طراحی الگوریتم

۲۶ آبان ۹۸

ملکی مجد

Topic	Reference
Recursion and Backtracking	Ch.1 and Ch.2 JeffE
Dynamic Programming	Ch.3 JeffE and Ch.15 CLRS
Greedy Algorithms	Ch.4 JeffE and Ch.16 CLRS
Amortized Analysis	Ch.17 CLRS
Elementary Graph algorithms	Ch.6 JeffE and Ch.22 CLRS
Minimum Spanning Trees	Ch.7 JeffE and Ch.23 CLRS
Single-Source Shortest Paths	Ch.8 JeffE and Ch.24 CLRS
All-Pairs Shortest Paths	Ch.9 JeffE and Ch.25 CLRS
Maximum Flow	Ch.10 JeffE and Ch.26 CLRS
String Matching	Ch.32 CLRS
NP-Completeness	Ch.12 JeffE and Ch.34 CLRS

Dijkstra’s algorithm

Aim of Dijkstra's algorithm

- Dijkstra's algorithm solves the **single-source shortest-paths** problem on a **weighted**, **directed** graph $G = (V, E)$ for the case in which all edge weights are **nonnegative**.

- The running time of Dijkstra's algorithm is **lower** than that of the Bellman-Ford algorithm

Dijkstra's algorithm - idea

- Dijkstra's algorithm maintains a **set S** of vertices whose final shortest-path weights from the source s have **already been determined**.
- The algorithm repeatedly selects the vertex $u \in V - S$ with the minimum shortest-path estimate,
  - adds u to S, and
  - relaxes all edges leaving u.

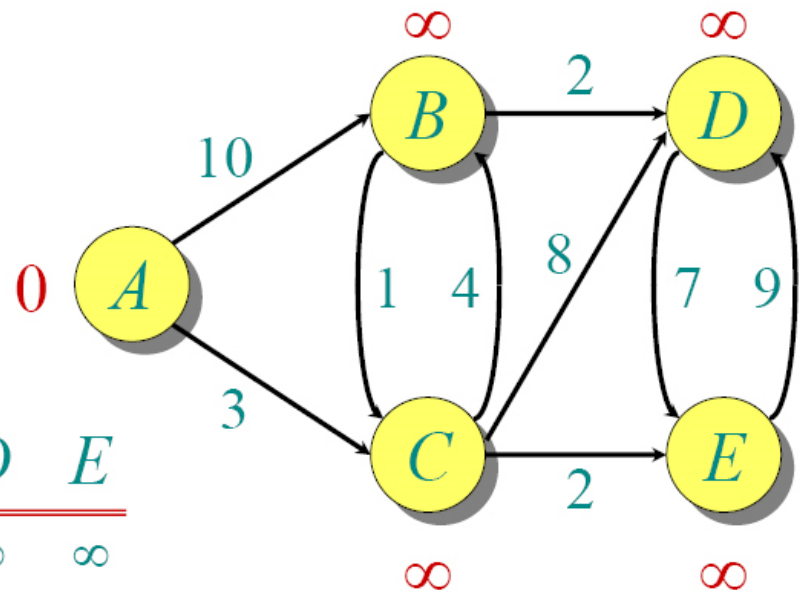
- min-priority queue Q of vertices can be used!

Pseudocode

DIJKSTRA(G, w, s)
1 INITIALIZE-SINGLE-SOURCE(G, s)
2 $S \leftarrow \emptyset$
3 $Q \leftarrow V[G]$
4 **while** $Q = \emptyset$
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)

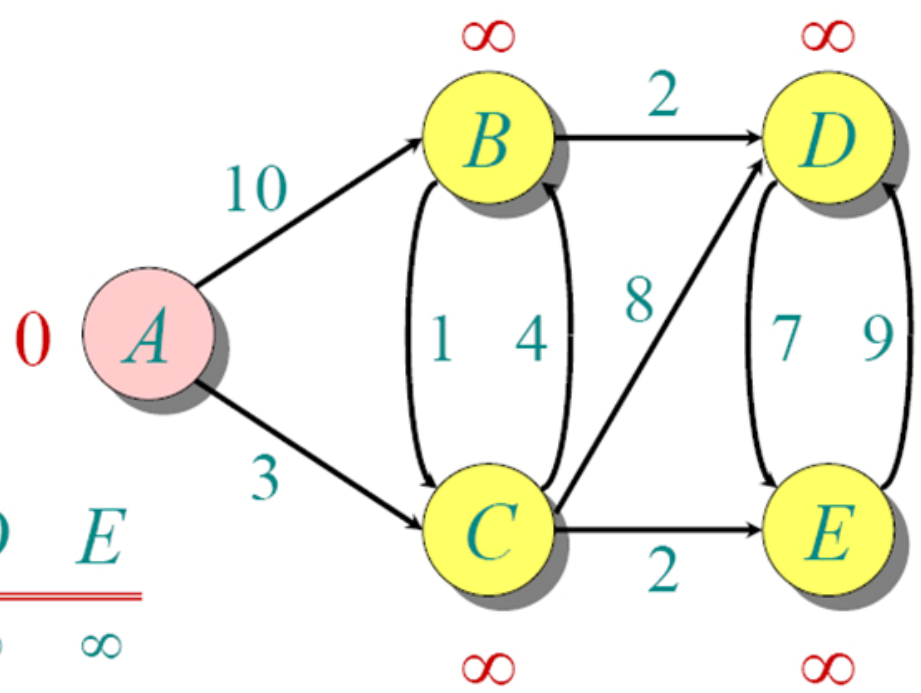
Dijkstra Example

Initialize:

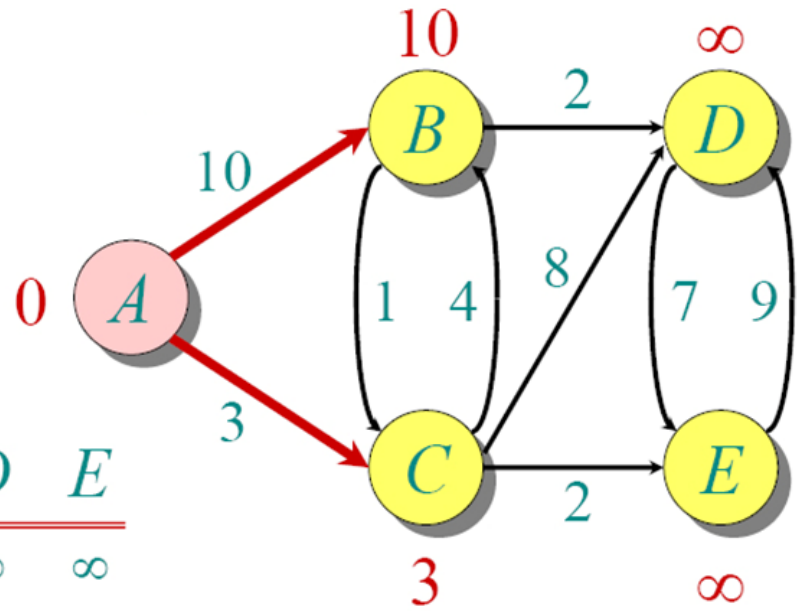


Q:	A	B	C	D	E
	0	∞	∞	∞	∞

S: {}

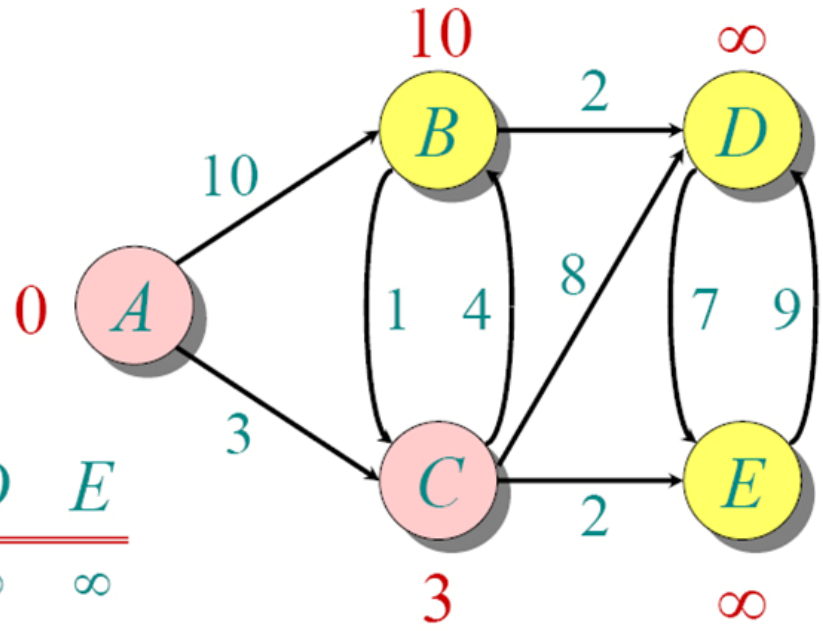


Q:	A	B	C	D	E
	0	∞	∞	∞	∞



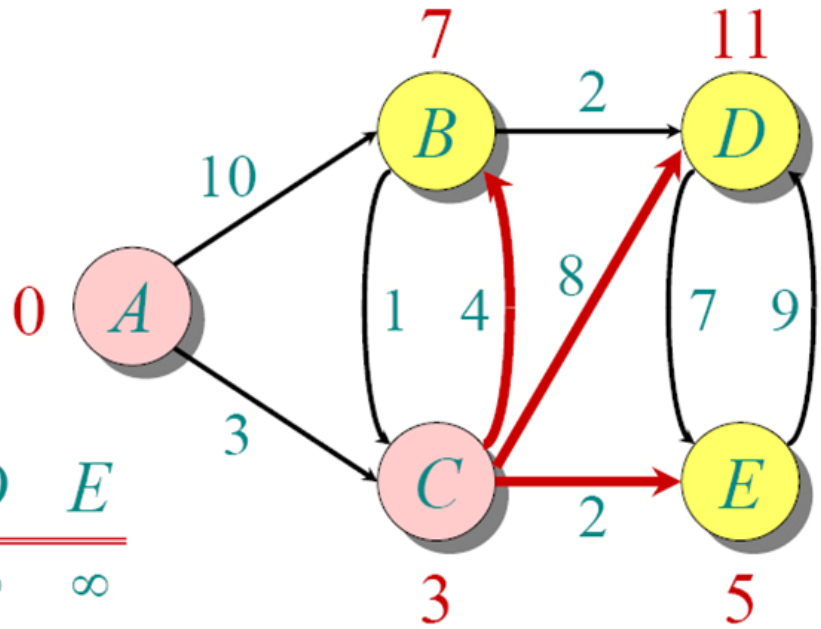
Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞

S: { A }



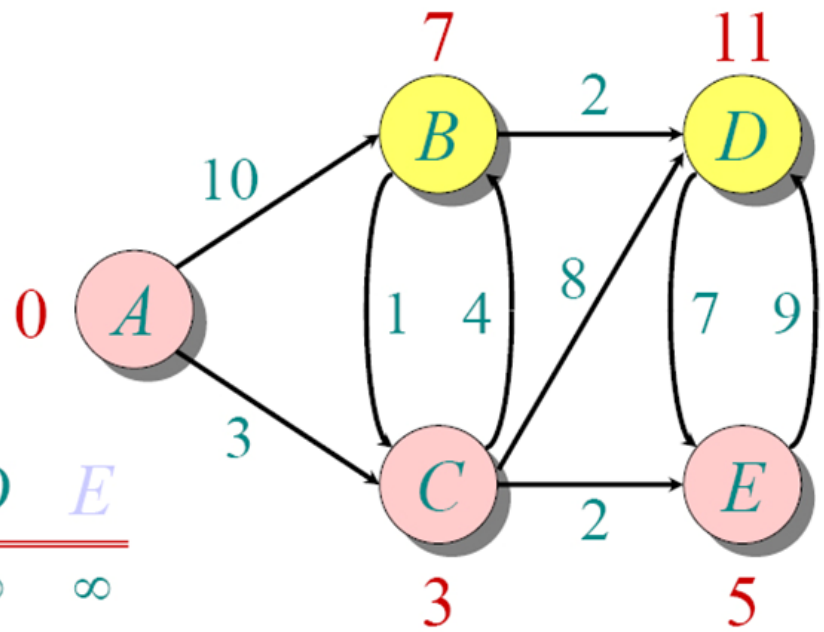
Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞

S: { A, C }



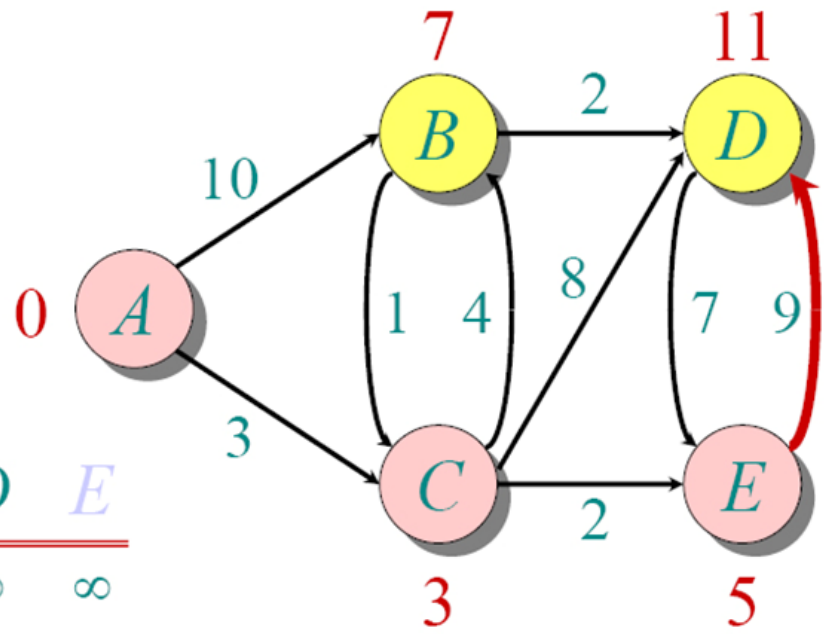
Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5

S: { A, C }



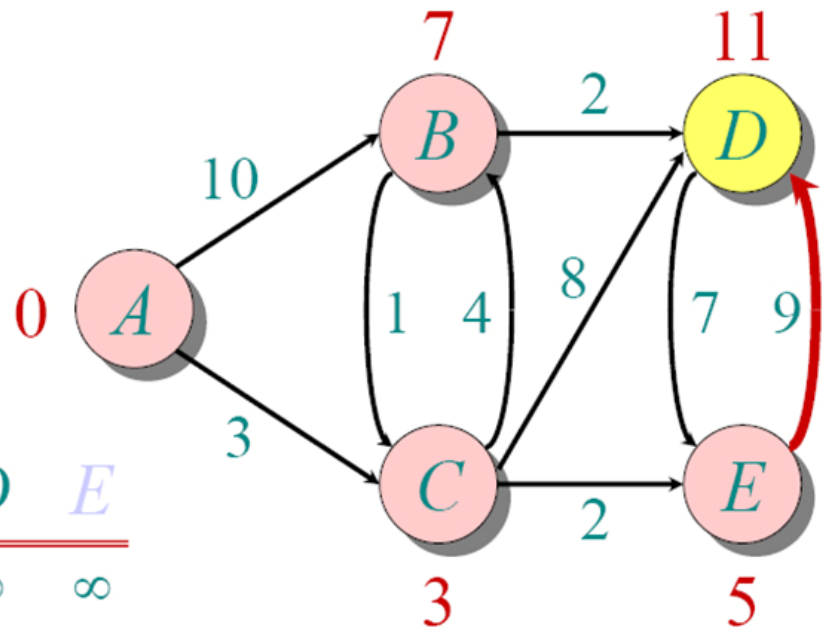
Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5

S: { A, C, E }



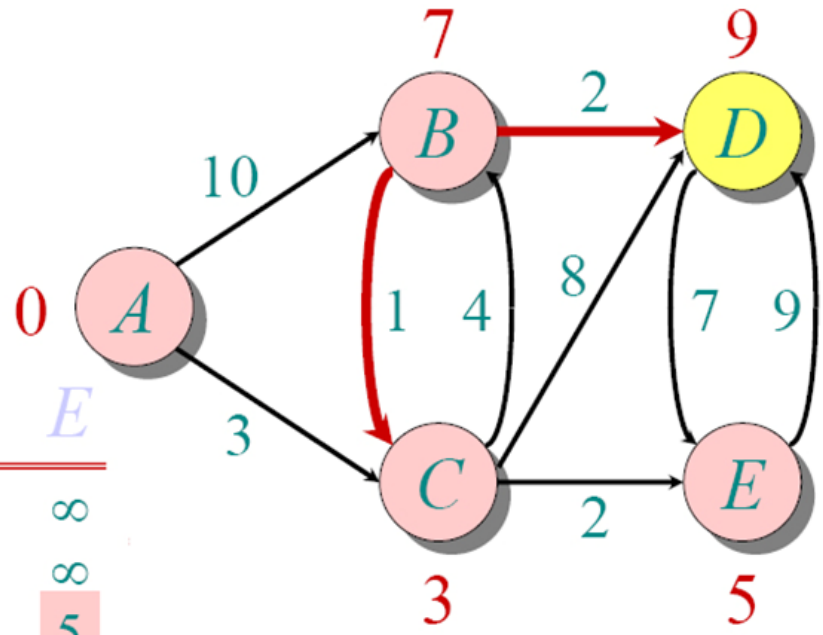
Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5
		7		11	

S: { A, C, E }



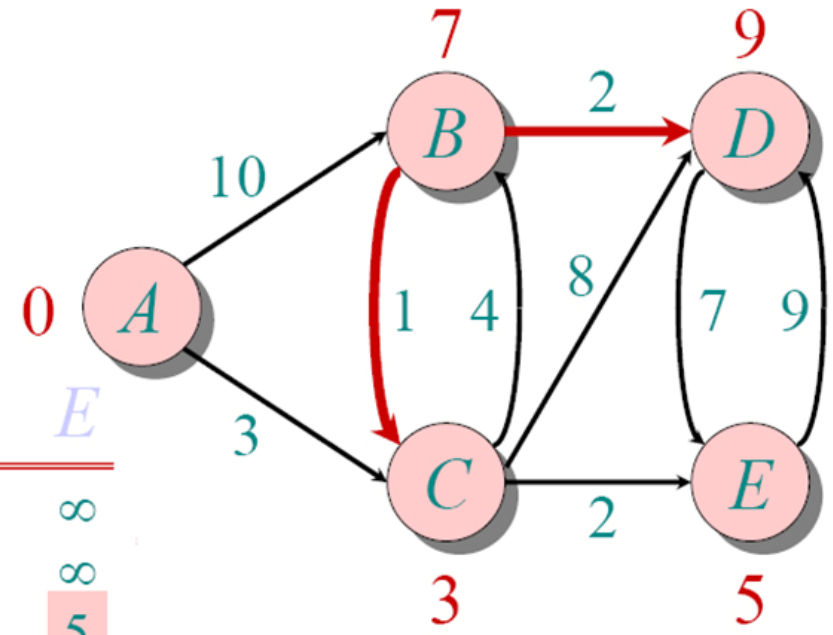
Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5
		7		11	

S: { A, C, E, B }



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

S: { A, C, E, B }

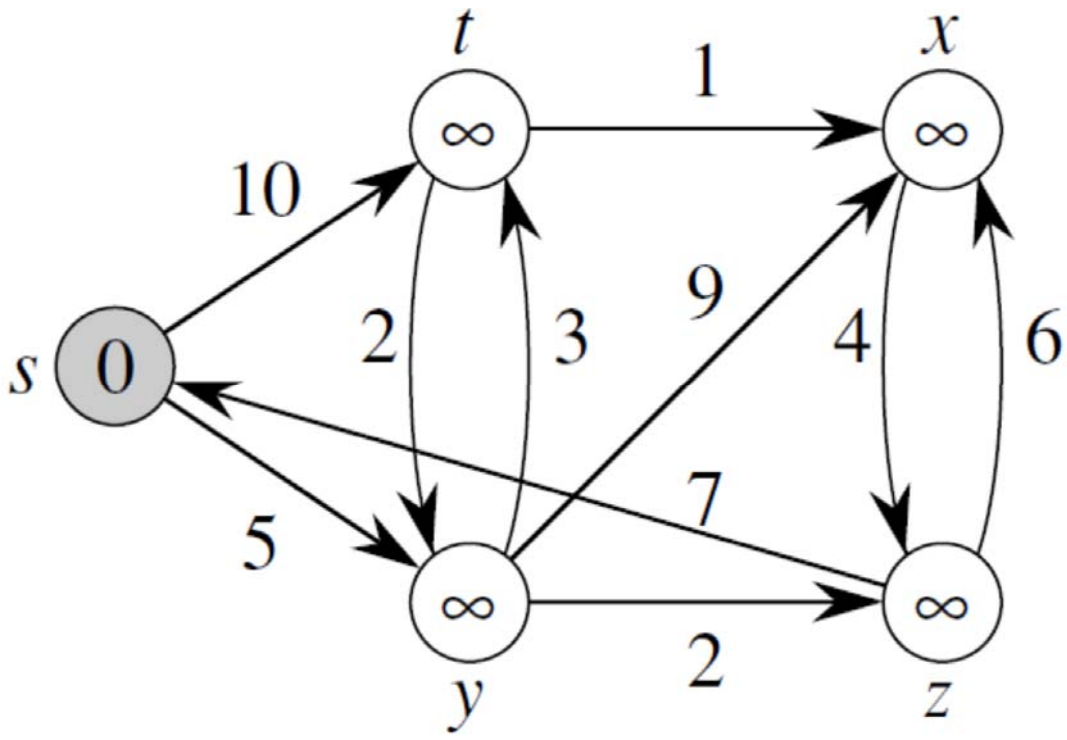


Q:

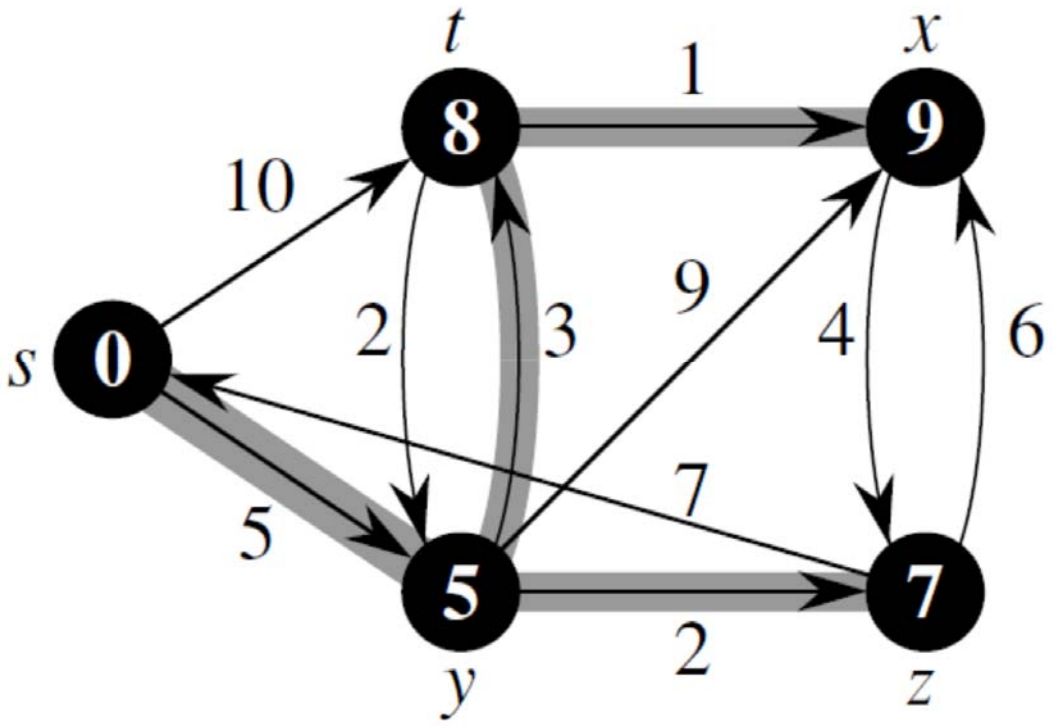
A	B	C	D	E
0	∞	∞	∞	∞
	10	3	∞	∞
	7		11	5
	7		11	
			9	

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

Do Dijkstra ?



result



Algorithm properties

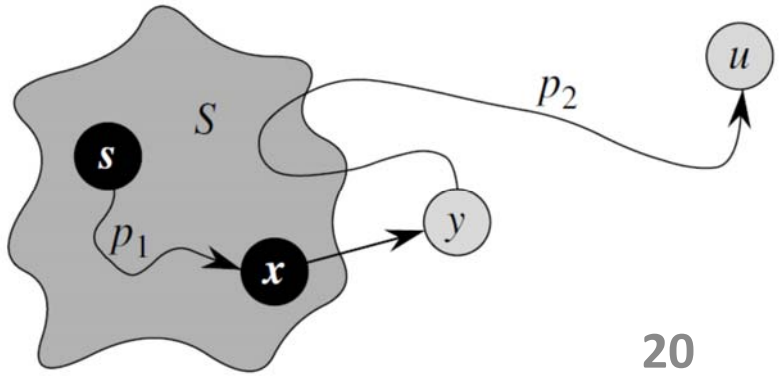
DIJKSTRA(G, w, s)
1 INITIALIZE-SINGLE-SOURCE(G, s)
2 $S \leftarrow \emptyset$
3 $Q \leftarrow V[G]$
4 **while** $Q = \emptyset$
5 **do** $u \leftarrow EXTRACT - MIN(Q)$
6 $S \leftarrow S \cup \{u\}$
7 **for** each vertex $v \in Adj[u]$
8 **do** RELAX(u, v, w)

- The algorithm maintains the invariant that $Q = V - S$ at the start of each iteration of the *while* loop of lines 4–8.
- The *while* loop of lines 4–8 iterates exactly $|V|$ times.
- it uses a greedy strategy.
  - always chooses the "closest" vertex in $V - S$ to add to set S

Theorem 24.6 (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 $d[u] = \delta(s, u)$ for all vertices $u \in V$.

- ***Proof*** the following loop invariant:
  - At the start of each iteration of the **while** loop of lines 4–8, $d[v] = \delta(s, v)$ for each vertex $v \in S$.



Analysis

- Dijkstra maintains the min-priority queue Q by calling three priority-queue operations:
  - INSERT -> is invoked once per vertex
  - EXTRACT-MIN -> is invoked once per vertex
  - DECREASE-KEY -> at most $|E|$ by using aggregate analysis

Analysis (1)

- Dijkstra maintains the min-priority queue Q by calling three priority-queue operations:
  - INSERT -> is invoked once per vertex $\boldsymbol{O(1)}$
  - EXTRACT-MIN -> is invoked once per vertex $\boldsymbol{O(V)}$
  - DECREASE-KEY -> at most $|E|$ by using aggregate analysis $\boldsymbol{O(1)}$

- Implementing the min-priority queue: **ordinary array**
  - Number vertices from 1 to v
  - Store $d[v]$ in the vth entry of array
  - total time of $O(V^2 + E) = O(V^2)$.

Analysis (2)

- Dijkstra maintains the min-priority queue Q by calling three priority-queue operations:
  - INSERT -> is invoked once per vertex $\boldsymbol{O(V)(overall)}$
  - EXTRACT-MIN -> is invoked once per vertex $\boldsymbol{O(lg\ V)}$
  - DECREASE-KEY -> at most $|E|$ by using aggregate analysis $\boldsymbol{O(lg\ V)}$

- Implementing the min-priority queue: **min-Heap**
  - total time of $O((V + E) \lg V) = O(E \lg V)$.

- *This running time is an improvement over the straightforward $O(V^2)$ time implementation if $E = o(V^2/lg\ V)$.*

Analysis (3)

- Dijkstra maintains the min-priority queue Q by calling three priority-queue operations:
  - INSERT -> is invoked once per vertex $\boldsymbol{O(V)(overall)}$
  - EXTRACT-MIN -> is invoked once per vertex amortized $\boldsymbol{O(lg\ V)}$
  - DECREASE-KEY -> at most $|E|$ by using aggregate analysis amortized $\boldsymbol{O(1)}$

- Implementing the min-priority queue: **Fibonacci heap**
  - total time of $O(V \lg V + E)$