

# Algorithmen und Datenstrukturen 2

## Lerneinheit 6: Minimal aufspannende Bäume

Prof. Dr. Christoph Karg

Studiengang Informatik  
Hochschule Aalen



Sommersemester 2016



# Einleitung

Thema dieser Lerneinheit sind Algorithmen für Minimal aufspannende Bäume (Minimum Spanning Trees).

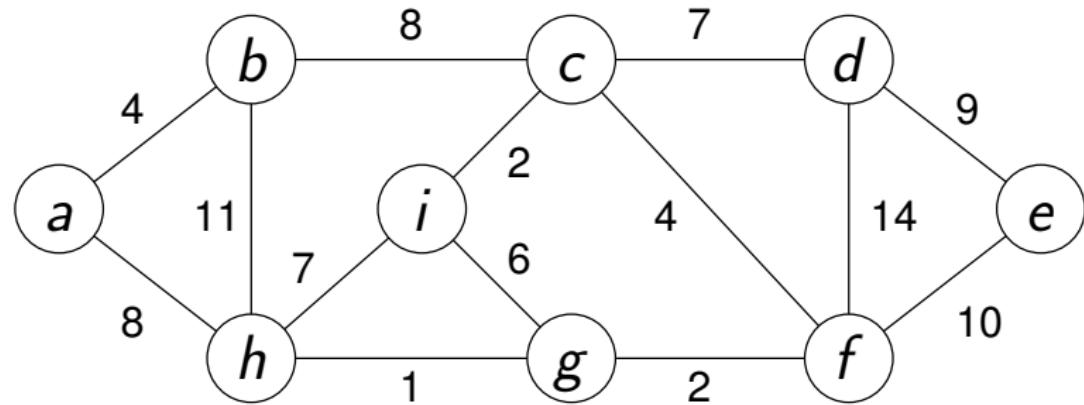
Sie gliedert sich in folgende Abschnitte:

- Generischer Algorithmus
- Algorithmus von Kruskal
- Algorithmus von Prim

# Gewichtete Graphen

- Ein **gewichteter Graph** ist ein gerichteter oder ungerichteter Graph, dessen Kanten mit Zahlenwerten versehen sind
- Die Zahlenwerte werden mittels **Gewichtsfunktion**  $w : E \mapsto \mathbb{R}$  zugewiesen
- Die Kantengewichte werden in modifizierten Adjazenzlisten bzw.-matrizen gespeichert

# Beispiel gewichteter Graph



# Minimum Spanning Tree Problem

Gegeben:

- Ein ungerichteter zusammenhängender Graph  $G = (V, E)$
- Gewichtsfunktion  $w : E \mapsto \mathbb{R}$

Aufgabe: Berechne  $T \subseteq E$  mit folgenden Eigenschaften:

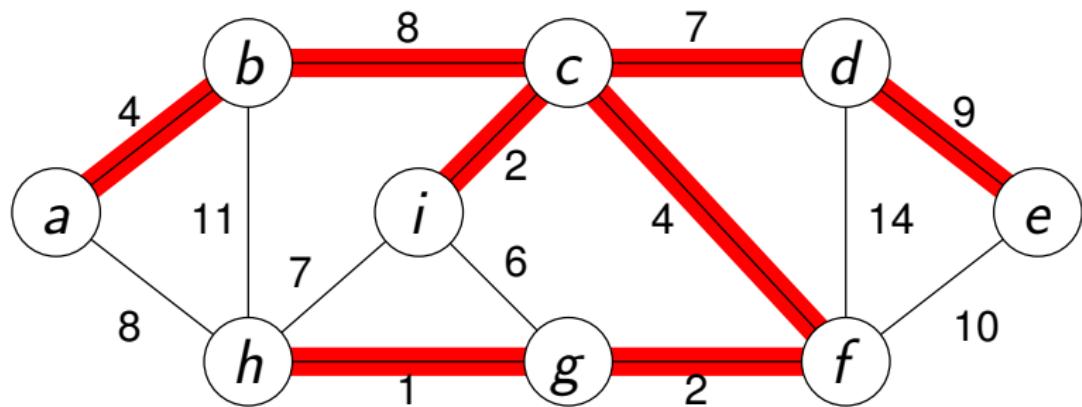
- $G_T = (V, T)$  ist azyklischer zusammenhängender Graph
- Das Gewicht der Kanten in  $T$

$$w(T) = \sum_{(u,v) \in T} w(u, v)$$

ist minimal

# Beispiel Minimal aufspannender Baum

Ein Minimal aufspannender Baum (MST) für das obige Beispiel:



# Generischer Algorithmus

- **Gegeben:**  $G = (V, E)$ ,  $w : E \mapsto \mathbb{R}$
- **Ziel:** Berechnung eines MST mittels einer Greedy Strategie
- **Idee:** Konstruiere schrittweise eine Kantenmenge  $A$  durch Hinzunahme einer geeigneten Kante
- **Schleifeninvariante:** Zu Beginn jeder Iteration ist  $A$  die Teilmenge eines minimal aufspannenden Baums
- Eine Kante  $(u, v)$  heißt **sicher** für  $A$ , falls  $A \cup \{(u, v)\}$  die Teilmenge eines minimal aufspannenden Baums ist

# Algorithmus GENERICMST( $G, w$ )

GENERICMST( $G, w$ )

**Input:** Ungerichteter zusammenhängender Graph  $G = (V, E)$ ,  
Gewichtsfunktion  $w : E \mapsto \mathbb{R}$

**Output:** Minimal aufspannender Baum für  $G$

- 1  $A := \emptyset$
- 2 while  $A$  ist kein Spannbaum do
- 3   Finde eine Kante  $(u, v)$ , die sicher für  $A$  ist
- 4    $A := A \cup \{(u, v)\}$
- 5 return  $A$

# Korrektheit von GENERICMST( $G, w$ )

Die Schleifeninvariante ist korrekt:

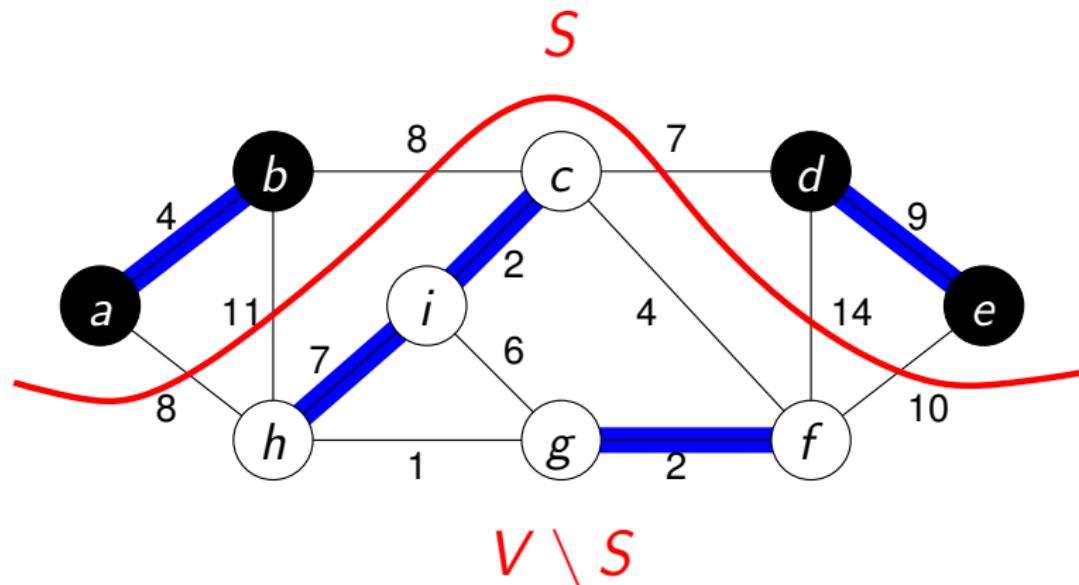
- **Initialisierung:** Die leere Menge ist immer eine Teilmenge eines MST
- **Aufrechterhaltung:** Ist  $A$  die Teilmenge eines MST, dann ist es auch  $A \cup \{(u, v)\}$ , da  $(u, v)$  eine sichere Kante für  $A$  ist
- **Terminierung:** Da  $A$  ausschließlich Kanten eines MST enthält, ist das berechnete Ergebnis ein MST

**Frage:** Wie bestimmt man eine sichere Kante?

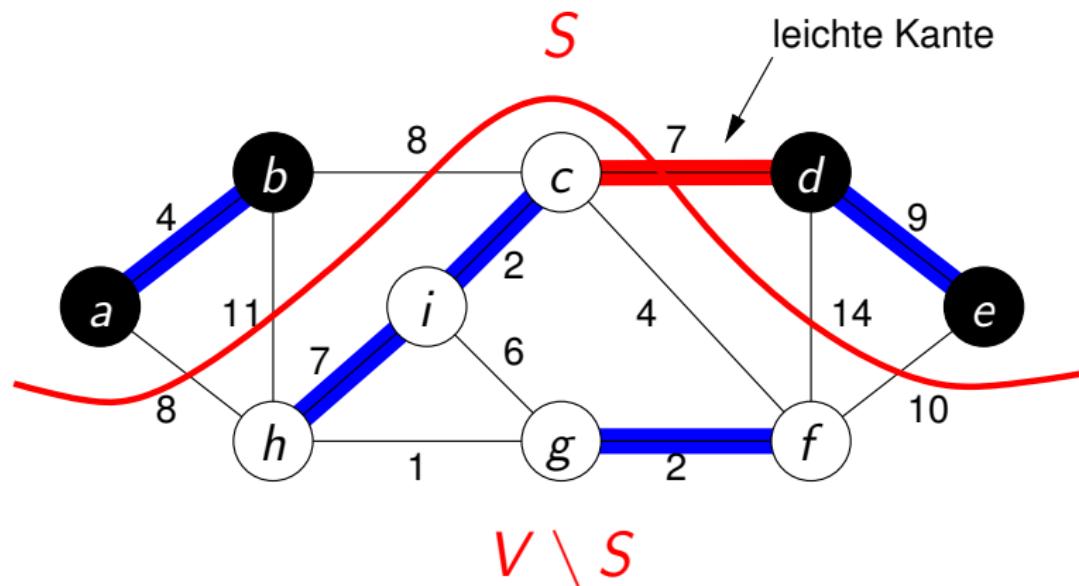
# Begriffsdefinitionen

- Ein **Schnitt**  $(S, V \setminus S)$  eines ungerichteten Graphen  $G = (V, E)$  ist eine Partition von  $V$
- Eine Kante  $(u, v) \in E$  **kreuzt** den Schnitt  $(S, V \setminus S)$ , wenn einer der Knoten  $u$  und  $v$  in  $S$  ist und der andere in  $V \setminus S$
- Ein Schnitt **respektiert** eine Kantenmenge  $A \subseteq E$ , falls keine der Kanten in  $A$  den Schnitt kreuzt
- Eine Kante nennt man **leichte Kante**, wenn sie von allen einen Schnitt kreuzenden Kanten eine mit minimalem Gewicht ist

# Beispiel für einen Schnitt



# Beispiel für einen Schnitt (Forts.)



# Bestimmung einer sicheren Kante

**Satz.** Gegeben ist ein zusammenhängender ungerichteter Graph  $G = (V, E)$  mit zugehöriger Gewichtsfunktion  $w : E \mapsto \mathbb{R}$ .

- Sei  $A \subseteq E$  die Teilmenge eines minimal aufspannenden Baums von  $G$
- Sei  $(S, V \setminus S)$  ein Schnitt von  $G$  der  $A$  respektiert, und
- sei  $(u, v)$  eine leichte Kante, die  $(S, V \setminus S)$  kreuzt.

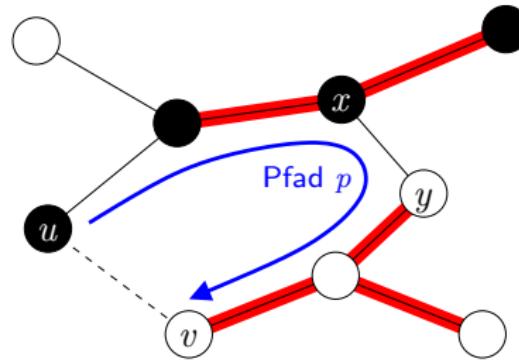
Dann ist die Kante  $(u, v)$  sicher für  $A$ .

**Bemerkung:** Der Satz ist die Basis für die Algorithmen von Kruskal und Prim

# Beweis

Sei  $T$  ein minimaler Spannbaum, der die Kantenmenge  $A$  aber nicht die leichte Kante  $(u, v)$  enthält

Idee: Konstruiere aus  $T$  einen minimalen Spannbaum  $T'$ , der  $(u, v)$  enthält



## Beweis (Forts.)

Die Kante  $(u, v)$  bildet mit dem Pfad  $p$  von  $u$  nach  $v$  in  $T$  einen Zyklus

Da  $u$  und  $v$  auf verschiedenen Seiten des Schnitts  $(S, V \setminus S)$  liegen, gibt es auf dem Pfad  $p$  mindestens eine weitere Kante  $(x, y)$ , die den Schnitt kreuzt

$(x, y)$  ist nicht in  $A$ , da der Schnitt  $A$  respektiert

Da  $(x, y)$  genau einmal auf dem Pfad vorkommt, kann man in  $T$  die Kante  $(x, y)$  durch  $(u, v)$  ersetzen

Der neue Baum  $T' = T \setminus \{(x, y)\} \cup \{(u, v)\}$

## Beweis (Forts.)

Da die Kanten  $(u, v)$  und  $(x, y)$  beide den Schnitt kreuzen und  $(u, v)$  eine leichte Kante ist, gilt  $w(u, v) \leq w(x, y)$

$T'$  ist ein minimaler Spannbaum, denn

$$w(T') = w(T) - w(x, y) + w(u, v) \leq w(T)$$

Da  $T$  ein minimaler Spannbaum ist, gilt  $w(T') \geq w(T)$

Wegen  $A \subseteq T$  und  $(x, y) \notin A$ , gilt  $A \subset T'$ .

Da  $A \cup \{(u, v)\} \in T'$  und  $T'$  ein minimaler Spannbaum ist, ist die Kante  $(u, v)$  sicher für  $A$

Somit ist der Satz bewiesen. ✓

# Algorithmus von Kruskal

- Die Kantenmenge  $A$  ist ein Wald, dessen Komponenten Schritt für Schritt verbunden werden
- Hilfsdatenstruktur: Menge von disjunkten Teilmengen von  $V$  mit folgenden Operationen:
  - ▷  $\text{MAKESET}(v) \rightsquigarrow$  Menge  $\{v\}$  erstellen
  - ▷  $\text{FINDSET}(v) \rightsquigarrow$  Menge finden, die  $v$  enthält
  - ▷  $\text{UNION}(u, v) \rightsquigarrow$  Vereinigung der Mengen die  $u$  bzw.  $v$  enthalten
- Greedy Strategie: Auswahl einer leichten Kante  $(u, v)$  mit  $\text{FINDSET}(u) \neq \text{FINDSET}(v)$

# Algorithmus MST-KRUSKAL( $G, w$ )

MST-KRUSKAL( $G, w$ )

**Input:** Zusammenhängender ungerichteter Graph  $G = (V, E)$   
Gewichtsfunktion  $w : E \mapsto \mathbb{R}$

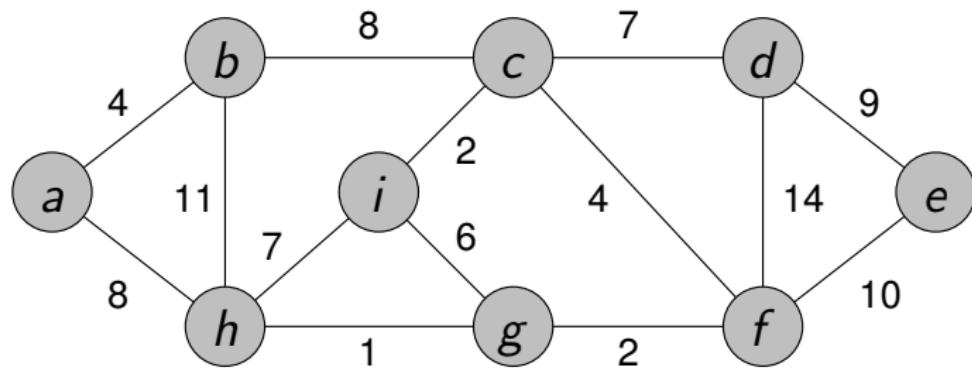
**Output:** Minimaler Spannbaum  $T$  für  $G$

- 1  $A := \emptyset$
- 2 **for** jeden Knoten  $v \in V$  **do**
- 3   MAKESET( $v$ )
- 4   *Sortiere die Kanten in  $E$  gemäß  $w$  in aufsteigender Reihenfolge*
- 5   **for** jede Kante  $(u, v) \in E$  in obiger Reihenfolge **do**
- 6     **if** FINDSET( $u$ )  $\neq$  FINDSET( $v$ ) **then**
- 7        $A := A \cup \{(u, v)\}$
- 8       UNION( $u, v$ )
- 9 **return**  $A$

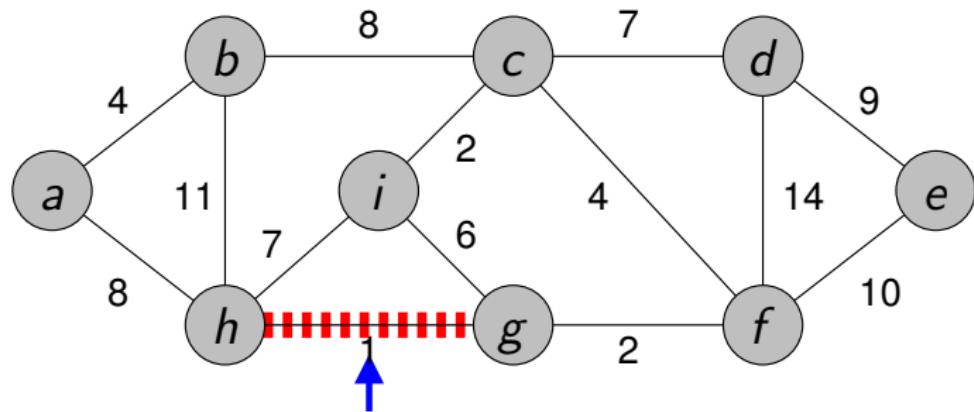
# Bemerkungen

- In Zeile 3 werden  $\|V\|$  disjunkte Mengen erzeugt
- Die absteigende Sortierung in Zeile 4 ermöglicht eine optimale Kantenauswahl
- Eine Kante wird nur ausgewählt, wenn sie zwei disjunkte Knotenmengen verbindet
- Für Details zur Implementierung der Mengenoperationen siehe Cormen: Abschnitt 21.3 (disjoint set data structures)
- Laufzeit:  $O(\|E\| \log_2 \|V\|)$

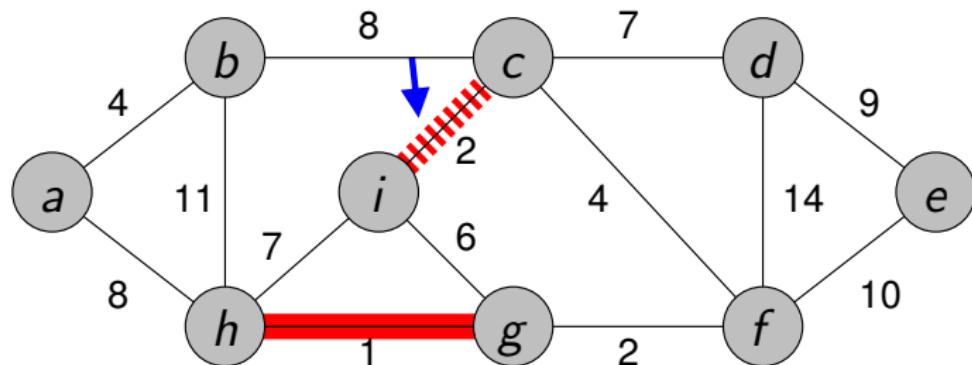
# Beispiel Kruskal



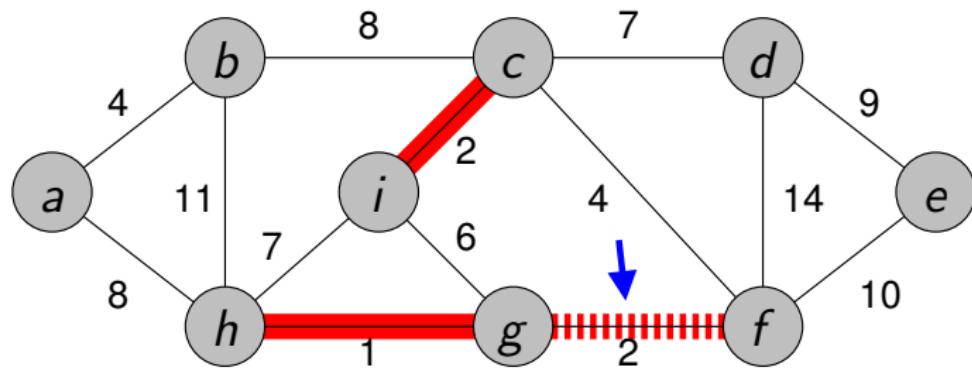
# Beispiel Kruskal (Forts.)



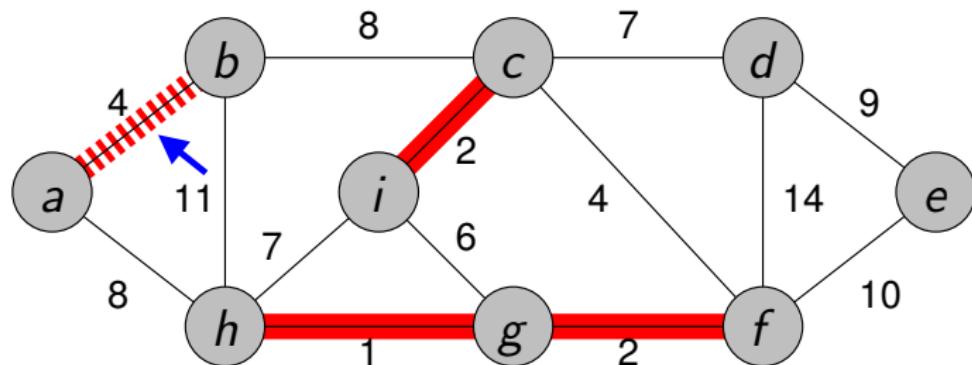
# Beispiel Kruskal (Forts.)



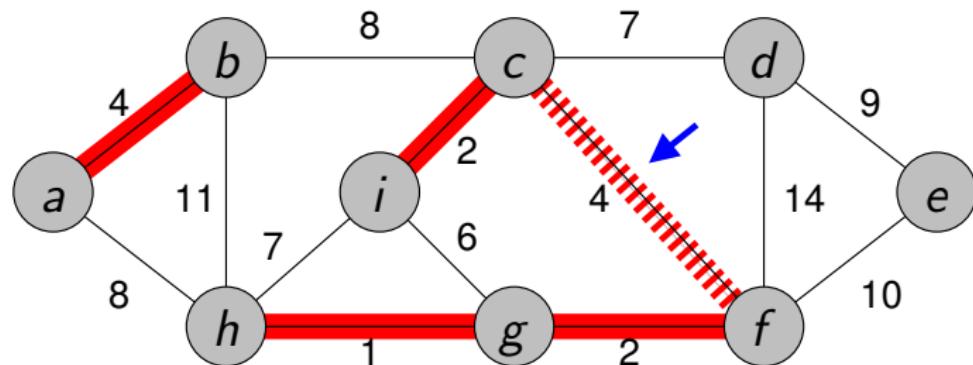
## Beispiel Kruskal (Forts.)



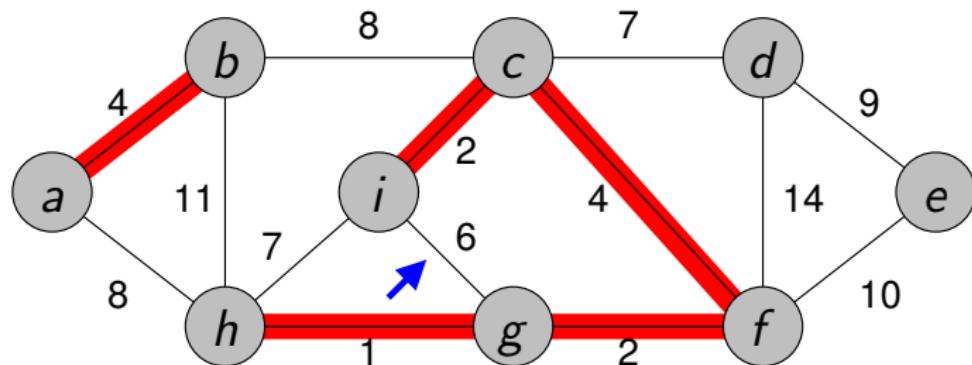
# Beispiel Kruskal (Forts.)



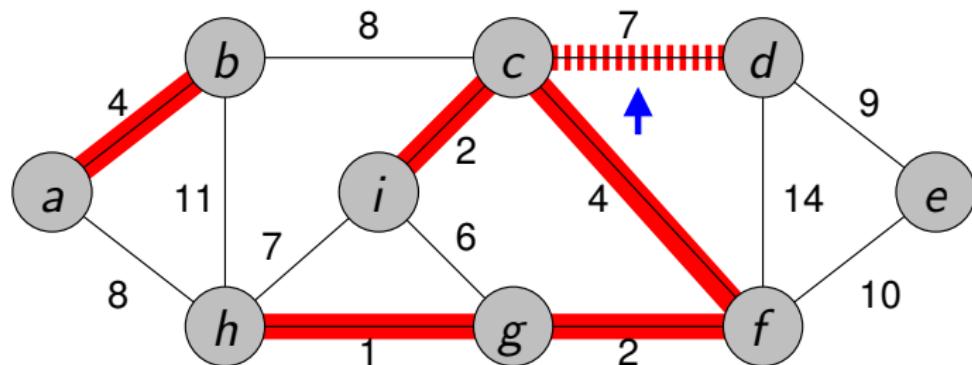
## Beispiel Kruskal (Forts.)



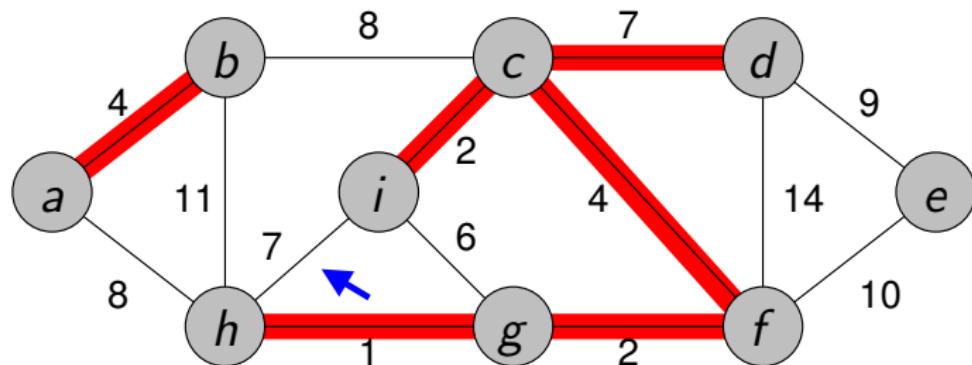
## Beispiel Kruskal (Forts.)



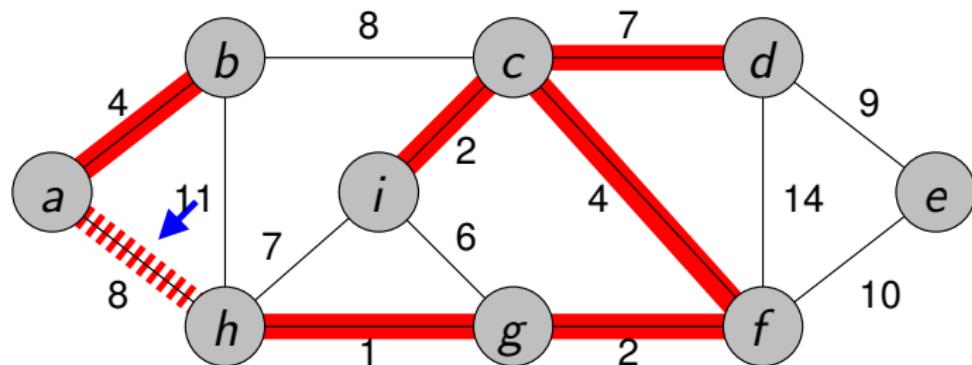
## Beispiel Kruskal (Forts.)



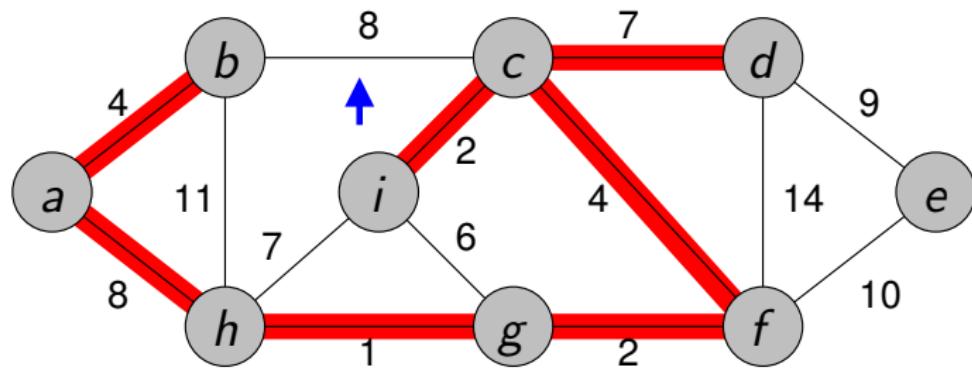
## Beispiel Kruskal (Forts.)



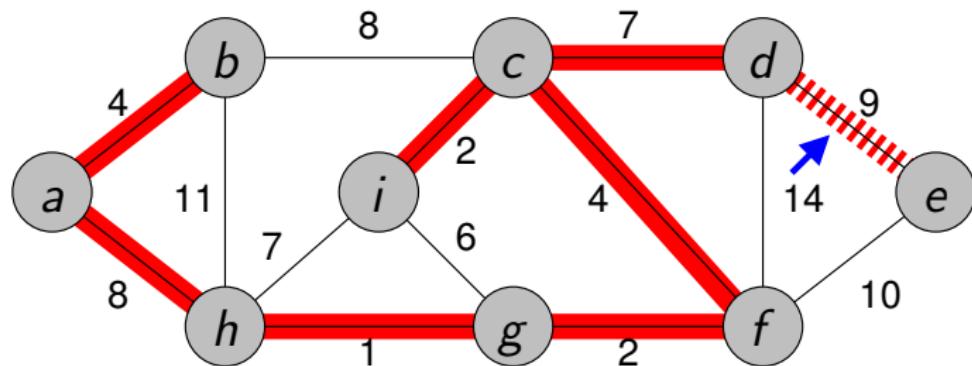
# Beispiel Kruskal (Forts.)



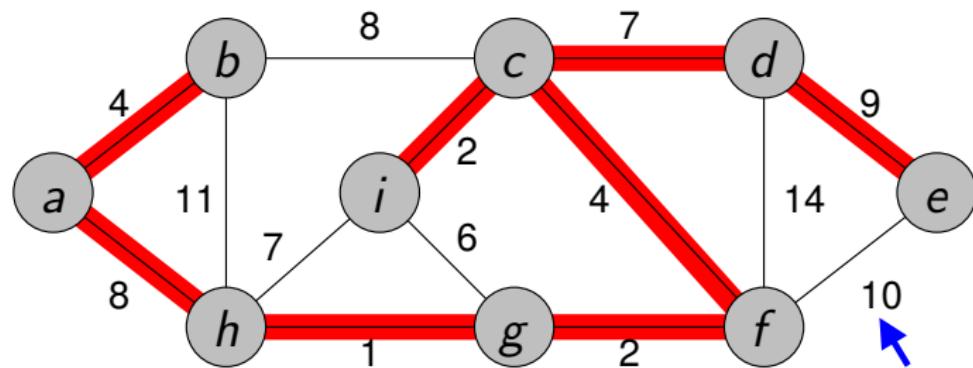
# Beispiel Kruskal (Forts.)



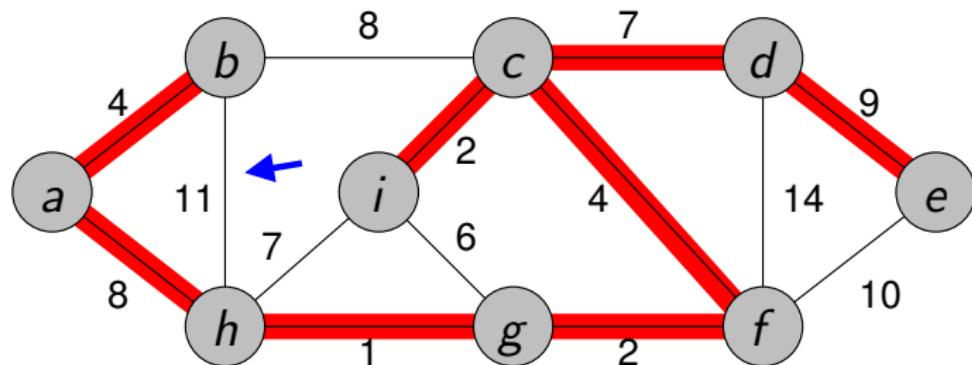
## Beispiel Kruskal (Forts.)



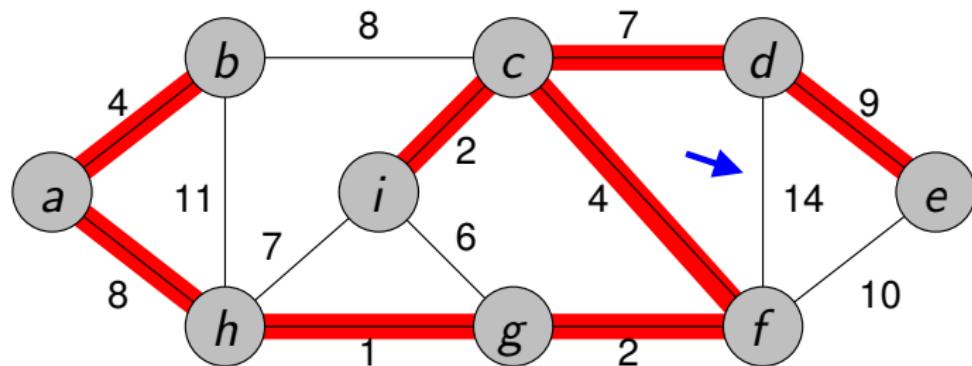
## Beispiel Kruskal (Forts.)



## Beispiel Kruskal (Forts.)



## Beispiel Kruskal (Forts.)



# Algorithmus von Prim

- Berechnung eines minimalen Spannbaums durch Erweiterung eines Teilbaums ausgehend vom Startknoten  $r$
- Ein Min-Priority Queue enthält alle Knoten, die *nicht* im Teilbaum enthalten sind
- Knoteninformationen:
  - ▷  $key[v] \rightsquigarrow$  Wert einer minimalen Kante vom Teilgraph zum Knoten  $v$
  - ▷  $\pi[v] \rightsquigarrow$  Endknoten der minimalen Kante  $(v, \pi[v])$
- Die Kantenmenge des minimalen Spannbaums ist

$$A = \{(v, \pi[v]) \mid v \in V \setminus \{r\}\}$$

- Laufzeit:  $O(\|E\| \log_2 \|V\|)$

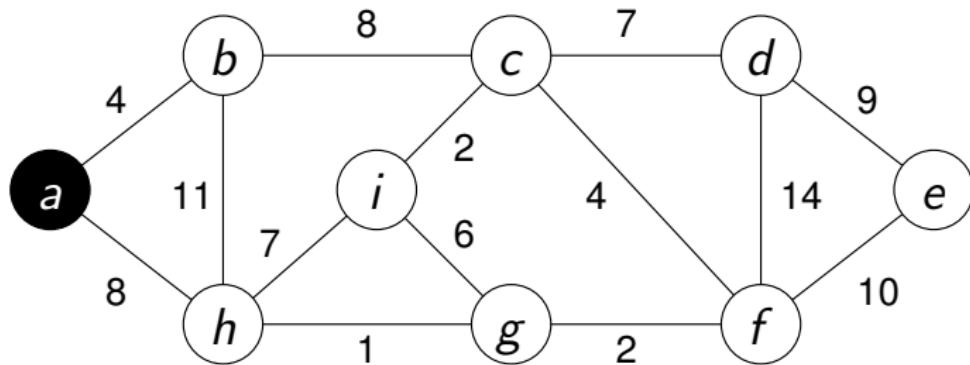
# Algorithmus MSTPRIM( $G, w, r$ )

MSTPRIM( $G, w, s$ )

**Input:** Graph  $G = (V, E)$ , Gewichtsfunktion  $w : E \mapsto \mathbb{R}$ ,  
Startknoten  $s \in V$

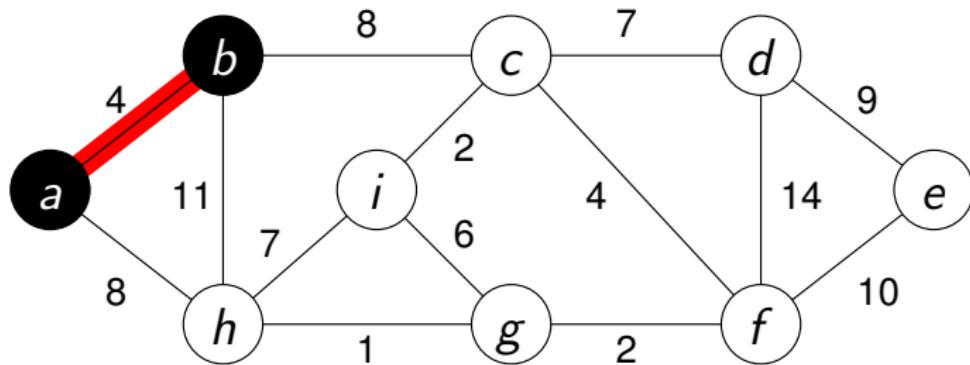
- 1 **for** jeden Knoten  $u \in V$  **do**
- 2    $key[u] := \infty$
- 3    $\pi[u] := \text{NIL}$
- 4    $key[s] := 0$
- 5   *Füge alle Knoten in  $V$  in die Priority Queue  $Q$  ein*
- 6 **while**  $Q$  ist nicht leer **do**
- 7    $u := \text{EXTRACTMIN}(Q)$
- 8   **for** jeden Knoten  $v \in Adj[u]$  **do**
- 9     **if**  $v \in Q$  und  $w(u, v) < key[v]$  **then**
- 10        $\pi[v] := u$
- 11        $\text{DECREASEKEY}(Q, v, w(u, v))$

# Beispiel Prim



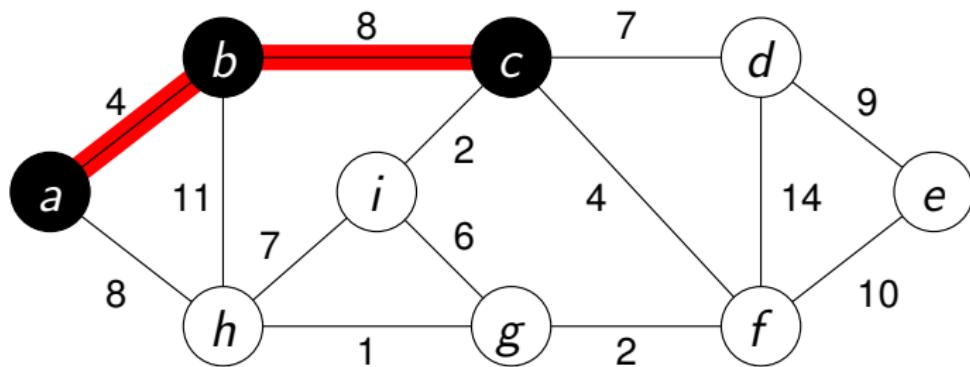
$v$	$a$	$b$	$c$	$d$	$e$	$f$	$g$	$h$	$i$
$key[v]$	0	4	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	8	$\infty$
$\pi[v]$	NIL	$a$	NIL	NIL	NIL	NIL	NIL	$a$	NIL

# Beispiel Prim (Forts.)



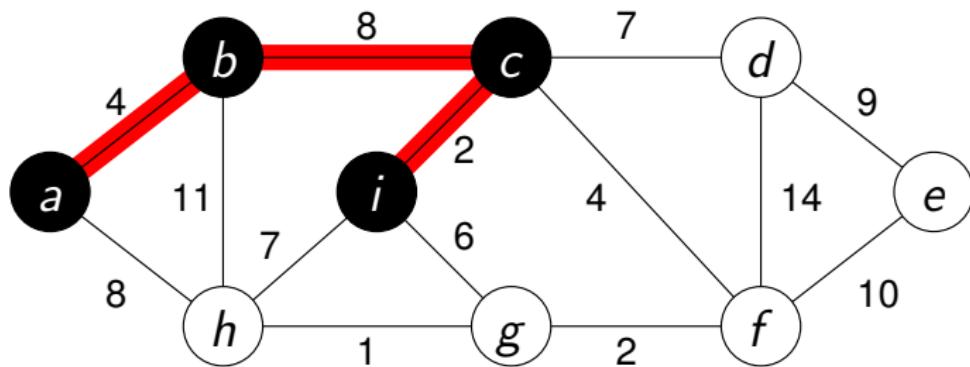
$v$	$a$	$b$	$c$	$d$	$e$	$f$	$g$	$h$	$i$
$key[v]$	0	4	8	$\infty$	$\infty$	$\infty$	$\infty$	8	$\infty$
$\pi[v]$	NIL	$a$	$b$	NIL	NIL	NIL	NIL	$a$	NIL

# Beispiel Prim (Forts.)



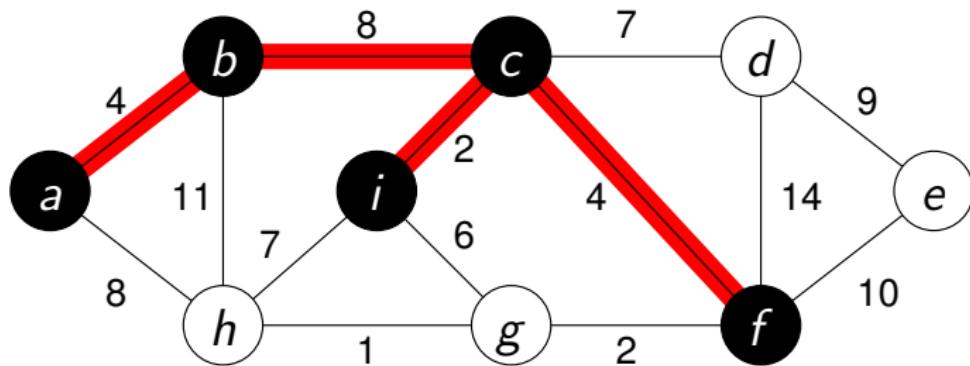
$v$	$a$	$b$	$c$	$d$	$e$	$f$	$g$	$h$	$i$
$key[v]$	0	4	8	7	$\infty$	4	$\infty$	8	2
$\pi[v]$	NIL	$a$	$b$	$c$	NIL	$c$	NIL	$a$	$c$

# Beispiel Prim (Forts.)



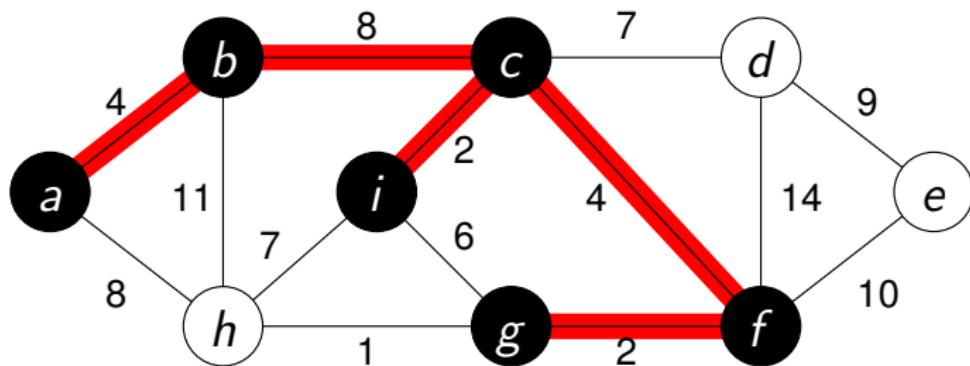
$v$	$a$	$b$	$c$	$d$	$e$	$f$	$g$	$h$	$i$
$key[v]$	0	4	8	7	$\infty$	4	6	7	2
$\pi[v]$	NIL	$a$	$b$	$c$	NIL	$c$	$i$	$i$	$c$

# Beispiel Prim (Forts.)



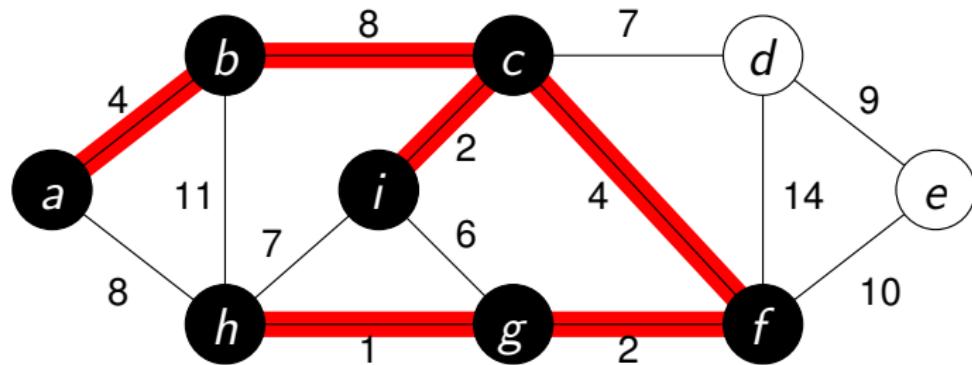
$v$	$a$	$b$	$c$	$d$	$e$	$f$	$g$	$h$	$i$
$key[v]$	0	4	8	7	10	4	2	7	2
$\pi[v]$	NIL	$a$	$b$	$c$	$f$	$c$	$f$	$i$	$c$

# Beispiel Prim (Forts.)



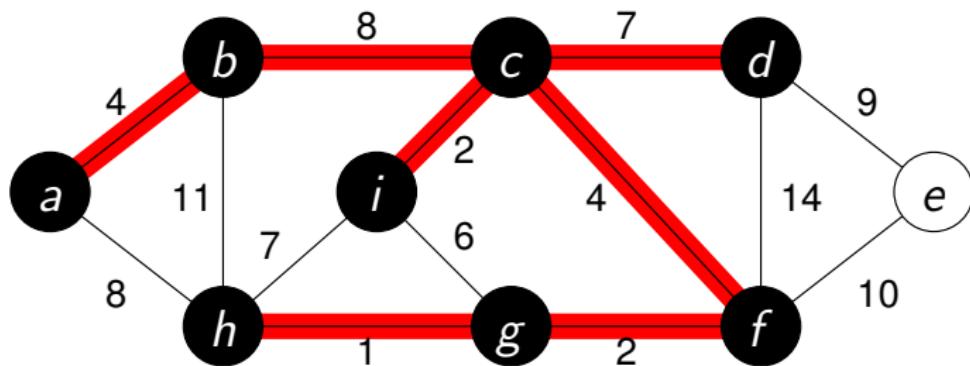
$v$	$a$	$b$	$c$	$d$	$e$	$f$	$g$	$h$	$i$
$key[v]$	0	4	8	7	10	4	2	1	2
$\pi[v]$	NIL	$a$	$b$	$c$	$f$	$c$	$f$	$g$	$c$

# Beispiel Prim (Forts.)



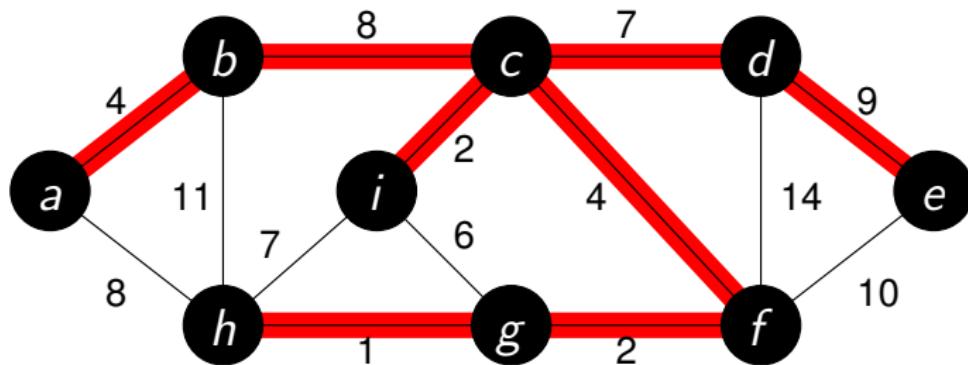
$v$	$a$	$b$	$c$	$d$	$e$	$f$	$g$	$h$	$i$
$key[v]$	0	4	8	7	10	4	2	1	2
$\pi[v]$	NIL	$a$	$b$	$c$	$f$	$c$	$f$	$g$	$c$

# Beispiel Prim (Forts.)



$v$	$a$	$b$	$c$	$d$	$e$	$f$	$g$	$h$	$i$
$key[v]$	0	4	8	7	9	4	2	1	2
$\pi[v]$	NIL	$a$	$b$	$c$	$d$	$c$	$f$	$g$	$c$

# Beispiel Prim (Forts.)



$v$	$a$	$b$	$c$	$d$	$e$	$f$	$g$	$h$	$i$
$key[v]$	0	4	8	7	9	4	2	1	2
$\pi[v]$	NIL	$a$	$b$	$c$	$d$	$c$	$f$	$g$	$c$

# Zusammenfassung

- Generischer MST-Algorithmus auf Basis von Schnitten in Graphen
- Implementierung des generischen Algorithmus:
  - ▷ Algorithmus von Kruskal
  - ▷ Algorithmus von Prim
- Voraussetzung: effiziente Datenstrukturen
  - ▷ Mengen ( $\rightsquigarrow$  Kruskal)
  - ▷ Priority Queue ( $\rightsquigarrow$  Prim)