BITS Pilani
Pilani Campus

Data Structures & Algorithms Design- SS ZG519
Lecture - 16

Dr. Padma Murali

BITS Pilani
Pilani Campus

- **Shortest Paths**

Shortest Paths

- Dijkstra's algorithm
- Floyd Warshall's algorithm

Shortest Path Problems

- How can we find the shortest route between two points on a map?
- Model the problem as a graph problem:
  - Road map is a weighted graph:

    vertices = cities

    edges = road segments between cities

    edge weights = road distances
  - Goal: find a shortest path between two vertices (cities)

Shortest Path Problems

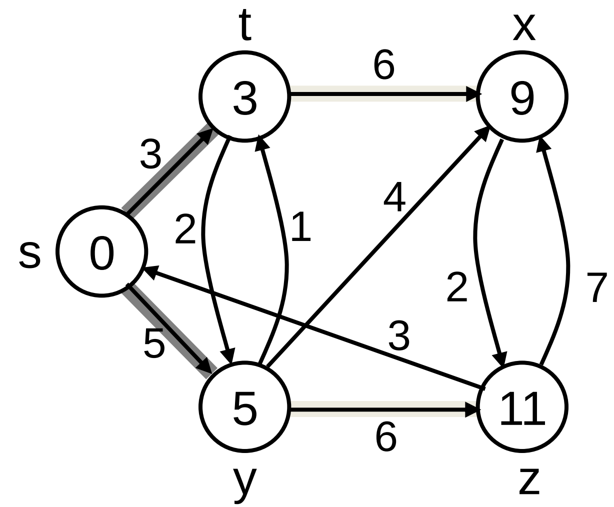
- **Input:**

  Directed graph $G = (V, E)$

  Weight function $w : E \rightarrow \mathbf{R}$

- **Weight of path** $p = \langle v_0, v_1, \ldots, v_k \rangle$

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



- **Shortest-path weight** from u to v:

$$\delta(u, v) = \begin{cases} \min \left\{ w(p) : u \overset{p}{\rightsquigarrow} v \right\} & \text{if there exists a path from } u \text{ to } v \\ \infty & \text{otherwise} \end{cases}$$

- Shortest path u to v is any path p such that $w(p) = \delta(u, v)$

Variants of Shortest Paths

Single-source shortest path

- G = (V, E) $\Rightarrow$ find a shortest path from a given source vertex *s* to each vertex $v \in V$

Single-destination shortest path

- Find a shortest path to a given destination vertex **t** from each vertex v
- Reverse the direction of each edge $\Rightarrow$ single-source

Single-pair shortest path

- Find a shortest path from u to v for given vertices u and v
- Solve the single-source problem

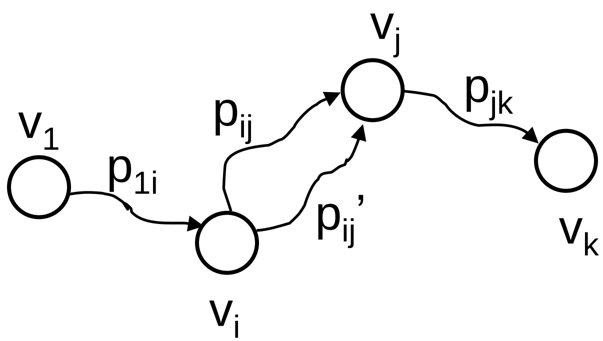
All-pairs shortest-paths

- Find a shortest path from u to v for every pair of vertices u and v

Optimal Substructure of Shortest Paths

Given:

- A weighted, directed graph $G = (V, E)$
- A weight function $w: E \to \mathbf{R}$,
- A shortest path $p = \langle v_1, v_2, \ldots, v_k \rangle$ from v_1 to v_k
- A subpath of p: $p_{ij} = \langle v_i, v_{i+1}, \ldots, v_j \rangle$, with $1 \le i \le j \le k$

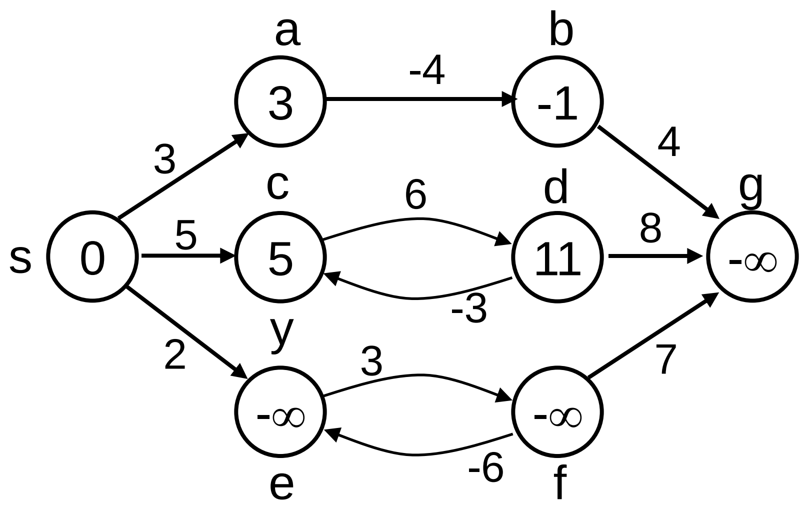


Then: p_{ij} is a shortest path from v_i to v_j

Negative-Weight Edges

What if we have negative-weight edges?

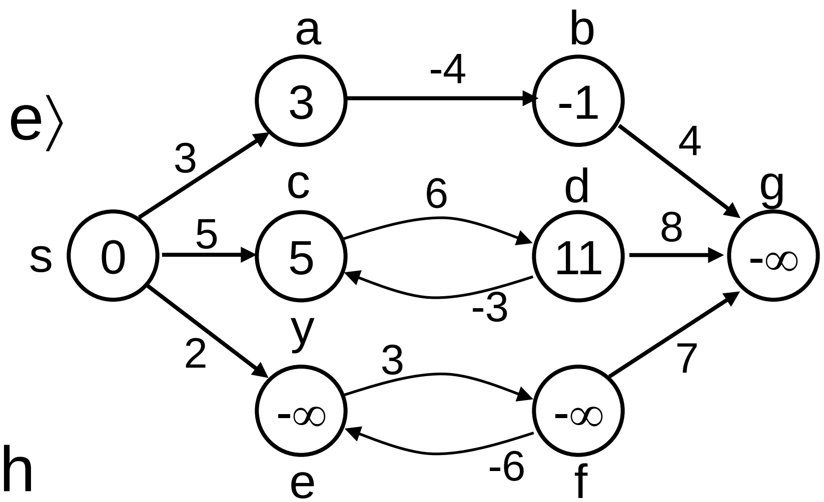
- $s \rightarrow a$: only one path
  - $\delta(s, a) = w(s, a) = 3$
- $s \rightarrow b$: only one path
  - $\delta(s, b) = w(s, a) + w(a, b) = -1$
- $s \rightarrow c$: infinitely many paths
  - $\langle s, c \rangle, \langle s, c, d, c \rangle, \langle s, c, d, c, d, c \rangle$
  - cycle has positive weight (6 - 3 = 3)
  - $\langle s, c \rangle$ is shortest path with weight $\delta(s, c) = w(s, c) = 5$

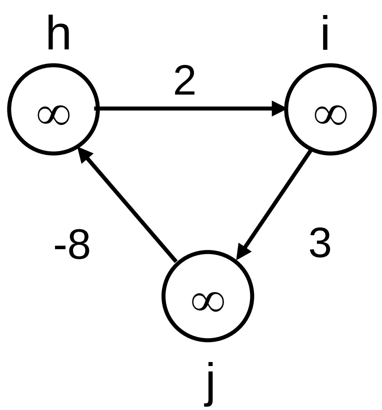


Negative-Weight Edges

- $s \rightarrow e$: infinitely many paths:
  - $\langle s, e \rangle$, $\langle s, e, f, e \rangle$, $\langle s, e, f, e, f, e \rangle$
  - cycle $\langle e, f, e \rangle$ has negative weight:

    $3 + (-6) = -3$
  - can find paths from *s* to *e* with arbitrarily large negative weights
  - $\delta(s, e) = -\infty \Rightarrow$ no shortest path exists between *s* and *e*
  - Similarly: $\delta(s, f) = -\infty$, $\delta(s, g) = -\infty$

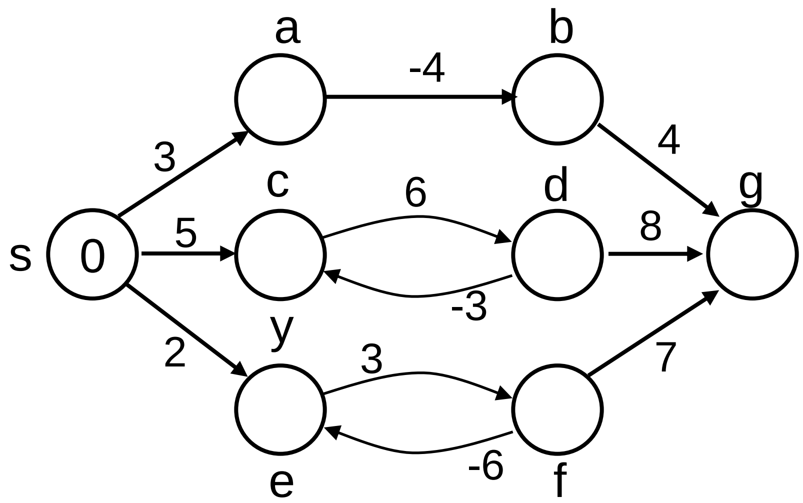




h, i, j not reachable from s

$\delta(s, h) = \delta(s, i) = \delta(s, j) = \infty$

Negative-Weight Edges



- Negative-weight edges may form negative-weight cycles
- If such cycles are reachable from the source: $\delta(s, v)$ is not properly defined
  - Keep going around the cycle, and get $w(s, v) = -\infty$ for all v on the cycle

Cycles

Can shortest paths contain cycles?

Negative-weight cycles No!

Positive-weight cycles: No!

- By removing the cycle we can get a shorter path

Zero-weight cycles

- No reason to use them
- Can remove them to obtain a path with similar weight

We will assume that when we are finding shortest paths, the paths will have no cycles

Shortest-Path Representation

For each vertex $v \in V$:

$d[v] = \delta(s, v)$: a **shortest-path estimate**

– Initially, $d[v]=\infty$

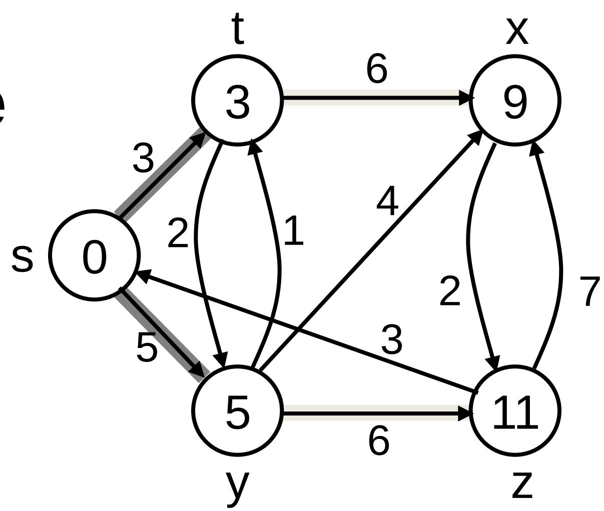
– Reduces as algorithms progress

$\pi[v]$ = **predecessor** of v on a shortest path from ***s***

– If no predecessor, $\pi[v]$ = NIL

– π induces a tree—**shortest-path tree**

Shortest paths & shortest path trees are not unique



Initialization

Alg.: INITIALIZE-SINGLE-SOURCE(V, s)

1. **for** each $v \in V$
2. **do** $d[v] \leftarrow \infty$
3. $\pi[v] \leftarrow$ NIL
4. $d[s] \leftarrow 0$

All the shortest-paths algorithms start with INITIALIZE-SINGLE-SOURCE

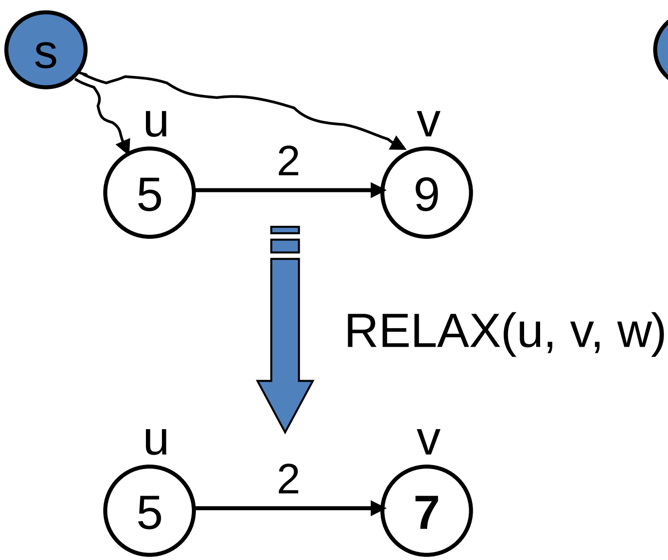
Relaxation

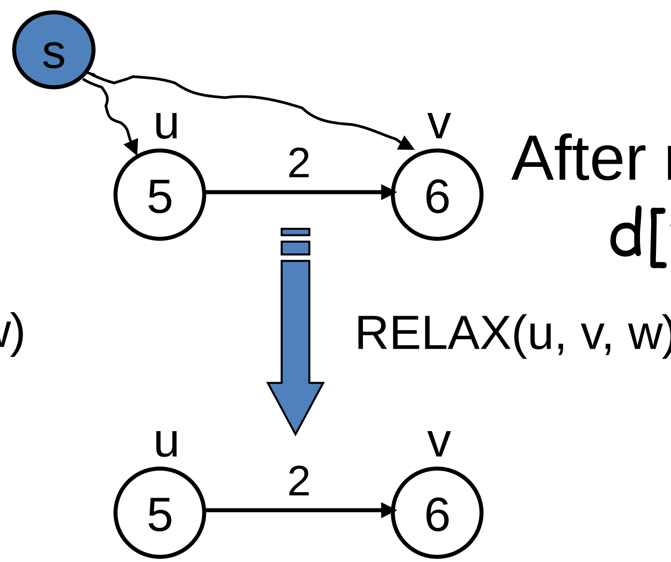
- **Relaxing** an edge (u, v) = testing whether we can improve the shortest path to v found so far by going through u

  If d[v] > d[u] + w(u, v)

  we can improve the shortest path to v

  $\Rightarrow$ update d[v] and π[v]





After relaxation:

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

RELAX(u, v, w)

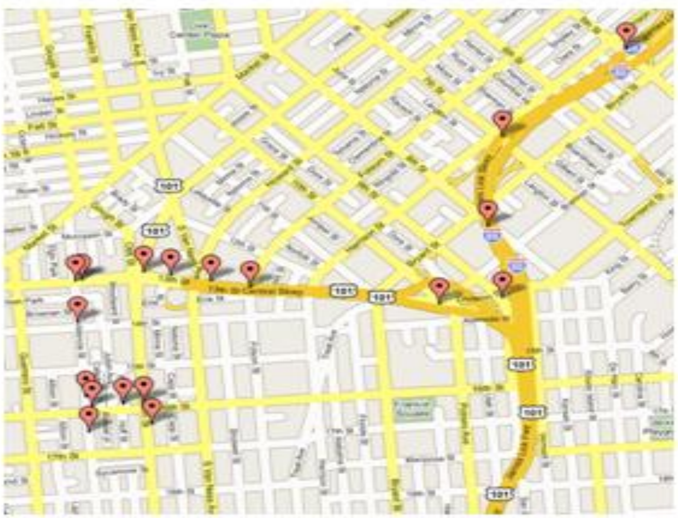
1. **if** $d[v] > d[u] + w(u, v)$
2. **then** $d[v] \leftarrow d[u] + w(u, v)$
3. $\pi[v] \leftarrow u$

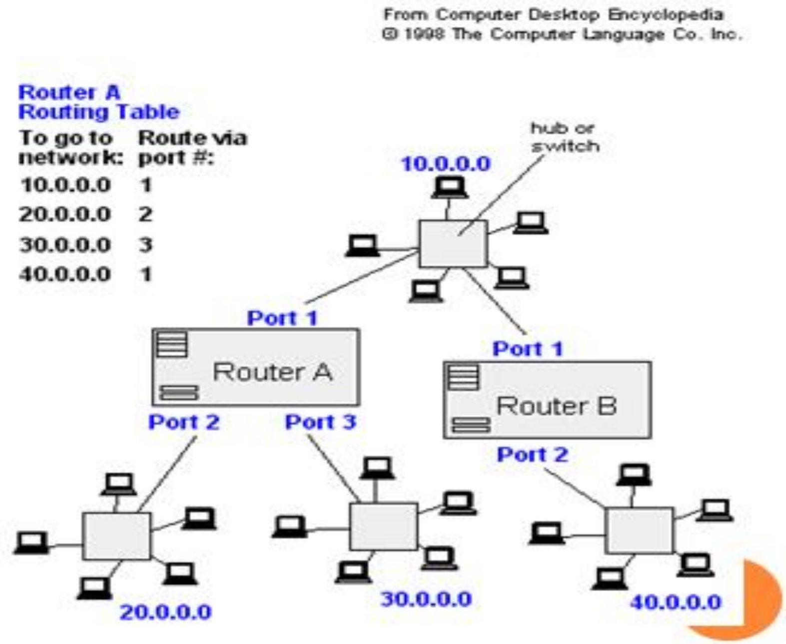
- All the single-source shortest-paths algorithms
  - start by calling INIT-SINGLE-SOURCE
  - then relax edges
- The algorithms differ in the order and how many times they relax each edge

Shortest Path Problem- Dijkstra’s Algorithm

Applications of Dijkstra's Algorithm

- Traffic Information Systems are most prominent use
- Mapping (Map Quest, Google Maps)
- Routing Systems

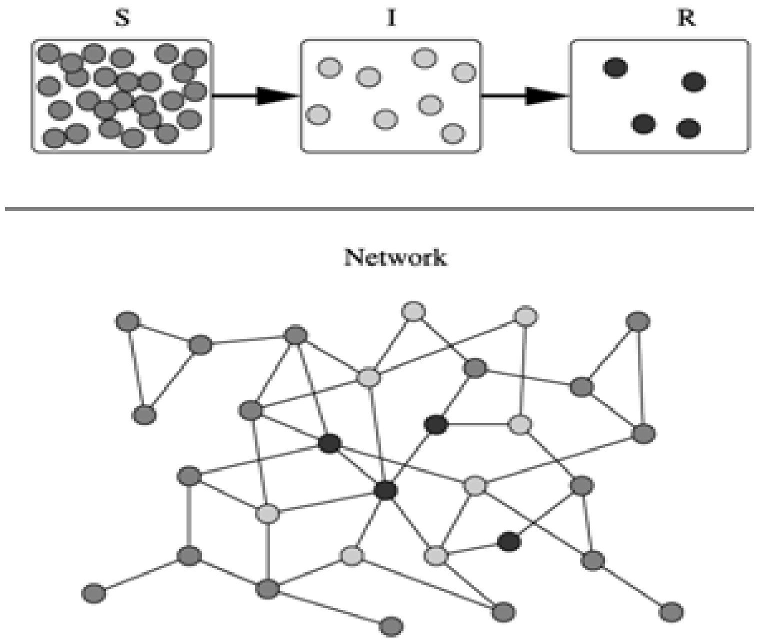




Shortest Path Problem- Dijkstra's Algorithm

Applications of Dijkstra's Algorithm

- epidemiology
- Prof. Lauren Meyers (Biology Dept.) uses networks to model the spread of infectious diseases and design prevention and response strategies.
- Vertices represent individuals, and edges their possible contacts. It is useful to calculate how a particular individual is connected to others.
- Knowing the shortest path lengths to other individuals can be a relevant indicator of the potential of a particular individual to infect others.



Shortest Path Problem- Dijkstra's Algorithm

* Single-source shortest-paths problem
* For a given vertex called the source in a weighted connected graph, find shortest paths to all its other vertices.
* This algorithm is applicable to graphs with nonnegative weights only.

Shortest Path Problem- Dijkstra's Algorithm

* Dijkstra's algorithm finds shortest paths to a graph's vertices in order of their distance from a given source.
* First, it finds the shortest path from the source to a vertex nearest to it, then to a second nearest, and so on.

Shortest Path Problem- Dijkstra's Algorithm

* In general, before its i^{th} iteration, the algorithm has identified the shortest paths to $i-1$ other vertices nearest to the source.
* Example of a greedy algorithm.

Shortest Path Problem- Dijkstra's Algorithm

Shortest Path Algorithm

Dijkstra's algorithm for getting a shortest path from a vertex to every other vertex in a connected weighted graph.

..tex with

Shortest Path Problem- Dijkstra's Algorithm

weigh…

Step 1 Let u_1 be the vertex with which we are going to start and are going to find the shortest path to every other vertex. Assign u_1 a permanent label 0. Assign every other vertex a temporary label ∞.

… vertex has been … the

innovate achieve lead

Shortest Path Problem- Dijkstra's Algorithm

vertex ...

Step 2 until every vertex has been assigned a permanent label, do the following:

(i) Take the vertex u_i which has most recently got a permanent label

→ say d. " label is d ".

For each vertex v which is adjacent to u_i and has not received a permanent label if $d + weight(u_i, v) < \lambda(v)$

→ the temporary label of v.

Shortest Path Problem- Dijkstra's Algorithm

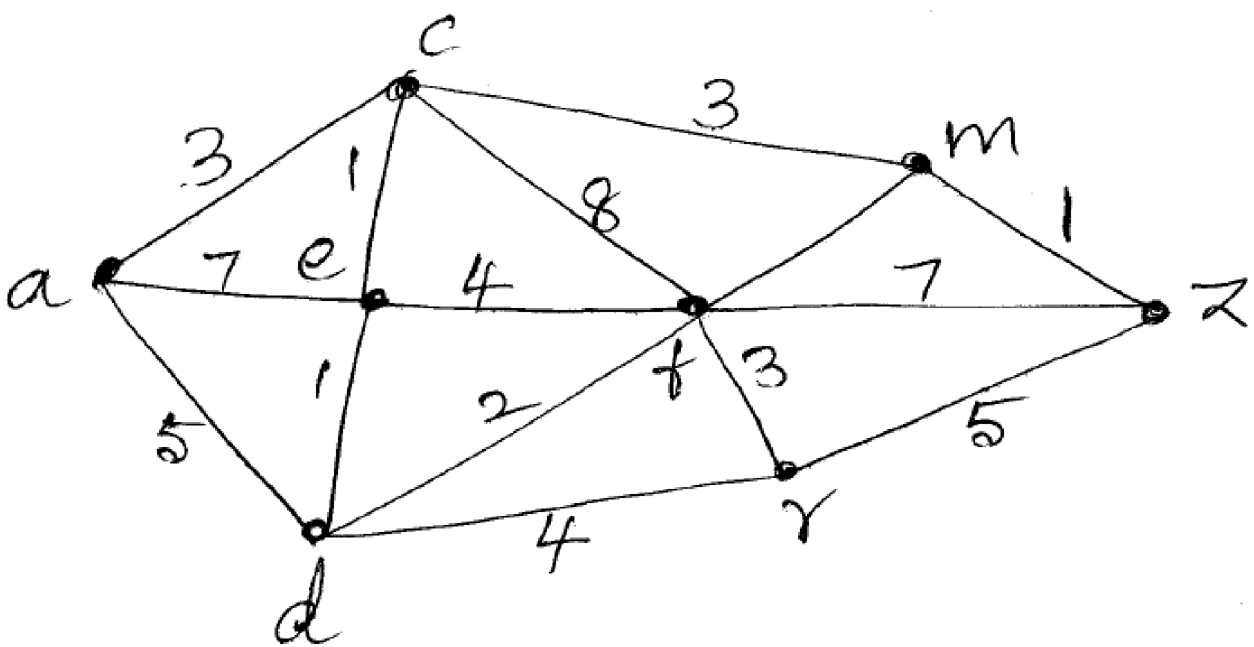
change the temporary label of v to d + weight(u_i, v). Otherwise label of v is unchanged.

(ii) Take a vertex v which has a temporary label, smallest among all temporary labels in the graph. Set $u_i + 1 = v$ and make its' label permanent. If there is a tie for smallest temporary level, choose any of them for permanent label.

Step 3 Algorithm ends only if all vertices have got a permanent label.

Example 1

Find shortest paths and their lengths from vertex a to every other vertex



Example 1

Recent Permanent Labelled Vertex	$\lambda(a)$	$\lambda(c)$	$\lambda(e)$	$\lambda(d)$	$\lambda(f)$	$\lambda(m)$	$\lambda(r)$	$\lambda(z)$
	$\boxed{0}$	∞	∞	∞	∞	∞	∞	∞
a	$\boxed{0}$	$\boxed{3}$	7	5	∞	∞	∞	∞
c	$\boxed{0}$	$\boxed{3}$	$\boxed{4}$	5	11	6	∞	∞
e	$\boxed{0}$	$\boxed{3}$	$\boxed{4}$	$\boxed{5}$	8	6	∞	∞
d	$\boxed{0}$	$\boxed{3}$	$\boxed{4}$	$\boxed{5}$	7	$\boxed{6}$	9	∞
m	$\boxed{0}$	$\boxed{3}$	$\boxed{4}$	$\boxed{5}$	$\boxed{7}$	$\boxed{6}$	9	7
f	$\boxed{0}$	$\boxed{3}$	$\boxed{4}$	$\boxed{5}$	$\boxed{7}$	$\boxed{6}$	9	$\boxed{7}$
z	$\boxed{0}$	$\boxed{3}$	$\boxed{4}$	$\boxed{5}$	$\boxed{7}$	$\boxed{6}$	$\boxed{9}$	$\boxed{7}$

$\boxed{}$ → denotes permanent labelling.

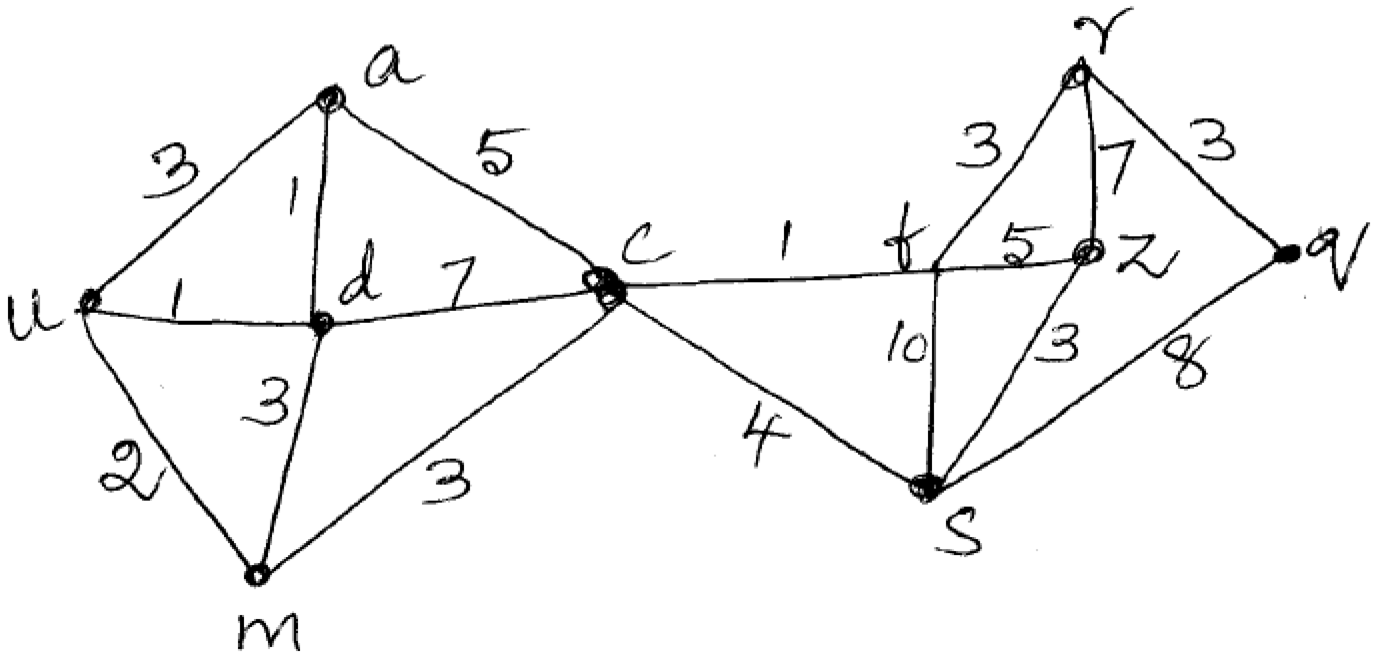
Example 1

The last row of the table denotes the shortest paths to all other vertices from vertex a.

Example 2

Example: 2

Find the shortest path from u to every other vertex.



Example 2

Recent permanent labelled vertex	λ(u)	λ(a)	λ(d)	λ(m)	λ(c)	λ(s)	λ(r)	λ(z)	λ(q)	λ(f)
	$\boxed{0}$	∞	∞	∞	∞	∞	∞	∞	∞	∞
u	$\boxed{0}$	3	$\boxed{1}$	2	∞	∞	∞	∞	∞	∞
d	$\boxed{0}$	$\boxed{2}$	$\boxed{1}$	2	8	∞	∞	∞	∞	∞
a	$\boxed{0}$	$\boxed{2}$	$\boxed{1}$	$\boxed{2}$	7	∞	∞	∞	∞	∞
m	$\boxed{0}$	$\boxed{2}$	$\boxed{1}$	$\boxed{2}$	$\boxed{5}$	∞	∞	∞	∞	∞
c	$\boxed{0}$	$\boxed{2}$	$\boxed{1}$	$\boxed{2}$	$\boxed{5}$	9	∞	∞	∞	$\boxed{6}$
f	$\boxed{0}$	$\boxed{2}$	$\boxed{1}$	$\boxed{2}$	$\boxed{5}$	$\boxed{9}$	9	11	∞	$\boxed{6}$
s	$\boxed{0}$	$\boxed{2}$	$\boxed{1}$	$\boxed{2}$	$\boxed{5}$	$\boxed{9}$	$\boxed{9}$	11	17	$\boxed{6}$
r	$\boxed{0}$	$\boxed{2}$	$\boxed{1}$	$\boxed{2}$	$\boxed{5}$	$\boxed{9}$	$\boxed{9}$	$\boxed{11}$	12	$\boxed{6}$
z	$\boxed{0}$	$\boxed{2}$	$\boxed{1}$	$\boxed{2}$	$\boxed{5}$	$\boxed{9}$	$\boxed{9}$	$\boxed{11}$	$\boxed{12}$	$\boxed{6}$
q	same as above row.									

Dijkstra's Algorithm

Single-source shortest path problem:

- No negative-weight edges: $w(u, v) > 0 \ \forall \ (u, v) \in E$

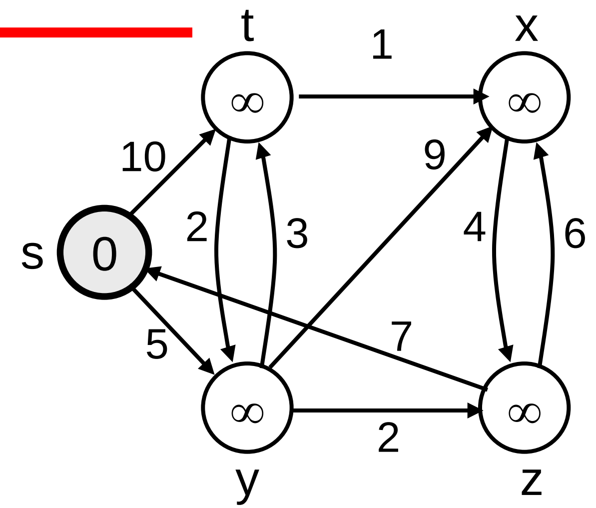
Maintains two sets of vertices:

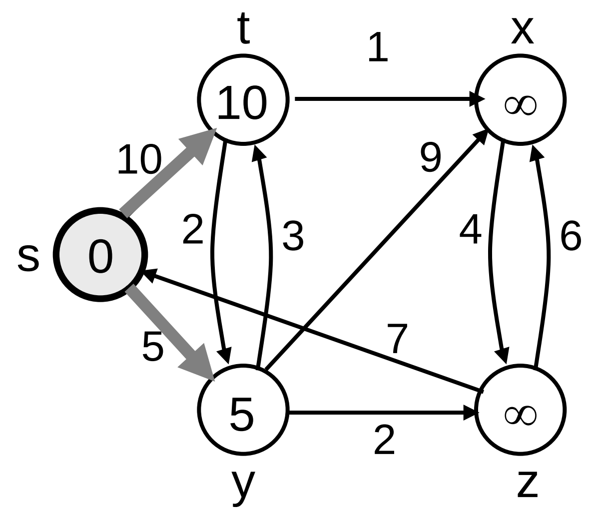
- S = vertices whose final shortest-path weights have already been determined
- Q = vertices in V – S: min-priority queue
    - Keys in Q are estimates of shortest-path weights (d[v])

Repeatedly select a vertex $u \in V - S$, with the minimum shortest-path estimate d[v]

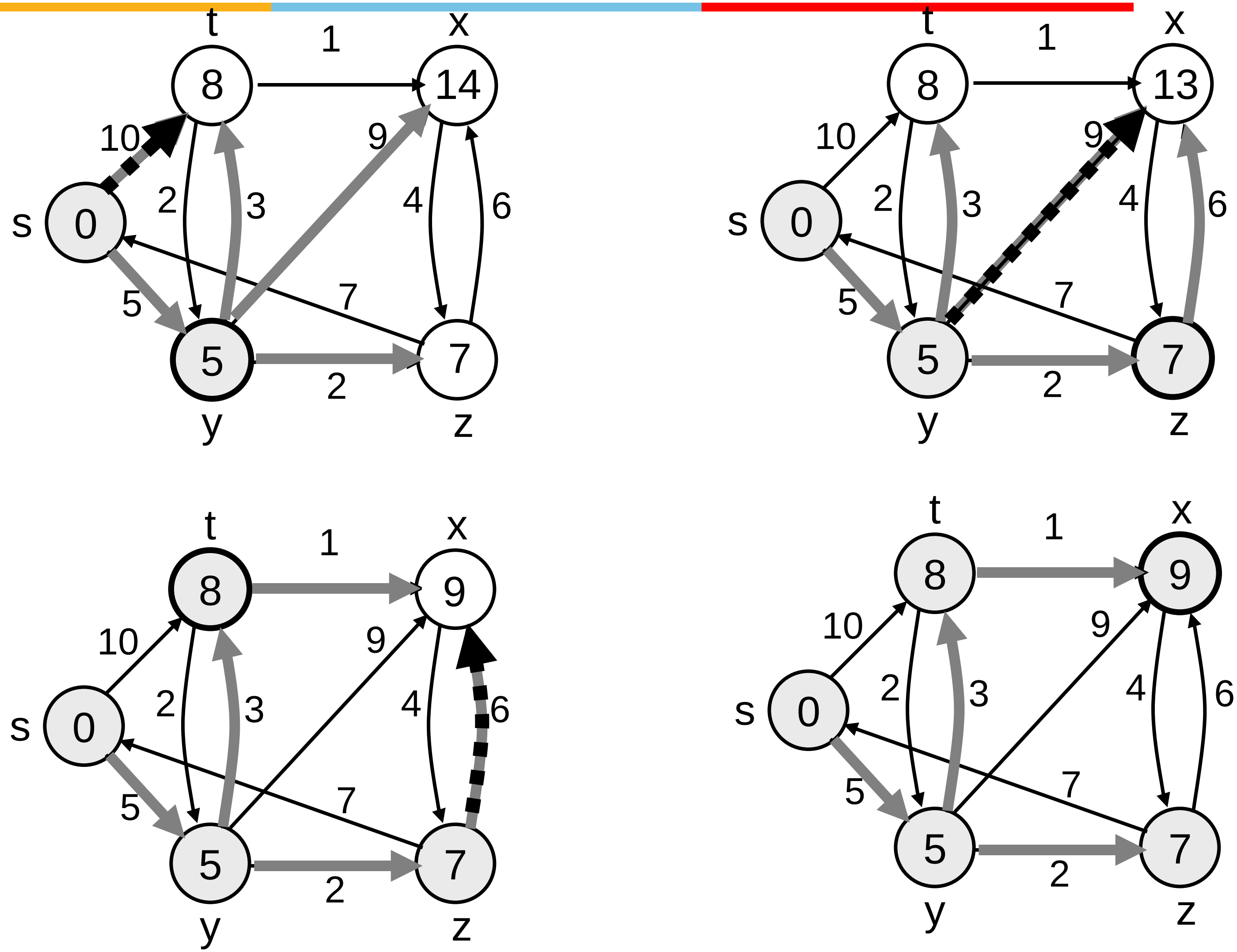
Dijkstra (G, w, s)

```
INITIALIZE-SINGLE-SOURCE(V, s)
S ← ∅
Q ← V[G]
while Q ≠ ∅
    do u ← EXTRACT-MIN(Q)
        S ← S ∪ {u}
        for each vertex v ∈ Adj[u]
            do RELAX(u, v, w)
```





Example



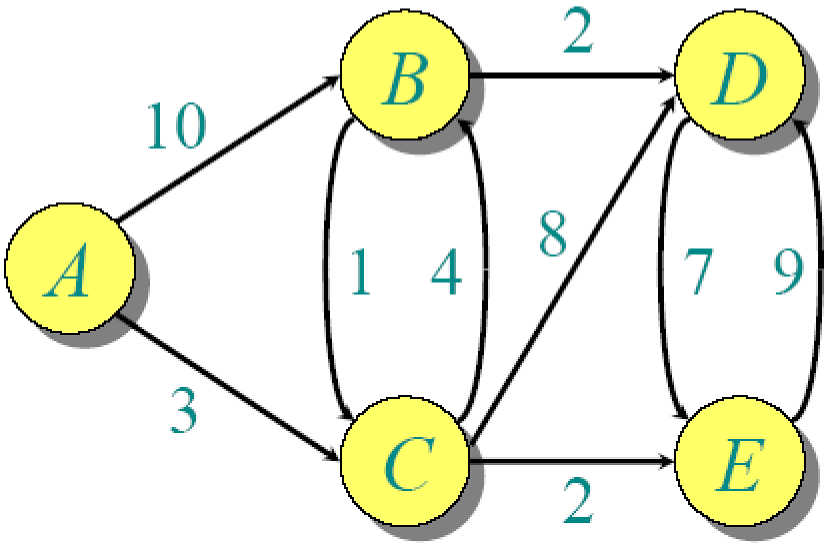
t x s y z
8 14 0 5 7
1 10 2 3 9 4 6 5 7 2
t x s y z
8 13 0 5 7
1 10 2 3 9 4 6 5 7 2
t x s y z
8 9 0 5 7
1 10 2 3 9 4 6 5 7 2
t x s y z
8 9 0 5 7
1 10 2 3 9 4 6 5 7 2

Dijkstra (G, w, s)

1. INITIALIZE-SINGLE-SOURCE(V, *s*) $\leftarrow$ $\Theta(V)$
2. S $\leftarrow \varnothing$
3. Q $\leftarrow$ V[G] $\leftarrow$ O(V) build min-heap
4. **while** Q $\neq \varnothing$ $\leftarrow$ Executed O(V) times
5. **do** u $\leftarrow$ EXTRACT-MIN(Q) $\leftarrow$ O(lgV)
6. S $\leftarrow$ S $\cup$ {u}
7. **for** each vertex v $\in$ *Adj*[u]
8. **do** RELAX(u, v, w) $\leftarrow$ O(E) times; O(lgV)

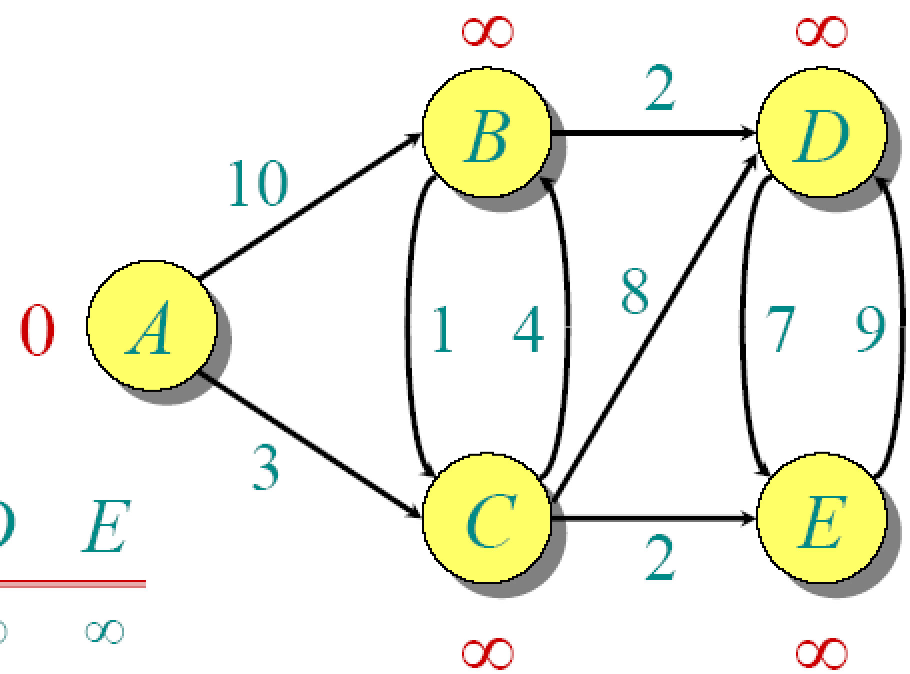
Running time: $O(V\lg V + E\lg V) = O(E\lg V)$

Example of Dijkstra's Algorithm



Example of Dijkstra's Algorithm

Initialize:

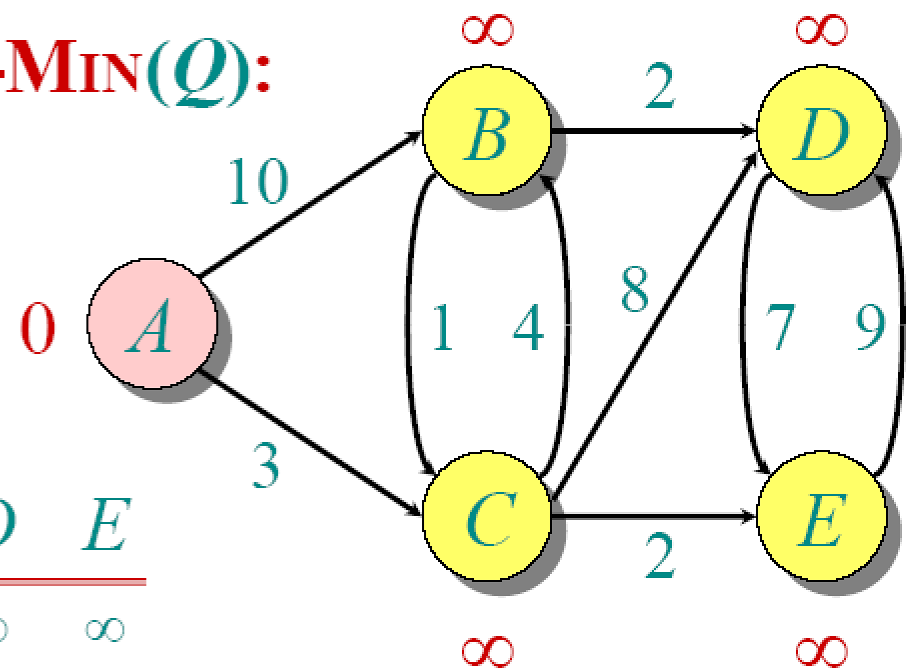


Q:	A	B	C	D	E
	0	∞	∞	∞	∞

$S: \{\}$

Example of Dijkstra's Algorithm

"A" $\leftarrow$ EXTRACT-MIN(Q):

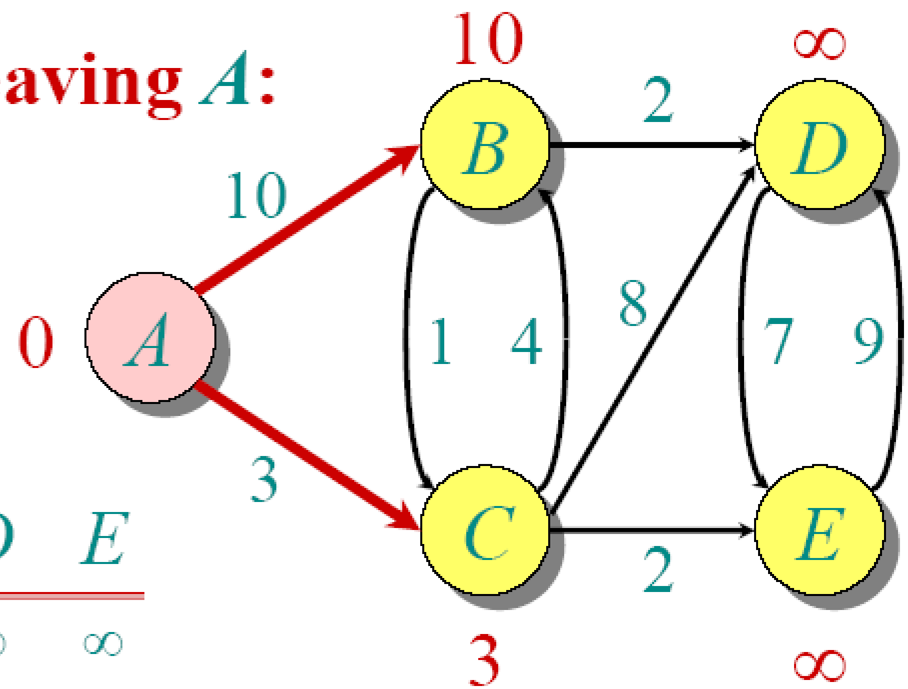


Q:	A	B	C	D	E
	0	∞	∞	∞	∞

S: { A }

Example of Dijkstra's Algorithm

Relax all edges leaving A:

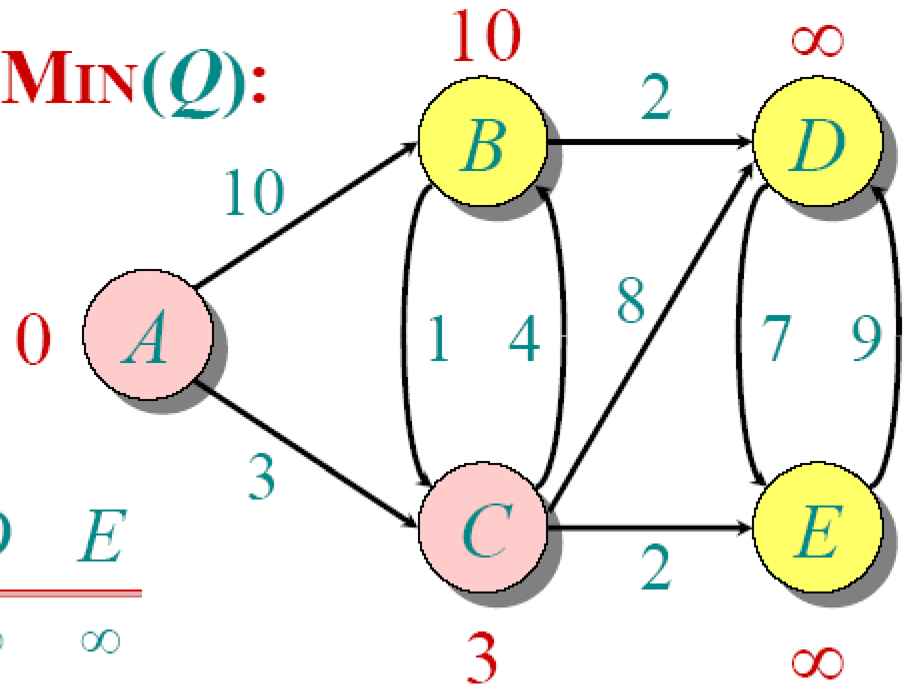


Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞

S: { A }

Example of Dijkstra's Algorithm

"C" ← EXTRACT-MIN(Q):

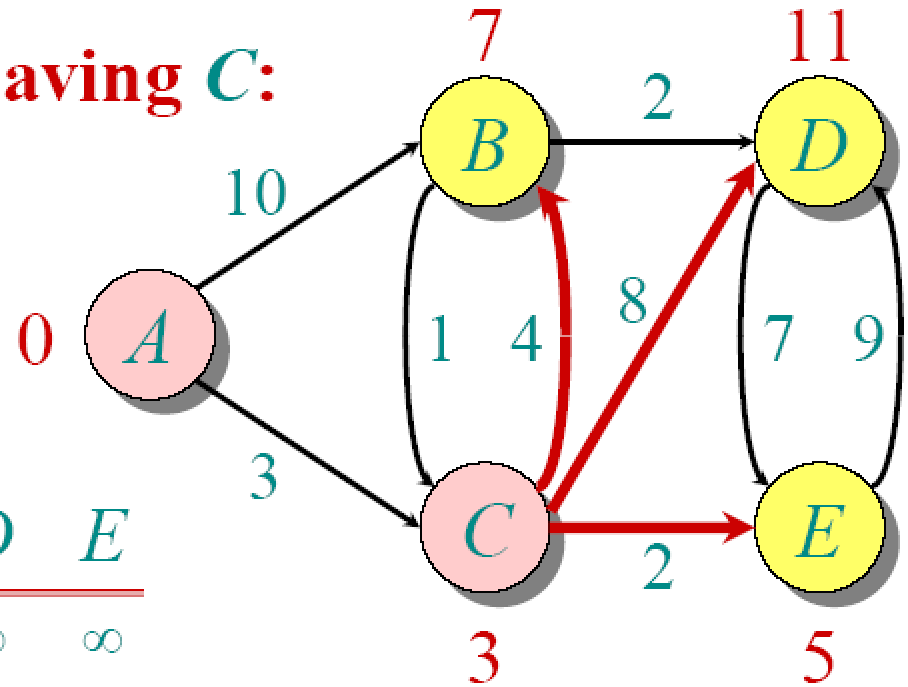


Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞

S: { A, C }

Example of Dijkstra’s Algorithm

Relax all edges leaving *C*:

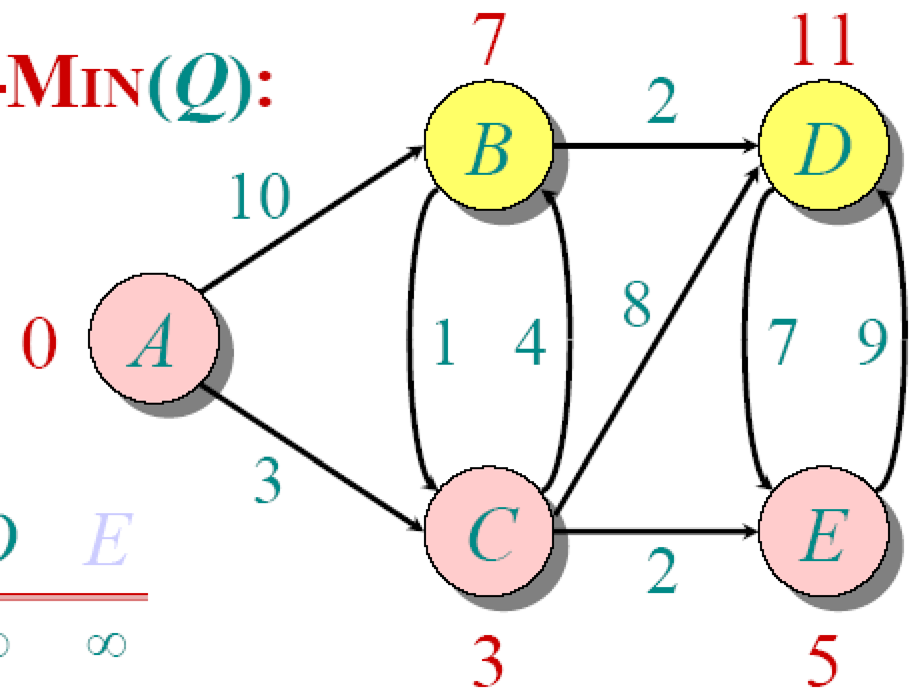


Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5

S: { A, C }

Example of Dijkstra's Algorithm

"E" $\leftarrow$ EXTRACT-MIN(Q):

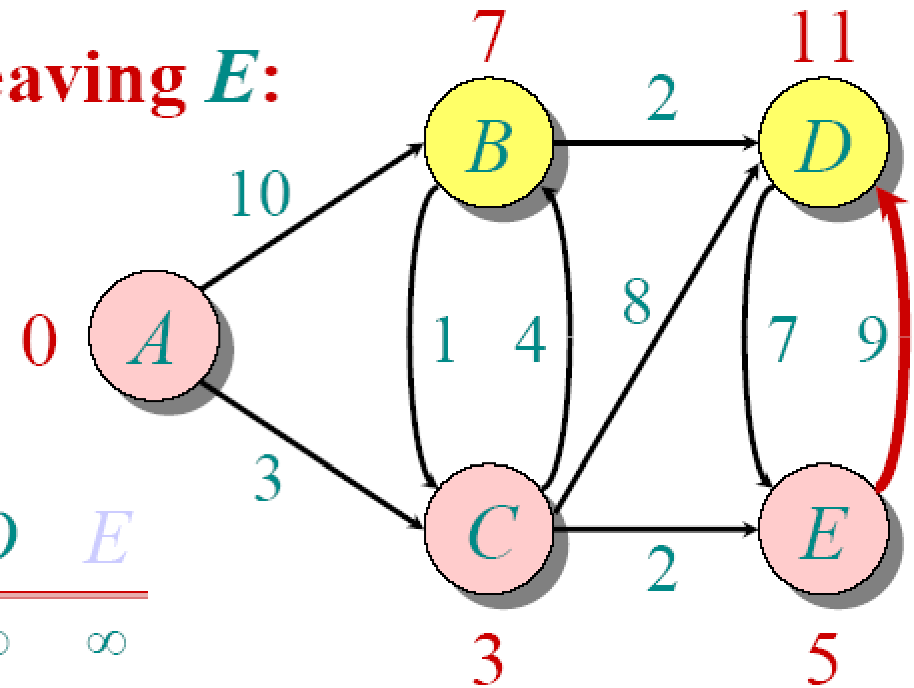


Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5

S: { A, C, E }

Example of Dijkstra's Algorithm

Relax all edges leaving *E*:

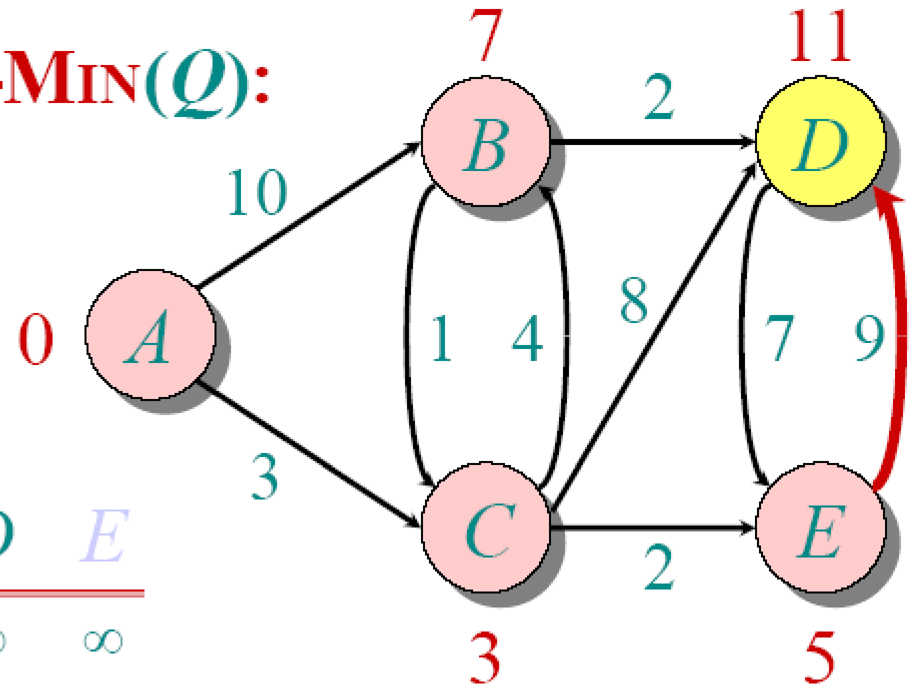


Q:	*A*	*B*	*C*	*D*	*E*
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5
		7		11	

S: { *A*, *C*, *E* }

Example of Dijkstra's Algorithm

"B" $\leftarrow$ EXTRACT-MIN(Q):

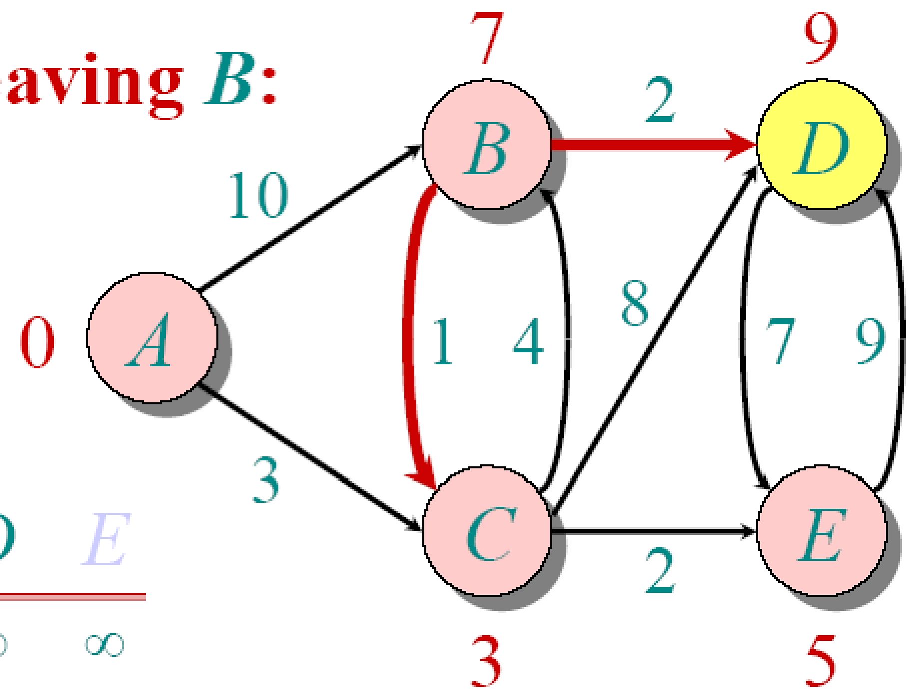


Q:	A	B	C	D	E
	$\boxed{0}$	∞	∞	∞	∞
		10	$\boxed{3}$	∞	∞
		7		11	$\boxed{5}$
		$\boxed{7}$		11	

S: { A, C, E, B }

Example of Dijkstra’s Algorithm

Relax all edges leaving *B*:

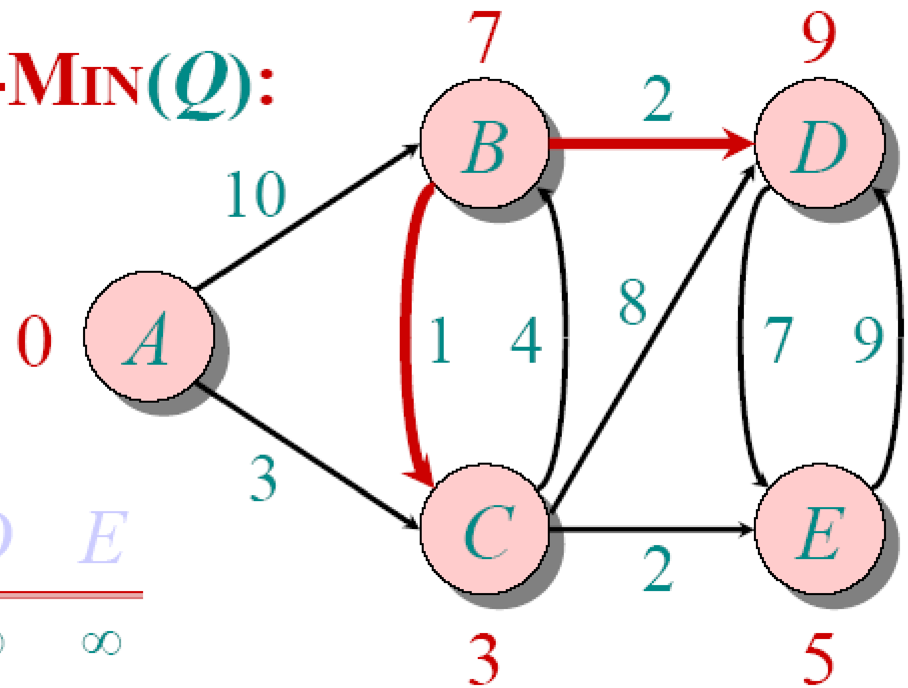


Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5
		7		11	
				9	

S: { A, C, E, B }

Example of Dijkstra's Algorithm

"D" ← EXTRACT-MIN(Q):



Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5
		7		11	
				9	

S: { A, C, E, B, D }

EXAMPLE