

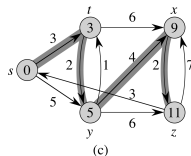
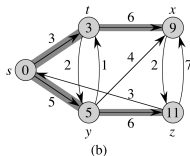
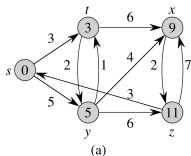
Korteste veje

Korteste veje i vægtede grafer

Længde af sti = sum af vægte af kanter på sti.

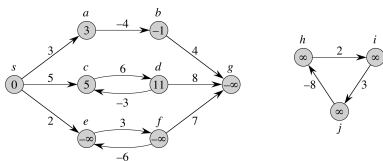
$\delta(u, v)$ = længden af en korteste sti fra u til v . Sættes til ∞ hvis ingen sti findes.

Single-source shortest-path problemet: Givet $s \in V$, find $\delta(s, v)$ (og en konkret sti) for alle $v \in V$.



Korteste veje i vægtede grafer

Bemærk: problemet er ikke veldefineret hvis der findes kredse (som kan nås fra s) med negativ sum:



Omvendt: hvis der ikke findes sådanne negative kredse, kan vi nøjes med at så på simple stier (ingen knuder gentages). Der er et endeligt antal sådanne ($< n^n$, f.eks.), så "længde af korteste sti" er veldefineret.

Relaxation

INIT-SINGLE-SOURCE(G, s)

for each $v \in G.V$

$v.d = \infty$

$v.\pi = \text{NIL}$

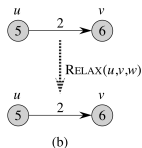
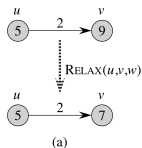
$s.d = 0$

RELAX(u, v, w)

if $v.d > u.d + w(u, v)$

$v.d = u.d + w(u, v)$

$v.\pi = u$



(Bemærk: overvej implementering af “ ∞ ” vs. “ $>$ ”).

Invariant hvis man efter INIT-SINGLE-SOURCE kun ændrer $v.d$ og $v.\pi$ via RELAX:

Hvis $v.d < \infty$ er der en sti fra s til v af længde $v.d$, og denne sti kan gennemløbes (baglæns) ved at følge π -pointere.

Heraf følger, at der altid gælder $\delta(s, v) \leq v.d$, og derfor (da $v.d$ kun kan falde) at hvis $\delta(s, v) = v.d$ på et tidspunkt, vil dette ikke ændres senere.

Bellman-Ford-Moore [1956-57-58]

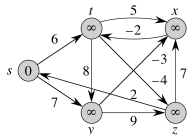
```
BELLMAN-FORD( $G, w, s$ )
  INIT-SINGLE-SOURCE( $G, s$ )
  for  $i = 1$  to  $|G.V| - 1$ 
    for each edge  $(u, v) \in G.E$ 
      RELAX( $u, v, w$ )
  for each edge  $(u, v) \in G.E$ 
    if  $v.d > u.d + w(u, v)$ 
      return FALSE
  return TRUE
```

Køretid: $O(nm)$

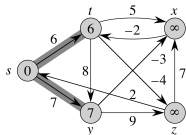
Sætning: Hvis der findes en negativ kreds, som kan nås fra s , svarer algoritmen FALSE. Ellers svarer den TRUE, og $v.d$ og $v.\pi$ er sat korrekt for alle $v \in V$ når den stopper.

Bevis: Invarianten er, at efter k iterationer af første for-løkke passer $v.d$ og $v.\pi$ værdierne for de korteste veje med højst k kanter.

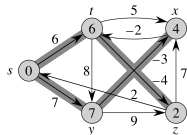
Bellman-Ford, eksempel



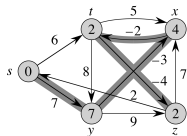
(a)



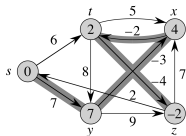
(b)



(c)



(d)



(e)

Dijkstras algoritme [1959]

Grådig algoritme som trinvis opbygger mængde S af knuder med korrekte $v.d$ og $v.\pi$. Bruger en prioritetskø Q . Kræver alle kantvægte ≥ 0 .

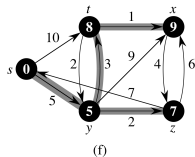
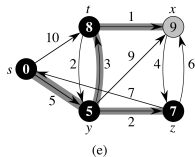
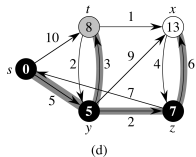
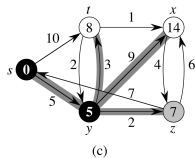
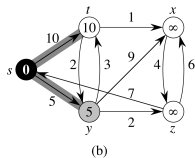
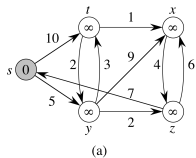
```
DIJKSTRA( $G, w, s$ )
  INIT-SINGLE-SOURCE( $G, s$ )
   $S = \emptyset$ 
   $Q = G.V$  // i.e., insert all vertices into  $Q$ 
  while  $Q \neq \emptyset$ 
     $u = \text{EXTRACT-MIN}(Q)$ 
     $S = S \cup \{u\}$ 
    for each vertex  $v \in G.Adj[u]$ 
      RELAX( $u, v, w$ )
```

Køretid: n INSERT (eller een BUILD-HEAP), n EXTRACT-MIN og m DECREASE-KEY (i RELAX). I alt $O(m \log n)$ hvis prioritetskøen implementeres med en heap.

Sætning: Når når v indlemmes i S er $v.d = v.\pi$ (hvis alle kantvægte er ≥ 0).

Bevis: en modstrid kan opnås ved at se på første knude indlemmet i S for hvilket det ikke gælder.

Dijkstra, eksempel



Algoritme for DAGs [unknown]

Recall: DAG = Directed Acyclic Graph.

Recall: En topologisk sortering kan findes via DFS i tid $O(n + m)$.

DAG-SHORTEST-PATHS(G, w, s)

topologically sort the vertices

INIT-SINGLE-SOURCE(G, s)

for each vertex u , taken in topologically sorted order

for each vertex $v \in G.Adj[u]$

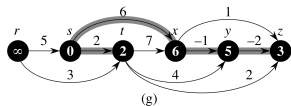
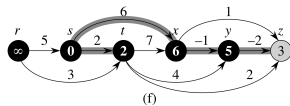
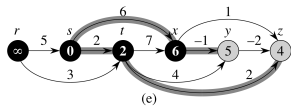
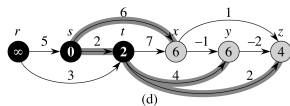
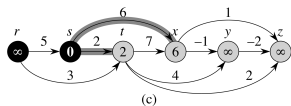
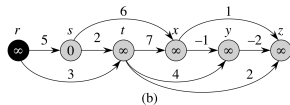
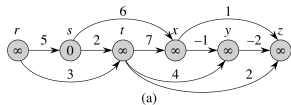
 RELAX(u, v, w)

Køretid: $O(n + m)$.

Sætning: Når algoritmen stopper er $v.d$ og $v.\pi$ sat korrekt for alle $v \in V$.

Bevis: For en knude v med en sti fra s til v : alle knuder på en korteste sti er blevet relaxeret i rækkefølge (hvorved korrekte δ -værdier sættes på denne sti). For alle andre knuder gælder $\infty = \delta(s, v)$ så korrekthed her følger af $\delta(s, v) \leq v.d$.

Algoritmen for DAG, eksempel



Korteste veje i vægtede grafer

All-pairs shortest-path problemet: For alle $s \in V$, find $\delta(s, v)$ (og en konkret sti) for alle $v \in V$.

Een mulighed: køre Dijkstra fra hver source $s \in V$ (kræver ikke-negative vægte): $O(nm \log n)$ tid.

Eller: køre Bellman-Ford-Moore fra hver source $s \in V$ (hvis der er negative vægte): $O(n^2m)$ tid.

En anden mulighed: Floyd-Warshalls algoritme. $O(n^3)$ tid. Klarer negative vægte (men ikke negative kredse).

Bruger adjacency-matrix repræsentationen.

Output også på matrice-form:

$D = (d_{ij})$, $d_{ij} = \delta(v_i, v_j)$ = længden af en korteste sti fra v_i til v_j . Sættes til ∞ hvis ingen sti findes.

$\Pi = (\pi_{ij})$, π_{ij} = sidste knude før v_j på en korteste sti fra knude v_i til knude v_j . Sættes til NIL hvis ingen sti findes.

Floyd-Warshalls algoritme [1962]

Dynamisk programmeringalgoritme (kun konstruktion af D -matricen vises).

FLOYD-WARSHALL(W, n)

$D^{(0)} = W$

for $k = 1$ **to** n

 let $D^{(k)} = (d_{ij}^{(k)})$ be a new $n \times n$ matrix

for $i = 1$ **to** n

for $j = 1$ **to** n

$d_{ij}^{(k)} = \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$

return $D^{(n)}$

Køretid: $O(n^3)$. **Plads:** $O(n^2)$ (kun forrige $D^{(k)}$ matrice behøves gemmes).

Sætning: Når algoritmen stopper er d_{ij} og π_{ij} sat korrekt for alle $v_i, v_j \in V$ (hvis ingen negativ kreds er i grafen).

Bevis: Invarianten er, at $D^{(k)}$ indeholder længden af korteste veje mellem v_i og v_j som passerer knuderne v_1, v_2, \dots, v_k (udover endepunkterne v_i og v_j).

Floyd-Warshall, eksempel

$$D^{(0)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(0)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & \text{NIL} & 4 & \text{NIL} & \text{NIL} \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(1)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & 1 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(2)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(2)} = \begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 1 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(3)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(3)} = \begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(4)} = \begin{pmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \quad \Pi^{(4)} = \begin{pmatrix} \text{NIL} & 1 & 4 & 2 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(5)} = \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \quad \Pi^{(5)} = \begin{pmatrix} \text{NIL} & 3 & 4 & 5 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$