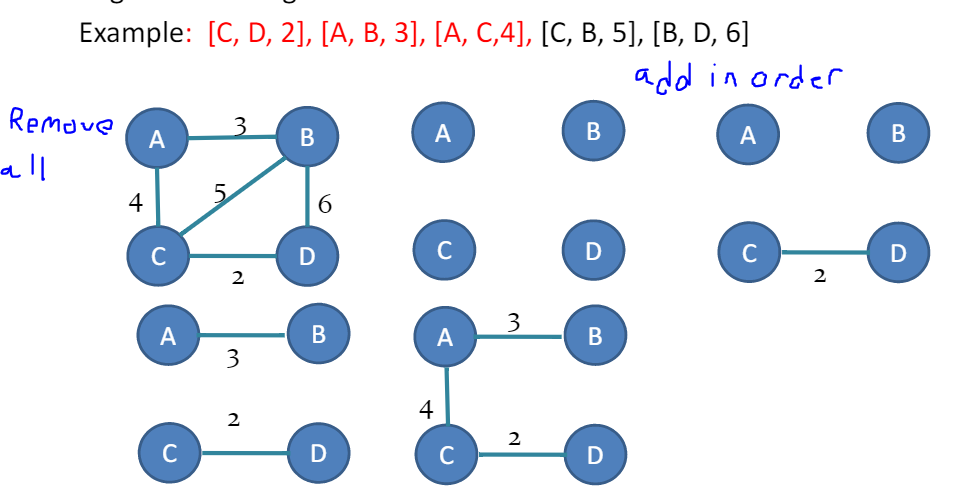
**Minimum Spanning Tree**

* A minimum spanning tree (MST) of a weight connected undirected graph is defined as a spanning tree of minimum weight over all spanning trees of the graph.
* There are many spanning trees.
* For a non-weighted graph, any spanning tree is an MST

**Kruskal’s Algorithm**

The idea: sorting the edges of G in increasing order, start with a forest with no edges and add edges in order to the forest as long as the edge connects two forests



1. Create a forest in such a way that each graph is a separate tree
2. Create a priority queue Q (min heap) that contains all the edges of the graph
3. Repeat steps 4 and 5 while Q is NOT EMPTY
4. Remove an edge from Q
5. If the edge obtained in step 4 connects two different trees, then add it to the forest.
6. Stop

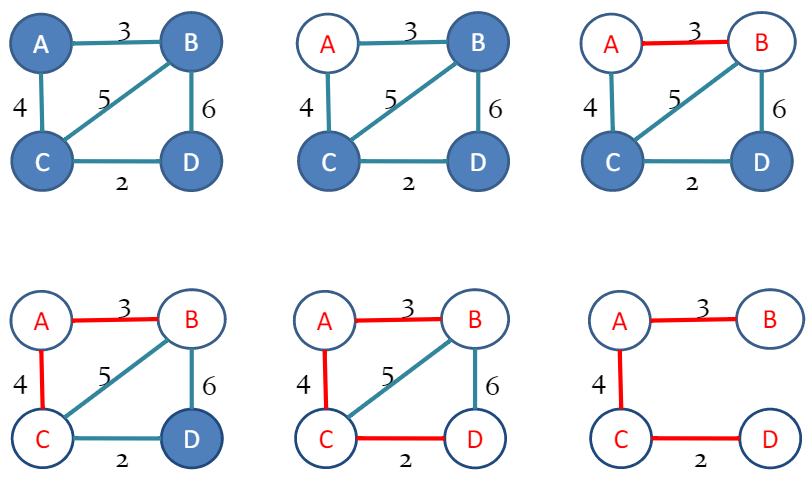
Time: O(m log m) Space: O(m)

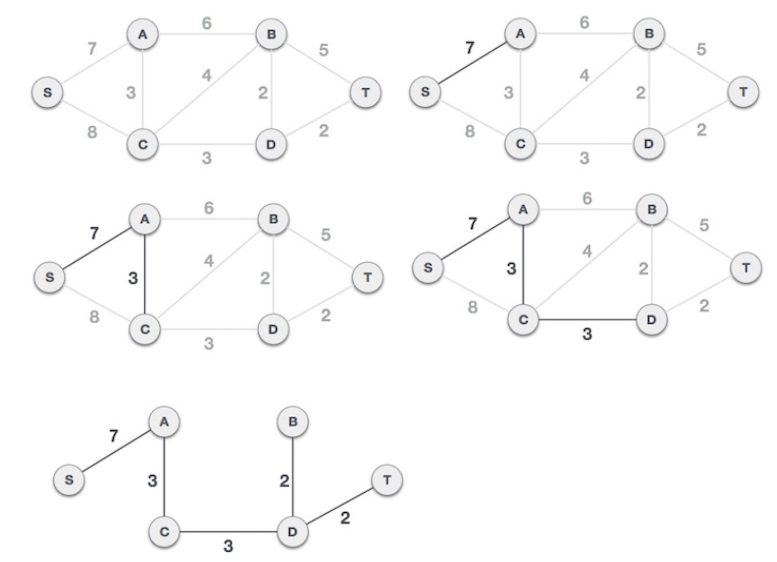
**Prim’s algorithm**

Idea: grow a subtree by adding outgoing edge of minimum weight

1. Start a current tree T by adding a node of G
2. If T is not a spanning tree of G, add the minimum weighted edge from T to nodes not in T
3. Repeat step 2 until the current tree T is a spanning tree of G
4. Output the current tree T

Time: O(nm), Space: O(n)



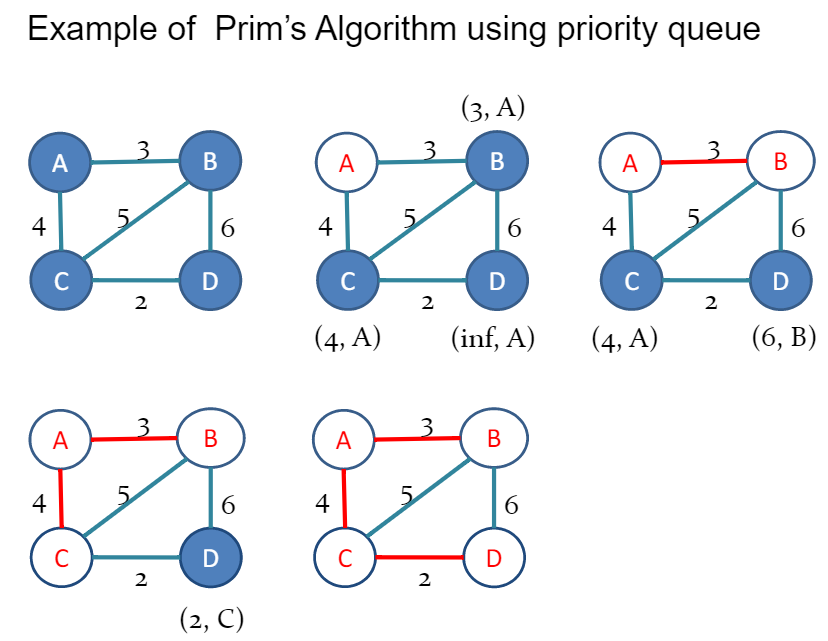


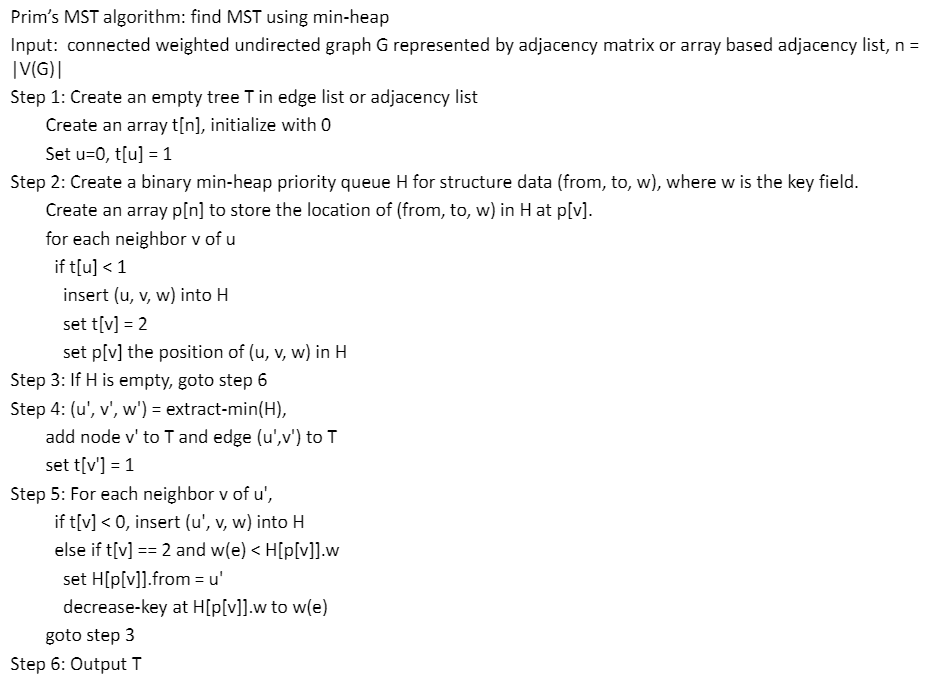
**Improving Prim’s Algorithm using a min-heap priority queue**

Main operation: to find the minimum weighted edge from current tree to nodes not in current tree

Solution to main operation:

1. For each node I, use a key to represent the minimum weight from the node to the current tree, and a parent variable to represent the other end of the minimum weight edge
2. Use min-heap to store the nodes in the current tree
3. Use extract-min to find the node of the minimum weight edge
4. Use decrease-key to update the key of neighbor nodes





Time: O(n log n) for step 4, O(m log n) for step 5

Overall time complexity of Prim’s algorithm with the heap structures is O((m+n) log n)

Space: the auxiliary space complexity is O(n) for arrays t[] and p[], graph T, and binary min-heap H. We see that the Prim’s algorithm with the efficient data structures has better time performance.

**Complexity of Prim’s algorithm**

Using binary min-heap and adjacent list:

Time: O((m+n) log n)

Space: O(n)

Using Fibonacci min-heap:

Time: O(m+n log n)

Space: O(n)

**Greedy algorithms**

* Prim’s and Kruskal’s algorithms are examples of so called greedy algorithms.
  + Greedy strategy: at each step, choose an allowed minimum weighted edge
* For MST problem, the greedy algorithms solve the problem
  + Theorem: Prim’s and Kruskal’s algorithms each derive an MST