Shortest Paths

- **Input: weighted, directed graph $G = (V, E)$, with weight function $w : E \to \mathbf{R}$.**
- **The weight of path $p =< v_0, v_1, \ldots, v_k >$ is the sum of the weights of its constituent edges:**

$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i) \ .$$

- **The shortest-path weight from u to v is**

$$\delta(u, v) = \begin{cases} \min\{w(p)\} & \text{if there is a path } p \text{ from } u \text{ to } v \ , \\ \infty & \text{otherwise .} \end{cases}$$

- **A shortest path from vertex u to vertex v is then defined as any path p with weight $w(p) = \delta(u, v)$.**

Shortest Paths

- **Input: weighted, directed graph $G = (V, E)$, with weight function $w : E \to \mathbf{R}$.**
- **The weight of path $p =< v_0, v_1, \ldots, v_k >$ is the sum of the weights of its constituent edges:**

$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i) \ .$$

- **The shortest-path weight from u to v is**

$$\delta(u, v) = \begin{cases} \min\{w(p)\} & \text{if there is a path } p \text{ from } u \text{ to } v \ , \\ \infty & \text{otherwise .} \end{cases}$$

- **A shortest path from vertex u to vertex v is then defined as any path p with weight $w(p) = \delta(u, v)$.**

All Pairs Shortest Paths

- **Input: weighted, directed graph $G = (V, E)$, with weight function $w : E \to \mathbf{R}$.**
- **The weight of path $p =< v_0, v_1, \ldots, v_k >$ is the sum of the weights of its constituent edges:**

$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i) \ .$$

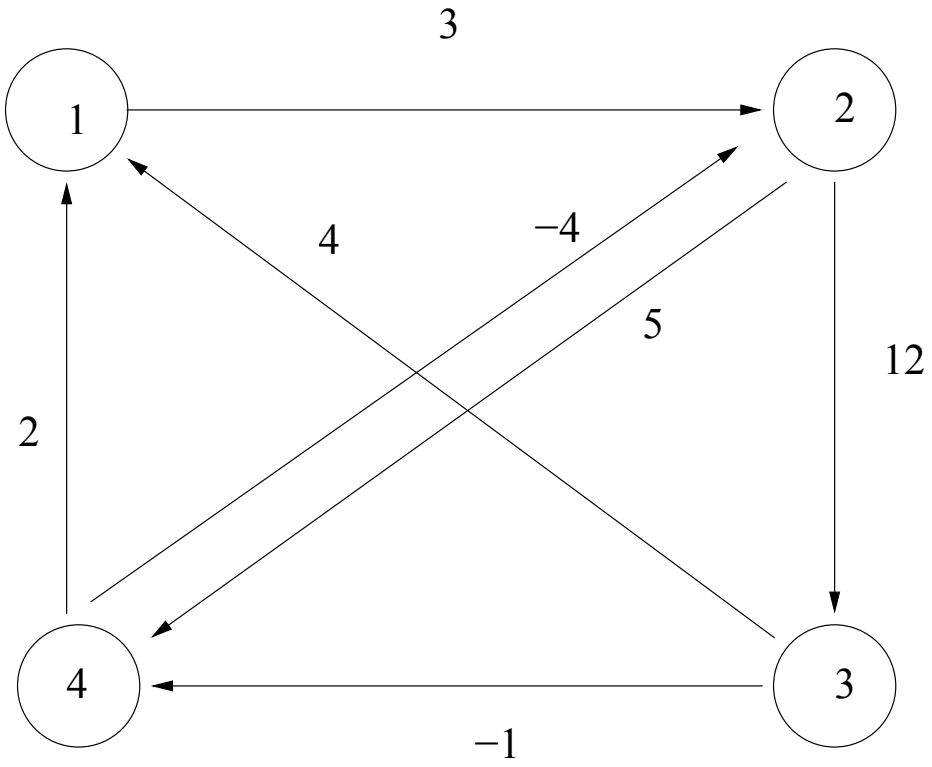
- **The shortest-path weight from u to v is**

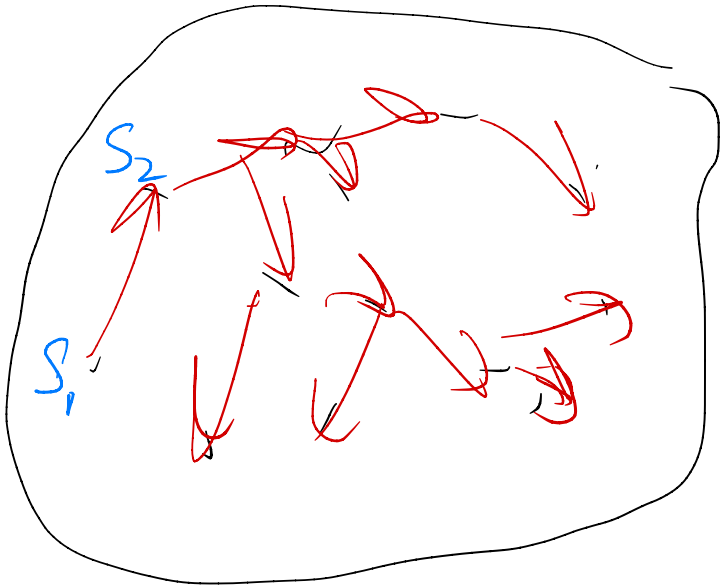
$$\delta(u, v) = \begin{cases} \min\{w(p)\} & \text{if there is a path } p \text{ from } u \text{ to } v \ , \\ \infty & \text{otherwise .} \end{cases}$$

- **A shortest path from vertex u to vertex v is then defined as any path p with weight $w(p) = \delta(u, v)$.**

All Pairs Shortest Paths: Compute $d(u, v)$ the shortest path distance from u to v for all pairs of vertices u and v.

Example





Solution

$$\begin{pmatrix} 0 & 3 & 15 & 8 \\ 7 & 0 & 12 & 5 \\ 1 & -5 & 0 & -1 \\ 2 & -4 & 8 & 0 \end{pmatrix}$$

Approach 1

Run Single source shortest paths V times

- $O(V^2E)$ **for general graphs**
- $O(VE + V^2 \log V)$ **for graphs with non-negative edge weights**

Other approaches : Share information between the various computations

Floyd-Warshall, Dynamic Programming

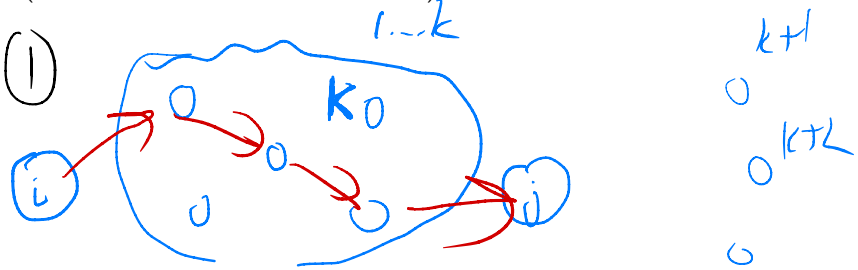
- **Let $d_{ij}^{(k)}$ be the weight of a shortest path from vertex i to vertex j for which all intermediate vertices are in the set $\{1, 2, \ldots, k\}$.**
- **When $k = 0$, a path from vertex i to vertex j with no intermediate vertex numbered higher than 0 has no intermediate vertices at all, hence $d_{ij}^{(0)} = w_{ij}$.**

don't use k use k

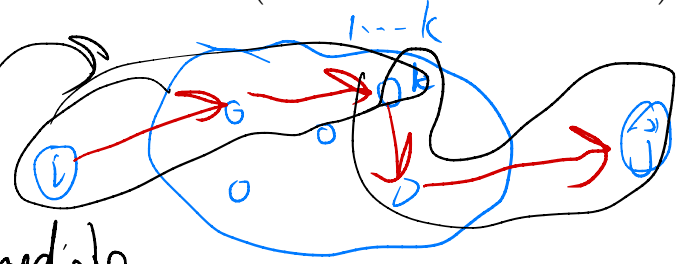
$$d_{ij}^{(k)} = \begin{cases} w_{ij} & \textbf{if } k = 0 \ , \\ \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right) & \textbf{if } k \geq 1 \ . \end{cases} \tag{1}$$

Floyd-Warshall(W)

```
1  n ← rows[W]
2  D^(0) ← W
3  for k ← 1 to n
4        do for i ← 1 to n
5              do for j ← 1 to n
6                    do d_ij^(k) ← min(d_ij^(k-1), d_ik^(k-1) + d_kj^(k-1))
7  return D^(n)
```

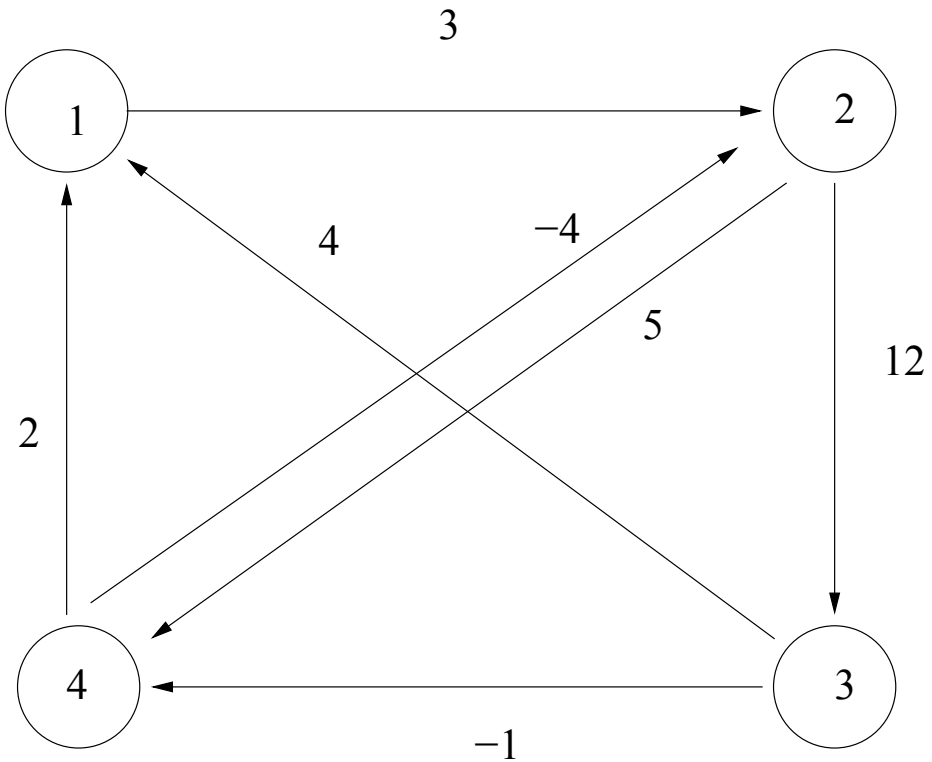


Running time $O(V^3)$



shortest path from i to k using 1..k-1 as intermediate

Example



$$D^0 = \begin{pmatrix} 0 & 3 & \infty & \infty \\ \infty & 0 & 12 & 5 \\ 4 & \infty & 0 & -1 \\ 2 & -4 & \infty & 0 \end{pmatrix} \quad D^1 = \begin{pmatrix} 0 & 3 & \infty & \infty \\ \infty & 0 & 12 & 5 \\ 4 & 7 & 0 & -1 \\ 2 & -4 & \infty & 0 \end{pmatrix} \quad D^2 = \begin{pmatrix} 0 & 3 & 15 & 8 \\ \infty & 0 & 12 & 5 \\ 4 & 7 & 0 & -1 \\ 2 & -4 & 8 & 0 \end{pmatrix}$$

$$D^3 = \begin{pmatrix} 0 & 3 & 15 & 8 \\ 16 & 0 & 12 & 5 \\ 4 & 7 & 0 & -1 \\ 2 & -4 & 8 & 0 \end{pmatrix} \quad D^4 = \begin{pmatrix} 0 & 3 & 15 & 8 \\ 7 & 0 & 12 & 5 \\ 1 & -5 & 0 & -1 \\ 2 & -4 & 8 & 0 \end{pmatrix}$$

Another Algorithm

RESET ALL DEFINITIONS OF D.

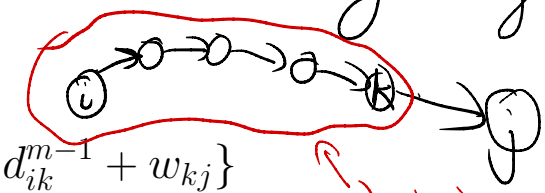
- **Let w_{ij} be the length of edge ij**
- **Let $w_{ii} = 0$**
- **Let d_{ij}^m be the shortest path from i to j using m or fewer edges**

1) uses $< m$ edges

2) use exactly m edges

$$d_{ij}^1 = w_{ij}$$

$$d_{ij}^m = \min\{d_{ij}^{m-1}, \min_{1 \le k \le n, k \ne j} d_{ik}^{m-1} + w_{kj}\}$$



shortest path from i to k using $m-1$ or fewer edges

d_{ik}^{m-1}

Combining these two, we get

$$d_{ij}^m = \min_{1 \le k \le n} \{d_{ik}^{m-1} + w_{kj}\}$$

This would give an $O(V^4)$ algorithm

loops m i j k

D_{ij}^m

Using matrix multiplication analogy

Note the similarity of

$$d_{ij}^m = \min_{1 \le k \le n} \{d_{ik}^{m-1} + w_{kj}\}$$

with matrix multiplication:

$$c_{ij} = \mathbf{sum}_{1 \le k \le n} \{a_{ik} \cdot b_{kj}\}$$

Make the following substitutions (which have the right algebraic properties:

$$\begin{aligned} \mathbf{sum} &\rightarrow \min \\ a_{ik} &\rightarrow d_{ik}^{m-1} \\ \cdot &\rightarrow + \\ b_{kj} &\rightarrow w_{kj} \\ c &\rightarrow d^m \end{aligned}$$

Using this matrix multiplication terminology, we have

$$\begin{array}{lll} D^1 & & = W \\ D^2 & = D^1 \cdot W & = W^2 \\ D^3 & = D^2 \cdot W & = W^3 \\ \cdots & \cdots & \cdots \\ D^m & = D^{m-1} W & = W^m \end{array}$$

But we can execute W^m be repeated squaring and get $O(V^3 \log V)$ time.

W^m

$W^{27} = W^{16} \cdot W^8 \cdot W^2 \cdot W^1$

W

$W^2 = W \cdot W$

$W^4 = W^2 \cdot W^2$

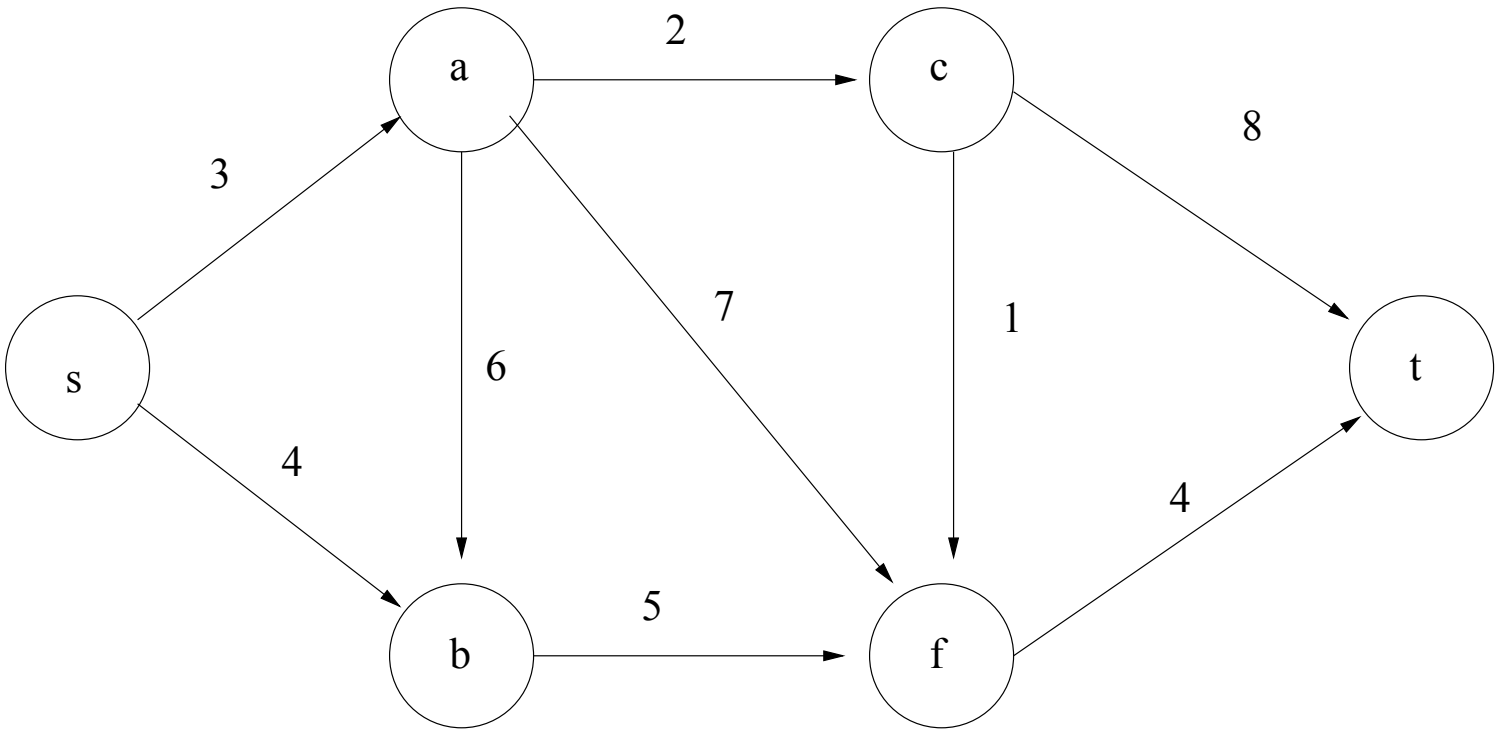
$W^8 = W^4 \cdot W^4$

$W^{16} = W^8 \cdot W^8$

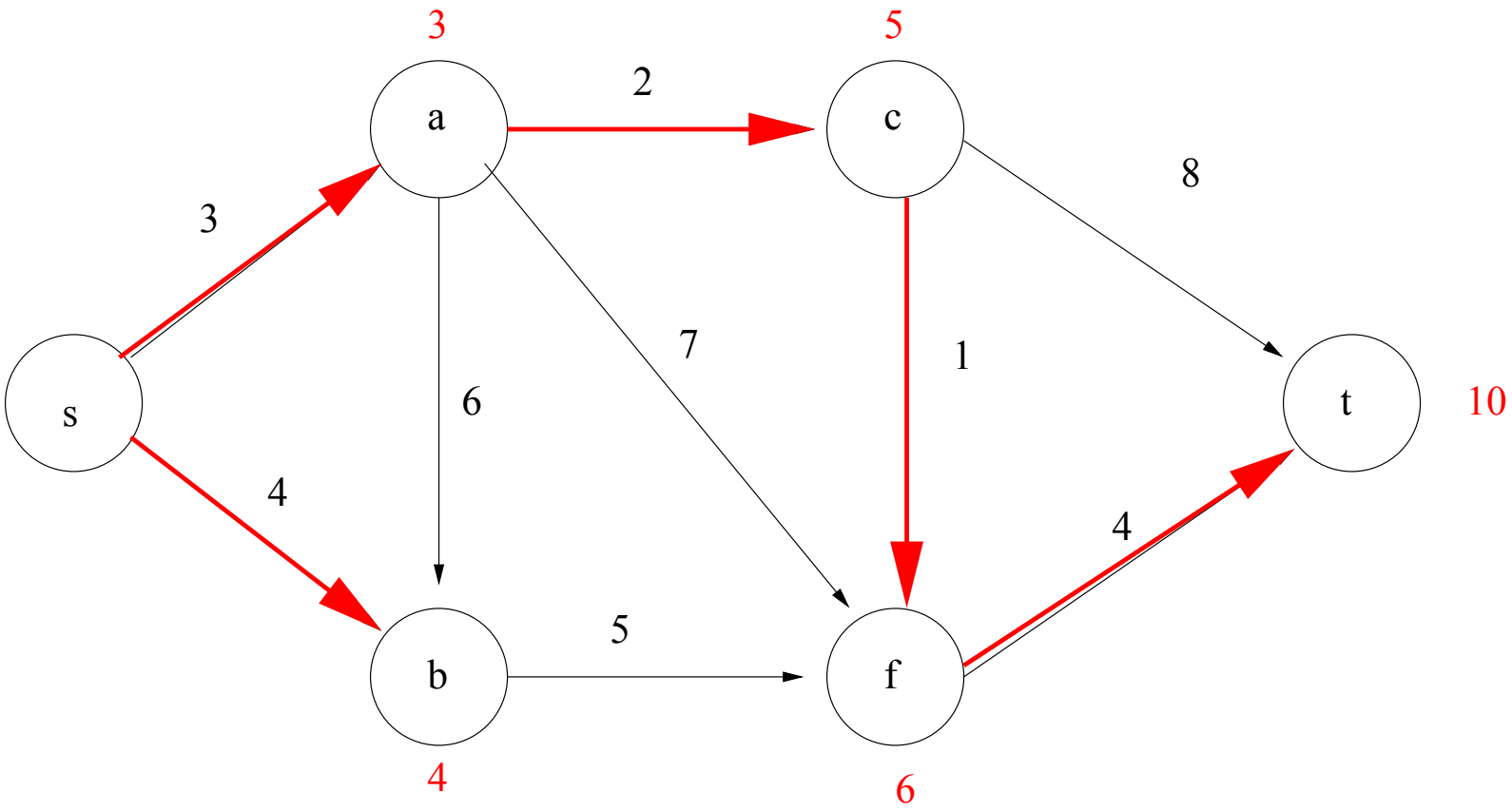
$W^{27} = W^{16} \cdot W^8 \cdot W^2 \cdot W^1$

$O(\lg m)$ matrix mult.

Example



Solution



Shortest Paths

Shortest Path Variants

- **Single Source-Single Sink**
- **Single Source (all destinations from a source s)**
- **All Pairs**

Defs:

- **Let $\delta(v)$ be the real shortest path distance from s to v**
- **Let $d(v)$ be a value computed by an algorithm**

Edge Weights

- **All non-negative**
- **Arbitrary**

Note: **Must have no negative cost cycles**

Single Source Shortest Paths

Key Property: Subpaths of shortest paths are shortest paths **Given a weighted, directed graph $G = (V, E)$ with weight function $w : E \rightarrow \mathbf{R}$, let $p =< v_1, v_2, \ldots, v_k >$ be a shortest path from vertex v_1 to vertex v_k and, for any i and j such that $1 \leq i \leq j \leq k$, let $p_{ij} =< v_i, v_{i+1}, \ldots, v_j >$ be the subpath of p from vertex v_i to vertex v_j. Then, p_{ij} is a shortest path from v_i to v_j.**

Note: this is optimal substructure

Corollary 1 **For all edges $(u, v) \in E$,**

$$\delta(v) \leq \delta(u) + w(u, v)$$

Corollary 2 **Shortest paths follow a tree of edges for which**

$$\delta(v) = \delta(u) + w(u, v)$$

More precisely, any edge in a shortest path must satisfy

$$\delta(v) = \delta(u) + w(u, v)$$

Relaxation

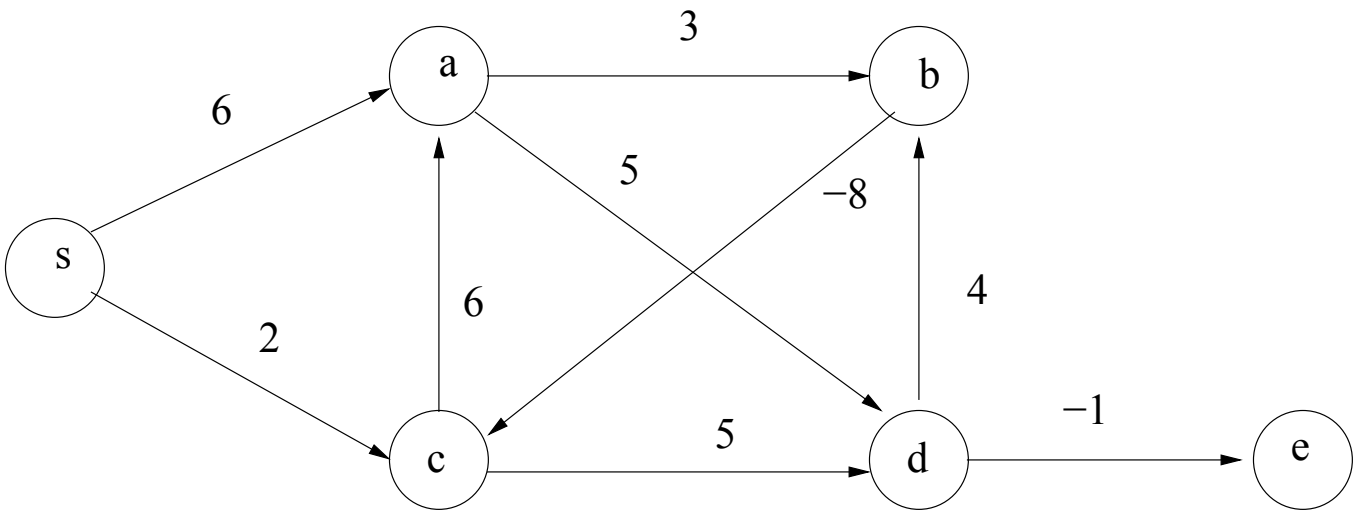
Relax(u, v, w)

1 **if** $d[v] > d[u] + w(u, v)$
2 **then** $d[v] \leftarrow d[u] + w(u, v)$
3 $\pi[v] \leftarrow u$ **(keep track of actual path)**

Lemma: **Assume that we initialize all $d(v)$ to ∞, $d(s) = 0$ and execute a series of Relax operations. Then for all v, $d(v) \geq \delta(v)$.**

Lemma: **Let $P = e_1, \ldots, e_k$ be a shortest path from s to v. After initialization, suppose that we relax the edges of P in order (but not necessarily consecutively). Then $d(v) = \delta(v)$.**

Example



Algorithms

Goal of an algorithm: **Relax the edges in a shortest path in order (but not necessarily consecutively).**

Algorithms

Goal of an algorithm: **Relax the edges in a shortest path in order (but not necessarily consecutively).**

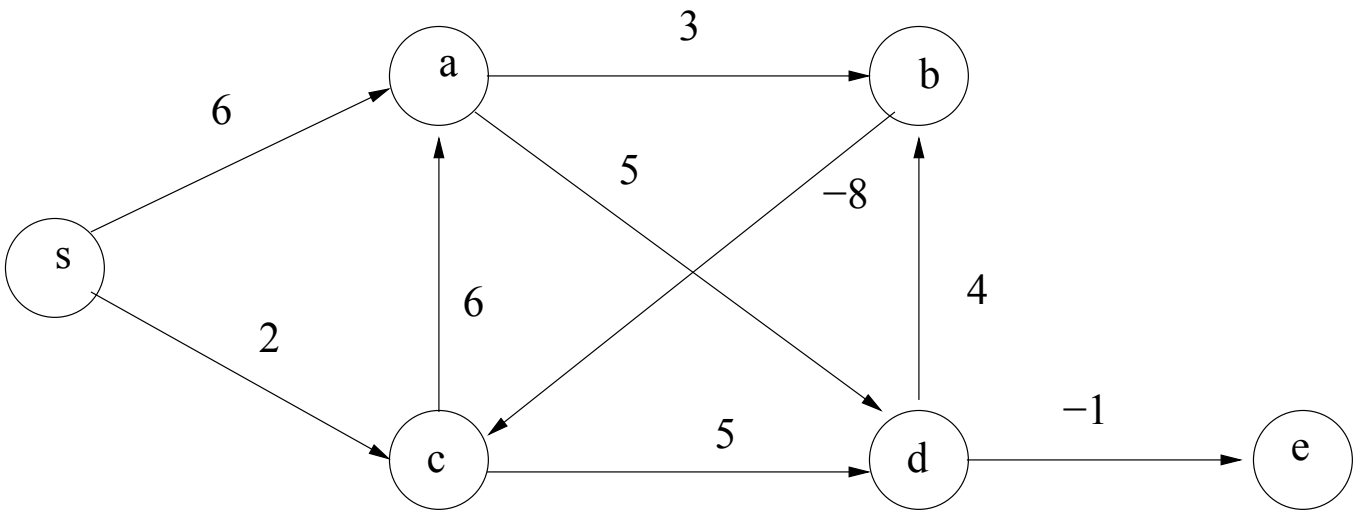
Bellman-Ford(G, w, s)

```
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  for i ← 1 to |V[G]| − 1
3        do for each edge (u, v) ∈ E[G]
4              do RELAX(u, v, w)
5  for each edge (u, v) ∈ E[G]
6        do if d[v] > d[u] + w(u, v)
7              then return FALSE
8  return TRUE
```

$Initialize - Single - Source(G, s)$

```
1  for each vertex v ∈ V[G]
2        do d[v] ← ∞
3           π[v] ← NIL
4  d[s] ← 0
```

Example



Correctness of Bellman Ford

- **Every shortest path must be relaxed in order**
- **If there are negative weight cycles, the algorithm will return false**

Running Time $O(VE)$

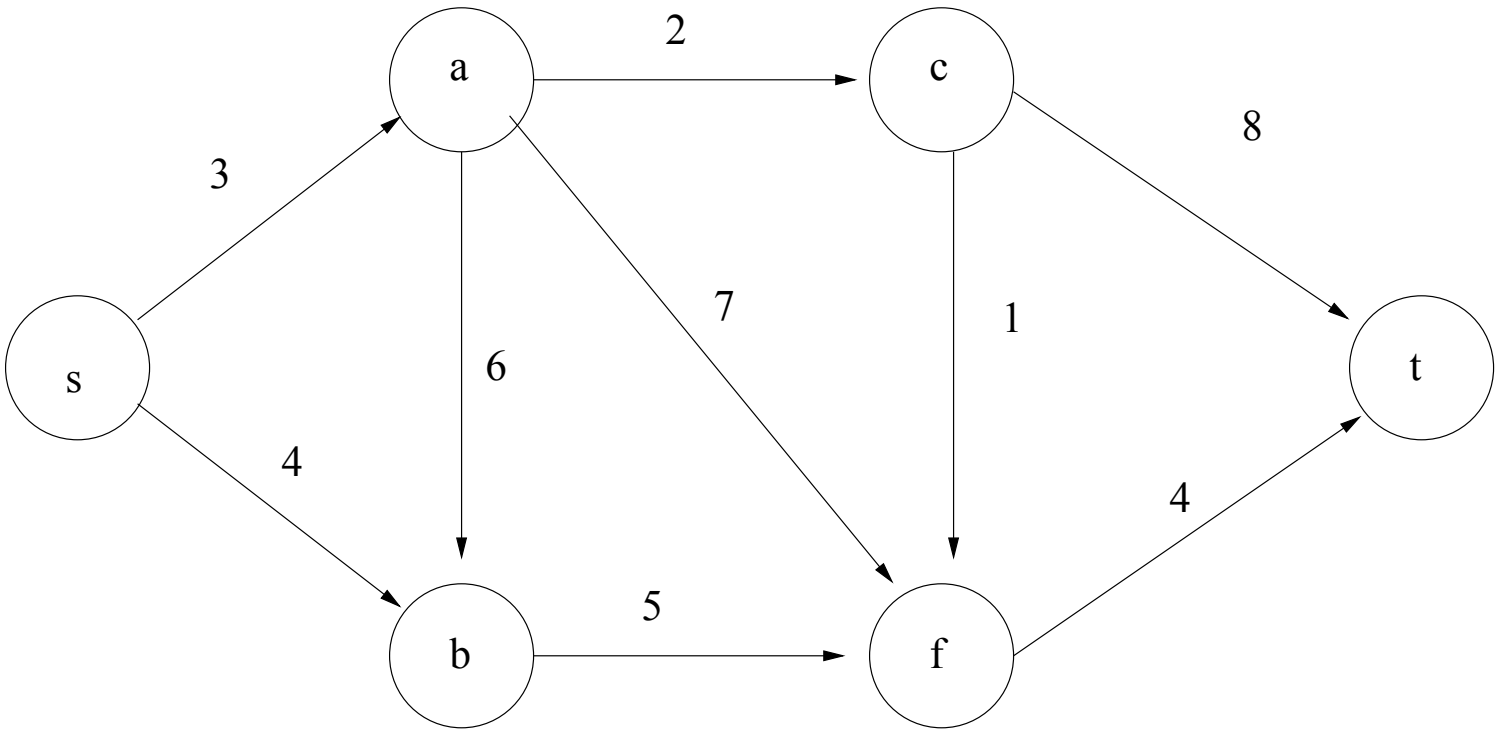
All edges non-negative

- **Dijkstra's algorithm, a greedy algorithm**
- **Similar in spirit to Prim's algorithm**
- **Idea: Run a discrete event simulation of breadth-first-search. Figure out how to implement it efficiently**
- **Can relax edges out of each vertex exactly once.**

$Dijkstra(G, w, s)$

1 INITIALIZE-SINGLE-SOURCE(G, s)
2 $S \leftarrow \emptyset$
3 $Q \leftarrow V[G]$ ▷ **This line does** V INSERTS
4 **while** $Q \neq \emptyset$
5 **do** $u \leftarrow$ EXTRACT-MIN(Q)
6 $S \leftarrow S \cup \{u\}$
7 **for each vertex** $v \in Adj[u]$
8 **do** RELAX(u, v, w) ▷ **This line does a** DECREASE-KEY

Example



Correctness

Correctness of Dijkstra's algorithm **Dijkstra's algorithm, run on a weighted, directed graph $G = (V, E)$ with nonnegative weight function w and source s, terminates with $d[v] = \delta(s, v)$ for all vertices $u \in V$.**

Claim to Prove: **When v is put in S, $d(v) = \delta(v)$.**

Proof

Claim to Prove: When v is put in S, $d(v) = \delta(v)$.

Proof

- $d(v) \geq \delta(v)$ becauase any algorithm that does a sequence of Relax calls has this property.
- Assume fpoc that $d(v) > \delta(v)$, and that v is the first such vertex that is permanently labelled that has this property.
- Consider the state of the world just before v is put in S
  - shortest path from s to v goes through an edge (x, y) where $x \in S$ and $y \notin S$ (it is possible that $y = v$ and/or $x = s$.
  - $d(x) = \delta(x)$, because $x \in S$
  - $d(y) = \delta(y)$ because (x, y) was relaxed when x was put in S.
- Putting these together with $\delta(y) \leq \delta(v)$ because y is before v on a shortest path, we have

$$d(y) = \delta(y) \leq \delta(v) < d(v)$$

.

- $d(y) < d(v)$, so the algorithm would have chosen to permanently label y and not v, which is a contradiction.

Running Time

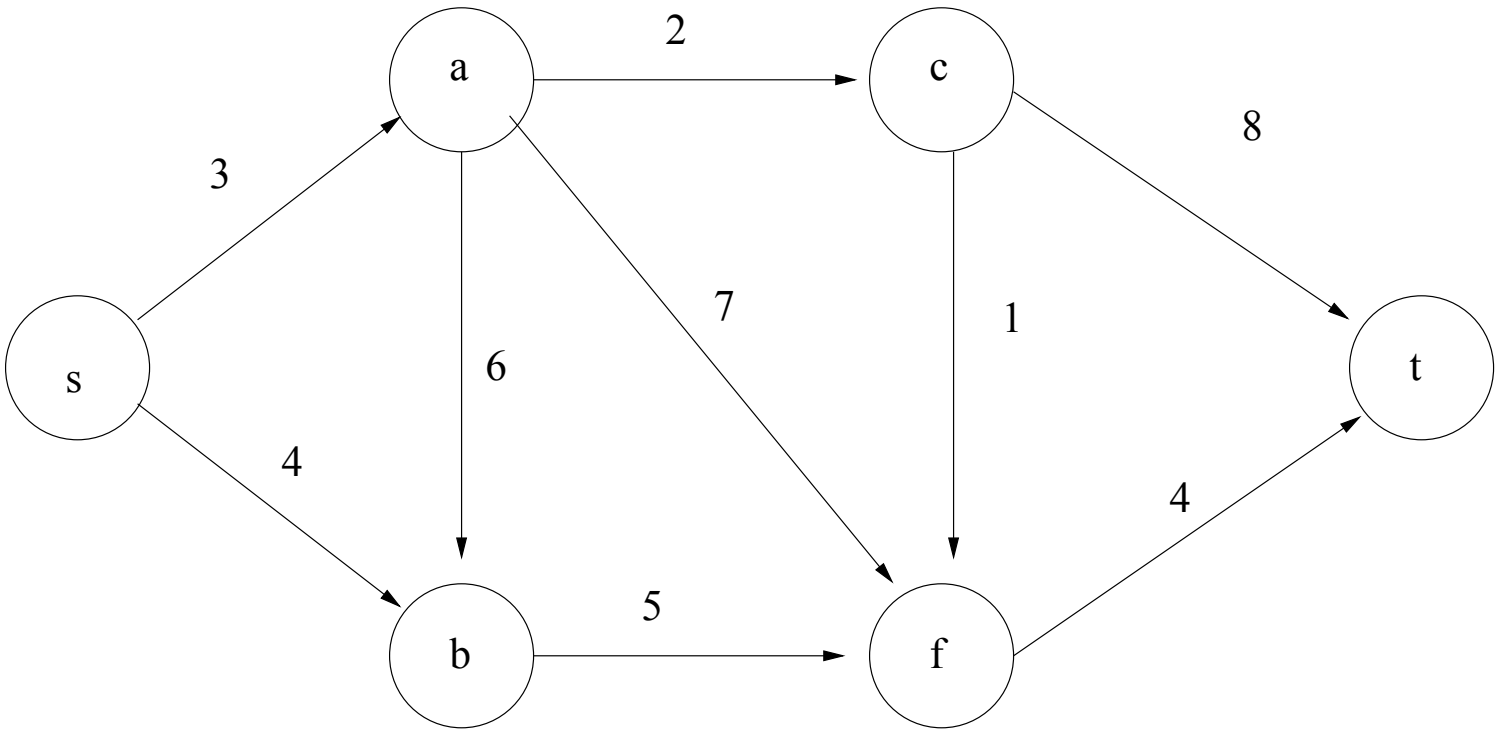
- E **decrease keys and** V **delete-min's**
- $O(E \log V)$ **using a heap**
- $O(E + V \log V)$ **using a Fibonacci heap**

Shortest Path in a DAG

Dag-Shortest-Paths(G, w, s)

1 **topologically sort the vertices of** G
2 INITIALIZE-SINGLE-SOURCE$'(G, s)$
3 **for each** u **taken in topological order**
4 **do for each** $v \in Adj[u]$
5 **do** RELAX(u, v, w)

Example



Correctness and Running Time

Correctness **If a weighted, directed graph $G = (V, E)$ has source vertex s and no cycles, then at the termination of the DAG-SHORTEST-PATHS procedure, $d[v] = \delta(s, v)$ for all vertices $v \in V$, and the predecessor subgraph G_π is a shortest-paths tree.**

Running Time

- **Topological sort is linear time**
- **Each edge is relaxed once**
- **No additional data structure overhead**

$O(V + E)$ time.