



# Graph II

## MST, Shortest Path

Adapted from Y. Huang's slides



# Graph Terminology

- Node (vertex)
- Edge (arc)
- Directed graph, undirected graph
- Degree, in-degree, out-degree
- Subgraph
- Simple path
- Cycle
- Directed acyclic graph
- Weighted graph

# Graph representation

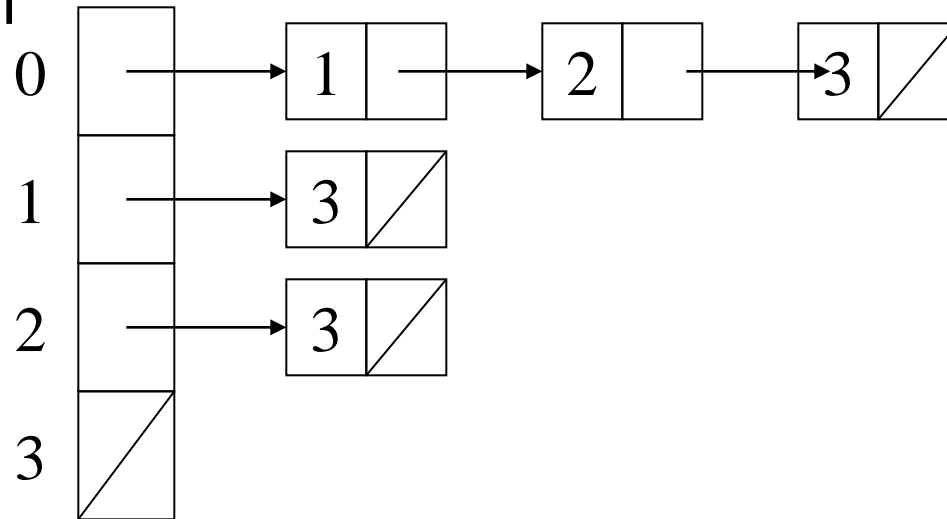
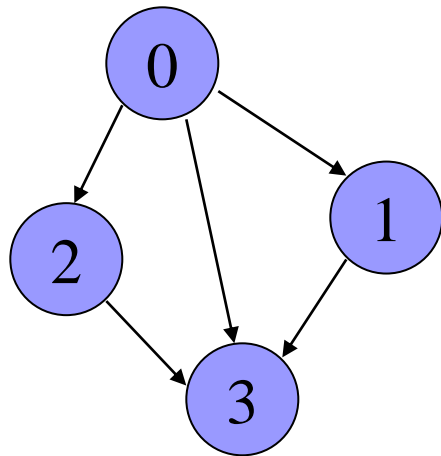
## Adjacency Matrix

- Assume  $N$  nodes in graph
- Use Matrix  $A[0 \dots N-1][0 \dots N-1]$ 
  - if vertex  $i$  and vertex  $j$  are adjacent in graph,  $A[i][j] = 1$ ,
  - otherwise  $A[i][j] = 0$
  - if vertex  $i$  has a loop,  $A[i][i] = 1$
  - if vertex  $i$  has no loop,  $A[i][i] = 0$

# Graph representation

## Adjacency List

- An array of list
- the  $i$ th element of the array is a list of vertices that connect to vertex  $i$



vertex 0 connect to vertex 1, 2 and 3

vertex 1 connects to 3

vertex 2 connects to 3



# Graph Traversal

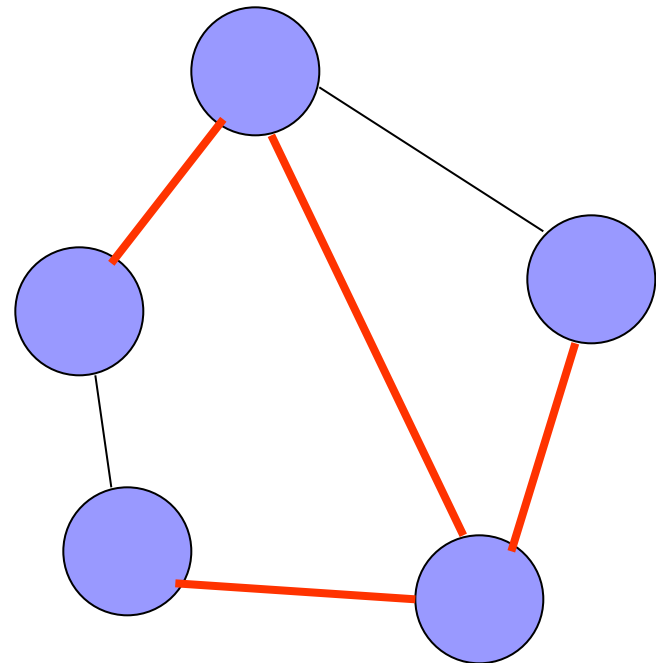
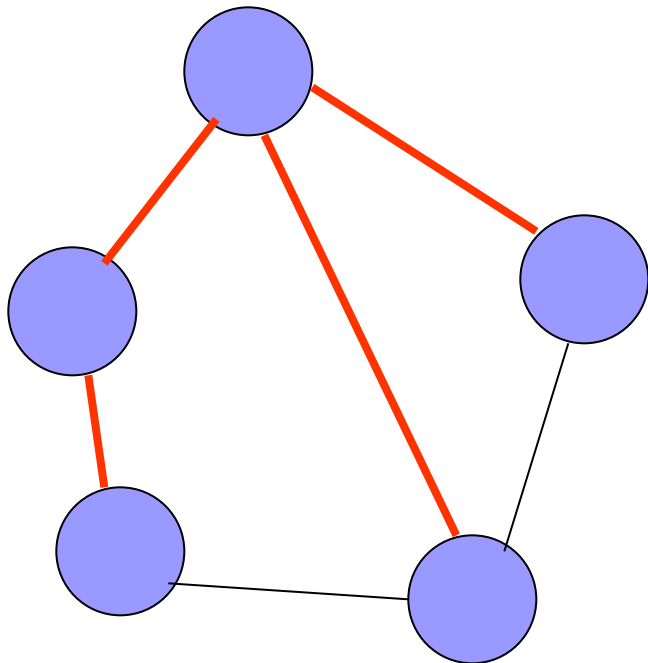
- From one vertex, list out all vertices that can be reached in graph  $G$
- Set of nodes to expand
- Each node has a flag to indicate visited or not
- Depth First Traversal
- Breadth First Traversal

# Spanning Tree

- Connected subgraph that includes all vertices of the original connected graph
- Subgraph is a tree
  - If original graph has  $n$  vertices, the spanning tree has  $n$  vertices and  $n-1$  edges.
  - No circle in this subgraph

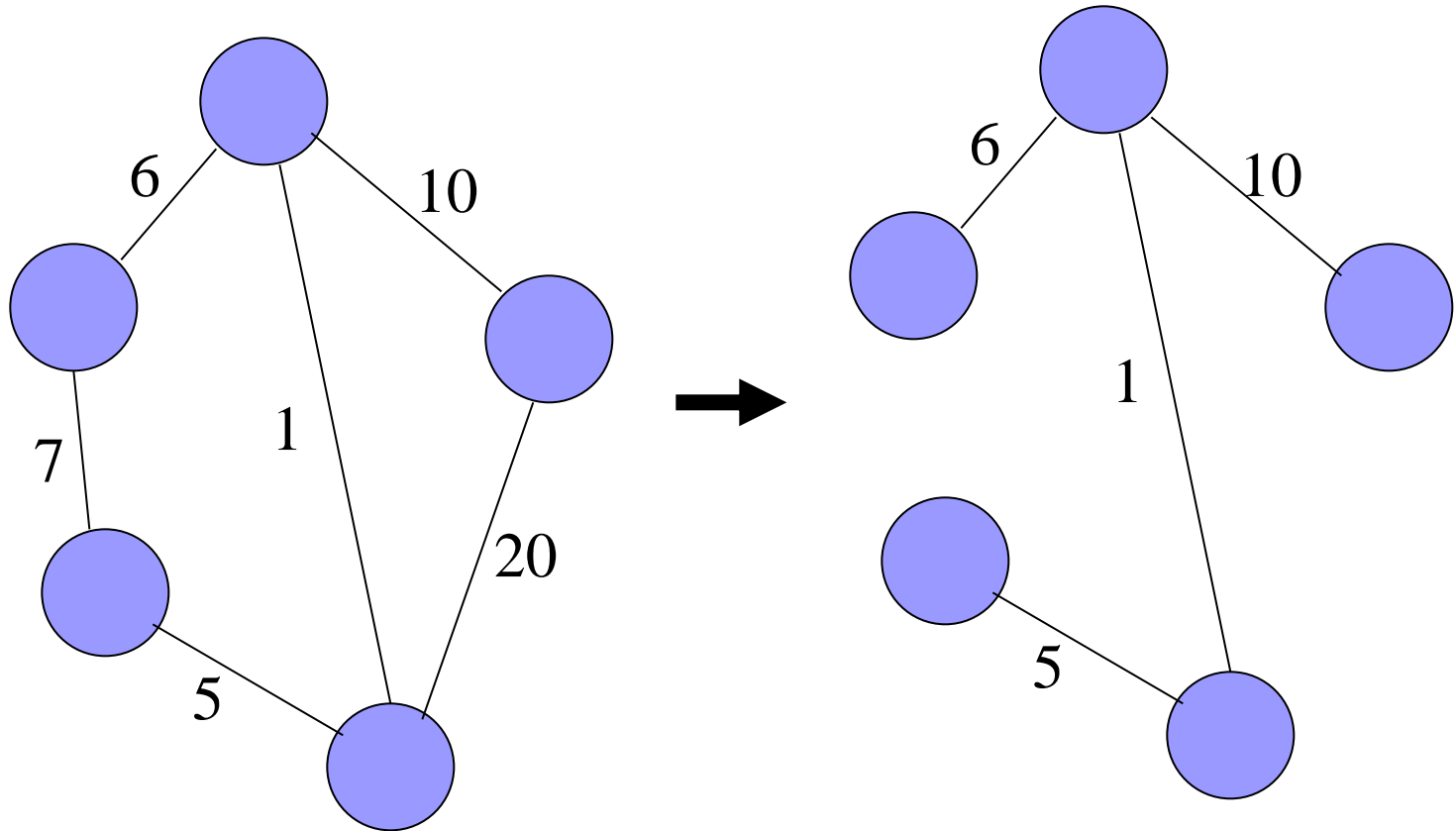
# Spanning Tree

- Minimum number of edges to keep it connected
- If  $N$  vertices, spanning tree has  $N-1$  edges



# Minimum Spanning Tree (MST)

Spanning tree with minimum weight



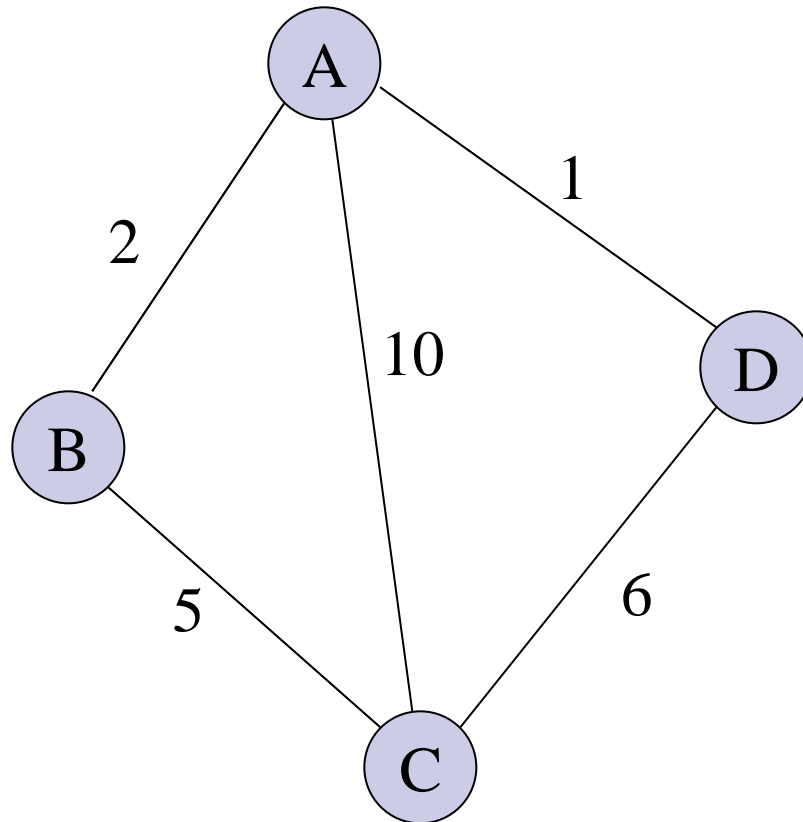


# Prim's Algorithm For Finding MST

1. All nodes are unselected, mark node  $v$  selected
2. For each node of graph,
  - {  
Find the edge with minimum weight that connects an unselected node with a selected node  
  
Mark this unselected node as selected  
}

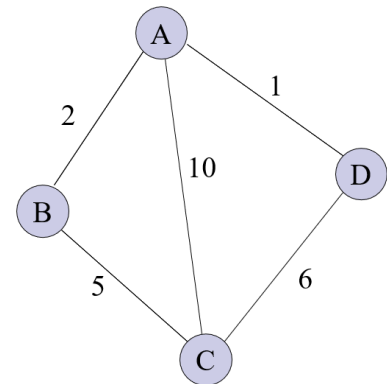
# Example

- Find the MST of the following graph



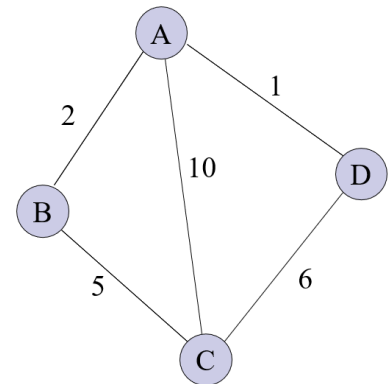
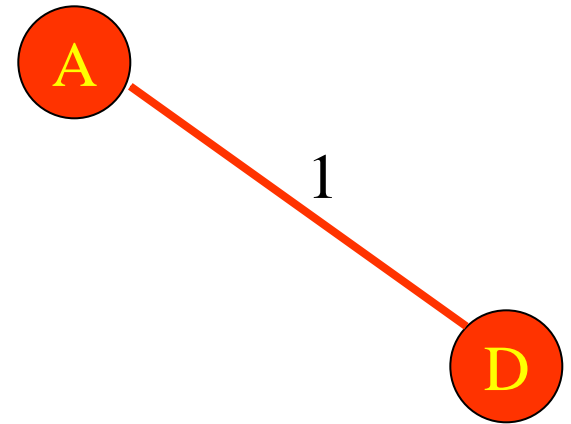
# Demos of Prim's Algorithm

- Step 1: mark vertex **A** as selected



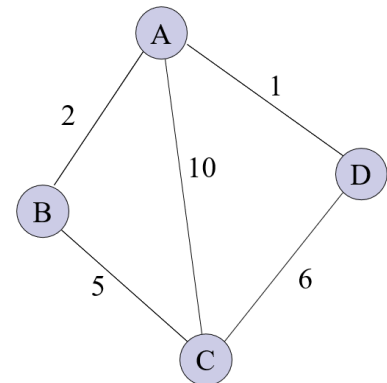
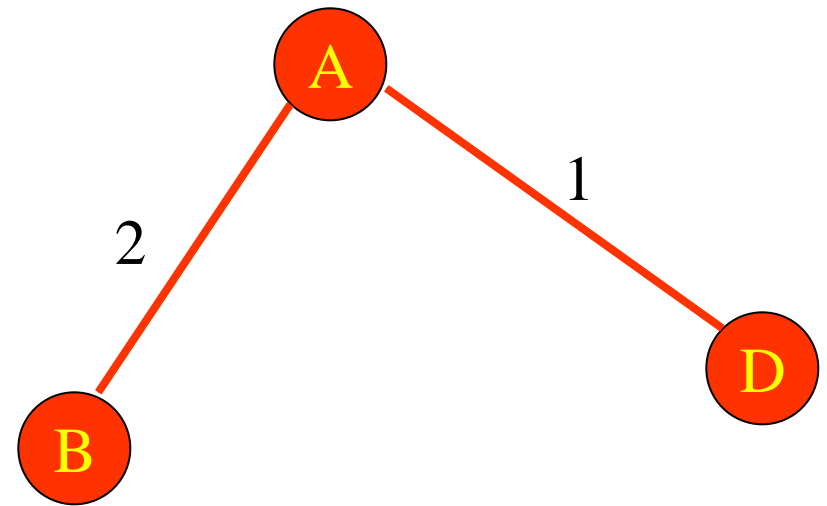
# Demos of Prim's Algorithm

- Step 2: find the minimum weighted edge connected to vertex A, and mark the other vertex on this edge as selected.



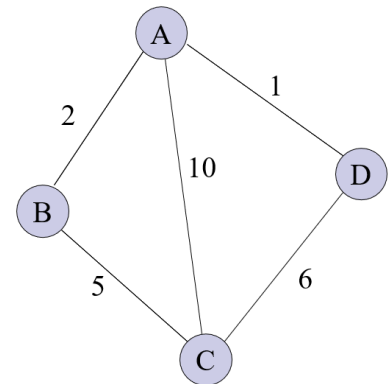
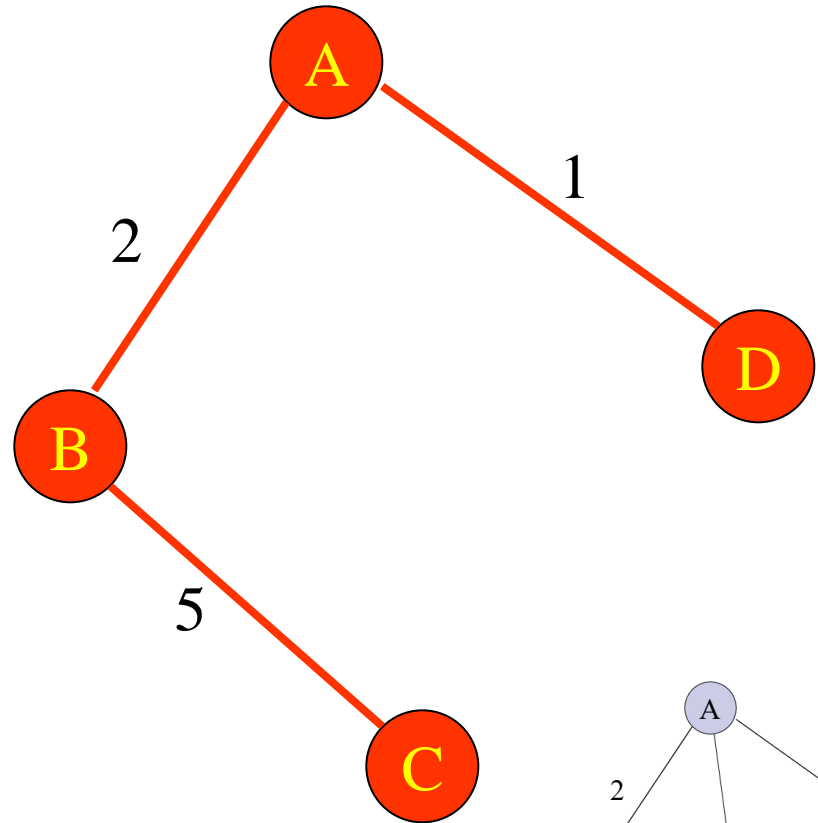
# Demos of Prim's Algorithm

- Step 3: find the minimum weighted edge connected to vertices set  $\{A, D\}$ , and mark the other vertex on this edge as selected.



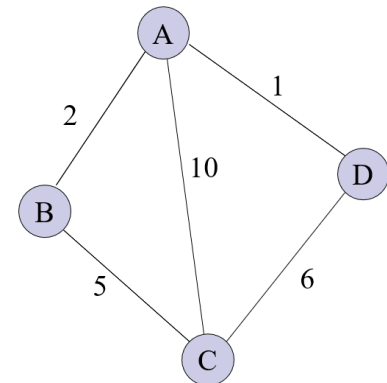
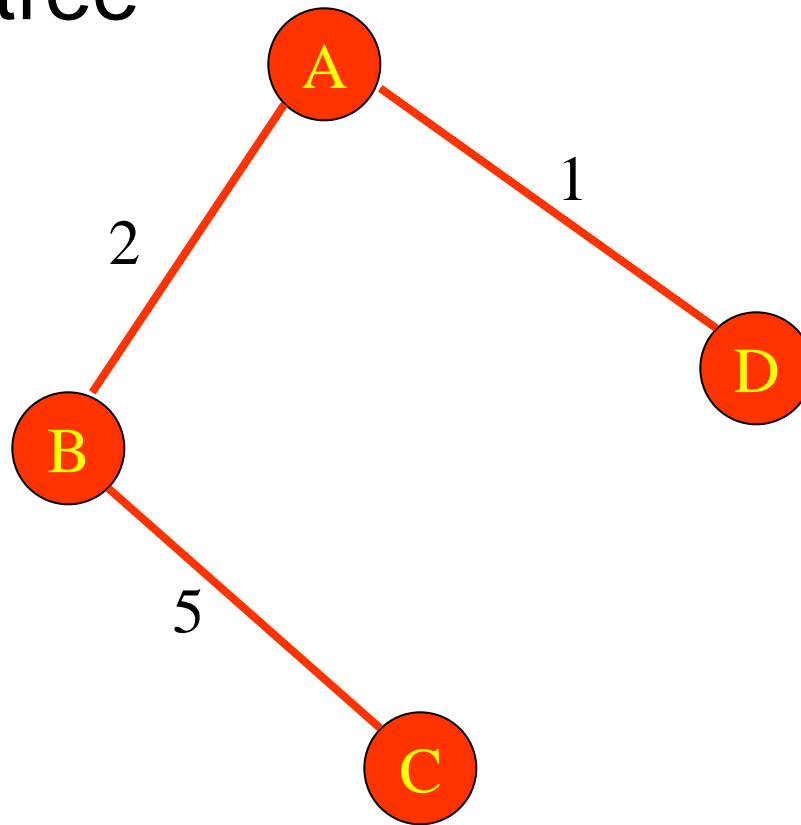
# Demos of Prim's Algorithm

- Step 4: find the minimum weighted edge connected to vertices set  $\{A, D, B\}$ , and mark the other vertex on this edge as selected.



# Demos of Prim's Algorithm

- Step 5: All vertex are marked as selected, So we find the minimum spanning tree



# Pseudo code for Prim's Alg.

- Minimum-Spanning-Tree-by-Prim( $G$ , weight-function, source)
  - **for each** vertex  $u$  **in** graph  $G$ 
    - **set** key **of**  $u$  **to**  $\infty$
    - **set** parent **of**  $u$  **to** *nil*
  - **set** key **of** source vertex **to** zero
  - **enqueue all vertices to**  $Q$
  - **while**  $Q$  is not *empty*
    - **extract** vertex  $u$  from  $Q$     *//  $u$  is the vertex with the lowest key that is in  $Q$*
    - **for each** adjacent vertex  $v$  **of**  $u$  **do**
      - **if** ( $v$  is *still* in  $Q$ ) and (weight-function( $u, v$ ) < key of  $v$ ) **then**
        - **set**  $u$  **to** be parent **of**  $v$     *// in minimum-spanning-tree*
        - update  $v$ 's key to equal weight-function( $u, v$ )

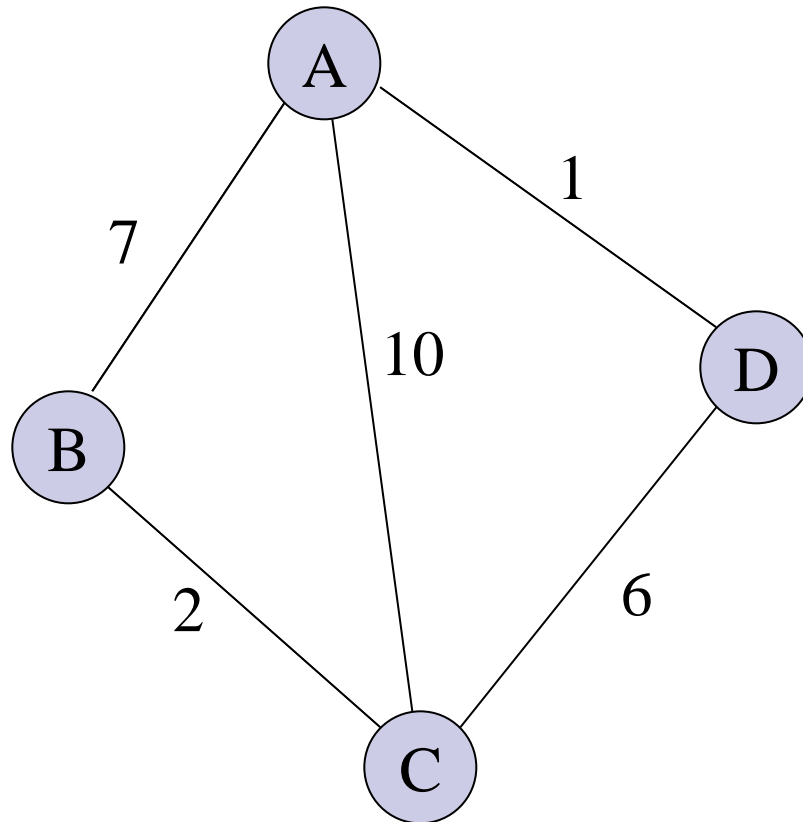


# Kruskal's Algorithm For Finding MST

1. All edges are unselected
2. Sort all edges and store them in set S
3. For each edge in S
  - {
    - If adding the edge to MST **does not form a circle**, add this edge to MST;
    - Delete this edge from S;
    - If  $|\text{edges in MST}| = |\text{nodes in graph}| - 1$ , exit;}

# Example

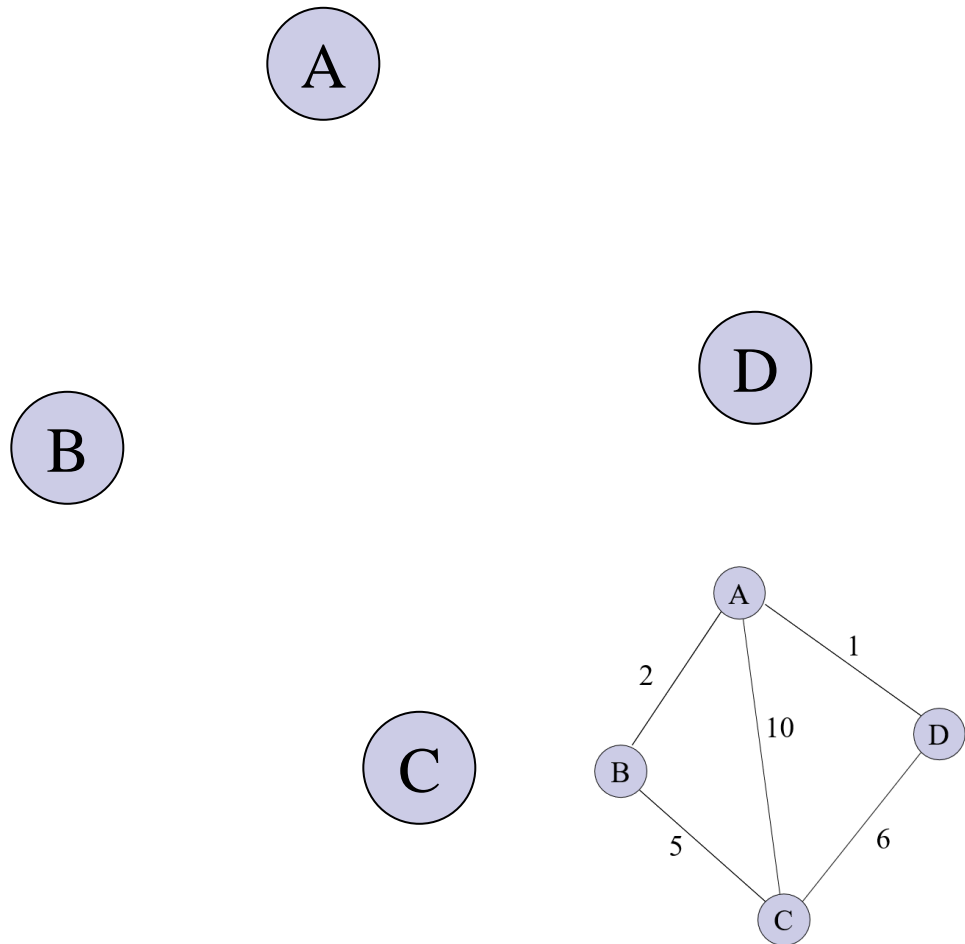
- Find the MST of the following graph



# Demos of Kruskal's Algorithm

- Step 1: Initialize S and empty MST

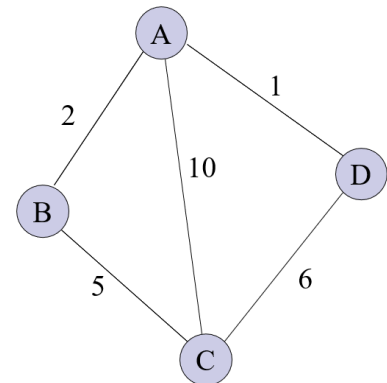
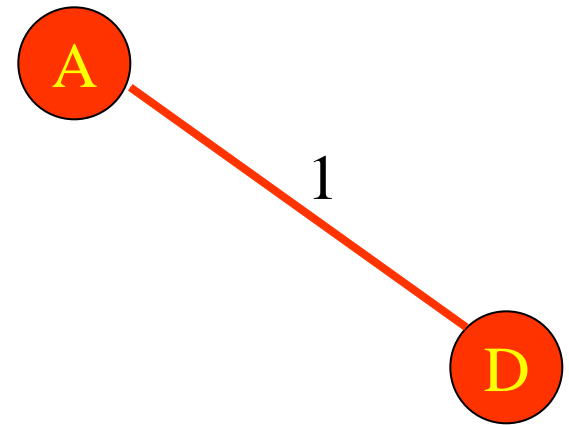
$S = \{(A,D)=1, (B,C)=2, (C,D)=6, (A,B)=7, (A,C)=10\}$



# Demos of Kruskal's Algorithm

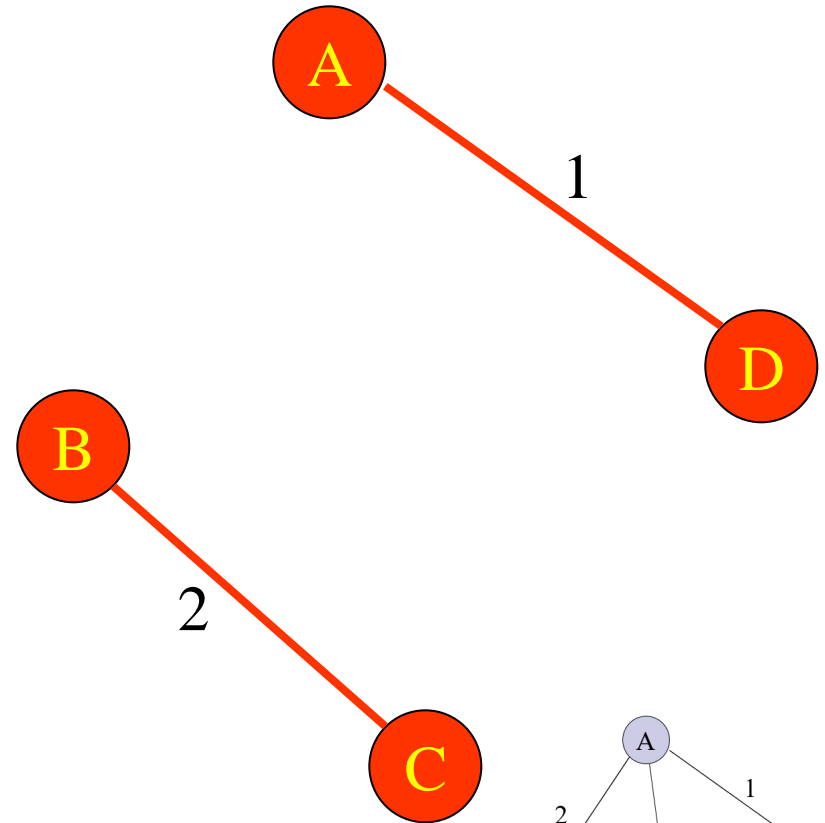
- Step 2: Add AD to MST, and delete it edge from S.

$S = \{(B,C)=2, (C,D)=6, (A,B)=7, (A,C)=10\}$

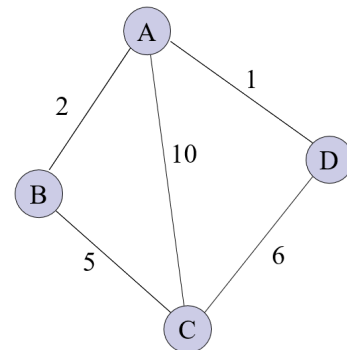


# Demos of Kruskal's Algorithm

- Step 3: Add BC to MST, and delete it from S.



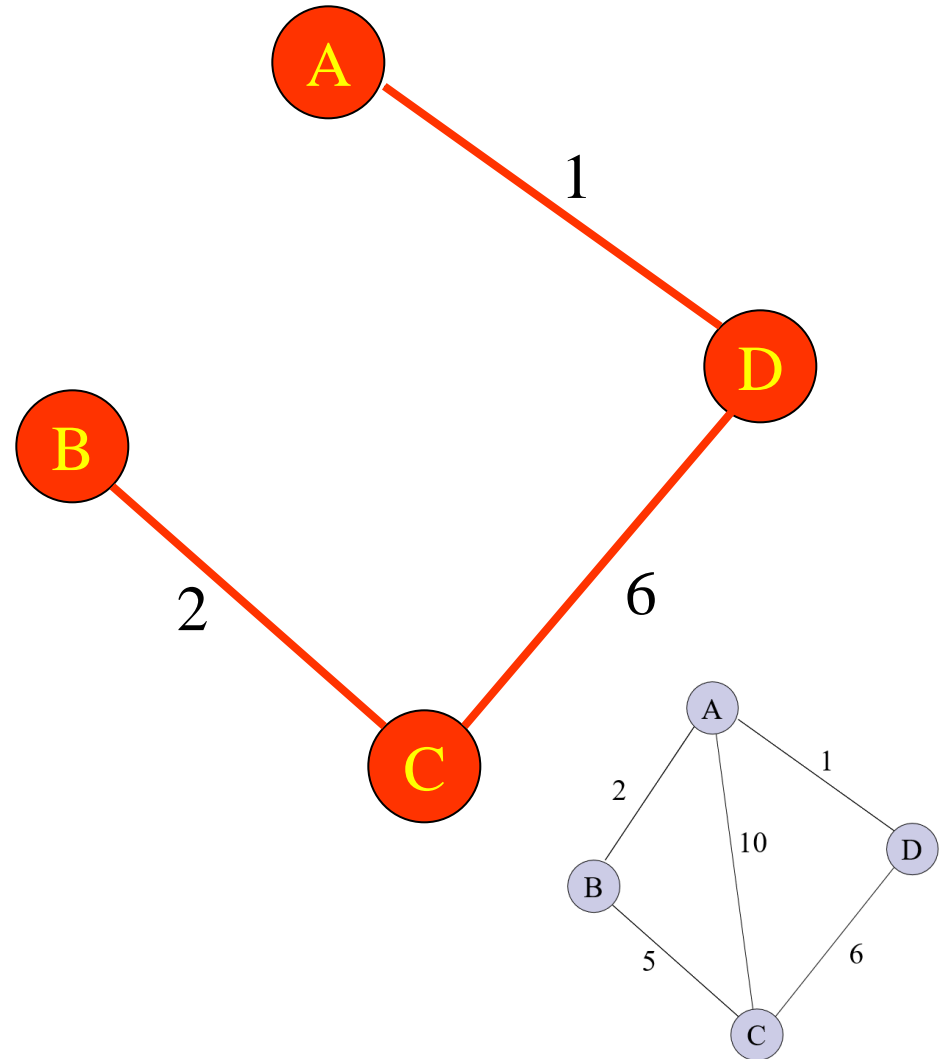
$S = \{(C,D)=6, (A,B)=7, (A,C)=10\}$



# Demos of Kruskal's Algorithm

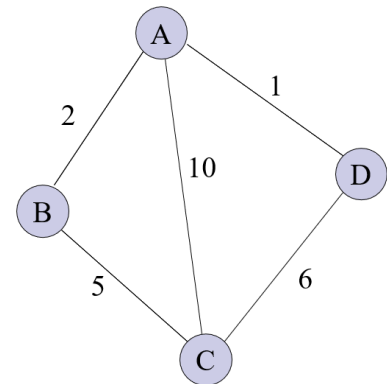
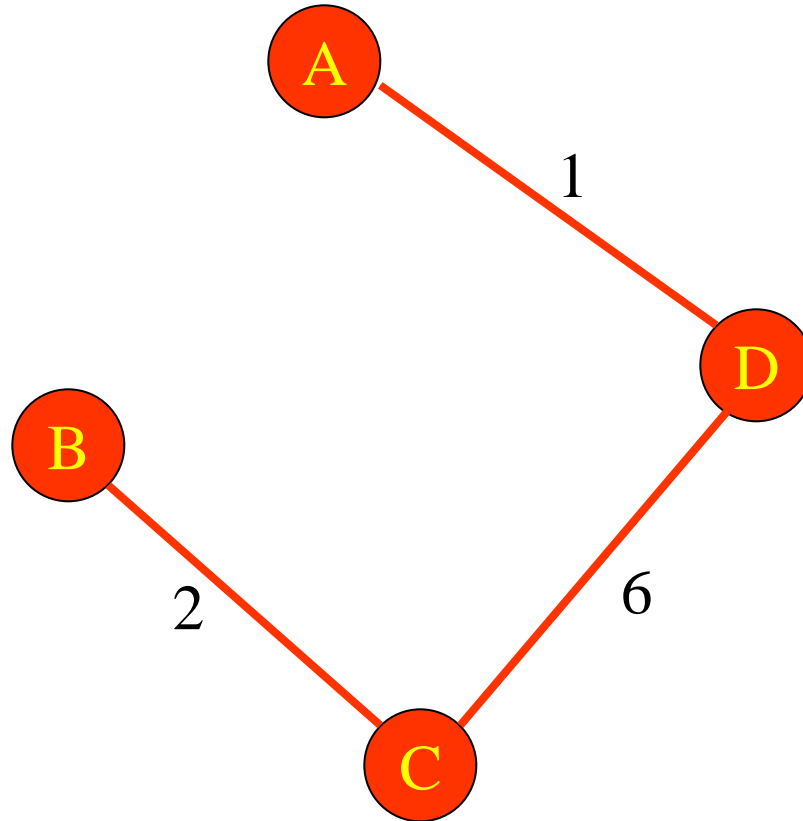
- Step 4: Add CD to MST, and delete it from S

$S = \{(A,B)=7, (A,C)=10\}$

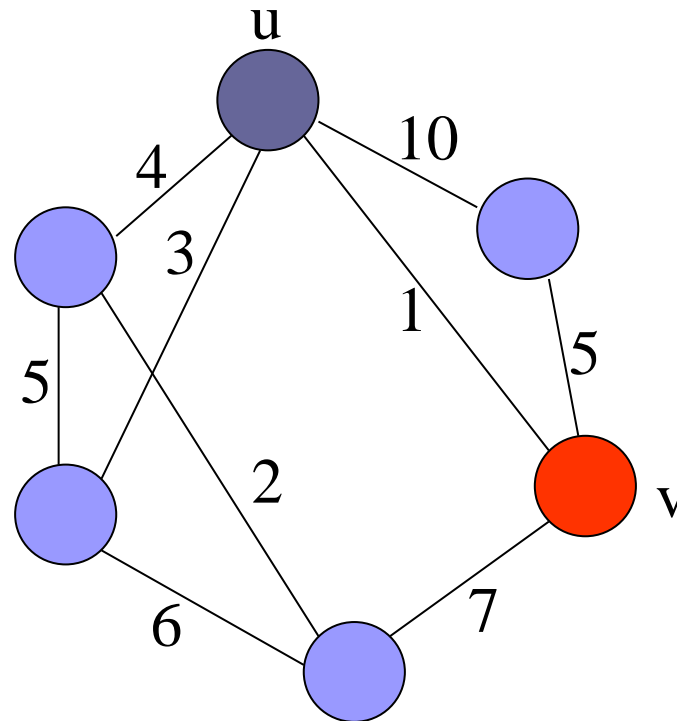


# Demos of Kruskal's Algorithm

- Step 5: Satisfy the exiting condition, So we find the minimum spanning tree



# Shortest Path Problem



- Weight: cost, distance, travel time, hop ...



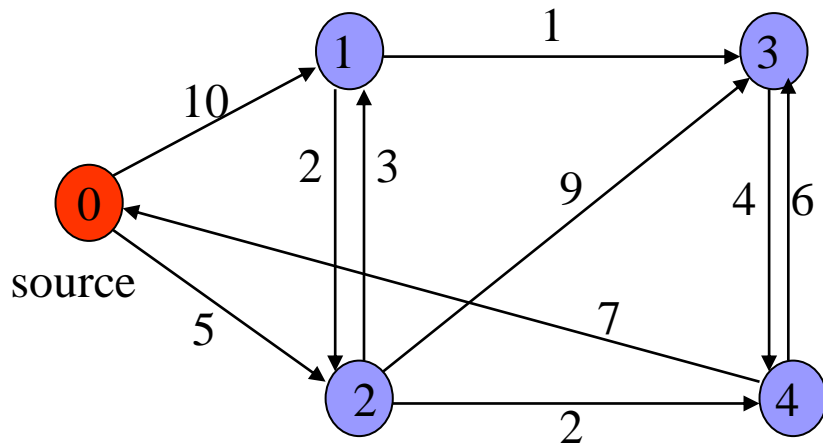


# Single Source Shortest Path Problem

- Single source shortest path problem
  - Find the shortest path to all other nodes
- Dijkstra's shortest path algorithm
  - Find shortest path greedily by updating distance to all other nodes
  - Not applicable for negative weights

# Example – Dijkstra Algorithm

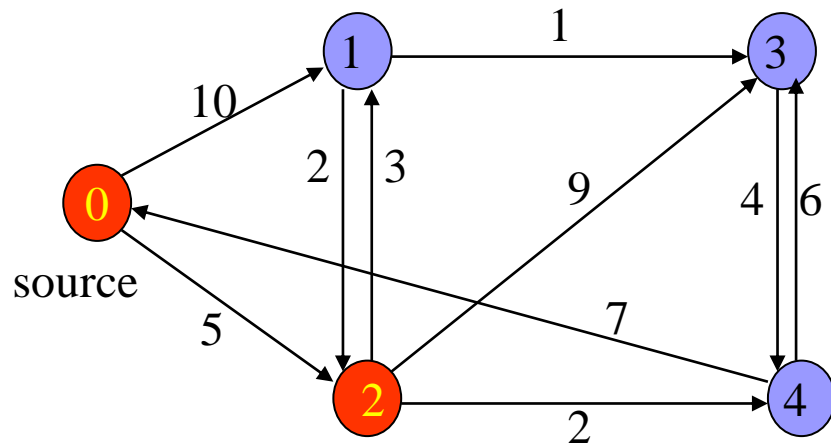
- Greedy Algorithm
- Assume all weight of edge  $>0$



node	from node $V_0$ to other nodes			
$V_1$	10 ( $V_0$ )			
$V_2$	5 ( $V_0$ )			
$V_3$	$\infty$ ( $V_0$ )			
$V_4$	$\infty$ ( $V_0$ )			
best				

# Example – Dijkstra Algorithm

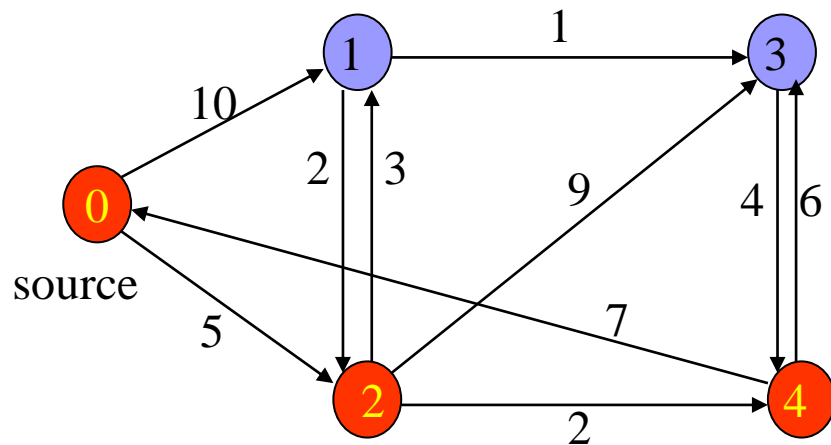
- step 1: find the shortest path to node 0
  - node 2 is selected



node	from node $V_0$ to other nodes			
$V_1$	10 ( $V_0$ )			
$V_2$	5 ( $V_0$ )			
$V_3$	$\infty$ ( $V_0$ )			
$V_4$	$\infty$ ( $V_0$ )			
best	$V_2$			

# Example – Dijkstra Algorithm

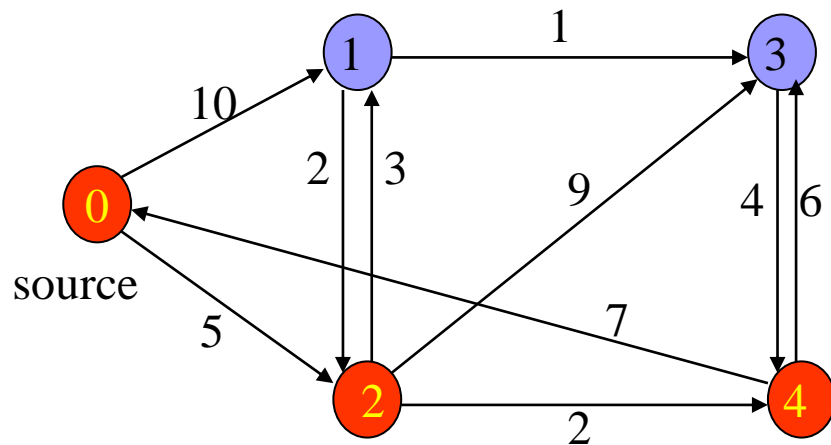
- step 2: recalculate the path to all other nodes
  - find the shortest path to node 0. Node 4 is selected



node	from node $V_0$ to other nodes			
$V_1$	10 ( $V_0$ )	8 ( $V_2$ )		
$V_2$	5 ( $V_0$ )			
$V_3$	$\infty$ ( $V_0$ )	14 ( $V_2$ )		
$V_4$	$\infty$ ( $V_0$ )	7 ( $V_2$ )		
best	$V_2$			

# Example – Dijkstra Algorithm

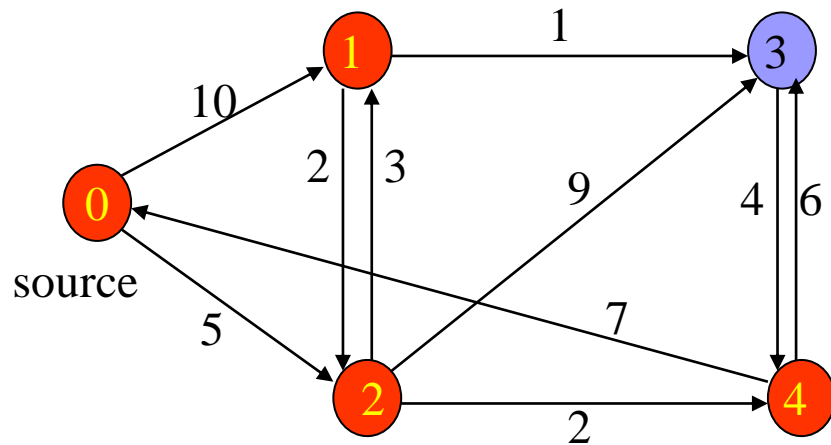
- step 2: recalculate the path to all other nodes
  - find the shortest path to node 0. Node 4 is selected



node	from node $V_0$ to other nodes			
$V_1$	10 ( $V_0$ )	8 ( $V_2$ )		
$V_2$	5 ( $V_0$ )			
$V_3$	$\infty$ ( $V_0$ )	14 ( $V_2$ )		
$V_4$	$\infty$ ( $V_0$ )	7 ( $V_2$ )		
best	$V_2$	$V_4$		

# Example – Dijkstra Algorithm

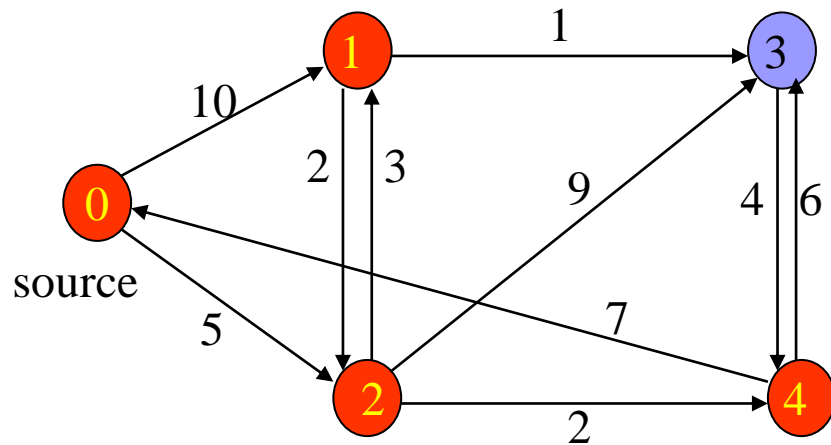
- step 3: recalculate the path to all other nodes
  - find the shortest path to node 0. node 1 is selected



node	from node $V_0$ to other nodes			
$V_1$	10 ( $V_0$ )	8 ( $V_2$ )	8 ( $V_2$ )	
$V_2$	5 ( $V_0$ )			
$V_3$	$\infty$ ( $V_0$ )	14 ( $V_2$ )	13 ( $V_4$ )	
$V_4$	$\infty$ ( $V_0$ )	7 ( $V_2$ )		
best	$V_2$	$V_4$		

# Example – Dijkstra Algorithm

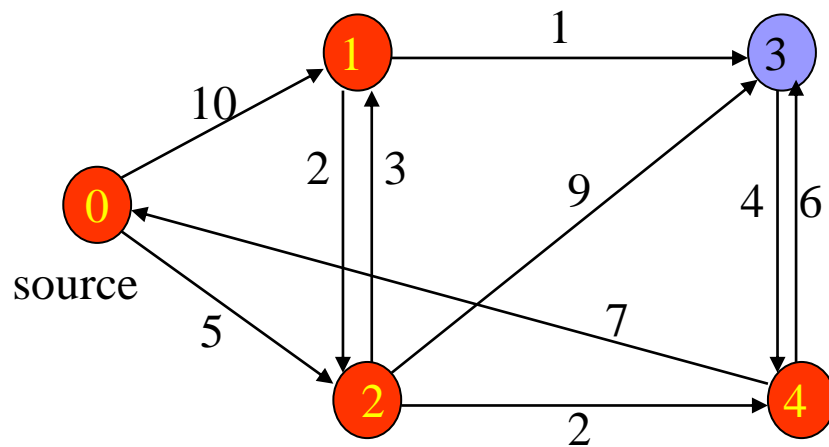
- step 3: recalculate the path to all other nodes
  - find the shortest path to node 0. node 1 is selected



node	from node $V_0$ to other nodes			
$V_1$	10 ( $V_0$ )	8 ( $V_2$ )	8 ( $V_2$ )	
$V_2$	5 ( $V_0$ )			
$V_3$	$\infty$ ( $V_0$ )	14 ( $V_2$ )	13 ( $V_4$ )	
$V_4$	$\infty$ ( $V_0$ )	7 ( $V_2$ )		
best	$V_2$	$V_4$	$V_1$	

# Example – Dijkstra Algorithm

- step 3: recalculate the path to all other nodes
  - find the shortest path to node 0. node 1 is selected

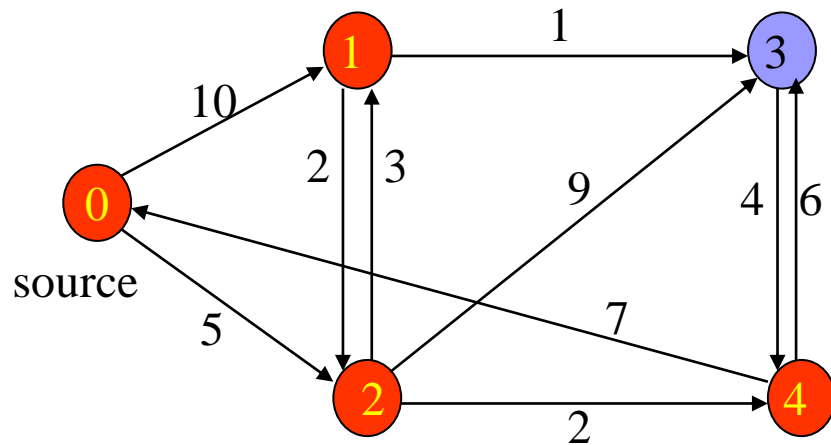


node	from node $V_0$ to other nodes			
$V_1$	10 ( $V_0$ )	8 ( $V_2$ )	8 ( $V_2$ )	
$V_2$	5 ( $V_0$ )			
$V_3$	$\infty$ ( $V_0$ )	14 ( $V_2$ )	13 ( $V_4$ )	9 ( $V_1$ )
$V_4$	$\infty$ ( $V_0$ )	7 ( $V_2$ )		
best	$V_2$	$V_4$	$V_1$	



# Example – Dijkstra Algorithm

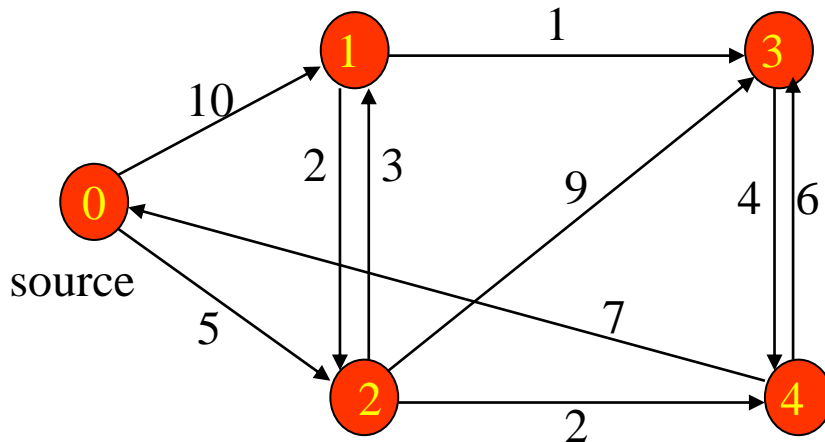
- step 3: recalculate the path to all other nodes
  - find the shortest path to node 0. node 1 is selected



node	from node $V_0$ to other nodes			
$V_1$	10 ( $V_0$ )	8 ( $V_2$ )	8 ( $V_2$ )	
$V_2$	5 ( $V_0$ )			
$V_3$	$\infty$ ( $V_0$ )	14 ( $V_2$ )	13 ( $V_4$ )	9 ( $V_1$ )
$V_4$	$\infty$ ( $V_0$ )	7 ( $V_2$ )		
best	$V_2$	$V_4$	$V_1$	$V_3$

# Example – Dijkstra Algorithm

- Now we get all shortest paths to each node



node	from node $V_0$ to other nodes			
$V_1$	10 ( $V_0$ )	8 ( $V_2$ )	8 ( $V_2$ )	
$V_2$	5 ( $V_0$ )			
$V_3$	$\infty$ ( $V_0$ )	14 ( $V_2$ )	13 ( $V_4$ )	9 ( $V_1$ )
$V_4$	$\infty$ ( $V_0$ )	7 ( $V_2$ )		
best	$V_2$	$V_4$	$V_1$	$V_3$

# Dijkstra Algorithm

Mark source node selected

Initialize all distances to Infinite, source node distance to 0.

Make source node the current node.

While (there is unselected node)

{

    Expand on current node

    Update distance for neighbors of current node

    Find an unselected node with smallest distance,  
    and make it current node and mark this node  
    selected

}

# Pseudo-code For Dijkstra's Algorithm

```
function Dijkstra( $G, w, s$ )  
  for each vertex  $v$  in  $V[G]$  // Initializations  
     $d[v] := \text{infinity}$   
     $\text{previous}[v] := \text{undefined}$   
   $d[s] := 0$   
   $S := \text{empty set}$   
   $Q := V[G]$   
  while  $Q$  is not an empty set // The algorithm itself  
     $u := \text{Extract\_Min}(Q)$   
     $S := S \text{ union } \{u\}$   
    for each edge  $(u,v)$  outgoing from  $u$   
      if  $d[u] + w(u,v) < d[v]$  // Relax  $(u,v)$   
         $d[v] := d[u] + w(u,v)$   
         $\text{previous}[v] := u$ 
```