BFS and Shortest Path

CMSC 498J: Social Media Computing

Department of Computer Science
University of Maryland
Spring 2015

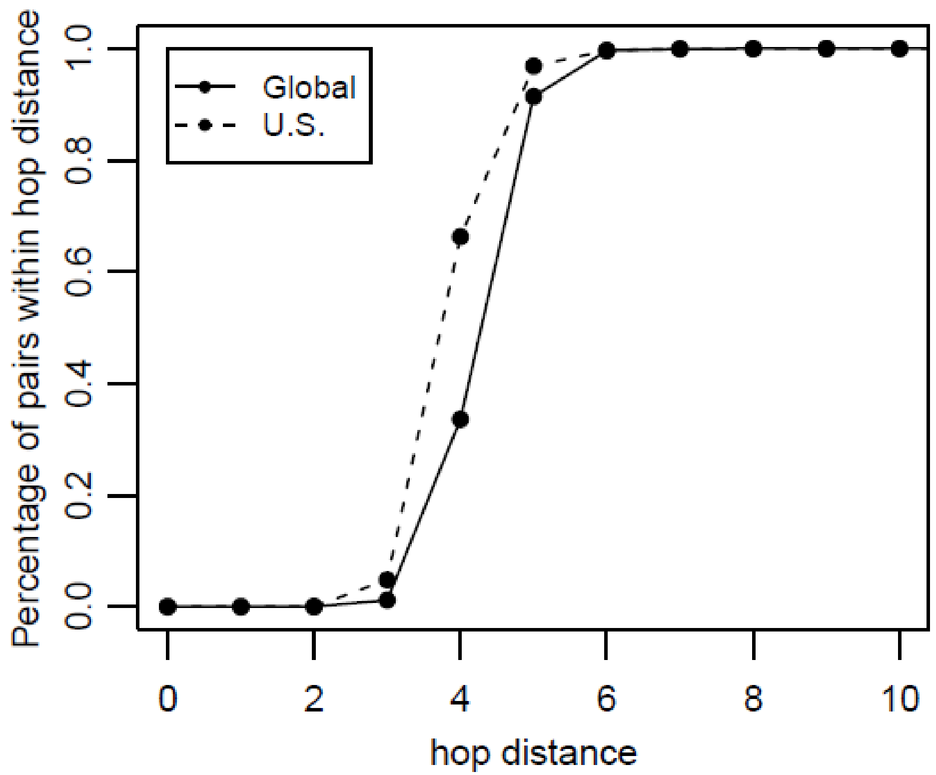
Hadi Amiri
hadi@umd.edu

Lecture Topics

- Connected Components
- Breadth-First Search
- Shortest Path Algorithms
    - Dijkstra's algorithm

Why do we need them?

- Small World Phenomenon



Global
92.0%: within 5 degrees,
99.6%: within six degrees.

U.S. only
96.0%: within 5 degrees,
99.7%: within six degrees.

Figure 2. Diameter. The neighborhood function $N(h)$ showing the percentage of user pairs that are within h hops of each other. The average distance between users on Facebook in May 2011 was 4.7, while the average distance within the U.S. at the same time was 4.3.

Connected Components

- Connected component of a graph is a subset of nodes such that:
  - every node in the subset has a path to every other; and
  - the subset is not part of a bigger component.

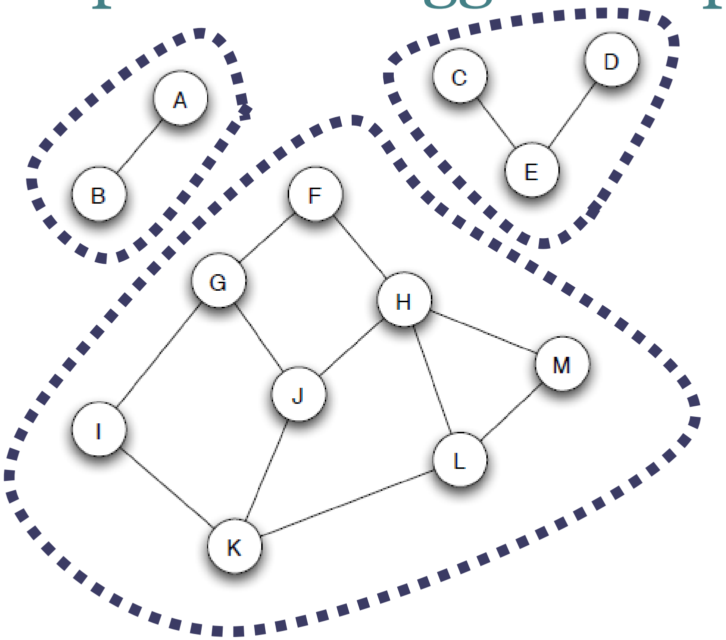


Figure 2.5: A graph with three connected components.

Connected Components

- Connected component of a graph is a subset of nodes such that:
  - every node in the subset has a path to every other; and
  - the subset is not part of a bigger component.

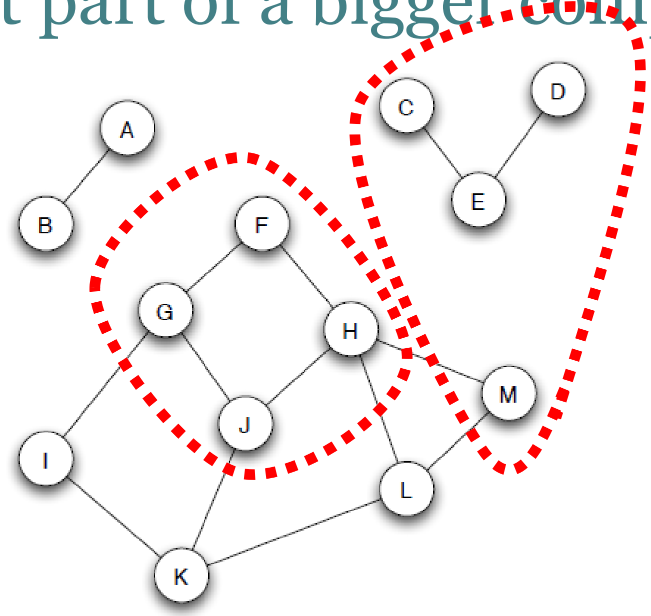


Figure 2.5: A graph with three connected components.

Connected Components- Cnt.

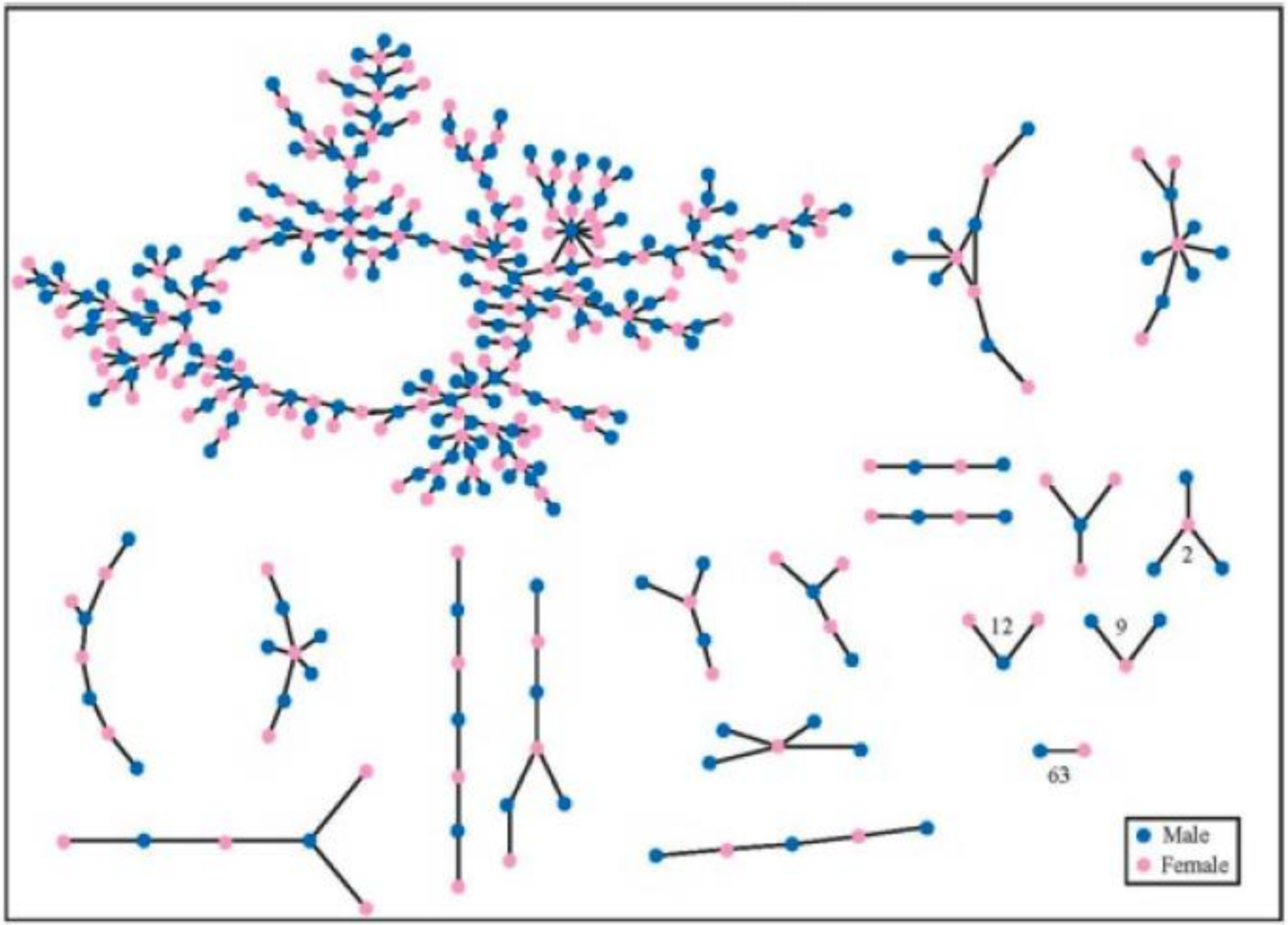


Figure 2.7: A network in which the nodes are students in a large American high school, and an edge joins two who had a romantic relationship at some point during the 18-month period in which the study was conducted [49].

Breadth-First Search

- A general technique for traversing graphs!
  - Start from a given node *s* (i.e. start node) and visit all nodes and edges in the graph.
- Determines whether graph is connected!
- Computes the connected components of graph!
- Find shortest path btw *s* and other nodes (**in terms of number of edges**)

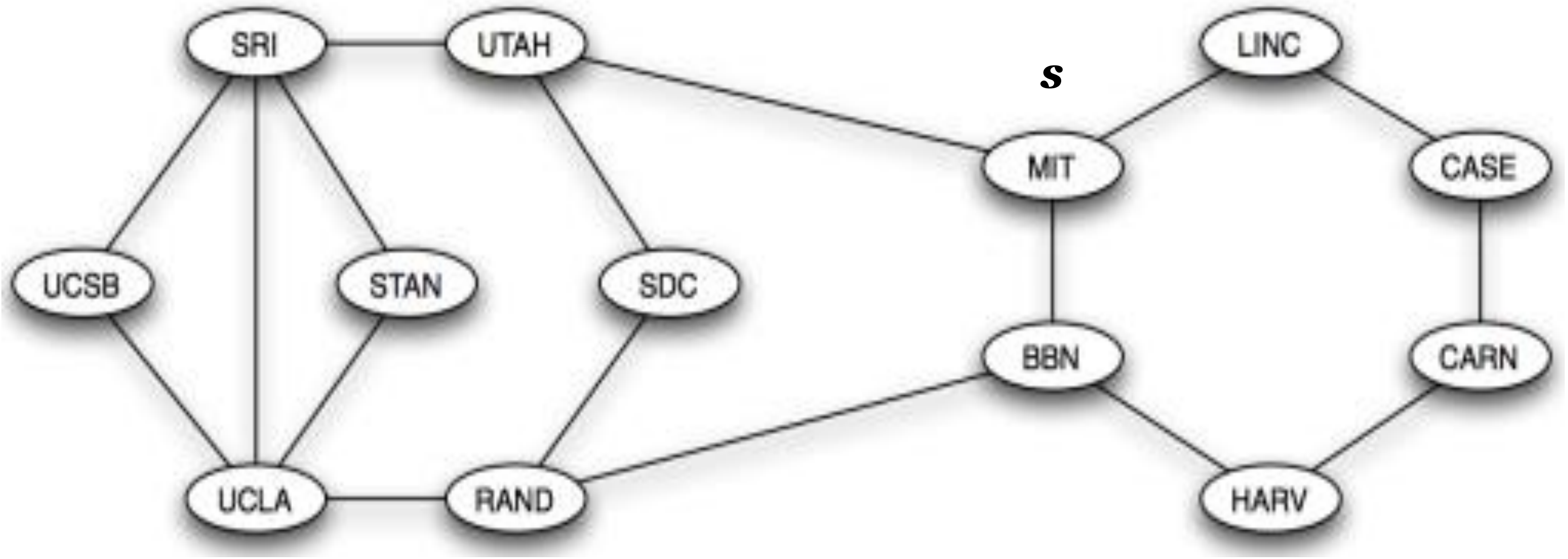
Breadth-First Search- Cnt.

- Start with *s*
- Visit all neighbors of *s* (these are called level-1 nodes)
- Visit all neighbors of level-1 nodes (these are called level-2 nodes)
- and so on.
- Each Node is only visited once.

- Key Point:
  - All level-k nodes should be visited before any level-(k+1) node!

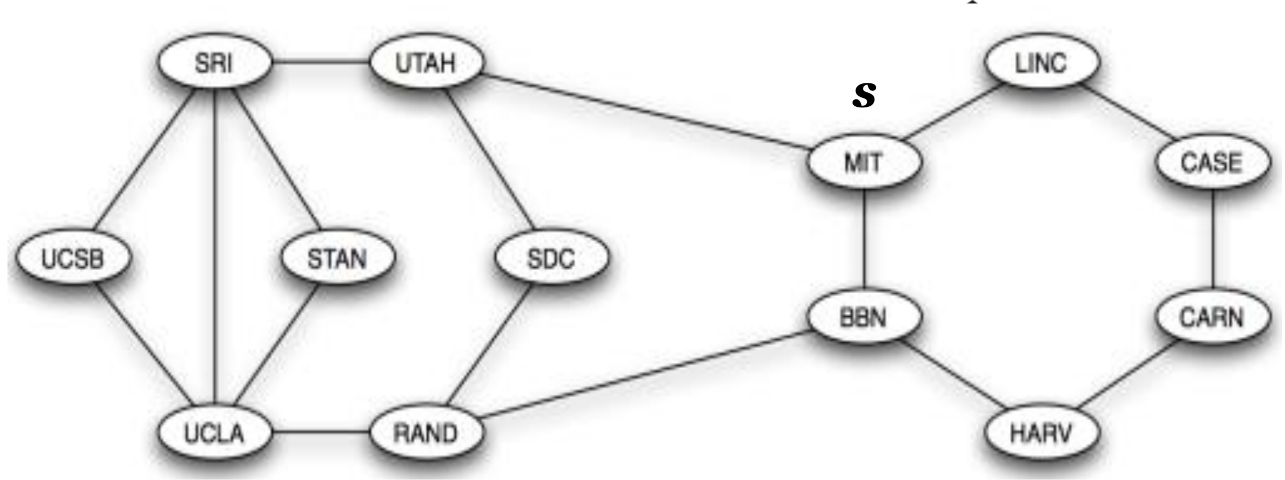
Read CLRS Ch. 22 for detail algorithm and analysis.

Example 1. BFS

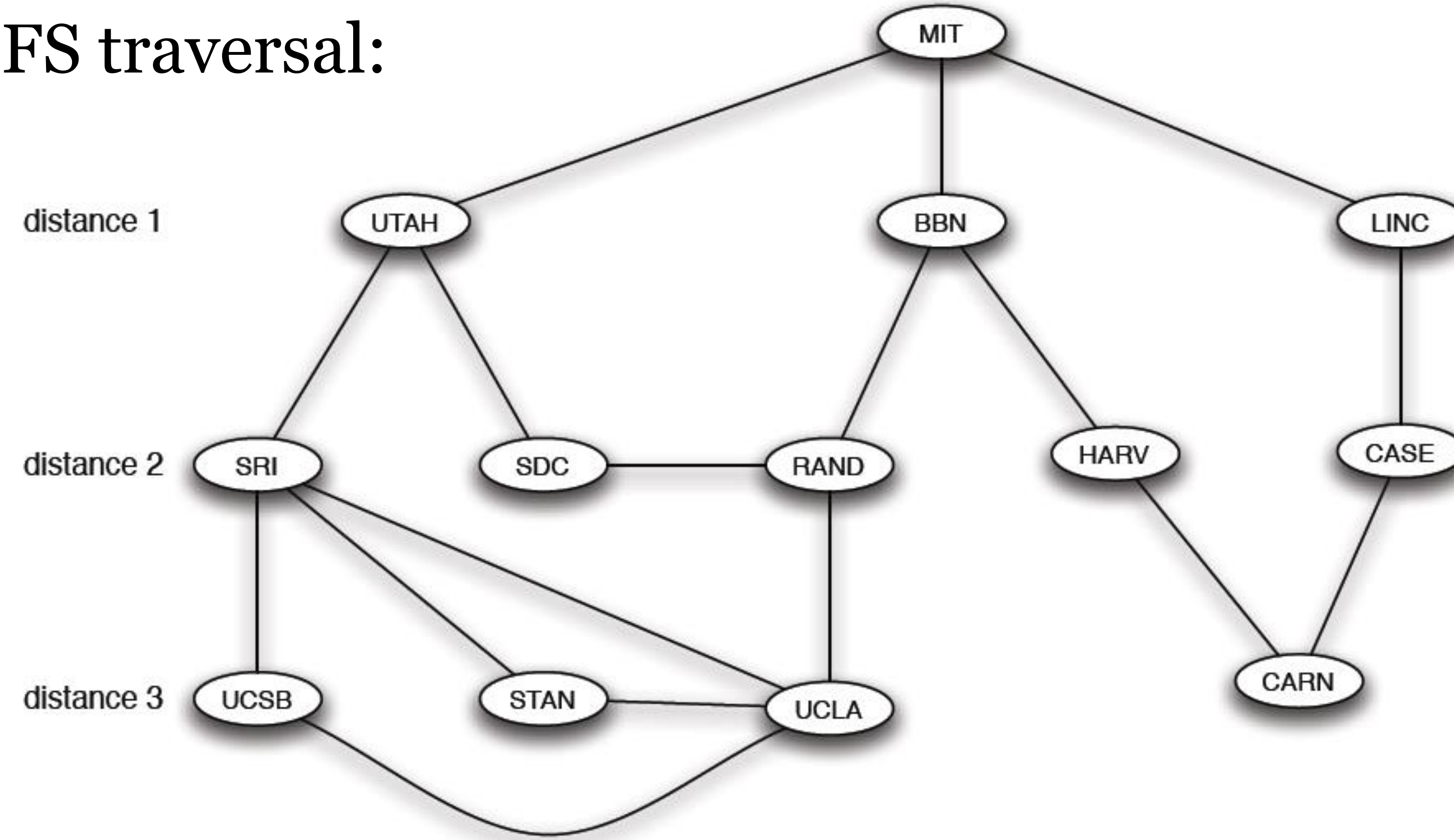


Example 1.

- Graph G:



- Its BFS traversal:

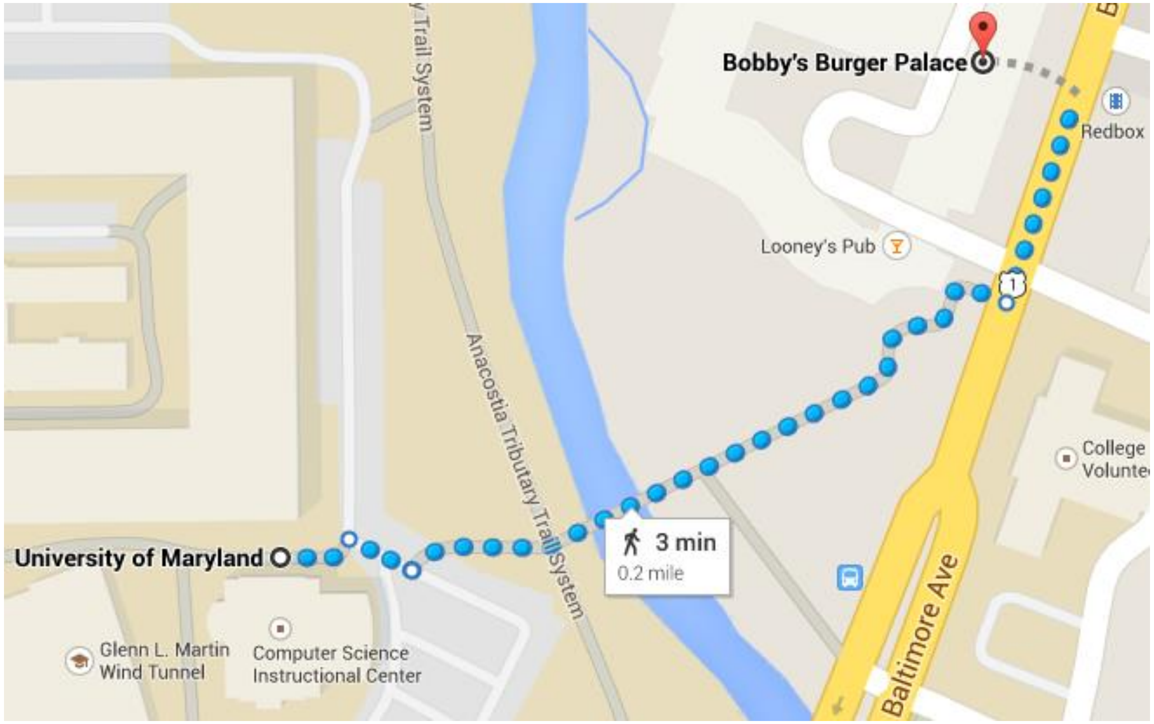


Weighted Graphs

- Edges may carry additional information
  - In a weighted graph each edge has an associated numerical value called the weight of the edge!
    - Distance → btw two places.
    - Delay → btw two metro stations.
    - Cost → btw two flight destinations, and
    - Etc.

Shortest Path Algorithms

- Given a weighted graph and two nodes *s* and *t*, find the shortest path from *s* to *t*.
  - Cost of path = sum of edge weights in path



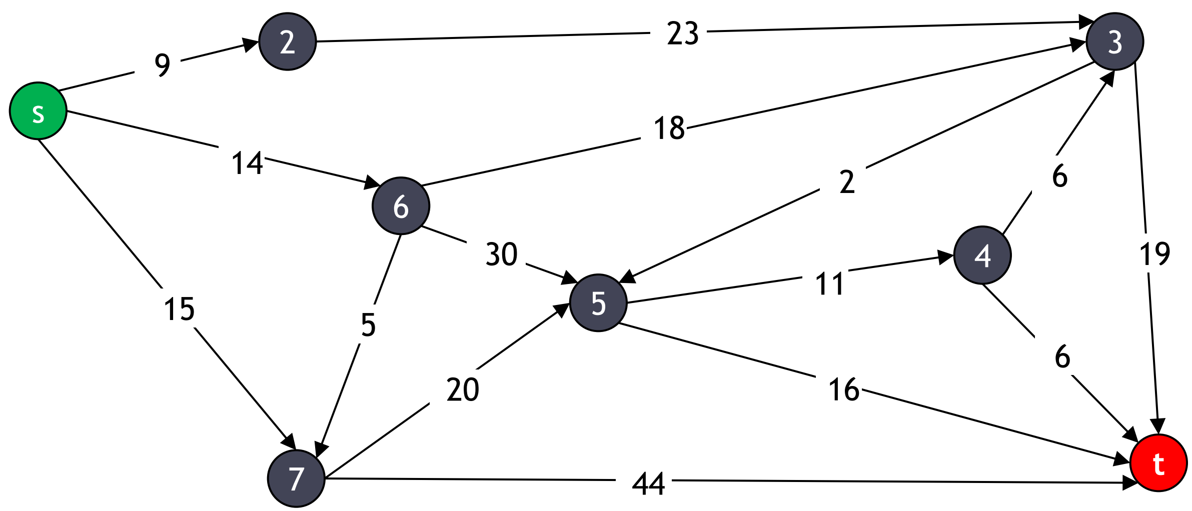
Shortest path from our class to Bobby's Burger

Shortest Path Algorithms- Cnt.

COMPUTER SCIENCE
UNIVERSITY OF MARYLAND

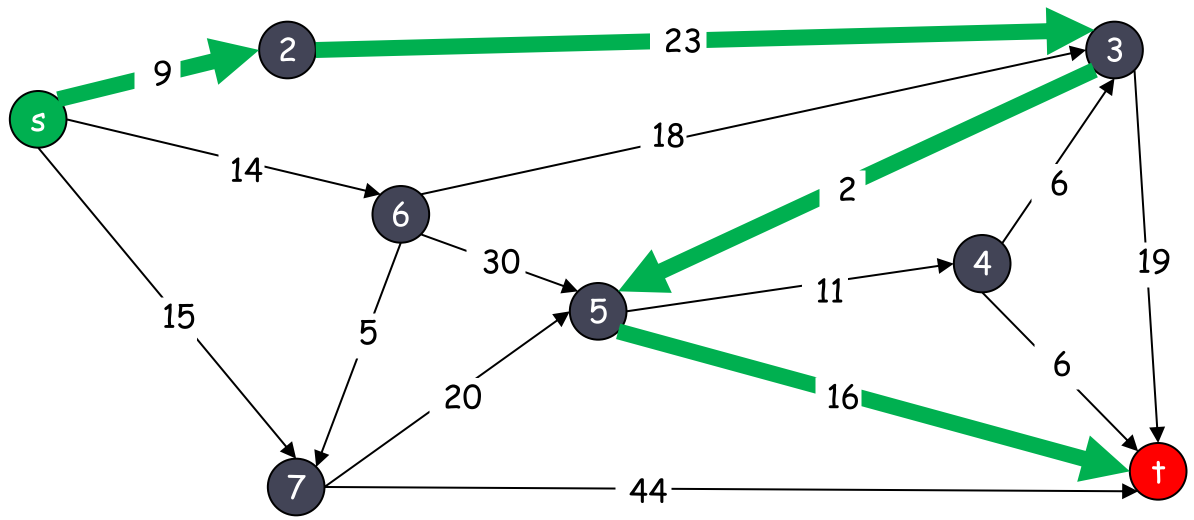
- Dijkstra’s algorithm
- The Bellman-Ford algorithm
- The Floyd-Warshall algorithm
- Johnson's algorithm
- Etc.

Shortest Path Algorithms- Cnt.



- Shortest path from *s* to *t*?

Shortest Path Algorithms- Cnt.



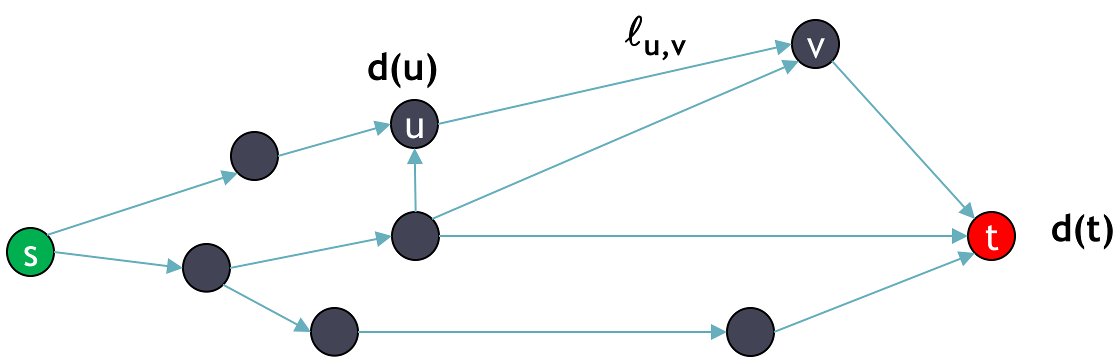
- Shortest Path= s-2-3-5-t
- Cost of path = 9 + 23 + 2 + 16 = 48.

Shortest Path Algorithms- Cnt.

- Applications
  - Small World Phenomenon
  - Internet packet routing
  - Flight reservations
  - Driving directions
  - …

Dijkstra's algorithm

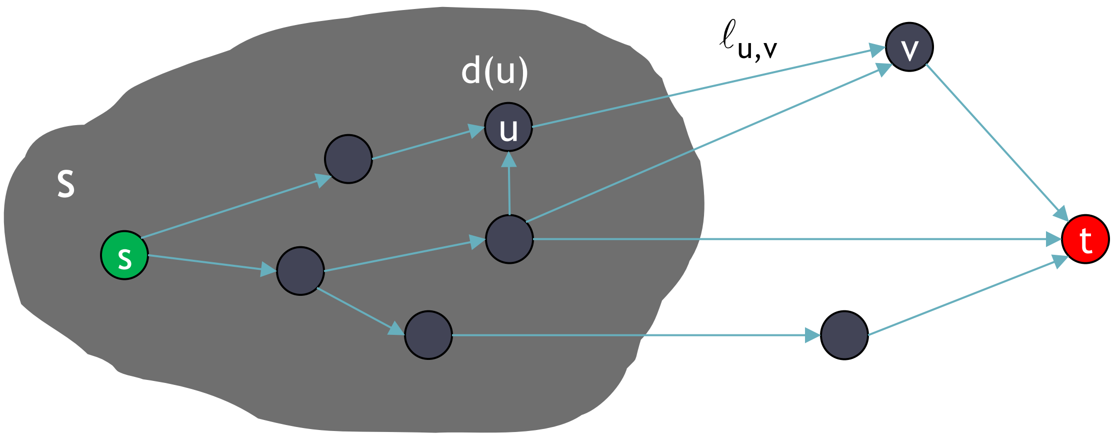
- Weighted Directed graph G = (**N**, **E**),
  - s: Source node
  - t: Destination node
  - $l_{(u,v)}$: weight of the edge btw nodes u and v
  - d(u): shortest path distance from s to u.
    - sum of edge weights in path
- We aim to compute d(t)!



Dijkstra’s algorithm- Cnt.

COMPUTER SCIENCE
UNIVERSITY OF MARYLAND

- To find the shortest path from *s* to *t*:
  - Maintain a set of ***explored nodes*** **S** for which we have determined the shortest path distance from s to any u ϵ **S**.
  - Repeatedly expand **S**.

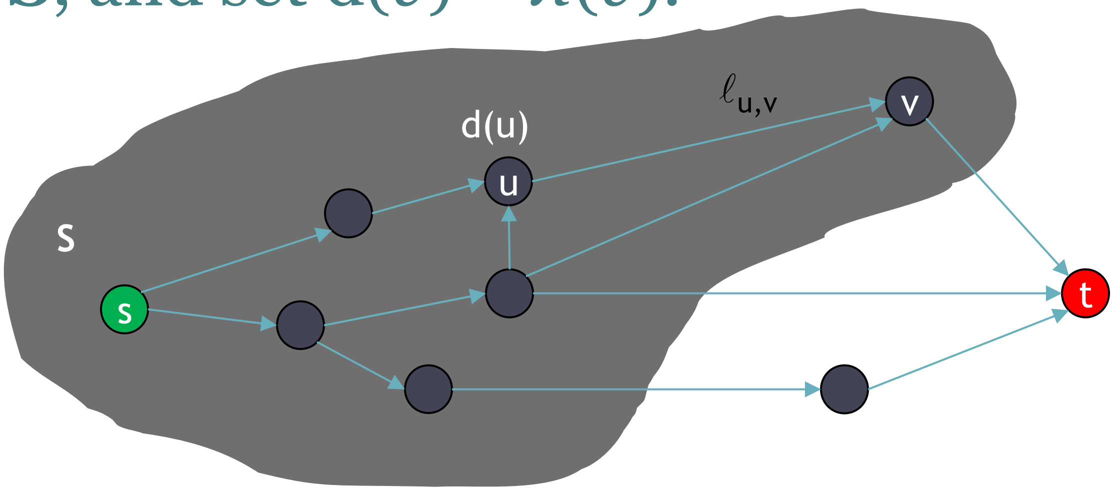


Dijkstra's algorithm- Cnt.

- Repeatedly expand **S**?
  - Repeatedly choose the unexplored node v that minimizes:

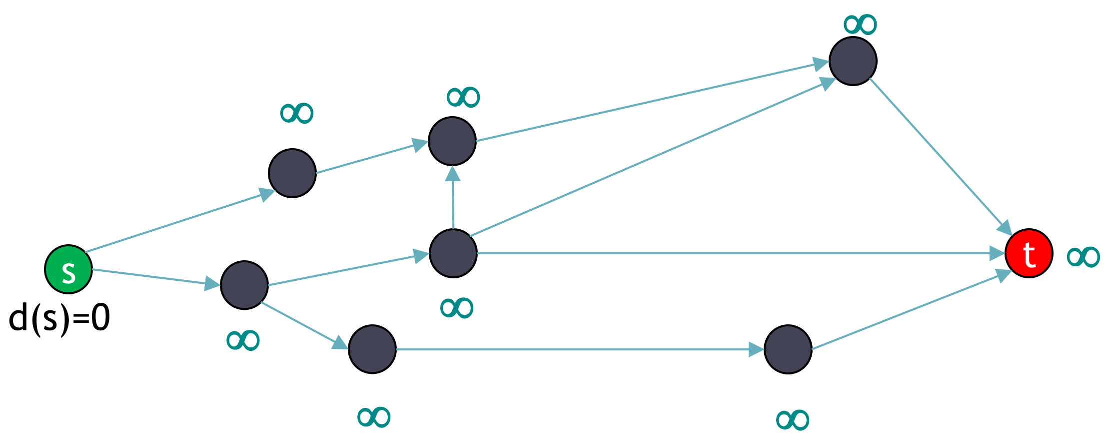
$$\textbf{if } d(v) > d(u) + l_{(u,v)} \textbf{ then } d(v) \leftarrow d(u) + l_{(u,v)}$$

  - add v to **S**, and set $d(v) = \pi(v)$.



Dijkstra's algorithm- Cnt.

- Initialization?
  - $d(s) = 0$
  - $d(u) = \infty$ for all other nodes



Dijkstra's algorithm- Cnt.

- $d(s) \leftarrow 0$
- **for** each $v \in N - \{s\}$
  - **do** $d(v) \leftarrow \infty$
- $S \leftarrow \emptyset$
- $Q \leftarrow N$ $\triangleright$ Q is a set maintaining $N - S$
- **while** $Q \neq \emptyset$
  - **do** $u \leftarrow$ EXTRACT-MIN(Q)
    - $S \leftarrow S \cup \{u\}$
    - **for** each $v \in Adj(u)$
      - **do if** $d(v) > d(u) + l_{(u,v)}$
        - **then** $d(v) \leftarrow d(u) + l_{(u,v)}$

Set of explored nodes

Set of unexplored nodes

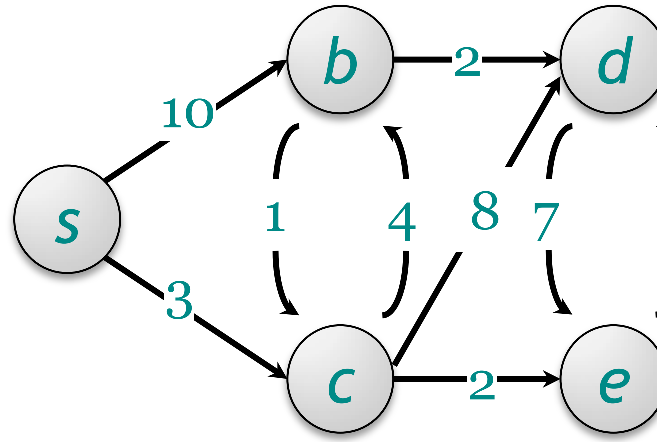
Returns node u ∈ Q that has minimum d(u)

Add it to explored nodes

Update d(.) for all neighbors of u: this is called **relaxation**!

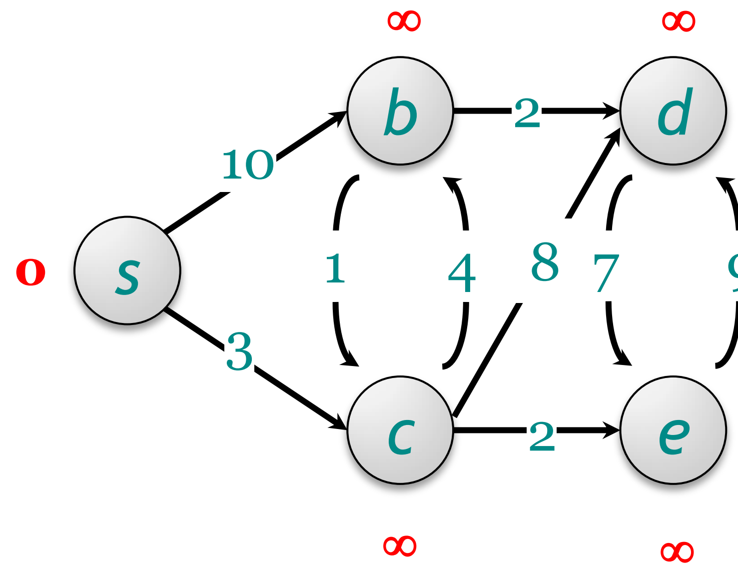
Example 1.

- $d(s) \leftarrow 0$
- **for** each $v \in \boldsymbol{N} - \{s\}$
  - **do** $d(v) \leftarrow \infty$
- $\boldsymbol{S} \leftarrow \varnothing$
- $\boldsymbol{Q} \leftarrow \boldsymbol{N}$
- **while** $\boldsymbol{Q} \neq \varnothing$
  - **do** $u \leftarrow$ EXTRACT-MIN$(\boldsymbol{Q})$
    - $\boldsymbol{S} \leftarrow \boldsymbol{S} \cup \{u\}$
    - **for** each $v \in Adj(u)$
      - **do if** $d(v) > d(u) + l_{(u,\, v)}$
        - **then** $d(v) \leftarrow d(u) + l_{(u,\, v)}$



Example 1.

- $d(s) \leftarrow 0$
- **for** each $v \in N - \{s\}$
  - **do** $d(v) \leftarrow \infty$
- $S \leftarrow \emptyset$
- $Q \leftarrow N$
- **while** $Q \neq \emptyset$
  - **do** $u \leftarrow$ EXTRACT-MIN(Q)
    - $S \leftarrow S \cup \{u\}$
    - **for** each $v \in Adj(u)$
      - **do if** $d(v) > d(u) + l_{(u, v)}$
        - **then** $d(v) \leftarrow d(u) + l_{(u, v)}$

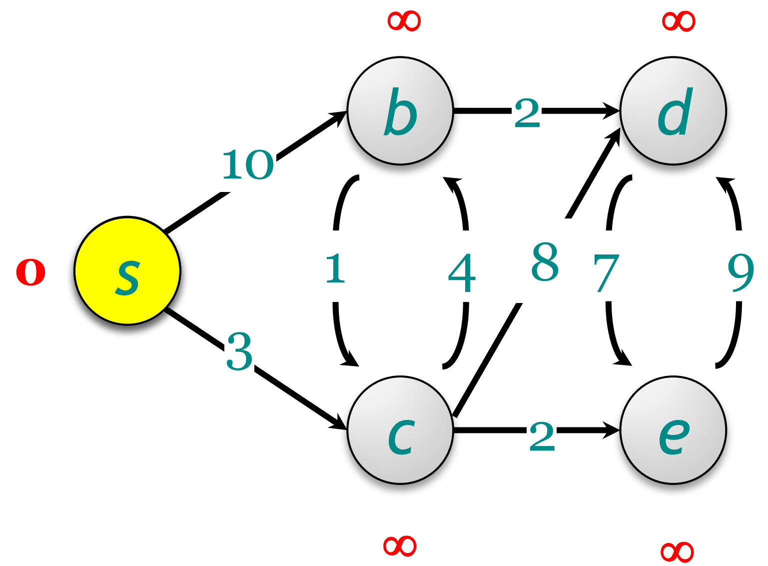


S={}

Q={s, b, c, d, e}

Example 1.

- $d(s) \leftarrow 0$
- **for** each $v \in \boldsymbol{N} - \{s\}$
  - **do** $d(v) \leftarrow \infty$
- $\boldsymbol{S} \leftarrow \varnothing$
- $\boldsymbol{Q} \leftarrow \boldsymbol{N}$
- **while** $\boldsymbol{Q} \neq \varnothing$
  - **do** $u \leftarrow$ EXTRACT-MIN$(\boldsymbol{Q})$
    - $\boldsymbol{S} \leftarrow \boldsymbol{S} \cup \{u\}$
    - **for** each $v \in Adj(u)$
      - **do if** $d(v) > d(u) + l_{(u,\, v)}$
        - **then** $d(v) \leftarrow d(u) + l_{(u,\, v)}$



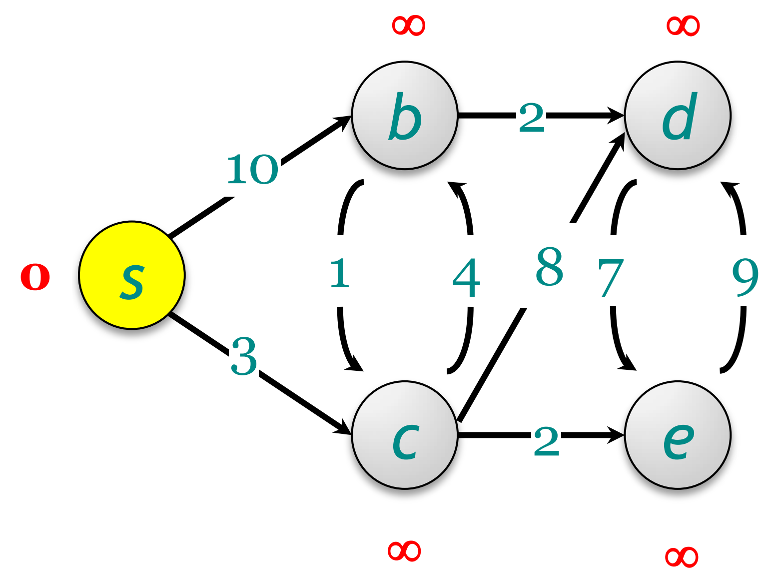
S={}

s

Q={b, c, d, e}

Example 1.

- $d(s) \leftarrow 0$
- **for** each $v \in \boldsymbol{N} - \{s\}$
  - **do** $d(v) \leftarrow \infty$
- $\boldsymbol{S} \leftarrow \emptyset$
- $\boldsymbol{Q} \leftarrow \boldsymbol{N}$
- **while** $\boldsymbol{Q} \neq \emptyset$
  - **do** $u \leftarrow$ EXTRACT-MIN($\boldsymbol{Q}$)
    - $\boldsymbol{S} \leftarrow \boldsymbol{S} \cup \{u\}$
    - **for** each $v \in Adj(u)$
      - **do if** $d(v) > d(u) + l_{(u, v)}$
        - **then** $d(v) \leftarrow d(u) + l_{(u, v)}$

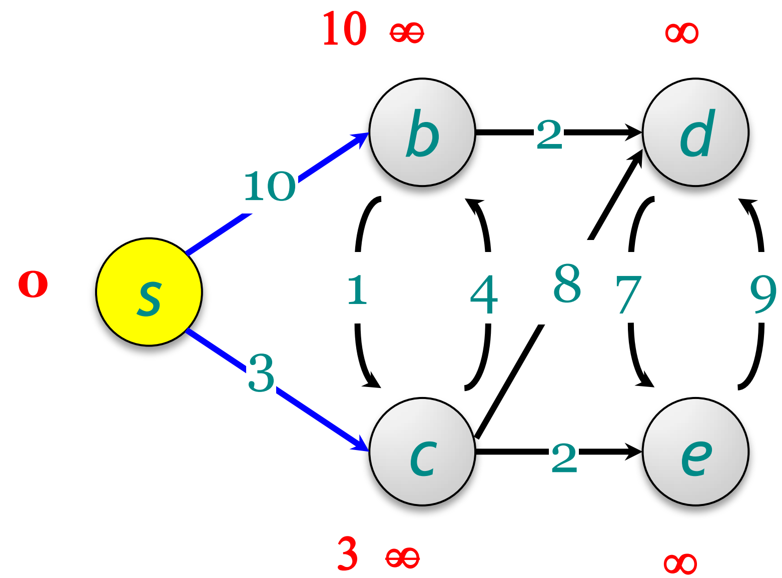


S={s}

Q={b, c, d, e}

Example 1.

- $d(s) \leftarrow 0$
- **for** each $v \in \boldsymbol{N} - \{s\}$
  - **do** $d(v) \leftarrow \infty$
- $\boldsymbol{S} \leftarrow \emptyset$
- $\boldsymbol{Q} \leftarrow \boldsymbol{N}$
- **while** $\boldsymbol{Q} \neq \emptyset$
  - **do** $u \leftarrow$ EXTRACT-MIN$(\boldsymbol{Q})$
    - $\boldsymbol{S} \leftarrow \boldsymbol{S} \cup \{u\}$
    - **for** each $v \in Adj(u)$
      - **do if** $d(v) > d(u) + l_{(u, v)}$
        - **then** $d(v) \leftarrow d(u) + l_{(u, v)}$

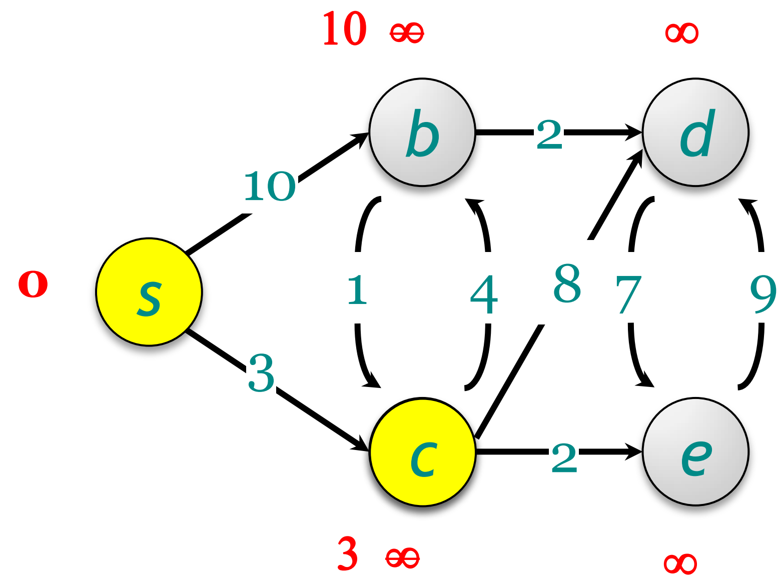


S={s}

Q={b, c, d, e}

Example 1.

- $d(s) \leftarrow 0$
- **for** each $v \in \boldsymbol{N} - \{s\}$
  - **do** $d(v) \leftarrow \infty$
- $\boldsymbol{S} \leftarrow \varnothing$
- $\boldsymbol{Q} \leftarrow \boldsymbol{N}$
- **while** $\boldsymbol{Q} \neq \varnothing$
  - **do** $u \leftarrow$ EXTRACT-MIN($\boldsymbol{Q}$)
    - $\boldsymbol{S} \leftarrow \boldsymbol{S} \cup \{u\}$
    - **for** each $v \in Adj(u)$
      - **do if** $d(v) > d(u) + l_{(u,v)}$
        - **then** $d(v) \leftarrow d(u) + l_{(u,v)}$



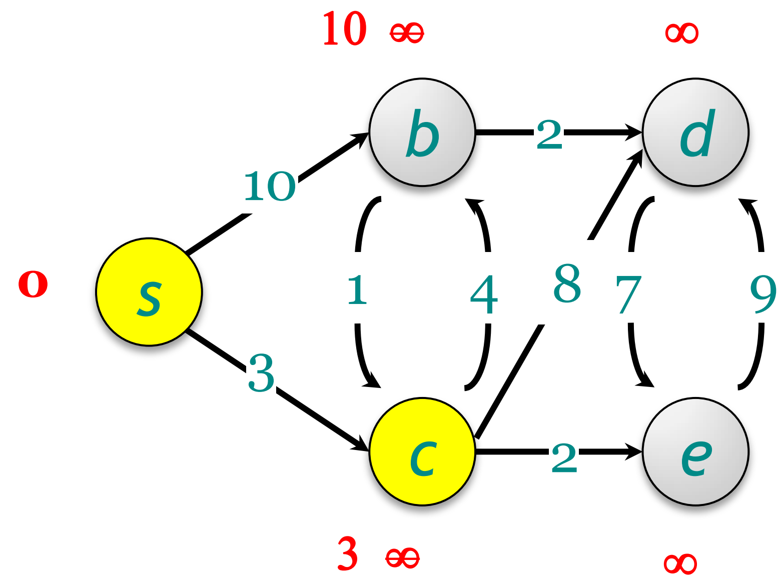
S={s}

c

Q={b, d, e}

Example 1.

- $d(s) \leftarrow 0$
- **for** each $v \in \boldsymbol{N} - \{s\}$
  - **do** $d(v) \leftarrow \infty$
- $\boldsymbol{S} \leftarrow \varnothing$
- $\boldsymbol{Q} \leftarrow \boldsymbol{N}$
- **while** $\boldsymbol{Q} \neq \varnothing$
  - **do** $u \leftarrow$ EXTRACT-MIN($\boldsymbol{Q}$)
    - $\boldsymbol{S} \leftarrow \boldsymbol{S} \cup \{u\}$
    - **for** each $v \in Adj(u)$
      - **do if** $d(v) > d(u) + l_{(u,\, v)}$
        - **then** $d(v) \leftarrow d(u) + l_{(u,\, v)}$

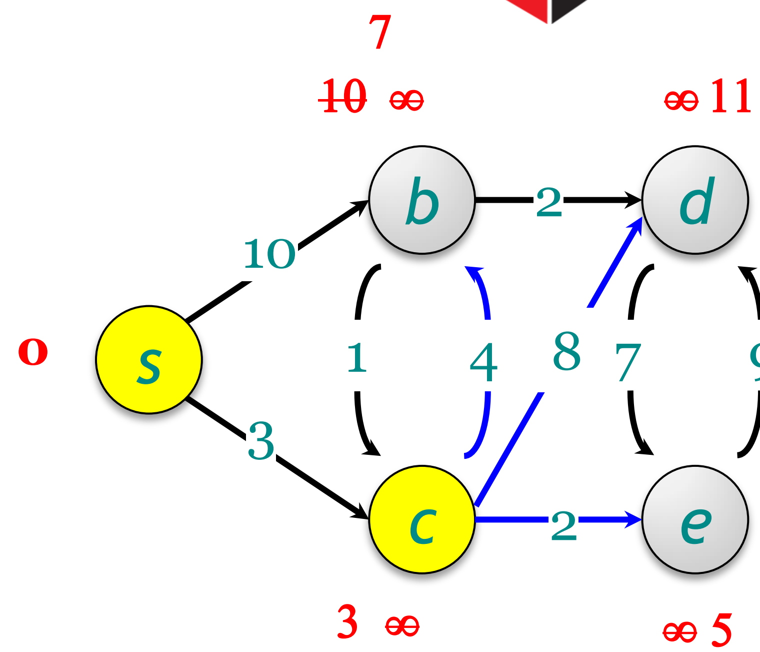


S={s, c}

Q={b, d, e}

Example 1.

- $d(s) \leftarrow 0$
- **for** each $v \in N - \{s\}$
  - **do** $d(v) \leftarrow \infty$
- $S \leftarrow \varnothing$
- $Q \leftarrow N$
- **while** $Q \neq \varnothing$
  - **do** $u \leftarrow$ EXTRACT-MIN(Q)
    - $S \leftarrow S \cup \{u\}$
    - **for** each $v \in Adj(u)$
      - **do if** $d(v) > d(u) + l_{(u,v)}$
        - **then** $d(v) \leftarrow d(u) + l_{(u,v)}$

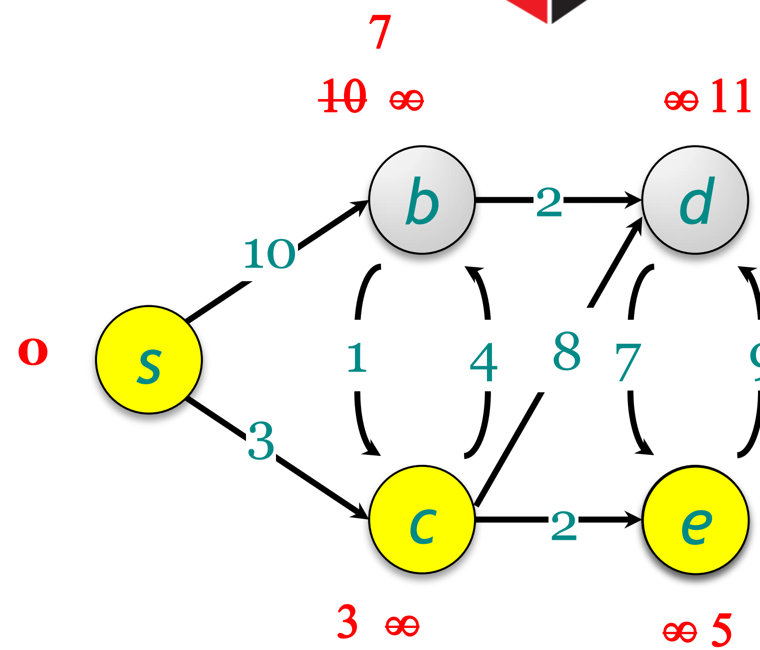


S={s, c}

Q={b, d, e}

Example 1.

- $d(s) \leftarrow 0$
- **for** each $v \in N - \{s\}$
  - **do** $d(v) \leftarrow \infty$
- $S \leftarrow \varnothing$
- $Q \leftarrow N$
- **while** $Q \neq \varnothing$
  - **do** $u \leftarrow$ EXTRACT-MIN(Q)
    - $S \leftarrow S \cup \{u\}$
    - **for** each $v \in Adj(u)$
      - **do if** $d(v) > d(u) + l_{(u,v)}$
        - **then** $d(v) \leftarrow d(u) + l_{(u,v)}$



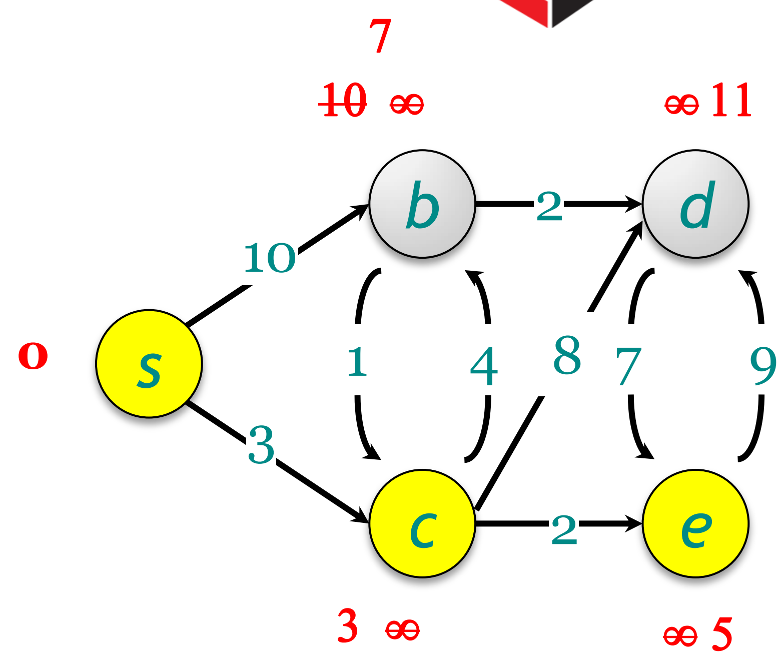
S={s, c}

e

Q={b, d}

Example 1.

- $d(s) \leftarrow 0$
- **for** each $v \in \boldsymbol{N} - \{s\}$
  - **do** $d(v) \leftarrow \infty$
- $\boldsymbol{S} \leftarrow \varnothing$
- $\boldsymbol{Q} \leftarrow \boldsymbol{N}$
- **while** $\boldsymbol{Q} \neq \varnothing$
  - **do** $u \leftarrow$ EXTRACT-MIN$(\boldsymbol{Q})$
    - $\boldsymbol{S} \leftarrow \boldsymbol{S} \cup \{u\}$
    - **for** each $v \in Adj(u)$
      - **do if** $d(v) > d(u) + l_{(u,v)}$
        - **then** $d(v) \leftarrow d(u) + l_{(u,v)}$

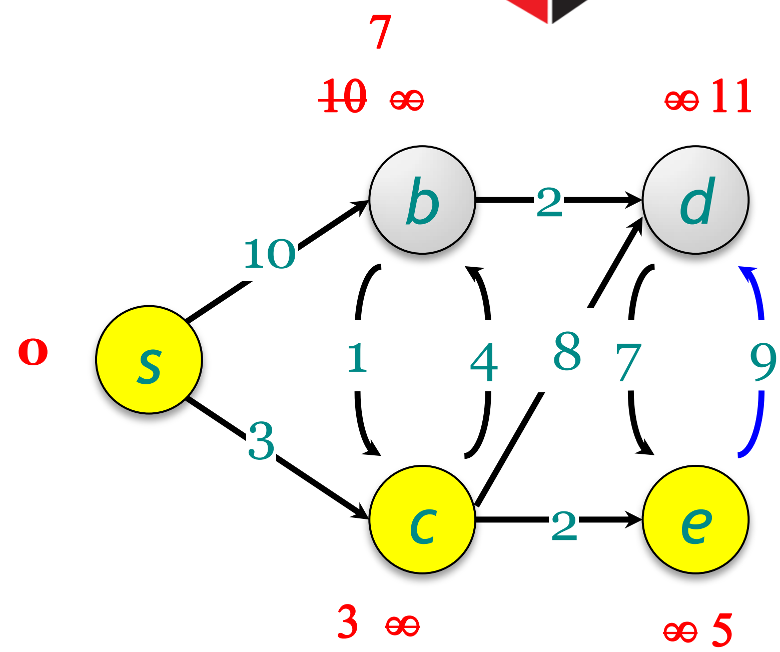


S={s, c, e}

Q={b, d}

Example 1.

- $d(s) \leftarrow 0$
- **for** each $v \in \boldsymbol{N} - \{s\}$
  - **do** $d(v) \leftarrow \infty$
- $\boldsymbol{S} \leftarrow \varnothing$
- $\boldsymbol{Q} \leftarrow \boldsymbol{N}$
- **while** $\boldsymbol{Q} \neq \varnothing$
  - **do** $u \leftarrow$ EXTRACT-MIN($\boldsymbol{Q}$)
    - $\boldsymbol{S} \leftarrow \boldsymbol{S} \cup \{u\}$
    - **for** each $v \in Adj(u)$
      - **do if** $d(v) > d(u) + l_{(u,v)}$
        - **then** $d(v) \leftarrow d(u) + l_{(u,v)}$

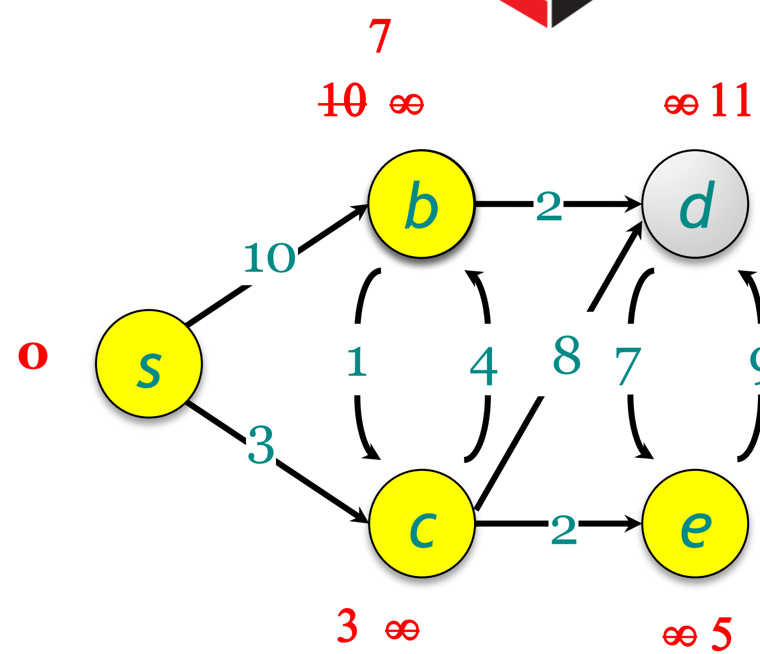


S={s, c, e}

Q={b, d}

Example 1.

- $d(s) \leftarrow 0$
- **for** each $v \in \boldsymbol{N} - \{s\}$
  - **do** $d(v) \leftarrow \infty$
- $\boldsymbol{S} \leftarrow \varnothing$
- $\boldsymbol{Q} \leftarrow \boldsymbol{N}$
- **while** $\boldsymbol{Q} \neq \varnothing$
  - **do** $u \leftarrow$ EXTRACT-MIN($\boldsymbol{Q}$)
    - $\boldsymbol{S} \leftarrow \boldsymbol{S} \cup \{u\}$
    - **for** each $v \in Adj(u)$
      - **do if** $d(v) > d(u) + l_{(u,v)}$
        - **then** $d(v) \leftarrow d(u) + l_{(u,v)}$



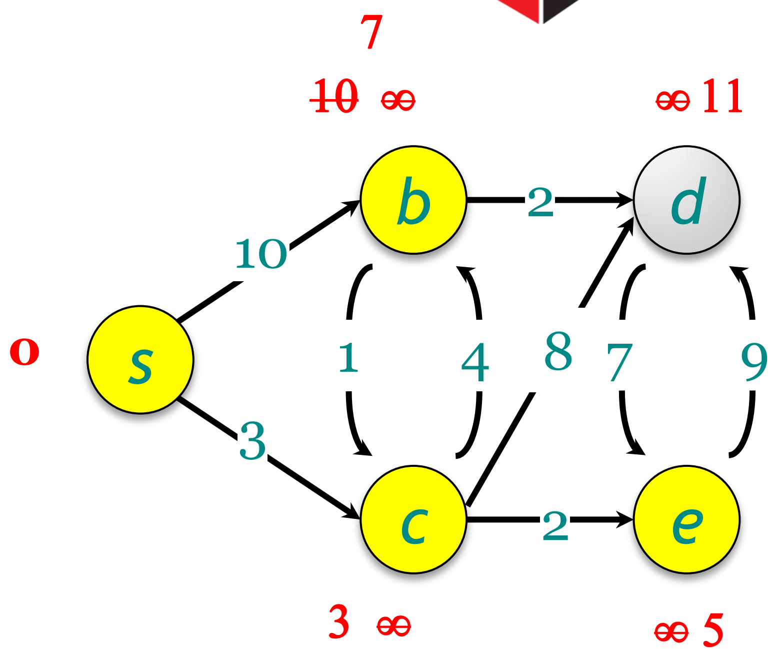
S={s, c, e}

b

Q={d}

Example 1.

- $d(s) \leftarrow 0$
- **for** each $v \in \boldsymbol{N} - \{s\}$
  - **do** $d(v) \leftarrow \infty$
- $\boldsymbol{S} \leftarrow \emptyset$
- $\boldsymbol{Q} \leftarrow \boldsymbol{N}$
- **while** $\boldsymbol{Q} \neq \emptyset$
  - **do** $u \leftarrow$ EXTRACT-MIN($\boldsymbol{Q}$)
    - $\boldsymbol{S} \leftarrow \boldsymbol{S} \cup \{u\}$
    - **for** each $v \in Adj(u)$
      - **do if** $d(v) > d(u) + l_{(u,\, v)}$
        - **then** $d(v) \leftarrow d(u) + l_{(u,\, v)}$

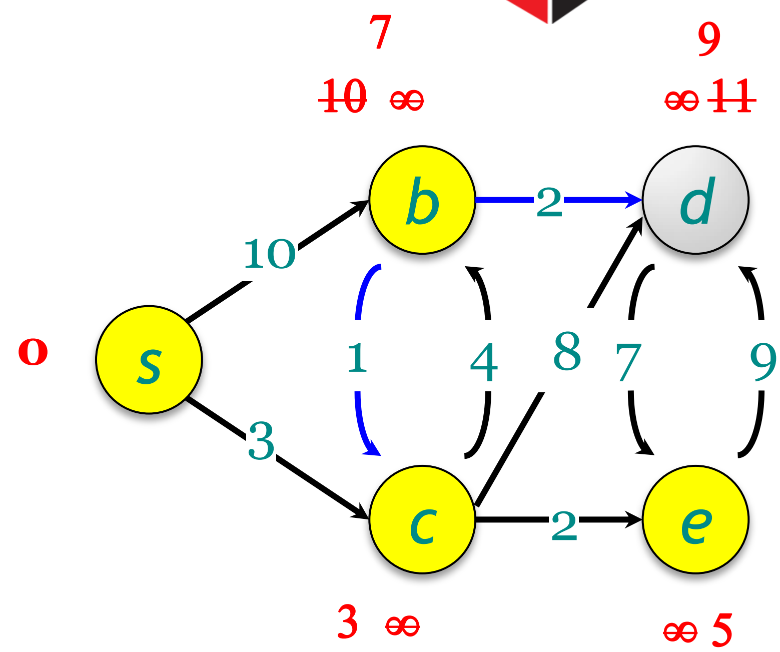


S={s, c, e, b}

Q={d}

Example 1.

- $d(s) \leftarrow 0$
- **for** each $v \in \boldsymbol{N} - \{s\}$
  - **do** $d(v) \leftarrow \infty$
- $\boldsymbol{S} \leftarrow \varnothing$
- $\boldsymbol{Q} \leftarrow \boldsymbol{N}$
- **while** $\boldsymbol{Q} \neq \varnothing$
  - **do** $u \leftarrow$ EXTRACT-MIN$(\boldsymbol{Q})$
    - $\boldsymbol{S} \leftarrow \boldsymbol{S} \cup \{u\}$
    - **for** each $v \in Adj(u)$
      - **do if** $d(v) > d(u) + l_{(u, v)}$
        - **then** $d(v) \leftarrow d(u) + l_{(u, v)}$

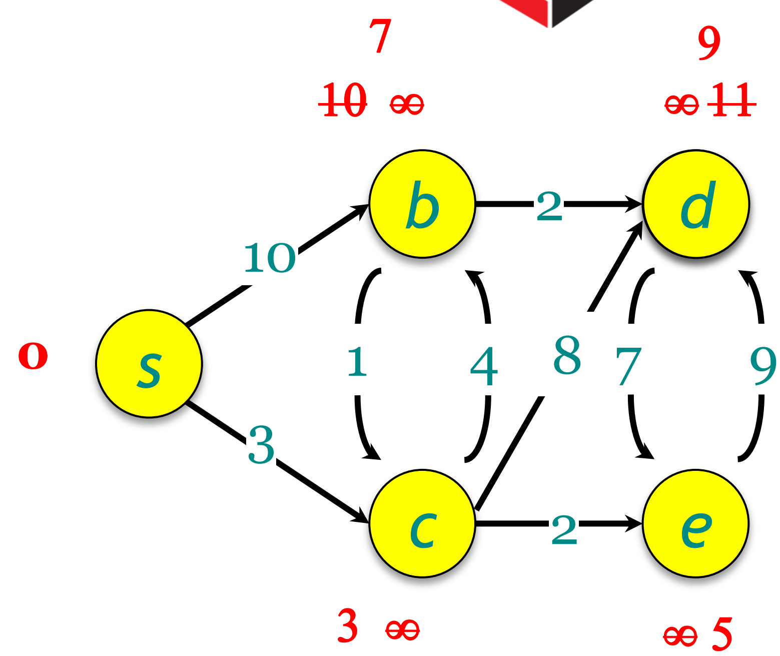


S={s, c, e, b}

Q={d}

Example 1.

- $d(s) \leftarrow 0$
- **for** each $v \in \boldsymbol{N} - \{s\}$
  - **do** $d(v) \leftarrow \infty$
- $\boldsymbol{S} \leftarrow \varnothing$
- $\boldsymbol{Q} \leftarrow \boldsymbol{N}$
- **while** $\boldsymbol{Q} \neq \varnothing$
  - **do** $u \leftarrow$ EXTRACT-MIN($\boldsymbol{Q}$)
    - $\boldsymbol{S} \leftarrow \boldsymbol{S} \cup \{u\}$
    - **for** each $v \in Adj(u)$
      - **do if** $d(v) > d(u) + l_{(u, v)}$
        - **then** $d(v) \leftarrow d(u) + l_{(u, v)}$



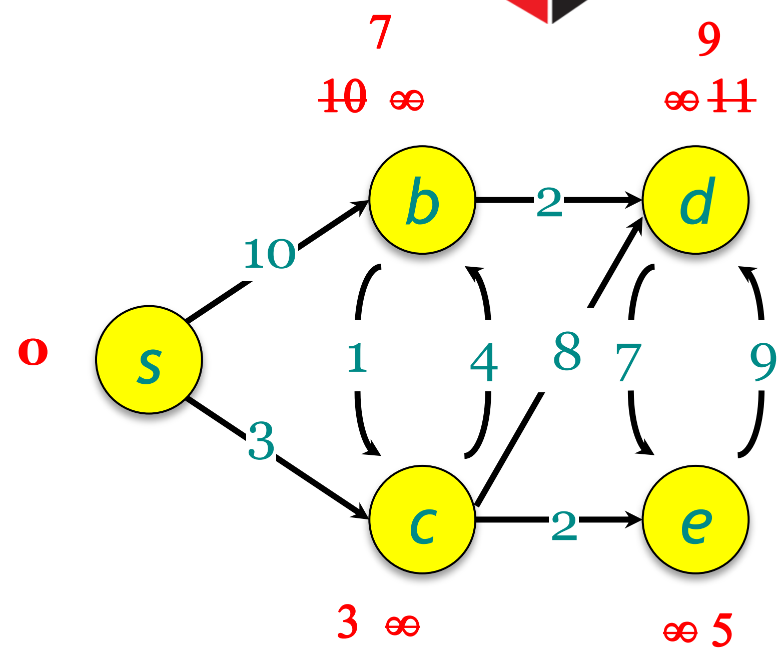
S={s, c, e, b}

d

Q={}

Example 1.

- $d(s) \leftarrow 0$
- **for** each $v \in \boldsymbol{N} - \{s\}$
    - **do** $d(v) \leftarrow \infty$
- $\boldsymbol{S} \leftarrow \varnothing$
- $\boldsymbol{Q} \leftarrow \boldsymbol{N}$
- **while** $\boldsymbol{Q} \neq \varnothing$
    - **do** $u \leftarrow$ EXTRACT-MIN($\boldsymbol{Q}$)
        - $\boldsymbol{S} \leftarrow \boldsymbol{S} \cup \{u\}$
        - **for** each $v \in Adj(u)$
            - **do if** $d(v) > d(u) + l_{(u,v)}$
                - **then** $d(v) \leftarrow d(u) + l_{(u,v)}$

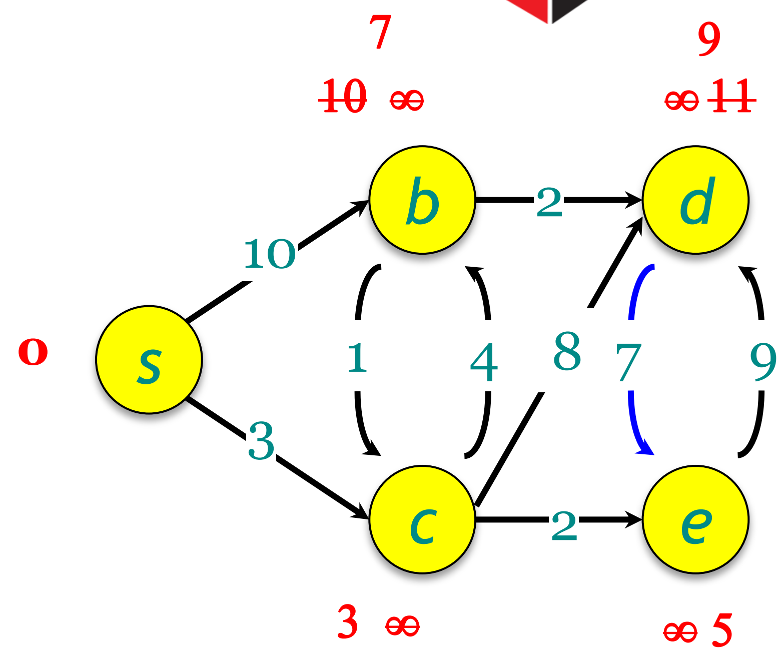


S={s, c, e, b, d}

Q={}

Example 1.

- $d(s) \leftarrow 0$
- **for** each $v \in \boldsymbol{N} - \{s\}$
  - **do** $d(v) \leftarrow \infty$
- $\boldsymbol{S} \leftarrow \emptyset$
- $\boldsymbol{Q} \leftarrow \boldsymbol{N}$
- **while** $\boldsymbol{Q} \neq \emptyset$
  - **do** $u \leftarrow$ EXTRACT-MIN($\boldsymbol{Q}$)
    - $\boldsymbol{S} \leftarrow \boldsymbol{S} \cup \{u\}$
    - **for** each $v \in Adj(u)$
      - **do if** $d(v) > d(u) + l_{(u, v)}$
        - **then** $d(v) \leftarrow d(u) + l_{(u, v)}$

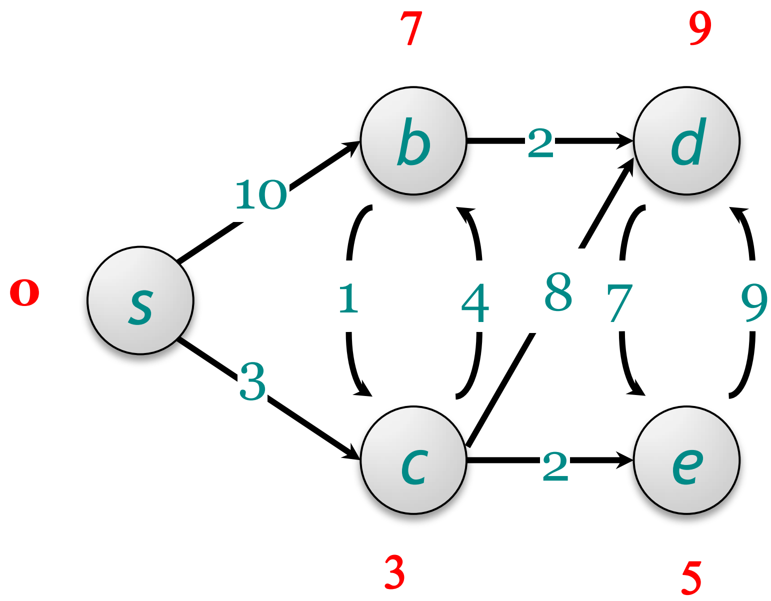


S={s, c, e, b, d}

Q={}

Example 1.

- $d(s) \leftarrow 0$
- **for** each $v \in N - \{s\}$
  - **do** $d(v) \leftarrow \infty$
- $S \leftarrow \varnothing$
- $Q \leftarrow N$
- **while** $Q \neq \varnothing$
  - **do** $u \leftarrow$ EXTRACT-MIN(Q)
    - $S \leftarrow S \cup \{u\}$
    - **for** each $v \in Adj(u)$
      - **do if** $d(v) > d(u) + l_{(u,v)}$
        - **then** $d(v) \leftarrow d(u) + l_{(u,v)}$



S={s, c, e, b, d}

Q={}

Dijkstra's algorithm- Cnt.

- Dijkstra's algorithm computes the shortest distances btw *s* and all other nodes in the graph (not only *t*)!

- Assumptions:
  - the graph is connected, and
  - the weights are nonnegative

Dijkstra's algorithm- Analysis

- $d(s) \leftarrow 0$
- **for** each $v \in \boldsymbol{N} - \{s\}$
  - **do** $d(v) \leftarrow \infty$
- $\boldsymbol{S} \leftarrow \varnothing$
- $\boldsymbol{Q} \leftarrow \boldsymbol{N}$
- **while** $\boldsymbol{Q} \neq \varnothing$
  - **do** $u \leftarrow$ EXTRACT-MIN($\boldsymbol{Q}$)
    - $\boldsymbol{S} \leftarrow \boldsymbol{S} \cup \{u\}$
    - **for** each $v \in Adj(u)$
      - **do if** $d(v) > d(u) + l_{(u,v)}$
        - **then** $d(v) \leftarrow d(u) + l_{(u,v)}$

degree (u) times

$|N|$ times

Time = $\Theta\,(N \cdot T_{\text{EXTRACT-MIN}} + E \cdot T_{\text{Relaxation}})$, Handshaking Lemma!

Dijkstra's algorithm- Analysis- Cnt.

COMPUTER SCIENCE
UNIVERSITY OF MARYLAND

Time = $\Theta\,(N \cdot T_{\text{EXTRACT-MIN}} + E \cdot T_{\text{Relaxation}})$

Q	$T_{\text{EXTRACT-MIN}}$	$T_{\text{DECREASE-KEY}}$	Total
Array	$O(N)$	$O(1)$	$O(N^2)$
Binary Heap	$O(\lg N)$	$O(\lg N)$	$O(E \lg N)$

Questions?

Reading

- Ch.02 Graphs [NCM]
- Ch.22 Elementary Graph Algorithms [CLRS]
- Ch.24 Single-Source Shortest Paths [CLRS].