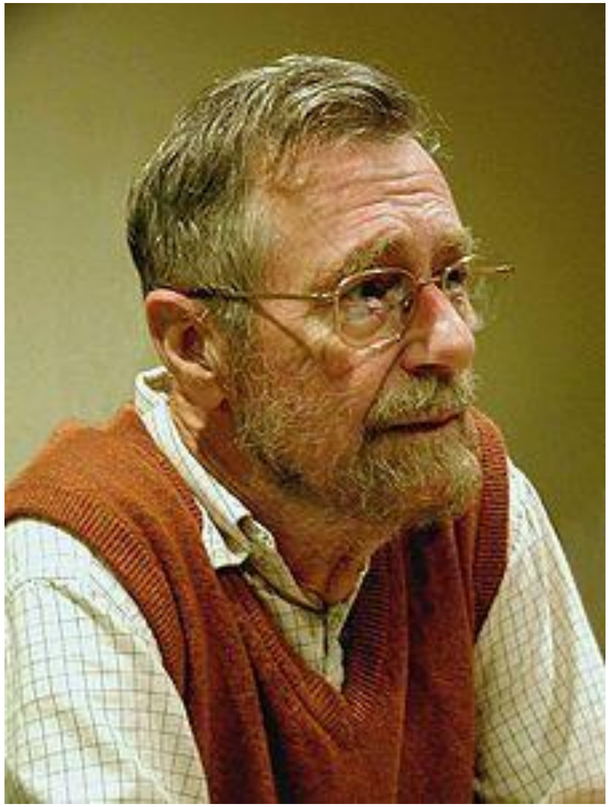
Dijkstra's algorithm

The author: Edsger Wybe Dijkstra



"Computer Science is no more about computers than astronomy is about telescopes."

http://www.cs.utexas.edu/~EWD/

Edsger Wybe Dijkstra

- May 11, 1930 – August 6, 2002

- Received the 1972 A. M. Turing Award, widely considered the most prestigious award in computer science.

- The Schlumberger Centennial Chair of Computer Sciences at The University of Texas at Austin from 1984 until 2000

- Made a strong case against use of the GOTO statement in programming languages and helped lead to its deprecation.

- Known for his many essays on programming.

Single-Source Shortest Path Problem

<u>Single-Source Shortest Path Problem</u> - The problem of finding shortest paths from a source vertex *v* to all other vertices in the graph.

Shortest Path Problem for Graphs

- Let G=(V,E) be a (di)graph. The shortest path between two vertices is a path with the shortest length (least number of edges).
- Breadth-first-search is an algorithm for finding shortest (link-distance) paths from a single source vertex to all other vertices.
- BFS processes vertices in increasing order of their distance from the root vertex.
- BFS has running time $O(|V|+|E|)$

Shortest Path Problem for Weighted Graphs

- Let G=(V,E) be a weighted digraph, with weight function $w: E \rightarrow R$ mapping edges to real-valued weights.
- The length of a path $p = \langle v_{0,} v_{1,} \ldots, v_k \rangle$ is the sum of the weights of its constituent edges:

  $$\text{length}(p) = w(v_{0,} v_1) + w(v_{1,} v_2) + \ldots\ldots + w(v_{k-1,} v_k)$$

- The distance from u to v, denoted d(u,v), is the length of the minimum length path if there is a path from u to v; and is ∞ otherwise.

Dijkstra's algorithm

Dijkstra's algorithm - is a solution to the single-source shortest path problem in graph theory.

Works on both directed and undirected graphs. However, all edges must have nonnegative weights.

Approach: Greedy

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

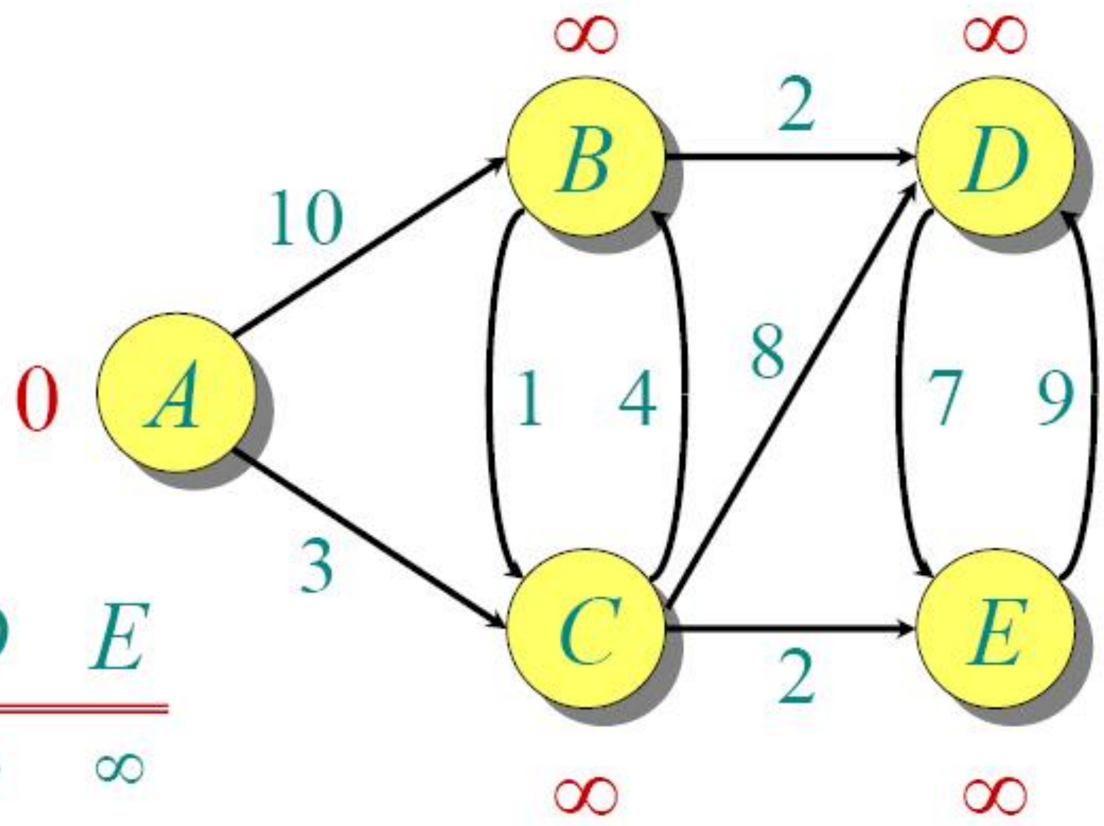
Output: Lengths of shortest paths (or the shortest paths themselves) from a given source vertex $v \in V$ to all other vertices

Dijkstra's algorithm - Pseudocode

dist[s] ←o (distance to source vertex is zero)
for all v ∈ V–{s}
    do dist[v] ←∞ (set all other distances to infinity)
        predecessor [v] = nil (list of predecessors of each node)
S←∅ (S, the set of visited vertices is initially empty)
Q←V (Q, the queue initially contains all vertices)
while Q ≠∅ (while the queue is not empty)
do u ← ExtractMin (Q,dist) (select the element of Q with the min. distance)
    S←S∪{u} (add u to list of visited vertices)
    for all v ∈ neighbors[u]
        do if dist[v] > dist[u] + w(u, v) (if new shortest path found)
            then dist[v] ←dist[u] + w(u, v) (set new value of shortest path)
                predecessor [v] = u (list of predecessors of each node
return dist

Dijkstra Animated Example

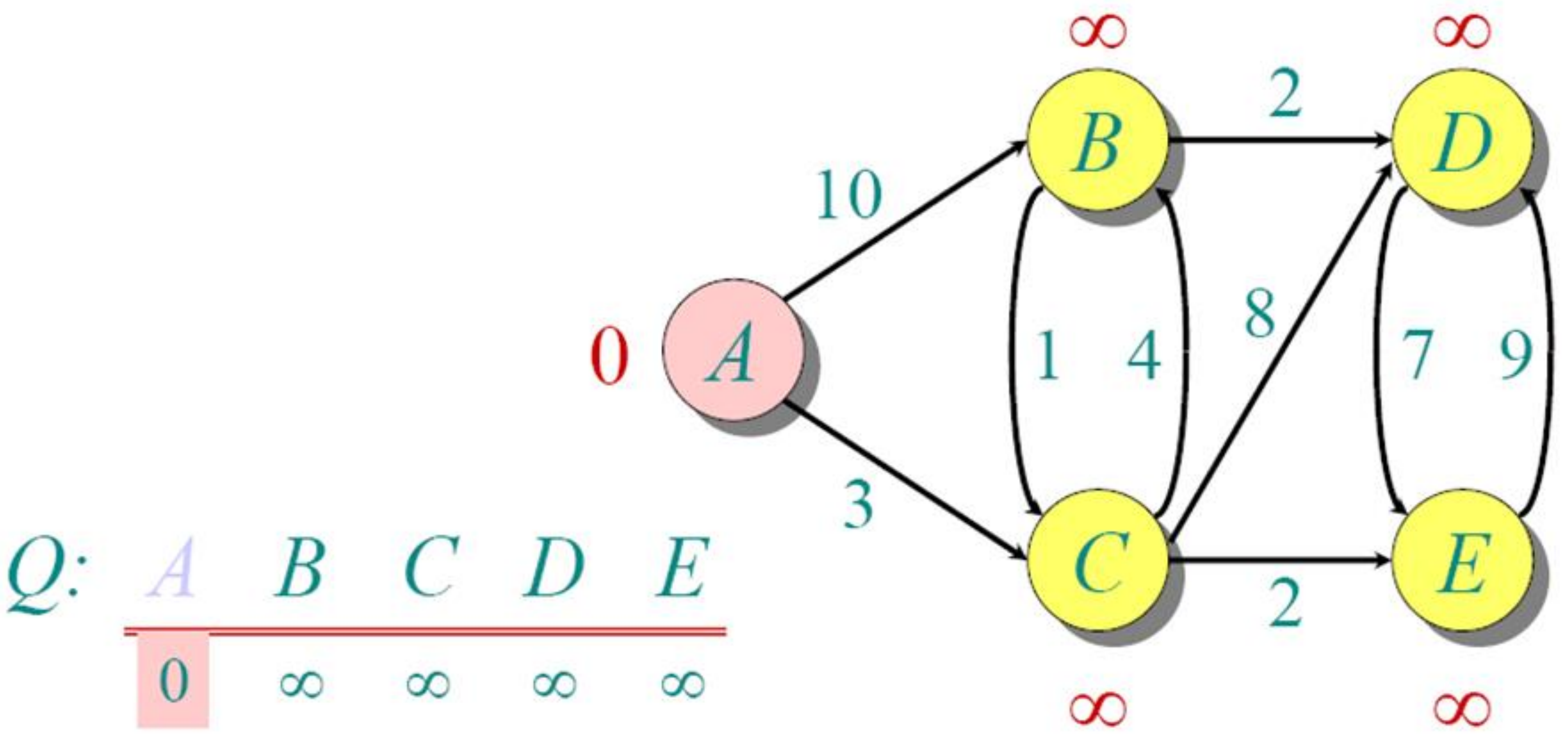
Initialize:



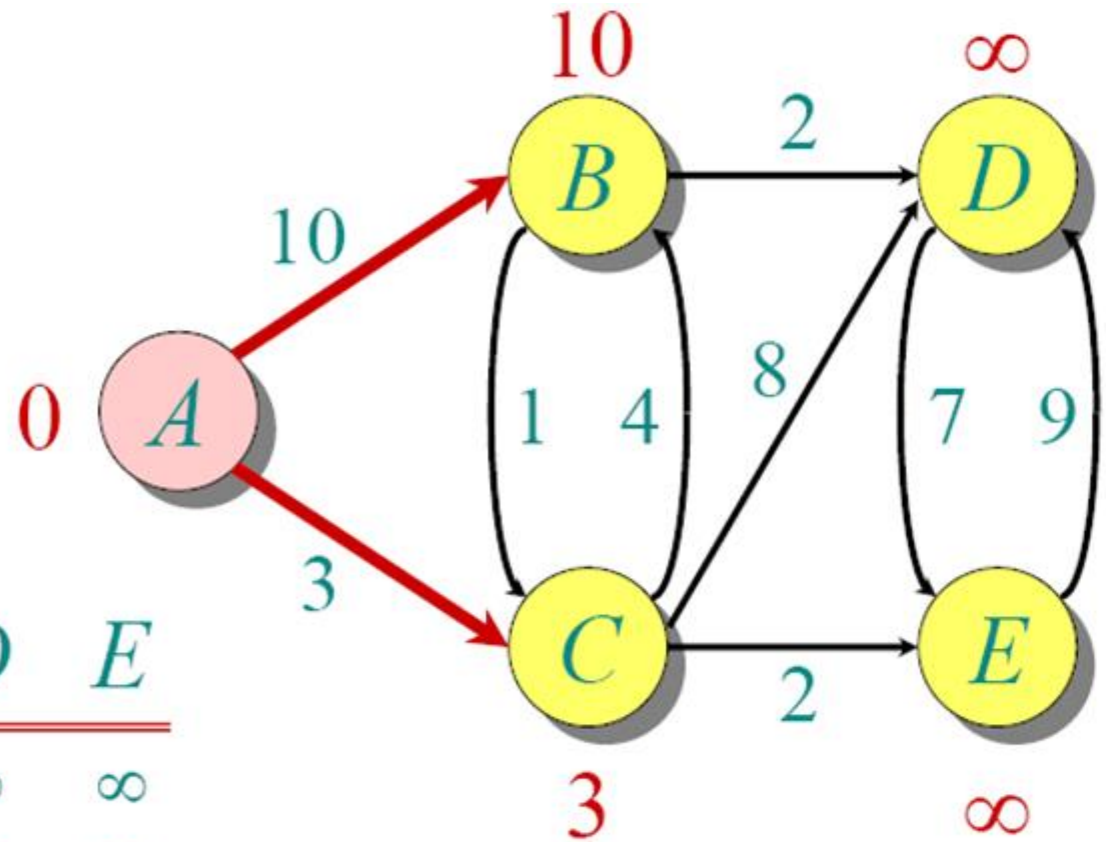
Q:	A	B	C	D	E
	0	∞	∞	∞	∞

S: {}

Dijkstra Animated Example



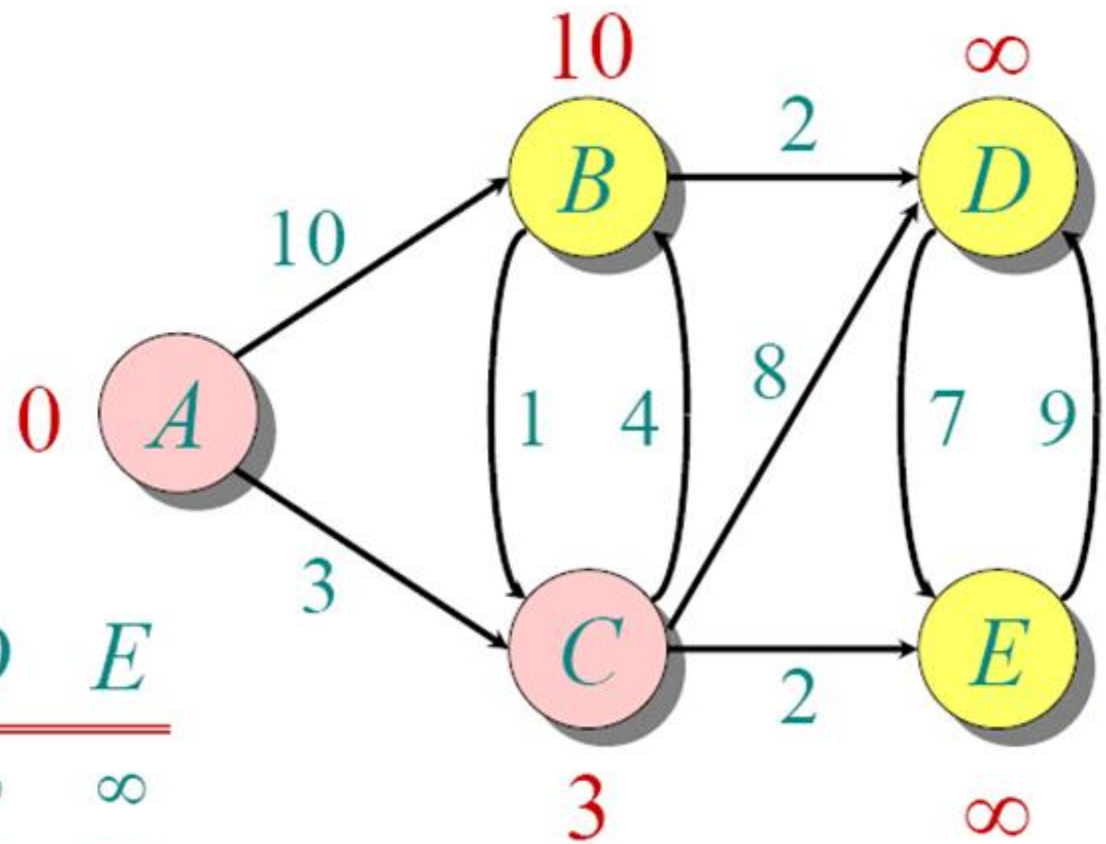
Dijkstra Animated Example



Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞

$S: \{ A \}$

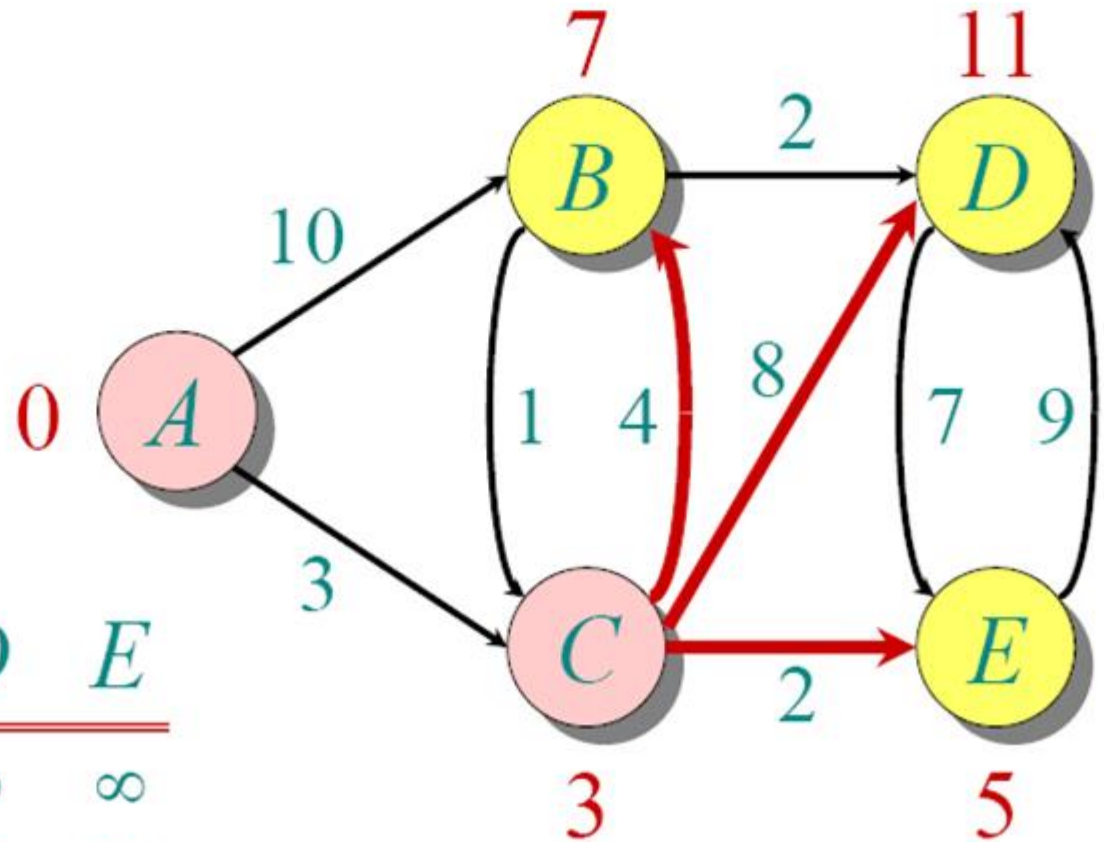
Dijkstra Animated Example



Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞

S: { A, C }

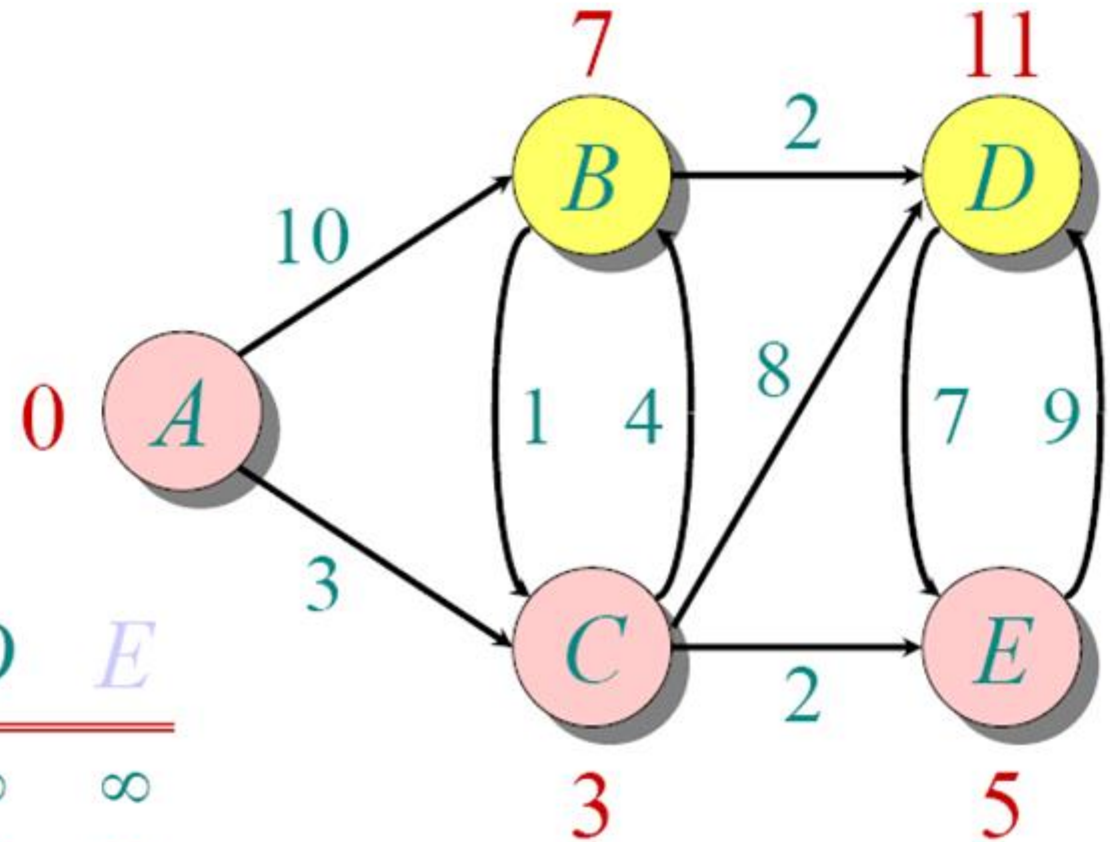
Dijkstra Animated Example



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

S: { A, C }

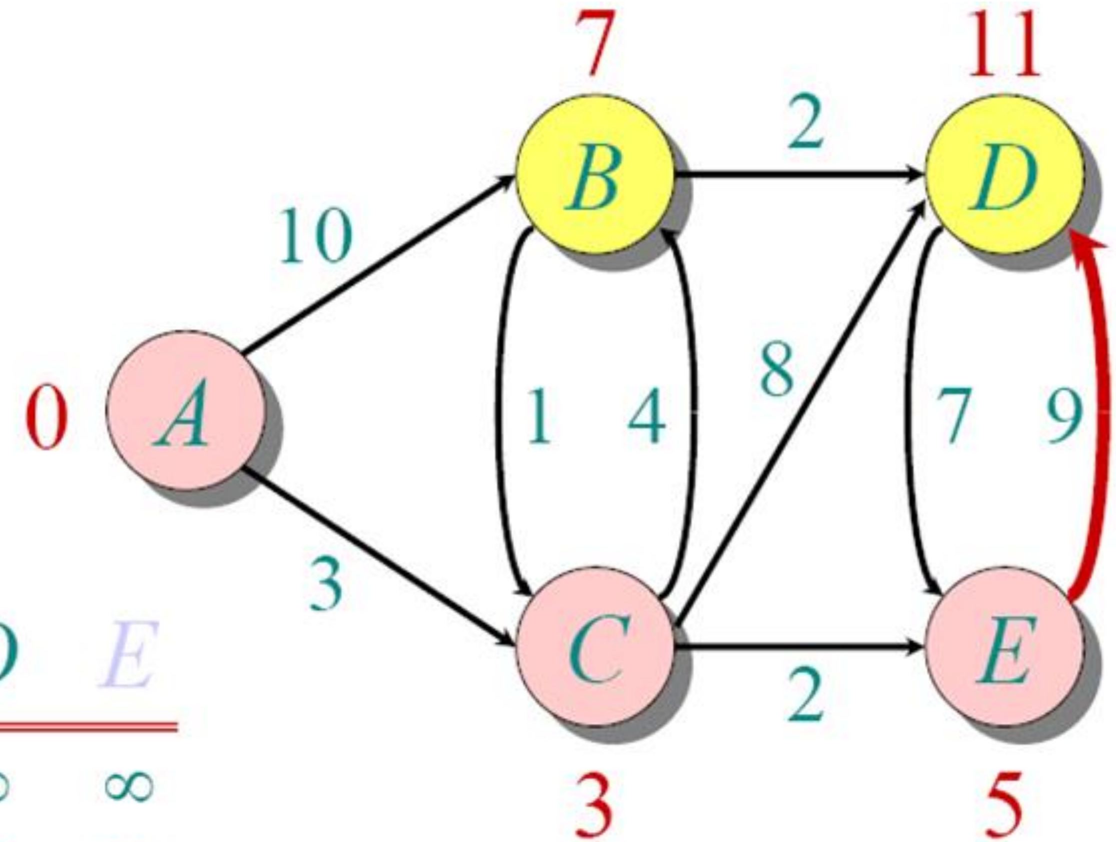
Dijkstra Animated Example



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

S: { A, C, E }

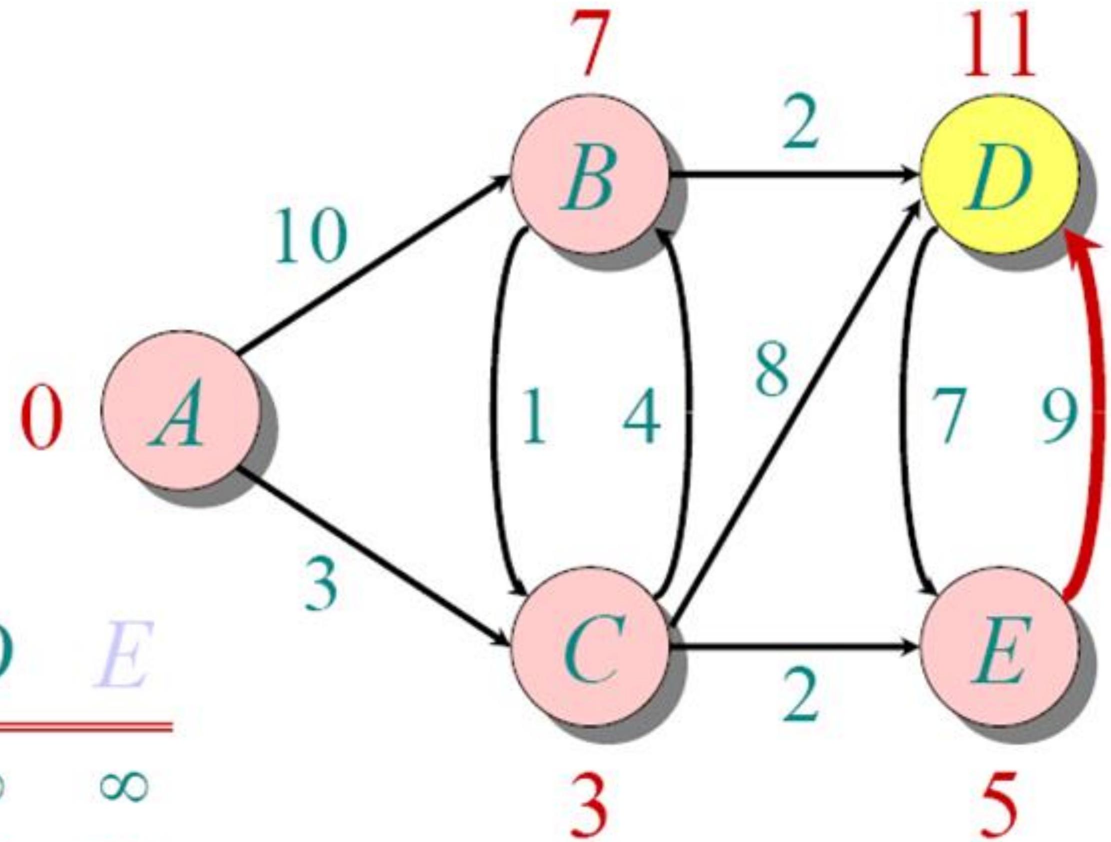
Dijkstra Animated Example



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

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

Dijkstra Animated Example

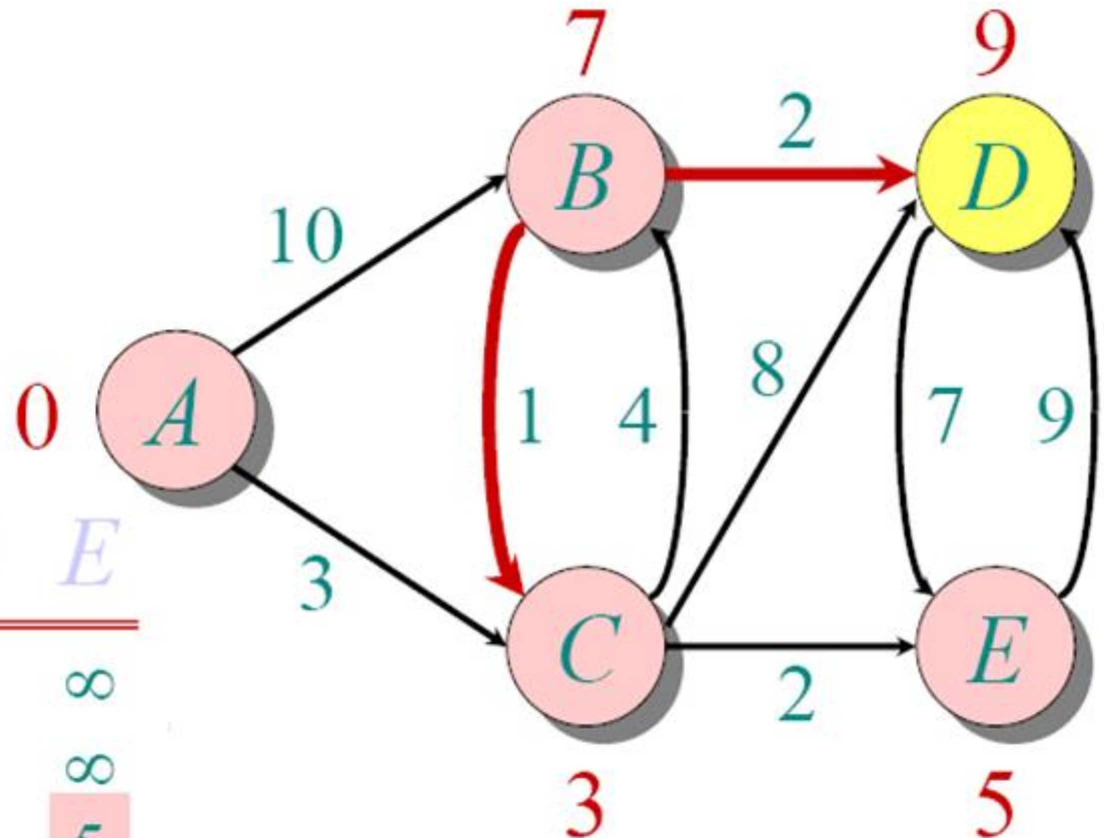


Q:	A	B	C	D	E
	$\boxed{0}$	∞	∞	∞	∞
		10	$\boxed{3}$	∞	∞
		7		11	$\boxed{5}$
		$\boxed{7}$		11	

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

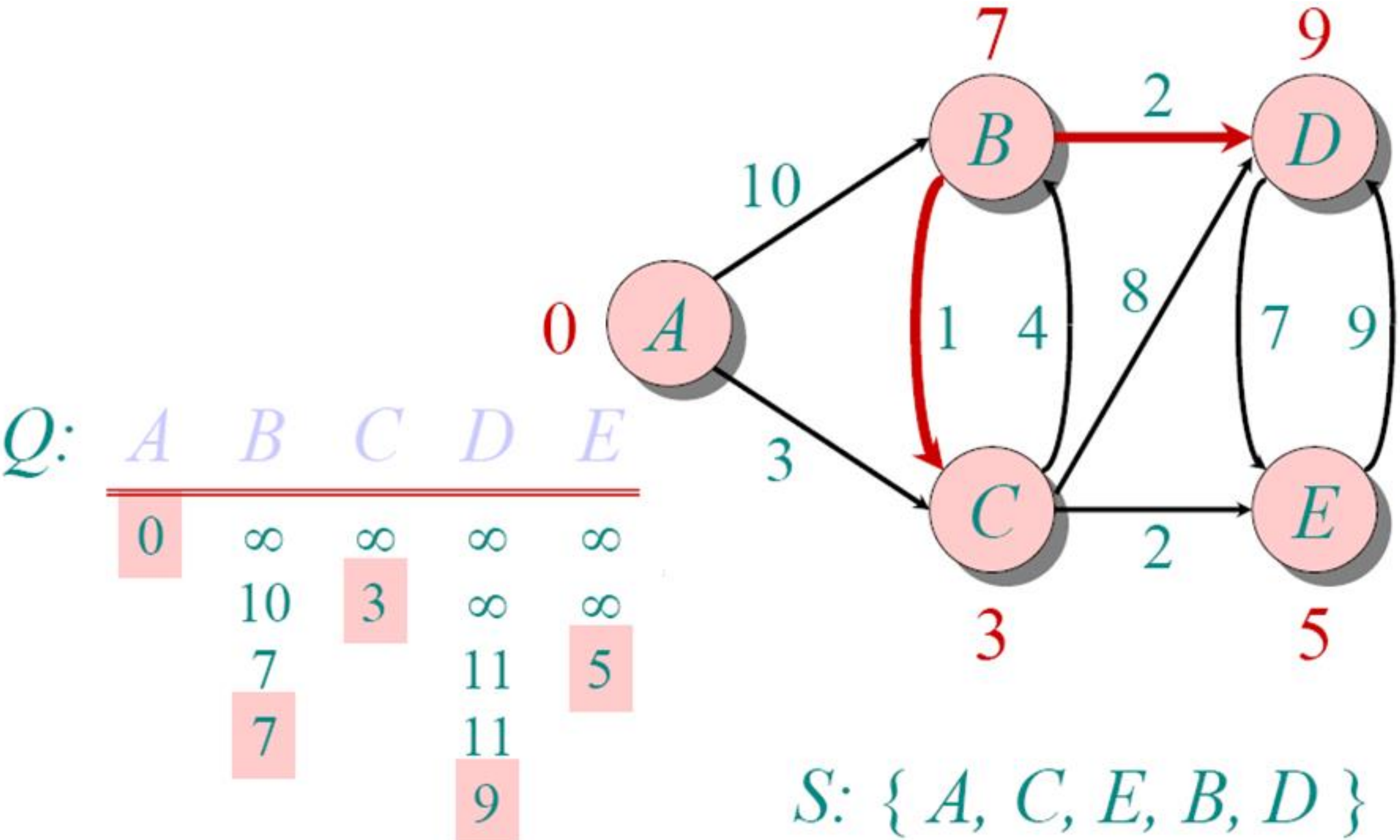
Dijkstra Animated Example

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



S: { A, C, E, B }

Dijkstra Animated Example



Dijkstra's algorithm - Pseudocode

```
dist[s] ←0
for  all v ∈ V–{s}                                   O(V)
       do  dist[v] ←∞
         predecessor [v] = nil
S←∅
Q←V
while Q ≠∅                                           O(V)
do   u ← ExtractMin(Q, dist)                         O(lg|V|) OR O(|V|)
     S←S∪{u}
      for all v ∈ neighbors[u]                        O(|E|)
            do  if   dist[v] > dist[u] + w(u, v)
                  then  dist[v] ←dist[u] + w(u, v) O(|lgV|) OR O(1)
                        // Decrease-Key (Q, v, dist[v])
                        predecessor [v] = u
return dist
```

Implementations and 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 sparse graphs, or graphs with very few edges and many nodes, 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

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

Dijkstra's Algorithm - Why It Works

- As with all greedy algorithms, we need to make sure that it is a correct algorithm (e.g., it *always* returns the right solution if it is given correct input).

- A formal proof would take longer than this presentation, but we can understand how the argument works intuitively.

DIJKSTRA'S ALGORITHM - WHY IT WORKS

- To understand how it works, we'll go over the previous example again. However, we need two mathematical results first:

- **Lemma 1**: Triangle inequality
  If $\delta(u,v)$ is the shortest path length between u and v,
  $\delta(u,v) \leq \delta(u,x) + \delta(x,v)$
- **Lemma 2**:
  The subpath of any shortest path is itself a shortest path.

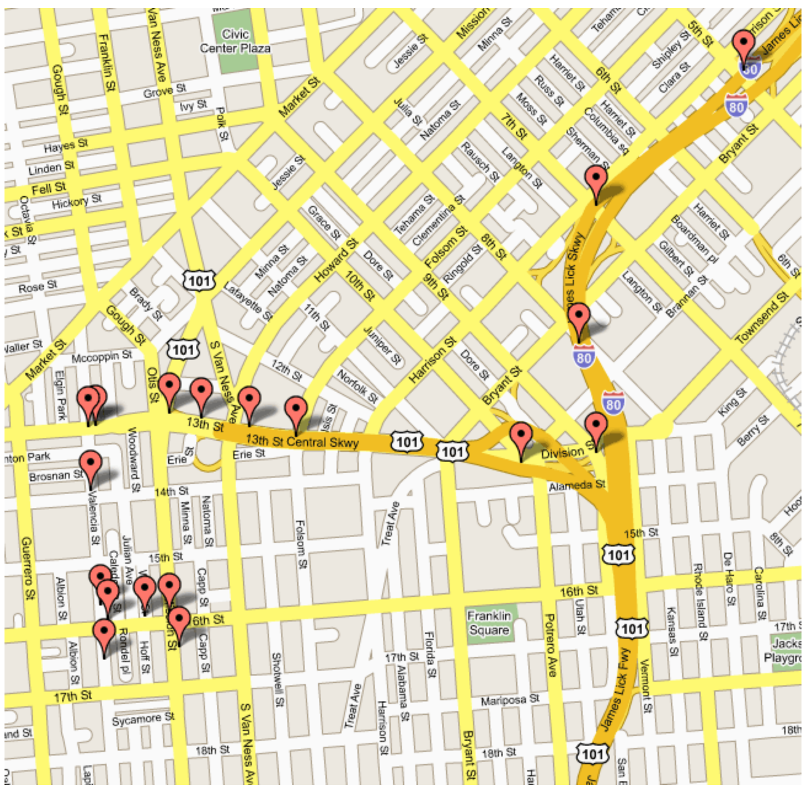
- The key is to understand why we can claim that anytime we put a new vertex in S, we can say that we already know the shortest path to it.
- Now, back to the example…

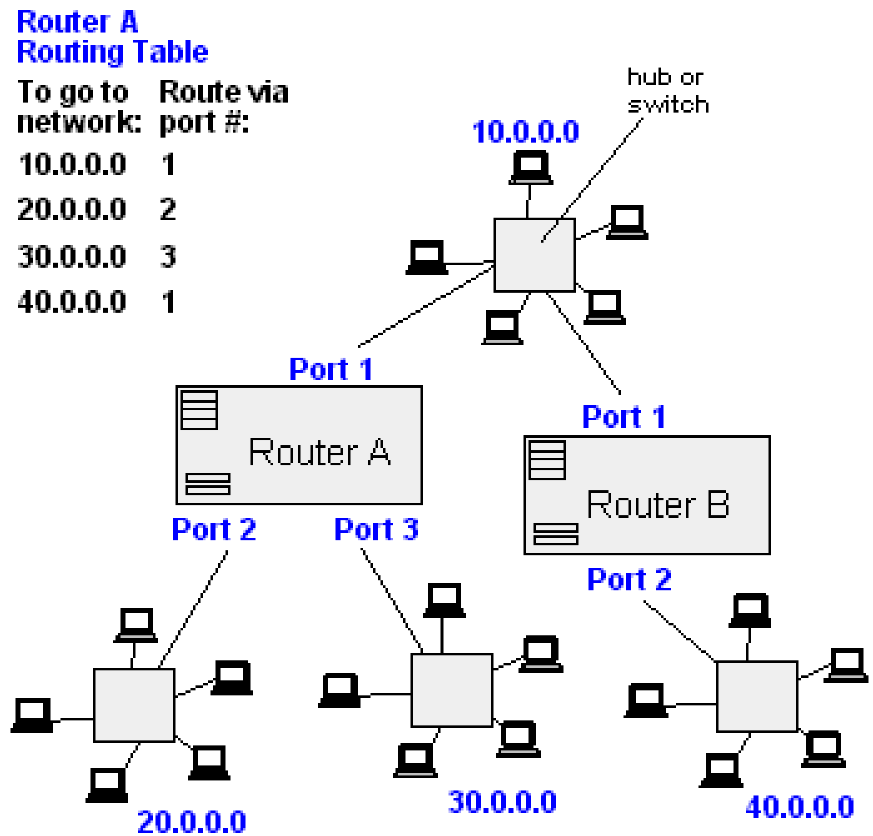
DIJKSTRA'S ALGORITHM - WHY USE IT?

- As mentioned, Dijkstra’s algorithm calculates the shortest path to every vertex.
- However, it is about as computationally expensive to calculate the shortest path from vertex *u* to every vertex using Dijkstra’s as it is to calculate the shortest path to some particular vertex *v*.
- Therefore, anytime we want to know the optimal path to some other vertex from a determined origin, we can use Dijkstra’s algorithm.

Applications of Dijkstra's Algorithm

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





References

- www.cs.utexas.edu/~tandy/barrera.**ppt**