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:**

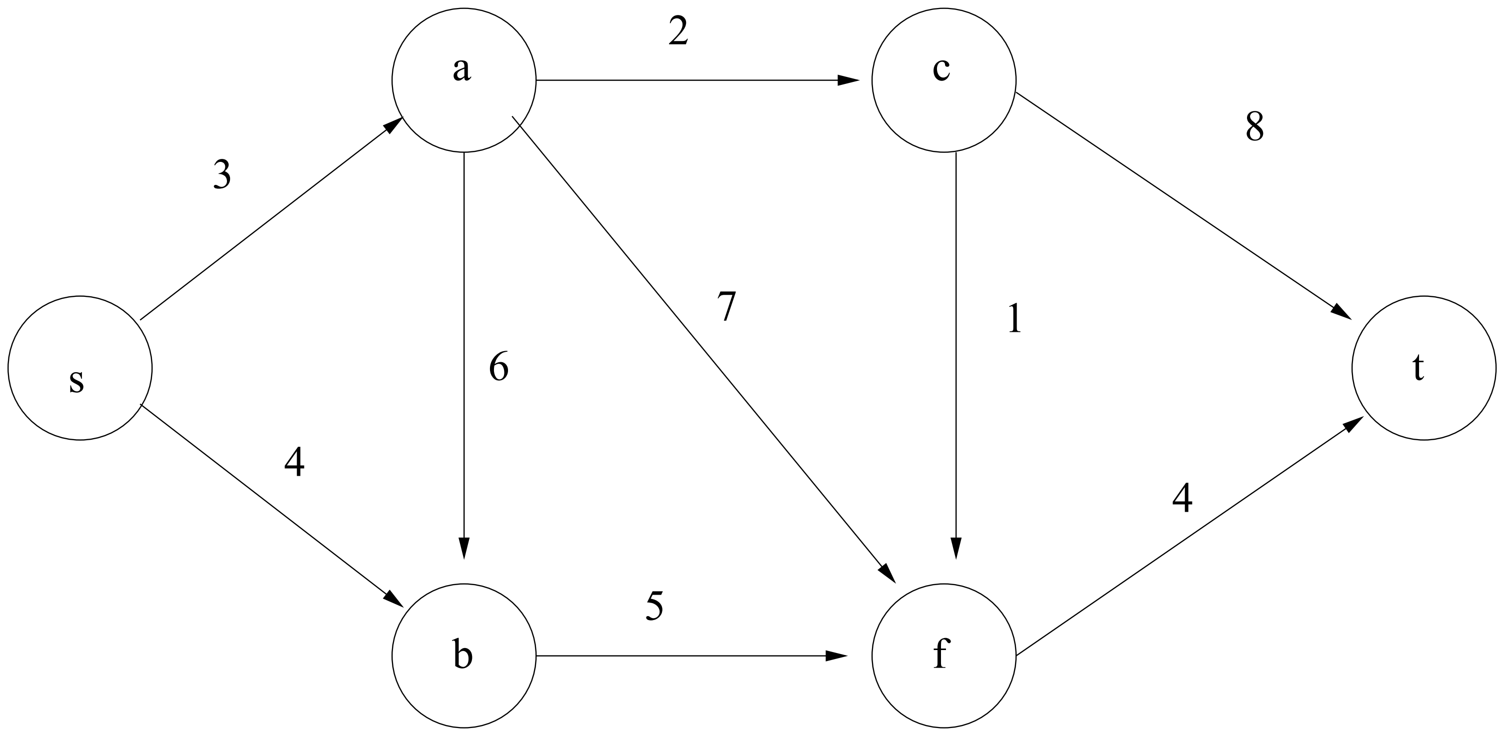
$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i) \ .$$

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

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

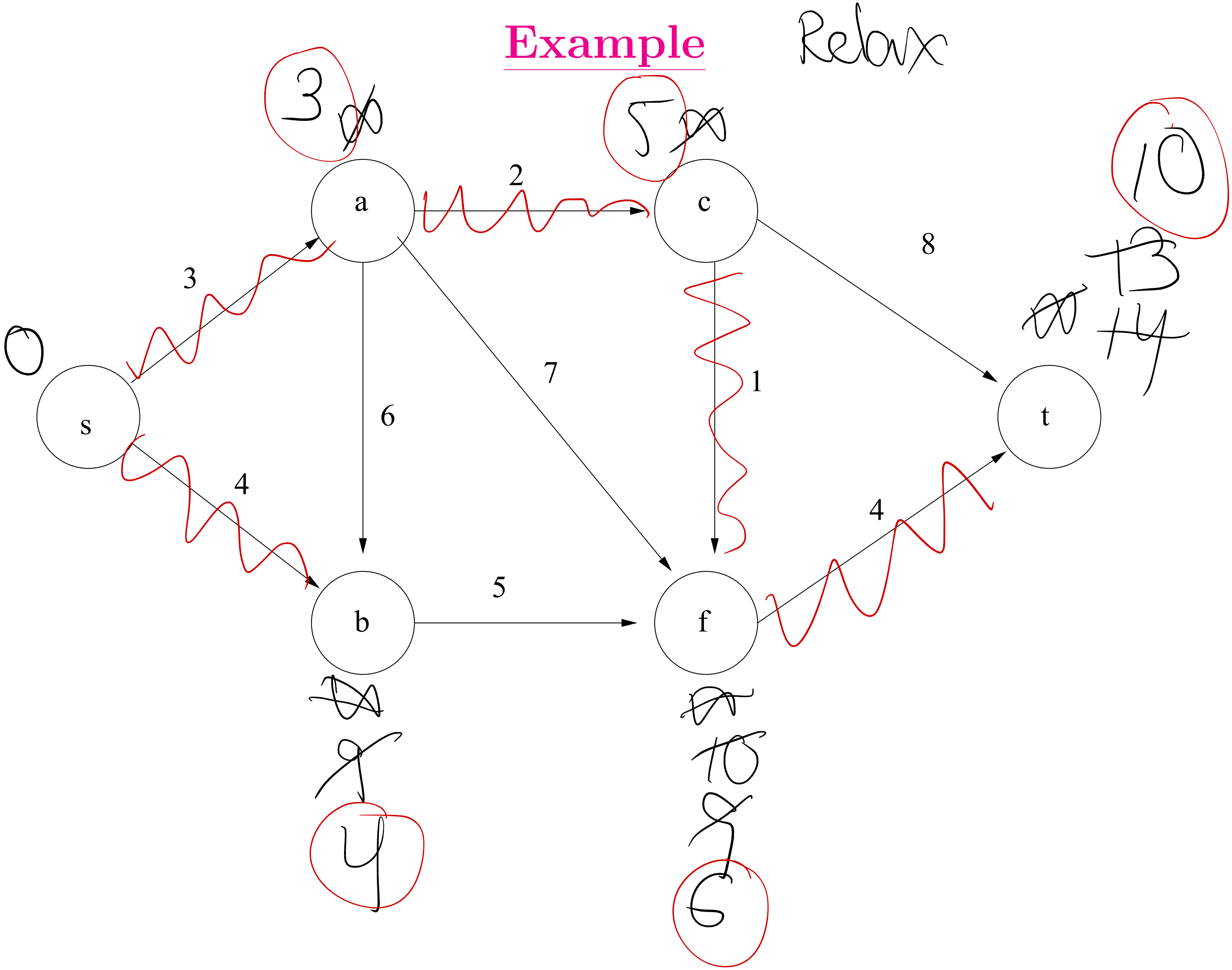
- **A shortest path from vertex u to vertex v is then defined as any path p with weight $w(p) = \delta(u, v)$.**

Example

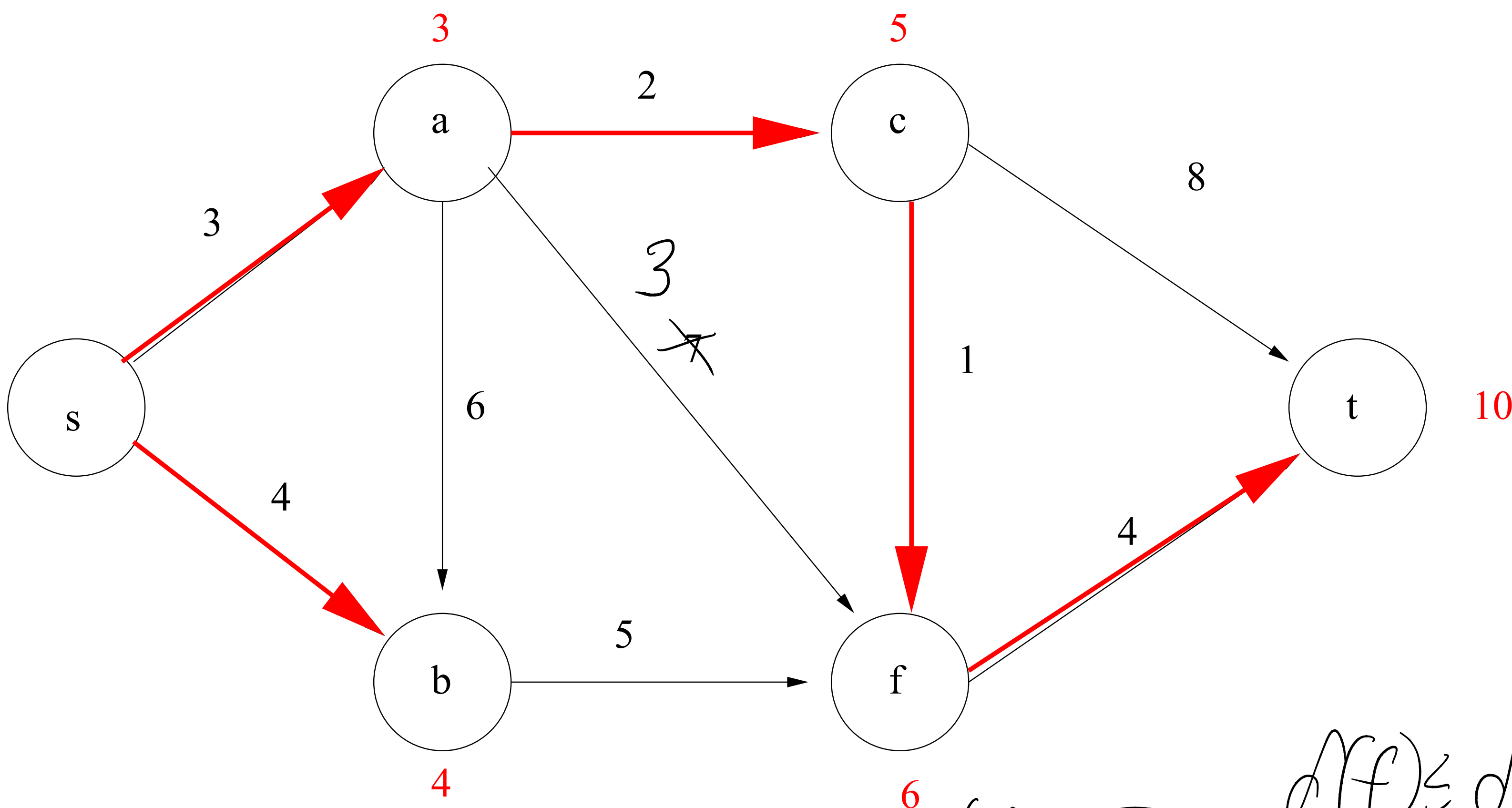


Example

Relax



Solution



$\delta(f) \leq \delta(b) + 5$

$\delta(f) \leq \delta(a) + 7$

$\delta(f) \leq \delta(c) + 1$

Shortest Paths

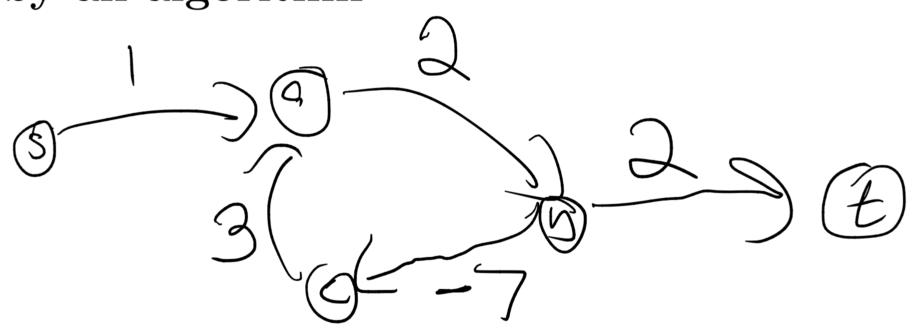
No directed cycles of negative total weight.

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 $w : E \to \mathbf{R}$, 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

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

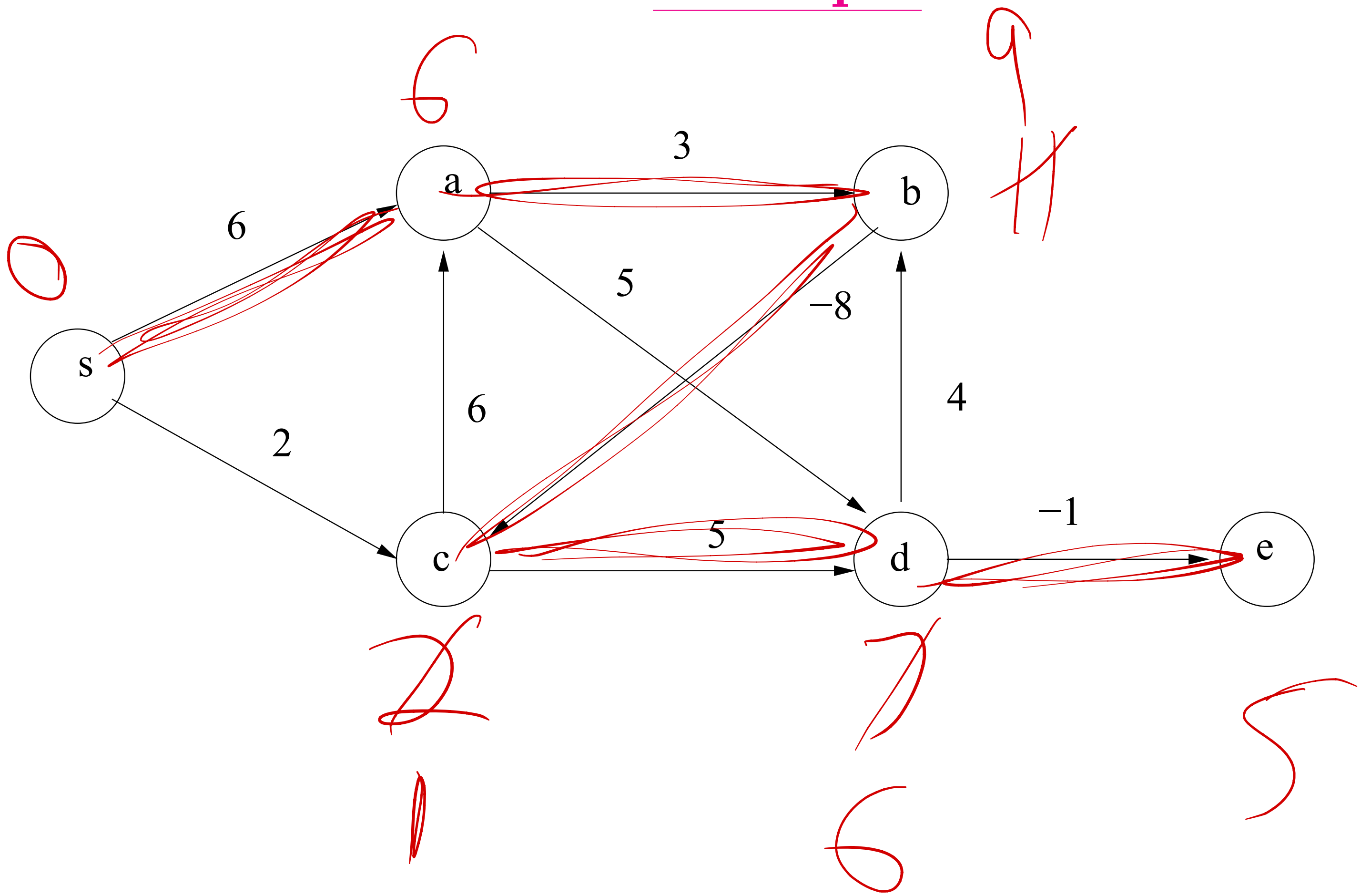
Relaxation

```
Relax(u, v, w)
  if d[v] > d[u] + w(u, v)
     then d[v] ← d[u] + w(u, v)
          π[v] ← 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

S.P., 10, 20, 107, 3, 112, 111, 16

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

edge 1, 2, 3, ..., E

$\underbrace{1, 2, 3, \ldots, E}, \underbrace{1, 2, \ldots, E}, \underbrace{\ldots \ldots}, \underbrace{1, 2, \ldots, E}$

V times

10 20 107 3, 112 111 16

Algorithms

$0 \le w(x,y) + w(y,z) + w(z,x)$

$\cancel{d(y)} + \cancel{d(z)} + \cancel{d(x)} \le \cancel{d(x)} + \cancel{d(y)} + \cancel{d(z)} + w(x,y) + w(y,z) + w(z,x)$

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

$d(y) \le d(x) + w(x,y)$

$d(z) \le d(y) + w(y,z)$

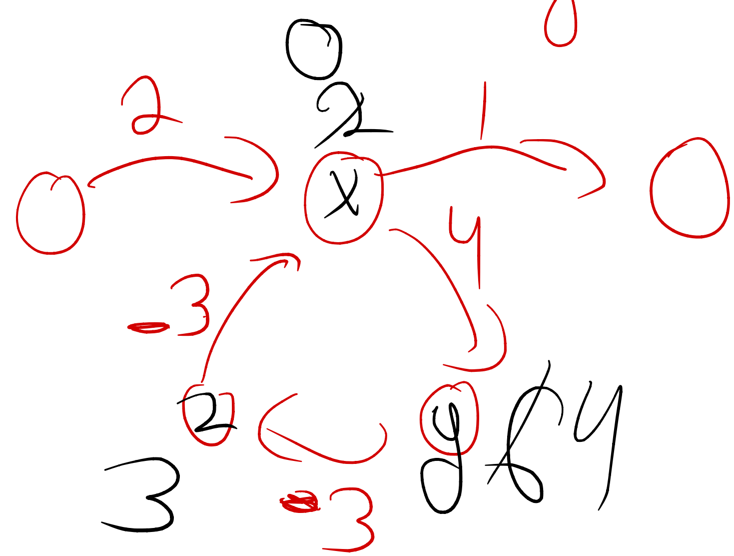
$d(x) \le d(z) + w(z,x)$

alg. stops

Bellman-Ford(G, w, s)

INITIALIZE-SINGLE-SOURCE(G, s)
for $i \leftarrow 1$ **to** $|V[G]| - 1$
  do for each edge $(u, v) \in E[G]$
    do RELAX(u, v, w)
for each edge $(u, v) \in E[G]$
  do if $d[v] > d[u] + w(u, v)$
    then return FALSE
return TRUE

checks for negative cycle

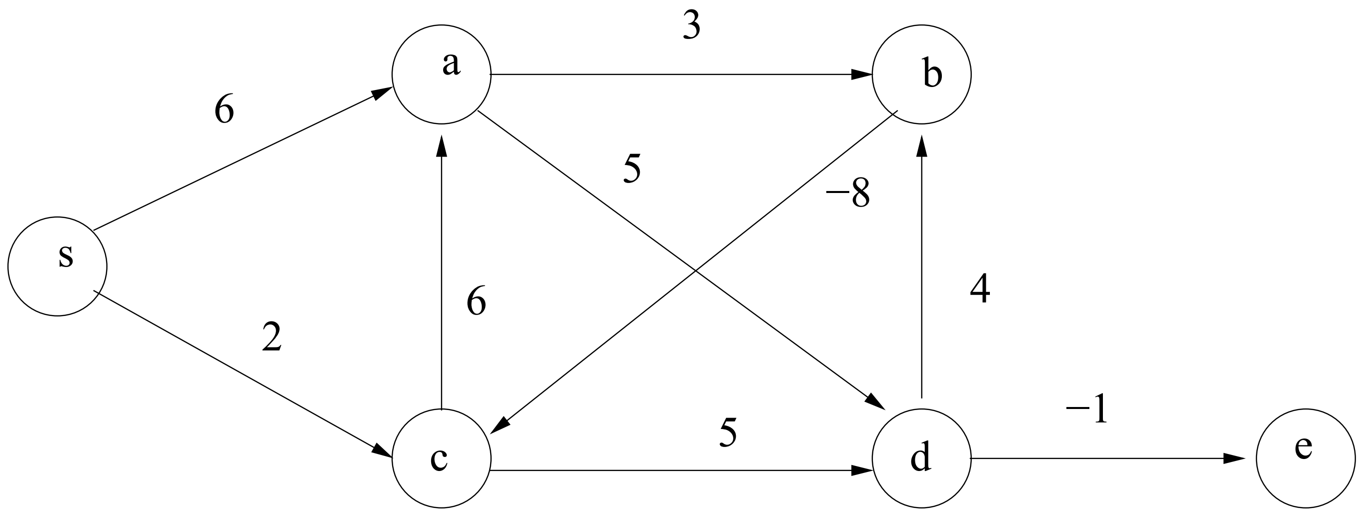


$Initialize - Single - Source(G, s)$

for each vertex $v \in V[G]$
  do $d[v] \leftarrow \infty$
    $\pi[v] \leftarrow$ NIL
$d[s] \leftarrow 0$

$O(VE)$

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 $O(VE)$

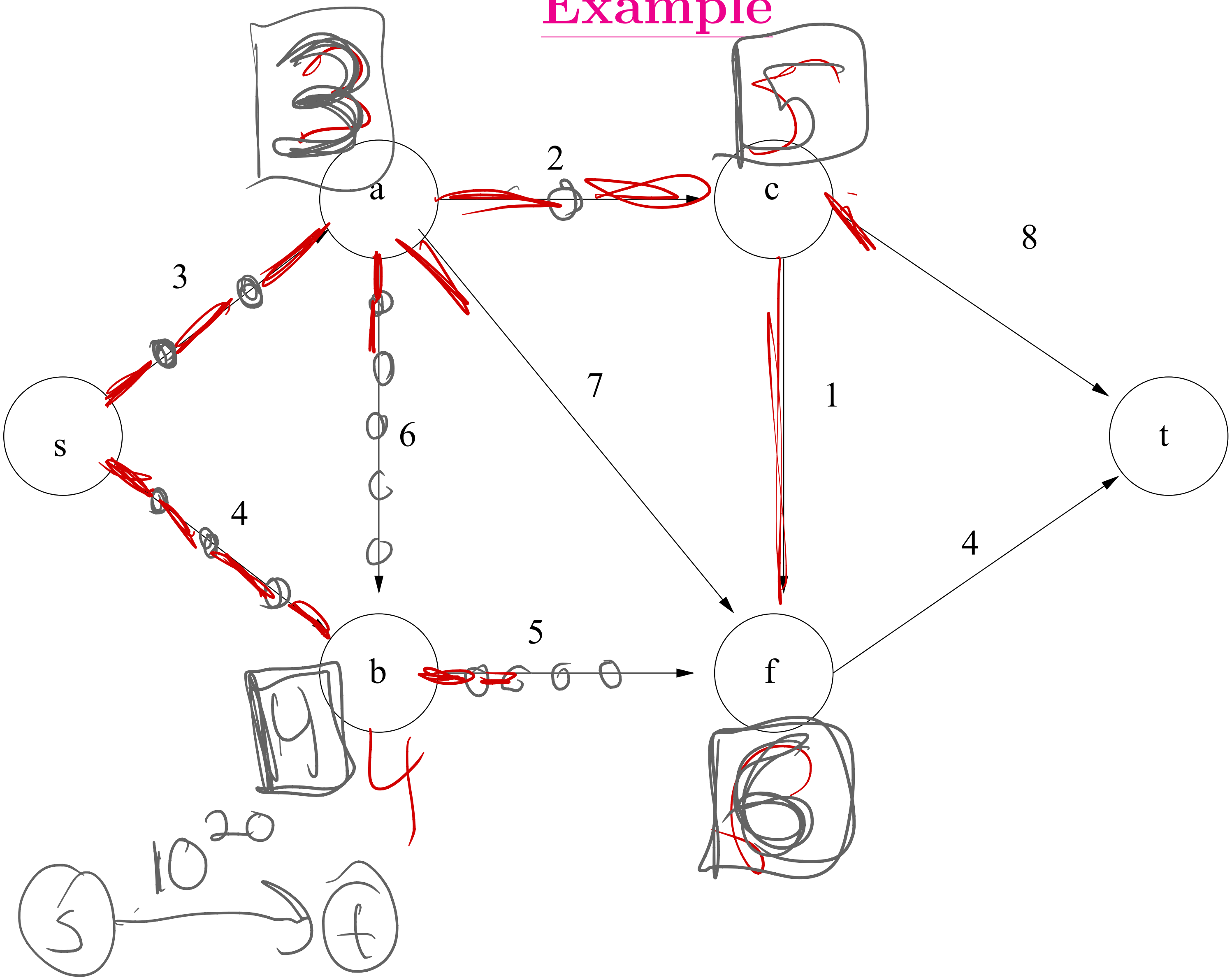
All edges non-negative

- 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.

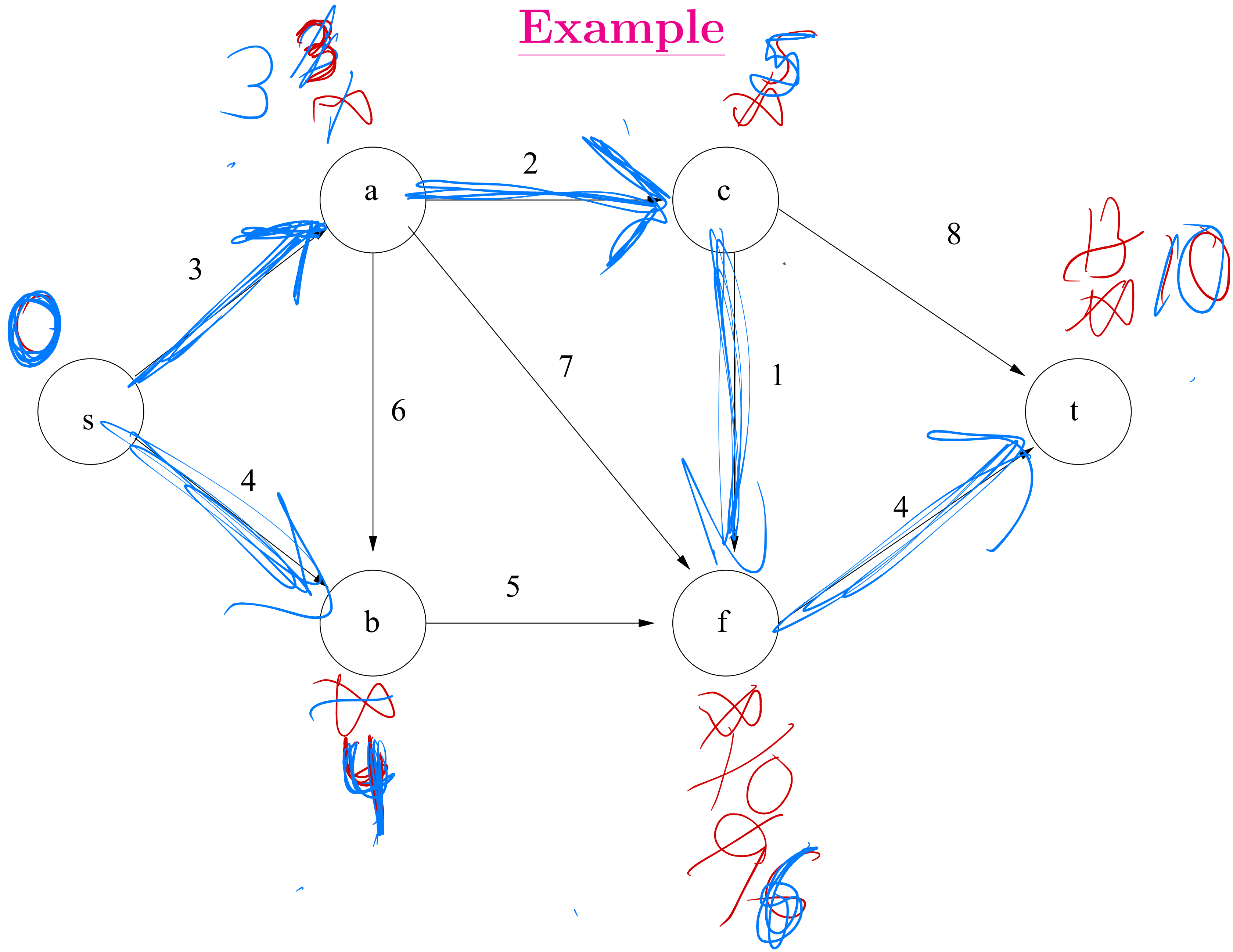
Dijkstra(G, w, s)

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

Example



Example

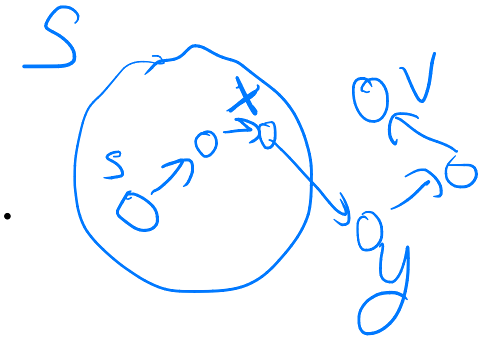


Correctness

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

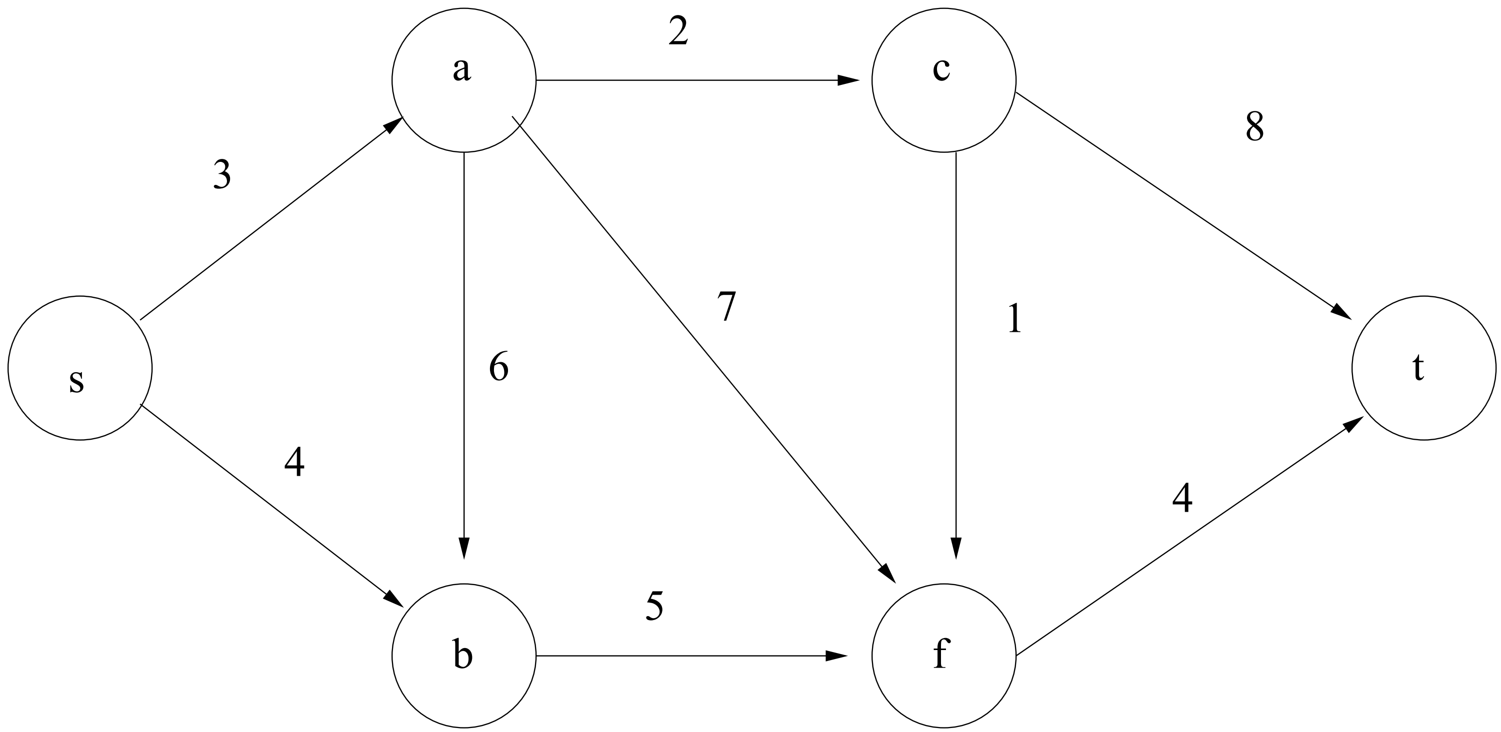
- E **decrease keys and** V **delete-min's**
- $O(E \log V)$ **using a heap**
- $O(E + V \log V)$ **using a Fibonacci heap**

Shortest Path in a DAG

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

topologically sort the vertices of G
Initialize-Single-Source$'(G, s)$
for each u **taken in topological order**
    do for each $v \in Adj[u]$
        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.**