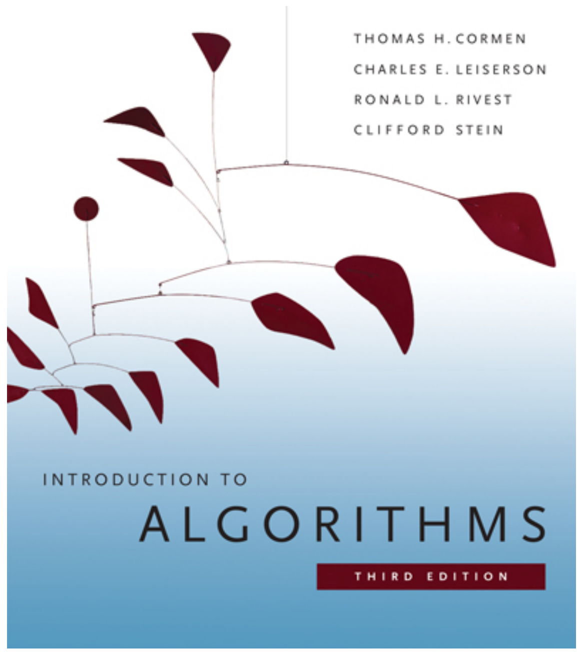
6.006- *Introduction to Algorithms*



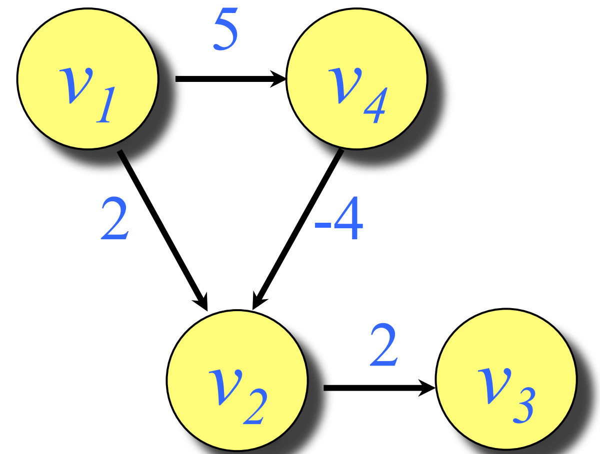
Lecture 16

Alan Deckelbaum

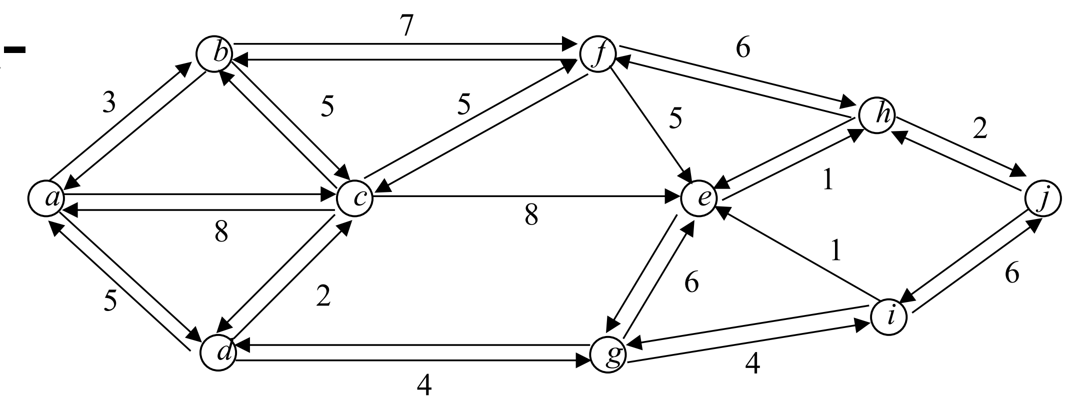
Lecture overview

Shortest paths III

- Bellman-Ford on a DAG (CLRS 24.2)

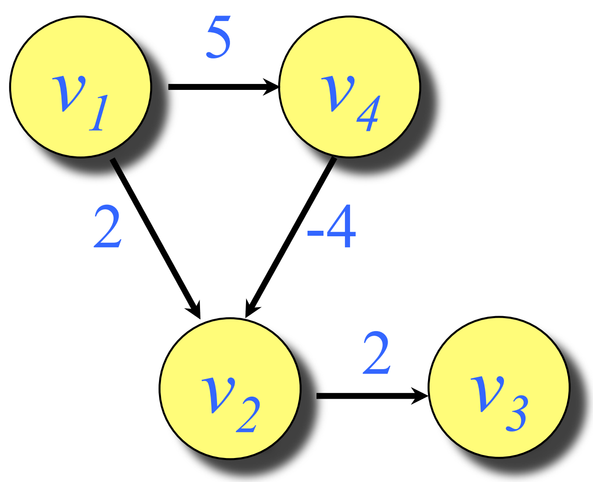


- Dijkstra algorithm for the case with non-negative weights (CLRS 24.3)

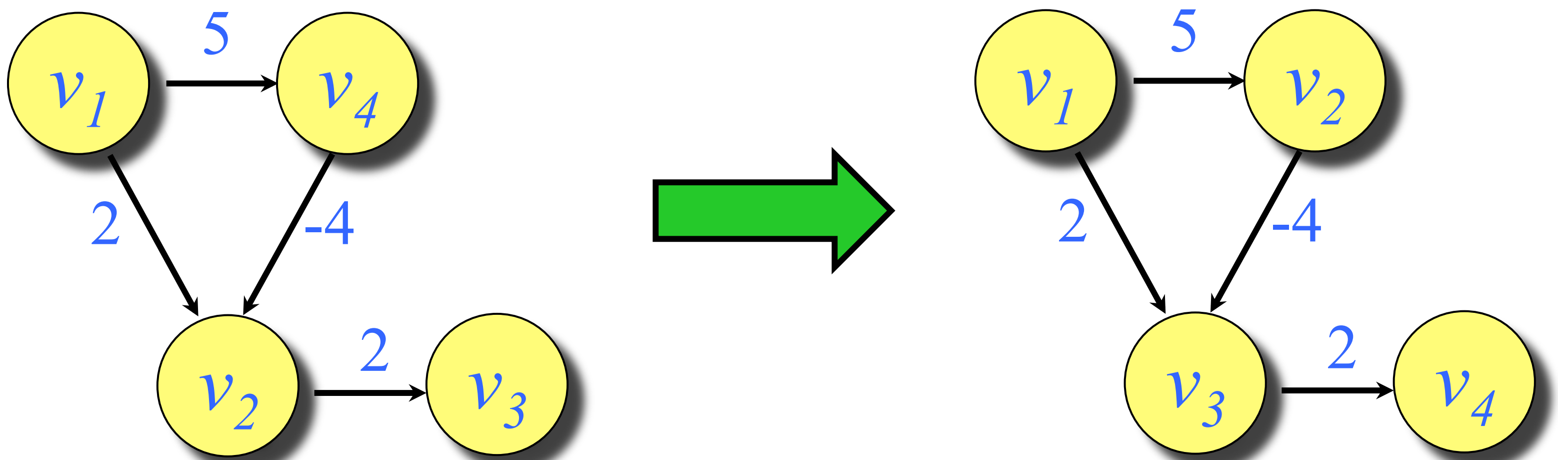


This graph has a special structure: DAG. How to use it within Bellman-Ford?

$$E=\{(v_1,v_2); (v_1,v_4); (v_2,v_3); (v_4,v_2)\}$$



... first use topological sorting ...



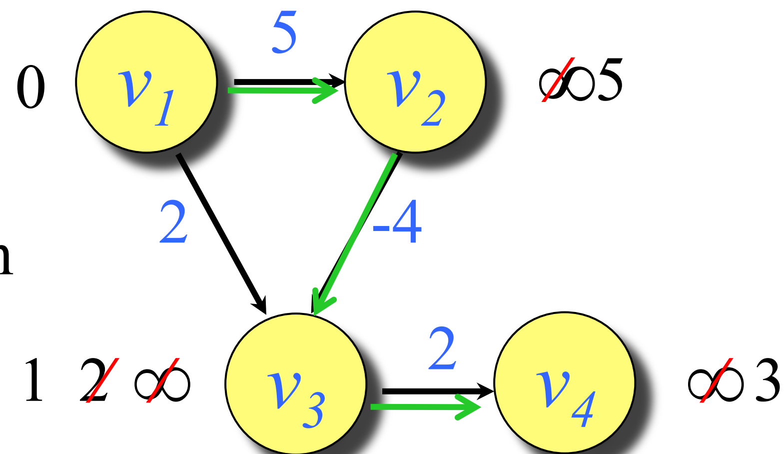
$E=\{(v_1,v_2); (v_1,v_4); (v_2,v_3); (v_4,v_2)\}$

$E=\{(v_1,v_2); (v_1,v_3); (v_2,v_3); (v_3,v_4)\}$

... Bellman-Ford ...

$E=\{(v_1,v_2); (v_1,v_3); (v_2,v_3); (v_3,v_4)\}$

end of first iteration



and we are done !

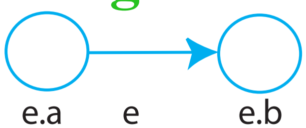
the shortest paths from v_1

Bellman-Ford algorithm on DAG

topologically sort the vertices V

($f: V \rightarrow \{1, 2, \ldots, |V|\}$ such that $(u,v) \in E \Rightarrow f(u) < f(v)$)

arrange E in lexicographical order of *(f(e.a), f(e.b))* — *O(n+m)*



$d[s] \leftarrow 0$; $\pi[s] \leftarrow$ s
for each $v \in V - \{s\}$
  do $d[v] \leftarrow \infty$; $\pi[v] \leftarrow$ nil

initialization — *O(n)*

do for each edge $(u, v) \in E$
  do if $d[v] > d[u] + w(u, v)$
    then $d[v] \leftarrow d[u] + w(u, v)$
      $\pi[v] \leftarrow$ u

one iteration of relaxation steps — *O(m)*

for each edge $(u, v) \in E$
  do if $d[v] > d[u] + w(u, v)$
    then report a negative cycle

final steps not needed

... why does this work? ...

- there are no cycles in a dag => even with negative-weight edges, there are no negative-weight cycles ...
- topological ordering implies a linear ordering of the vertices; every path in a dag is a subsequence of topologically sorted vertex order; processing vertices in that order, an edge can't be relaxed more than once ...

Proof of Correctness

- Let t be an arbitrary vertex. Suffices to show that we compute d[t] properly.
- Let $s = s_0, s_1, s_2, \ldots, s_k = t$ be a shortest path to t. Show by induction that we compute each d[s_i] correctly.
- d[s_{i-1}] computed correctly by inductive hypothesis.
- (s_{i-1}, s_i) relaxed AFTER d[s_{i-1}] computed.

Review of Dijkstra (Non-negative Edge Weights)

Problem: Given a directed graph $G = (V, E)$ with edge-weight function $w : E \rightarrow$ R+, and a node s, find the shortest-path weight $\delta(s, v)$ (and a corresponding shortest path) from s to each v in V.

Greedy iterative approach

1. maintain a set S of vertices whose shortest-path distances from s are known.
2. at each step add to S the vertex $v \in V - S$ whose distance estimate from s is minimal.
3. update distance estimates of vertices adjacent to v.

Dijkstra's algorithm

$d[s] \leftarrow 0$
for each $v \in V - \{s\}$
  do $d[v] \leftarrow \infty$
$S \leftarrow \emptyset$
$Q \leftarrow V$

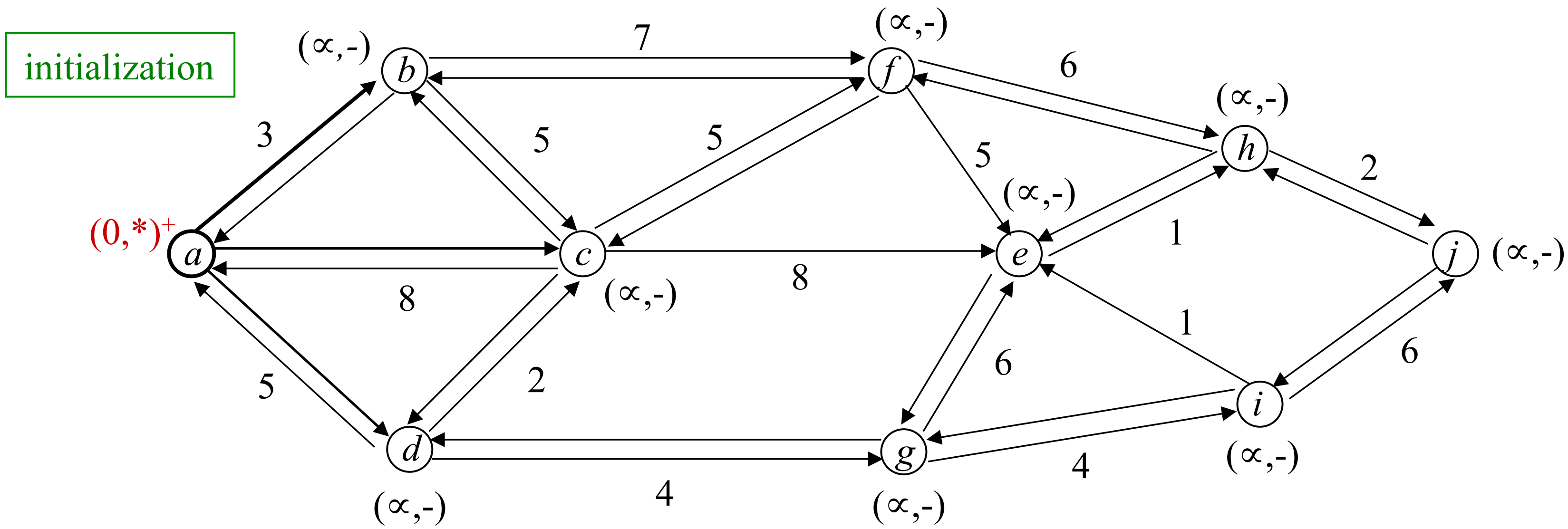
} ***initialization***

while $Q \neq \emptyset$ (Q min-priority queue maintaining $V - S$)
  do $u \leftarrow$ EXTRACT-MIN(Q)
    $S \leftarrow S \cup \{u\}$
    for each $v \in Adj[u]$
      do if $d[v] > d[u] + w(u, v)$
        then $d[v] \leftarrow d[u] + w(u, v)$

} ***relaxation steps***

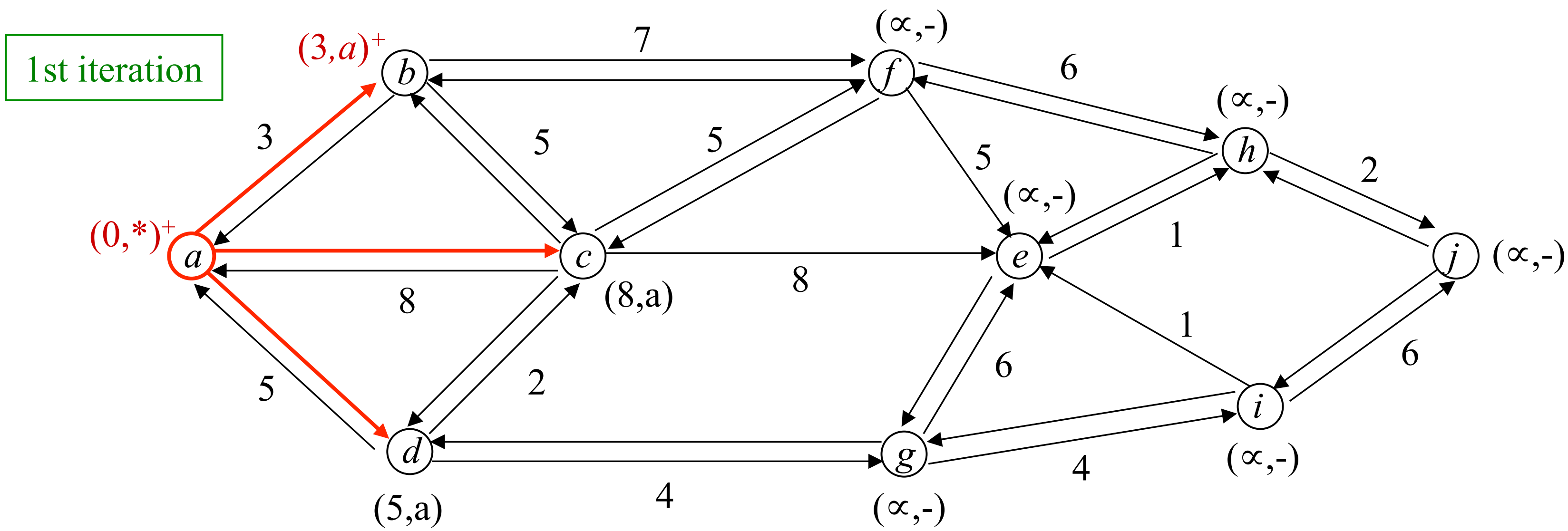
(Implicit DECREASE-KEY)

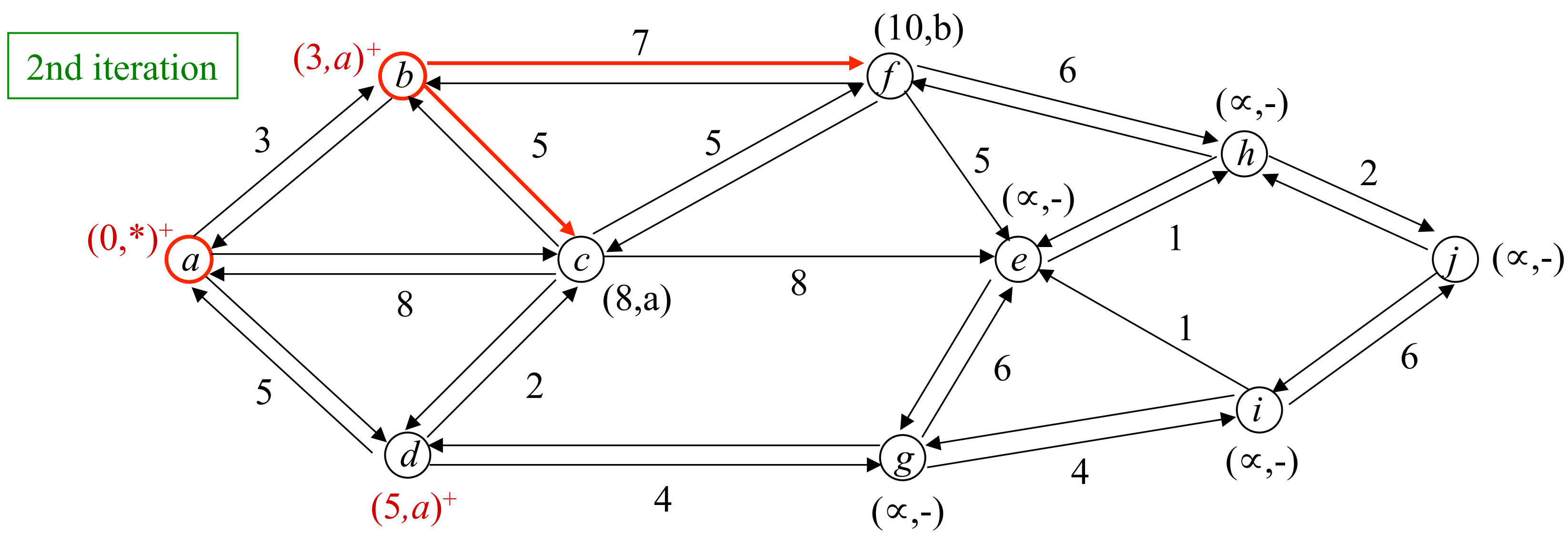
Dijkstra: Example



$Q = V$, a = EXTRACT-MIN(Q)

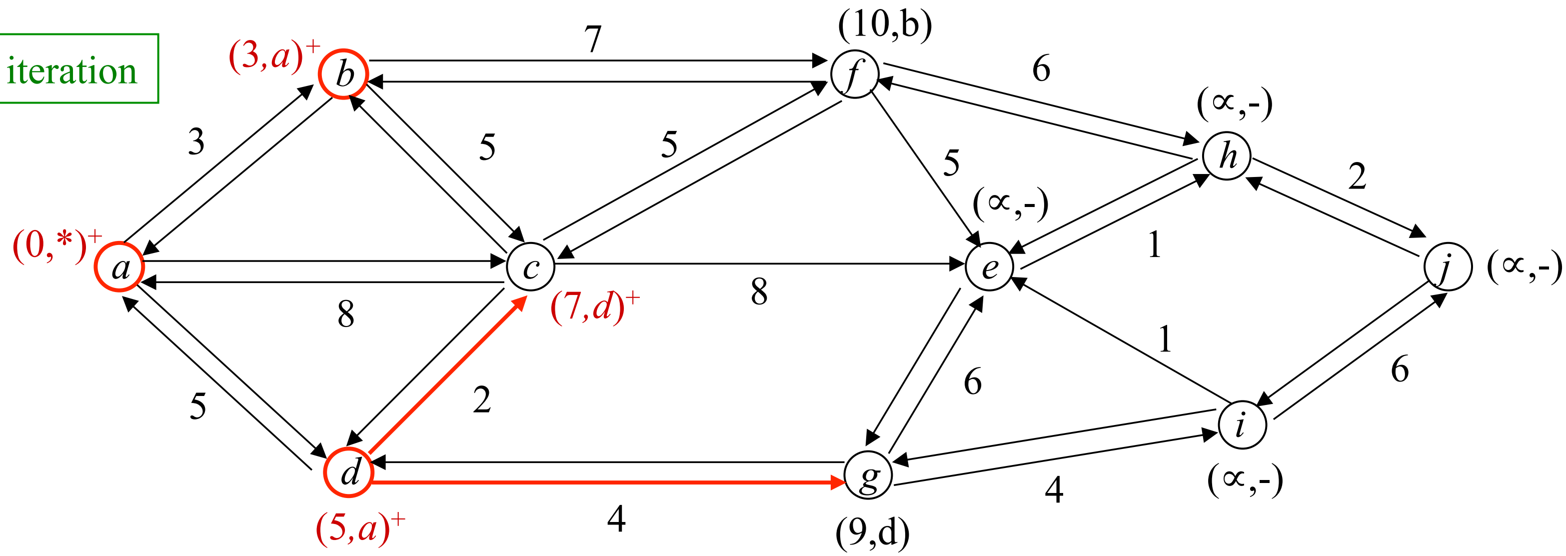
Dijkstra: Example



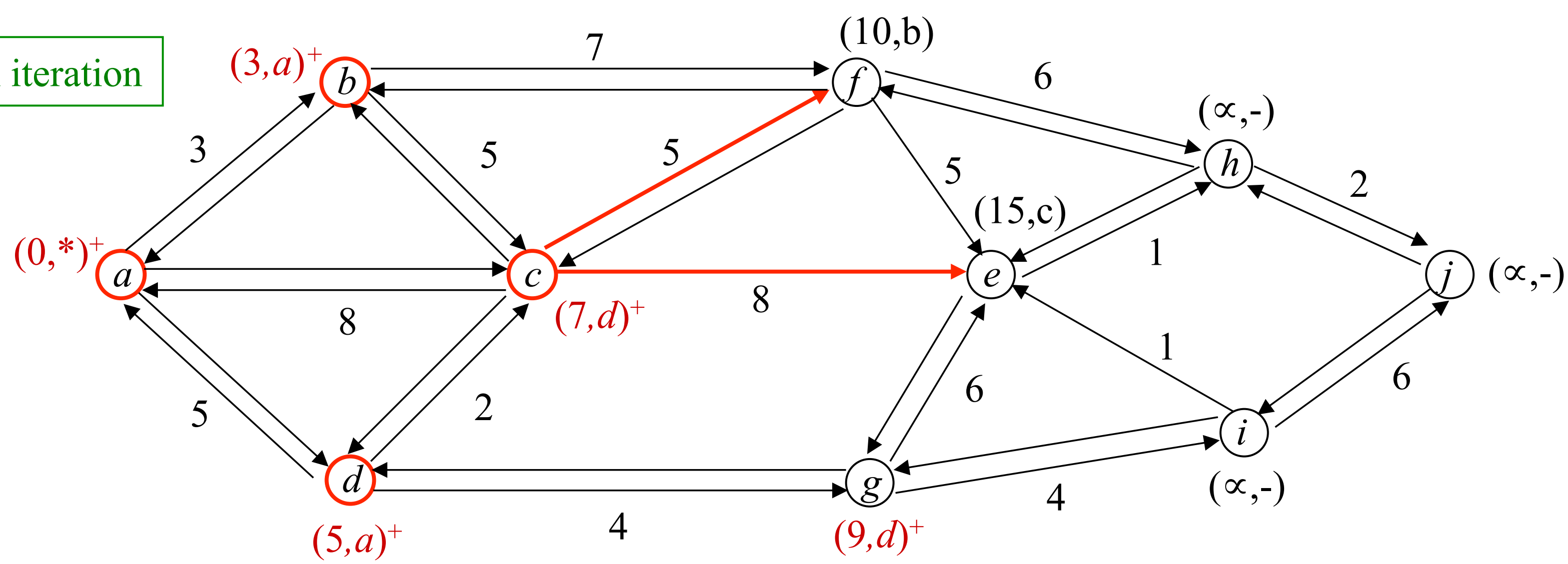


Dijkstra: Example

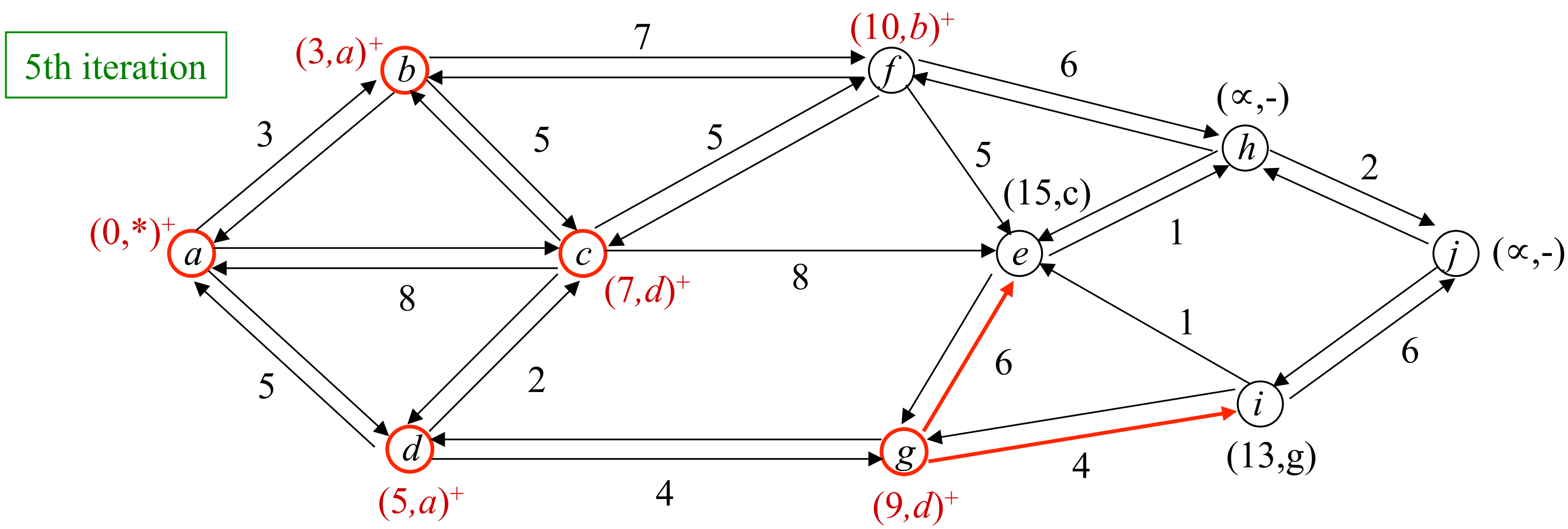
3rd iteration

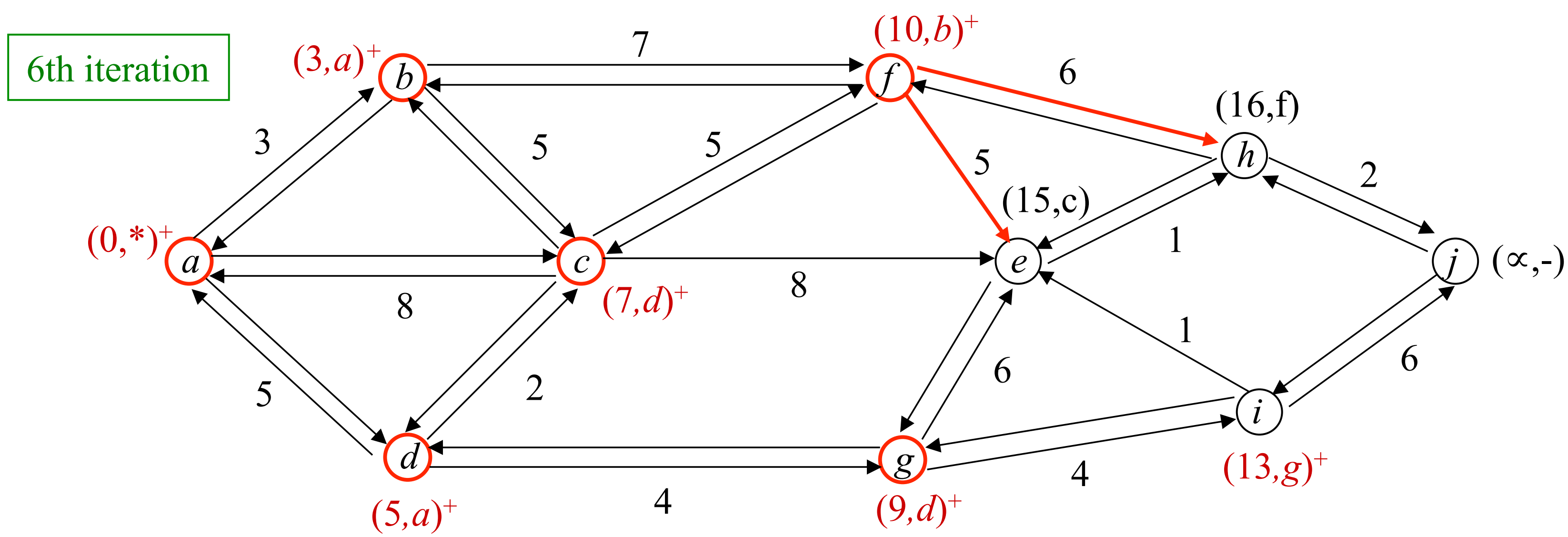


4th iteration

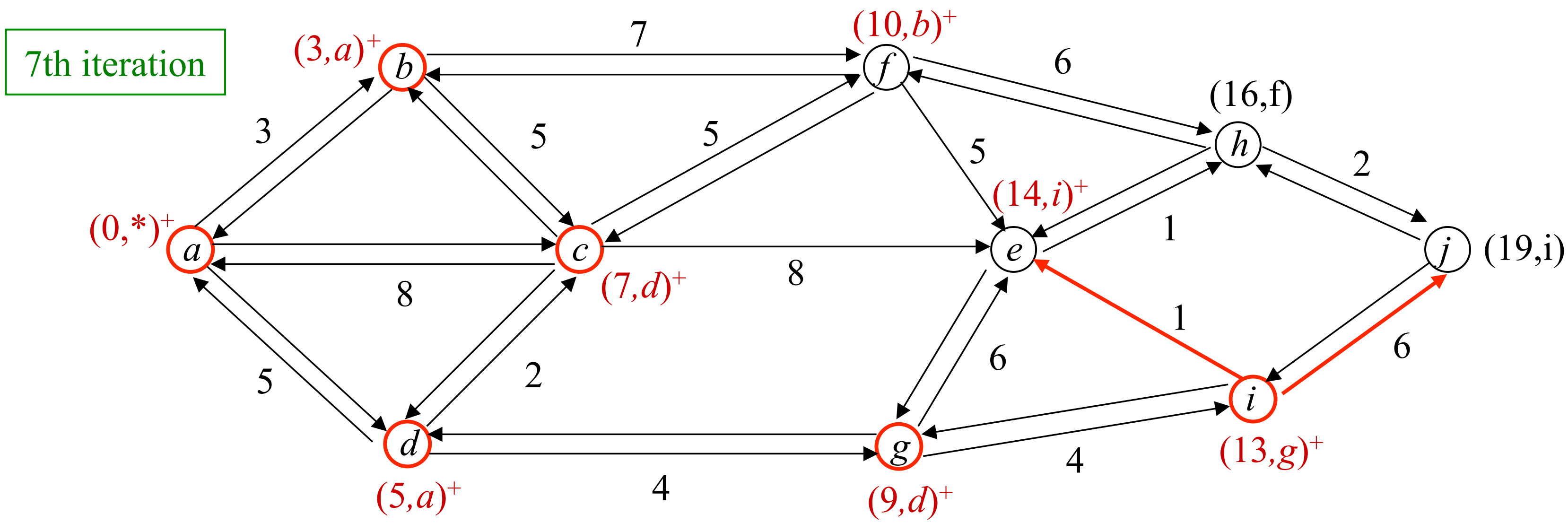


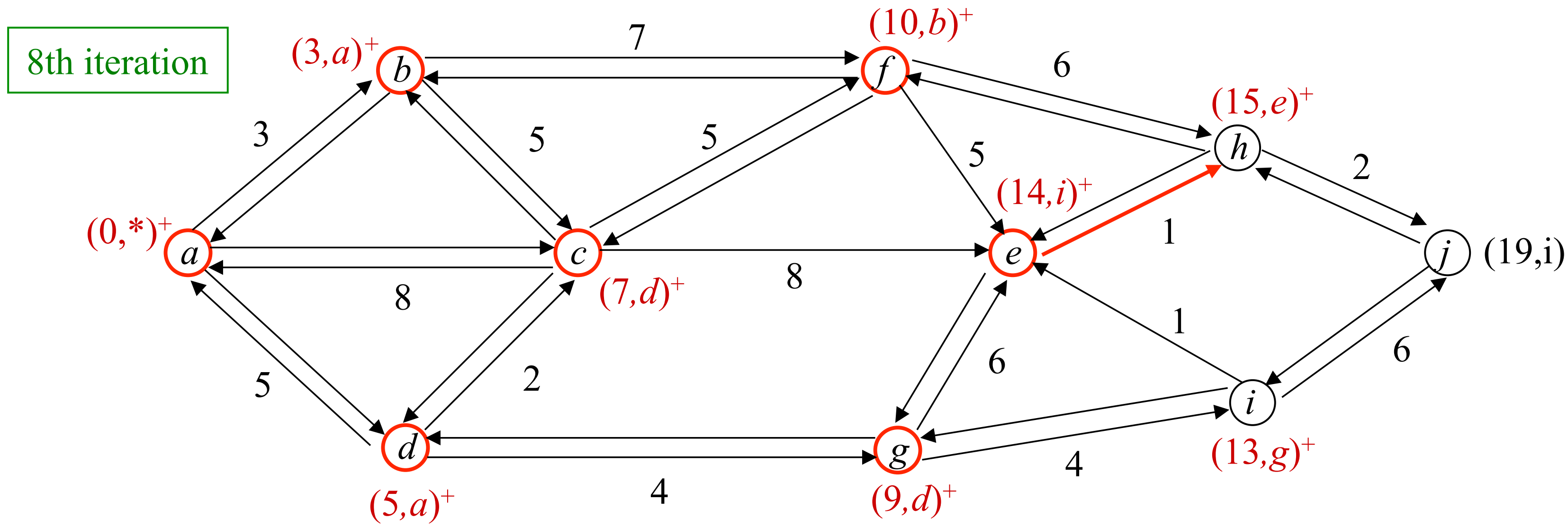
Dijkstra: Example





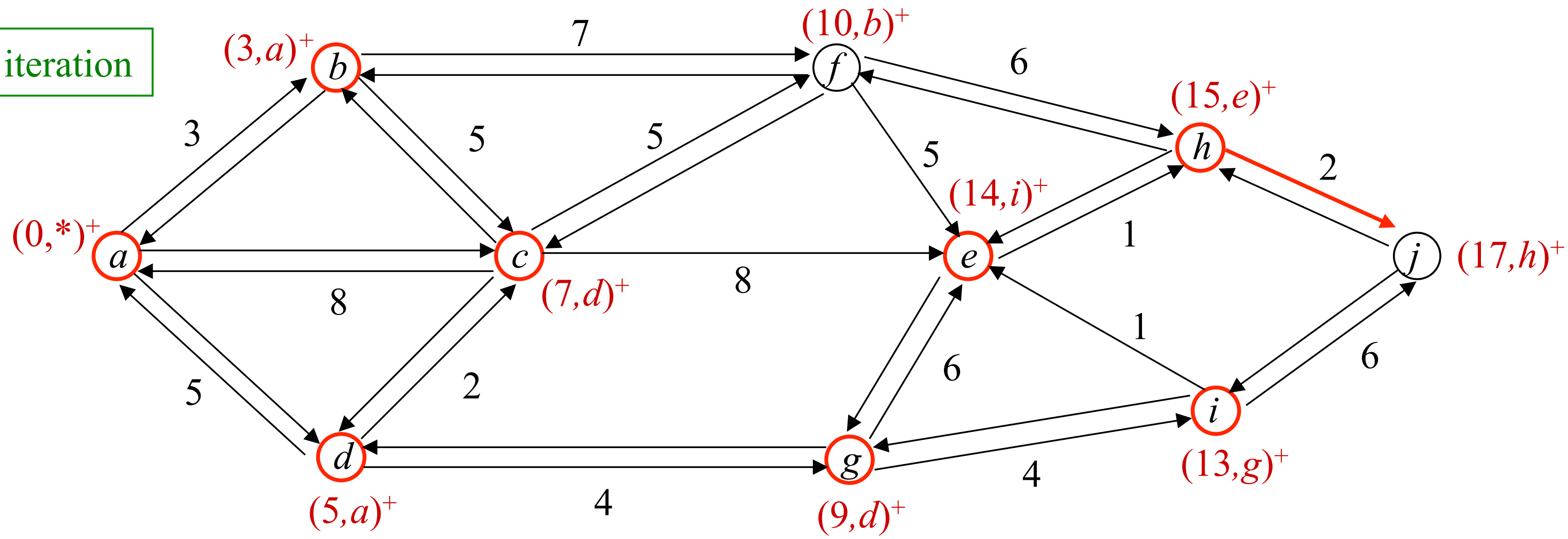
Dijkstra: Example



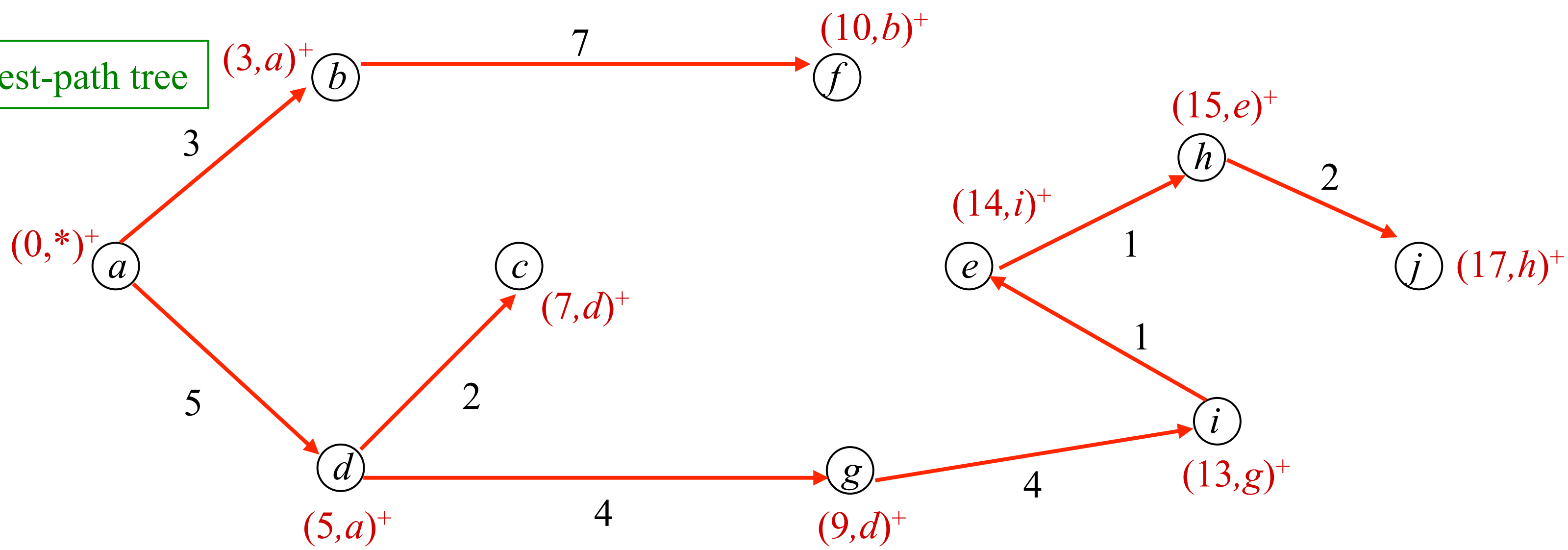


Dijkstra: Example

9th iteration



Shortest-path tree

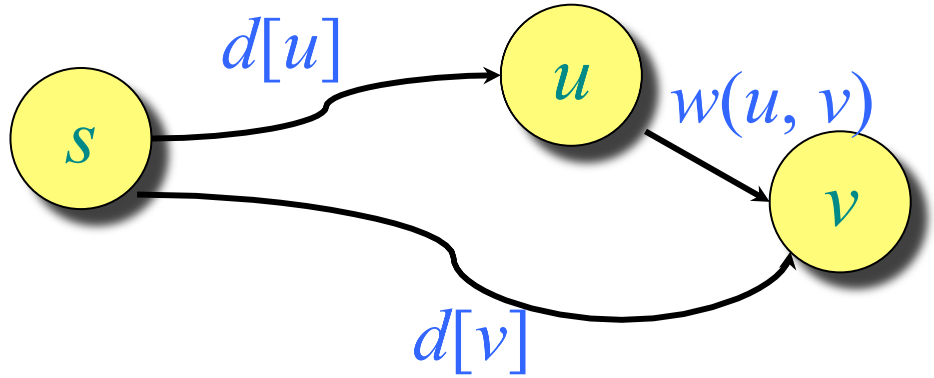


Correctness — Part I

Lemma. Initializing $d[s] \leftarrow 0$ and $d[v] \leftarrow \infty$ for all $v \in V - \{s\}$ establishes $d[v] \geq \delta(s, v)$ for all $v \in V$, and this invariant is maintained over any sequence of relaxation steps.

Proof. Recall relaxation step:

$$\textbf{if } d[v] > d[u] + w(u, v) \textbf{ set } d[v] \leftarrow d[u] + w(u, v)$$

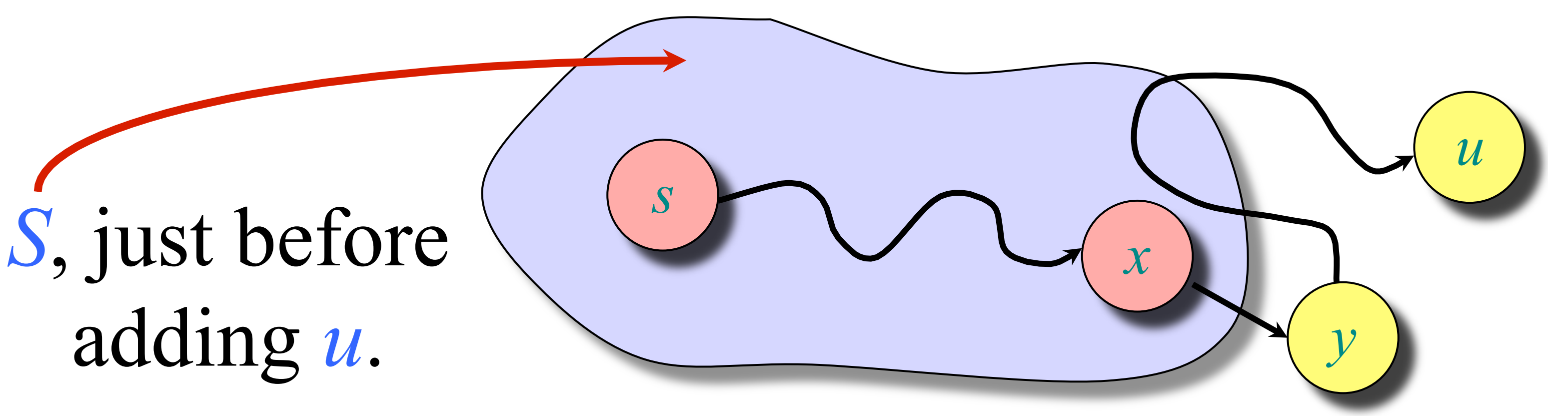


Correctness — Part II

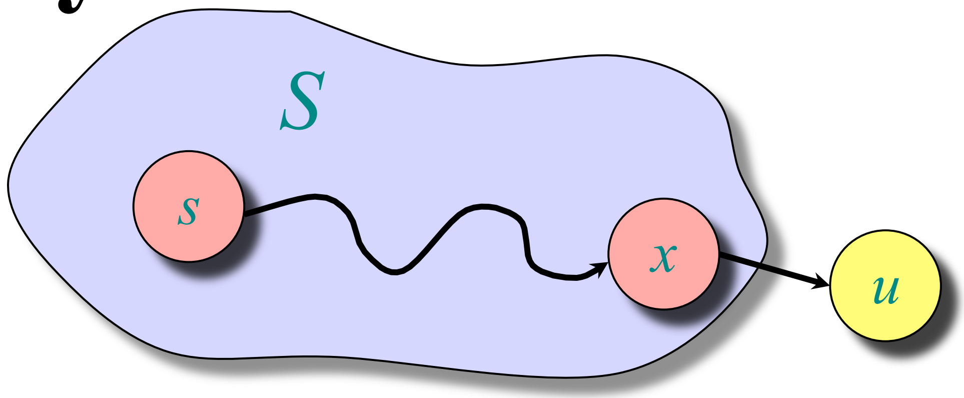
Theorem. Dijkstra's algorithm terminates with $d[v] = \delta(s, v)$ for all $v \in V$.

Proof.

- It suffices to show that $d[v] = \delta(s, v)$ for every $v \in V$ when v is added to S
- Suppose u is the first vertex added to S for which $d[u] > \delta(s, u)$. Let y be the first vertex in $V - S$ along a shortest path from s to u, and let x be its predecessor:

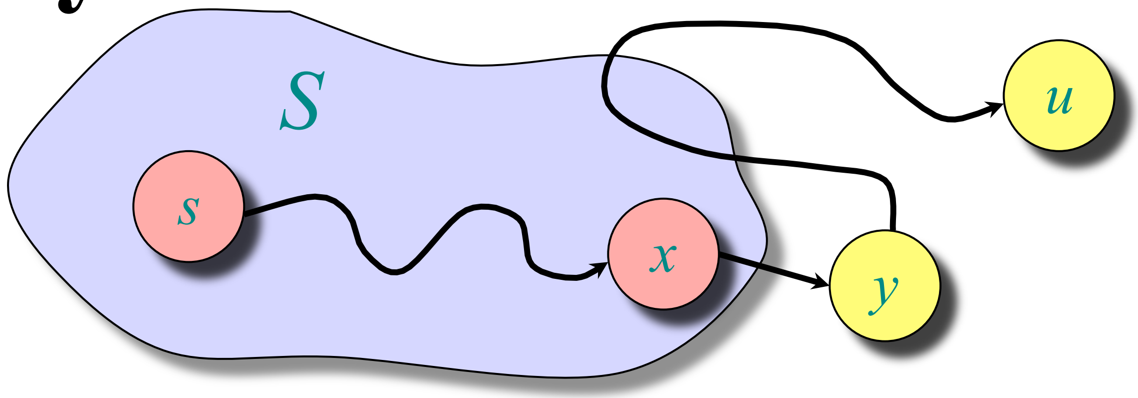


Correctness — Part II (continued)
Case 1: y =u



- Since u is the first vertex violating the claimed invariant, we have $d[x] = \delta(s, x)$ at the time x was added to S.
- Just after x was added to S, we therefore set $d[u] = \delta(s, u)$
- This is a contradiction, since $d[u]$ is never increased by edge relaxation.

Correctness — Part II (continued)
Case 2: y != u



• Since u is the first vertex violating the claimed invariant, we have $d[x] = \delta(s, x)$

• Since subpaths of shortest paths are shortest paths, it follows that $d[y]$ was set to $\delta(s, x) + w(x, y) = \delta(s, y)$ just after x was added to S

• Consequently, we have $d[y] = \delta(s, y) \leq \delta(s, u) < d[u]$

• But, $d[y] \geq d[u]$ since the algorithm chose u first => a contradiction

Analysis of Dijkstra

$|V|$ times

while $Q \neq \emptyset$

do $u \leftarrow \text{EXTRACT-MIN}(Q)$

$degree(u)$ times

$S \leftarrow S \cup \{u\}$

for each $v \in Adj[u]$

do if $d[v] > d[u] + w(u, v)$

then $d[v] \leftarrow d[u] + w(u, v)$

DECREASE-KEY

$$\text{Time} = \Theta(n) \cdot T_{\text{EXTRACT-MIN}} + \Theta(m) \cdot T_{\text{DECREASE-KEY}}$$

Analysis of Dijkstra (continued)

Time = $\Theta(n) \cdot T_{\text{EXTRACT-MIN}} + \Theta(m) \cdot T_{\text{DECREASE-KEY}}$

Q	$T_{\text{EXTRACT-MIN}}$	$T_{\text{DECREASE-KEY}}$	Total
array	$O(n)$	$O(1)$	$O(n^2)$
binary heap	$O(\lg n)$	$O(\lg n)$	$O(m \lg n)$
Fibonacci heap	$O(\lg n)$ amortized	$O(1)$ amortized	$O(m + n \lg n)$ worst case