Programming and Algorithms

COMP1038.PGA

Week 10 – Lecture 3: Trees and Graphs

Dr. Pushpendu Kar

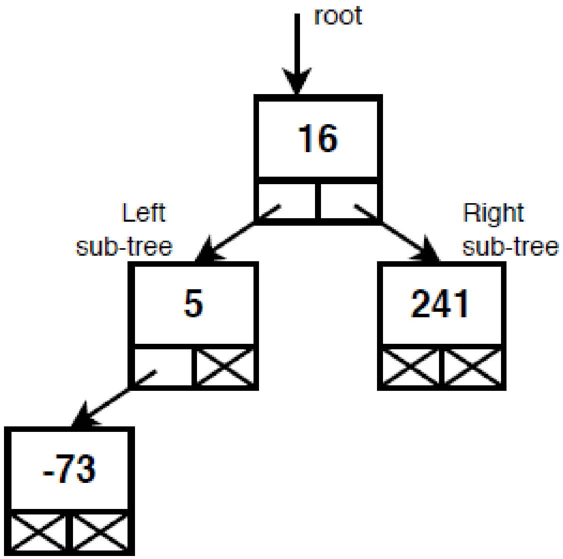
Overview

- Tree
- Graph
  - Dijkstra’s Algorithm

Tree

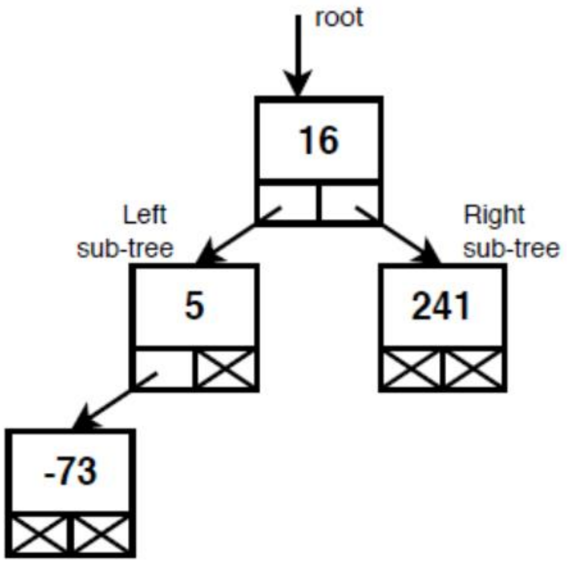
Tree: introduction

- Trees are hierarchical data structures containing nodes which store a value and references to 2 or more subtrees.
- The "start" of the tree is the root node.
- Nodes with no subtrees are called leaf nodes.
- A binary tree is a tree where each node has exactly two possible children.



Tree: binary search tree

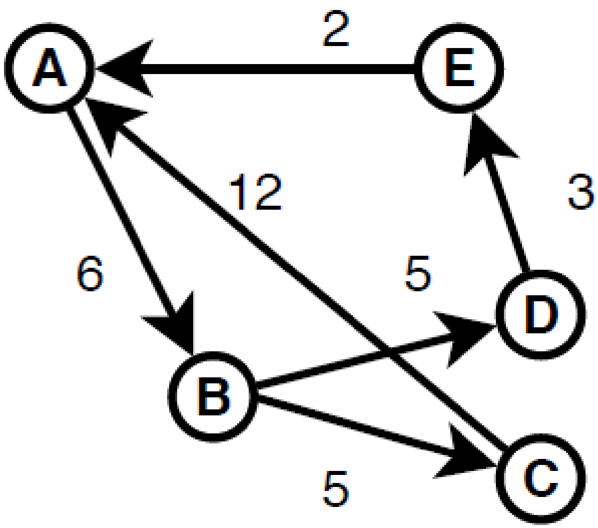
- BSTs are binary trees in which the values are stored in the tree in some specified order.
- Eg, for an integer BST,
  - every value in the left sub-tree < value in the node
  - every value in the right sub-tree >= the value in the node.
- Searching for values in a BST can be extremely quick because each comparison discards half the remaining values (on average).
- Inserting/removing nodes is more complex as it may require moving existing nodes.



Graph

Graph: introduction

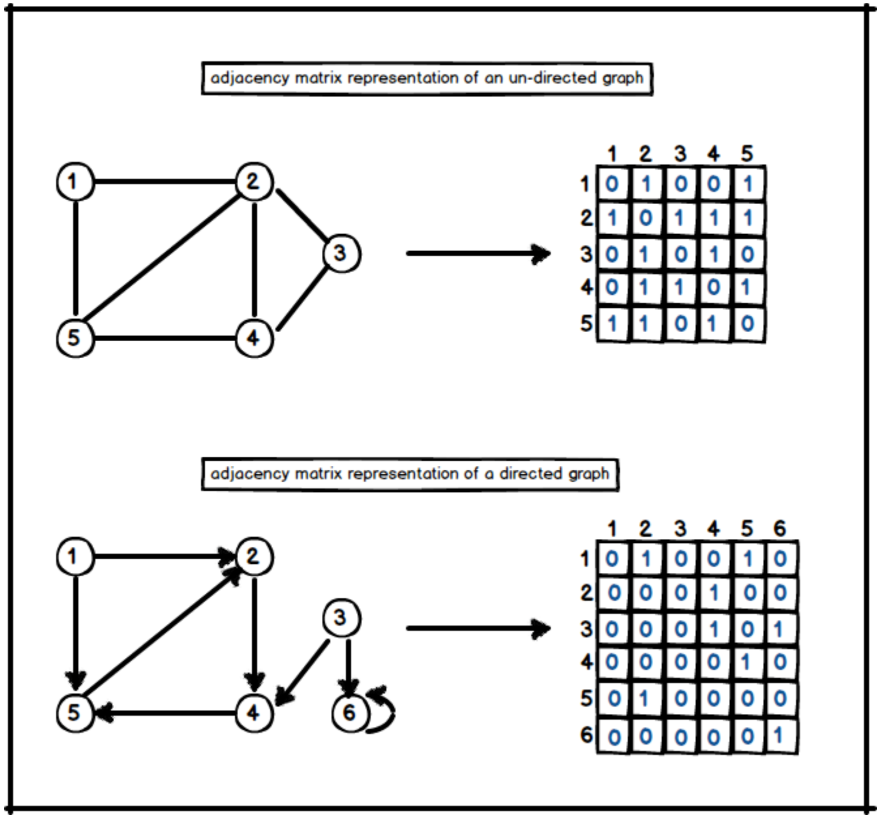
- A graph represents a set of vertices (or nodes) and a set of edges which are connections between vertices.
- Edges can be directed or undirected.
- Edges can be weighted or unweighted.
- Graphs can be connected (for any 2 vertices, there is a path between them) or unconnected.
- ...and many other possible properties
  - Graph theory is a major branch of discrete maths.



Graph: adjacency matrix

- Vertices are assigned to integer IDs.
- Store graph as a 2D array of integers.
- A_{ij} contains a value if there is an edge from i to j.
  - Store edge present? Number of edges? Weight of edge?
- Very fast look-up for edge between two vertices.
- Low memory usage but wastes lots of space for sparse graphs.
- Requires whole array to be changed when adding/removing vertices.

Graph: adjacency matrix cont...



adjacency matrix representation of an un-directed graph

	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

adjacency matrix representation of a directed graph

	1	2	3	4	5	6
1	0	1	0	0	1	0
2	0	0	0	1	0	0
3	0	0	0	1	0	1
4	0	0	0	0	1	0
5	0	1	0	0	0	0
6	0	0	0	0	0	1

Graph: applications

- Modeling relationships between users in social media networks (vertex=user, edge=friends).
- Navigation through cities vertex=road junction, edge=road between junction, weight=distance).
- Modelling state machines (vertex=state, edge=valid transition).

Introduction to Dijkstra's Algorithm

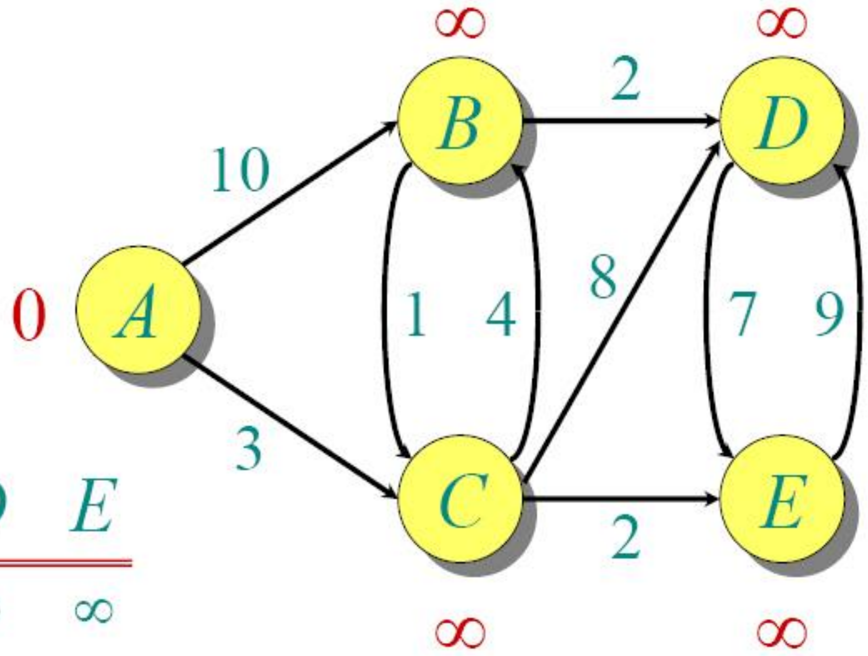
- **Dijkstra's algorithm** - is a solution to the single-source shortest path problem in graph theory.
- Works on both directed and undirected graphs. However, all edges must have nonnegative weights.
- Approach: Greedy
- Input: Weighted graph G={E,V} and source vertex *v*∈V, such that all edge weights are nonnegative
- Output: Lengths of shortest paths (or the shortest paths themselves) from a given source vertex *v*∈V to all other vertices

Dijkstra's Algorithm: Pseudocode

```
$dist[s] \leftarrow 0$                                (distance to source vertex is zero)
for all $v \in V - \{s\}$
        do $dist[v] \leftarrow \infty$                (set all other distances to infinity)
$S \leftarrow \emptyset$                              (S, the set of visited vertices is initially empty)
$Q \leftarrow V$                                      (Q, the queue initially contains all vertices)
while $Q \neq \emptyset$                              (while the queue is not empty)
do  $u \leftarrow$ mindistance$(Q, dist)$             (select the element of Q with the min. distance)
      $S \leftarrow S \cup \{u\}$                     (add u to list of visited vertices)
      for all $v \in neighbors[u]$
            do if $dist[v] > dist[u] + w(u, v)$              (if new shortest path found)
                   then $d[v] \leftarrow d[u] + w(u, v)$     (set new value of shortest path)
                                                             (if desired, add traceback code)
return $dist$
```

Dijkstra Example

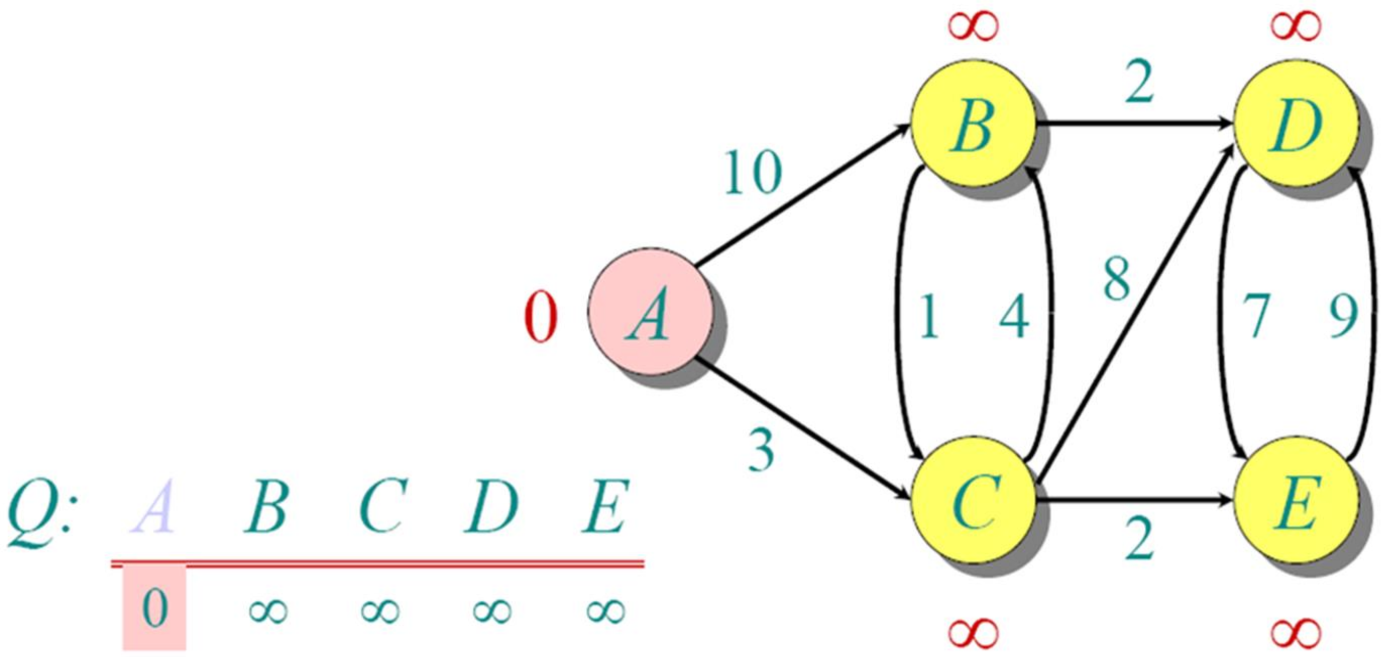
Initialize:



Q:	A	B	C	D	E
	0	∞	∞	∞	∞

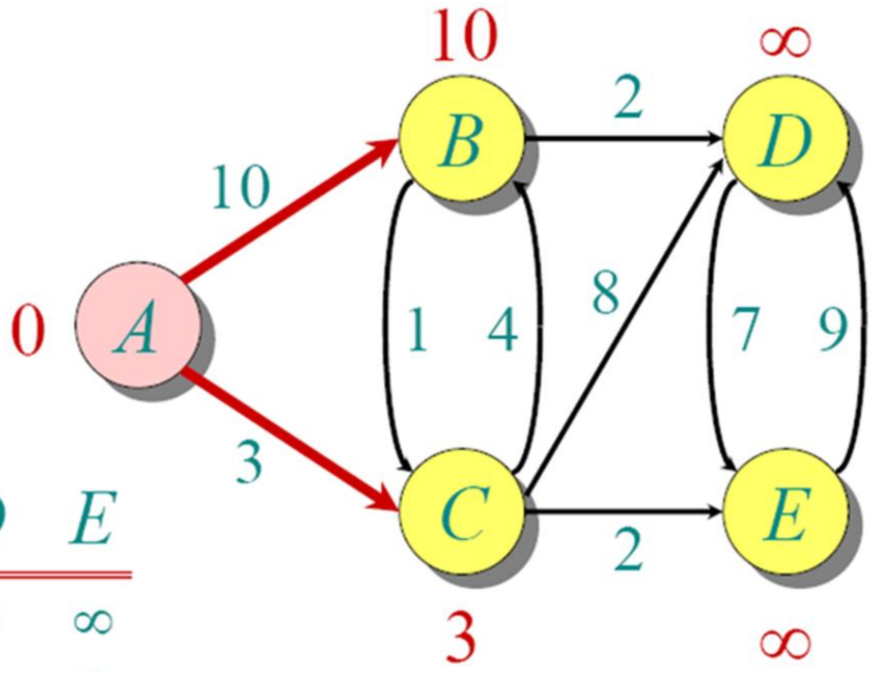
S: {}

Dijkstra Example



Q:	A	B	C	D	E
	0	∞	∞	∞	∞

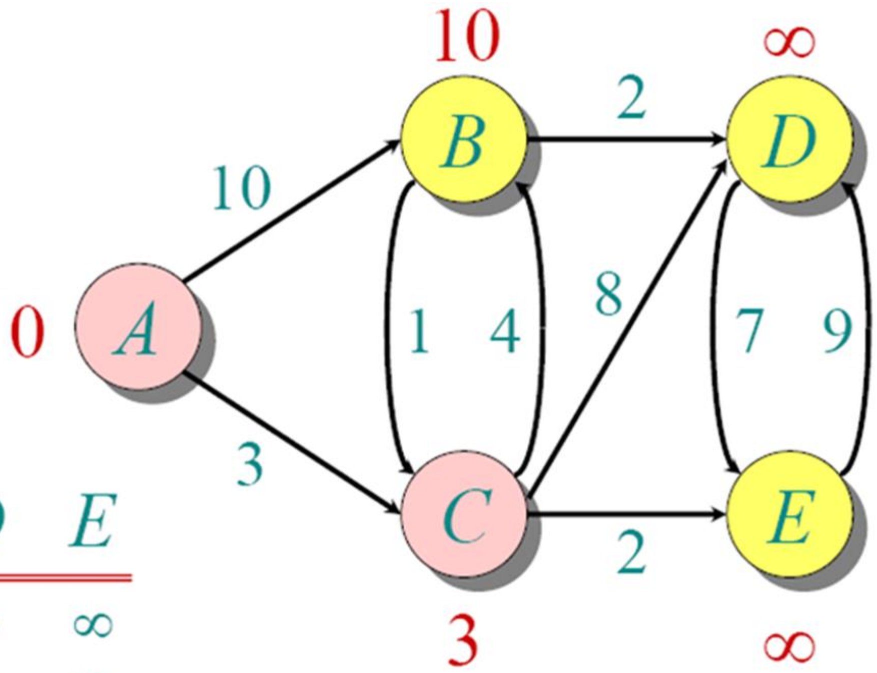
Dijkstra Example



Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞

S: { A }

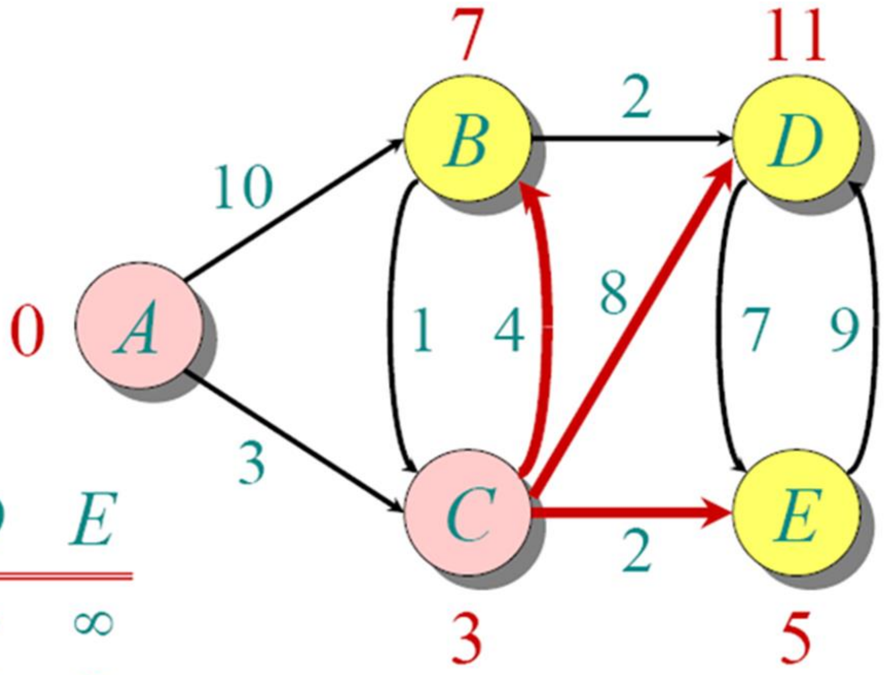
Dijkstra Example



Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞

S: { A, C }

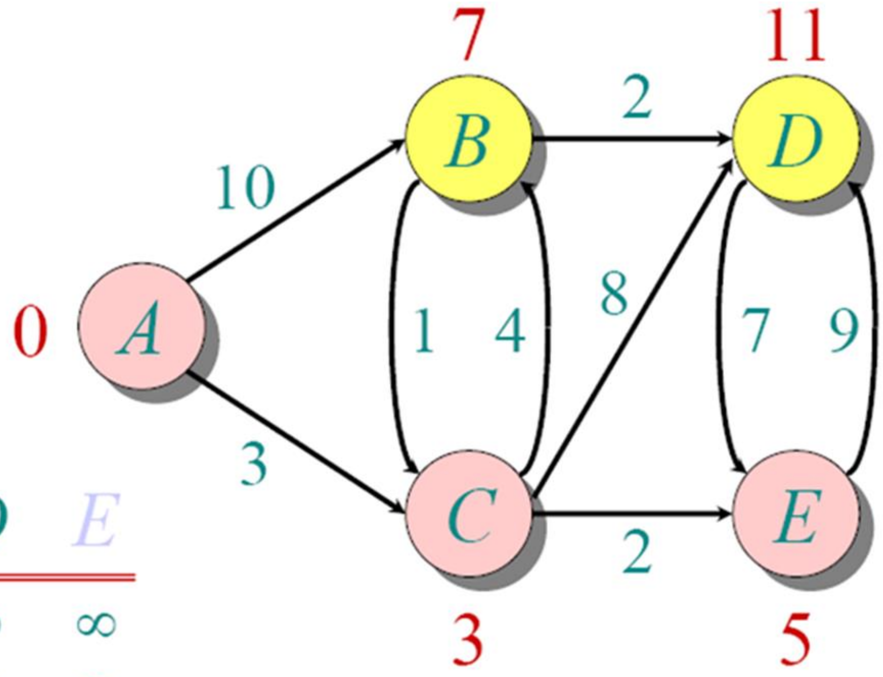
Dijkstra Example



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

S: { A, C }

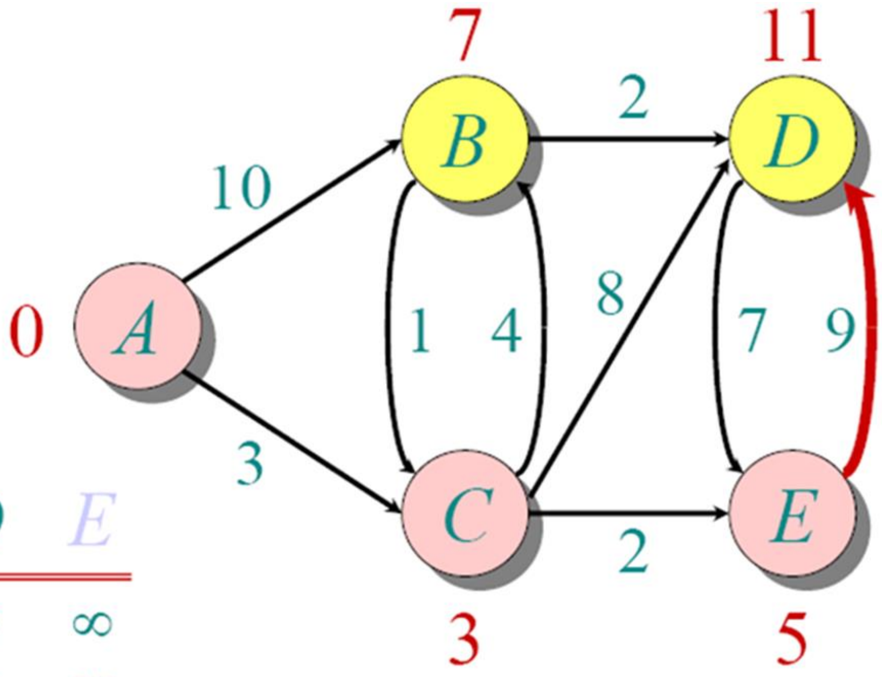
Dijkstra Example



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

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

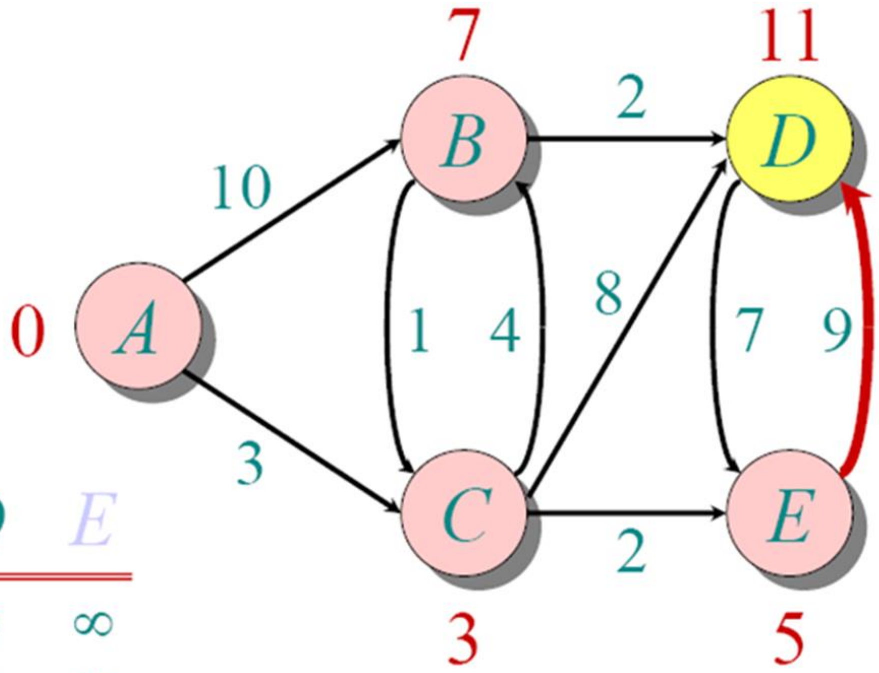
Dijkstra Example



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

S: { A, C, E }

Dijkstra Example

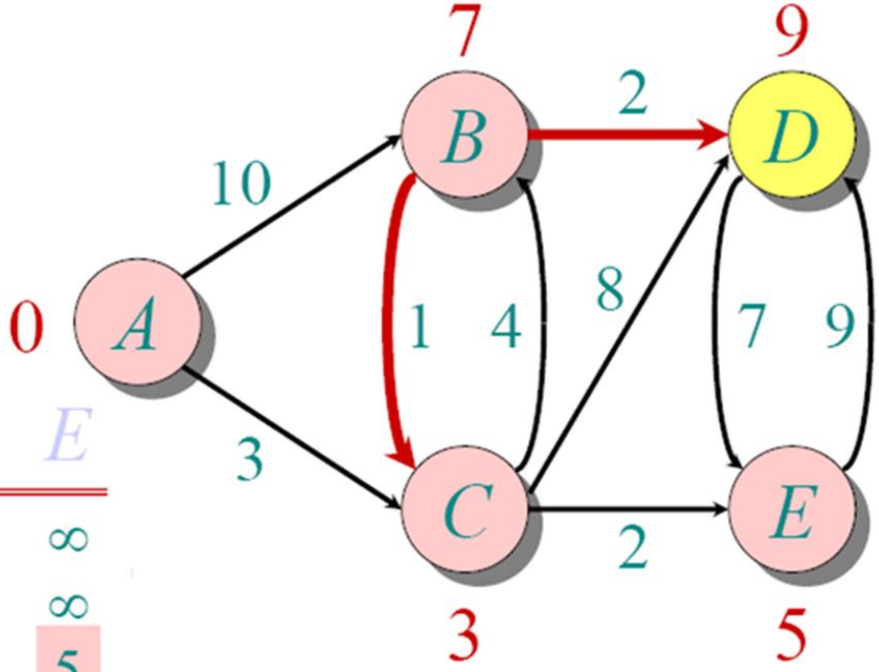


Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5
		7		11	

S: { A, C, E, B }

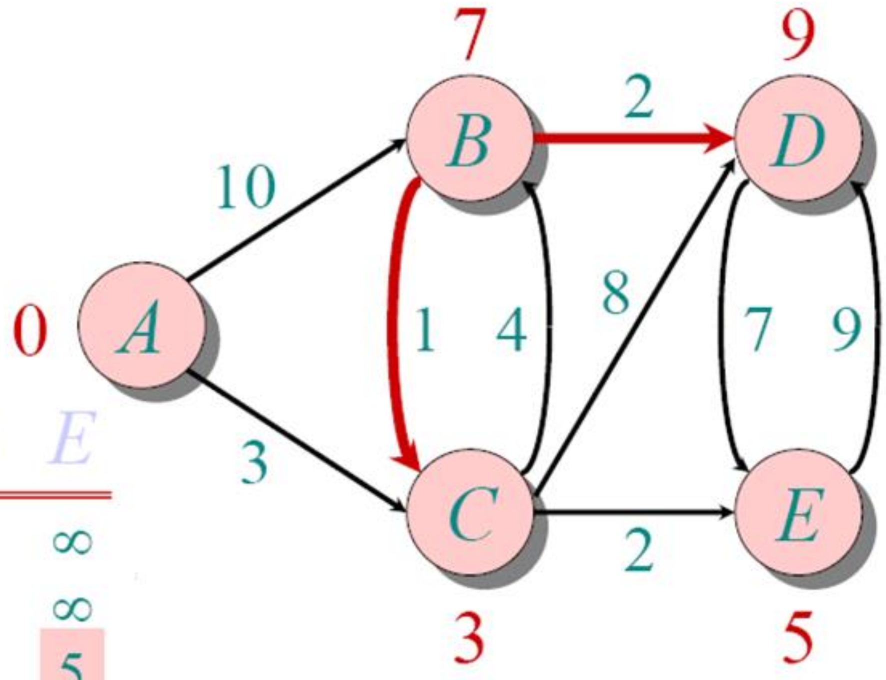
Dijkstra Example

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



S: { A, C, E, B }

Dijkstra Example

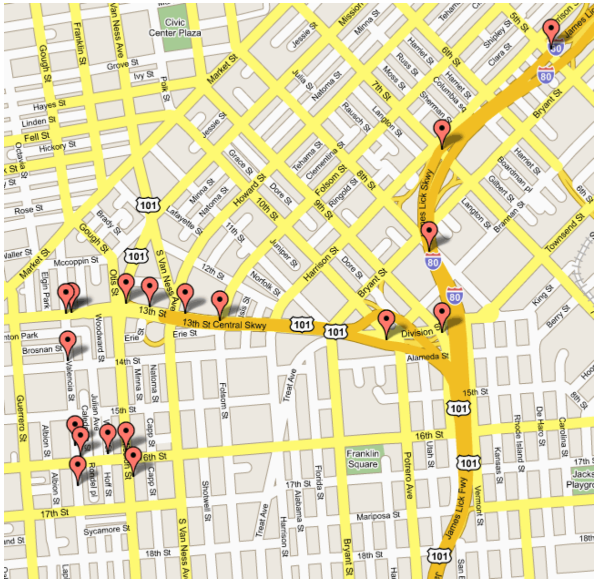


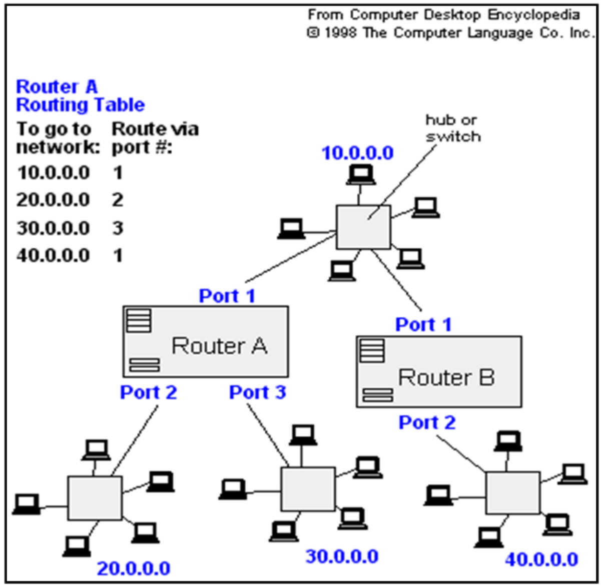
Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5
		7		11	
				9	

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

Applications of Dijkstra's Algorithm

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





From Computer Desktop Encyclopedia
© 1998 The Computer Language Co. Inc.

Router A Routing Table

To go to network:	Route via port #:
10.0.0.0	1
20.0.0.0	2
30.0.0.0	3
40.0.0.0	1

Thank you!