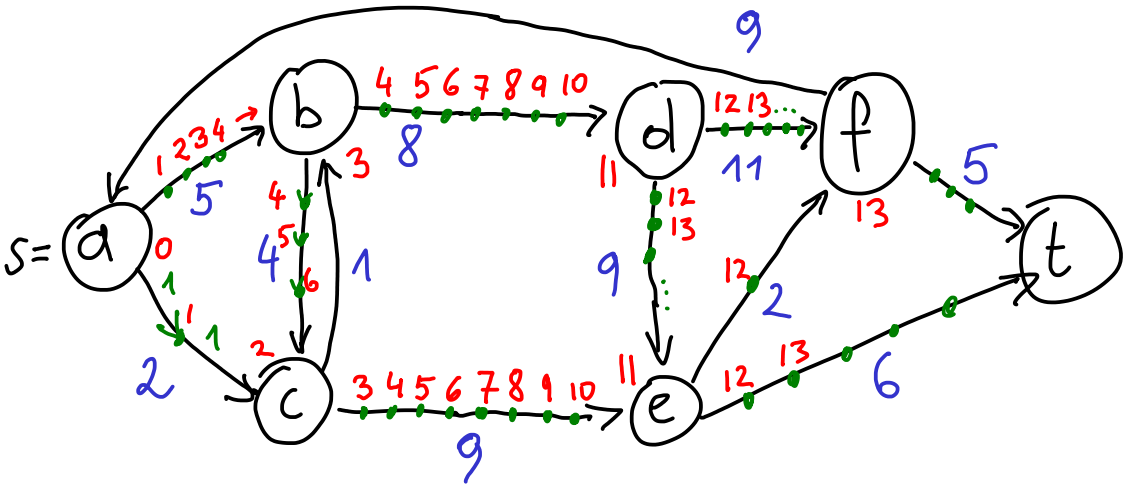
Kürzeste Wege, der Algorithmus von Dijkstra

Kantengewichte $c_{uv} \geq 0$ kürzester Weg von s nach t



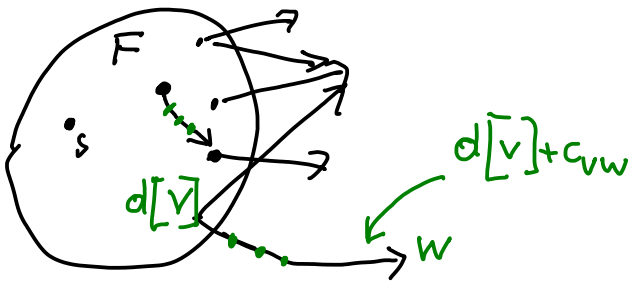
- Simulation durch Breitensuche
- Die meiste Zeit wandert der Algorithmus auf mehreren Kanten parallel Schritt für Schritt.
- Wenn ein Knoten u erreicht wird, können wir $d[u]$ endgültig setzen.
  Der Algorithmus betritt alle von u ausgehenden Kanten gleichzeitig.

¿ Welches ist der nächste Knoten, der erreicht wird ?

F := Menge der schon erreichten „fertigen" Knoten v : $d[v] \leq i$

Der Algorithmus ist auf folgenden Kanten (v, w) unterwegs: $v \in F$, $w \notin F$

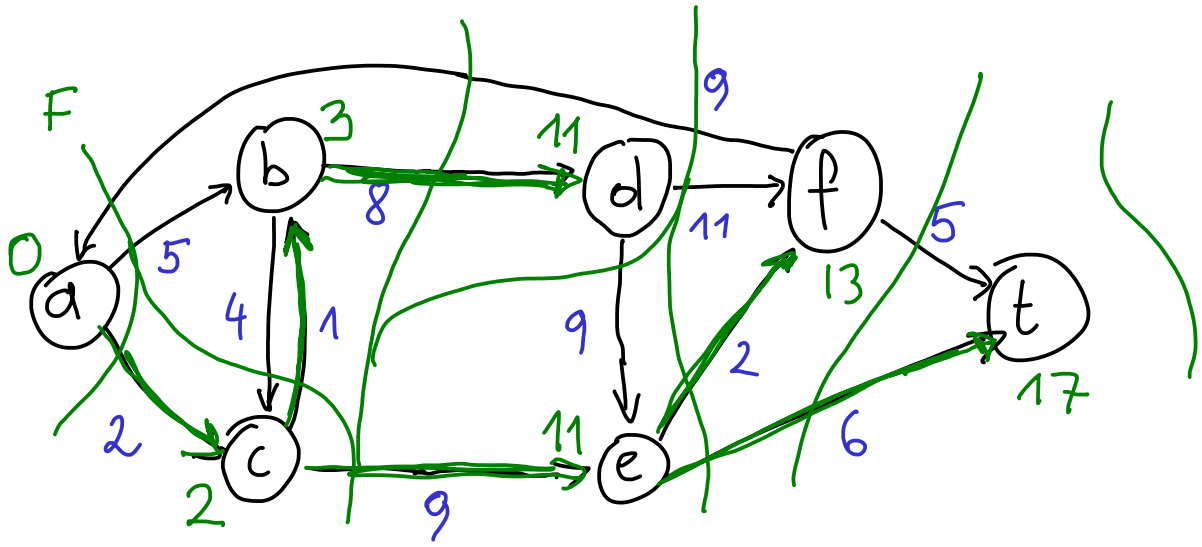
(auch von $v \in F$ zu $w \in F$, aber diese Kanten sind irrelevant.)



- Bestimme $\min \{ d[v] + c_{vw} \mid v \in F, w \notin F, vw \in E \}$

$= d[\bar{v}] + c_{\bar{v}\bar{w}}$

- $d[\bar{w}] := d[\bar{v}] + c_{\bar{v}\bar{w}}$, nimm $\bar{w}$ in F auf.



$d[s] := 0$; $F = \{s\}$

Schleife

Bestimme $\min \{ d[v] + c_{vw} \mid vw \in E, v \in F, w \notin F \}$

$= d[\bar{v}] + c_{\bar{v}\bar{w}} =: D$

← wenn $D = \infty$: STOP

$f[\bar{w}] := \bar{v}$

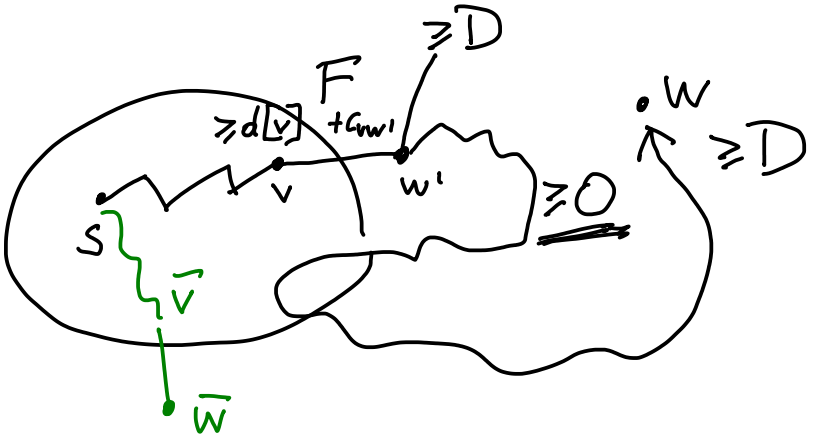
$d[\bar{w}] := d[\bar{v}] + c_{\bar{v}\bar{w}} = D$

$F := F \cup \{\bar{w}\}$

← wenn $\bar{w} = t$: STOP

INVARIANTE: Für jeden Knoten $v \in F$: $d(s,v) = d[v]$ (*)

Für jeden Knoten $w \notin F$: $d(s,w) \geq D$



Für $\bar{w}$ gibt es einen Weg der Länge D.

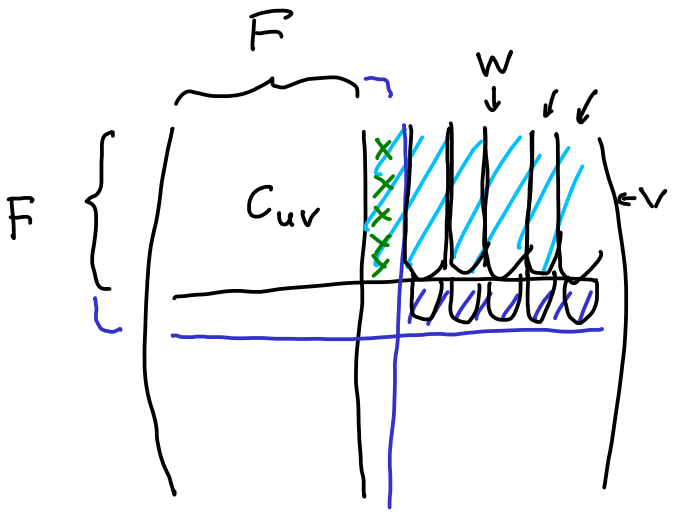
$d(s,\bar{w}) \leq D$ $\Rightarrow$ "$=$"
$\geq$

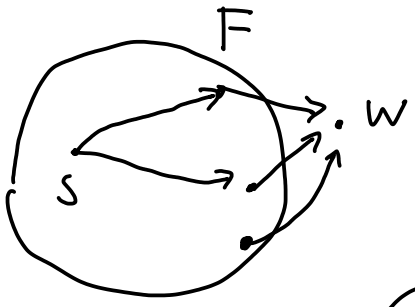
$$\min\{ d[v] + c_{vw} \mid vw \in E, v \in F, w \notin F \} = D$$

$$\min\{ d[v] + c_{vw} \mid vw \in E, v \in F \} =: d[w]$$

$$D = \min\{ d[w] \mid w \notin F \}$$

VORLÄUFIGE MARKIERUNG





falls $\bar{w}w \in E$

$$d_{F \cup \{\bar{w}\}}[w] = \min\{ d_F[w], d[\bar{w}] + c_{\bar{w}w} \}$$

($F \cup \{\bar{w}\}$: $\bar{w} \to u$; $d[\bar{w}]$: $\bar{w} \to u$; $c_{\bar{w}w}$: $\bar{w} \to u$)

Schleife

Bestimme $\min\{ d[v] \mid v \notin F \} = d[u] = D$ (statt $\bar{w}$) — $O(n)$ — $\times n$

← wenn $D = \infty$: STOP

← wenn $u = t$: STOP

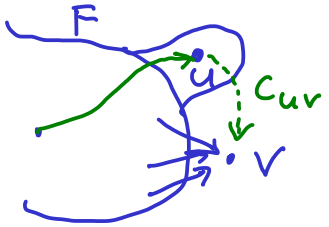
$F := F \cup \{u\}$

for v in $E[u]$:

  if $v \notin F$ and $d[u] + c_{uv} < d[v]$:

    $d[v] := d[u] + c_{uv}$

    $f[v] := u$



INITIALISIERUNG : $F = \emptyset$ $\quad d[s] := 0$

$d[u] := \infty$ für $u \neq s$

Laufzeit : $O(n^2)$ Bestimmen von u und D

$O(m)$ innere Schleife. $m \leq n^2$ } $O(n^2)$

$U \equiv \{ v \notin F \mid d[v] < \infty \}$ „UNFERTIG"

$U := \{s\}$ $\quad d[s] := 0$

while $U \neq \emptyset$:

  bestimme $d[u] = \min \{ d[v] \mid v \in U \}$ ← if $u = t$: break

  $U := U \setminus \{u\}$

  $F := F \cup \{u\}$

  for v in $E[u]$:

    if $v \notin F \cup U$ or ([$v \in U$ and] $d[u] + c_{uv} < d[v]$): (kann man weglassen)

      $d[v] := d[u] + c_{uv}$

      $f[v] := u$

      if $v \notin U$ then $U := U \cup \{v\}$

Menge U mit Schlüsseln $d[u]$

- entferneMin $\leq n$
- einfügen $\leq n$
- decreasekey $\leq m$

} Prioritätswarteschlange (abstrakter Datentyp)

Suchbäume? $O(\log n)$ pro Operation

... $O((m+n)\log n)$.

Halde (heap) $O(\log n)$ pro Operation

... für Dijkstra

Fibonacci-Halde

$O(1)$ decreasekey/einfügen
$O(\log n)$ entfernemin

] AMORTISIERT

... $O(m + n\log n)$ für Dijkstr.