Feasible Potentials and Linear Programming

As a consequence of Ford's Algorithm we get:

Theorem 6.15.

Let $D = (V, A)$ be a digraph, $r, s \in V$, and $c \in \mathbb{R}^A$. If, for every $v \in V$, there exists a least-cost dipath from r to v, then

$$\min\{c(P) \mid P \text{ an } r\text{-}s\text{-dipath}\} = \max\{y_s - y_r \mid y \text{ a feasible potential}\} \ .$$

Formulate the right-hand side as a linear program and consider the dual:

$$\begin{aligned} \max \quad & y_s - y_r \\ \text{s.t.} \quad & y_w - y_v \leq c_{(v,w)} \\ & \text{for all } (v, w) \in A \end{aligned}$$

$$\begin{aligned} \min \quad & c^T \cdot x \\ \text{s.t.} \quad & \sum_{a \in \delta^-(v)} x_a - \sum_{a \in \delta^+(v)} x_a = b_v \quad \forall v \in V \\ & x_a \geq 0 \qquad \text{for all } a \in A \end{aligned}$$

with $b_s = 1$, $b_r = -1$, and $b_v = 0$ for all $v \notin \{r, s\}$.

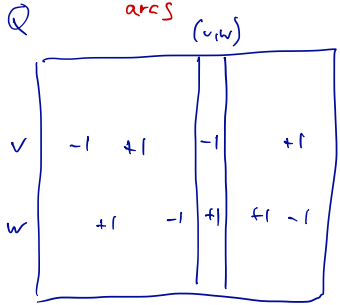
Notice: The dual is the LP relaxation of an ILP formulation of the shortest r-s-dipath problem ($x_a \hat{=}$ number of times a shortest r-s-dipath uses arc a).

Bases of Shortest Path LP

Consider again the dual LP:

$$\begin{aligned} \min \quad & c^T \cdot x \\ \text{s.t.} \quad & \sum_{a \in \delta^-(v)} x_a - \sum_{a \in \delta^+(v)} x_a = b_v \quad \text{for all } v \in V \\ & x_a \geq 0 \qquad \text{for all } a \in A \end{aligned}$$

The underlying matrix Q is the incidence matrix of D.



called the arc-node incident matrix

Bases of Shortest Path LP

Consider again the dual LP:

$$\begin{aligned} \min \quad & c^T \cdot x \\ \text{s.t.} \quad & \sum_{a \in \delta^-(v)} x_a - \sum_{a \in \delta^+(v)} x_a = b_v \quad \text{for all } v \in V \\ & x_a \geq 0 \qquad \text{for all } a \in A \end{aligned}$$

The underlying matrix Q is the incidence matrix of D.

Lemma 6.16.

Let $D = (V, A)$ be a connected digraph and Q its incidence matrix. A subset of columns of Q indexed by a subset of arcs $F \subseteq A$ forms a basis of the linear subspace of $\mathbb{R}^n$ spanned by the columns of Q if and only if F is the arc-set of a spanning tree of D.

Bases of Shortest Path LP

Consider again the dual LP:

$$\begin{aligned} \min \quad & c^T \cdot x \\ \text{s.t.} \quad & \sum_{a \in \delta^-(v)} x_a - \sum_{a \in \delta^+(v)} x_a = b_v \quad \text{for all } v \in V \\ & x_a \geq 0 \qquad \text{for all } a \in A \end{aligned}$$

The underlying matrix Q is the incidence matrix of D.

Lemma 6.16.

Let $D = (V, A)$ be a connected digraph and Q its incidence matrix. A subset of columns of Q indexed by a subset of arcs $F \subseteq A$ forms a basis of the linear subspace of $\mathbb{R}^n$ spanned by the columns of Q if and only if F is the arc-set of a spanning tree of D.

Proof: Exercise. □

Refinement of Ford's Algorithm

Ford's Algorithm

i Set $y_r := 0$, $p(r) := r$, $y_v := \infty$, and $p(v) := \text{null}$, for all $v \in V \setminus \{r\}$.

ii While there is an arc $a = (v, w) \in A$ with $y_w > y_v + c_{(v,w)}$, set

$$y_w := y_v + c_{(v,w)} \quad \text{and} \quad p(w) := v \ .$$

- # iterations crucially depends on order in which arcs are chosen.

Refinement of Ford's Algorithm

Ford's Algorithm

i. Set $y_r := 0$, $p(r) := r$, $y_v := \infty$, and $p(v) := \text{null}$, for all $v \in V \setminus \{r\}$.

ii. While there is an arc $a = (v, w) \in A$ with $y_w > y_v + c_{(v,w)}$, set

$$y_w := y_v + c_{(v,w)} \quad \text{and} \quad p(w) := v \ .$$

- # iterations crucially depends on order in which arcs are chosen.
- Suppose that arcs are chosen in order $\mathcal{S} = f_1, f_2, f_3, \ldots, f_\ell$.

Refinement of Ford's Algorithm

Ford's Algorithm

i. Set $y_r := 0$, $p(r) := r$, $y_v := \infty$, and $p(v) := \text{null}$, for all $v \in V \setminus \{r\}$.

ii. While there is an arc $a = (v, w) \in A$ with $y_w > y_v + c_{(v,w)}$, set

$$y_w := y_v + c_{(v,w)} \quad \text{and} \quad p(w) := v \ .$$

- # iterations crucially depends on order in which arcs are chosen.
- Suppose that arcs are chosen in order $\mathcal{S} = f_1, f_2, f_3, \ldots, f_\ell$.
- Dipath P is embedded in $\mathcal{S}$ if P's arc sequence is a subsequence of $\mathcal{S}$.

Refinement of Ford's Algorithm

Ford's Algorithm

i Set $y_r := 0$, $p(r) := r$, $y_v := \infty$, and $p(v) := \text{null}$, for all $v \in V \setminus \{r\}$.

ii While there is an arc $a = (v, w) \in A$ with $y_w > y_v + c_{(v,w)}$, set

$$y_w := y_v + c_{(v,w)} \quad \text{and} \quad p(w) := v \ .$$

- # iterations crucially depends on order in which arcs are chosen.
- Suppose that arcs are chosen in order $\mathcal{S} = f_1, f_2, f_3, \ldots, f_\ell$.
- Dipath P is embedded in $\mathcal{S}$ if P's arc sequence is a subsequence of $\mathcal{S}$.

Lemma 6.17.

If an r-v-dipath P is embedded in $\mathcal{S}$, then $y_v \leq c(P)$ after Ford's Algorithm has gone through the sequence $\mathcal{S}$.

Refinement of Ford's Algorithm

Ford's Algorithm

i Set $y_r := 0$, $p(r) := r$, $y_v := \infty$, and $p(v) := \text{null}$, for all $v \in V \setminus \{r\}$.

ii While there is an arc $a = (v, w) \in A$ with $y_w > y_v + c_{(v,w)}$, set

$$y_w := y_v + c_{(v,w)} \quad \text{and} \quad p(w) := v \ .$$

- # iterations crucially depends on order in which arcs are chosen.
- Suppose that arcs are chosen in order $\mathcal{S} = f_1, f_2, f_3, \ldots, f_\ell$.
- Dipath P is embedded in $\mathcal{S}$ if P's arc sequence is a subsequence of $\mathcal{S}$.

Lemma 6.17.

If an r-v-dipath P is embedded in $\mathcal{S}$, then $y_v \leq c(P)$ after Ford's Algorithm has gone through the sequence $\mathcal{S}$.

Goal: Find short sequence $\mathcal{S}$ such that, for all $v \in V$, a least-cost r-v-dipath is embedded in $\mathcal{S}$.

Ford-Bellman Algorithm

Basic idea:

- Every simple dipath is embedded in $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_{n-1}$ where, for all i, $\mathcal{S}_i$ is an ordering of A.

Ford-Bellman Algorithm

Basic idea:

- Every simple dipath is embedded in $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_{n-1}$ where, for all i, $\mathcal{S}_i$ is an ordering of A.
- This yields a shortest path algorithm with running time $O(nm)$.

Ford-Bellman Algorithm

Basic idea:

- Every simple dipath is embedded in $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_{n-1}$ where, for all i, $\mathcal{S}_i$ is an ordering of A.
- This yields a shortest path algorithm with running time $O(nm)$.

Ford-Bellman Algorithm

i initialize y, p (see Ford's Algorithm);

ii for $i = 1$ to $n - 1$ do

iii for all $a = (v, w) \in A$ do

iv if $y_w > y_v + c_{(v,w)}$, then set $y_w := y_v + c_{(v,w)}$ and $p(w) := v$;

Ford-Bellman Algorithm

Basic idea:

- Every simple dipath is embedded in $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_{n-1}$ where, for all i, $\mathcal{S}_i$ is an ordering of A.
- This yields a shortest path algorithm with running time $O(nm)$.

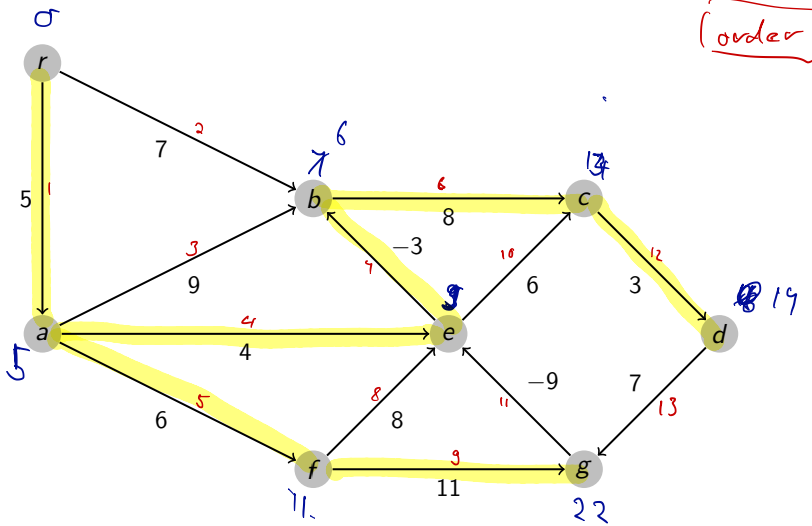
Ford-Bellman Algorithm

i initialize y, p (see Ford's Algorithm);
ii for $i = 1$ to $n - 1$ do
iii for all $a = (v, w) \in A$ do
iv if $y_w > y_v + c_{(v,w)}$, then set $y_w := y_v + c_{(v,w)}$ and $p(w) := v$;

Theorem 6.18.

The algorithm runs in $O(nm)$ time. If, at termination, y is a feasible potential, then p yields a least-cost r-v-dipath for each $v \in V$. Otherwise, the given digraph contains a negative-cost dicircuit. □

Ford-Bellman Example



Acyclic Digraphs and Topological Orderings

Definition 6.19.

Consider a digraph $D = (V, A)$.

a. An ordering $v_1, v_2, \ldots, v_n$ of V so that $i < j$ for each $(v_i, v_j) \in A$ is called a topological ordering.

Acyclic Digraphs and Topological Orderings

Definition 6.19.

Consider a digraph $D = (V, A)$.

a. An ordering $v_1, v_2, \ldots, v_n$ of V so that $i < j$ for each $(v_i, v_j) \in A$ is called a topological ordering.

b. If D has a topological ordering, then D is called acyclic.

Acyclic Digraphs and Topological Orderings

Definition 6.19.

Consider a digraph $D = (V, A)$.

a. An ordering $v_1, v_2, \ldots, v_n$ of V so that $i < j$ for each $(v_i, v_j) \in A$ is called a topological ordering.

b. If D has a topological ordering, then D is called acyclic.

Observations:

- Digraph D is acyclic if and only if it does not contain a dicircuit.

Acyclic Digraphs and Topological Orderings

Definition 6.19.

Consider a digraph $D = (V, A)$.

a. An ordering $v_1, v_2, \ldots, v_n$ of V so that $i < j$ for each $(v_i, v_j) \in A$ is called a topological ordering.

b. If D has a topological ordering, then D is called acyclic.

Observations:

- Digraph D is acyclic if and only if it does not contain a dicircuit.
- Let D be acyclic and $\mathcal{S}$ an ordering of A such that (v_i, v_j) precedes (v_k, v_ℓ) if $i < k$. Then every dipath of D is embedded in $\mathcal{S}$.

Acyclic Digraphs and Topological Orderings

Definition 6.19.

Consider a digraph $D = (V, A)$.

a. An ordering $v_1, v_2, \ldots, v_n$ of V so that $i < j$ for each $(v_i, v_j) \in A$ is called a topological ordering.

b. If D has a topological ordering, then D is called acyclic.

Observations:

- Digraph D is acyclic if and only if it does not contain a dicircuit.
- Let D be acyclic and $\mathcal{S}$ an ordering of A such that (v_i, v_j) precedes (v_k, v_ℓ) if $i < k$. Then every dipath of D is embedded in $\mathcal{S}$.

Theorem 6.20.

The shortest path problem on acyclic digraphs can be solved in time $O(m)$.

Dijkstra's Algorithm

Consider the special case of nonnegative costs, i. e., $c_a \geq 0$, for each $a \in A$.

Dijkstra's Algorithm

Consider the special case of nonnegative costs, i. e., $c_a \geq 0$, for each $a \in A$.

Dijkstra's Algorithm

i initialize y, p (see Ford's Algorithm); set $S := V$;

ii while $S \neq \emptyset$ do

iii choose $v \in S$ with y_v minimum and delete v from S;

iv for each $w \in V$ with $(v, w) \in A$ do

v if $y_w > y_v + c_{(v,w)}$, then set $y_w := y_v + c_{(v,w)}$ and $p(w) := v$;

Dijkstra's Algorithm

Consider the special case of nonnegative costs, i. e., $c_a \geq 0$, for each $a \in A$.

Dijkstra's Algorithm

i initialize y, p (see Ford's Algorithm); set $S := V$;

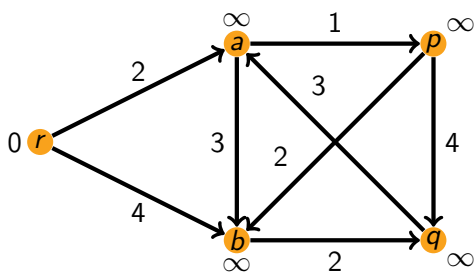
ii while $S \neq \emptyset$ do

iii choose $v \in S$ with y_v minimum and delete v from S;

iv for each $w \in V$ with $(v, w) \in A$ do

v if $y_w > y_v + c_{(v,w)}$, then set $y_w := y_v + c_{(v,w)}$ and $p(w) := v$;

Example:



Dijkstra's Algorithm

Consider the special case of nonnegative costs, i. e., $c_a \geq 0$, for each $a \in A$.

Dijkstra's Algorithm

i initialize y, p (see Ford's Algorithm); set $S := V$;

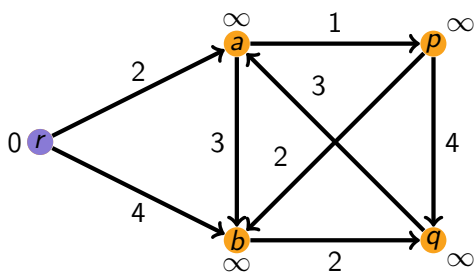
ii while $S \neq \emptyset$ do

iii choose $v \in S$ with y_v minimum and delete v from S;

iv for each $w \in V$ with $(v, w) \in A$ do

v if $y_w > y_v + c_{(v,w)}$, then set $y_w := y_v + c_{(v,w)}$ and $p(w) := v$;

Example:



Dijkstra's Algorithm

Consider the special case of nonnegative costs, i. e., $c_a \geq 0$, for each $a \in A$.

Dijkstra's Algorithm

i initialize y, p (see Ford's Algorithm); set $S := V$;

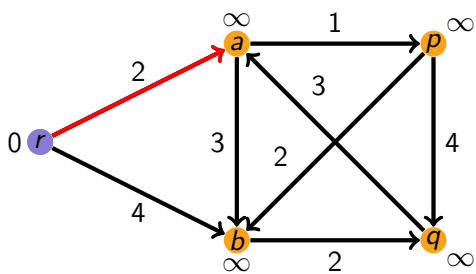
ii while $S \neq \emptyset$ do

iii choose $v \in S$ with y_v minimum and delete v from S;

iv for each $w \in V$ with $(v, w) \in A$ do

v if $y_w > y_v + c_{(v,w)}$, then set $y_w := y_v + c_{(v,w)}$ and $p(w) := v$;

Example:



Dijkstra's Algorithm

Consider the special case of nonnegative costs, i. e., $c_a \geq 0$, for each $a \in A$.

Dijkstra's Algorithm

i initialize y, p (see Ford's Algorithm); set $S := V$;

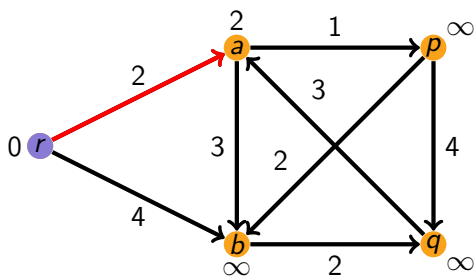
ii while $S \neq \emptyset$ do

iii choose $v \in S$ with y_v minimum and delete v from S;

iv for each $w \in V$ with $(v, w) \in A$ do

v if $y_w > y_v + c_{(v,w)}$, then set $y_w := y_v + c_{(v,w)}$ and $p(w) := v$;

Example:



Dijkstra's Algorithm

Consider the special case of nonnegative costs, i. e., $c_a \geq 0$, for each $a \in A$.

Dijkstra's Algorithm

i initialize y, p (see Ford's Algorithm); set $S := V$;

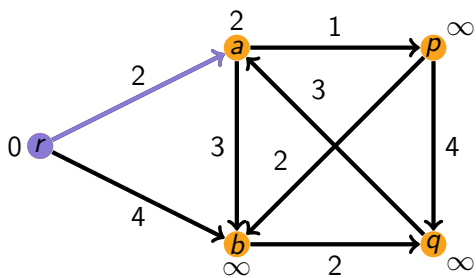
ii while $S \neq \emptyset$ do

iii choose $v \in S$ with y_v minimum and delete v from S;

iv for each $w \in V$ with $(v, w) \in A$ do

v if $y_w > y_v + c_{(v,w)}$, then set $y_w := y_v + c_{(v,w)}$ and $p(w) := v$;

Example:



Dijkstra's Algorithm

Consider the special case of nonnegative costs, i. e., $c_a \geq 0$, for each $a \in A$.

Dijkstra's Algorithm

i initialize y, p (see Ford's Algorithm); set $S := V$;

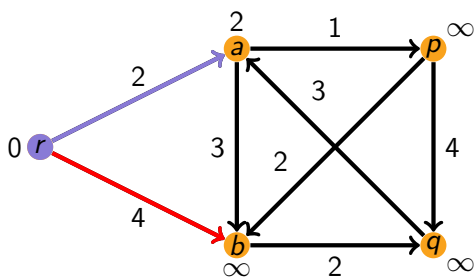
ii while $S \neq \emptyset$ do

iii choose $v \in S$ with y_v minimum and delete v from S;

iv for each $w \in V$ with $(v, w) \in A$ do

v if $y_w > y_v + c_{(v,w)}$, then set $y_w := y_v + c_{(v,w)}$ and $p(w) := v$;

Example:



Dijkstra's Algorithm

Consider the special case of nonnegative costs, i. e., $c_a \geq 0$, for each $a \in A$.

Dijkstra's Algorithm

i initialize y, p (see Ford's Algorithm); set $S := V$;

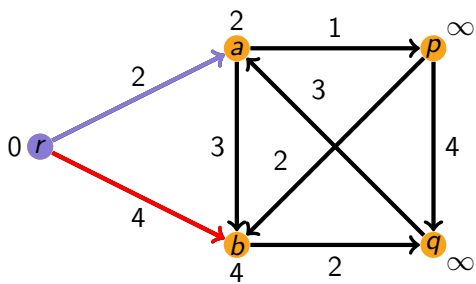
ii while $S \neq \emptyset$ do

iii choose $v \in S$ with y_v minimum and delete v from S;

iv for each $w \in V$ with $(v, w) \in A$ do

v if $y_w > y_v + c_{(v,w)}$, then set $y_w := y_v + c_{(v,w)}$ and $p(w) := v$;

Example:



Dijkstra's Algorithm

Consider the special case of nonnegative costs, i. e., $c_a \geq 0$, for each $a \in A$.

Dijkstra's Algorithm

i initialize y, p (see Ford's Algorithm); set $S := V$;

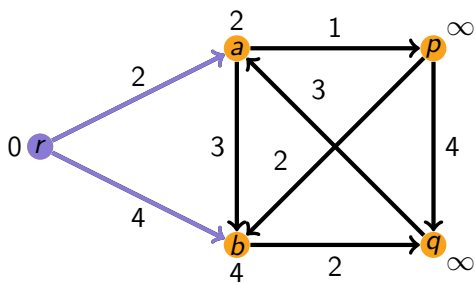
ii while $S \neq \emptyset$ do

iii choose $v \in S$ with y_v minimum and delete v from S;

iv for each $w \in V$ with $(v, w) \in A$ do

v if $y_w > y_v + c_{(v,w)}$, then set $y_w := y_v + c_{(v,w)}$ and $p(w) := v$;

Example:



Dijkstra's Algorithm

Consider the special case of nonnegative costs, i. e., $c_a \geq 0$, for each $a \in A$.

Dijkstra's Algorithm

i initialize y, p (see Ford's Algorithm); set $S := V$;

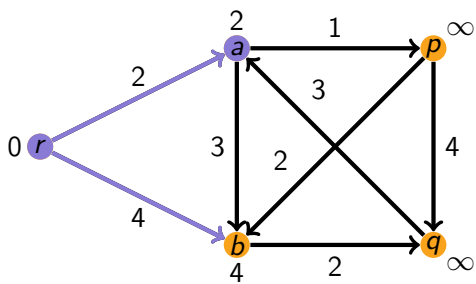
ii while $S \neq \emptyset$ do

iii choose $v \in S$ with y_v minimum and delete v from S;

iv for each $w \in V$ with $(v, w) \in A$ do

v if $y_w > y_v + c_{(v,w)}$, then set $y_w := y_v + c_{(v,w)}$ and $p(w) := v$;

Example:



Dijkstra's Algorithm

Consider the special case of nonnegative costs, i. e., $c_a \geq 0$, for each $a \in A$.

Dijkstra's Algorithm

i initialize y, p (see Ford's Algorithm); set $S := V$;

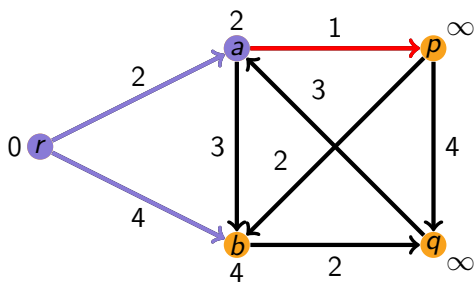
ii while $S \neq \emptyset$ do

iii choose $v \in S$ with y_v minimum and delete v from S;

iv for each $w \in V$ with $(v, w) \in A$ do

v if $y_w > y_v + c_{(v,w)}$, then set $y_w := y_v + c_{(v,w)}$ and $p(w) := v$;

Example:



Dijkstra's Algorithm

Consider the special case of nonnegative costs, i. e., $c_a \geq 0$, for each $a \in A$.

Dijkstra's Algorithm

i initialize y, p (see Ford's Algorithm); set $S := V$;

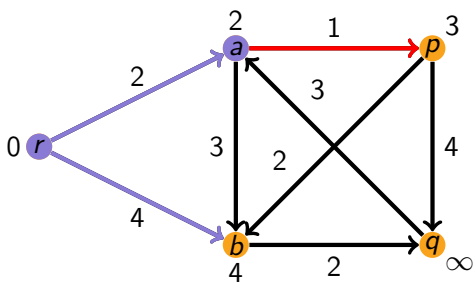
ii while $S \neq \emptyset$ do

iii choose $v \in S$ with y_v minimum and delete v from S;

iv for each $w \in V$ with $(v, w) \in A$ do

v if $y_w > y_v + c_{(v,w)}$, then set $y_w := y_v + c_{(v,w)}$ and $p(w) := v$;

Example:



Dijkstra's Algorithm

Consider the special case of nonnegative costs, i. e., $c_a \geq 0$, for each $a \in A$.

Dijkstra's Algorithm

i initialize y, p (see Ford's Algorithm); set $S := V$;

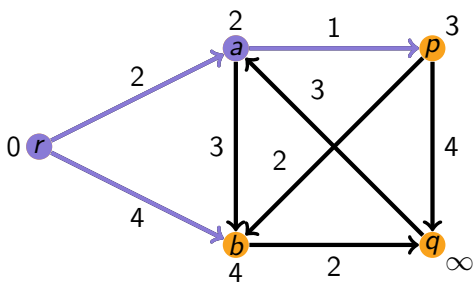
ii while $S \neq \emptyset$ do

iii choose $v \in S$ with y_v minimum and delete v from S;

iv for each $w \in V$ with $(v, w) \in A$ do

v if $y_w > y_v + c_{(v,w)}$, then set $y_w := y_v + c_{(v,w)}$ and $p(w) := v$;

Example:



Dijkstra's Algorithm

Consider the special case of nonnegative costs, i. e., $c_a \geq 0$, for each $a \in A$.

Dijkstra's Algorithm

i initialize y, p (see Ford's Algorithm); set $S := V$;

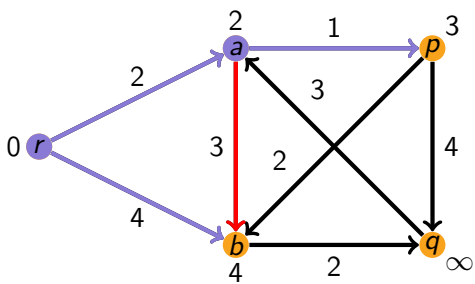
ii while $S \neq \emptyset$ do

iii choose $v \in S$ with y_v minimum and delete v from S;

iv for each $w \in V$ with $(v, w) \in A$ do

v if $y_w > y_v + c_{(v,w)}$, then set $y_w := y_v + c_{(v,w)}$ and $p(w) := v$;

Example:



Dijkstra's Algorithm

Consider the special case of nonnegative costs, i. e., $c_a \geq 0$, for each $a \in A$.

Dijkstra's Algorithm

i initialize y, p (see Ford's Algorithm); set $S := V$;

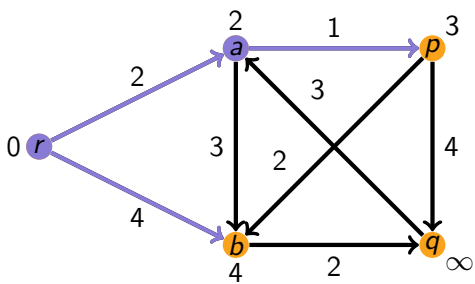
ii while $S \neq \emptyset$ do

iii choose $v \in S$ with y_v minimum and delete v from S;

iv for each $w \in V$ with $(v, w) \in A$ do

v if $y_w > y_v + c_{(v,w)}$, then set $y_w := y_v + c_{(v,w)}$ and $p(w) := v$;

Example:



Dijkstra's Algorithm

Consider the special case of nonnegative costs, i. e., $c_a \geq 0$, for each $a \in A$.

Dijkstra's Algorithm

i initialize y, p (see Ford's Algorithm); set $S := V$;

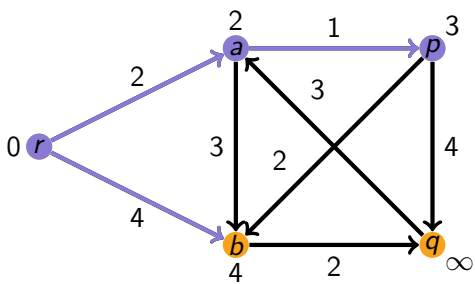
ii while $S \neq \emptyset$ do

iii choose $v \in S$ with y_v minimum and delete v from S;

iv for each $w \in V$ with $(v, w) \in A$ do

v if $y_w > y_v + c_{(v,w)}$, then set $y_w := y_v + c_{(v,w)}$ and $p(w) := v$;

Example:



Dijkstra's Algorithm

Consider the special case of nonnegative costs, i. e., $c_a \geq 0$, for each $a \in A$.

Dijkstra's Algorithm

i initialize y, p (see Ford's Algorithm); set $S := V$;

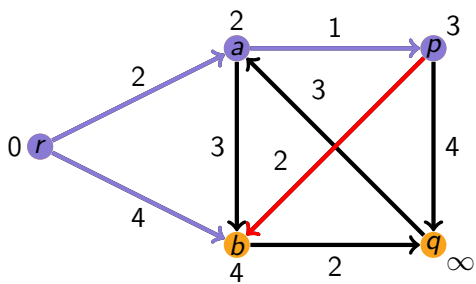
ii while $S \neq \emptyset$ do

iii choose $v \in S$ with y_v minimum and delete v from S;

iv for each $w \in V$ with $(v, w) \in A$ do

v if $y_w > y_v + c_{(v,w)}$, then set $y_w := y_v + c_{(v,w)}$ and $p(w) := v$;

Example:



Dijkstra's Algorithm

Consider the special case of nonnegative costs, i. e., $c_a \geq 0$, for each $a \in A$.

Dijkstra's Algorithm

i initialize y, p (see Ford's Algorithm); set $S := V$;

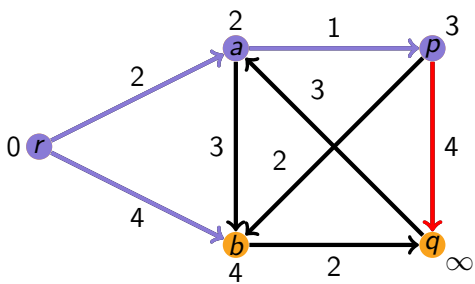
ii while $S \neq \emptyset$ do

iii choose $v \in S$ with y_v minimum and delete v from S;

iv for each $w \in V$ with $(v, w) \in A$ do

v if $y_w > y_v + c_{(v,w)}$, then set $y_w := y_v + c_{(v,w)}$ and $p(w) := v$;

Example:



Dijkstra's Algorithm

Consider the special case of nonnegative costs, i. e., $c_a \geq 0$, for each $a \in A$.

Dijkstra's Algorithm

i initialize y, p (see Ford's Algorithm); set $S := V$;

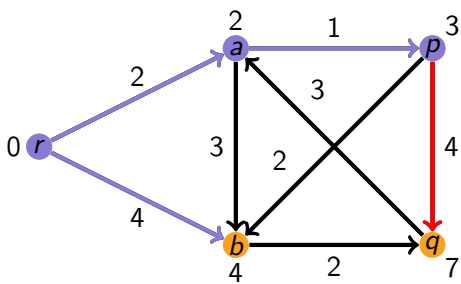
ii while $S \neq \emptyset$ do

iii choose $v \in S$ with y_v minimum and delete v from S;

iv for each $w \in V$ with $(v, w) \in A$ do

v if $y_w > y_v + c_{(v,w)}$, then set $y_w := y_v + c_{(v,w)}$ and $p(w) := v$;

Example:



Dijkstra's Algorithm

Consider the special case of nonnegative costs, i. e., $c_a \geq 0$, for each $a \in A$.

Dijkstra's Algorithm

i initialize y, p (see Ford's Algorithm); set $S := V$;

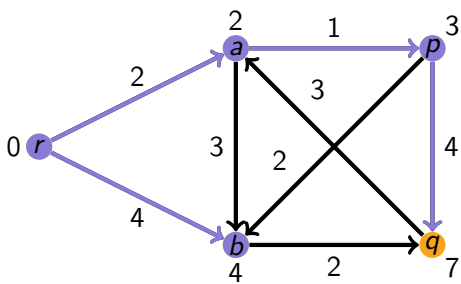
ii while $S \neq \emptyset$ do

iii choose $v \in S$ with y_v minimum and delete v from S;

iv for each $w \in V$ with $(v, w) \in A$ do

v if $y_w > y_v + c_{(v,w)}$, then set $y_w := y_v + c_{(v,w)}$ and $p(w) := v$;

Example:



Dijkstra's Algorithm

Consider the special case of nonnegative costs, i. e., $c_a \geq 0$, for each $a \in A$.

Dijkstra's Algorithm

i initialize y, p (see Ford's Algorithm); set $S := V$;

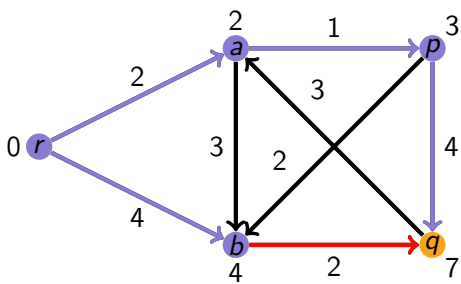
ii while $S \neq \emptyset$ do

iii choose $v \in S$ with y_v minimum and delete v from S;

iv for each $w \in V$ with $(v, w) \in A$ do

v if $y_w > y_v + c_{(v,w)}$, then set $y_w := y_v + c_{(v,w)}$ and $p(w) := v$;

Example:



Dijkstra's Algorithm

Consider the special case of nonnegative costs, i. e., $c_a \geq 0$, for each $a \in A$.

Dijkstra's Algorithm

i initialize y, p (see Ford's Algorithm); set $S := V$;

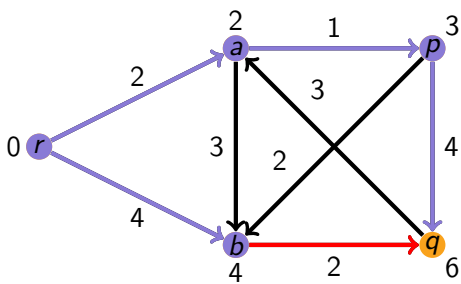
ii while $S \neq \emptyset$ do

iii choose $v \in S$ with y_v minimum and delete v from S;

iv for each $w \in V$ with $(v, w) \in A$ do

v if $y_w > y_v + c_{(v,w)}$, then set $y_w := y_v + c_{(v,w)}$ and $p(w) := v$;

Example:



Dijkstra's Algorithm

Consider the special case of nonnegative costs, i. e., $c_a \geq 0$, for each $a \in A$.

Dijkstra's Algorithm

i initialize y, p (see Ford's Algorithm); set $S := V$;

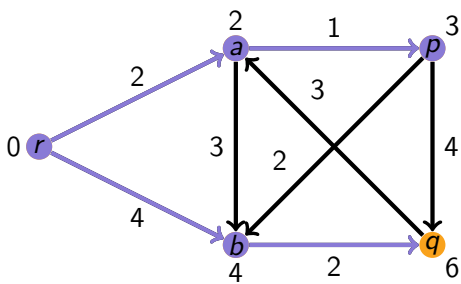
ii while $S \neq \emptyset$ do

iii choose $v \in S$ with y_v minimum and delete v from S;

iv for each $w \in V$ with $(v, w) \in A$ do

v if $y_w > y_v + c_{(v,w)}$, then set $y_w := y_v + c_{(v,w)}$ and $p(w) := v$;

Example:



Dijkstra's Algorithm

Consider the special case of nonnegative costs, i. e., $c_a \geq 0$, for each $a \in A$.

Dijkstra's Algorithm

i initialize y, p (see Ford's Algorithm); set $S := V$;

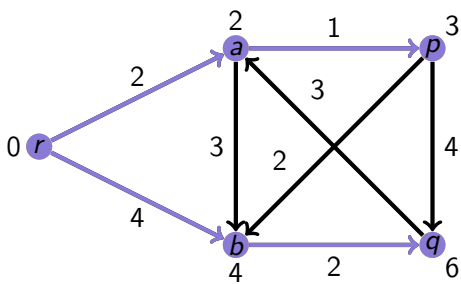
ii while $S \neq \emptyset$ do

iii choose $v \in S$ with y_v minimum and delete v from S;

iv for each $w \in V$ with $(v, w) \in A$ do

v if $y_w > y_v + c_{(v,w)}$, then set $y_w := y_v + c_{(v,w)}$ and $p(w) := v$;

Example:



Dijkstra's Algorithm

Consider the special case of nonnegative costs, i. e., $c_a \geq 0$, for each $a \in A$.

Dijkstra's Algorithm

i initialize y, p (see Ford's Algorithm); set $S := V$;

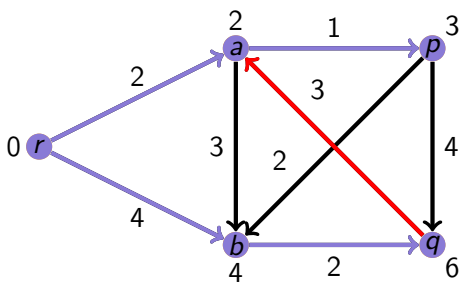
ii while $S \neq \emptyset$ do

iii choose $v \in S$ with y_v minimum and delete v from S;

iv for each $w \in V$ with $(v, w) \in A$ do

v if $y_w > y_v + c_{(v,w)}$, then set $y_w := y_v + c_{(v,w)}$ and $p(w) := v$;

Example:



Dijkstra's Algorithm

Consider the special case of nonnegative costs, i. e., $c_a \geq 0$, for each $a \in A$.

Dijkstra's Algorithm

i initialize y, p (see Ford's Algorithm); set $S := V$;

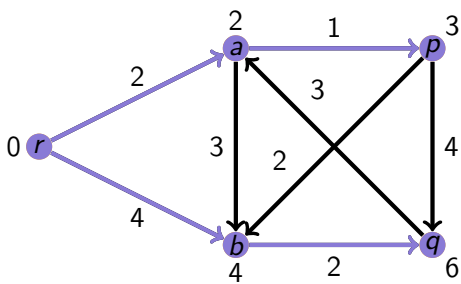
ii while $S \neq \emptyset$ do

iii choose $v \in S$ with y_v minimum and delete v from S;

iv for each $w \in V$ with $(v, w) \in A$ do

v if $y_w > y_v + c_{(v,w)}$, then set $y_w := y_v + c_{(v,w)}$ and $p(w) := v$;

Example:



Correctness of Dijkstra's Algorithm

Lemma 6.21.

For each $w \in V$, let y'_w be the value of y_w when w is removed from S.
If u is deleted from S before v, then $y'_u \leq y'_v$.

Correctness of Dijkstra's Algorithm

Lemma 6.21.

For each $w \in V$, let y'_w be the value of y_w when w is removed from S.
If u is deleted from S before v, then $y'_u \leq y'_v$.

Theorem 6.22.

If $c \geq 0$, then Dijkstra's Algorithm solves the shortest paths problem correctly in time $O(n^2)$. A heap-based implementation yields running time $O(m \log n)$.

Correctness of Dijkstra's Algorithm

Lemma 6.21.

For each $w \in V$, let y'_w be the value of y_w when w is removed from S.
If u is deleted from S before v, then $y'_u \leq y'_v$.

Theorem 6.22.

If $c \geq 0$, then Dijkstra's Algorithm solves the shortest paths problem correctly in time $O(n^2)$. A heap-based implementation yields running time $O(m \log n)$.

Remark: The for-loop in Dijkstra's Algorithm (step iv) can be modified such that only arcs (v, w) with $w \in S$ are considered.