CAD Algorithms

Shortest Path Algorithms

Mohammad Tehranipoor

ECE Department

Shortest Path

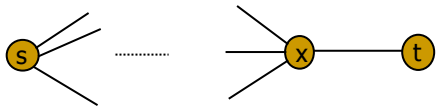
- Problem:
  - Find the best way of getting from **s** to **t** where **s** and **t** are vertices in a graph.
  - Best: Min (sum of the edge lengths of a path)
  - On a weighted graph:
    - Shortest path = Min $(\sum f(p))$, where $p \in P$ (P is the set of all paths)
- Example:
  - Traveling Salesman Problem (TSP)
- Fastest path or shortest path are the same.
- The length of a path can be actual mileage or time to drive it.

Assumption

- All edges have lengths (weights) that are positive numbers.
  - It makes the algorithm a lot easier.
  - It can be applied to both directed and undirected graph
- There are algorithms that handle negative weights/lengths/costs.
  - Bellman-Ford Algorithm

Path from Distance

Objective: Find the shortest path from *s* to *t*



Path: d(s,t)

d(s,t) = d(s,x) + d(x,t)

Therefore:

d(s,x) < d(s,t)

Dijkstra's Algorithm

- Given a connected weighted graph G=(V,E), a weight d:E⟶R^+ and a fixed vertex **s** in V, find the shortest path from **s** to each vertex **v** in V.

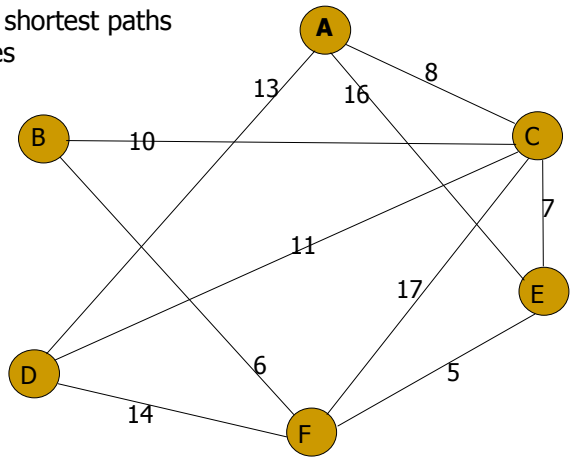
- **Dijkstra's algorithm** works almost the same as **Prim's algorithm**. The only difference is that it chooses an **edge** once at a time instead of **vertex** to find the shortest path or minimize d(s,x) + length(x,y).

Dijkstra's Algorithm

```
1: Let T be a single vertex s
2: While (T has fewer than n vertices) // n: total number of vertices
3: {
4:   Find edge (x,y)
5:        with x in T and y not in T Minimizing d(s,x)+length(x,y)
7:   Add (x,y) to T
8:  d(s,y) = d(s,x) + length (x,y)
9: }
```

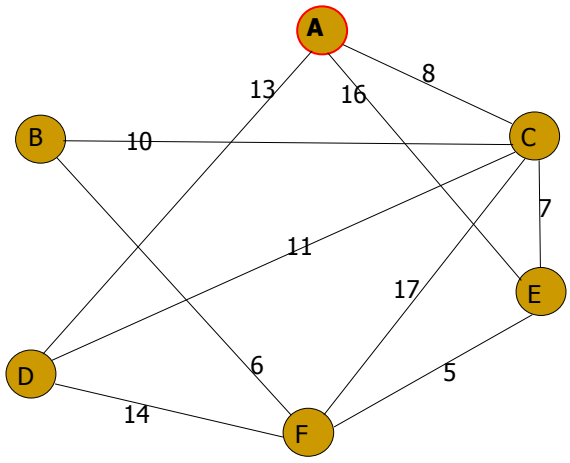
Example

Problem: Find the shortest paths from A to all vertices



Cont.

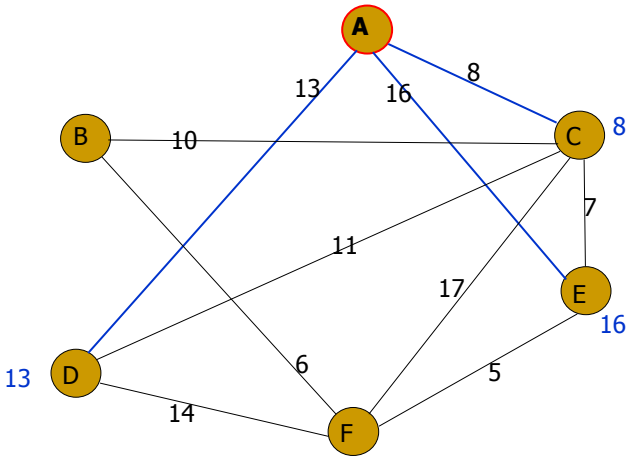
T = {A}



Cont.

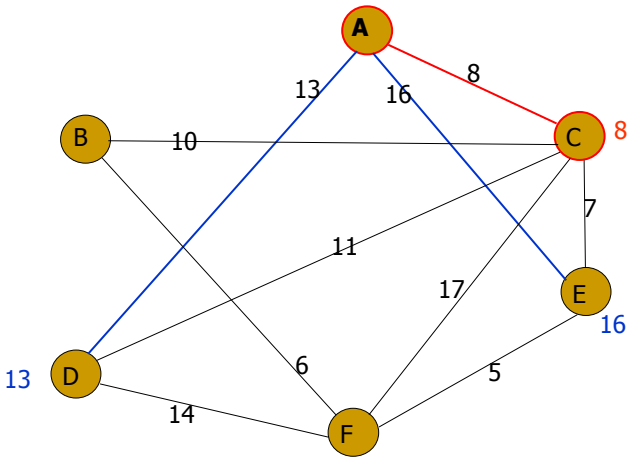
T = {A}

AC	8
AD	13
AE	16



Cont.

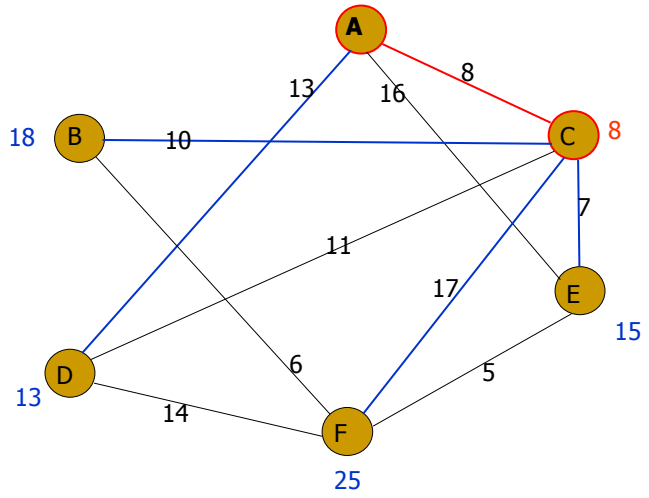
T = {A, C}



Cont.

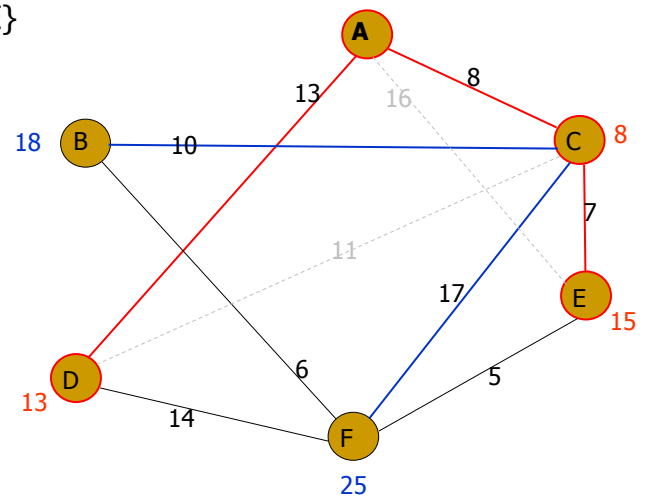
T = {A, C}

~~AD~~	13
ACD	19
ACB	18
~~AE~~	16
ACE	15
ACF	25



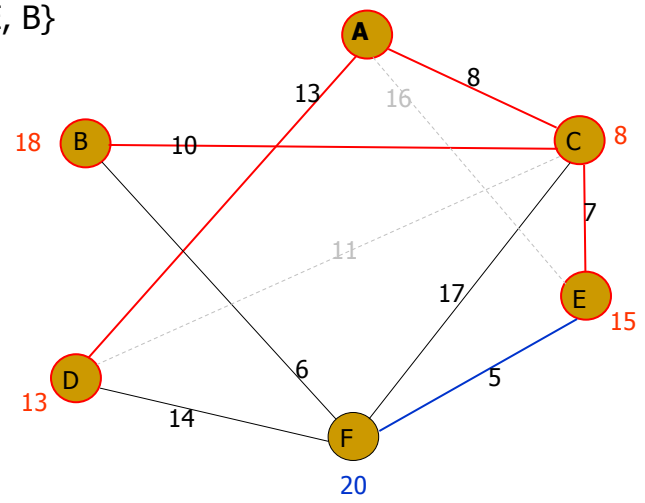
Cont.

T = {A, C, D, E}



Cont.

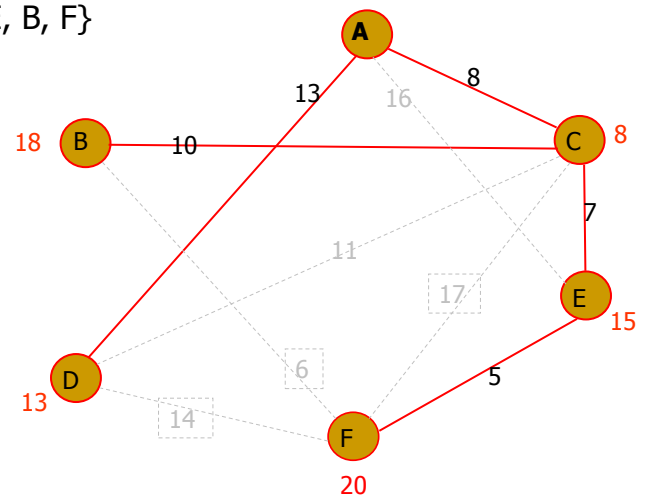
T = {A, C, D, E, B}



Cont.

T = {A, C, D, E, B, F}

AC	8
AD	13
ACE	15
ACB	18
ACEF	20



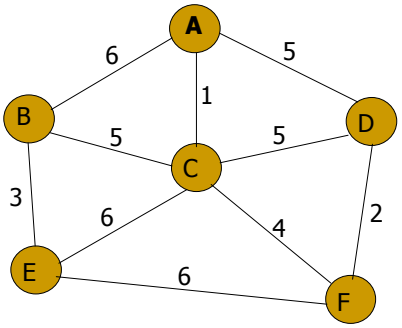
Prim's Algorithm

```
{
        T = ϕ ;
        U = {first vertex};
        while (U ≠ V)
         {
            let (u, v) be the lowest cost edge
            such that u ∈ U and v ∈ V - U;
T = T ∪ {(u, v)}
U = U ∪ {v}
         }
 }
```


Prim's Algorithm

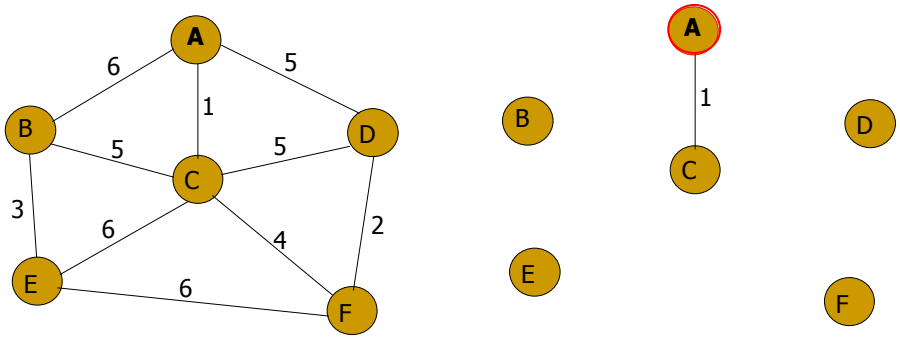
Problem: Find the shortest paths from A to all vertices

U = {}



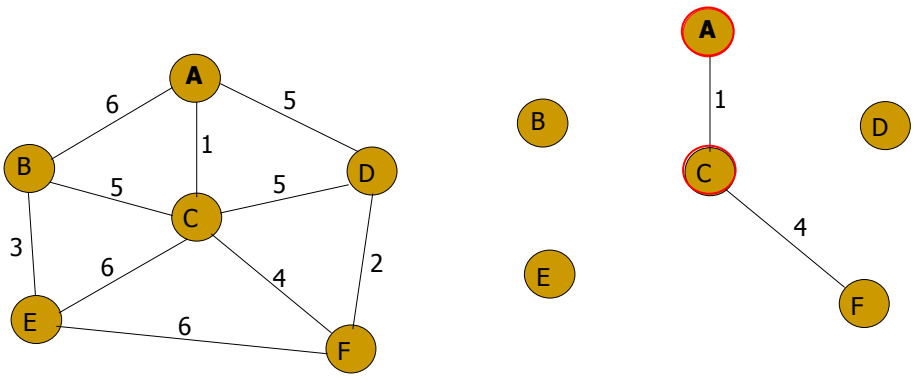
Cont.

U = {A}



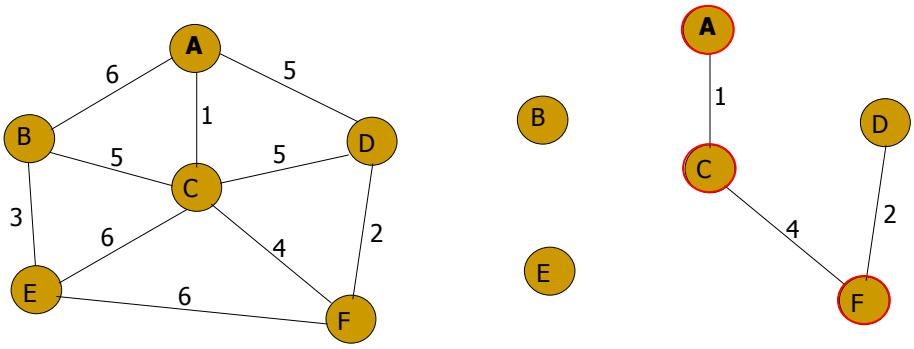
Cont.

U = {A, C}



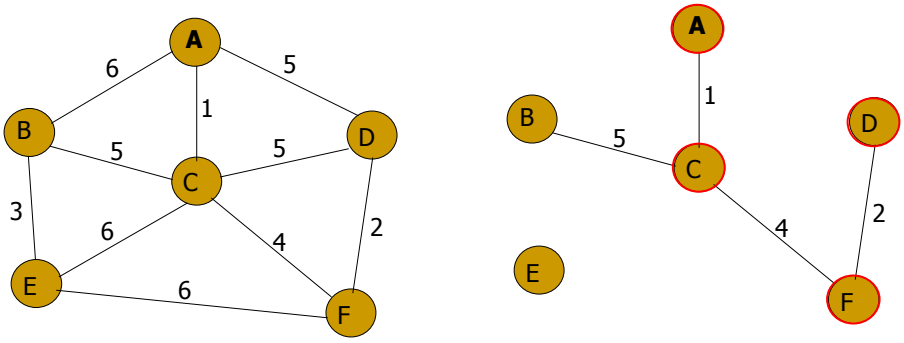
Cont.

U = {A, C, F}



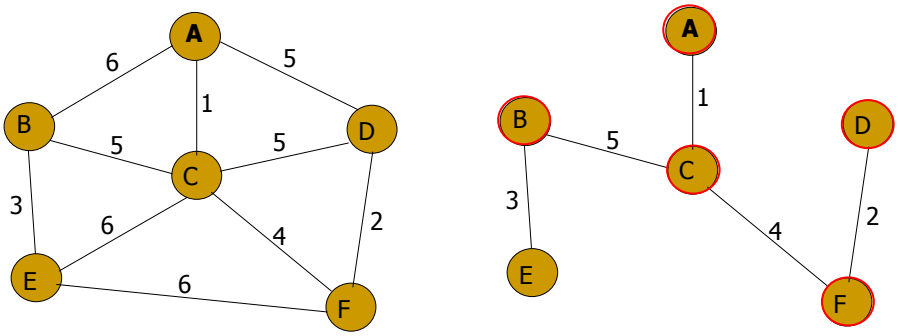
Cont.

T = {A, C, F, D}



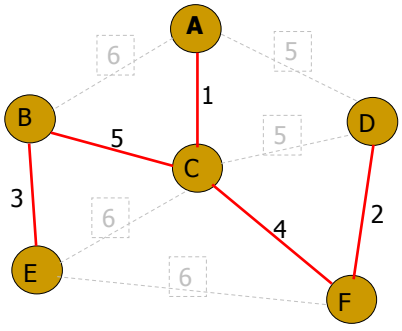
Cont.

T = {A, C, F, D, B}



Cont.

T = {A, C, F, D, B, E}



This algorithm is more suitable for MST problem.

CAD Algorithms

Minimum Spanning Tree Algorithms

Mohammad Tehranipoor

ECE Department

MST

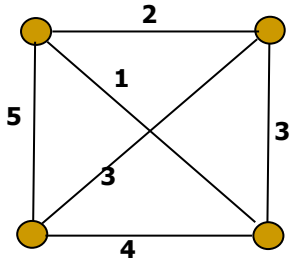
- MST is a well-solved combinatorial optimization problem.
- Definition:
  - A **tree** is a connected graph without cycles.
- Properties of Trees:
  - A graph is a tree if and only if there is only and only one path joining any two of its vertices.
  - A connected graph is a tree if and only if it has N vertices and N-1 edges.

MST (Cont.)

- Definition:
  - A **subgraph** that spans (reaches out to) all vertices of a graph is called a spanning subgraph.
  - A subgraph that is a tree and that spans (reaches out to) all vertices of the original graph is called a **spanning tree**.
  - Among all the spanning trees of a weighted and connected graph, the one (possibly more) with the least total weight is called a **minimum spanning tree (MST)**.

Example

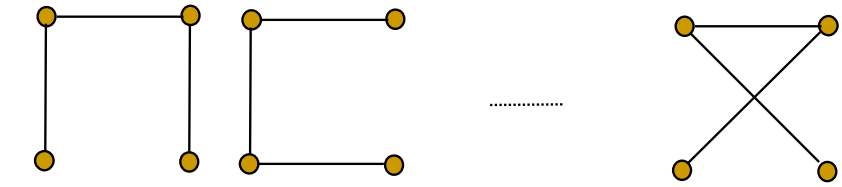
- A simple example:
  - A graph may have many spanning trees.



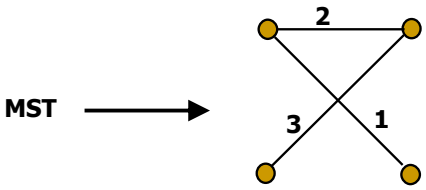
  - This graph has 16 spanning trees.

Example: Trees

- 16 spanning trees:



Tree #: 1 2 16



Problem

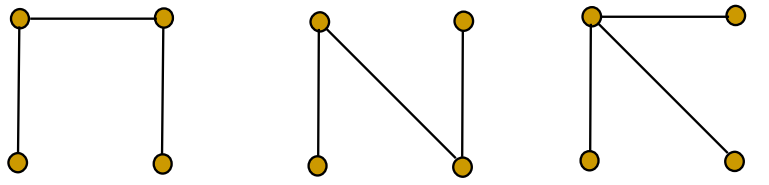
- Suppose the edges of the graph have weights.
  - The weights of a tree is sum of weights of its edges.
  - Different trees may have different weights.
- Problem:
  - How to find minimum spanning tree?
- **Solutions**:
  - Heuristic
  - Greedy
  - Exhaustive search
  - …

Why Minimum Spanning Tree?

- Example:
  - Phone Network Design:
    - One has a business with several offices and wants to lease phone lines to connect them up with each other; the phone company charges differently to connect different pair of cities. How he can find a set of lines that connects all the offices with a minimum total cost.
  - Traveling Salesman Problem (TSP):
    - Another convenient way to solve this problem is to find the shortest path that visits each point at least once.
  - VLSI Problem: Global Routing
    - MST is used in global routing
    - Steiner Tree is used in detailed (channel) routing

MST/TSP

- In general the MST weight is less than the TSP. In MST, we are not limited only to paths.
- What is path:



Path Path Not a path

Not a TSP solution

How to Find MST?

- An exhaustive search:
  - List all spanning trees and find the minimum one. Not applicable in practice especially in VLSI problems.
- Heuristic Algorithms
- Greedy Algorithms
  - Prim's Algorithm

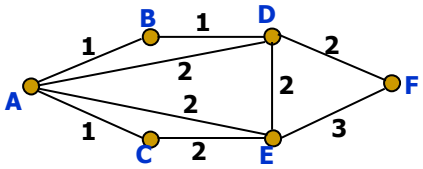
Kruskal's Algorithm

- It is easy to understand and the best for solving problems by hands.
- Kruskal's Algorithm:
  - Step 1: Sort the edges of graph **G** in increasing order by weight
  - Step 2: Keep a subgraph **S** of **G**, initially empty
  - Step 3: For each edge **e** in sorted order
    - If the endpoints of **e** are disconnected in **S**
      - Add **e** to **S**
  - Step 4: Return **S**

Example

Edge	AB	AC	BD	AD	AE	CE	DE	DF	EF
Weight	1	1	1	2	2	2	2	2	3

S={}

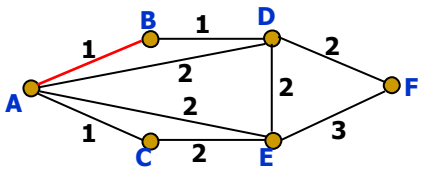


Cont.

Edge	AB	AC	BD	AD	AE	CE	DE	DF	EF
Weight	1	1	1	2	2	2	2	2	3

Edges: 1, 1, 1, 2, 2, 2, 2, 2, 3

S={AB}

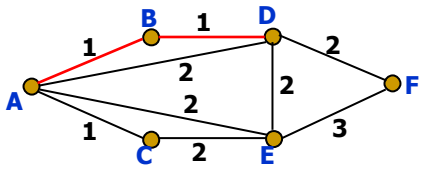


Cont.

Edge	AB	AC	BD	AD	AE	CE	DE	DF	EF
Weight	1	1	1	2	2	2	2	2	3

Edges: 1, 1, 1, 2, 2, 2, 2, 2, 3

S={AB, BD}

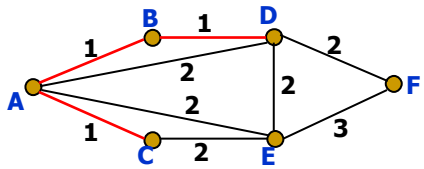


Cont.

Edge	AB	AC	BD	AD	AE	CE	DE	DF	EF
Weight	1	1	1	2	2	2	2	2	3

Edges: 1, 1, 1, 2, 2, 2, 2, 2, 3

S={AB, BD, AC}

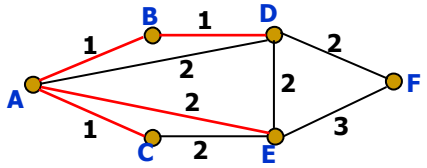


Cont.

Edge	AB	AC	BD	AD	AE	CE	DE	DF	EF
Weight	1	1	1	2	2	2	2	2	3

Edges: 1, 1, 1, 2, 2, 2, 2, 2, 3

S={AB, BD, AC, AE}

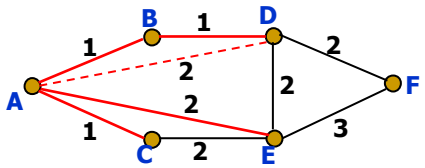


Cont.

Edge	AB	AC	BD	AD	AE	CE	DE	DF	EF
Weight	1	1	1	2	2	2	2	2	3

Edges: 1, 1, 1, 2, 2, ~~2~~, 2, 2, 3

S={AB, BD, AC, AE}

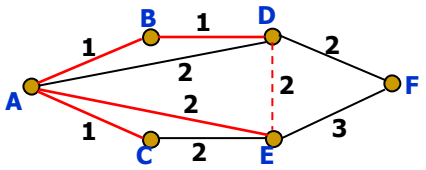


Cont.

Edge	AB	AC	BD	AD	AE	CE	DE	DF	EF
Weight	1	1	1	2	2	2	2	2	3

Edges: 1, 1, 1, 2, 2, 2, ~~2~~, 2, 3

S={AB, BD, AC, AE}

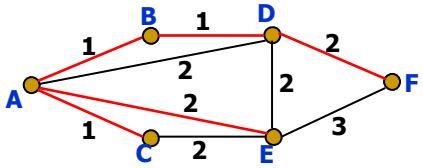


Cont.

Edge	AB	AC	BD	AD	AE	CE	DE	DF	EF
Weight	1	1	1	2	2	2	2	2	3

Edges: 1, 1, 1, 2, 2, 2, 2, 2, 3

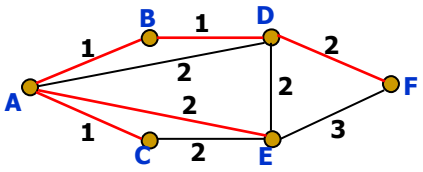
S={AB, BD, AC, AE, DF}



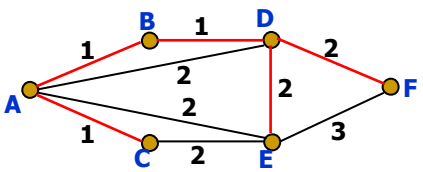
Total Cost = 7

Example

- A graph may have more than one MST.



Total Cost = 7



Total Cost = 7

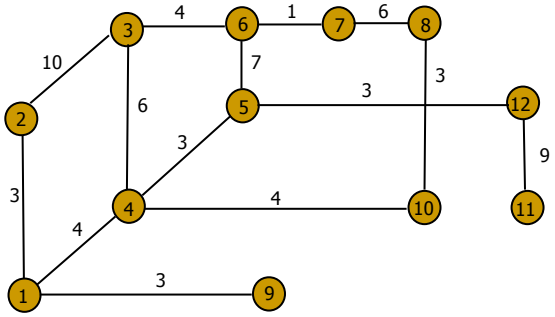
- This algorithm is known as a **greedy algorithm** because it chooses the cheapest edge at each step and adds to S.

Prim's Algorithm

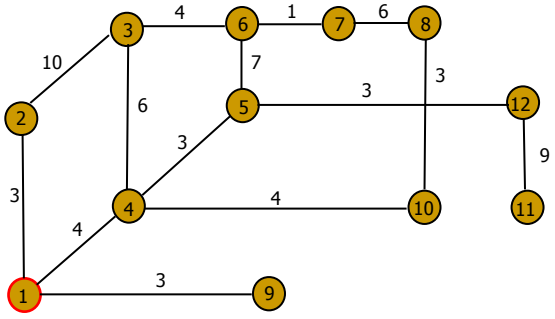
- Step 1: Pick any vertex as a starting vertex. Call it **s**.
- Step 2: Find the nearest neighbor of **s** (call it **p1**). Choose **sp1** and **p1** such that **sp1** is the cheapest edge in the graph that does not close the selected edges. Add **sp1** to **S**.
- Step 3: Return subgraph **S**.

Example

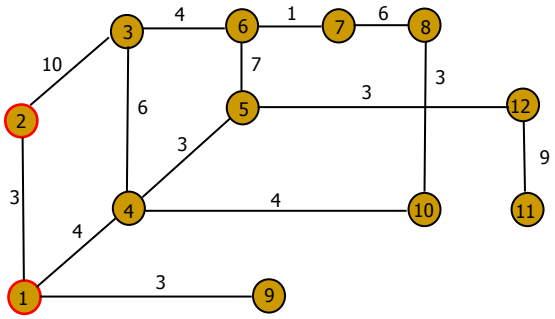
Prim's Algorithm:



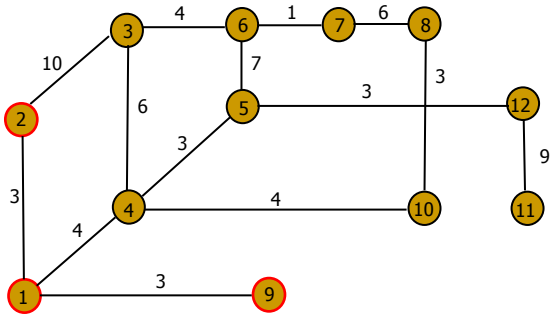
Cont.



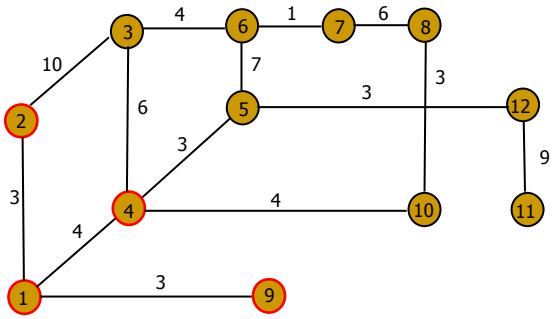
Cont.


1
2
3
4
5
6
7
8
9
10
11
12
4
1
6
10
7
3
6
3
3
9
3
4
4
3

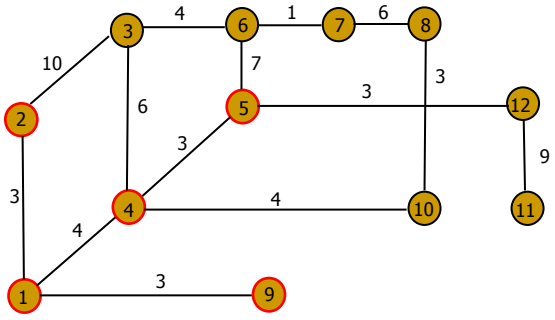
Cont.


1
2
3
4
5
6
7
8
9
10
11
12
4
1
6
10
7
3
6
3
3
9
3
4
4
3

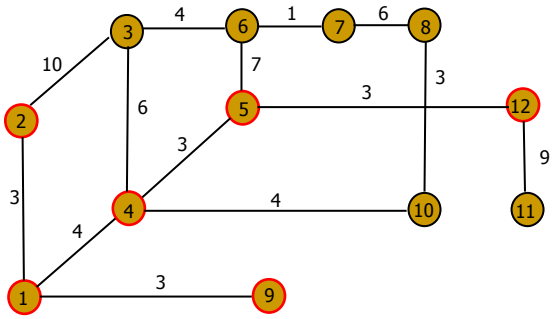
Cont.


3
4
6
1
7
6
8
10
7
3
2
6
5
3
12
3
9
3
4
4
10
11
4
1
3
9

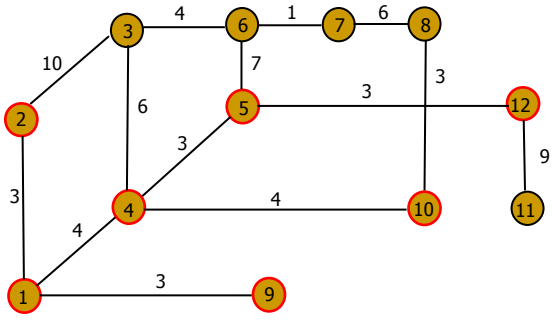
Cont.


3
4
6
1
7
6
8
10
7
3
2
6
5
3
12
3
9
3
4
4
10
11
4
1
3
9

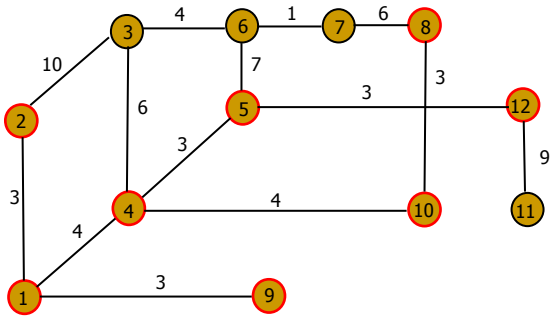
Cont.


3
4
6
1
7
6
8
10
7
3
2
6
5
3
12
3
9
3
4
4
10
11
4
1
3
9

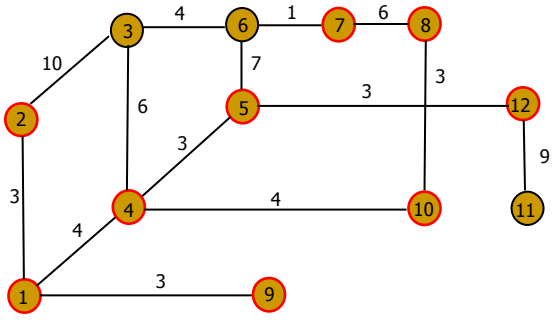
Cont.


3
4
6
1
7
6
8
10
7
3
2
6
5
3
12
3
9
3
4
4
10
11
4
1
3
9

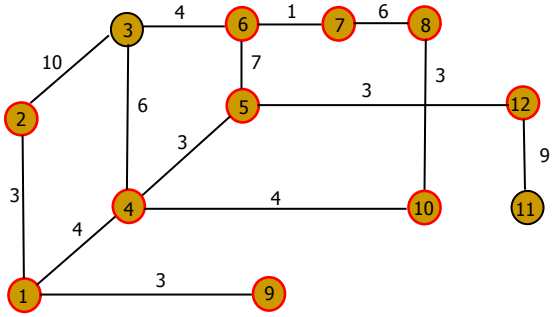
Cont.


3
4
6
1
7
6
8
10
7
3
2
6
5
3
12
3
9
3
4
4
10
11
4
1
3
9

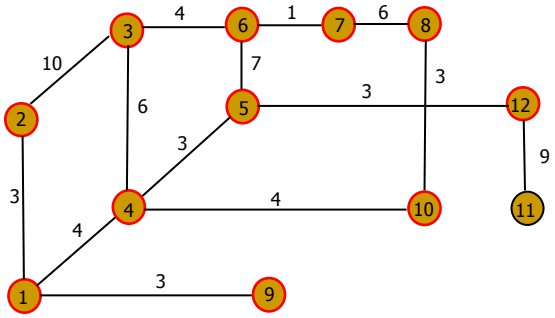
Cont.


3
4
6
1
7
6
8
10
7
3
2
6
5
3
12
3
9
3
4
4
10
11
4
1
3
9

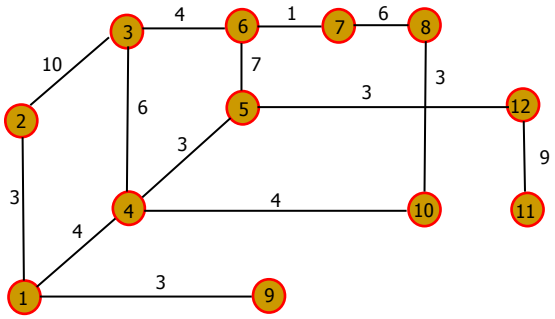
Cont.


3
4
6
1
7
6
8
10
7
3
2
6
5
3
12
3
9
3
4
4
10
11
4
1
3
9

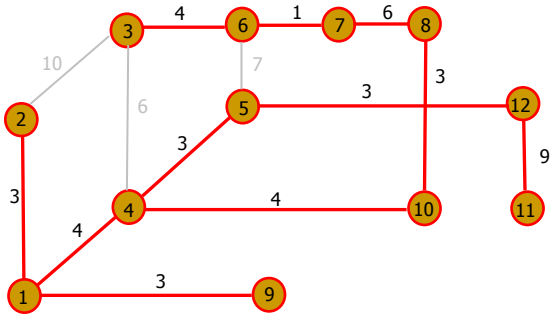
Cont.


3
4
6
1
7
6
8
10
7
3
2
6
5
3
12
3
9
3
4
4
10
11
4
1
3
9

Cont.


3
4
6
1
7
6
8
10
7
3
2
6
5
3
12
3
9
3
4
4
10
11
4
1
3
9

Cont.


3
4
6
1
7
6
8
10
7
3
2
6
5
3
12
3
9
3
4
4
10
11
4
1
3
9

More Algorithms

- Boruvka's Algorithm:
  - The idea is to do steps like Prim's algorithm in parallel all over the graph at the same time.
- Hybrid Algorithm:
  - Combine two of the classical algorithms and do better than either one alone.