengen, die  $x$  bzw,  $y$  enthalten.
  3. Find-Set( $x$ ): liefert den Repräsentanten derjenigen Menge, die  $x$  enthält.

# Union-Find mit verketteten Listen

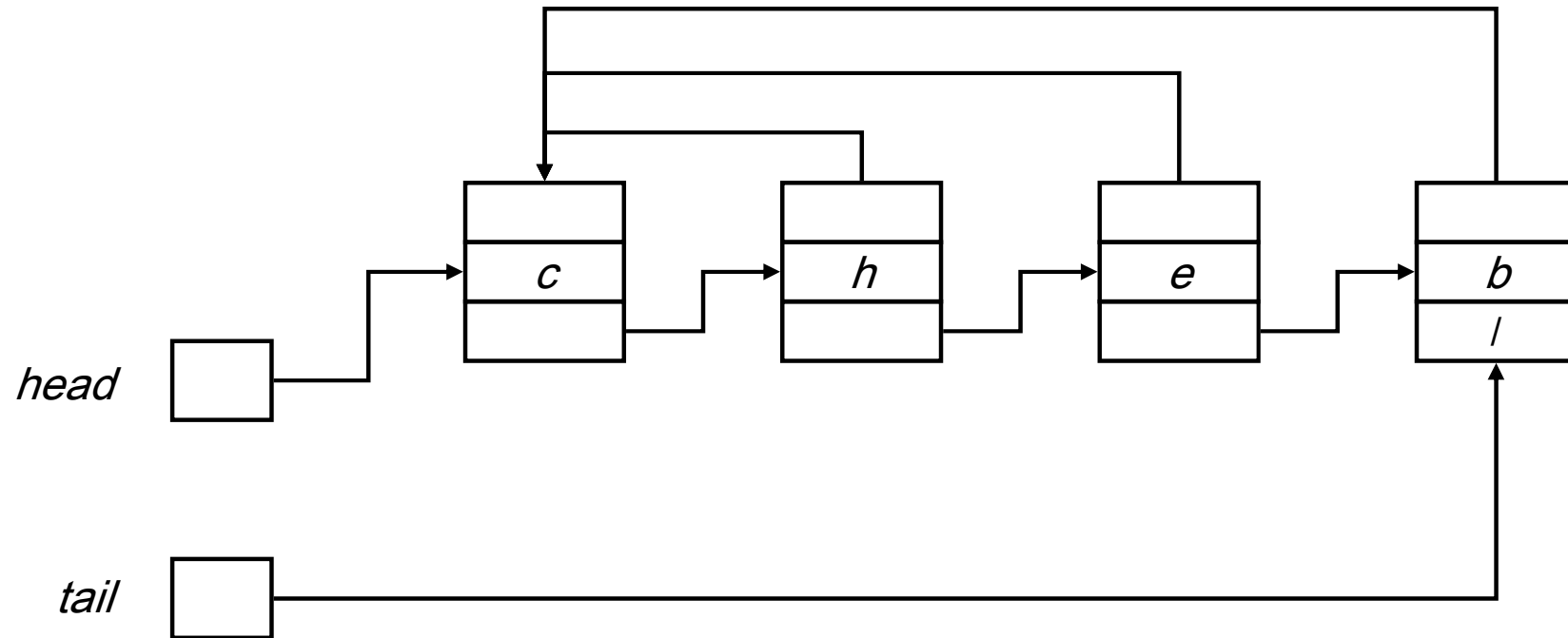
---

Realisierung einer Datenstruktur für disjunkte dynamische Menge kann mit verketteten Listen erfolgen:

1. Für jede Teilmenge  $S$  gibt es eine verkettete Liste  $S$  der Objekte in  $S$ .
2. Repräsentant ist dabei das erste Objekt der Liste.
3. Zusätzlich  $\text{head}[S]$  und  $\text{tail}[S]$  Verweise auf erstes bzw. letztes Element der Liste. Feld  $\text{size}[S]$  speichert Größe der Liste.
4. Für jedes Element  $x$  der Liste Verweis  $\text{rep}[x]$  auf Repräsentanten der Liste.

# Union-Find mit verketteten Listen - Illustration

---



# Operationen Make-Set und Find-Set

---

- *Make-Set*: Erzeugen einer 1-elementigen Liste möglich in Zeit  $O(1)$ .
- *Find-Set(x)*: Ausgabe des Repräsentanten  $\text{rep}[x]$  möglich in Zeit  $O(1)$

# Operation Union

---

*Union(x,y):*

$S_x$  enthalte  $x$  und  $S_y$  enthalte  $y$ .

1. Bestimmen, welche der Listen kürzer ist.
2. Hänge kürzere Liste an längere Liste.
3. Ersetze Repräsentanten für die Elemente der kürzeren Liste.

*Lemma 15.8:* Die Operation Union kann in Zeit proportional zur Länge der kürzeren Liste durchgeführt werden.

# Analyse vieler Operationen

---

*Satz 15.8:* Werden verkettete Listen als Union-Find-Datenstruktur benutzt und werden insgesamt  $m$  Operationen Make-Set, Find-Set und Union ausgeführt, von denen  $n$  Operationen Make-Set Operationen sind, so können alle Operationen zusammen in Zeit  $O(m + n \log(n))$  ausgeführt werden.

# Kruskals Algorithmus und Union-Find

---

- Benutzen Union-Find-Datenstruktur, um Zusammenhangskomponenten von  $G_A = (V, A)$  zu verwalten.
- Test, ob  $(u, v)$  zwei verschiedene Zusammenhangskomponenten verbindet durch Test, ob  $\text{Find-Set}(u) = \text{Find-Set}(v)$ .
- Verschmelzen von Zusammenhangskomponenten durch Union.

# Algorithmus von Kruskal - Pseudocode

---

Kruskal – MST( $G, w$ )

1  $A \leftarrow \{ \}$

2 **for** alle  $v \in V$

3     **do** Make - Set( $v$ )

4 Sortiere die Kanten von  $G$  nach aufsteigendem Gewicht.

5 **for** alle Kante  $(u, v)$  von  $G$  in der Reihenfolge aufsteigenden Gewichts

6     **do if** Find - Set( $u$ )  $\neq$  Find - Set( $v$ )

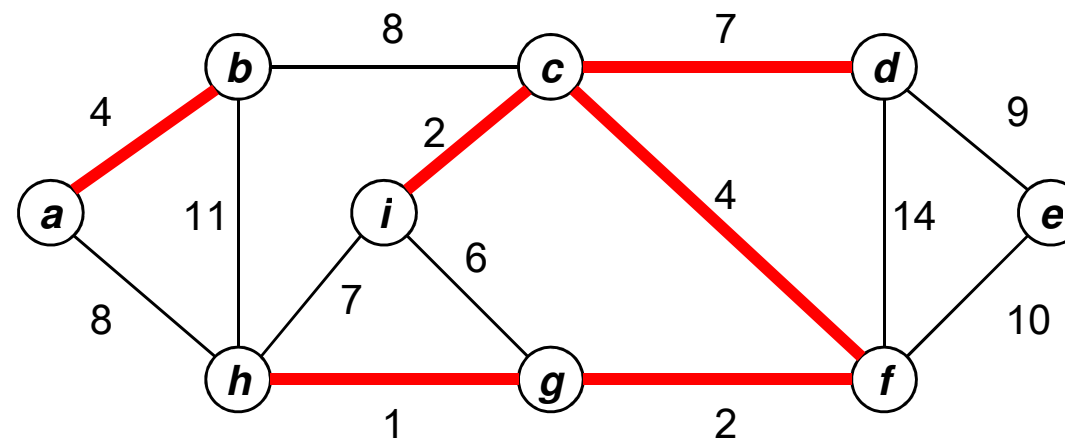
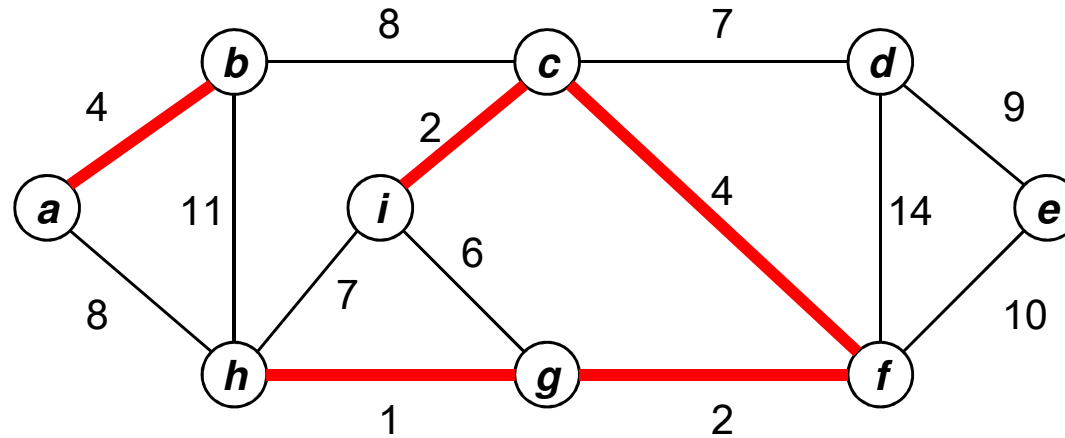
7         **then**  $A \leftarrow A \cup \{(u, v)\}$

8             Union( $u, v$ )

9 **return**  $A$

# Algorithmus von Kruskal – Illustration (3)

---



# Laufzeitanalyse von Kruskals Algorithmus

---

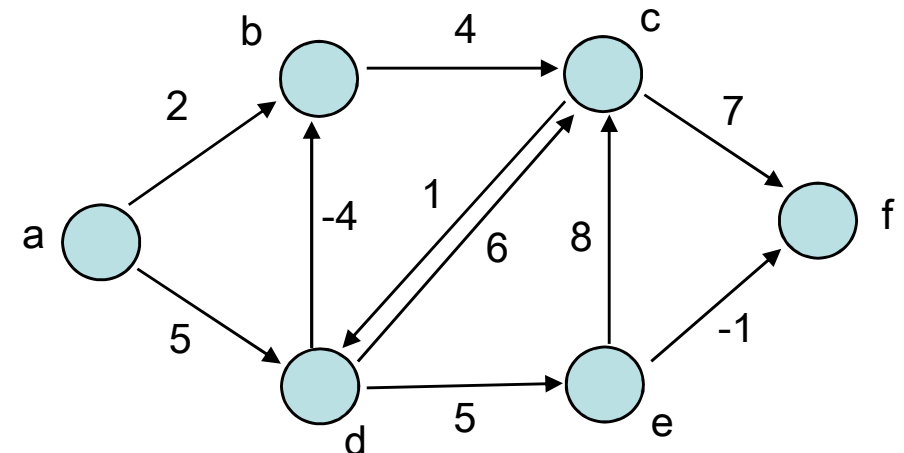
*Satz 15.9:* Werden verkettete Listen als Union-Find-Datenstruktur benutzt, so ist die Laufzeit von Kruskals Algorithmus  $O(|V|\log(|V|) + |E|\log(|E|))$ .

# 16. All Pairs Shortest Path (ASPS)

## All Pairs Shortest Path (APSP):

- Eingabe: Gewichteter Graph  $G=(V,E)$
- Ausgabe: Für jedes Paar von Knoten  $u,v \in V$  die Distanz von  $u$  nach  $v$  sowie einen kürzesten Weg

|   | a        | b        | c        | d        | e        | f  |
|---|----------|----------|----------|----------|----------|----|
| a | 0        | 1        | 5        | 5        | 10       | 9  |
| b | $\infty$ | 0        | 4        | 5        | 10       | 9  |
| c | $\infty$ | -3       | 0        | 1        | 6        | 5  |
| d | $\infty$ | -4       | 0        | 0        | 5        | 4  |
| e | $\infty$ | 5        | 8        | 9        | 0        | -1 |
| f | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | 0  |

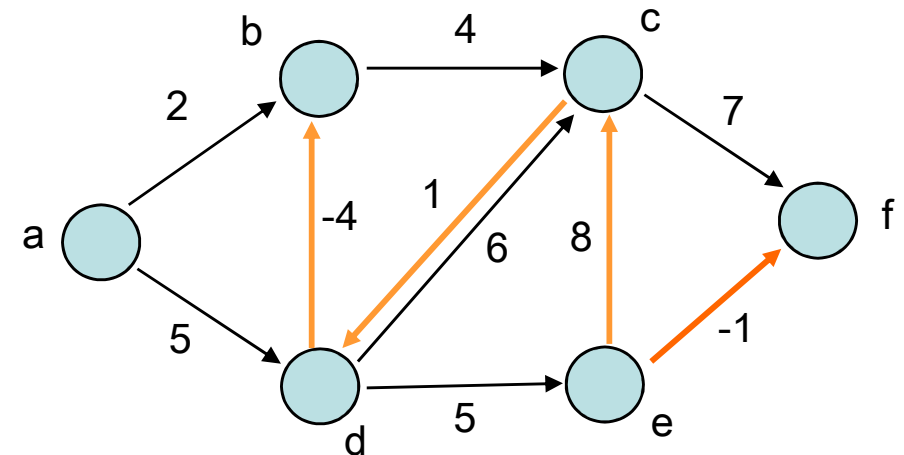


# All Pairs Shortest Path

## All Pairs Shortest Path (APSP):

- Eingabe: Gewichteter Graph  $G=(V,E)$
- Ausgabe: Für jedes Paar von Knoten  $u,v \in V$  die Distanz von  $u$  nach  $v$  sowie einen kürzesten Weg

|   | a        | b        | c        | d        | e        | f  |
|---|----------|----------|----------|----------|----------|----|
| a | 0        | 1        | 5        | 5        | 10       | 9  |
| b | $\infty$ | 0        | 4        | 5        | 10       | 9  |
| c | $\infty$ | -3       | 0        | 1        | 6        | 5  |
| d | $\infty$ | -4       | 0        | 0        | 5        | 4  |
| e | $\infty$ | 5        | 8        | 9        | 0        | -1 |
| f | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | 0  |



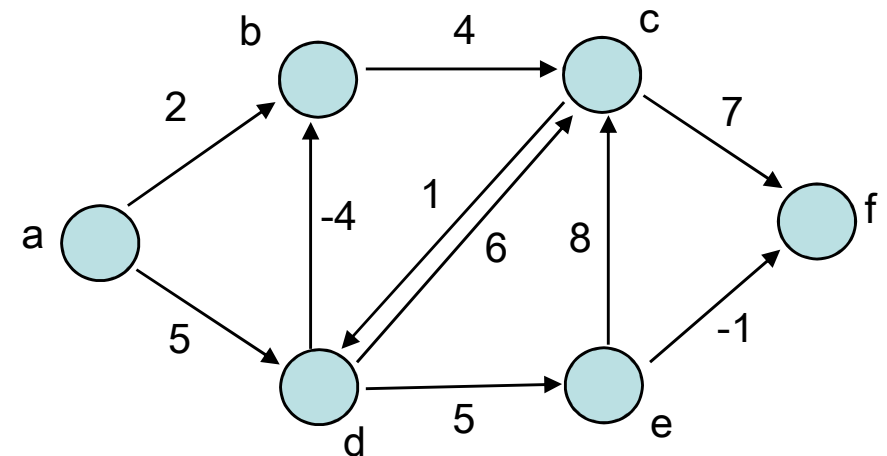
# All Pairs Shortest Path

## Eingabe APSP:

- Matrix  $W=(w_{ij})$ , die Graph repräsentiert

$$w_{ij} = \begin{cases} 0 & , \text{ wenn } i=j \\ \text{Gewicht der ger. Kante } (i,j), & \text{ wenn } i \neq j \text{ und } (i,j) \in E \\ \infty & , \text{ wenn } i \neq j \text{ und } (i,j) \notin E \end{cases}$$

|   | a        | b        | c        | d        | e        | f        |
|---|----------|----------|----------|----------|----------|----------|
| a | 0        | 2        | $\infty$ | 5        | $\infty$ | $\infty$ |
| b | $\infty$ | 0        | 4        | $\infty$ | $\infty$ | $\infty$ |
| c | $\infty$ | $\infty$ | 0        | 1        | $\infty$ | 7        |
| d | $\infty$ | -4       | 6        | 0        | 5        | $\infty$ |
| e | $\infty$ | $\infty$ | 8        | $\infty$ | 0        | -1       |
| f | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | 0        |



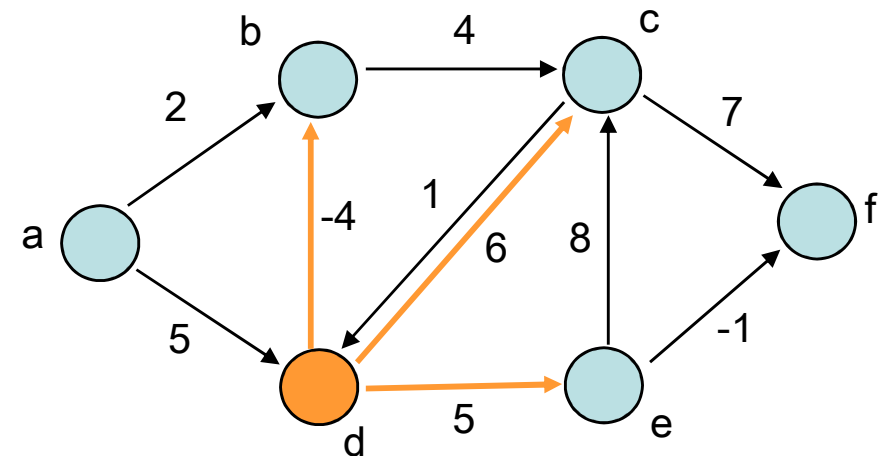
# All Pairs Shortest Path

## Eingabe APSP:

- Matrix  $W=(w_{ij})$ , die Graph repräsentiert

$$w_{ij} = \begin{cases} 0 & , \text{ wenn } i=j \\ \text{Gewicht der ger. Kante } (i,j), & \text{ wenn } i \neq j \text{ und } (i,j) \in E \\ \infty & , \text{ wenn } i \neq j \text{ und } (i,j) \notin E \end{cases}$$

|   | a        | b        | c        | d        | e        | f        |
|---|----------|----------|----------|----------|----------|----------|
| a | 0        | 2        | $\infty$ | 5        | $\infty$ | $\infty$ |
| b | $\infty$ | 0        | 4        | $\infty$ | $\infty$ | $\infty$ |
| c | $\infty$ | $\infty$ | 0        | 1        | $\infty$ | 7        |
| d | $\infty$ | -4       | 6        | 0        | 5        | $\infty$ |
| e | $\infty$ | $\infty$ | 8        | $\infty$ | 0        | -1       |
| f | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | 0        |



# All Pairs Shortest Path

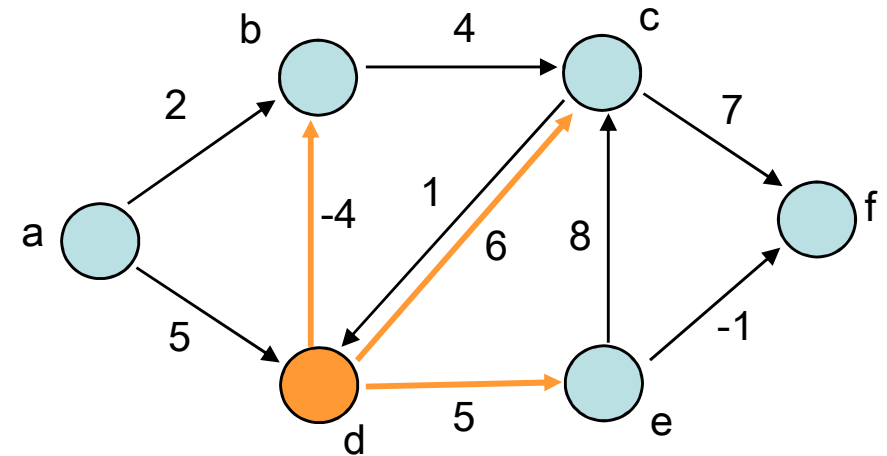
## Eingabe APSP:

- Matrix  $W=(w_{ij})$ , die Graph repräsentiert

Annahme:  
Keine negativen Zyklen!

$$w_{ij} = \begin{cases} 0 & , \text{ wenn } i=j \\ \text{Gewicht der ger. Kante } (i,j), & \text{ wenn } i \neq j \text{ und } (i,j) \in E \\ \infty & , \text{ wenn } i \neq j \text{ und } (i,j) \notin E \end{cases}$$

|   | a        | b        | c        | d        | e        | f        |
|---|----------|----------|----------|----------|----------|----------|
| a | 0        | 2        | $\infty$ | 5        | $\infty$ | $\infty$ |
| b | $\infty$ | 0        | 4        | $\infty$ | $\infty$ | $\infty$ |
| c | $\infty$ | $\infty$ | 0        | 1        | $\infty$ | 7        |
| d | $\infty$ | -4       | 6        | 0        | 5        | $\infty$ |
| e | $\infty$ | $\infty$ | 8        | $\infty$ | 0        | -1       |
| f | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | 0        |

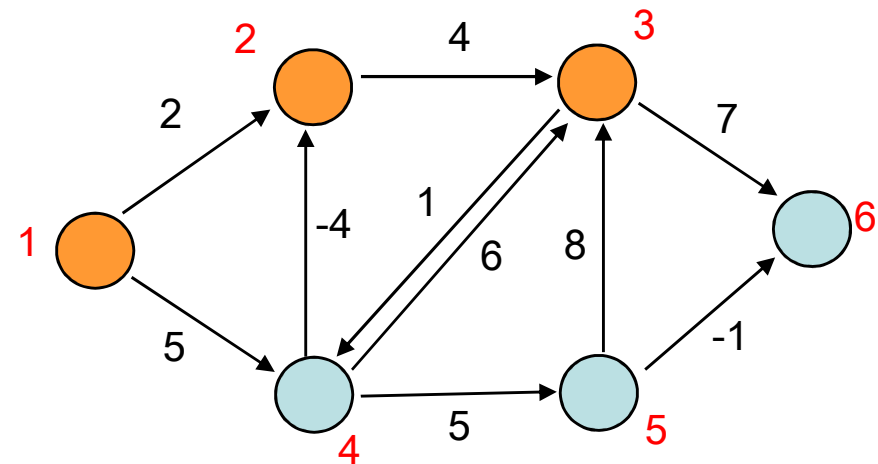


# All Pairs Shortest Path

- Nummeriere Knoten von 1 bis  $n=|V|$
- Betrachte kürzeste  $i$ - $j$ -Wege, die nur über Knoten 1 bis  $k$  laufen

|   | 1        | 2        | 3        | 4        | 5        | 6  |
|---|----------|----------|----------|----------|----------|----|
| 1 | 0        | 2        | 6        | 5        | $\infty$ | 13 |
| 2 | $\infty$ | 0        | 4        | 5        | $\infty$ | 11 |
| 3 | $\infty$ | $\infty$ | 0        | 1        | $\infty$ | 7  |
| 4 | $\infty$ | -4       | 0        | 0        | 5        | 7  |
| 5 | $\infty$ | $\infty$ | 8        | 14       | 0        | -1 |
| 6 | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | 0  |

$k=3$

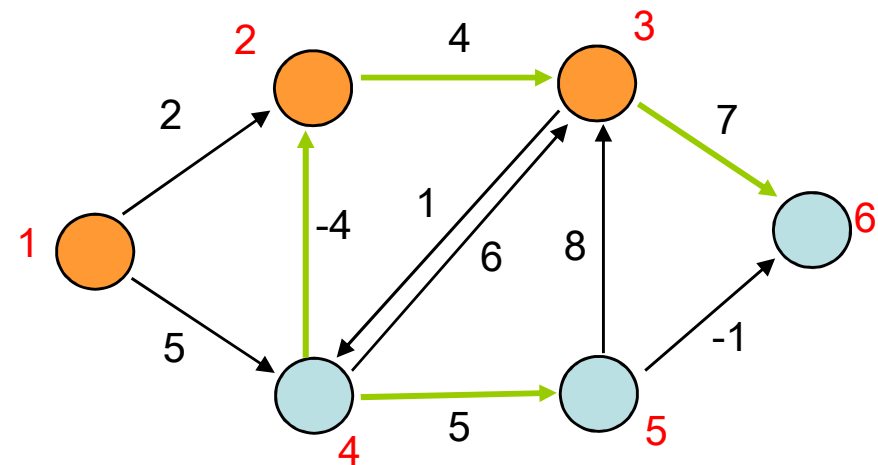


# All Pairs Shortest Path

- Nummeriere Knoten von 1 bis  $n=|V|$
- Betrachte kürzeste  $i$ - $j$ -Wege, die nur über Knoten 1 bis  $k$  laufen

|   | 1        | 2        | 3        | 4        | 5        | 6  |
|---|----------|----------|----------|----------|----------|----|
| 1 | 0        | 2        | 6        | 5        | $\infty$ | 13 |
| 2 | $\infty$ | 0        | 4        | 5        | $\infty$ | 11 |
| 3 | $\infty$ | $\infty$ | 0        | 1        | $\infty$ | 7  |
| 4 | $\infty$ | -4       | 0        | 0        | 5        | 7  |
| 5 | $\infty$ | $\infty$ | 8        | 14       | 0        | -1 |
| 6 | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | 0  |

$k=3$

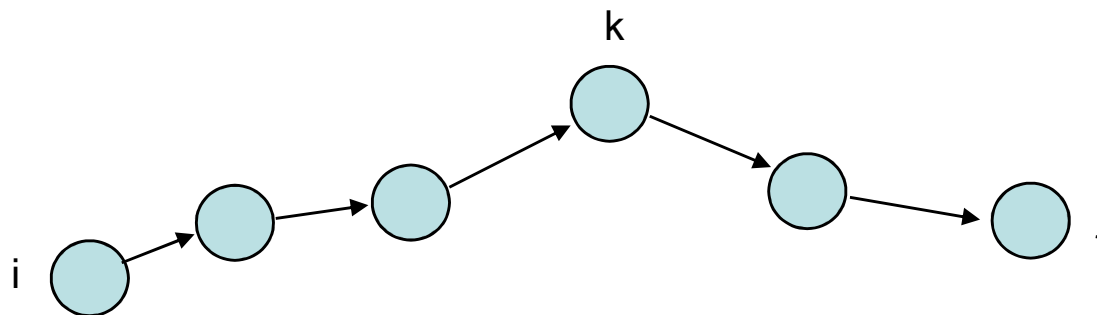


# All Pairs Shortest Path

---

## Zur Erinnerung:

- Sei  $G$  ein Graph ohne negative Zyklen und sei  $j$  von  $i$  aus erreichbar. Dann gibt es einen kürzesten  $i$ - $j$ -Weg, der keinen Knoten doppelt benutzt.
- Wir können also annehmen, dass jeder Knoten in jedem Weg maximal einmal vorkommt
- Betrachte  $i$ - $j$ -Weg, der nur über Knoten aus  $\{1, \dots, k\}$  läuft:

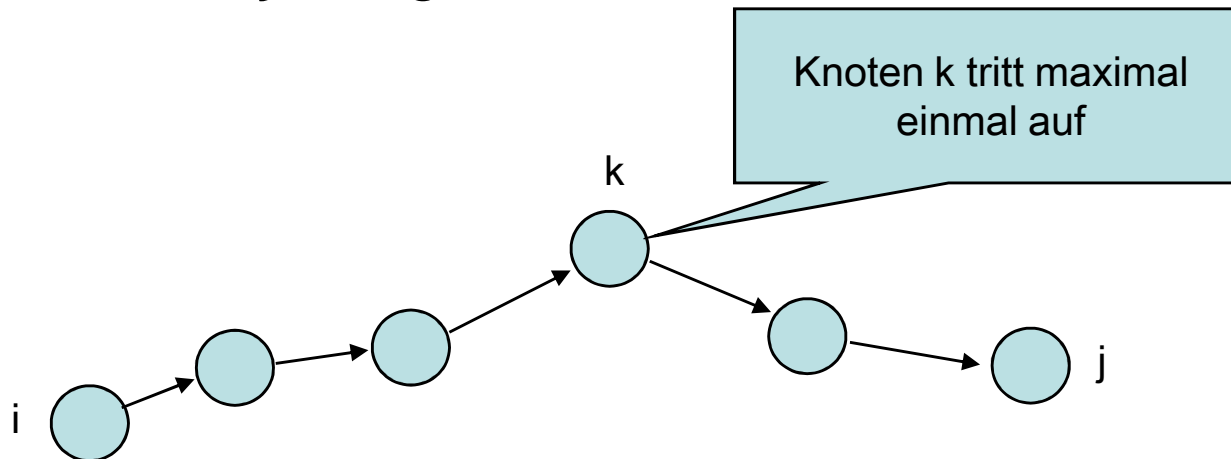


# All Pairs Shortest Path

---

## Zur Erinnerung:

- Sei  $G$  ein Graph ohne negative Zyklen und sei  $j$  von  $i$  aus erreichbar. Dann gibt es einen kürzesten  $i$ - $j$ -Weg, der keinen Knoten doppelt benutzt.
- Wir können also annehmen, dass jeder Knoten in jedem Weg maximal einmal vorkommt
- Betrachte  $i$ - $j$ -Weg, der nur über Knoten aus  $\{1, \dots, k\}$  läuft:

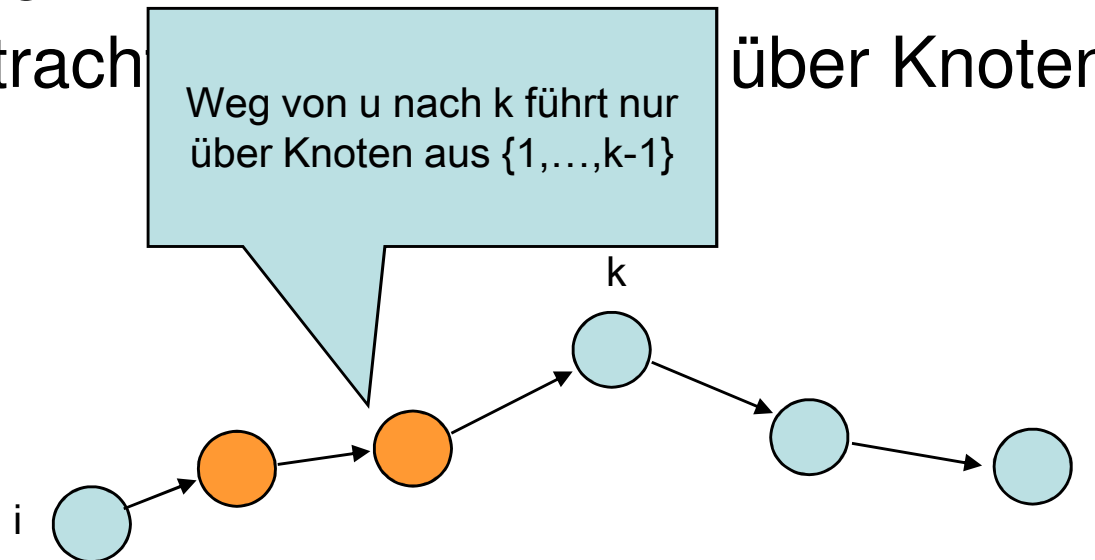


# All Pairs Shortest Path

---

## Zur Erinnerung:

- Sei  $G$  ein Graph ohne negative Zyklen und sei  $j$  von  $i$  aus erreichbar. Dann gibt es einen kürzesten  $i$ - $j$ -Weg, der keinen Knoten doppelt benutzt.
- Wir können also annehmen, dass jeder Knoten in jedem Weg maximal einmal vorkommt
- Betrachte einen kürzesten Weg über Knoten aus  $\{1, \dots, k\}$  läuft:

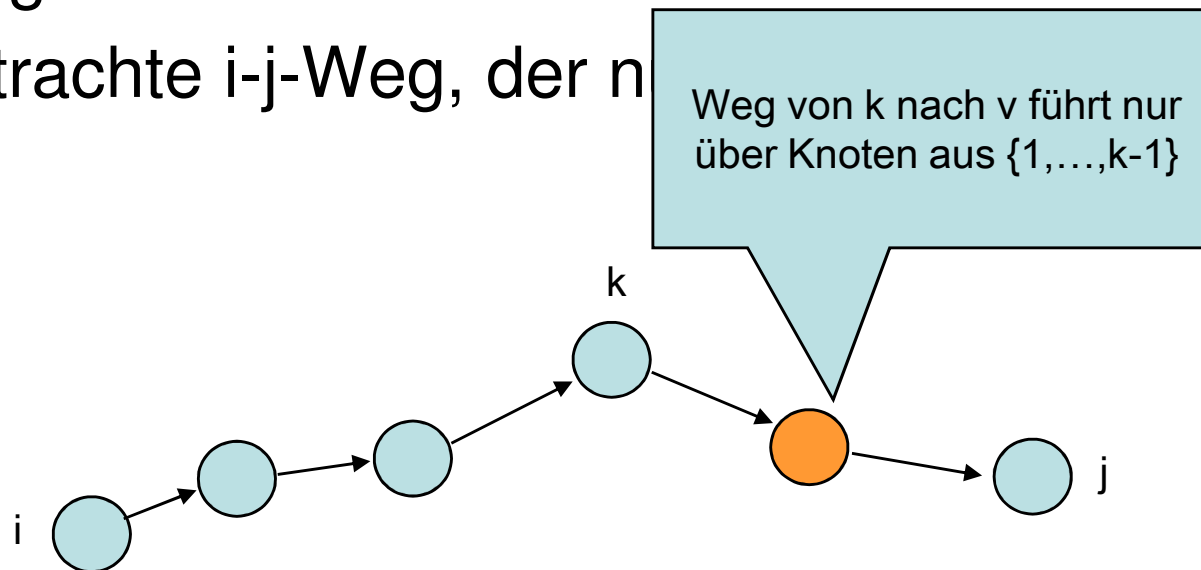


# All Pairs Shortest Path

---

## Zur Erinnerung:

- Sei  $G$  ein Graph ohne negative Zyklen und sei  $j$  von  $i$  aus erreichbar. Dann gibt es einen kürzesten  $i$ - $j$ -Weg, der keinen Knoten doppelt benutzt.
- Wir können also annehmen, dass jeder Knoten in jedem Weg maximal einmal vorkommt
- Betrachte  $i$ - $j$ -Weg, der  $n$  Knoten  $v_1, \dots, v_n$  enthält. Ein Weg  $v_k$  nach  $v_l$  für  $l > k$  läuft:



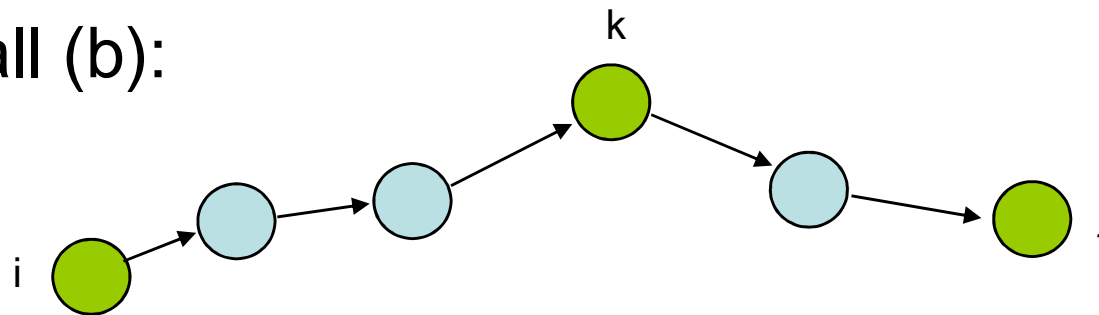
# All Pairs Shortest Path

---

## Die Rekursion:

- Kürzester i-j-Weg über Knoten aus  $\{1, \dots, k\}$  ist
- (a) kürzester i-j-Weg über Knoten aus  $\{1, \dots, k-1\}$  oder
- (b) kürzester i-k-Weg über Knoten aus  $\{1, \dots, k-1\}$  gefolgt von kürzestem k-j-Weg über Knoten aus  $\{1, \dots, k-1\}$

Fall (b):



# All Pairs Shortest Path

---

## Die Rekursion:

- Sei  $d_{ij}^{(k)}$  die Länge eines kürzesten i-j-Wegs mit über Knoten aus  $\{1, \dots, k\}$

$$d_{ij}^{(k)} = \begin{cases} w_{ij} & , \text{ falls } k=0 \\ \min ( d_{ij}^{(k-1)} , d_{ik}^{(k-1)} + d_{kj}^{(k-1)} ) & , \text{ falls } k \geq 1 \end{cases}$$

- Matrix  $D^{(n)} = (d_{ij}^{(n)})$  enthält die gesuchte Lösung

# All Pairs Shortest Path

---

Floyd-Warshall( $W, n$ )

1.  $D^{(0)} \leftarrow W$
2. **for**  $k \leftarrow 1$  **to**  $n$  **do**
3.   **for**  $i \leftarrow 1$  **to**  $n$  **do**
4.     **for**  $j \leftarrow 1$  **to**  $n$  **do**
5.        $d_{ij}^{(k)} \leftarrow \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$
6. **return**  $D^{(n)}$