# Graph Data Structures
# Minimum Spanning Tree

PROF. NAVRATI SAXENA

# Problems Specific with Graphs

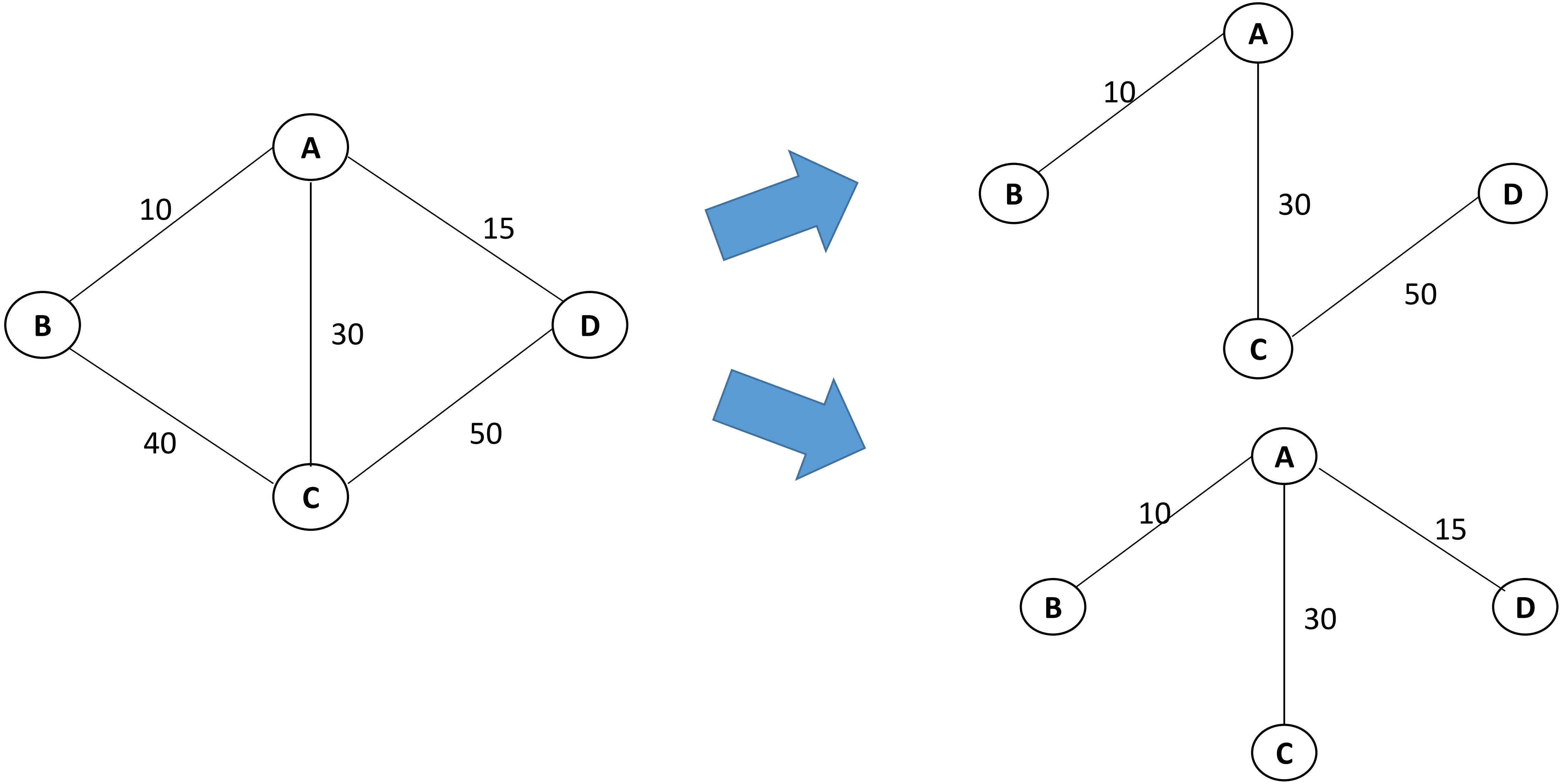
## Minimum Spanning Tree

- Path Problems
  1. Simple Paths
  2. **Shortest Path Problem**
     - Single source shortest paths
     - All-pair shortest paths
  3. Find Cycles
  4. Euler Path and Circuit Problem
  5. Hamiltonian Path and Circuit Problem (or TSP)

# Spanning Tree

- For a weighted undirected graph, Spanning Tree is a sub-graph that connects all the vertices together using the minimum number of edges required
- The graph should be connected: There should be no cycles
- For ***n*** number of *vertices*, *(n-1)* edges are needed
- Hence for four vertices, we need three edges without any cycles

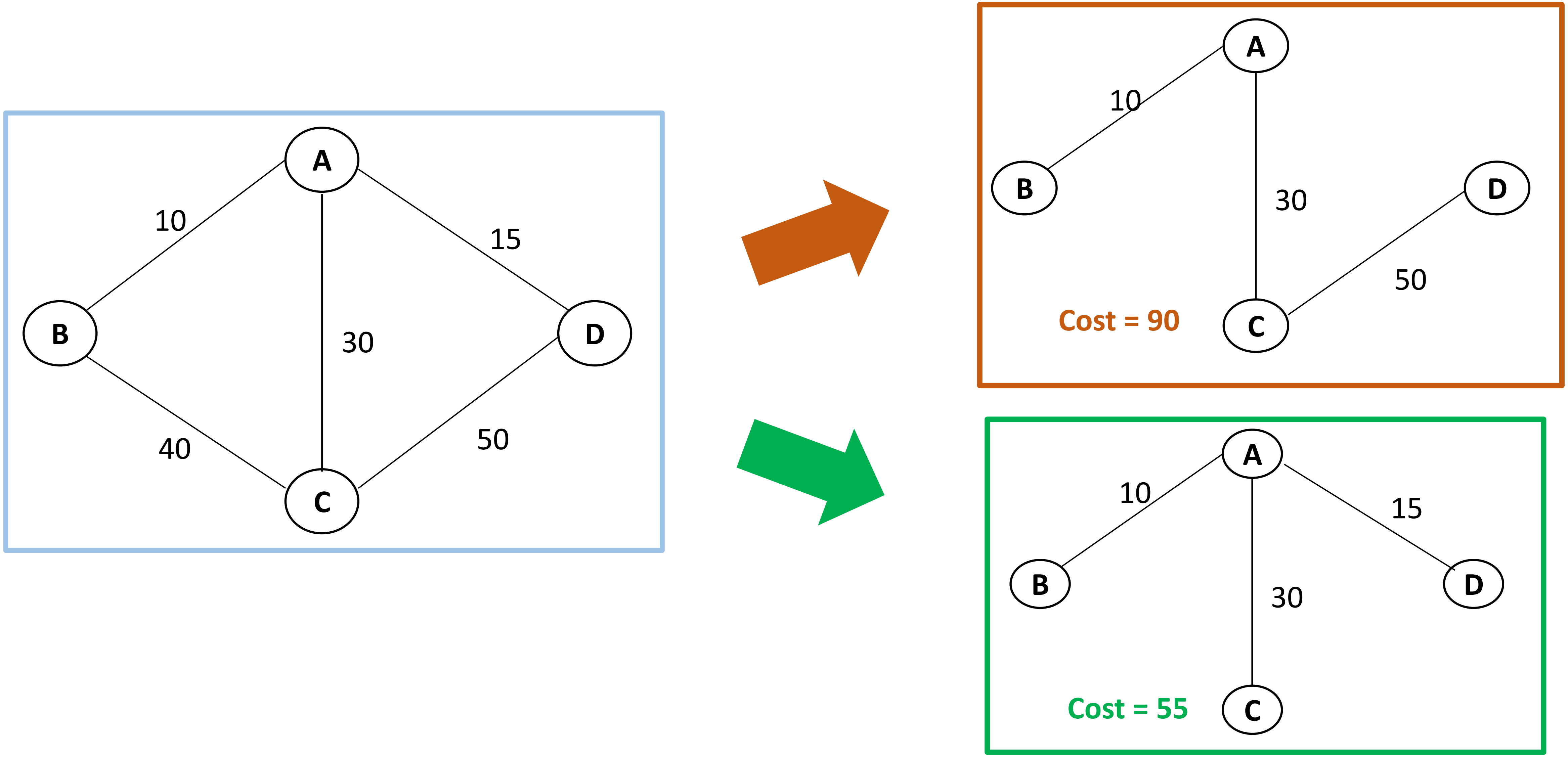
# Many Spanning Trees for a Single Graph



# Minimum Spanning Tree (MST)

- **Cost of any spanning tree**: Add weights of all the edges
- Many spanning tress for a single graph
- Calculate cost of all spanning trees and pick the minimum cost spanning tree
  - A tree with minimum weights is MST
- **Time complexity**: ***Exponential***
- Two Algorithms
  - Prim's Algorithm
  - Kruskal's Algorithm

# Minimum Spanning Trees



# Prim's Algorithm: At a Glance

- Maintains two sets
- **MST Set**: vertices that are included in the spanning tree
- **Set 2**: vertices that are not yet included in the spanning tree

# Prim’s MST Algorithm

- **Key Idea**: Find the local optimum in the hopes of finding a global optimum.
- Start from one vertex and keep adding edges with the lowest weight until all vertices are reached.

<u>**Steps**</u>:

1. Initialize the minimum spanning tree with a vertex chosen at random.
2. Find all the edges that connect the tree to new vertices, select the minimum and add it to the tree
3. Keep repeating step 2 until we get a minimum spanning tree

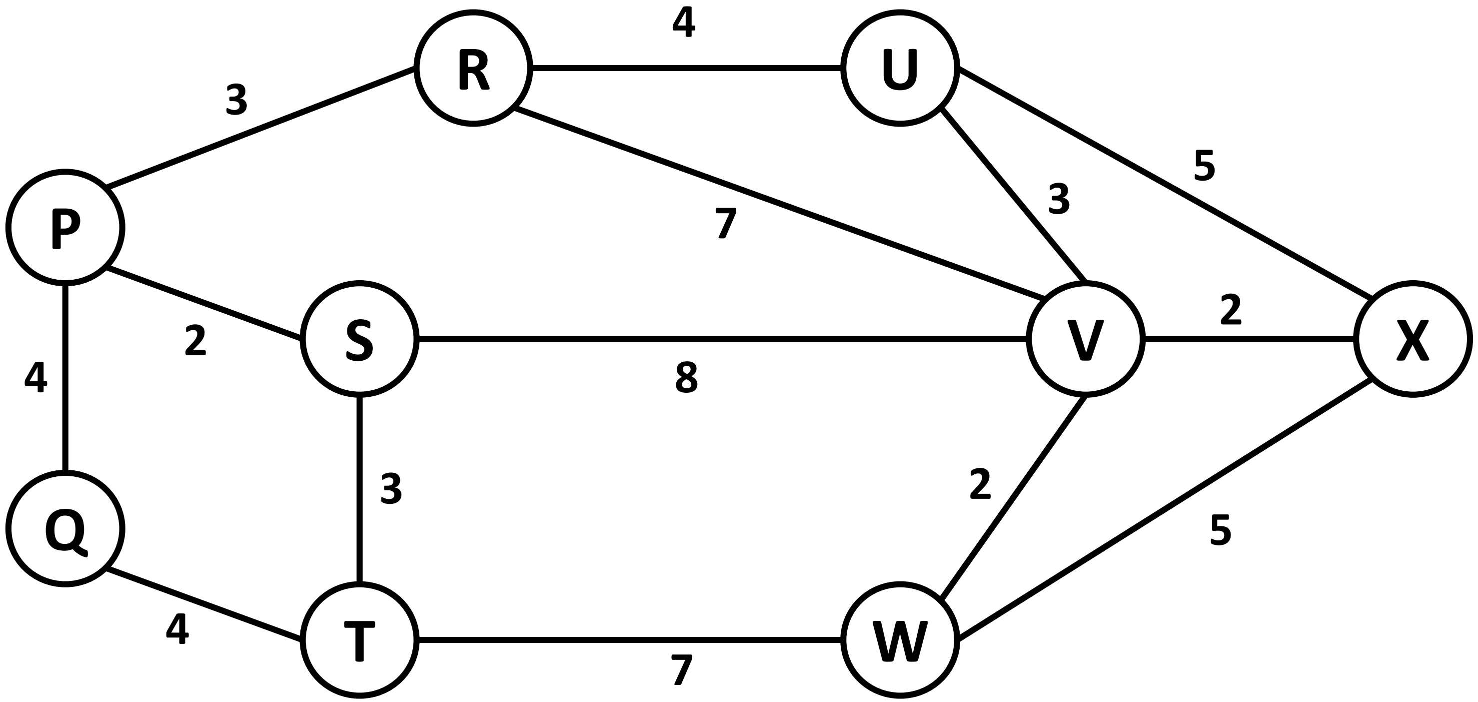
1. Create an array of size ***V*** and initialize it with NIL
2. Create a Priority Queue (min Heap) of size ***V***. Let the **min Heap be *Q***
3. Insert all vertices into **Q.** Assign **cost** of starting vertex to **0** and the **cost** of other vertices to infinite.

# Prim's Algorithm: Pseudo-code

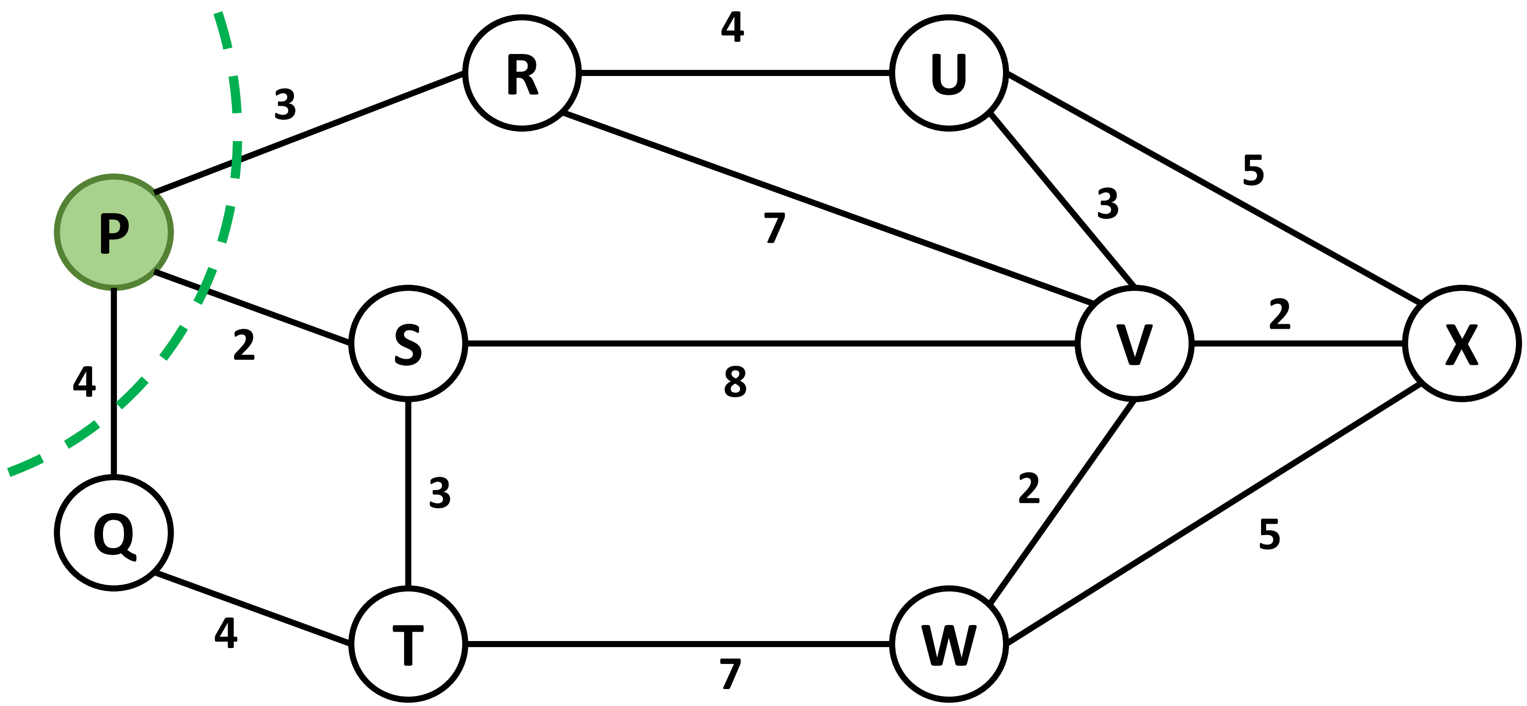
```
MST-PRIM(G, w, source)

for each u Є V [G] do // for all vertices
    cost[u] ← ∞
    π[u] ← NULL
cost[source] ← 0
Q ← V [G] // set 2
while Q ≠ Ø do
    u ← EXTRACT-MIN(Q)
    for each v Є Adj[u] do
        if v Є Q and w(u, v) < cost[v]
            π[v] ← u
            cost[v] ← w(u, v)
```

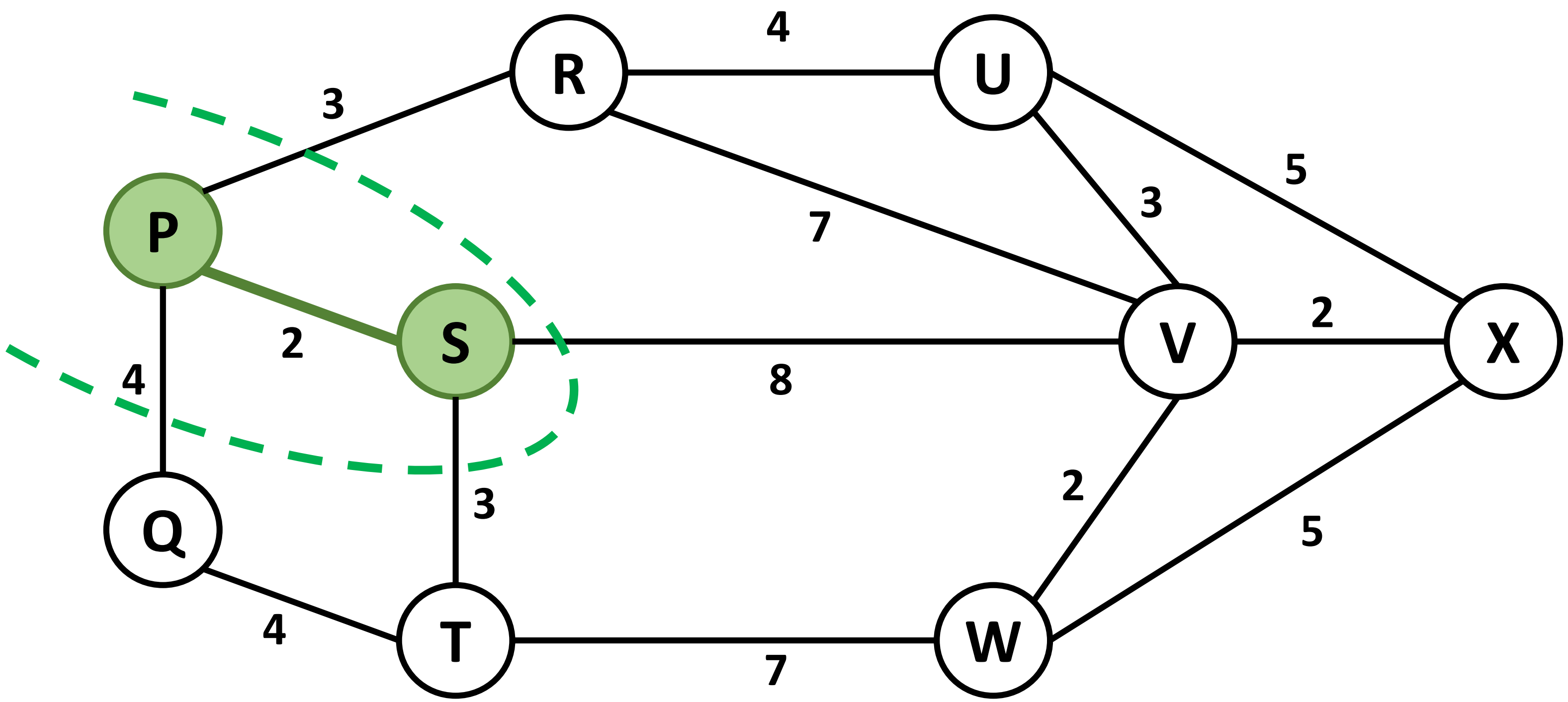
# Prim's Algorithm: An Example



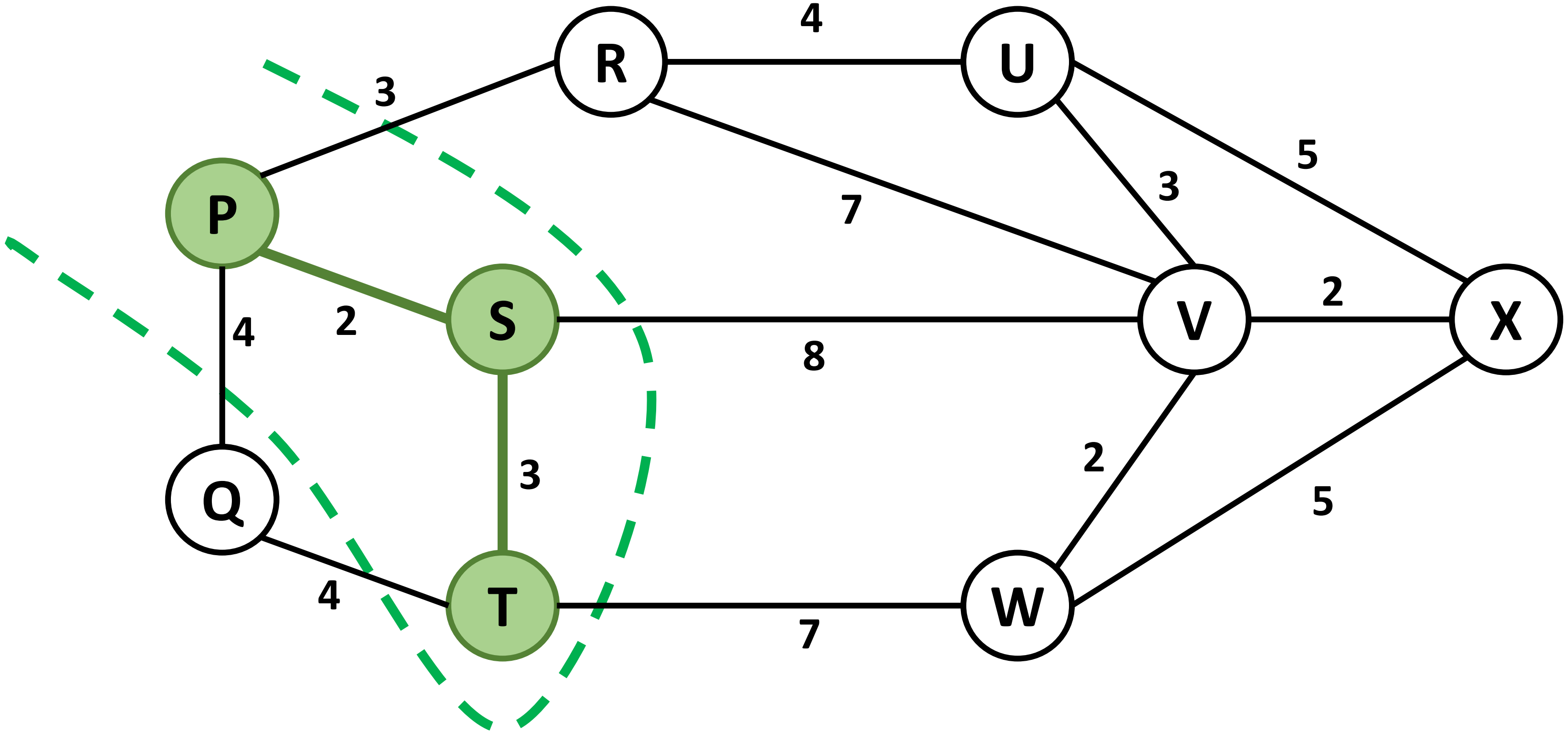
# Prim’s Algorithm – An Example: Step 1



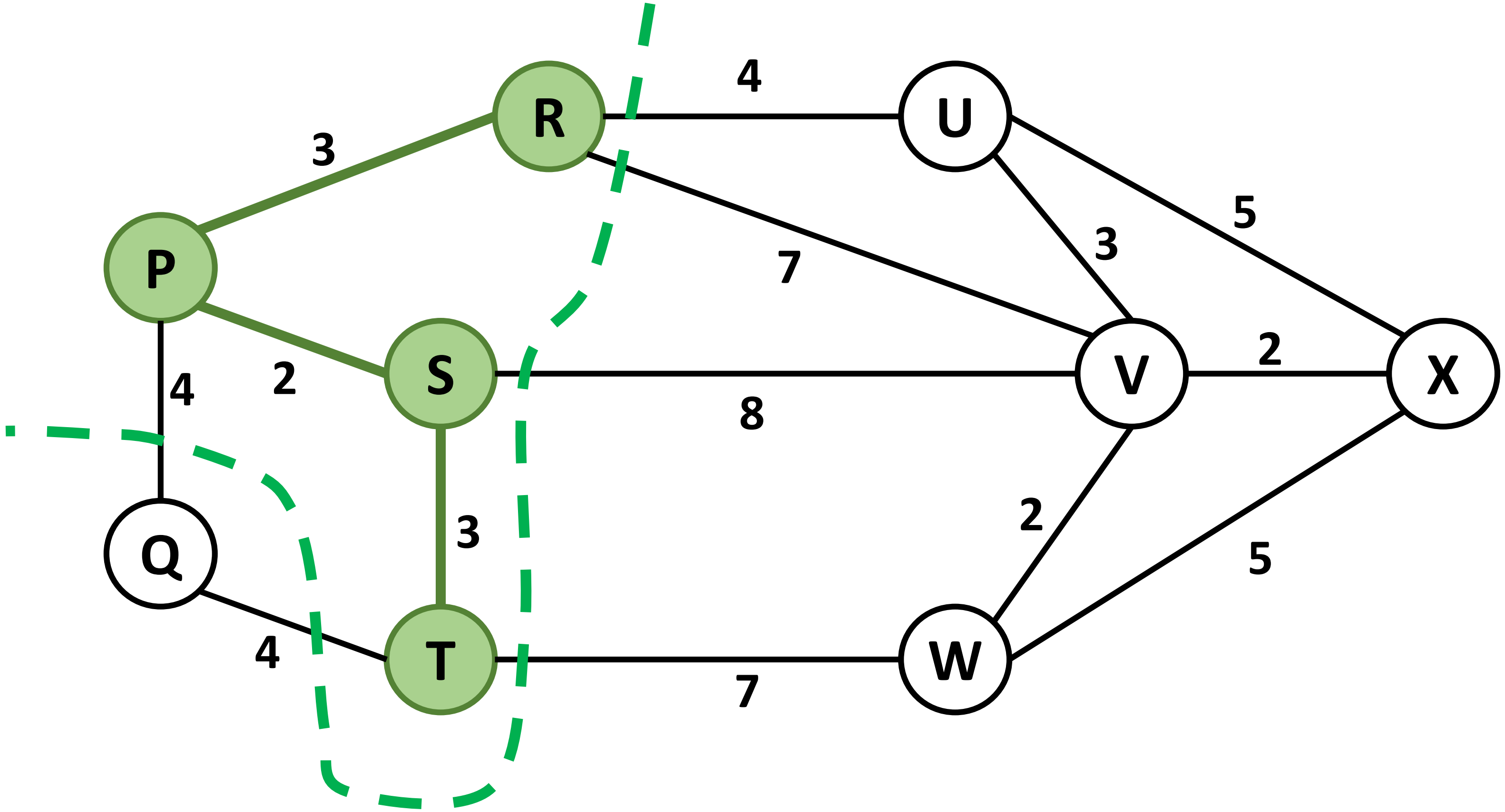
# Prim's Algorithm – An Example: Step 2



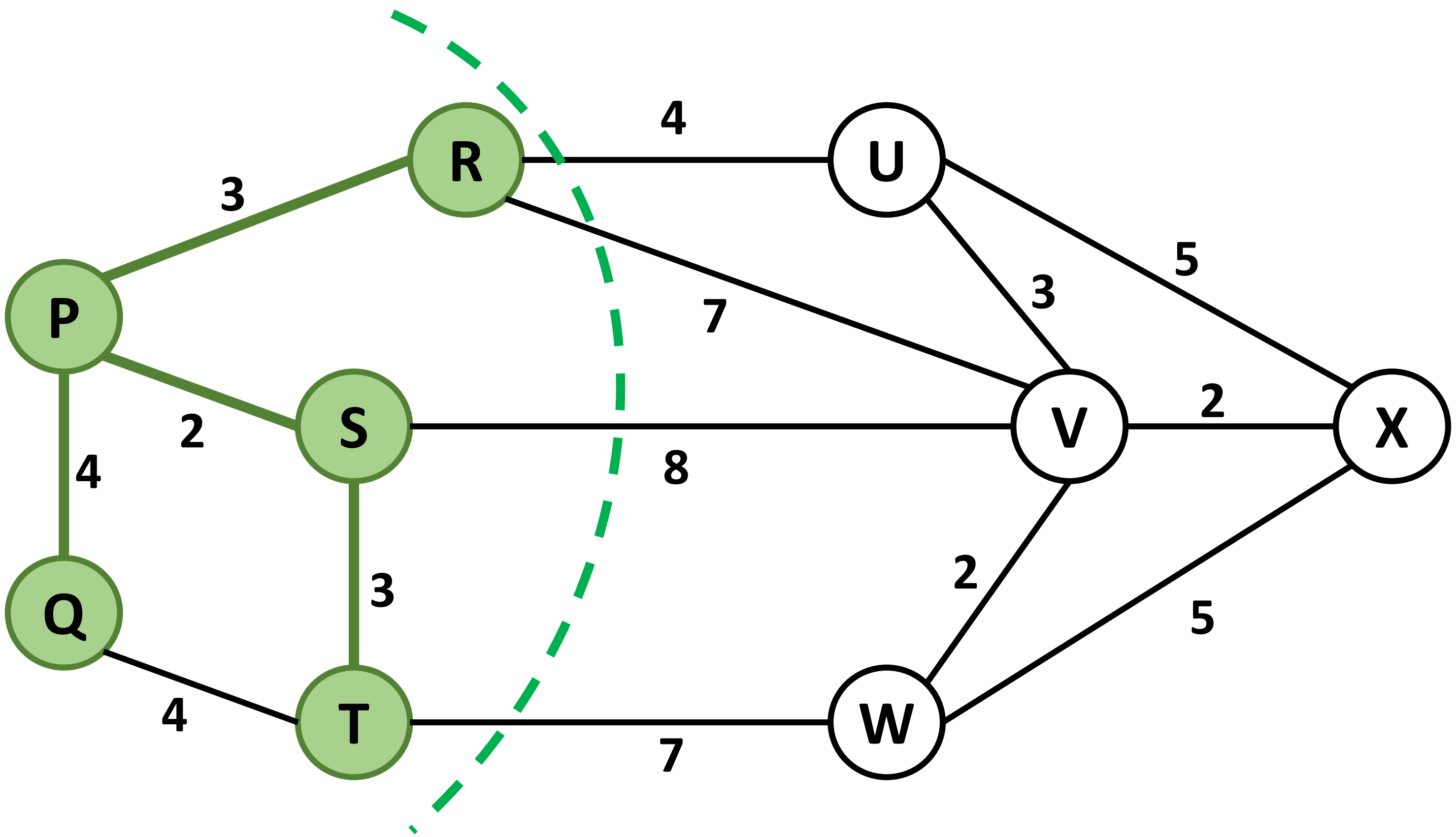
# Prim's Algorithm – An Example: Step 3



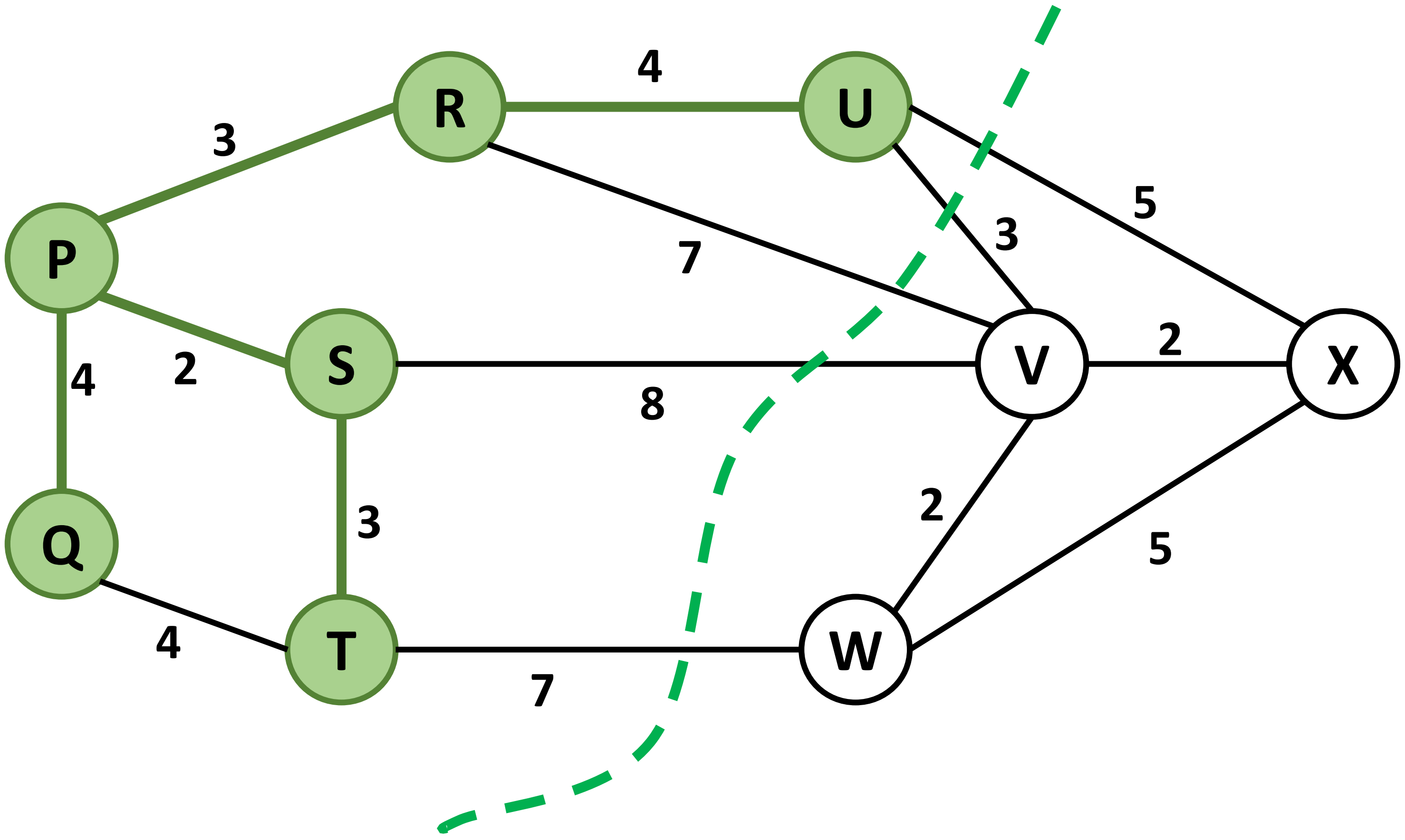
# Prim's Algorithm – An Example: Step 4



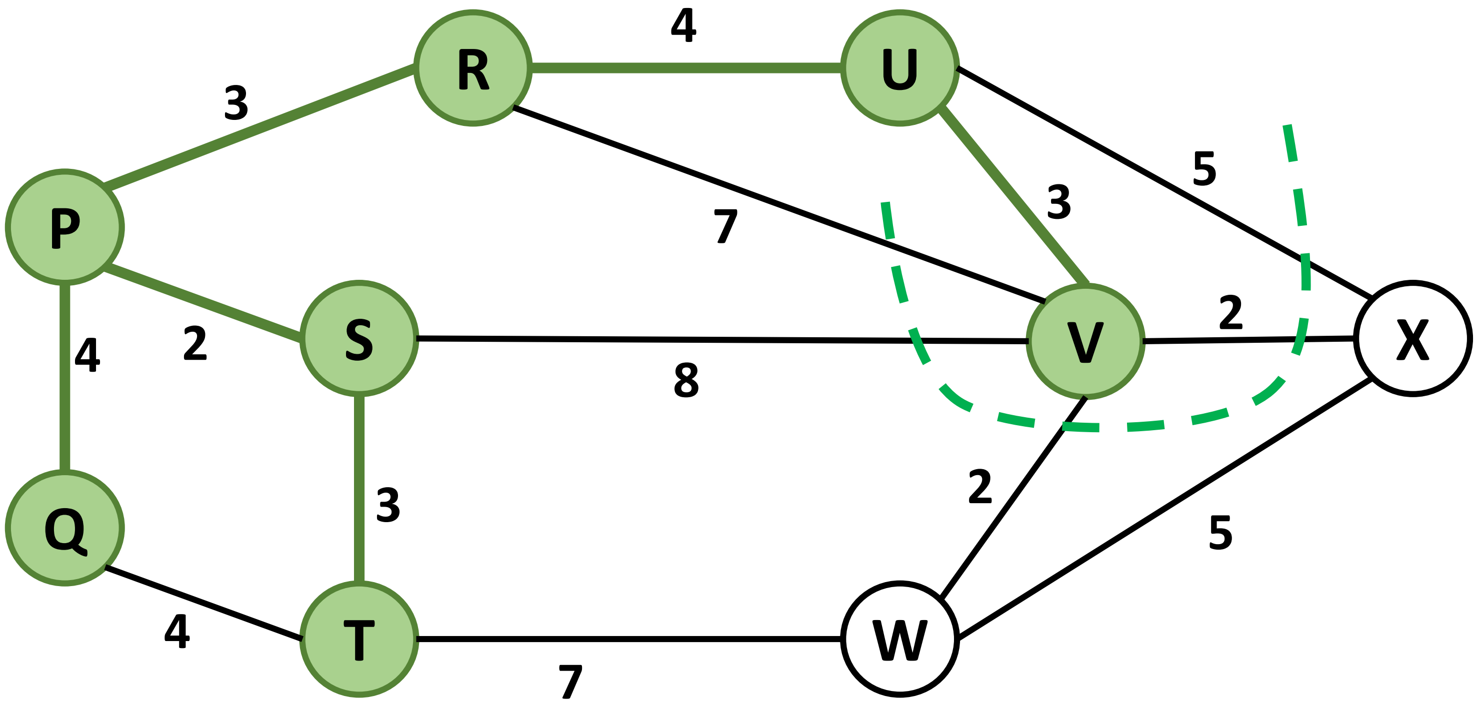
# Prim's Algorithm – An Example: Step 5



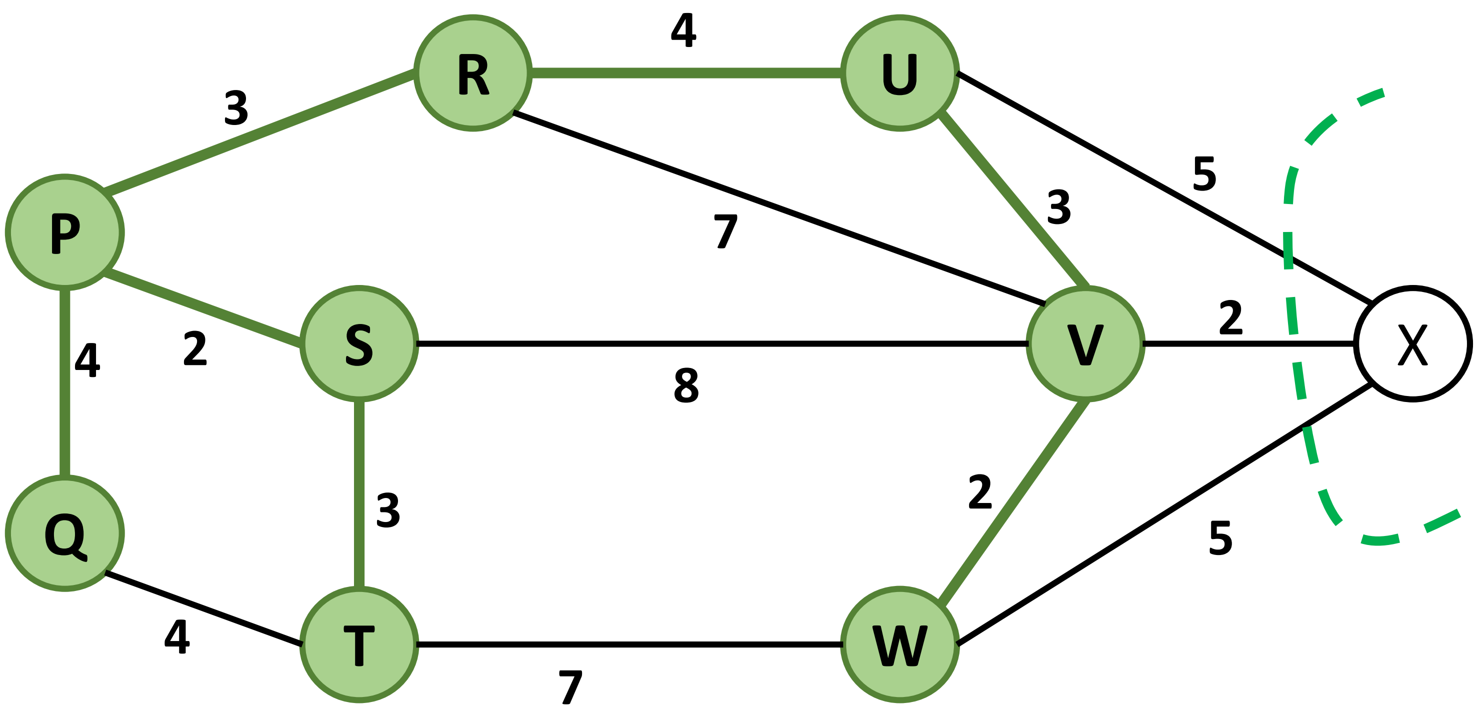
# Prim’s Algorithm – An Example: Step 6



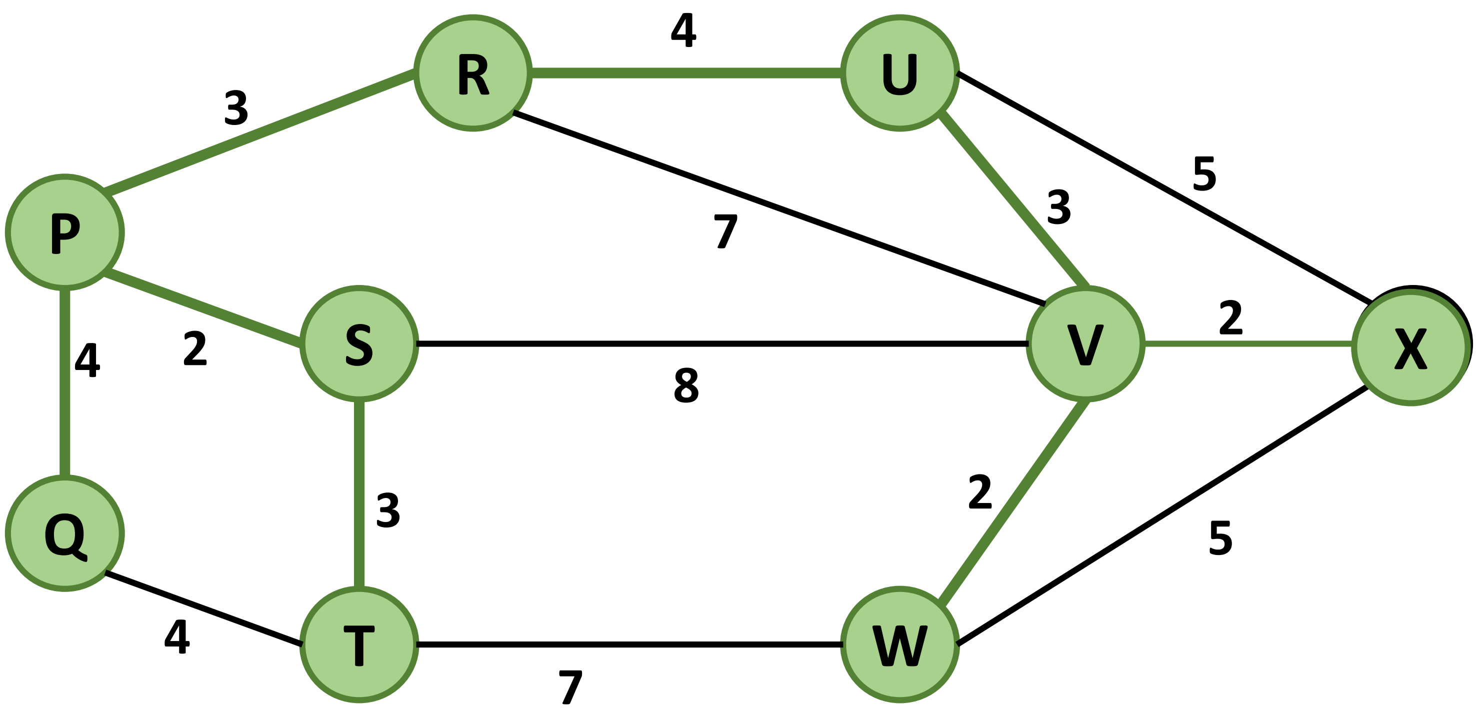
# Prim’s Algorithm – An Example: Step 7



# Prim’s Algorithm – An Example: Step 8



# Prim's Algorithm – An Example: Step 9



# Prim's Algorithm

- The algorithm was developed in 1930 by Czech mathematician Vojtěch Jarník
- Later rediscovered and republished by computer scientists Robert C. Prim in 1957 and Edsger W. Dijkstra in 1959.
- Therefore, it is also sometimes called the **Jarník's algorithm**, **Prim–Jarník algorithm**, **Prim–Dijkstra algorithm** or the **DJP algorithm**.

https://en.wikipedia.org/wiki/Prim%27s_algorithm

# Minimum Spanning Trees

- A tree of the nodes of the graph with minimum total edge weight.
- Both Prim's and Kruskal's algorithms are examples of greedy algorithms
- **Applications**
  - Reducing copper to connect multiple nodes in a electrical/electronic circuit
  - Minimizing network length (cable cost) to connect multiple routers / computers etc.

# Thank you!