Graph Algorithms: Minimum Spanning Trees

DR. OLENA SYROTKINA

University of Windsor

University of Windsor

Learning Objectives

- Understand Minimum Spanning Trees and their Algorithms
- Apply the Knowledge of Minimum Spanning Trees to Solve Practical Problems

Today’s discussion

What a spanning tree is and how it works

Minimum spanning tree

Importance of spanning trees

Applications of minimum spanning trees

Essential properties of spanning trees

Graphs and their different types

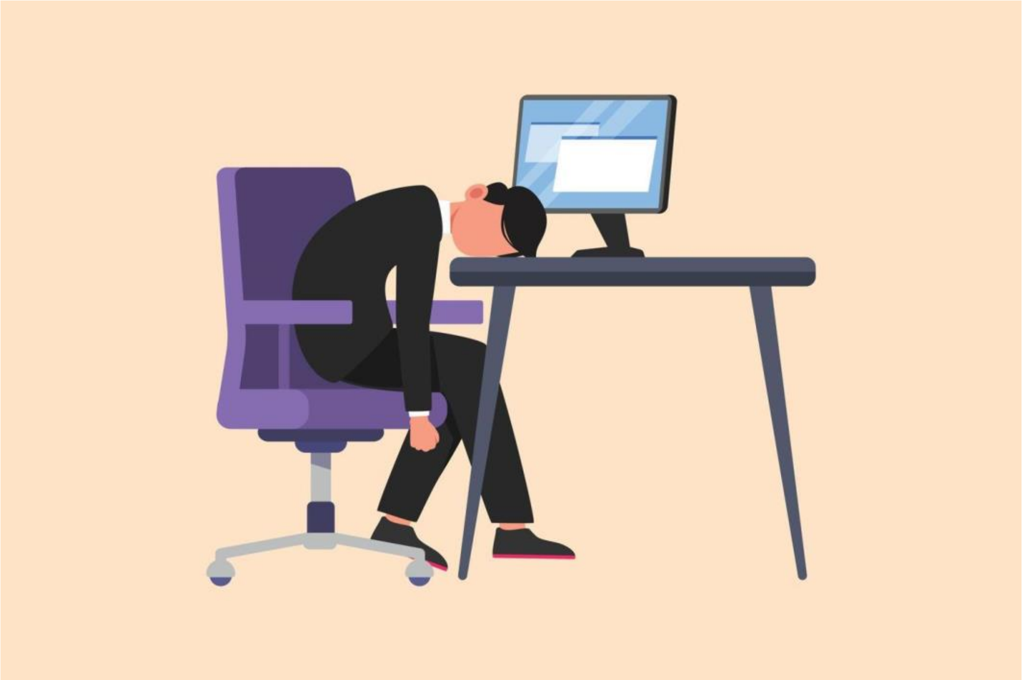
Minimum spanning tree algorithms

Scenario 1

Paul works as the head of the Network Administration department. He and his team are responsible for managing LAN for the office.

University of Windsor

Scenario 1



Paul observed network loop in the routing system which caused slow irregular network connection at the beginning and an eventual network failure.

Scenario 1

The office network went down due to network looping. Now Paul and his team need to do something in order to resolve this problem permanently.

Any ideas?

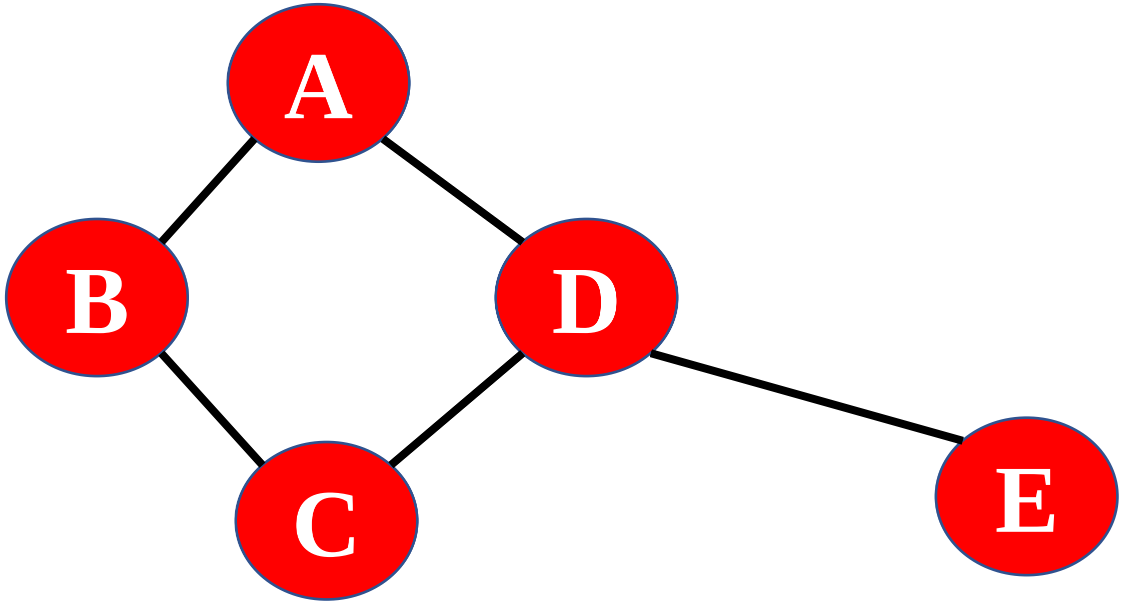
What is a Graph?

Graph (V,E)

A set of vertices (V) and edges (E)

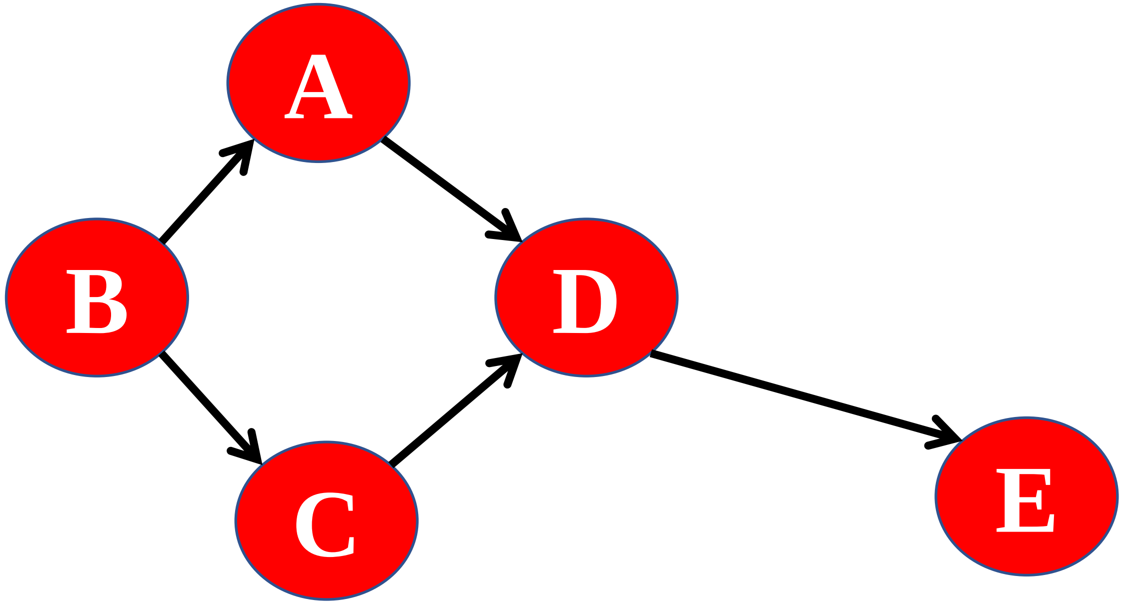
Undirected Graph

The graph in which all the edges do not point to any specific location.



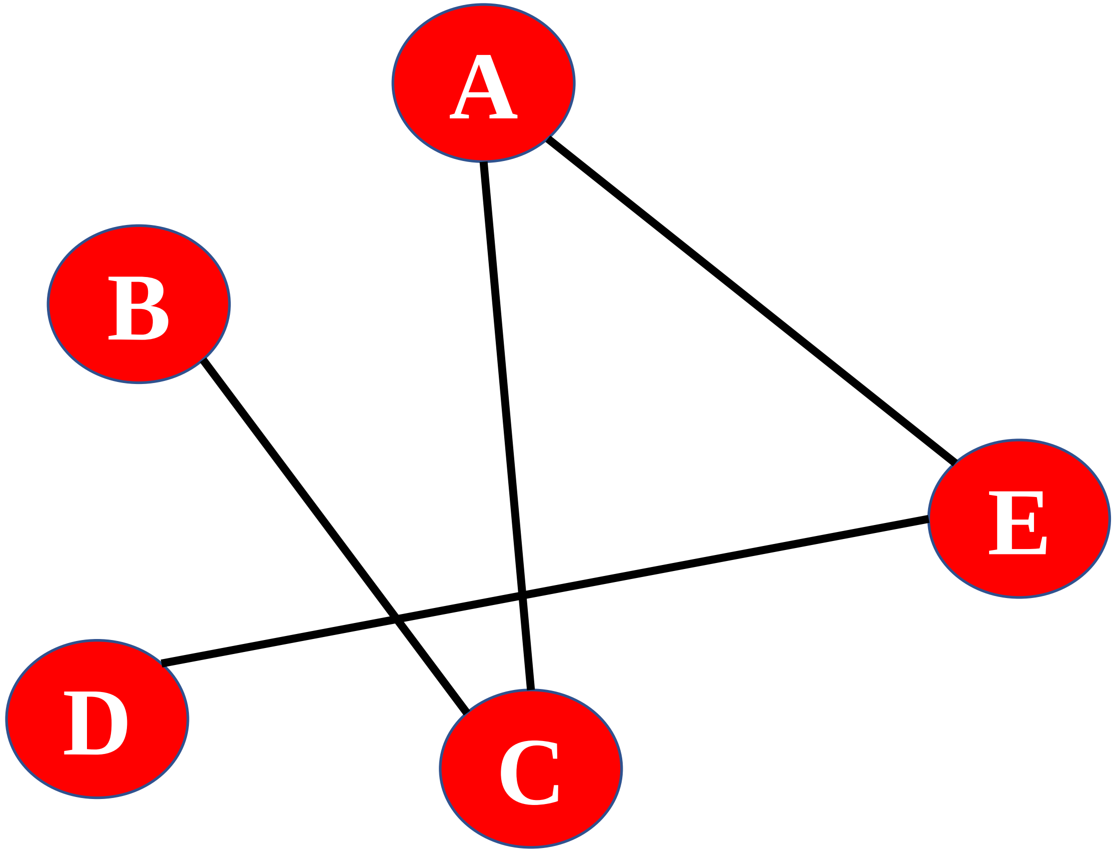
Directed Graph

The graph in which all the edges point to a specific location.



Connected Graph

The graph in which there is a path from one vertex to any other vertex



What is a Spanning Tree?

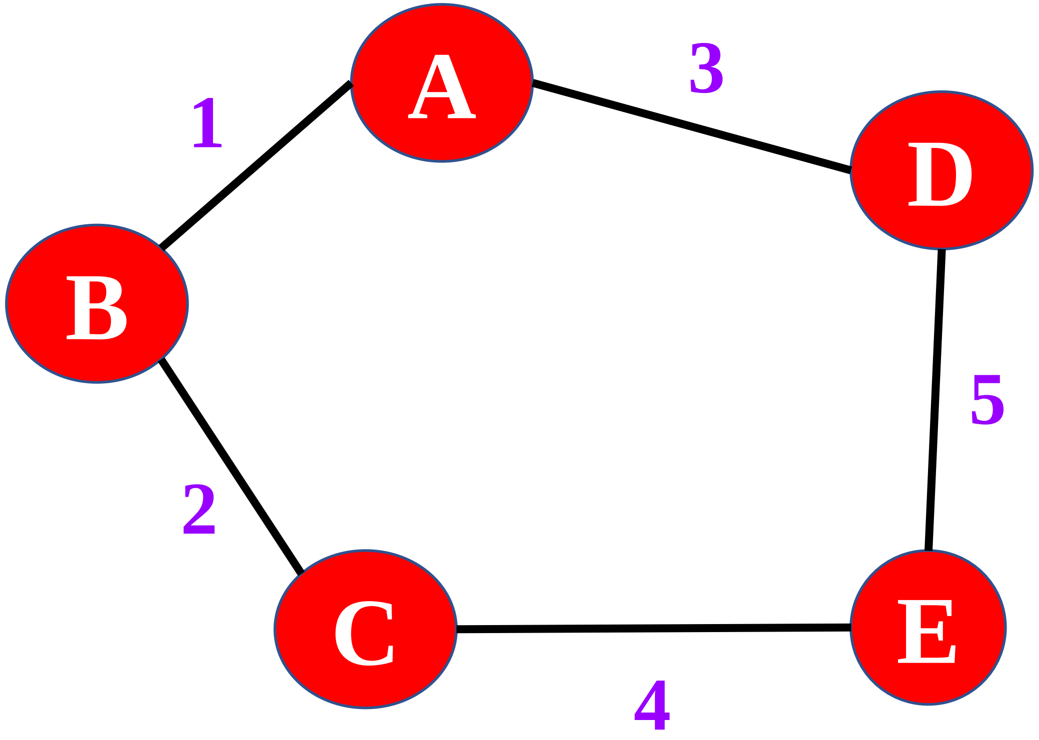
- If we have **graph G** with **vertices V** and **edges E** then that graph can be represented as **G(V, E)**.

- for this graph G(V, E) if we construct a tree structure G'(V', E') such that the formed tree structure follows constraints mentioned below, then that structure can be called as a **Spanning Tree**.

1. V' = V
2. E' = |V| - 1

Creating Spanning Trees for Given Graph

Graph G (V, E) consists of 5 edges and 5 vertices. Each edge has some weight W assigned to it.

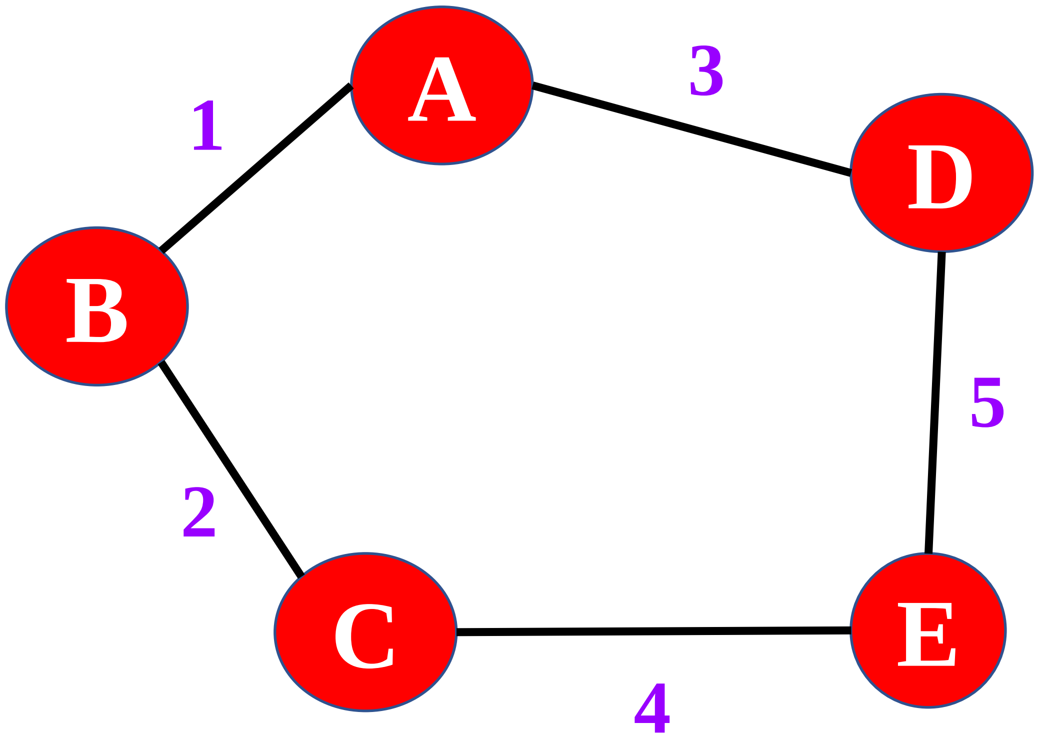


Creating Spanning Trees for Given Graph

To create G’ (V’, E’) from G(V, E) we need to make sure that

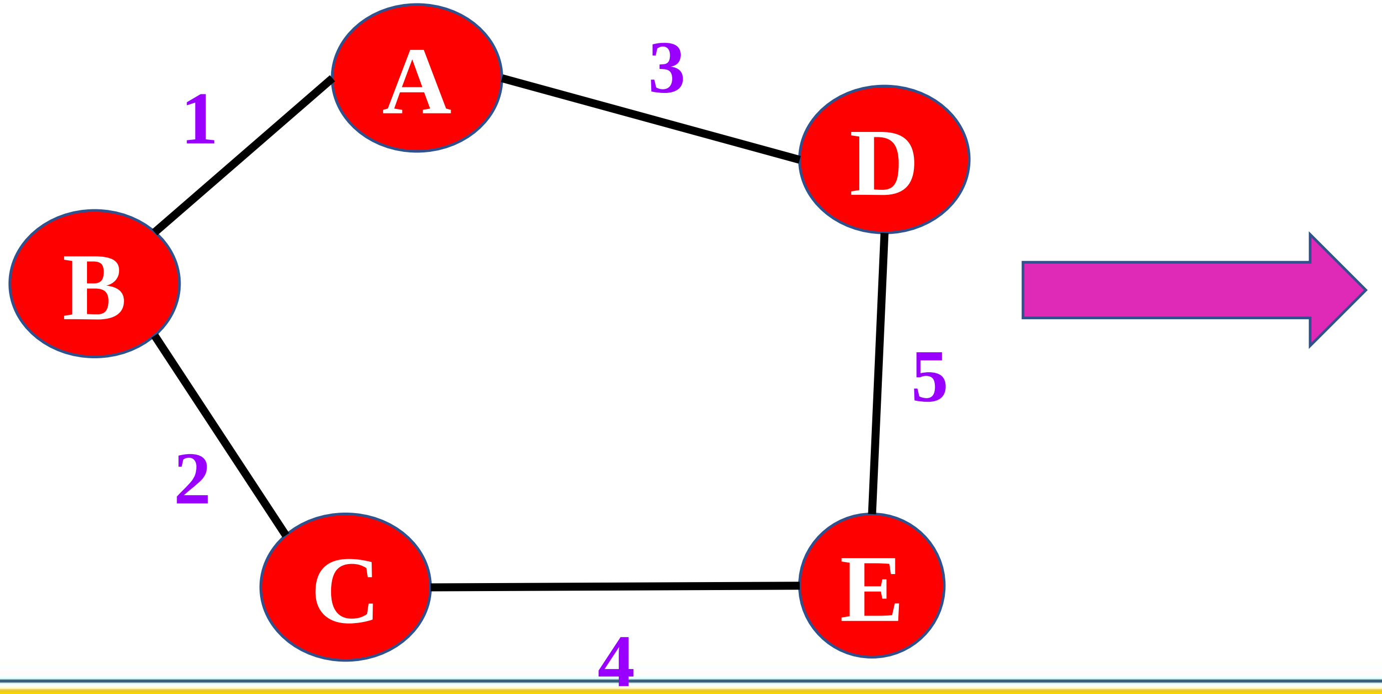
V’ = V

E’ = |V| -1

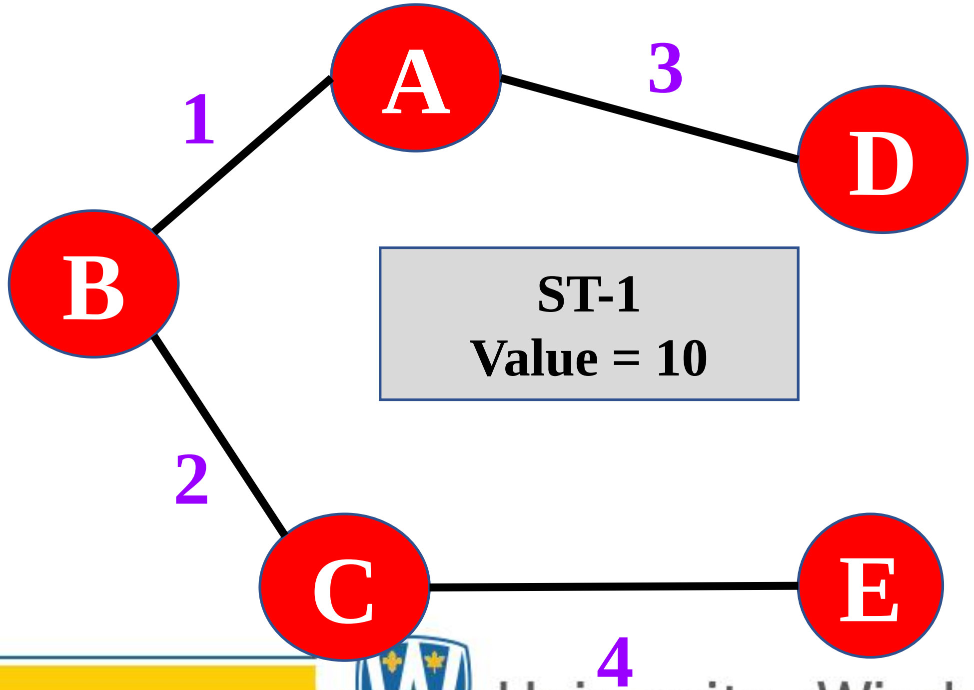


Creating Spanning Trees for Given Graph

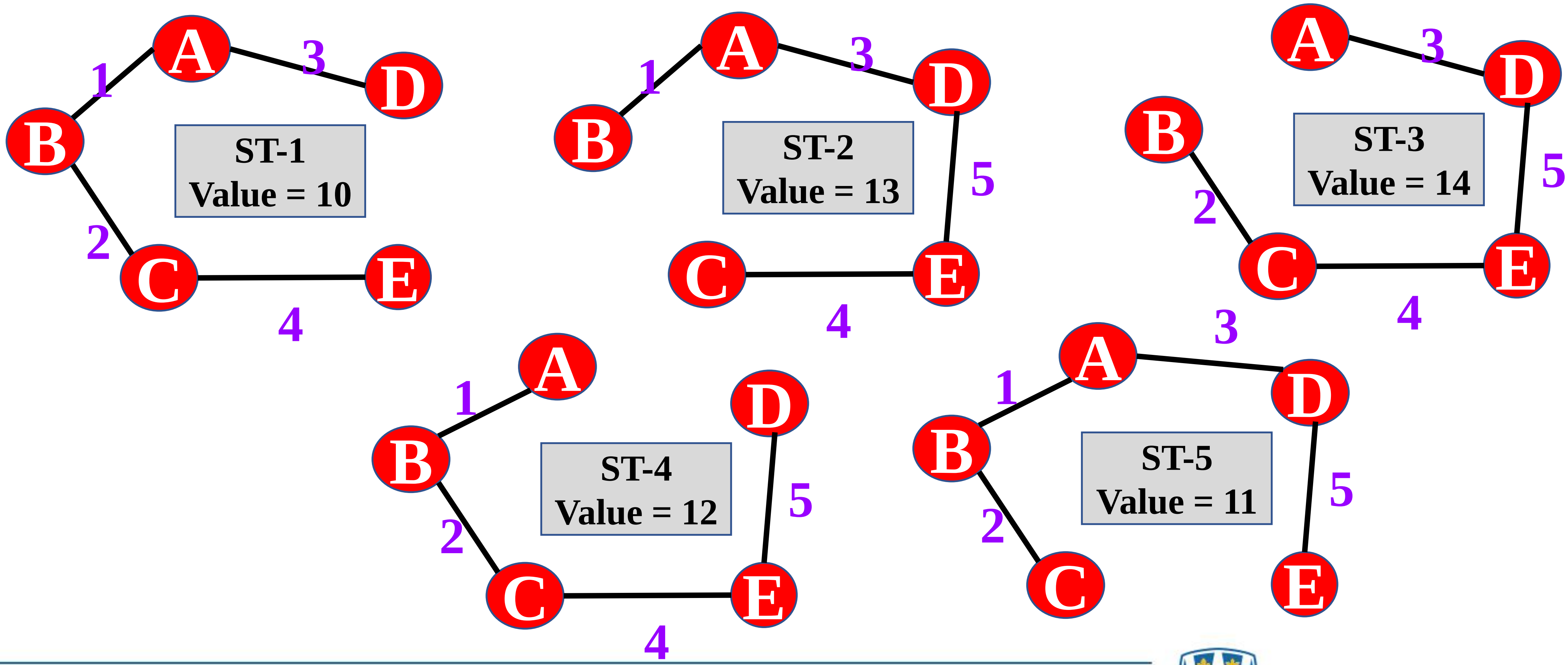
G(V, E)
where V=5, E=5



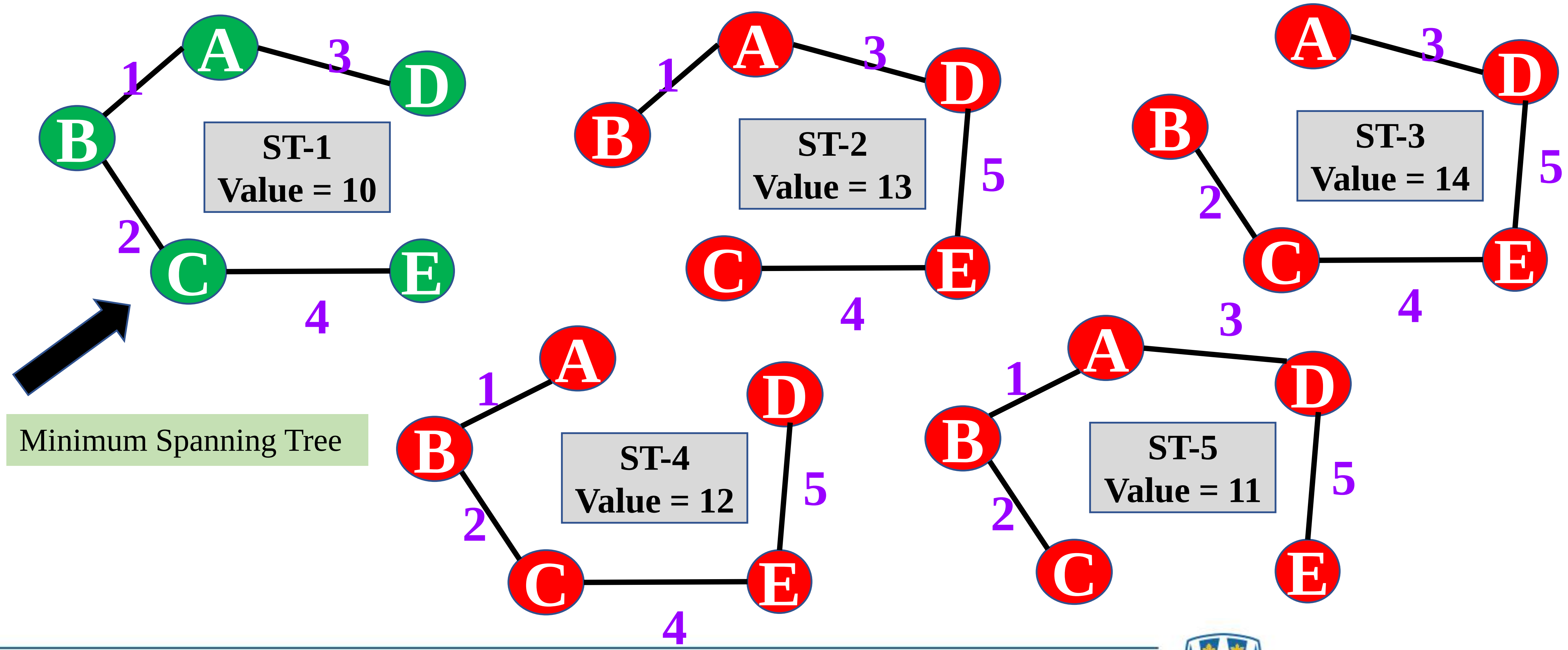
G’(V’, E’)
where V’=5, E’=4



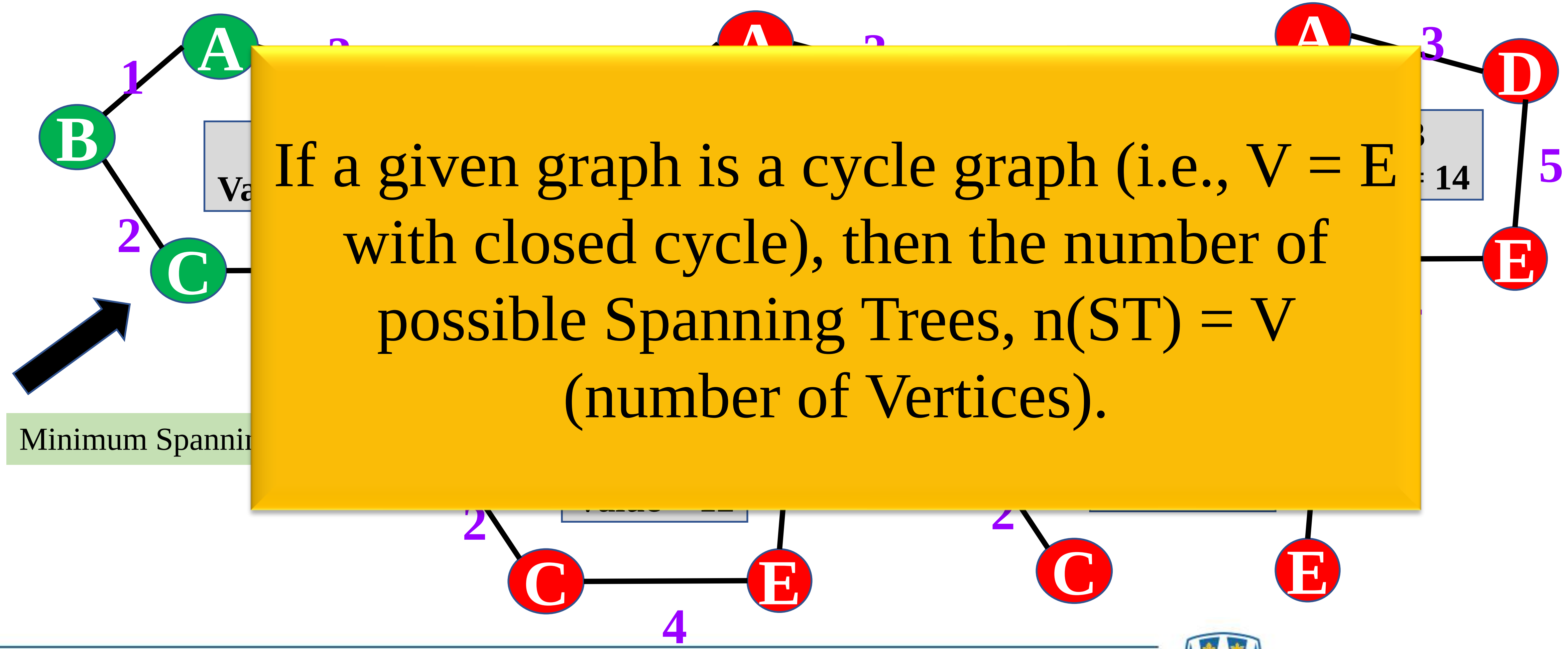
Creating Spanning Trees for Given Graph



Creating Spanning Trees for Given Graph



Creating Spanning Trees for Given Graph



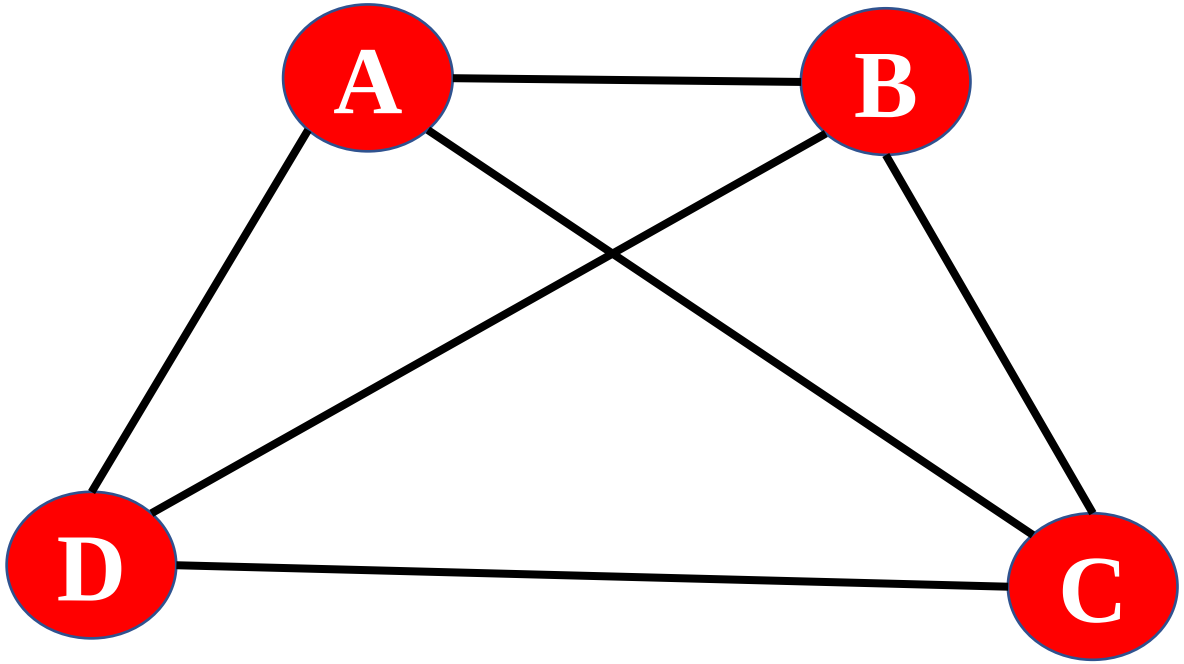
If a given graph is a cycle graph (i.e., V = E with closed cycle), then the number of possible Spanning Trees, n(ST) = V (number of Vertices).

Complete Graph

Graph G (V, E) is a Complete Graph (i.e. each pair of vertices is connected by unique edge), number of possible spanning trees will be

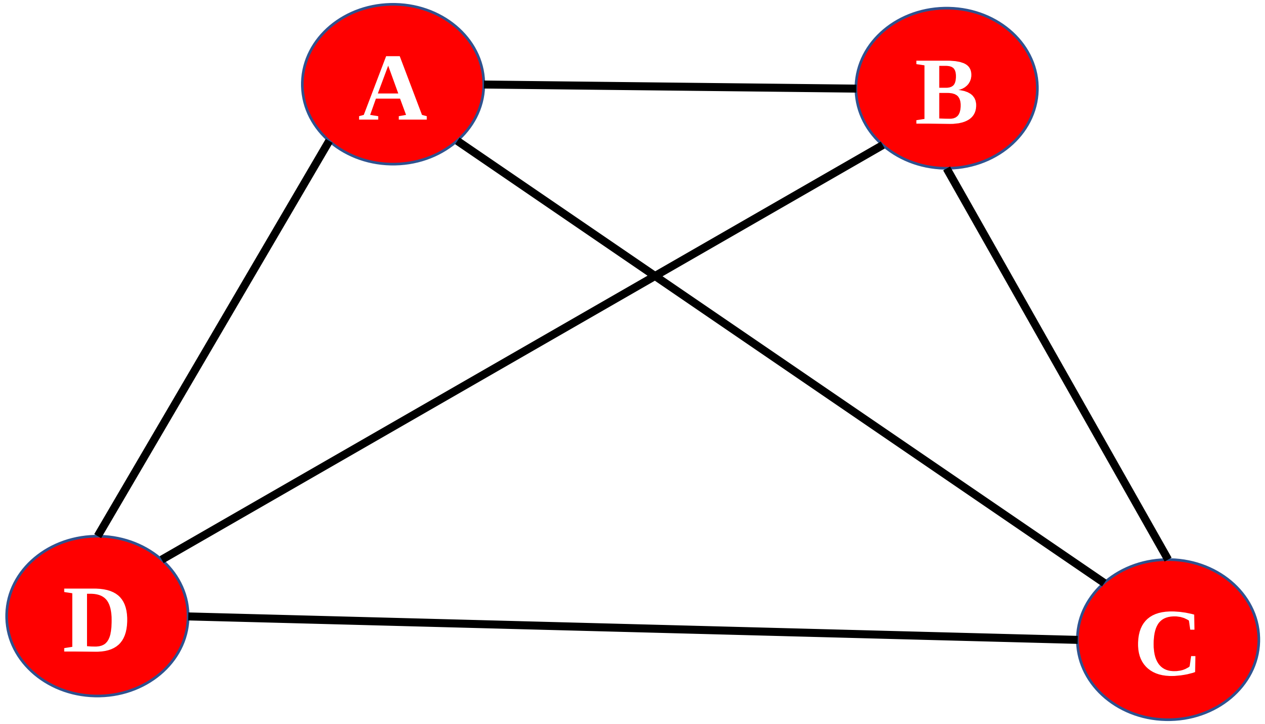
$$n(ST) = V^{(V-2)}$$

(Cayley's Formula)



In this graph each node is connected to every other node making a cycle

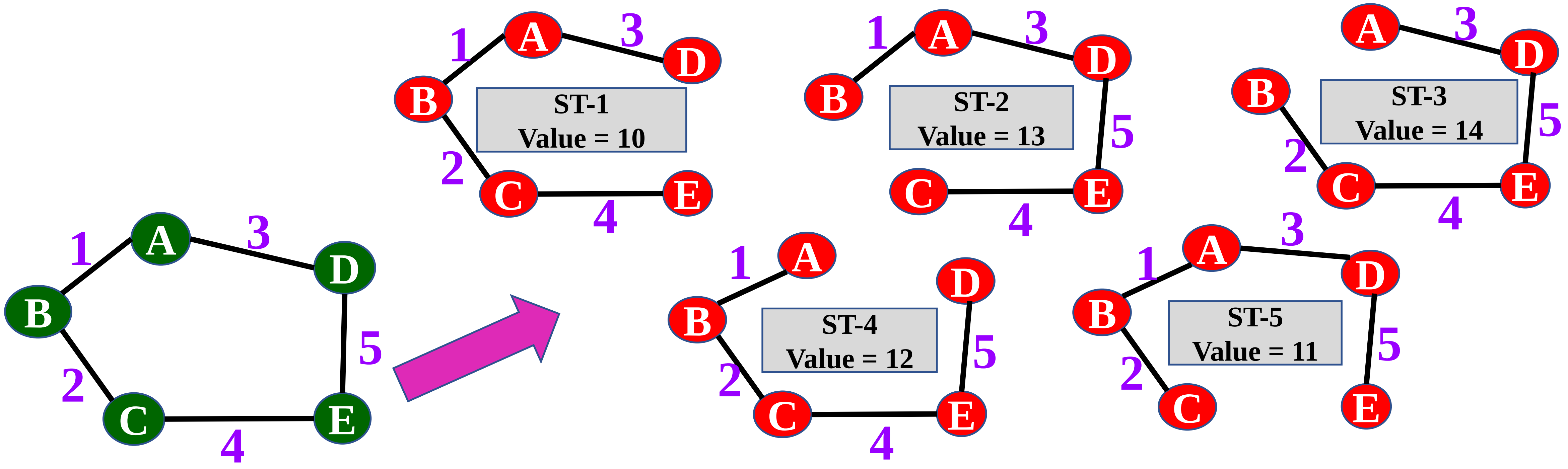
Complete Graph



$$n(ST) = 4^{(4-2)} = 16$$

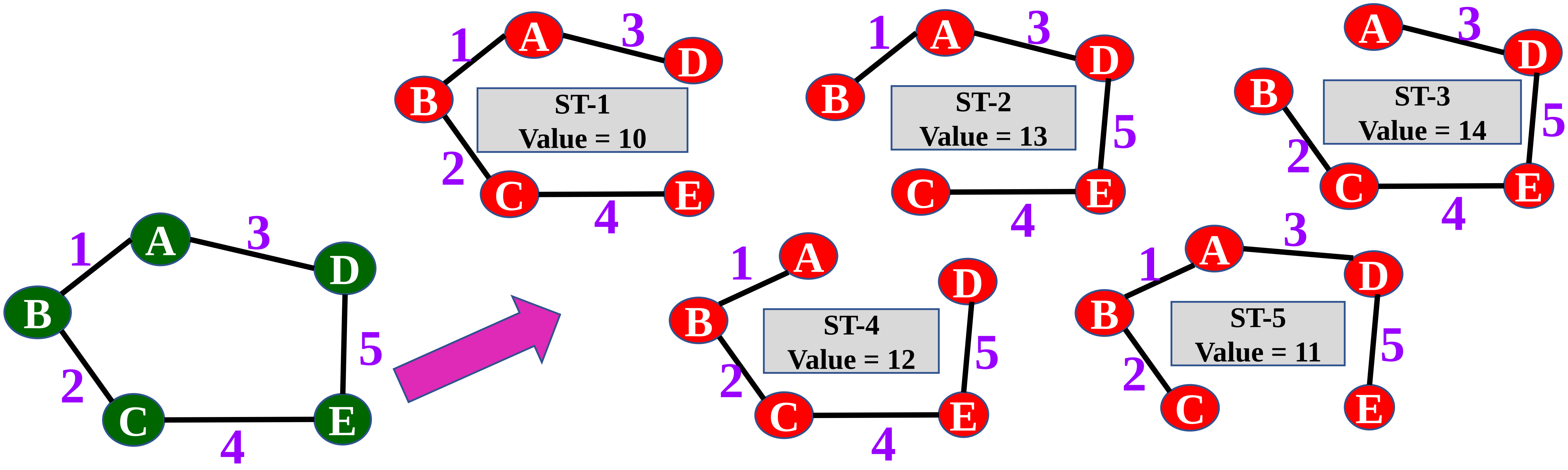
Properties of Spanning Tree

- A connected graph can contain more than one spanning tree



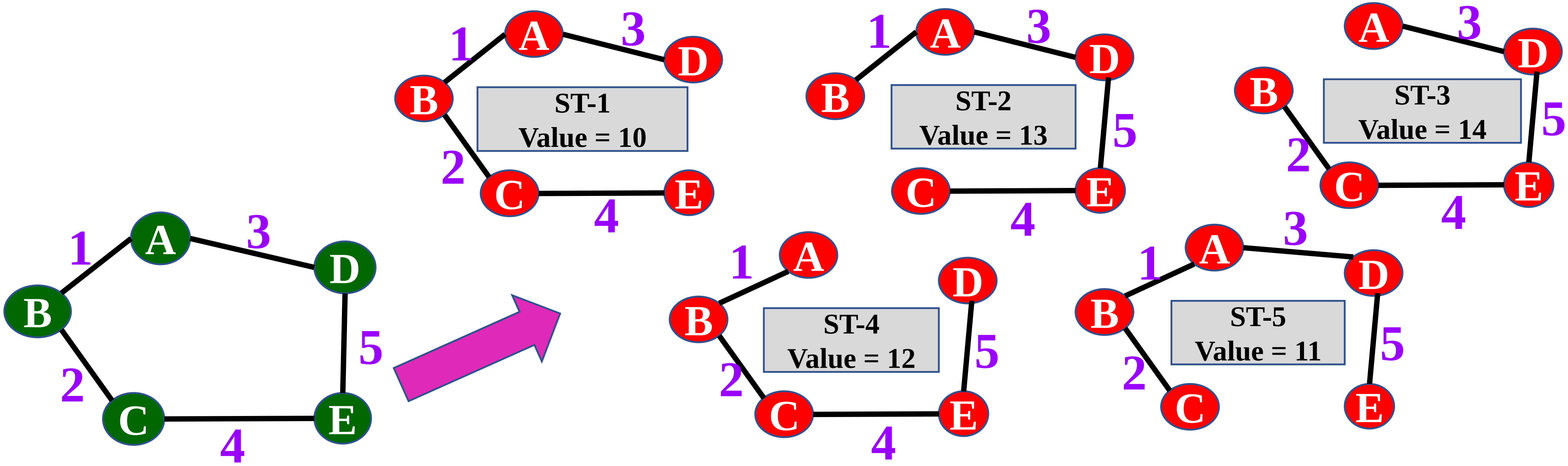
Properties of Spanning Tree

- All created spanning trees must contain same vertices equivalent to the graph and number of edges must be equal to |V| - 1



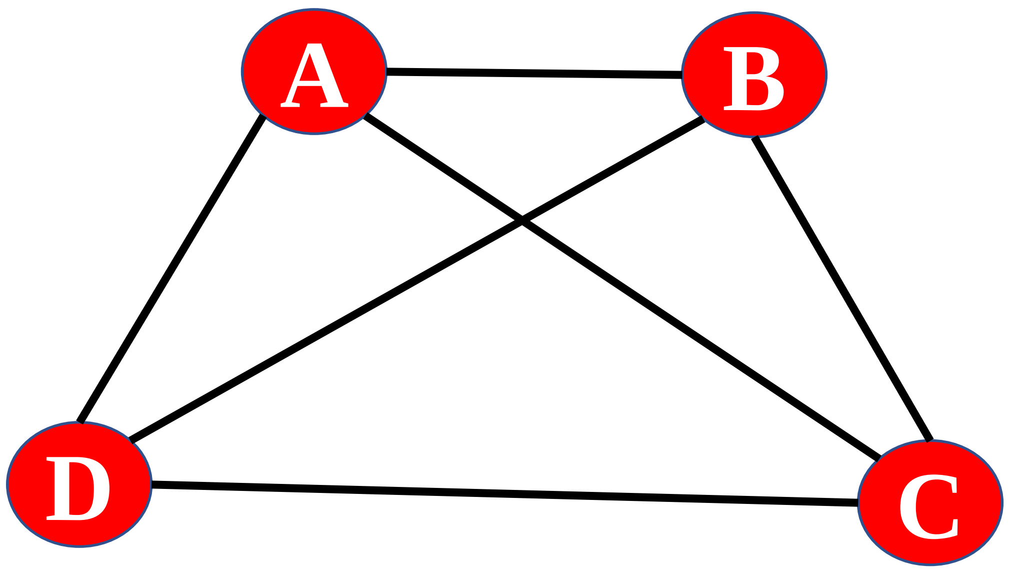
Properties of Spanning Tree

➢ Spanning tree must not contain any cycles



Properties of Spanning Tree

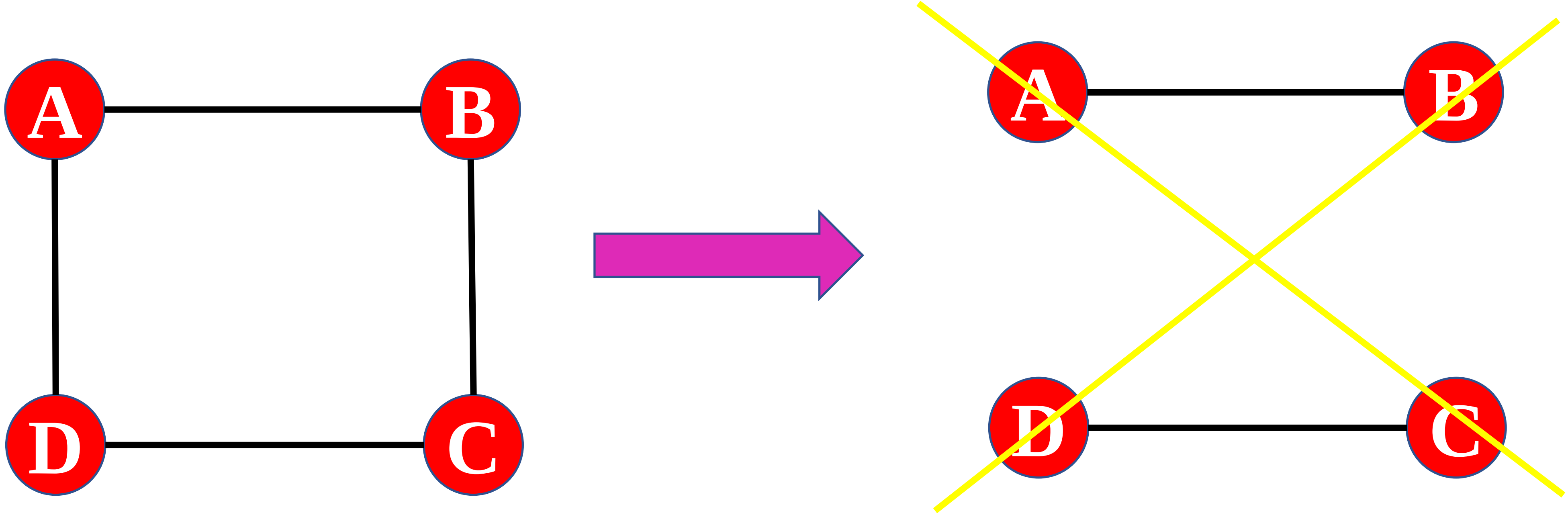
- If the given graph is complete graph then number of possible spanning trees will be $V^{(V-2)}$



$$n(ST) = 4^{(4-2)} = 16$$

Properties of Spanning Tree

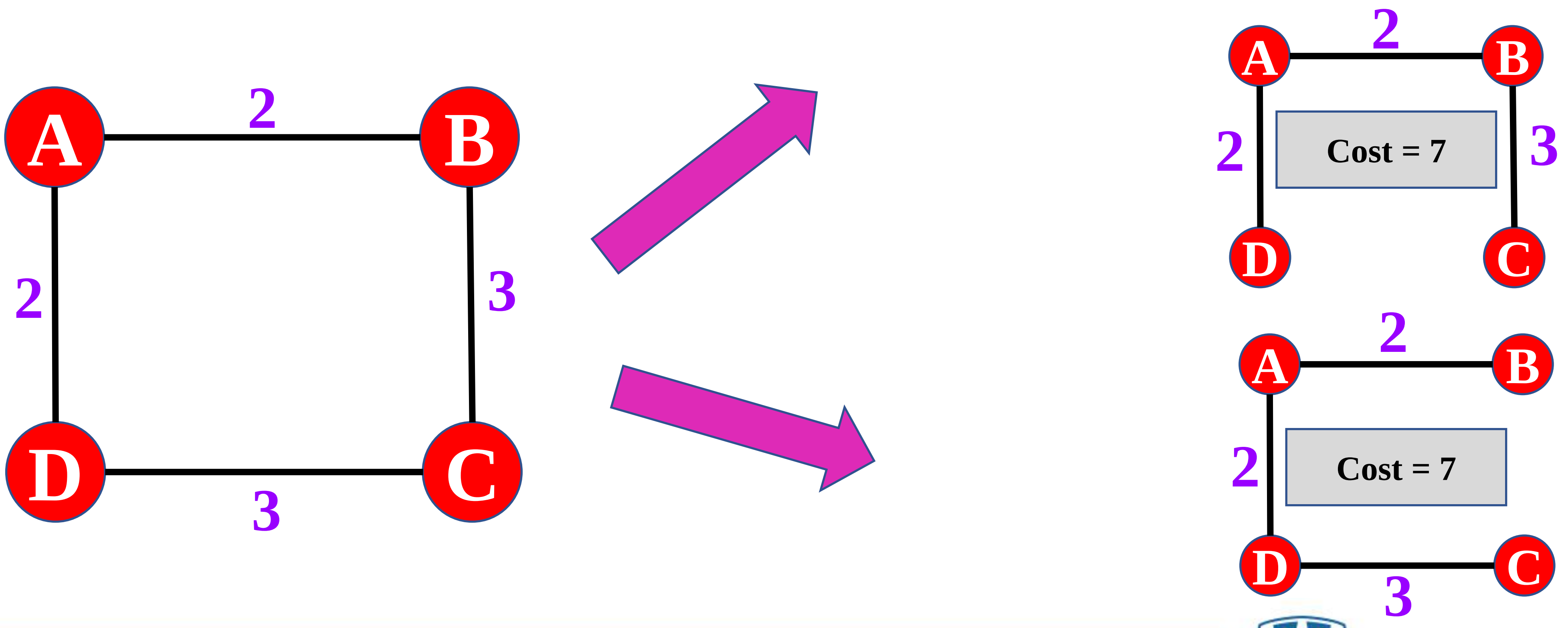
- Spanning trees **cannot** be disconnected



University of Windsor

Properties of Spanning Tree

- If there are multiple edges with same weight then there is possibility of having more than one minimum spanning tree



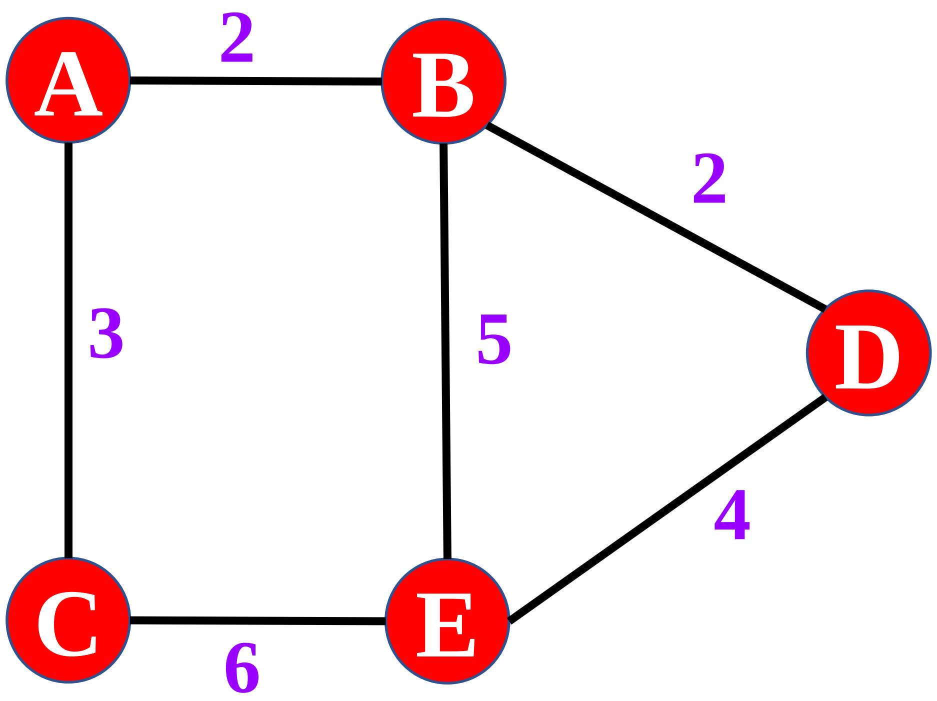
Prim’s Algorithm

Works with connected graphs

Prim’s algorithm is a **greedy** algorithm to find a minimal spanning tree for a weighted undirected graph

Finding MST with Prim's Algorithm

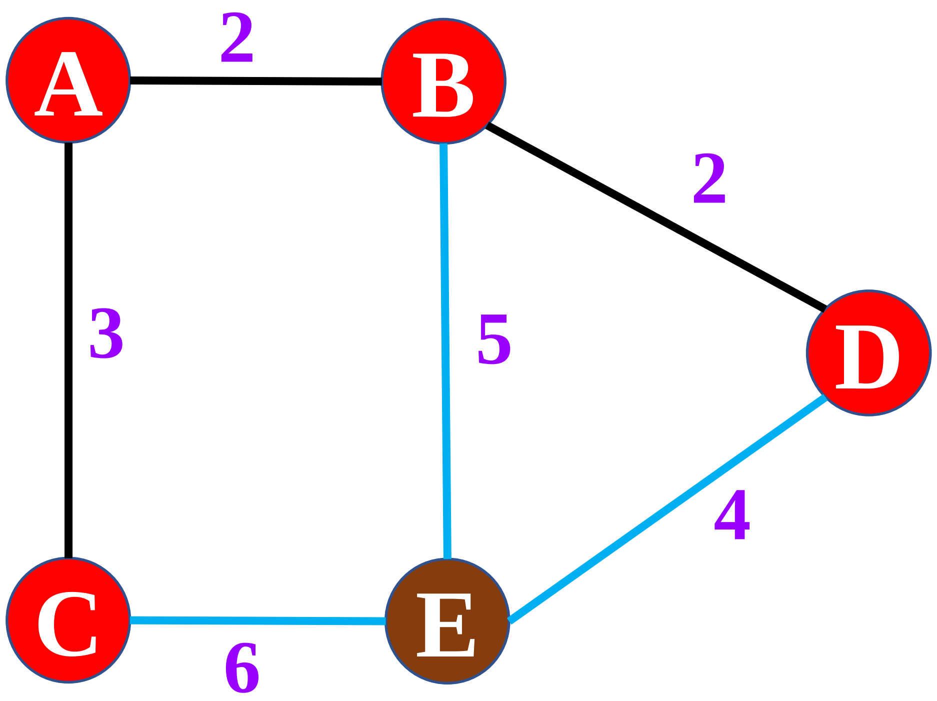
We need to find Minimum Spanning Tree G'(V', E') for graph G(V, E) such that the sum of edge weights is minimized.



Start anywhere, pick a node at random

Finding MST with Prim's Algorithm

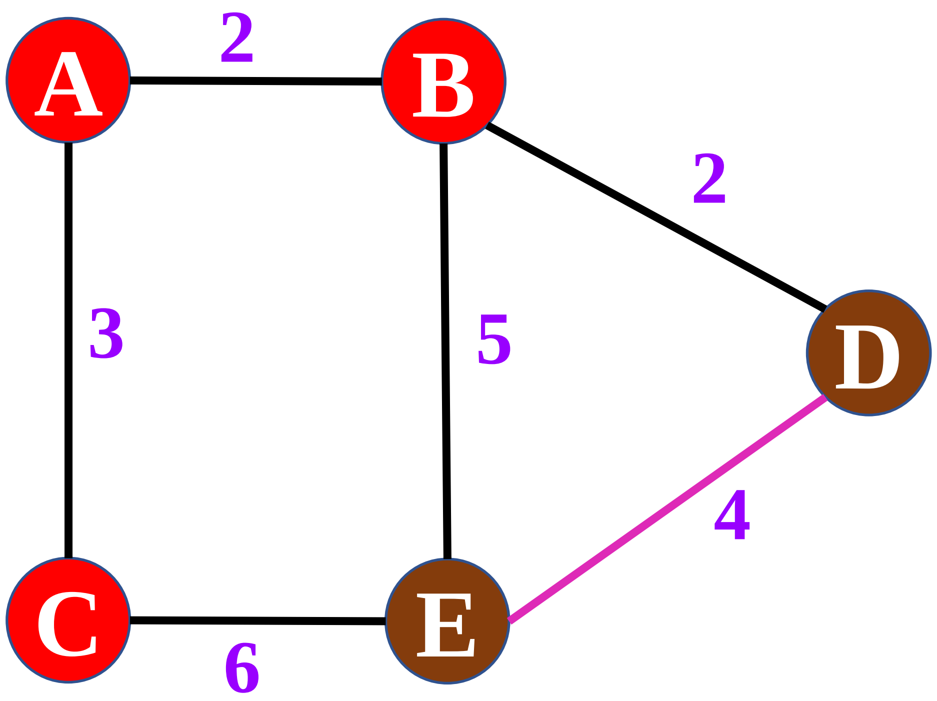
We need to find Minimum Spanning Tree G'(V', E') for graph G(V, E) such that the sum of edge weights is minimized.



Find the lowest weight edge out of that node

Finding MST with Prim’s Algorithm

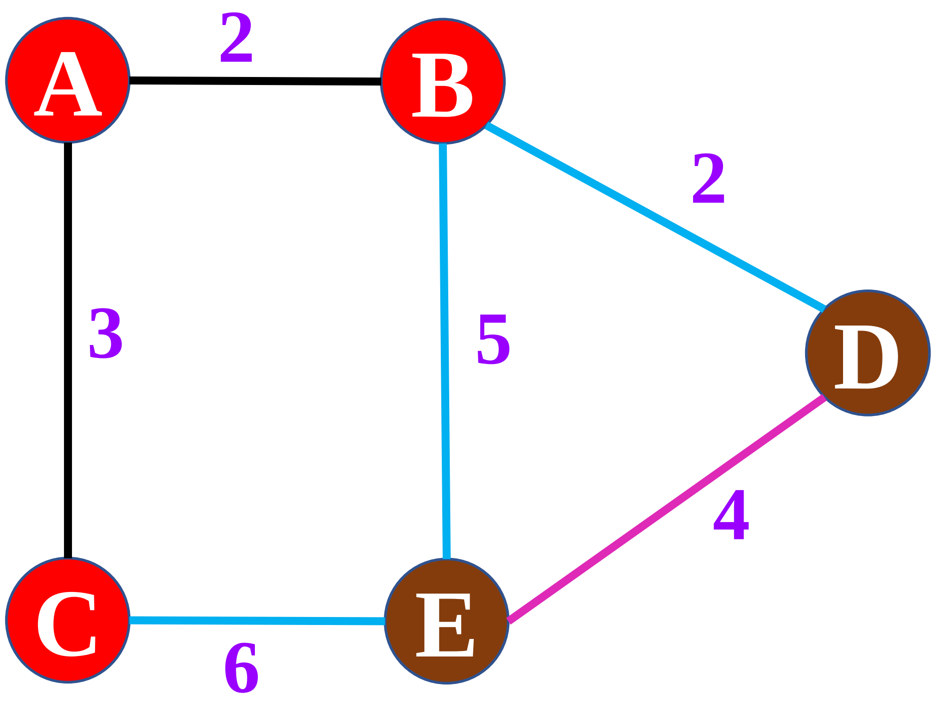
We need to find Minimum Spanning Tree G’(V’, E’) for graph G(V, E) such that the sum of edge weights is minimized.



Add that edge to the result

Finding MST with Prim's Algorithm

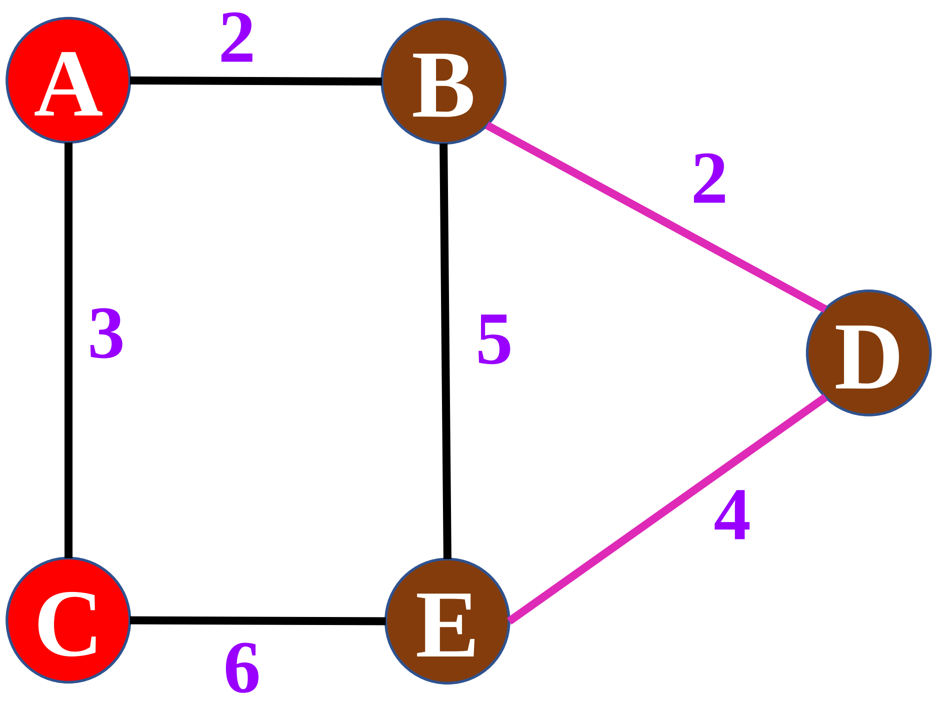
We need to find Minimum Spanning Tree G'(V', E') for graph G(V, E) such that the sum of edge weights is minimized.



Now find the lowest weight edge out of either node

Finding MST with Prim's Algorithm

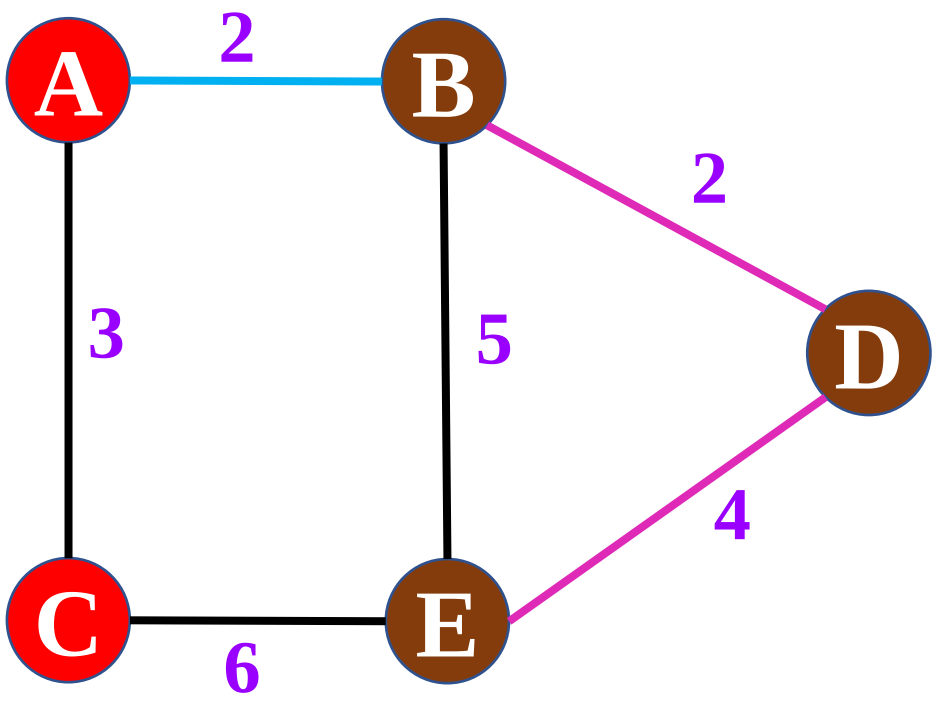
We need to find Minimum Spanning Tree G'(V', E') for graph G(V, E) such that the sum of edge weights is minimized.



Add that edge to result as well

Finding MST with Prim's Algorithm

We need to find Minimum Spanning Tree G'(V', E') for graph G(V, E) such that the sum of edge weights is minimized.

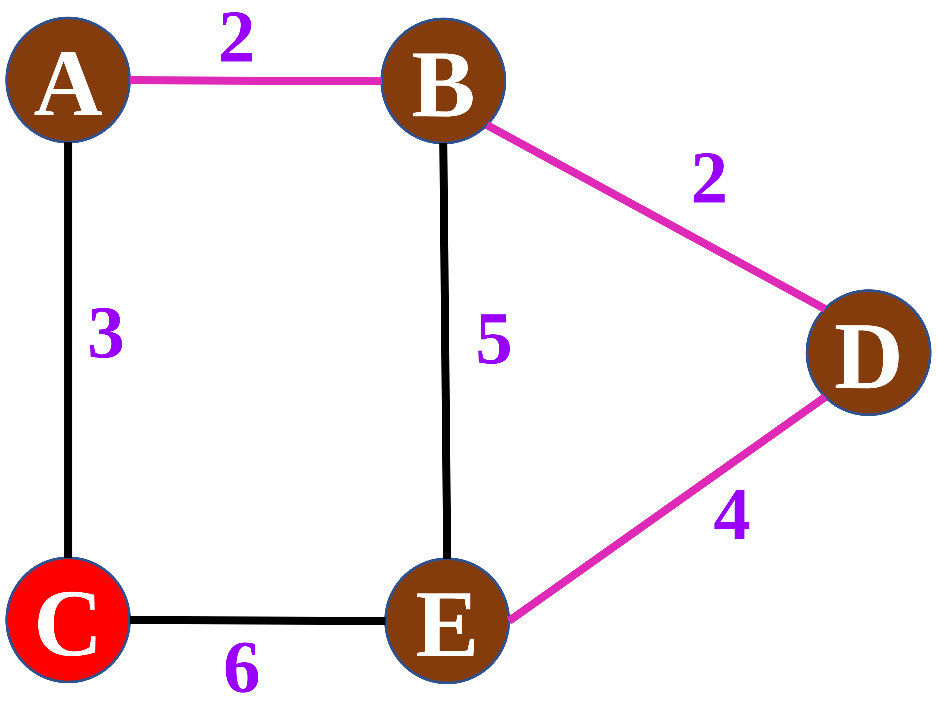


Once again, find lowest weight edge out of result set

Note that the edge B - E does not even count

Finding MST with Prim's Algorithm

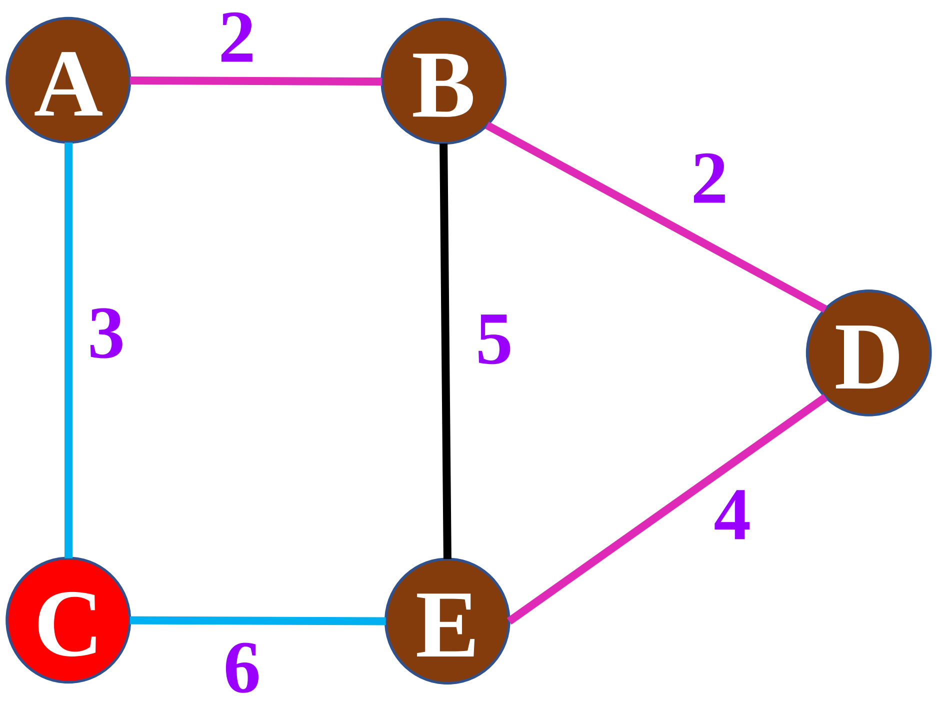
We need to find Minimum Spanning Tree G'(V', E') for graph G(V, E) such that the sum of edge weights is minimized.



Add that edge to result set

Finding MST with Prim's Algorithm

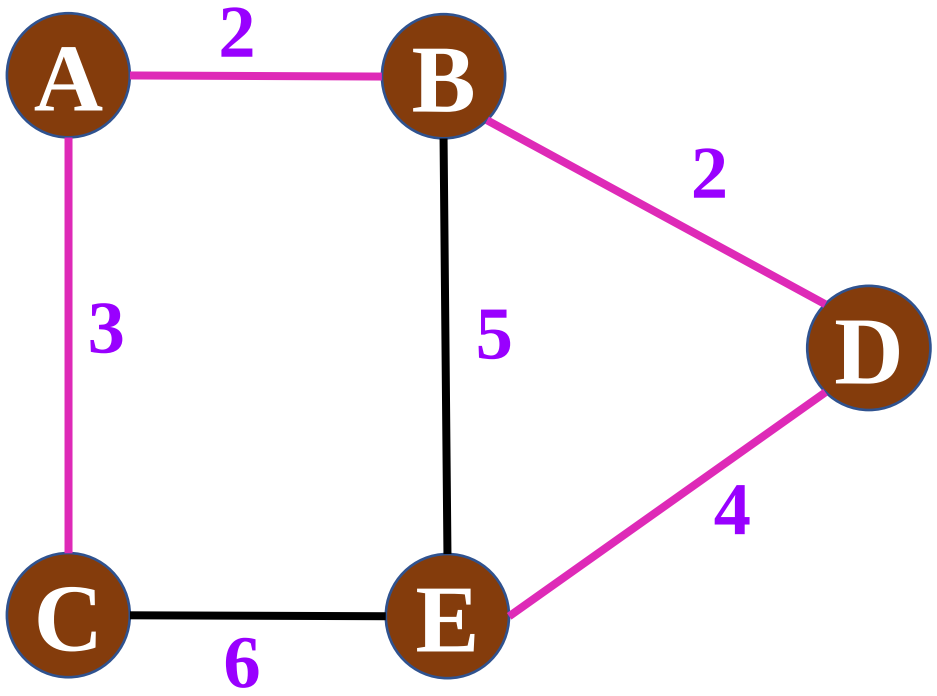
We need to find Minimum Spanning Tree G'(V', E') for graph G(V, E) such that the sum of edge weights is minimized.



Once again, find lowest weight edge out of result set

Finding MST with Prim's Algorithm

We need to find Minimum Spanning Tree G'(V', E') for graph G(V, E) such that the sum of edge weights is minimized.



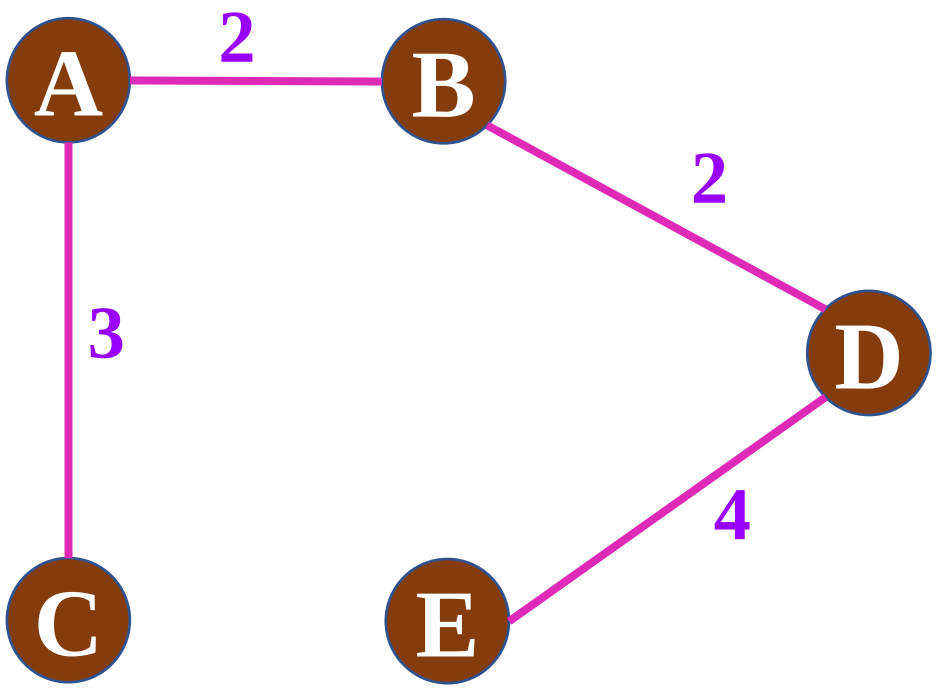
Add that edge to result set

All vertices in spanning tree, stop

Minimum spanning tree found

Finding MST with Prim's Algorithm

We need to find Minimum Spanning Tree G'(V', E') for graph G(V, E) such that the sum of edge weights is minimized.



Sum weights of edges in result set = 11

Prim’s Algorithm

- Algorithm considers edges in contiguous order
- **Benefit:** Intermediate result is a tree as well
- **Drawback:** Does not work for disconnected graphs

Prim's Algorithm
(Priority Queue)

- Binary Heap

  Running time: $O(E \ln(V))$
- Array

  Running time: $O(E + V^2)$

Kruskal’s Algorithm

Works even with disconnected graphs

Kruskal’s algorithm is a greedy algorithm to find a minimal spanning tree for a weighted undirected graph

The graph can be unconnected

Kruskal's Algorithm

Sort edges

Increasing order of weights

Can use priority queue

Initialize empty result

Empty set of edges

At end will hold minimum spanning tree

Find shortest edge

Not currently in result

Dequeue from priority queue

Reject if cycle introduced

Else add to result set

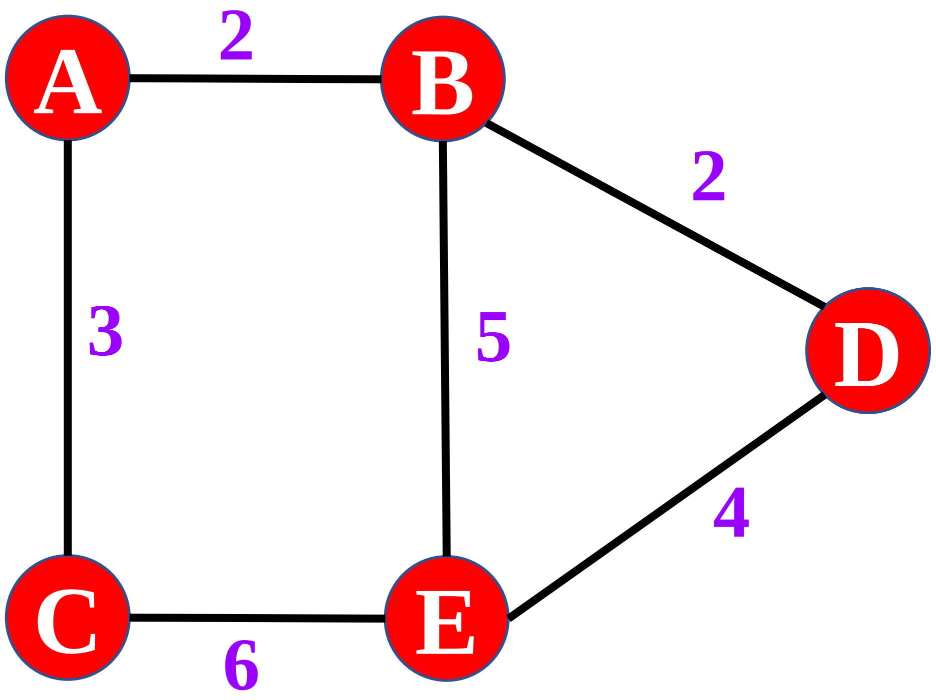
This is a greedy step

Stop

When N-1 edges in result

N = number of vertices in graph

Kruskal's Algorithm

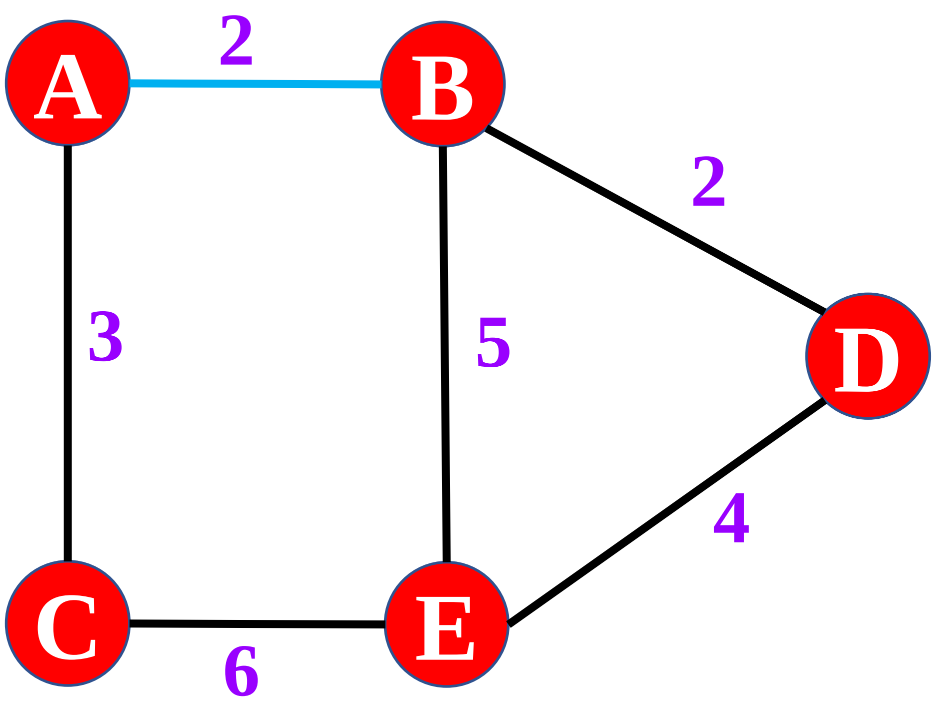


Step 1: Sort edges
Increasing order of weights
Can use priority queue

Edge	Weight

Priority Queue

Kruskal's Algorithm

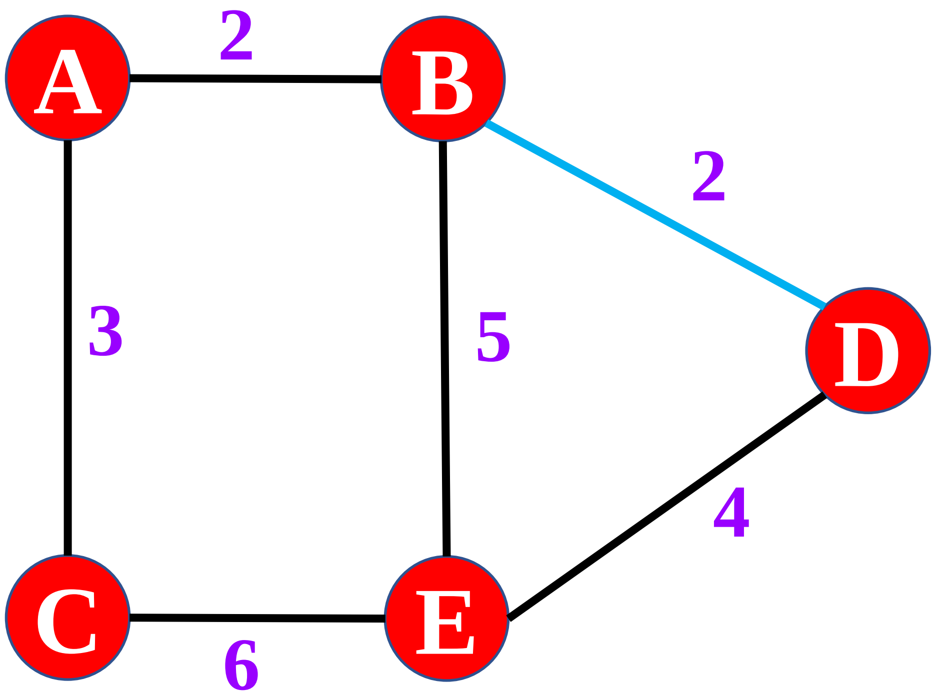


Step 1: Sort edges
Increasing order of weights
Can use priority queue

Edge	Weight
A – B	2

Priority Queue

Kruskal's Algorithm

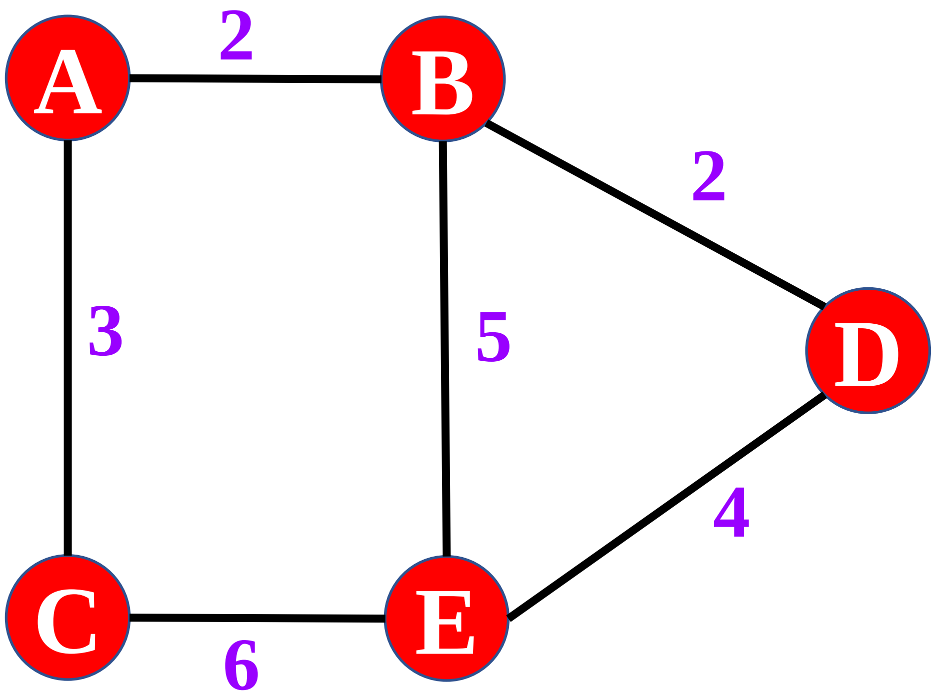


Step 1: Sort edges
Increasing order of weights
Can use priority queue

Edge	Weight
A – B	2
B – D	2

Priority Queue

Kruskal's Algorithm



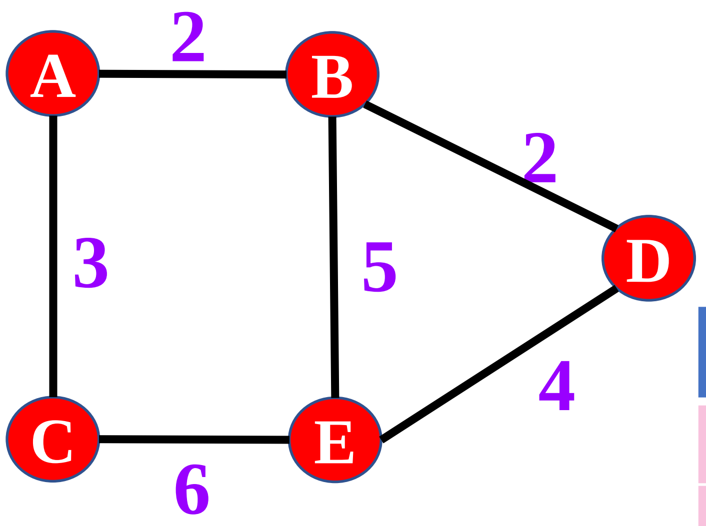
Step 1: Sort edges
Increasing order of weights
Can use priority queue

Edge	Weight
A – B	2
B – D	2
A – C	3
E – D	4
B – E	5
C – E	6

Priority Queue

Kruskal's Algorithm

Step 2: Initialize empty result
Empty set of edges
At the end it will hold minimum spanning tree



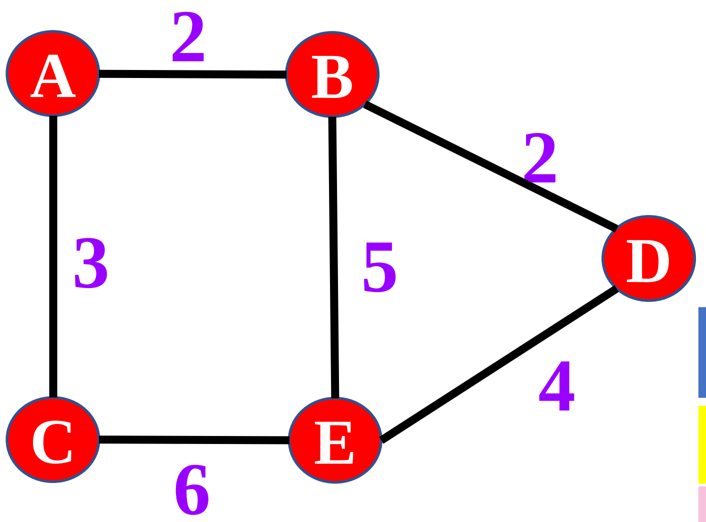
Edge	Weight
A – B	2
B – D	2
A – C	3
E – D	4
B – E	5
C – E	6

Priority Queue

Result

Edge	Weight

Kruskal's Algorithm



Step 3: Find the shortest edge
It should not be currently in result
Dequeue from priority queue

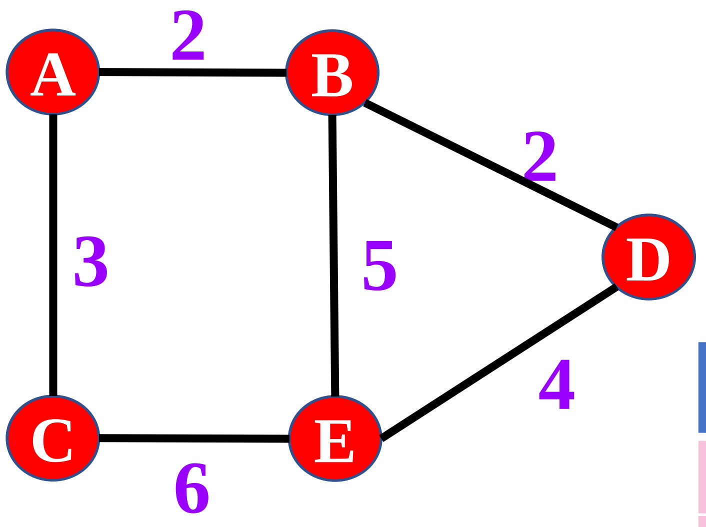
Edge	Weight
A – B	2
B – D	2
A – C	3
E – D	4
B – E	5
C – E	6

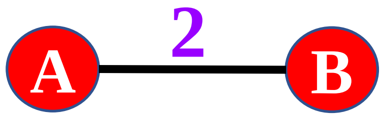
Priority Queue

Result

Edge	Weight

Kruskal's Algorithm





Step 3: Find the shortest edge
It should not be currently in result
Dequeue from priority queue

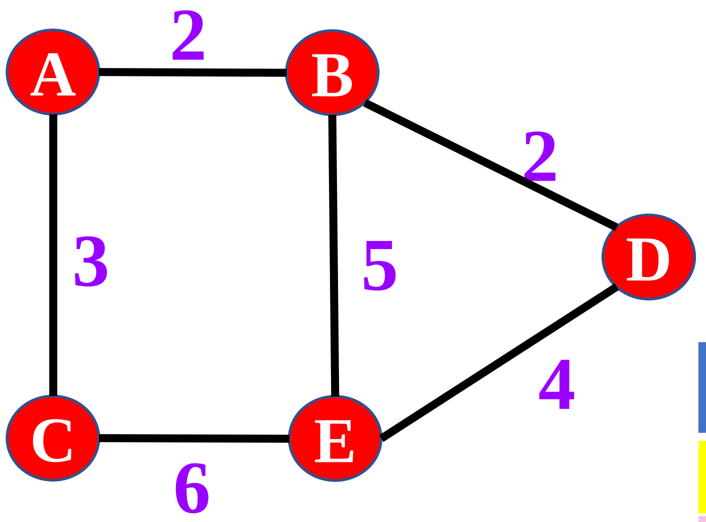
Edge	Weight
B – D	2
A – C	3
E – D	4
B – E	5
C – E	6

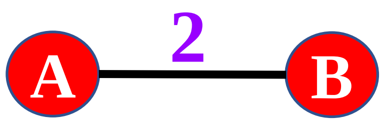
Priority Queue

Result

Edge	Weight
A – B	2

Kruskal's Algorithm





Step 3: Find the shortest edge
It should not be currently in result
Dequeue from priority queue

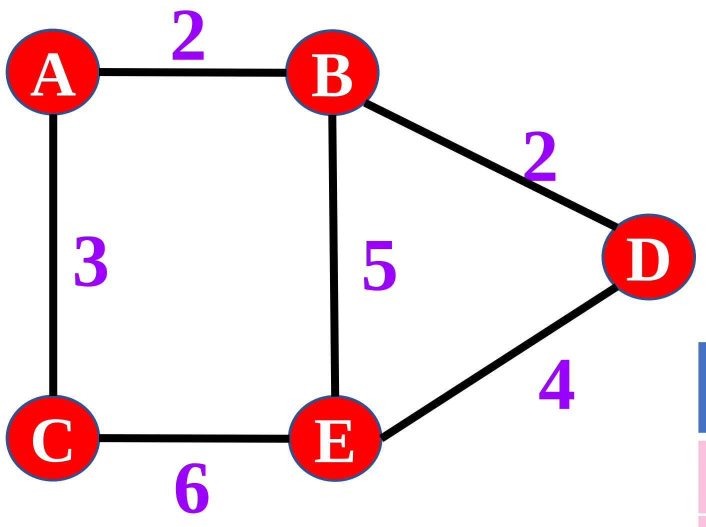
Edge	Weight
B – D	2
A – C	3
E – D	4
B – E	5
C – E	6

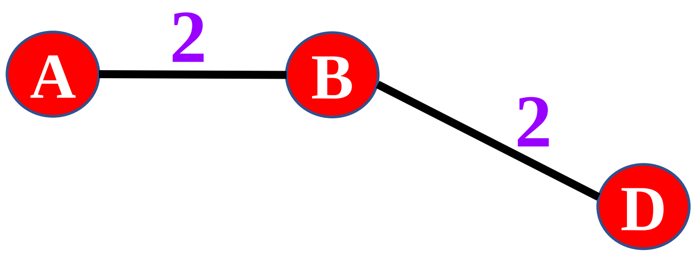
Priority Queue

Result

Edge	Weight
A – B	2

Kruskal's Algorithm





Step 3: Find the shortest edge
It should not be currently in result
Dequeue from priority queue

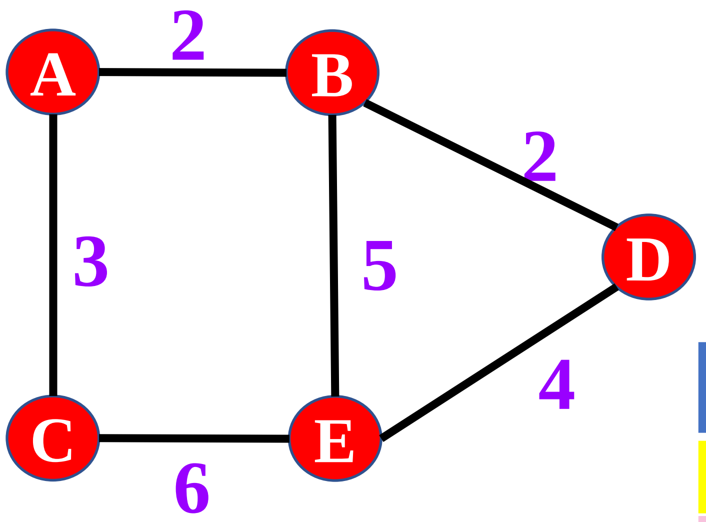
Edge	Weight
A – C	3
E – D	4
B – E	5
C – E	6

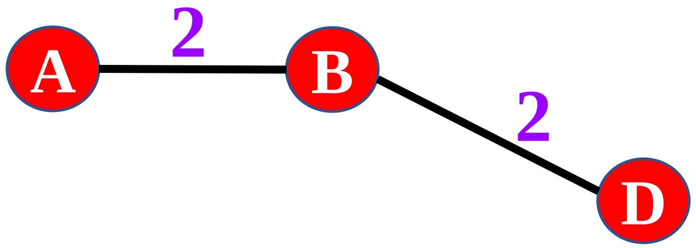
Priority Queue

Result

Edge	Weight
A – B	2
B – D	2

Kruskal's Algorithm





Step 3: Find the shortest edge
It should not be currently in result
Dequeue from priority queue

Edge	Weight
A – C	3
E – D	4
B – E	5
C – E	6

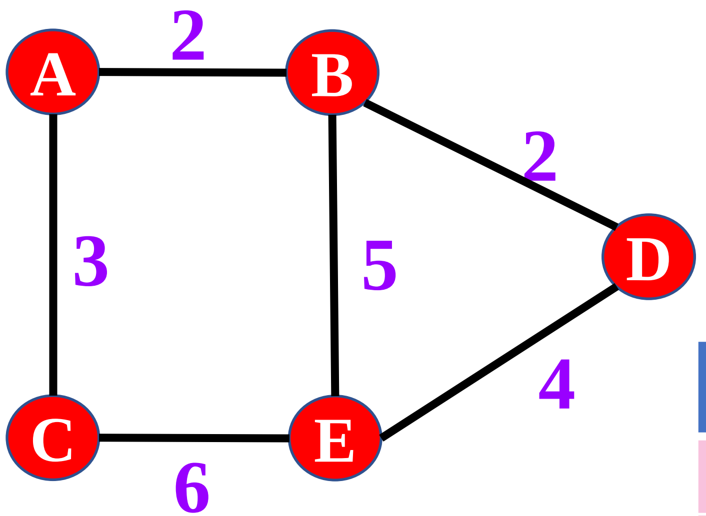
Priority Queue

Result

Edge	Weight
A – B	2
B – D	2

Kruskal's Algorithm

Step 3: Find the shortest edge
It should not be currently in result
Dequeue from priority queue

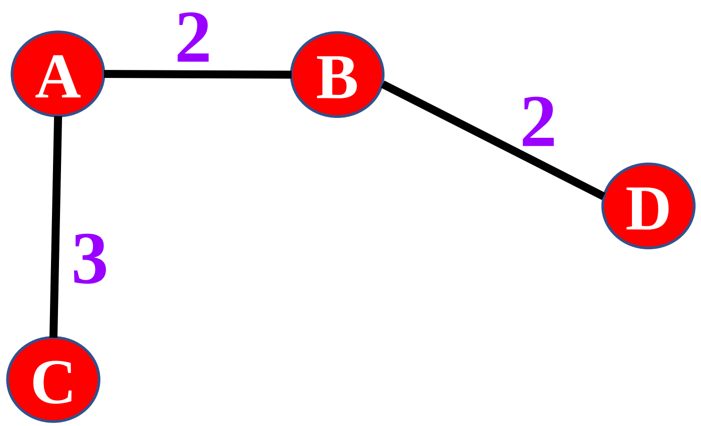


Edge	Weight
E – D	4
B – E	5
C – E	6

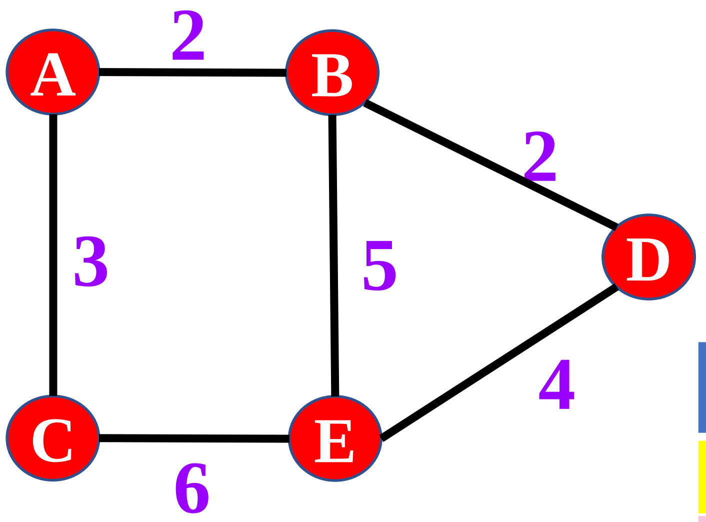
Priority Queue

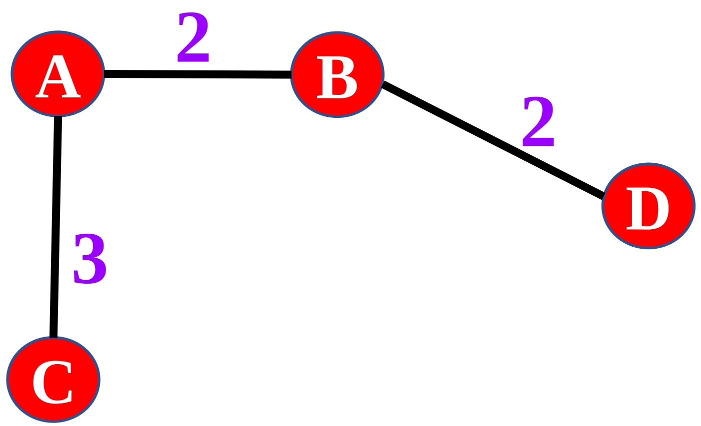
Result

Edge	Weight
A – B	2
B – D	2
A – C	3



Kruskal's Algorithm





Step 3: Find the shortest edge
It should not be currently in result
Dequeue from priority queue

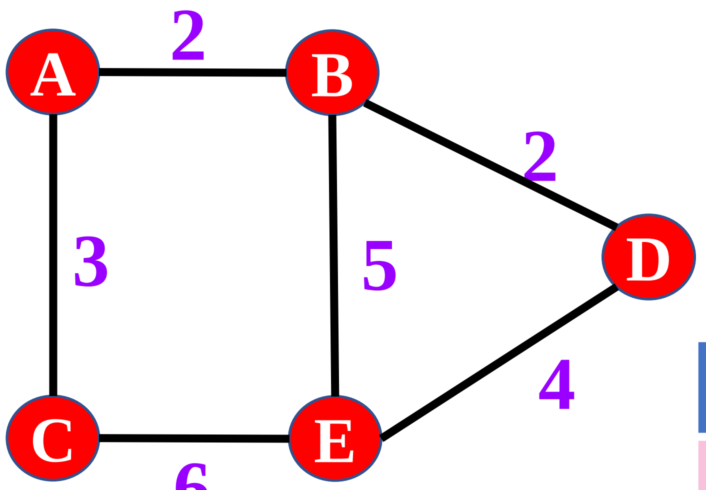
Edge	Weight
E – D	4
B – E	5
C – E	6

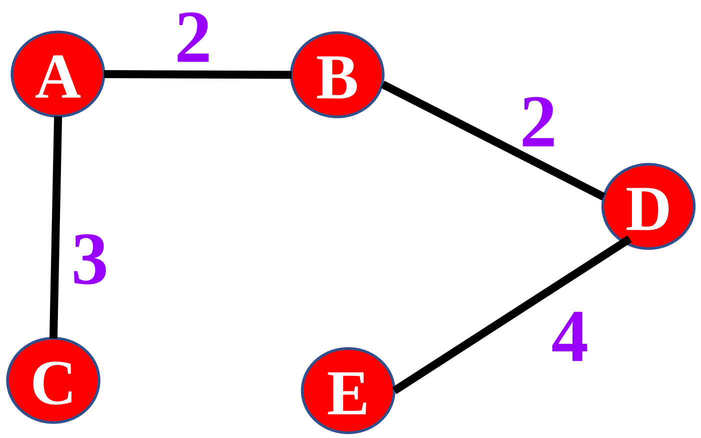
Priority Queue

Result

Edge	Weight
A – B	2
B – D	2
A – C	3

Kruskal's Algorithm





Step 3: Find the shortest edge
It should not be currently in result
Dequeue from priority queue

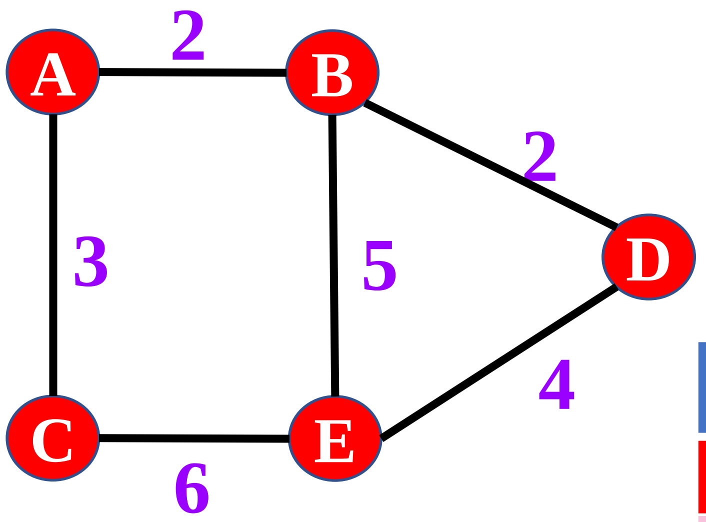
Edge	Weight
B – E	5
C – E	6

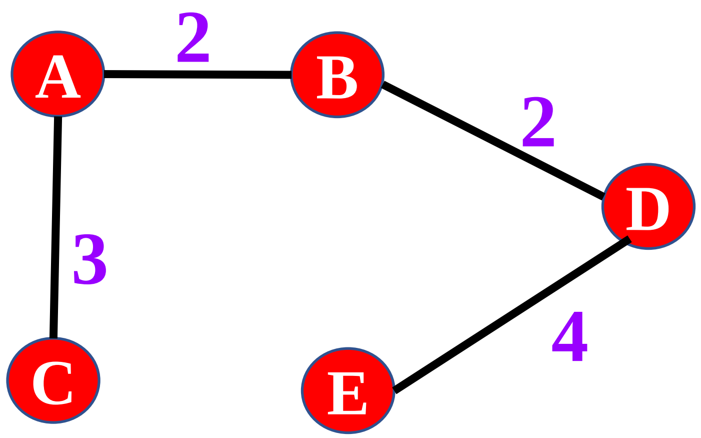
Priority Queue

Result

Edge	Weight
A – B	2
B – D	2
A – C	3
D – E	4

Kruskal's Algorithm





Step 3: Find the shortest edge
It should not be currently in result
Dequeue from priority queue

Edge	Weight
B – E	5
C – E	6

Priority Queue

Reject if cycle introduced
Else add to result set
This is a greedy step

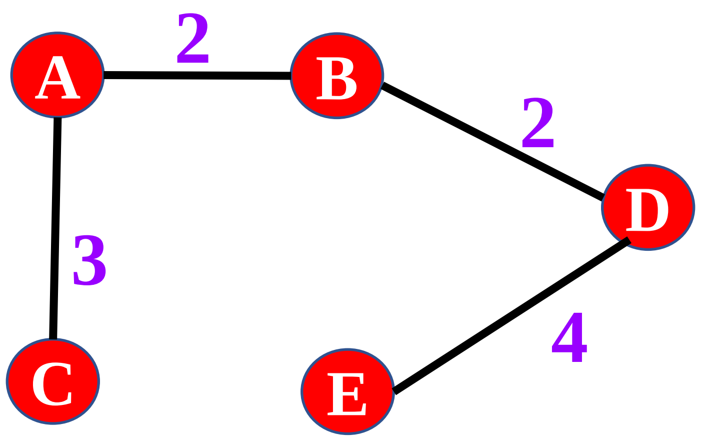
Result

Edge	Weight
A – B	2
B – D	2
A – C	3
D – E	4

Stop
When N-1 edges in result
N = number of vertices in graph

Kruskal's Algorithm

Minimum spanning tree found, weight = 11



Result

Edge	Weight
A – B	2
B – D	2
A – C	3
D – E	4

University of Windsor

Kruskal's Algorithm

- Algorithm does not consider edges in contiguous order
- **Benefit:** Works for disconnected graphs too
- **Drawback:** Intermediate result is not necessarily a tree

Kruskal's Algorithm

Sorting the edges dominates the running time

$O\ (E \ln(E))$

Spanning Tree Applications

Scenario 1

The office network went down due to network looping. Now Paul and his team need to do something in order to resolve this problem permanently.

Any ideas?

Scenario 1: Solution

His teammate who was good at data structures suggested that Paul’s team should implement Spanning Tree data structure to manage routing connections

Scenario 1: Solution

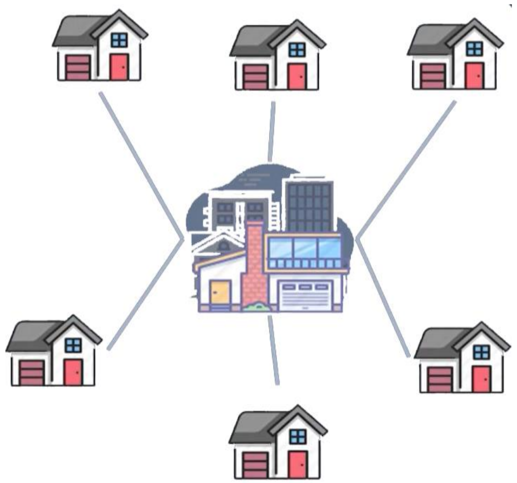
While implementing spanning tree the whole network was treated as a graph. The routers and end devices were treated as vertices whereas the wires connecting them were treated as the edges of the graph.

University of Windsor

Scenario 1: Solution

Using spanning tree algorithms Paul and his team managed to develop a routing spanning tree with minimal cost, accurate packet transfer and high speed.

Scenario 2



Suppose the government of Ontario has been given a major grant to install a large Wi-Fi network for each of its cities. The main idea is that communication cables can run from the main Internet access point to a city tower and cables can also run between pairs of towers. However, the challenge is to interconnect all the towers and the Internet access point as cheaply as possible.

How can we do this?

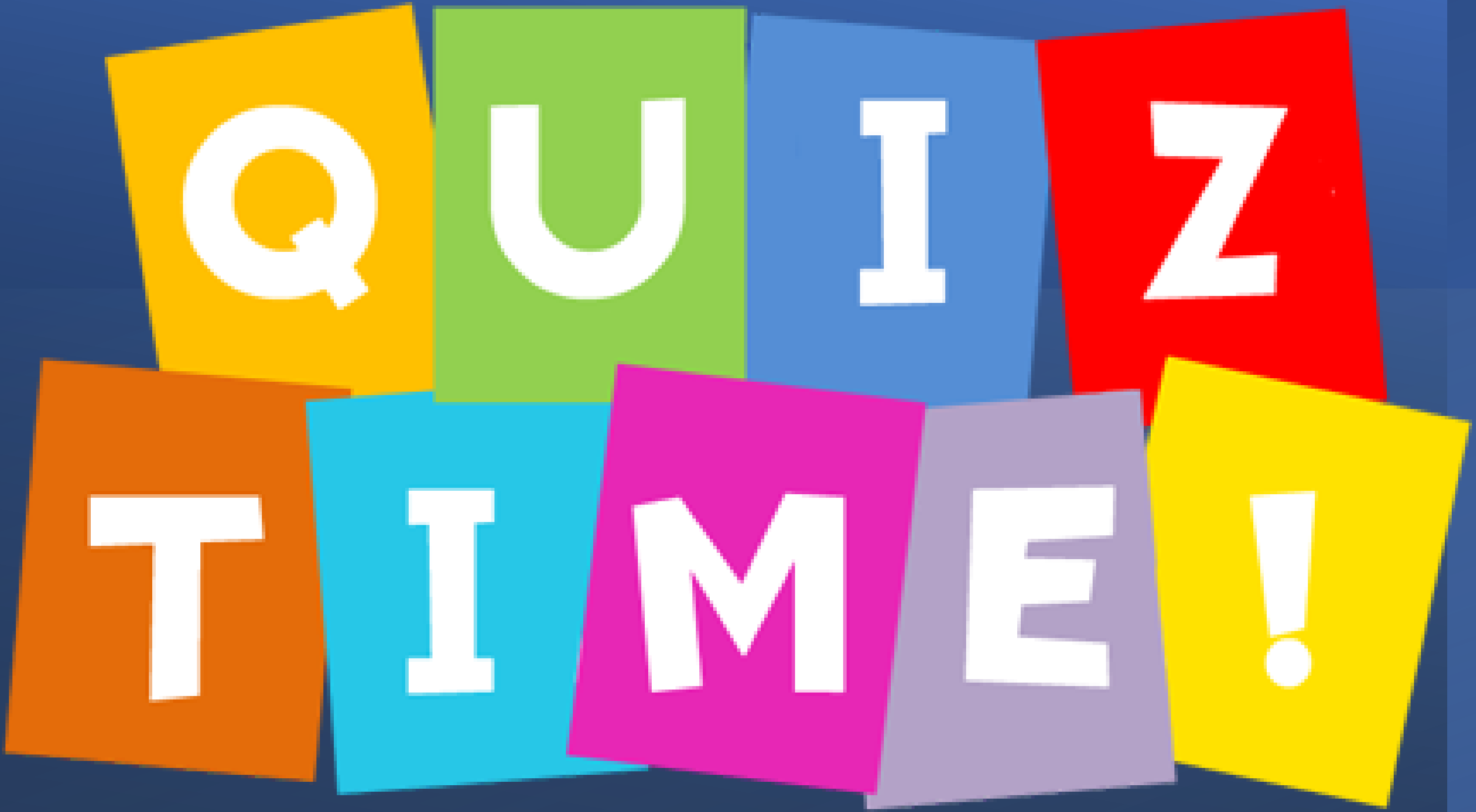
Scenario 2 (Solution)

We can model this problem using a graph, G, where each vertex in G is the location of a Wi-Fi (the Internet access point), and an edge in G is a possible cable we could run between two such vertices. Each edge in G could then be given a weight that is equal to the cost of running the cable that that edge represents. Thus, we are interested in finding a connected acyclic subgraph of G that includes all the vertices of G and has minimum total cost. Using the language of graph theory, we are interested in finding a minimum spanning tree (MST) of G.

University of Windsor

Summary

- Spanning tree algorithms seek to find the shortest way to cover all nodes
- Such algorithms are used when start and end nodes do not matter
- Prim’s algorithm works for connected graphs
- Kruskal’s algorithm works even for disconnected graphs



Kahoot.it