Dijkstra's Algorithm

Dr. Sasmita Kumari Nayak
Computer Science & Engineering

Dijkstra's Algorithm

- It is a single-source shortest path problem or algorithm in the graph theory.
- It is used for finding shortest paths from a source vertex *u* to all other vertices in the graph.
- This works on both directed and undirected graphs.
- All the edges must have non-negative weights.

Procedure

Approach: Greedy

Input: Weighted graph G={E,V} and source vertex $v \in V$, such that all edge weights are nonnegative

Output: Lengths of shortest paths (or the shortest paths themselves) from a given source vertex $v \in V$ to all other vertices

Pseudocode

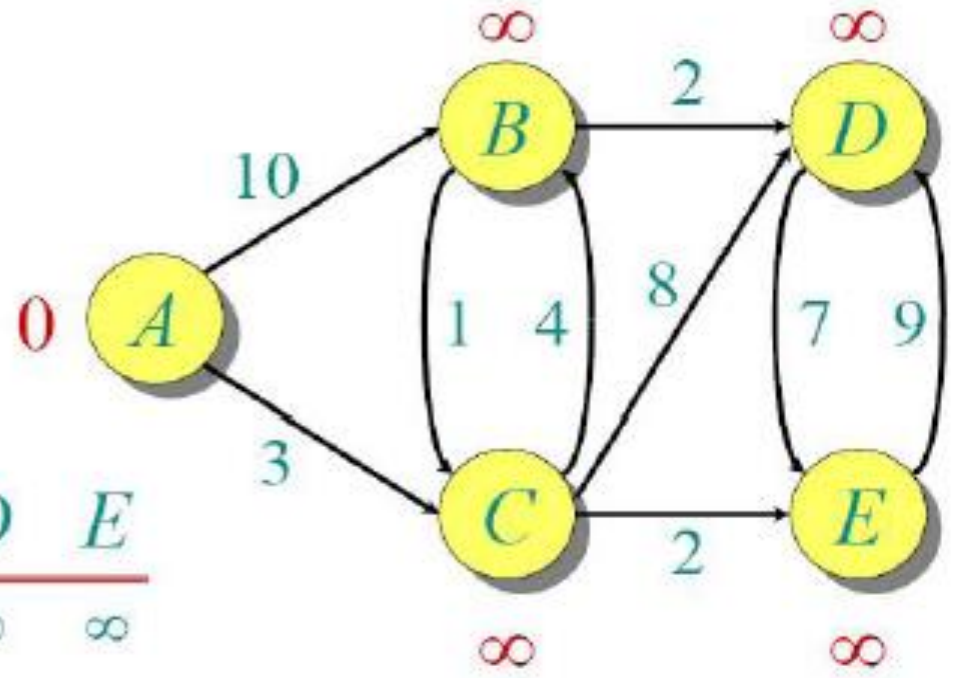
```
DIJKSTRA (G, w, s)
1. INITIALIZE-SINGLE-SOURCE (G, s)
2. S = ∅
3. Q = V [G]
4. while Q ≠ ∅
5.     u = EXTRACT-MIN(Q)
6.     S = S ∪ {u}
7.     for each vertex v ∈ Adj[u]
8.         RELAX (u, v, w)
```

```
INITIALIZE-SINGLE-SOURCE (G, s)
1. for each vertex v ∈ V [G]
2.     d [v] = ∞
3.     π [v] = NIL
4. d [s] = 0
```

```
RELAX (u, v, w)
1. if d [v] > d [u] + w(u, v)
2.     d [v] = d[u] + w(u, v)
3.     π [v] = u
```

Example

Initialize:



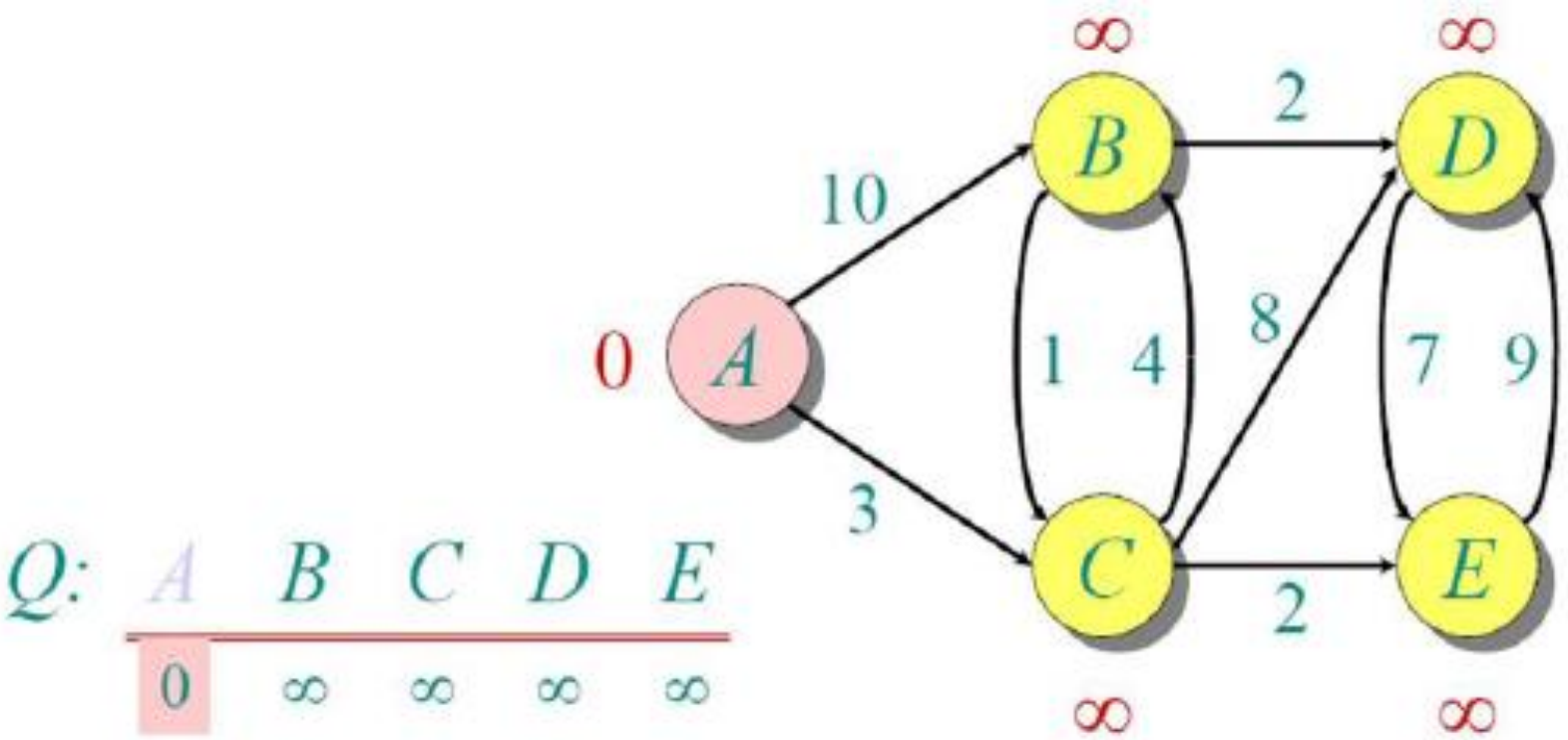
Q: A B C D E

0 ∞ ∞ ∞ ∞

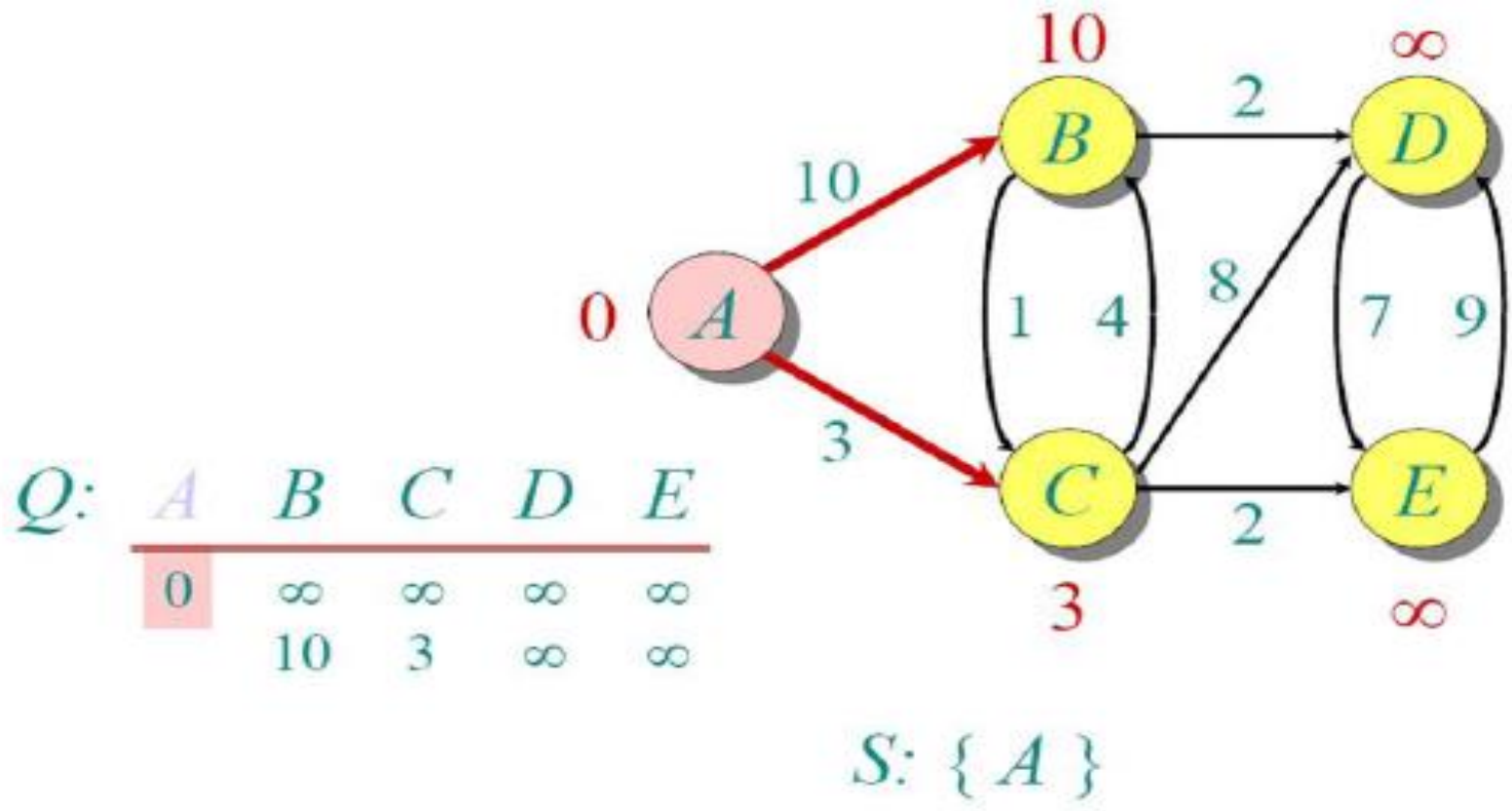
S: {}

u	A	B	C	D	E
π	N	N	N	N	N

Cont…



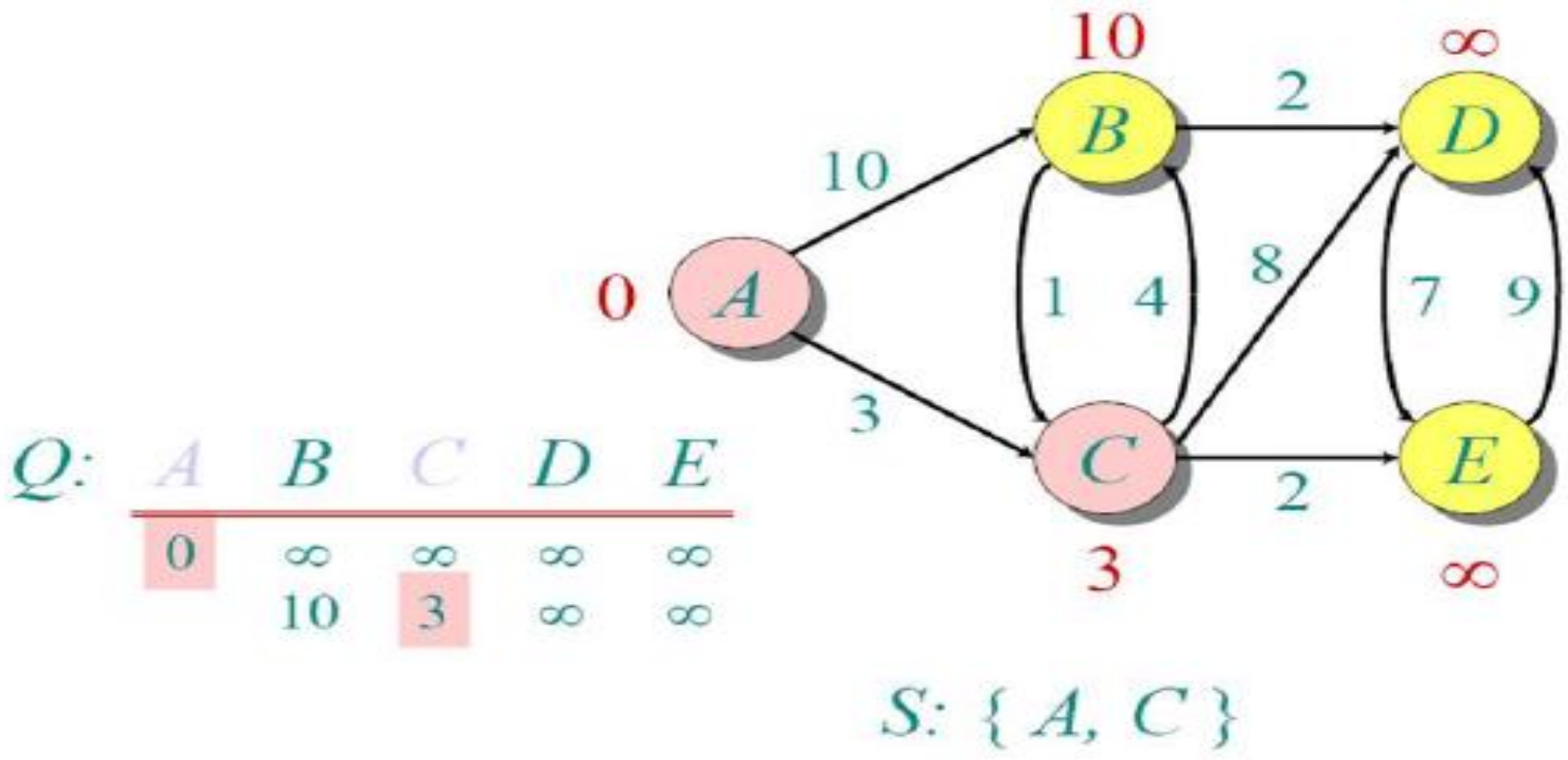
Cont…



Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞

u	A	B	C	D	E
π	N	A	A	N	N

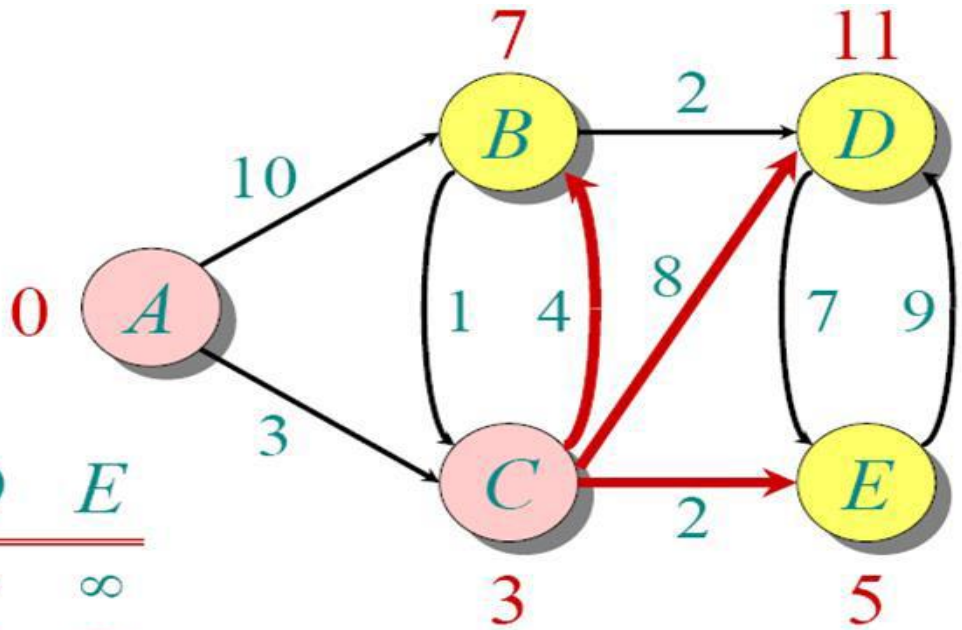
Cont…



Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞

u	A	B	C	D	E
π	N	A	A	N	N

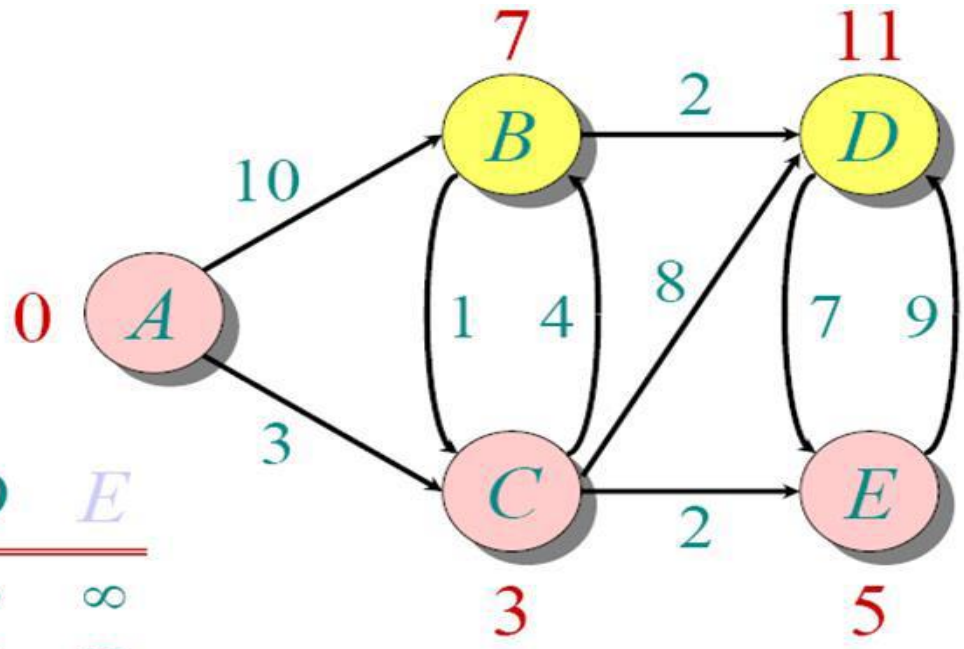
Cont…



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

u	A	B	C	D	E
π	N	C	A	C	C

Cont…

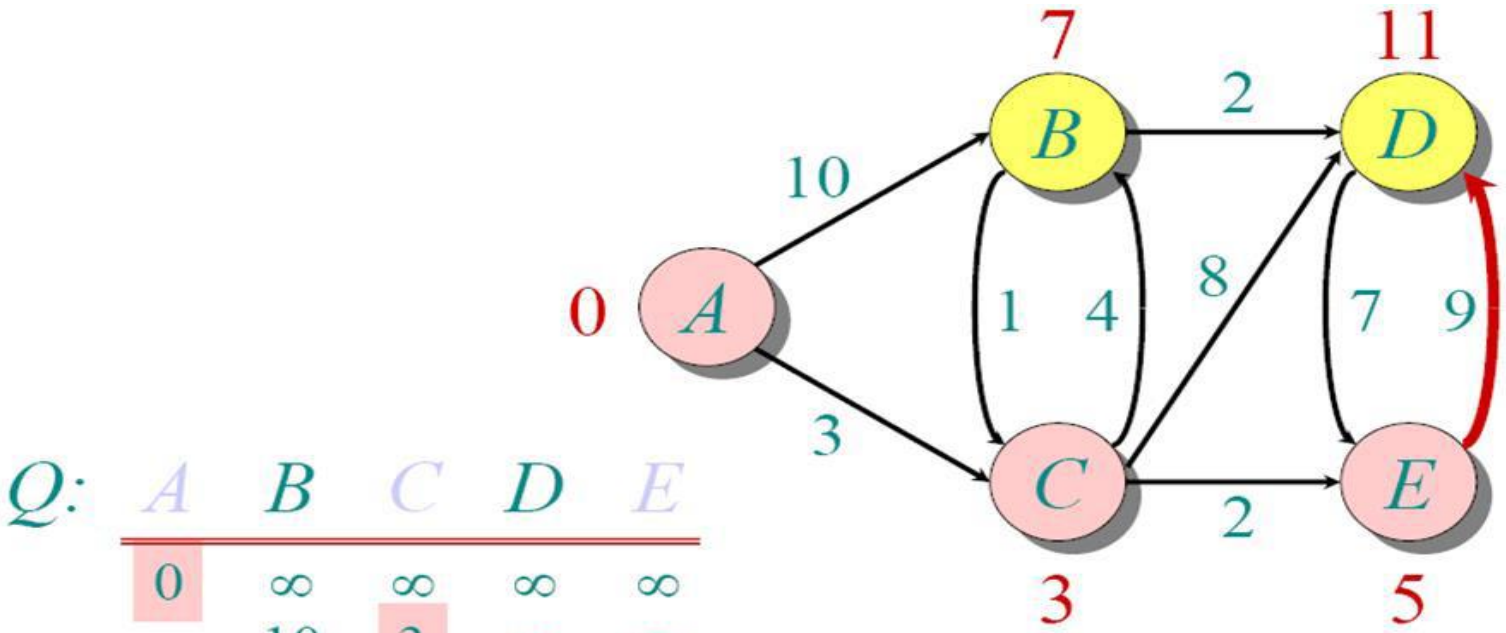


Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5

$S: \{ A, C, E \}$

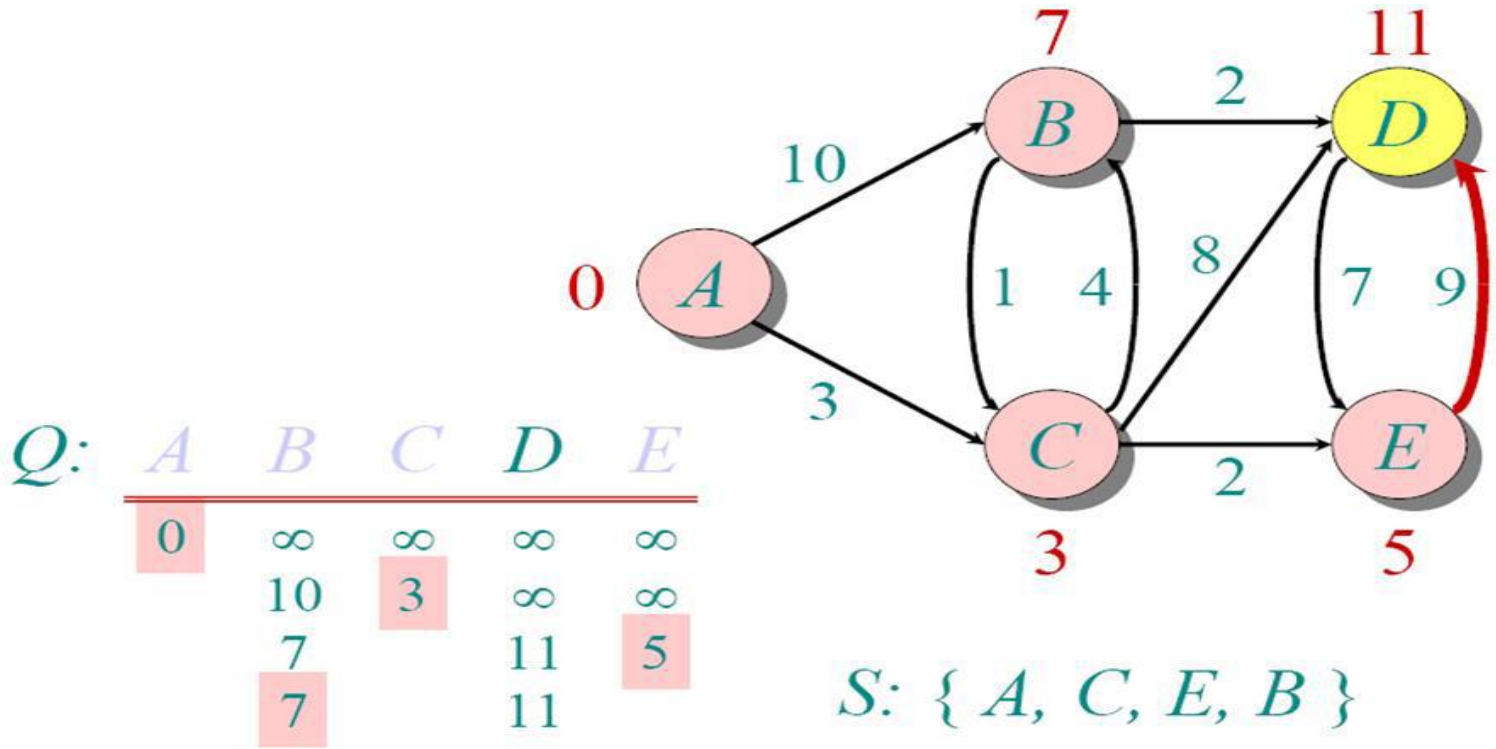
u	A	B	C	D	E
π	N	C	A	C	C

Cont…



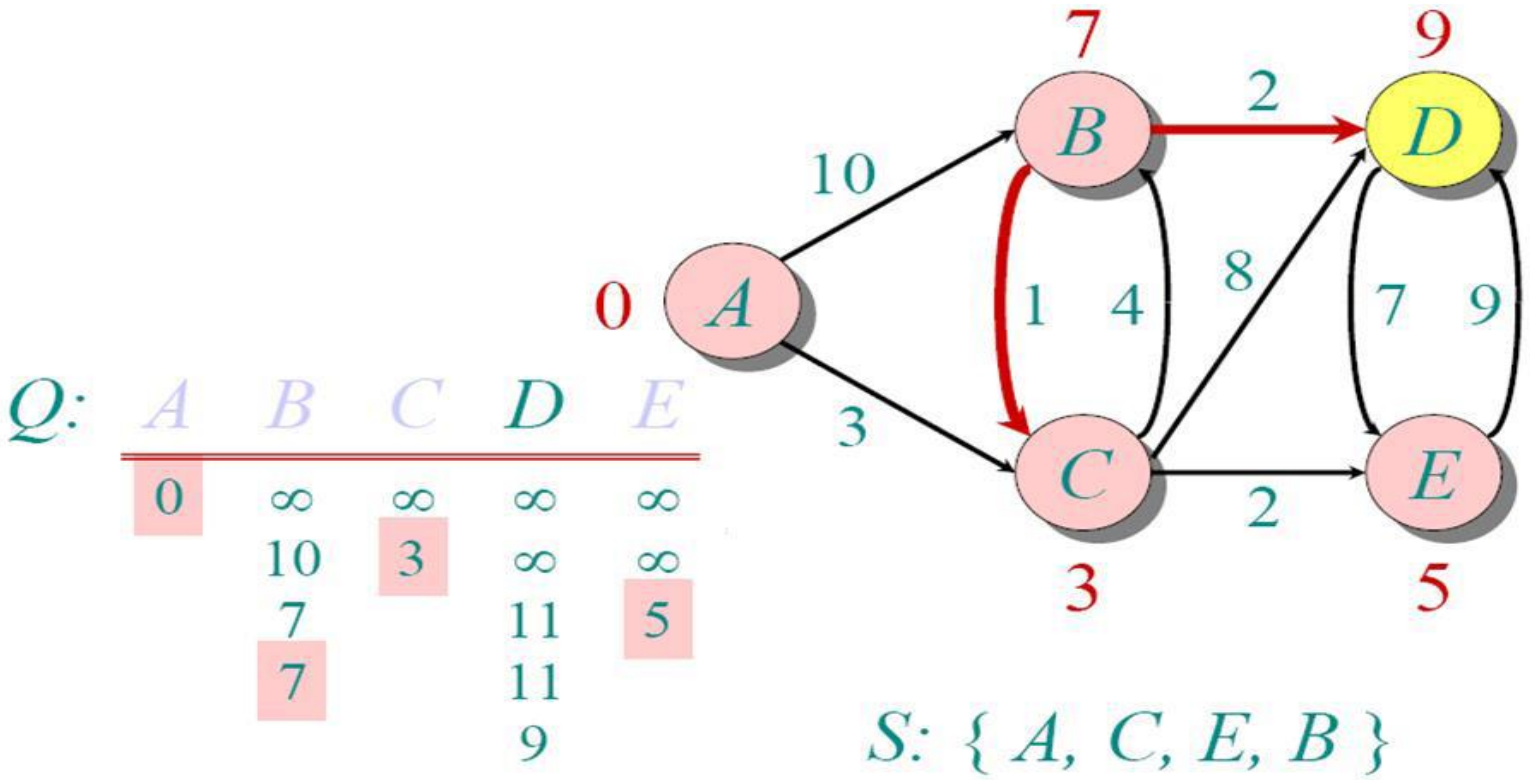
u	A	B	C	D	E
π	N	C	A	C	C

Cont…



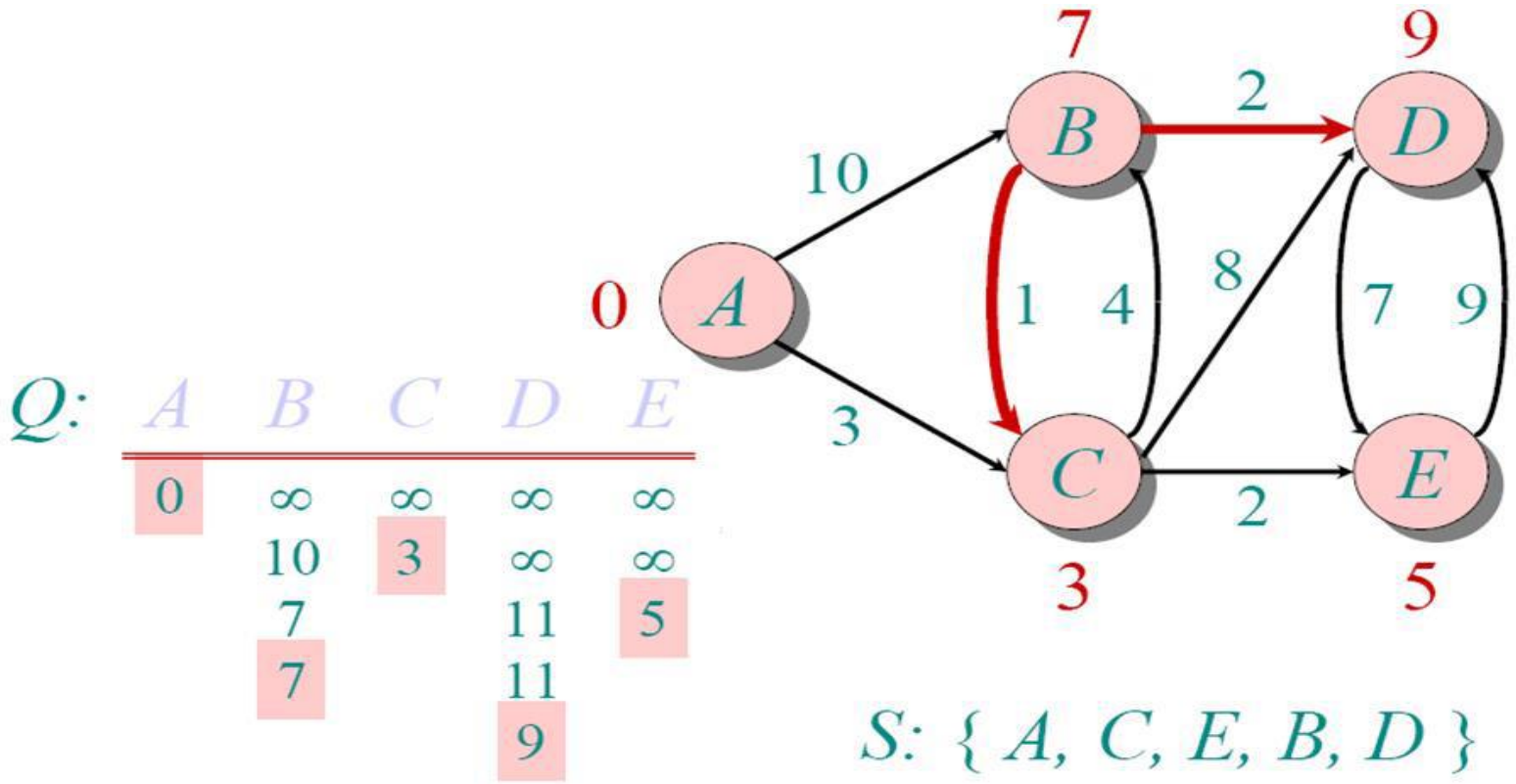
u	A	B	C	D	E
π	N	C	A	B	C

Cont…



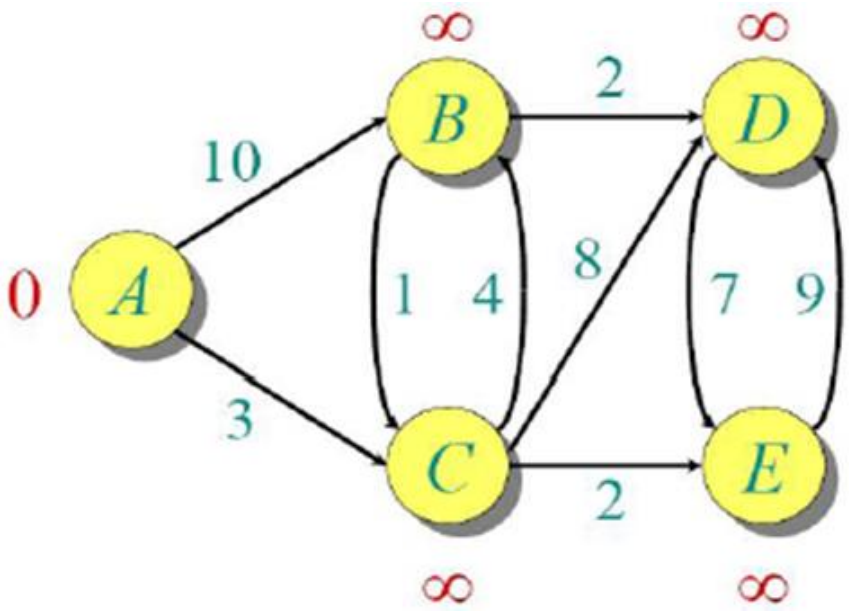
u	A	B	C	D	E
π	N	C	A	B	C

Cont…

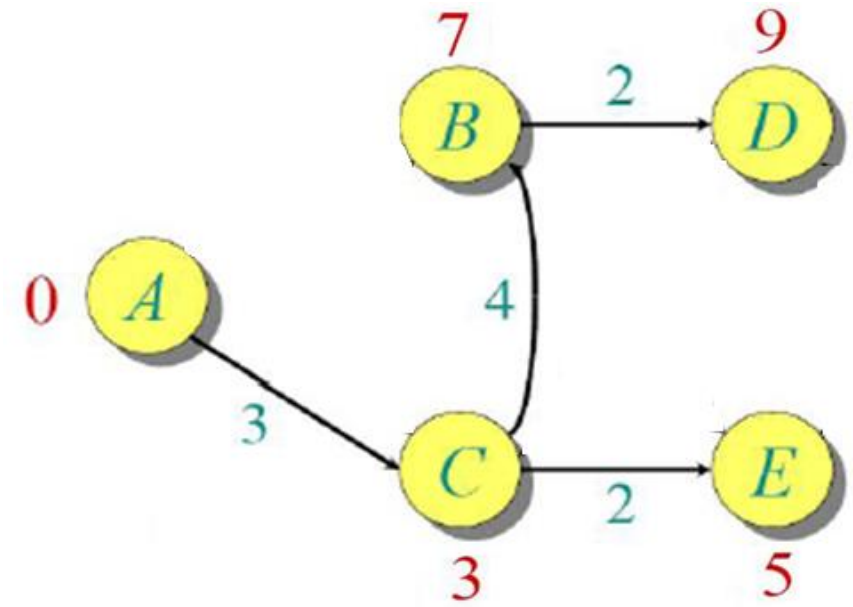


u	A	B	C	D	E
π	N	C	A	B	C

Cont…



Shortest Path



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

u	A	B	C	D	E
π	N	C	A	B	C

Running Times

- The simplest implementation is to store vertices in an array or linked list. This will produce a running time of $O(|V|^2 + |E|)$
- For any graphs with very few edges and many nodes or vertices, it can be implemented more efficiently storing the graph in an adjacency list using a binary heap or priority queue. This will produce a running time of

$$O((|E|+|V|) \log |V|)$$

Time Complexity: Using Linked list

The simplest implementation of the Dijkstra's algorithm stores vertices in an ordinary linked list or array

- Good for dense graphs (many edges)

- $|V|$ vertices and $|E|$ edges
- Initialization $O(|V|)$
- While loop $O(|V|)$
  - Find and remove min distance vertices $O(|V|)$
- Potentially $|E|$ updates
  - Update costs $O(1)$

Total time $O(|V^2| + |E|) = O(|V^2|)$