Fall, 2023

Discrete Mathematics

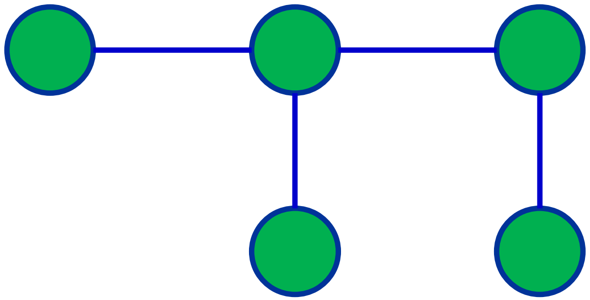
Lecture 12: Trees

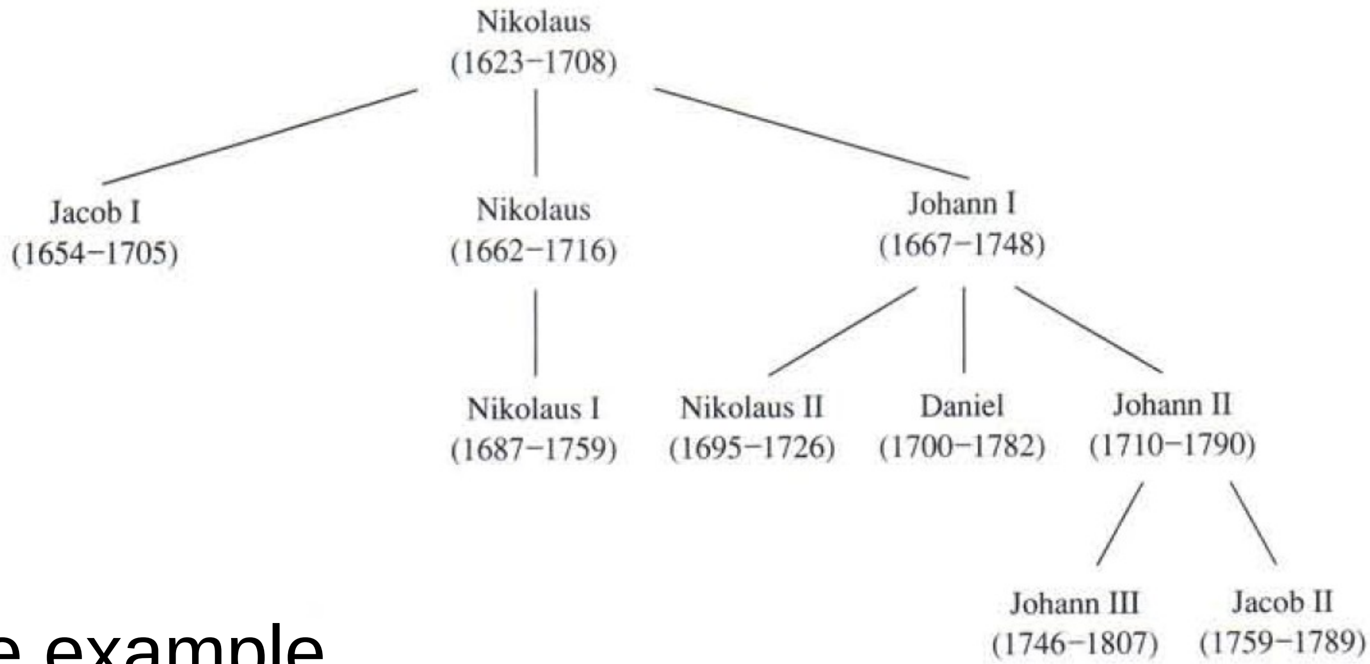
Ta Quang Hung

Contents

INTRODUCTION

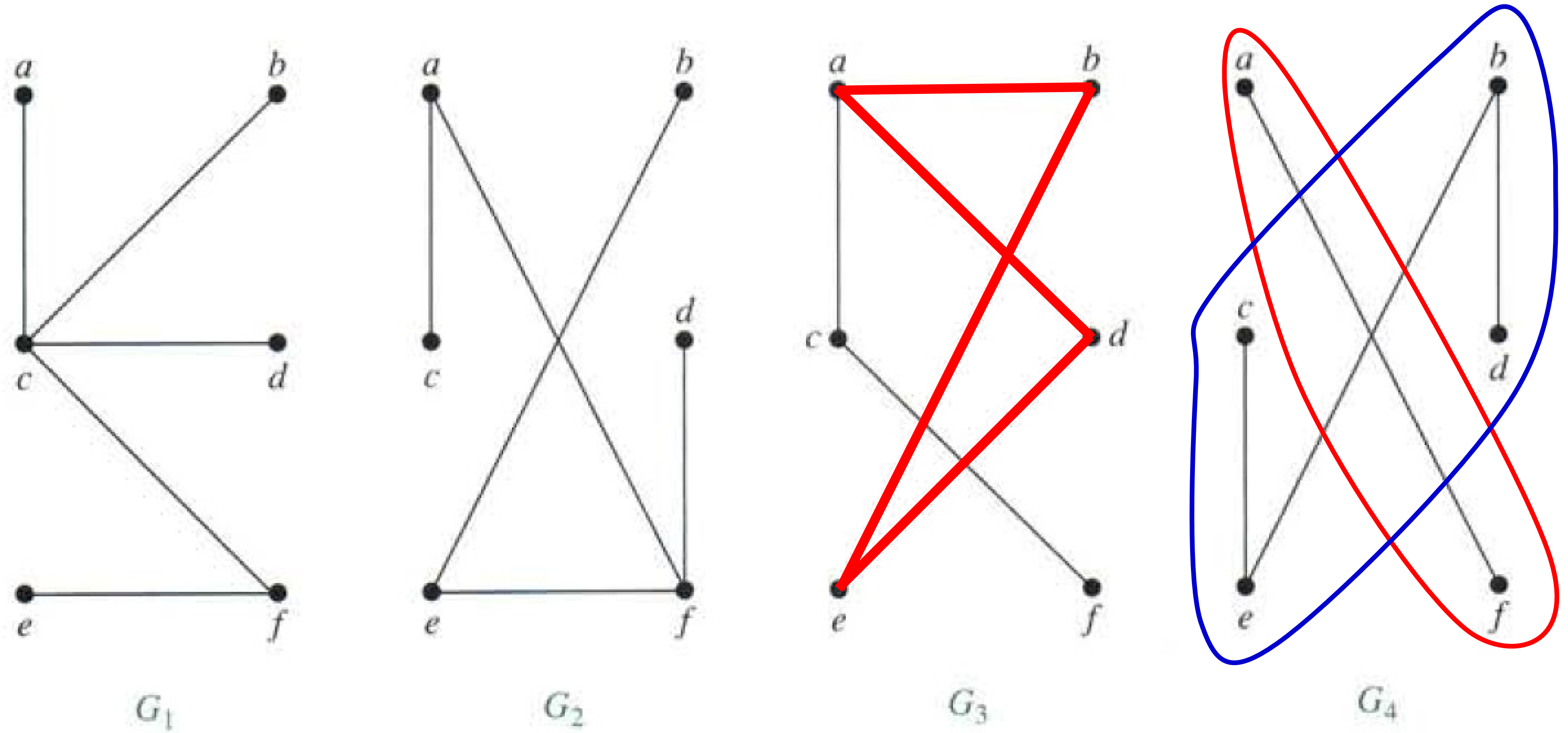
- A tree, is an undirected graph T such that
  - T is connected.
  - T has no cycles.





Tree example

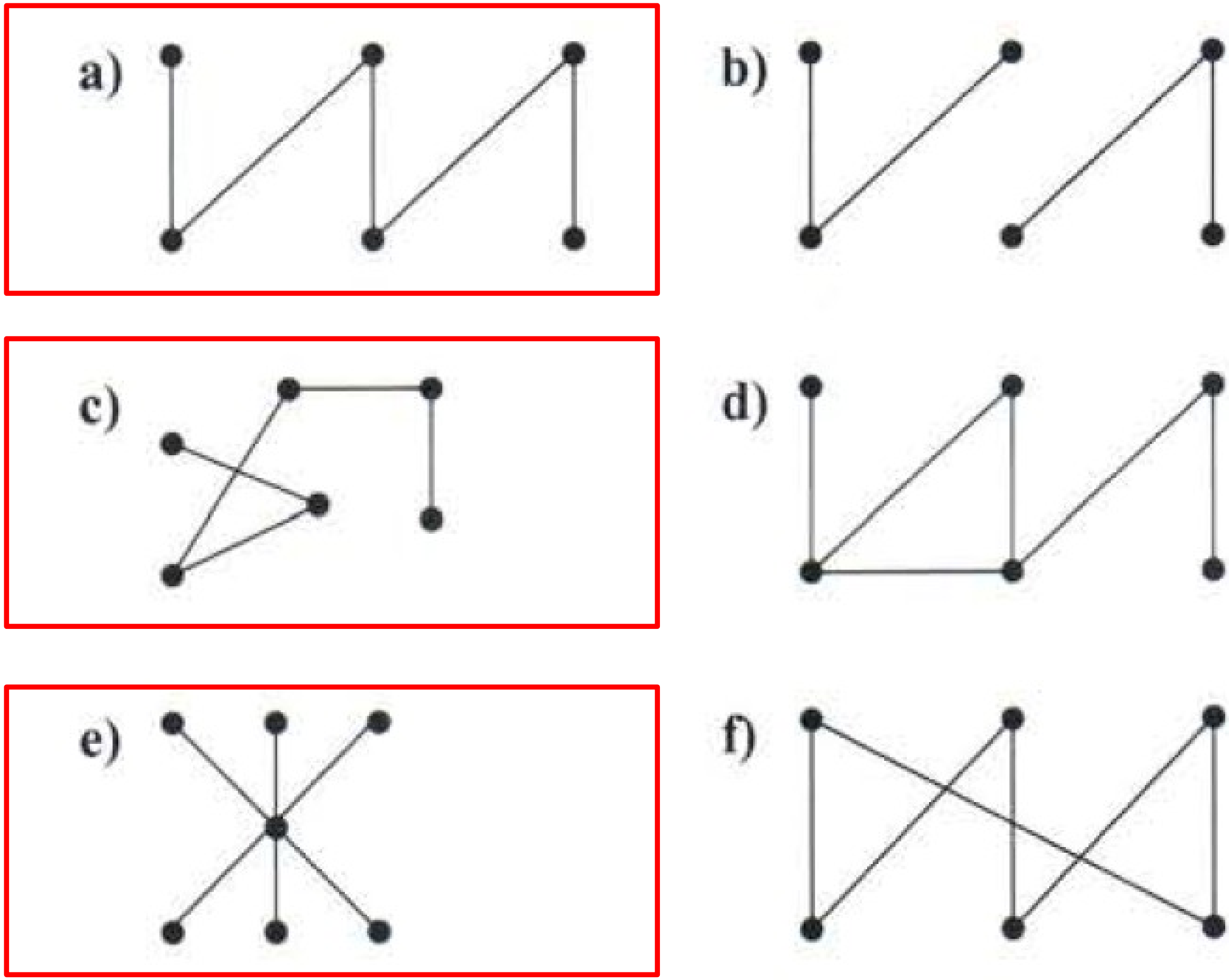
Trees



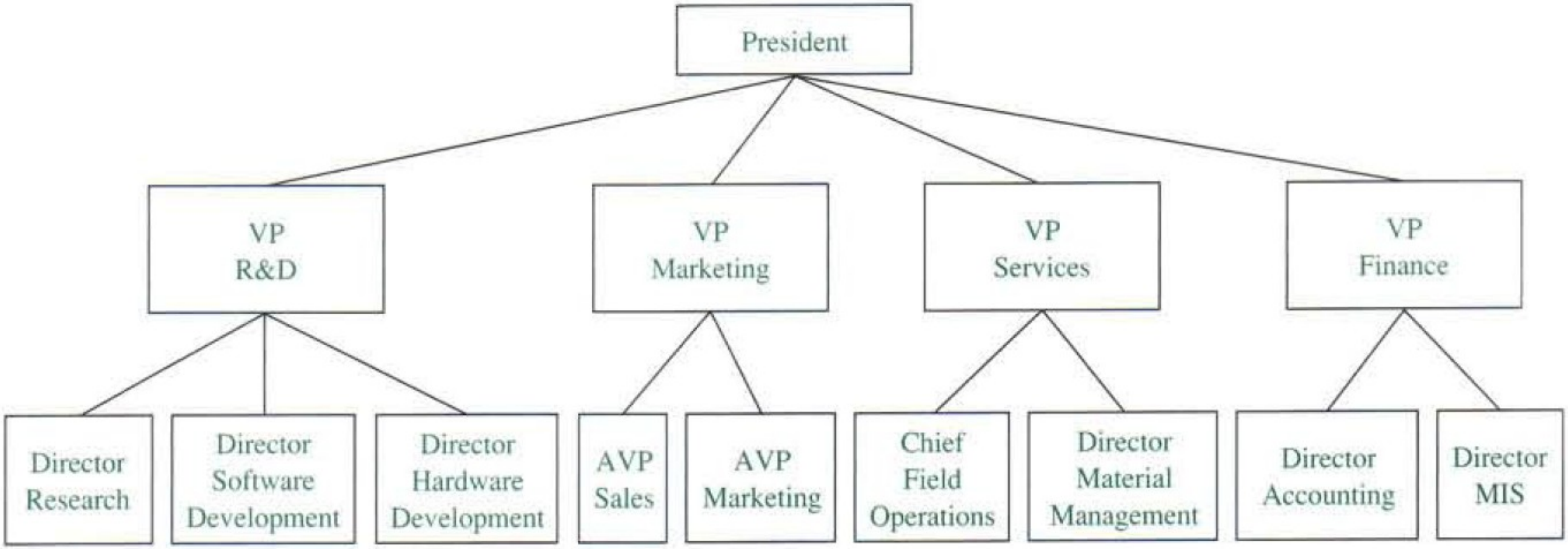
G1, G2 are trees because both G1 and G2 are connected graph without simple circuits

G3 is not a tree because e,b,a,d,e is a simple circuit in the graph

G4 is not a tree because it is not connected



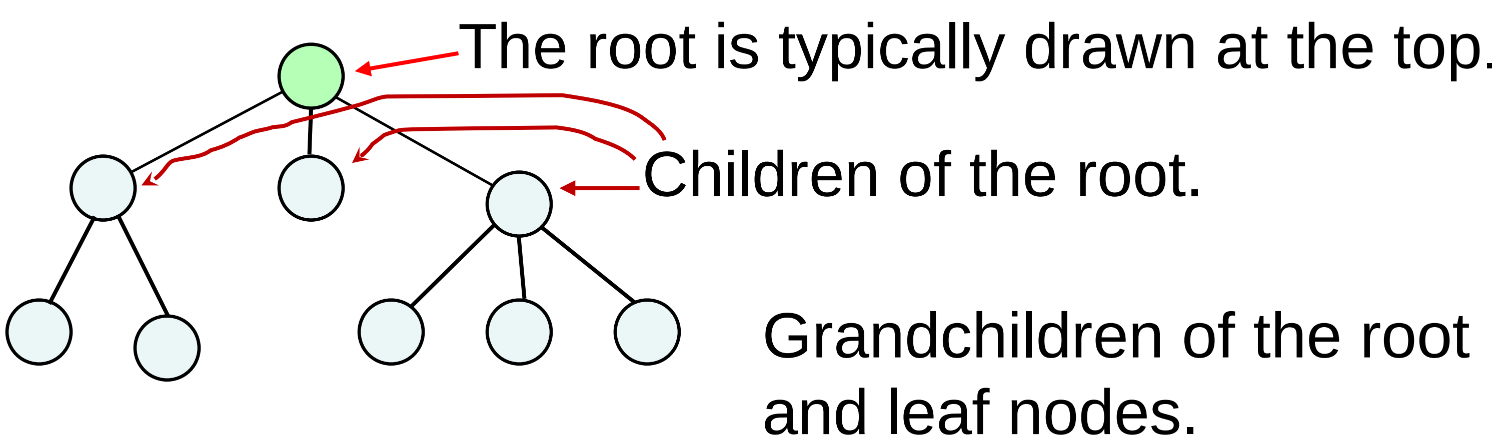
Which of these graphs are trees?



An Organizational Tree for a Computer Company

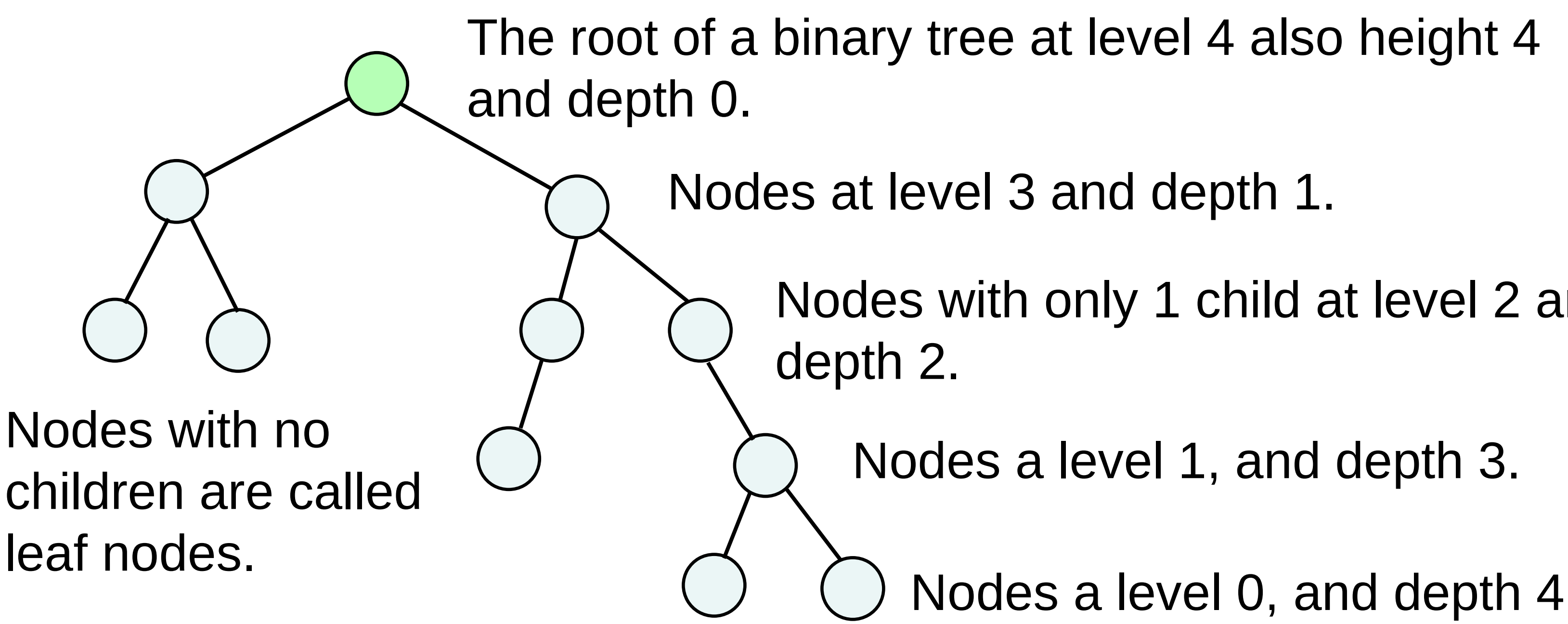
Rooted Trees

- A **rooted tree** is a tree that has a distinguished node called the root. Just as with all trees a rooted tree is acyclic (no cycles) and connected. Rooted trees have many applications in computer science, for example the binary search tree.



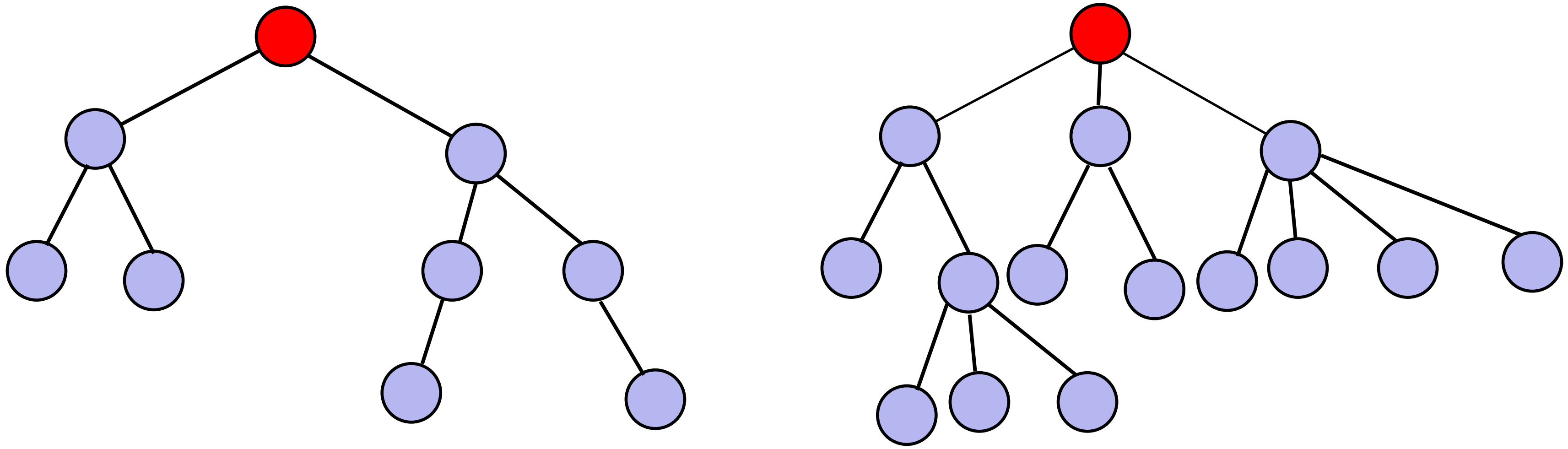
Rooted Binary Trees

- A **Binary tree** is a rooted tree where no node has more than two children. As shown in the previous slide all nodes, except for leaf nodes, have children and some have grandchildren. All nodes except the root have a parent and some have a grandparent and ancestors. A node has at most one parent and at most one grandparent.



Balanced Trees

- A **Balanced tree** is a rooted tree where the leaf nodes have depths that vary by no more than one. In other words, the depth of a leaf node is either equal to the height of the tree or one less than the height. All of the trees below are balanced.



What are the heights of each of these trees?

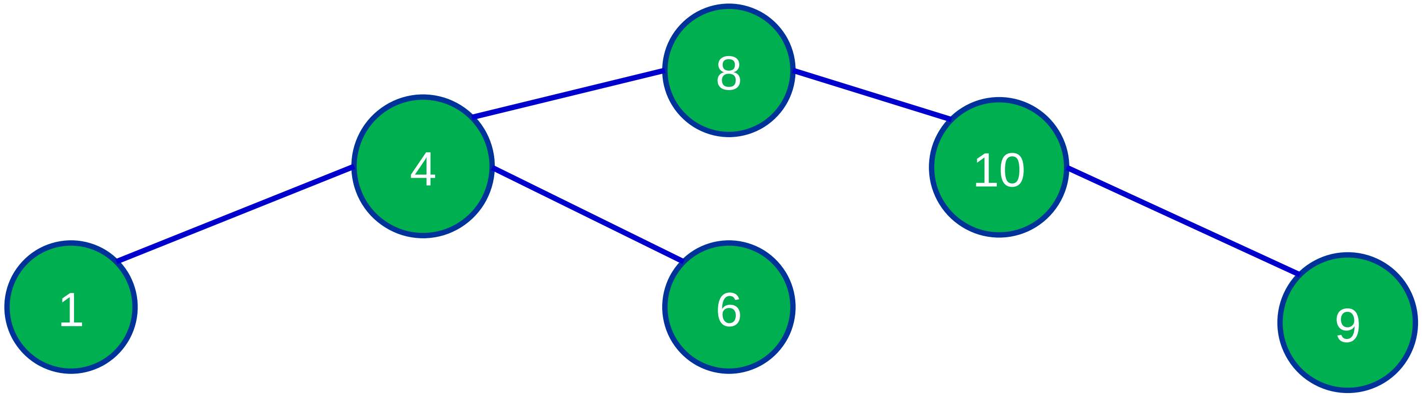
APPLICATIONS OF TREES

Tree Applications

- **Binary Search Trees** to store items for easy retrieval, insertion and deletion. For balanced trees, each of these steps takes *log(N)* time for an *N* node tree.
- Variations of the binary search tree include, the **AVL tree**, **Red-Black tree** and the **B-tree**. These are all designed to keep the tree nearly balanced. The first two are binary trees while the **B-tree** has more than two children for each internal node.
- **Game trees** are used extensively in AI.
- **Huffman trees** are used to compress data. They are most commonly used to compress faxes before transmission.
- **Spanning trees** are subgraphs that have applications to computer and telephone networks. **Minimum spanning trees** are of special interest.
- **Steiner trees** are a generalization of the spanning tree with in multicast communication.

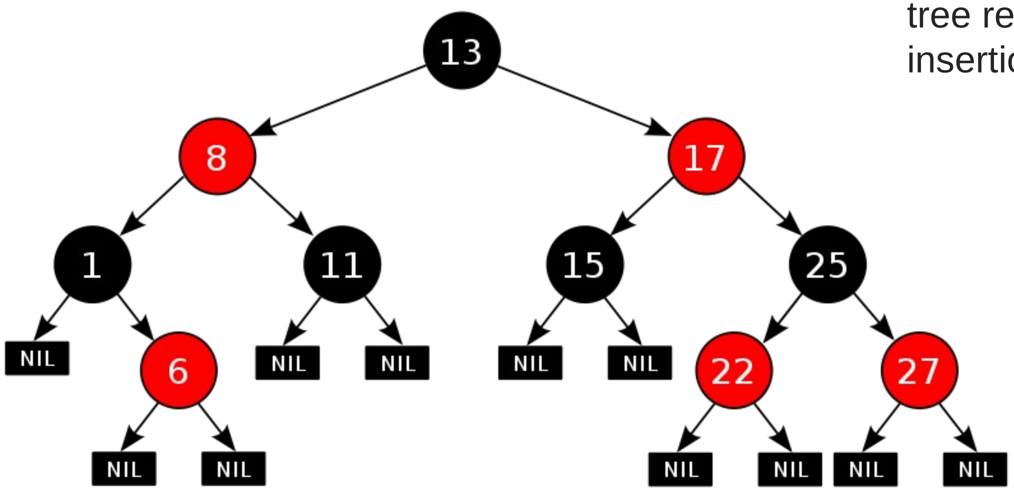
Binary Search Trees

- A binary search tree is a binary tree, whose internal nodes contain the keys $k = x.key \; \forall x \in S$



- AVL tree: self-balancing binary search tree. the heights of the two child sub-trees of any node differ by at most one.

- Red-Black tree: a type of self-balancing binary search tree. The self-balancing is provided by painting each node with one of two colors.



Each node of the binary tree has **an extra bit**, and that bit is often interpreted as the color (red or black) of the node. These color bits are used to ensure the tree remains approximately balanced during insertions and deletions.

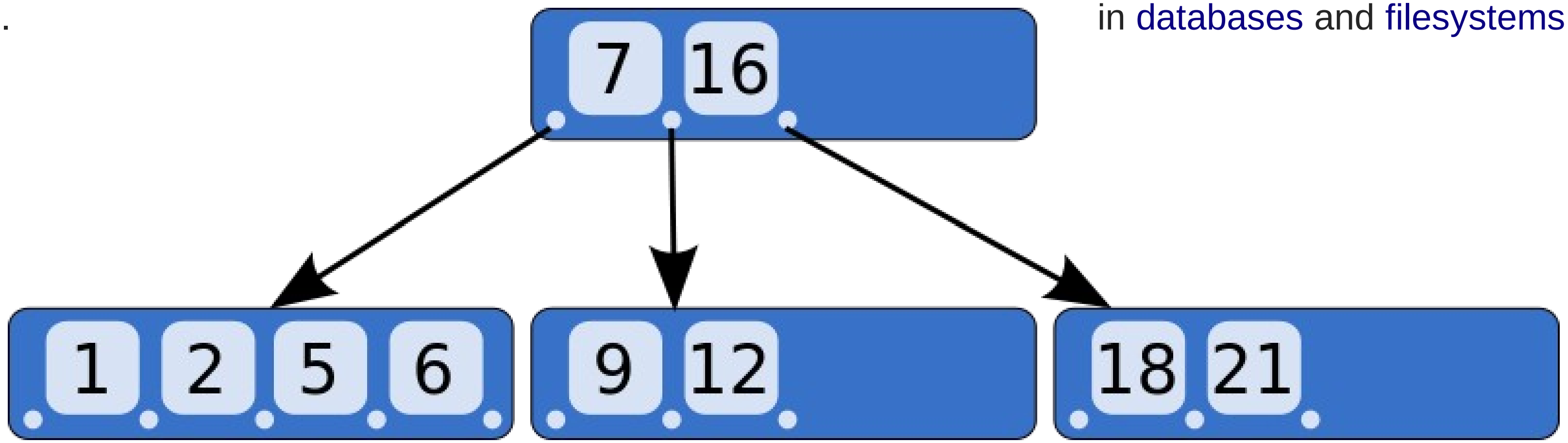
Balance is preserved by painting each node of the tree with one of two colors (typically called 'red' and 'black') in a way that satisfies certain properties, which collectively constrain how unbalanced the tree can become in the worst case.
When the tree is modified, the new tree is subsequently **rearranged and repainted to restore the coloring properties.** The properties are designed in such a way that this rearranging and recoloring can be performed efficiently.

Binary Search Trees

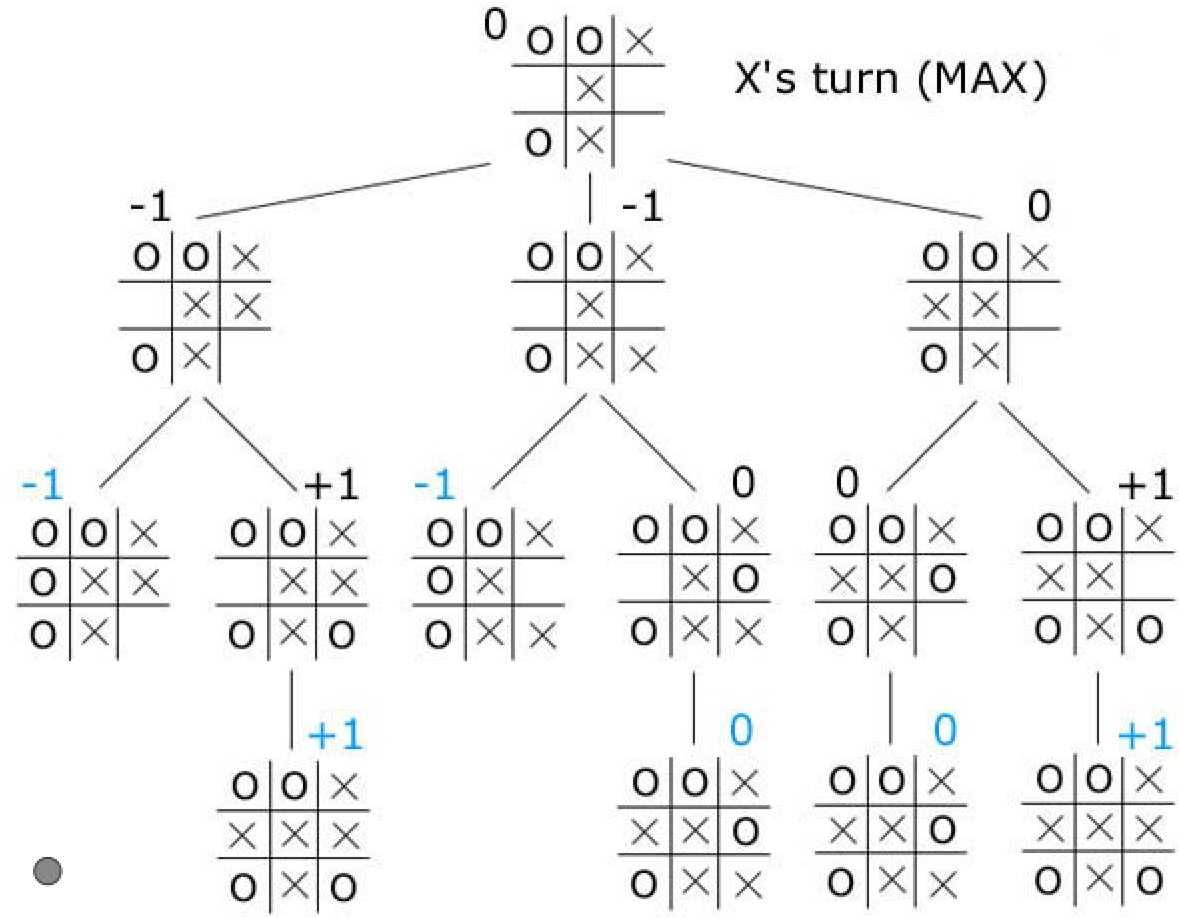
- B-tree: is a generalization of a binary search tree in that a node can have more than two children.

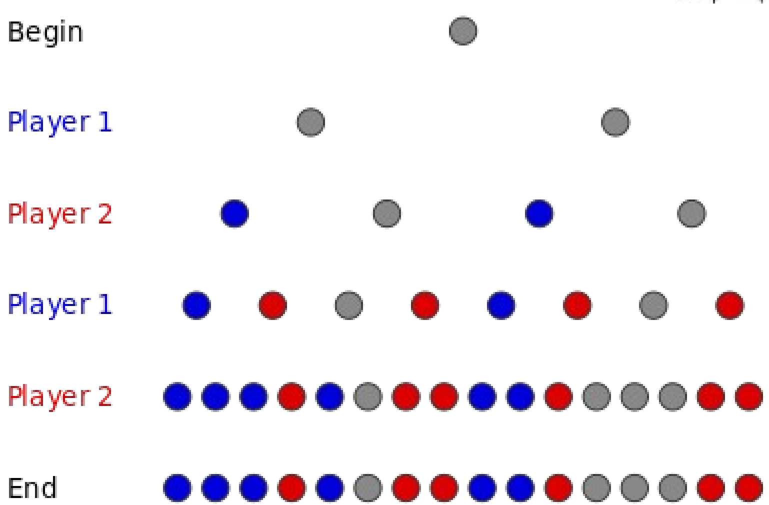
B-tree is a self-balancing tree data structure that keeps data sorted and allows searches, sequential access, insertions, and deletions in logarithmic time.

B-tree is optimized for systems that read and write large blocks of data. B-trees are a good example of a data structure for external memory. It is commonly used in databases and filesystems.



Game Trees

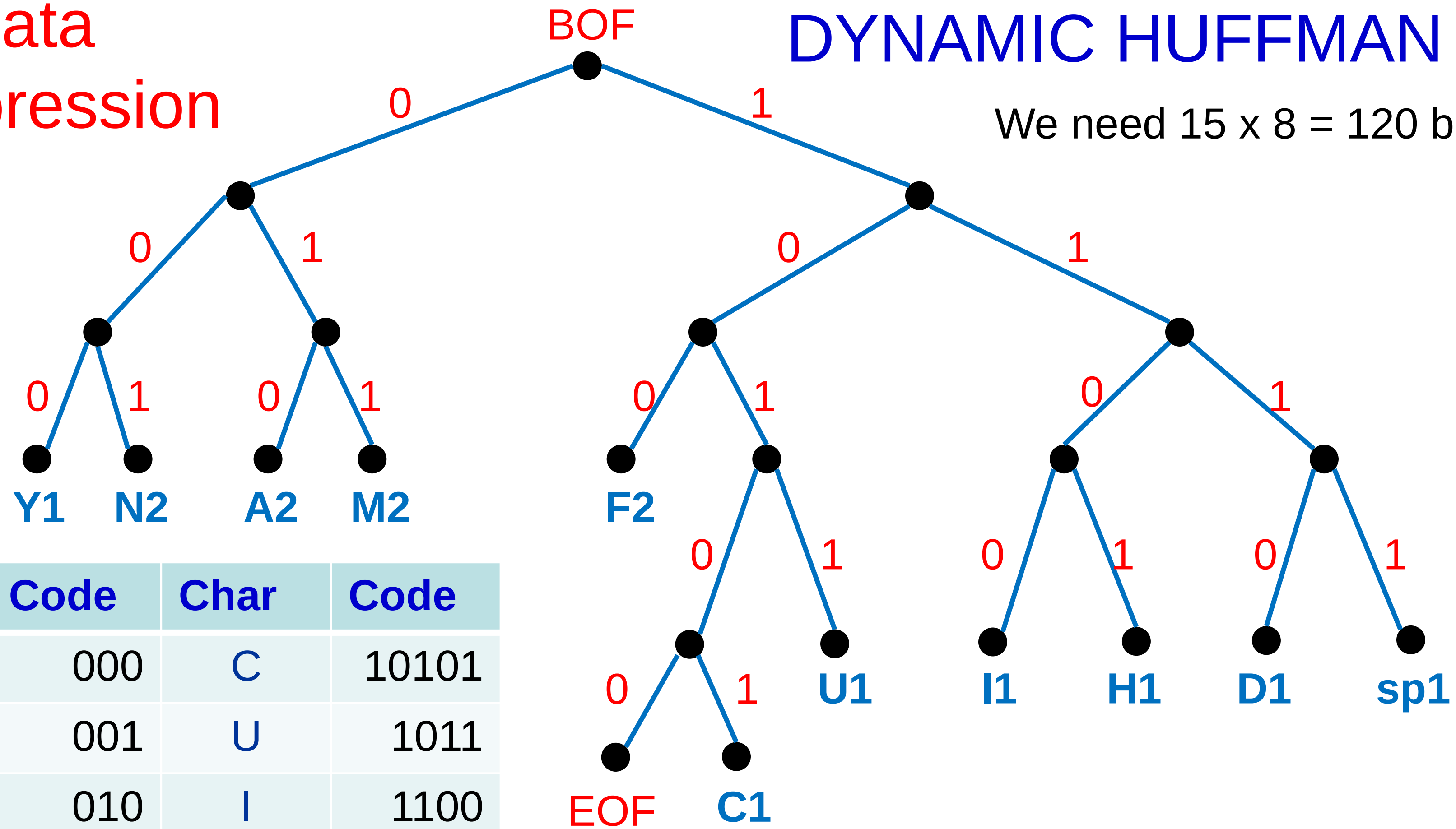




Data Compression

DYNAMIC HUFFMAN

We need 15 x 8 = 120 bits

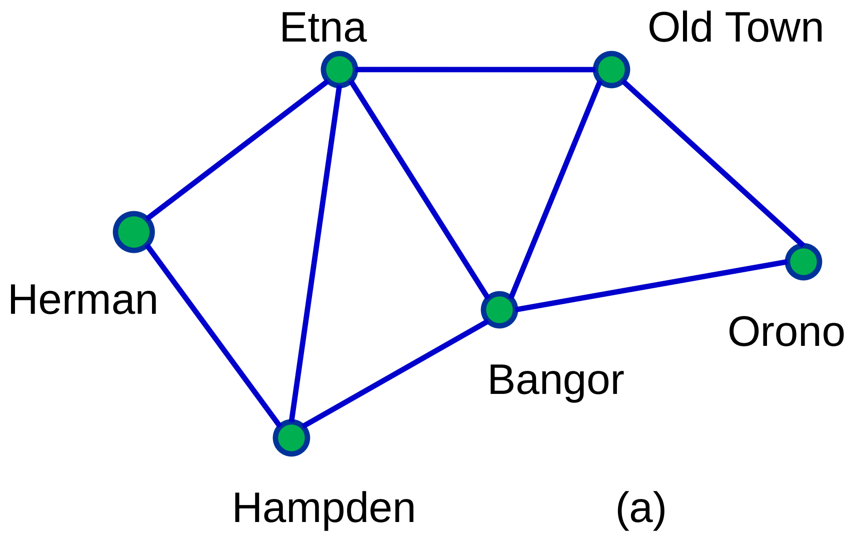


Char	Code	Char	Code
Y	000	C	10101
N	001	U	1011
A	010	I	1100
M	011	H	1101
F	100	D	1110
Space	1111	EOF	10100

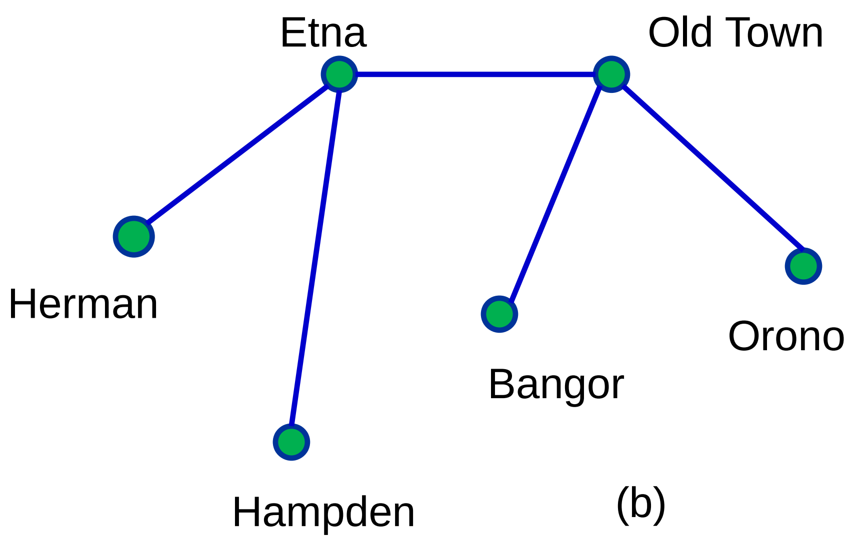
Compress this string by Huffman Coding, we need 45 bits << 120 bits

Spanning Trees

- System of roads in Maine represented by the simple graph shown in the (a). The only way the roads can be kept open in the winter is by frequently plowing them. The highway department wants to plow the fewest roads so that there will always be cleared roads connecting any two towns. How can this be done?
- Figure (b) show one set of roads. The graph in (b) is a tree.

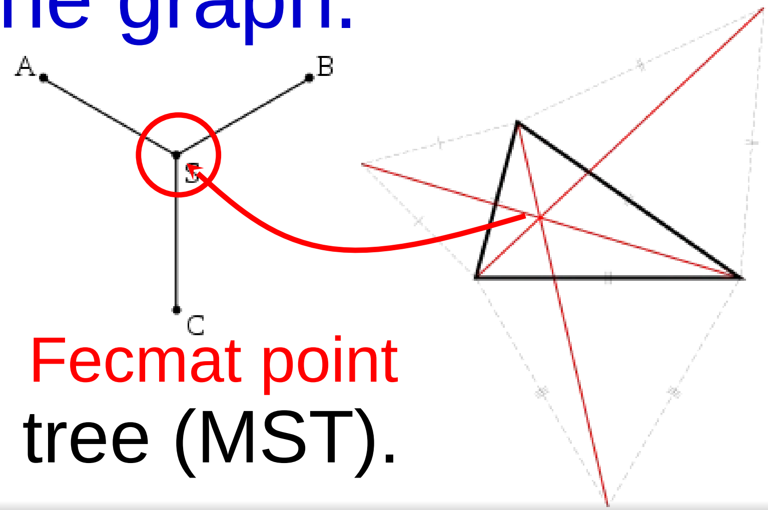


(a)



(b)

- Given a weighted graph in which a subset of vertices are identified as terminals, find a minimum-weight connected sub-graph that includes all the terminals.
- Find the shortest interconnect for a given set of objects. Extra intermediate vertices and edges may be added to the graph.
- E.g. Steiner tree
  - $|N| = 1$: trivial
  - $|N| = 2$: shortest path
  - $N = V$: minimum spanning tree (MST).



Fecmat point

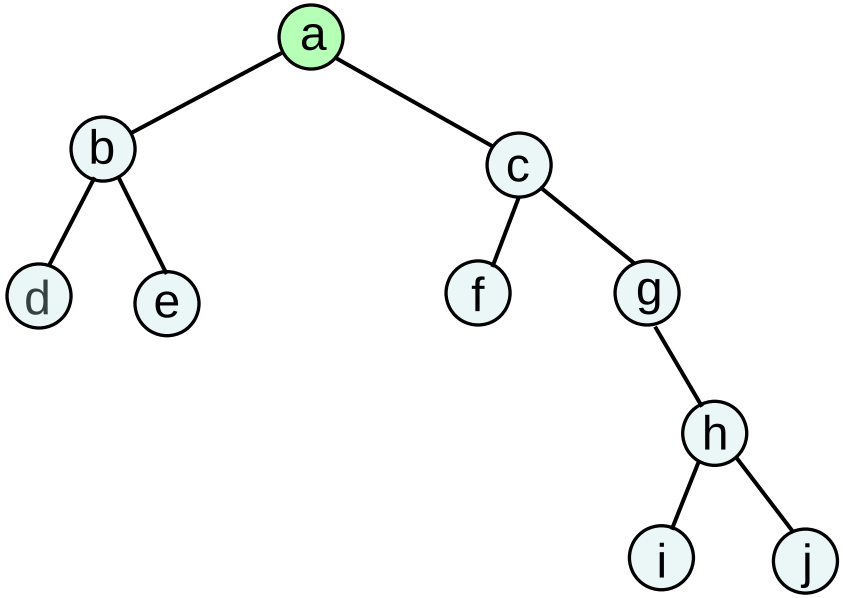
TREE TRAVERSAL

Tree Traversal

- A process of visisting (examining and/or updating) each node in a tree data structure, exactly once, an a systematic way. Such traversal are classified by the order in which the nodes are visited. There are 3 types:
  - pre-order (root,left,right),
  - in-order (left,root,right),
  - post-order (left,right,root).
- Depth-First-Search algorithm (Stack-LIFO).
- Breadth-First-Search algorithm (Queue-FIFO).

Binary tree traversal functions – Inorder

```
void in_order(node T)
        if (T ≠ null) {
                in_order(T−>left)
                cout << T −>data << “,  ”
                in_order(T −>right)
        }
end in_order
```



DFS: Left, Root, Right

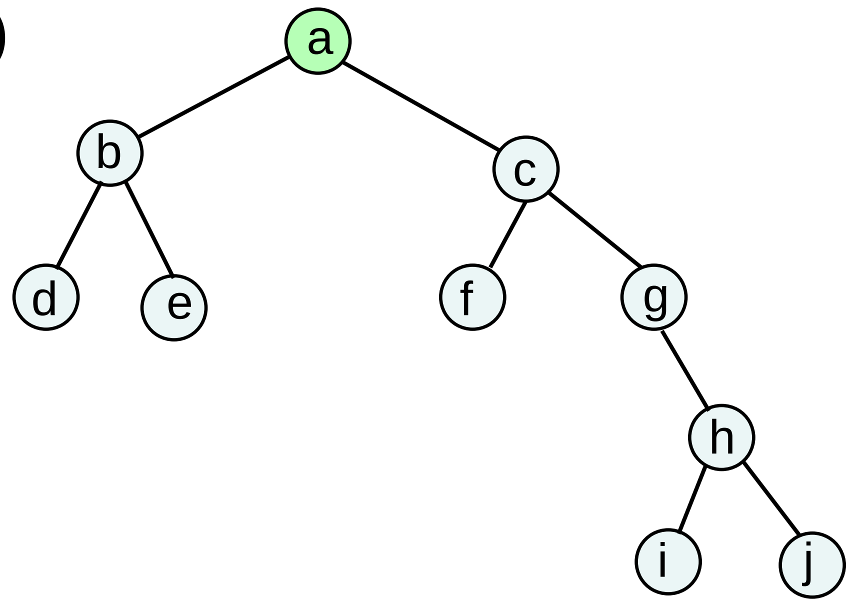
Output will be something like

d,b,e,a,f,c,g,I,h,j

Binary tree traversal functions – Inorder

```
void in_order(node T)
        if (T ≠ null) {
                in_order(T−>left)
                cout << T − >data << “,  ”
                in_order(T − >right)
        }
end in_order
```

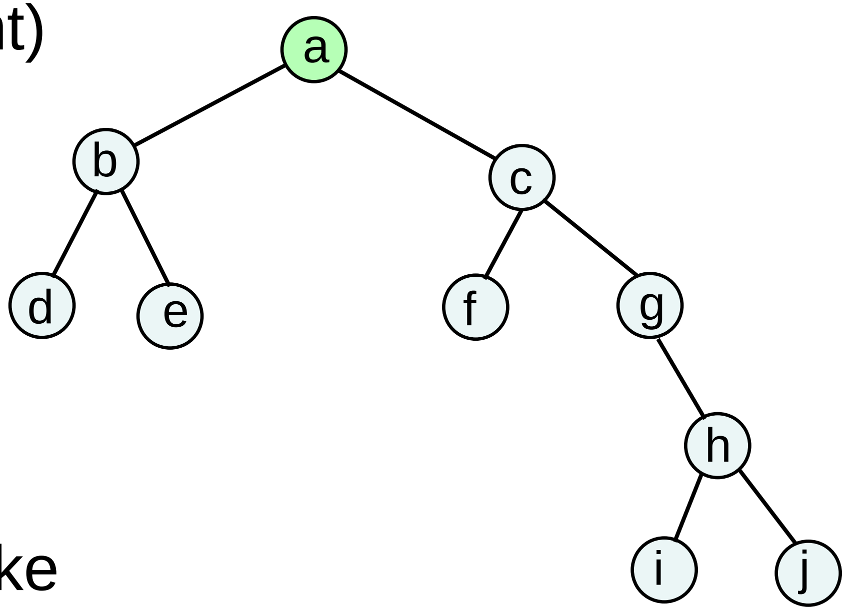
T is NULL
T is d prints d
T is b
T is a
main calls in_order



DFS: Left, Root, Right

Binary tree traversal functions – Preorder

```
void pre_order(node T)
        if (T ≠ null) {
                cout << T ->data << ",  "
                pre_order(T->left)
                pre_order(T ->right)
        }
end pre_order
```



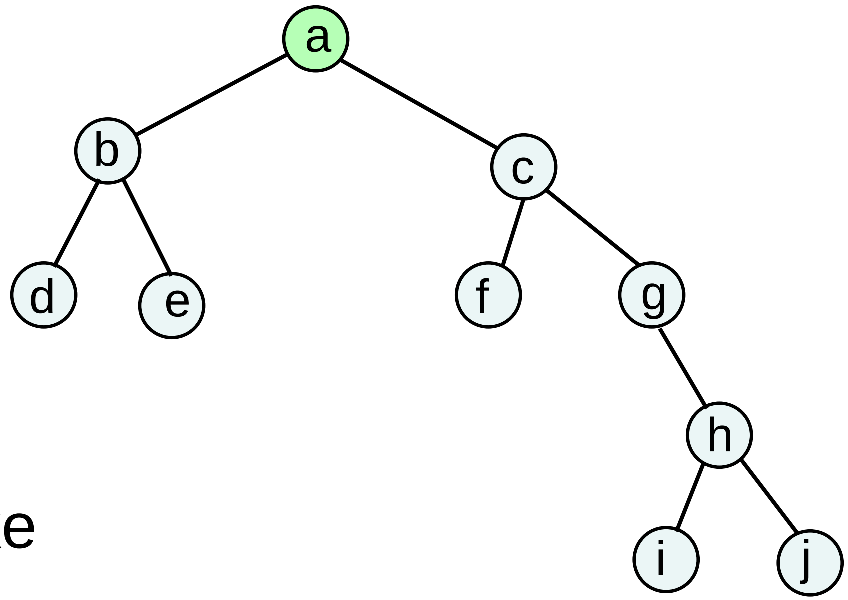
DFS: Root, Left, Right

output will be something like

a, b, d, e, c, f, g, h, i, j

Binary tree traversal functions – Postorder

```
void post_order(node T)
        if (T ≠ null) {
                post_order(T−>left)
                post_order(T−>right)
                 cout << T −>data << ",  "
        }
end post_order
```



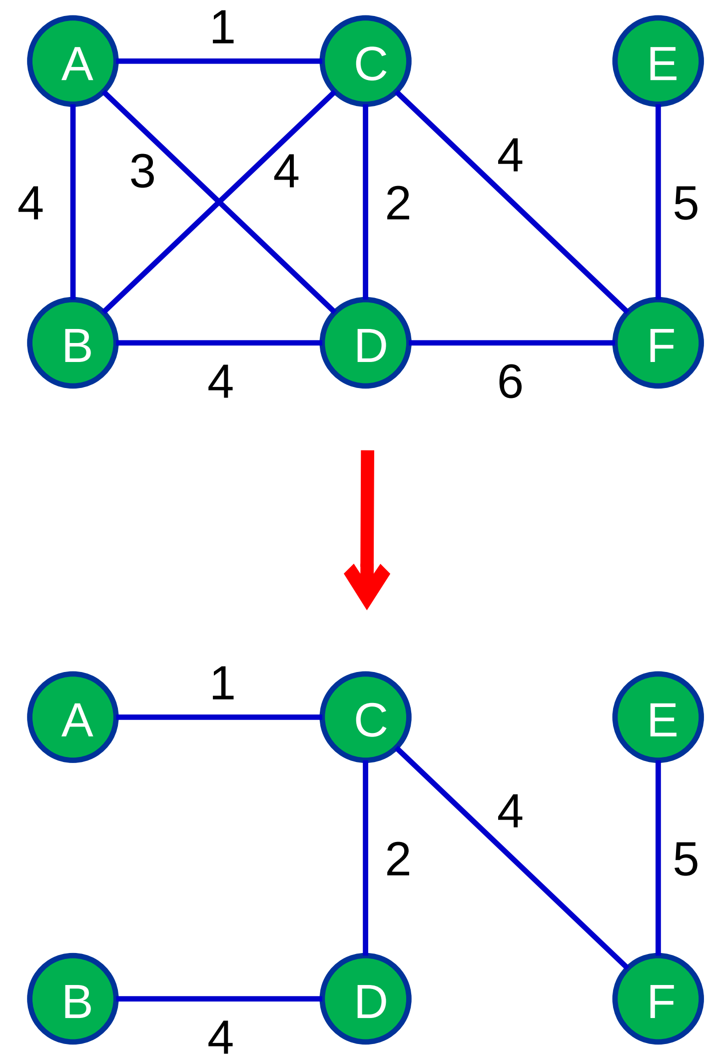
DFS: Left, Right, Root

output will be something like

d, e, b, f, i, j, h, g, c, a

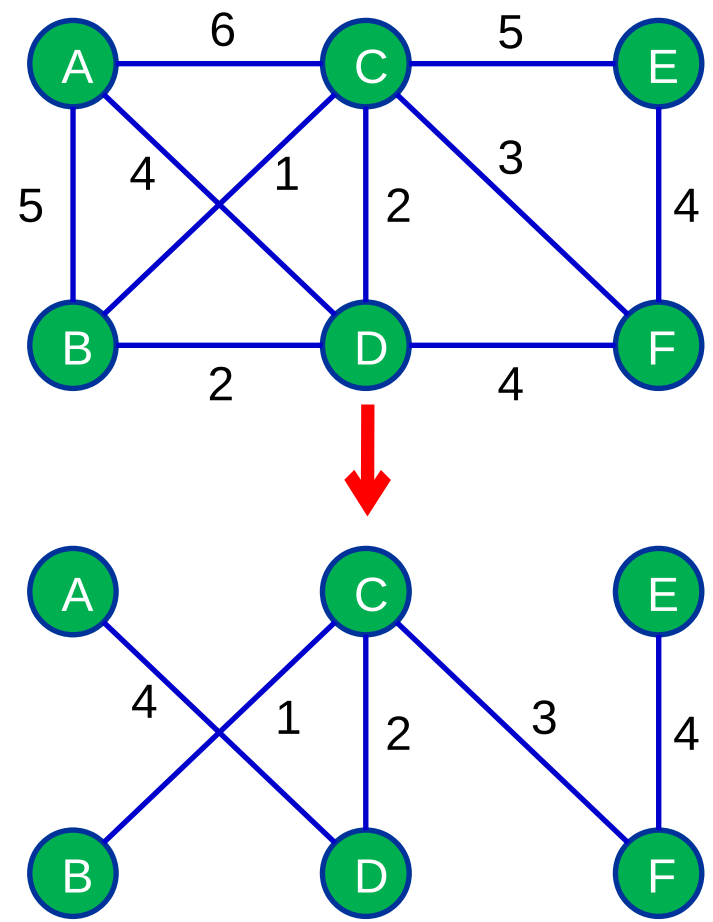
MINIMUM SPANNING TREES (MST)

- Link computers. The edge's weight is maintenance cost. What is the cheapest possible network?
- The particular tree we want is the one with minimum total weight, known as the minimum spanning tree.
- Definition: An undirected graph $G=(V,E)$; edge weights w_e. A tree $T=(V,E')$ with $E' \subseteq E$, minimize $\text{weight}(T) = \sum_{e \in E'} w_e$.



Minimum Spanning Trees (MST)

- A greedy approach: Kruskals MST algorithm.
- The correctness of Kruskal's method follows from a certain *cut property*, which is general enough to also justify a whole slew of other MST algorithms.
- Cut property: Suppose edges X are part of a MST of $G = (V, E)$ Pick any subset of nodes S for which X does not cross between S and $V - S$, and let e be the lightest edge across this



Then $X \cup \{e\}$ is part of some MST

Minimum Connector Algorithms

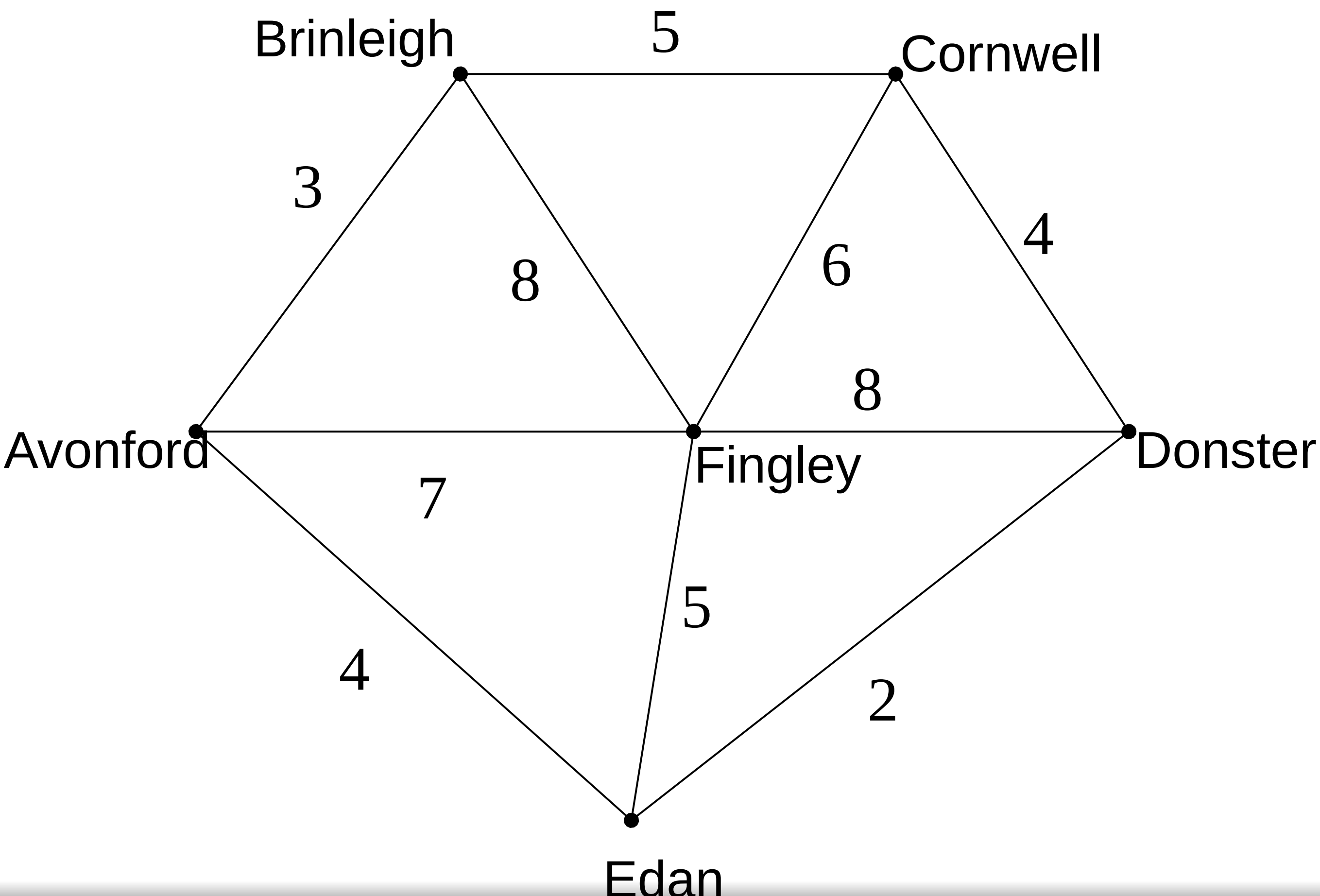
Kruskal's algorithm

1. Select the shortest edge in a network
2. Select the next shortest edge which does not create a cycle
3. Repeat step 2 until all vertices have been connected

Prim's algorithm

1. Select any vertex
2. Select the shortest edge connected to that vertex
3. Select the shortest edge connected to any vertex already connected
4. Repeat step 3 until all vertices have been connected

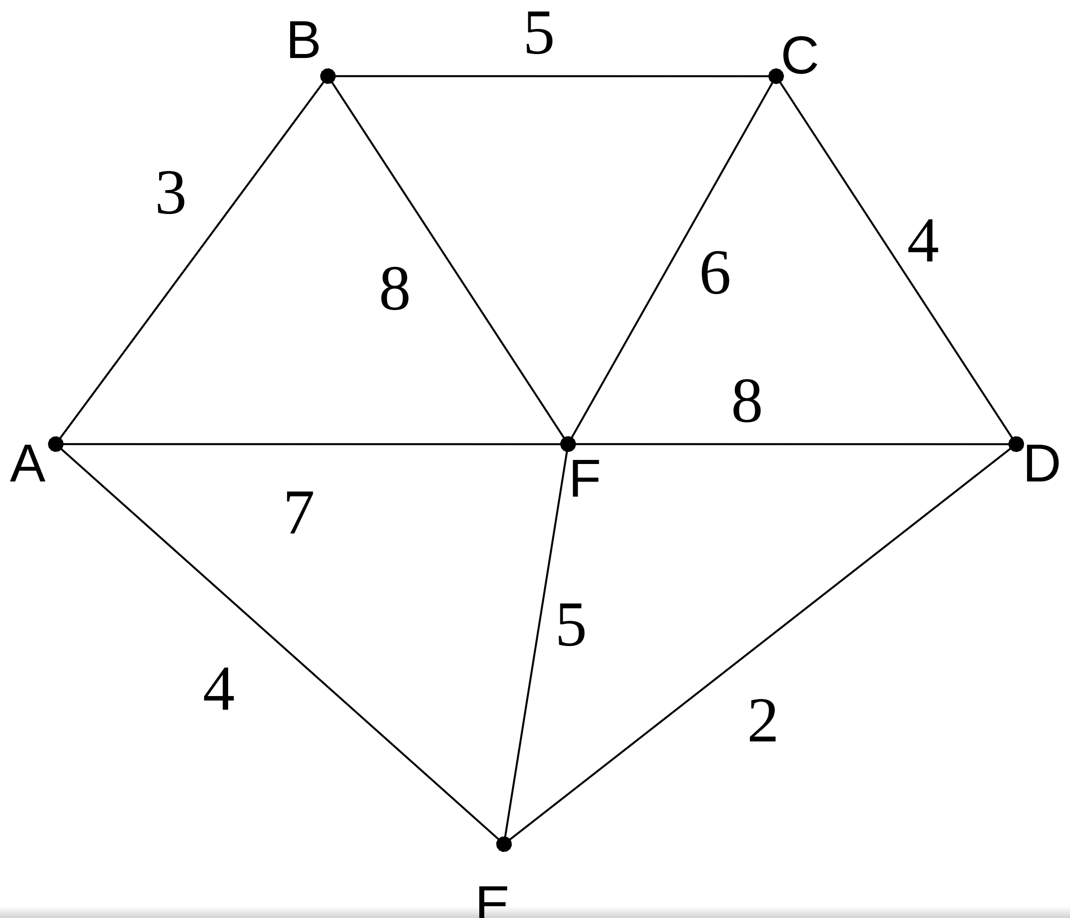
Example



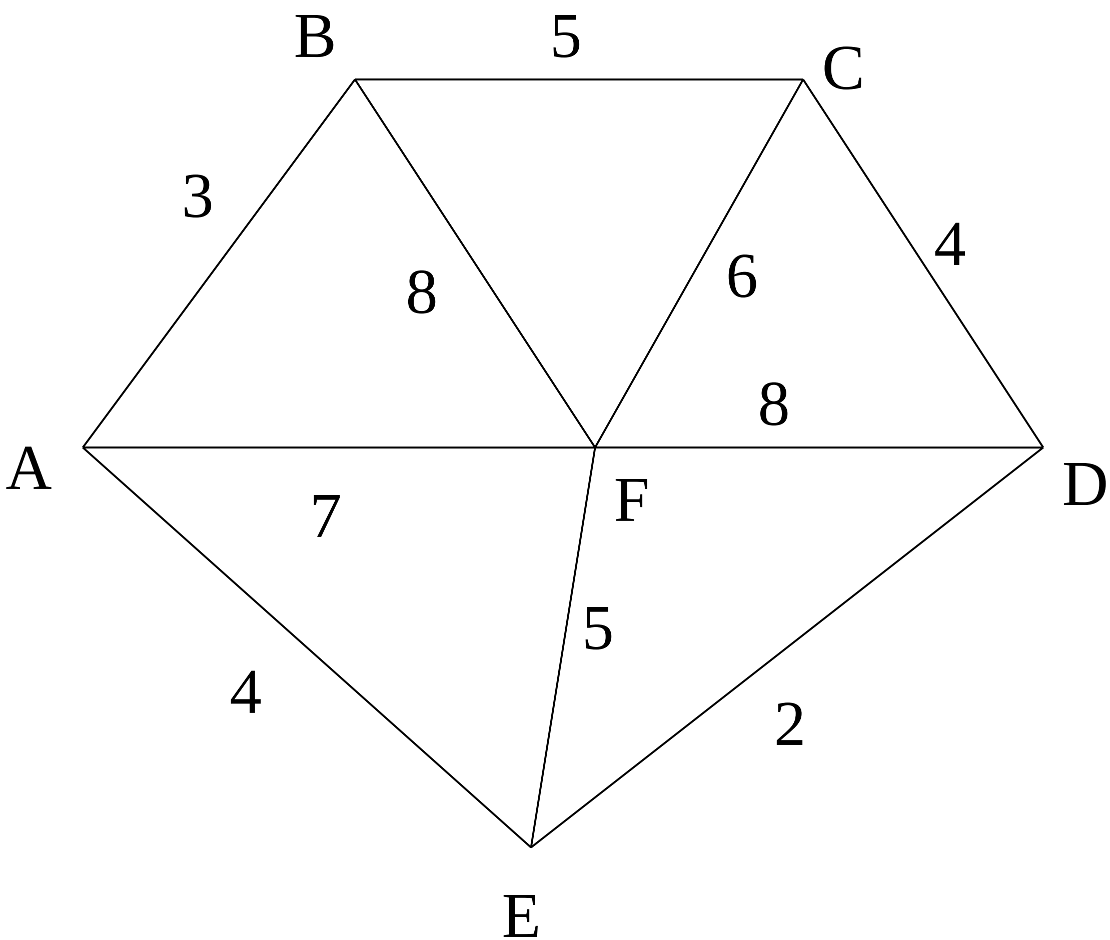
Brinleigh
5
Cornwell
3
8
6
4
8
Avonford
Fingley
Donster
7
5
4
2
Edan

Example

We model the situation as a network, then the problem is to find the minimum connector for the network



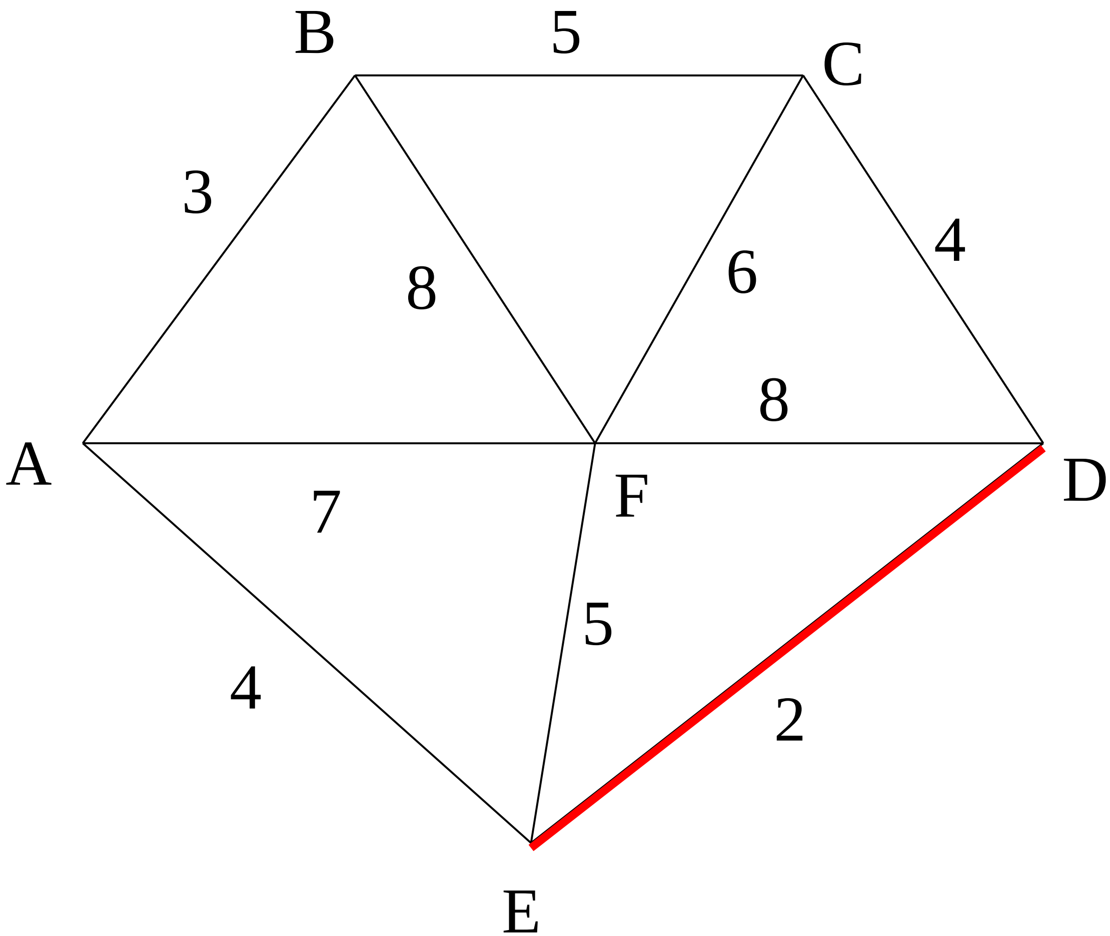
Kruskal's Algorithm



List the edges in order of size:

ED 2
AB 3
AE 4
CD 4
BC 5
EF 5
CF 6
AF 7
BF 8
CF 8

Kruskal's Algorithm

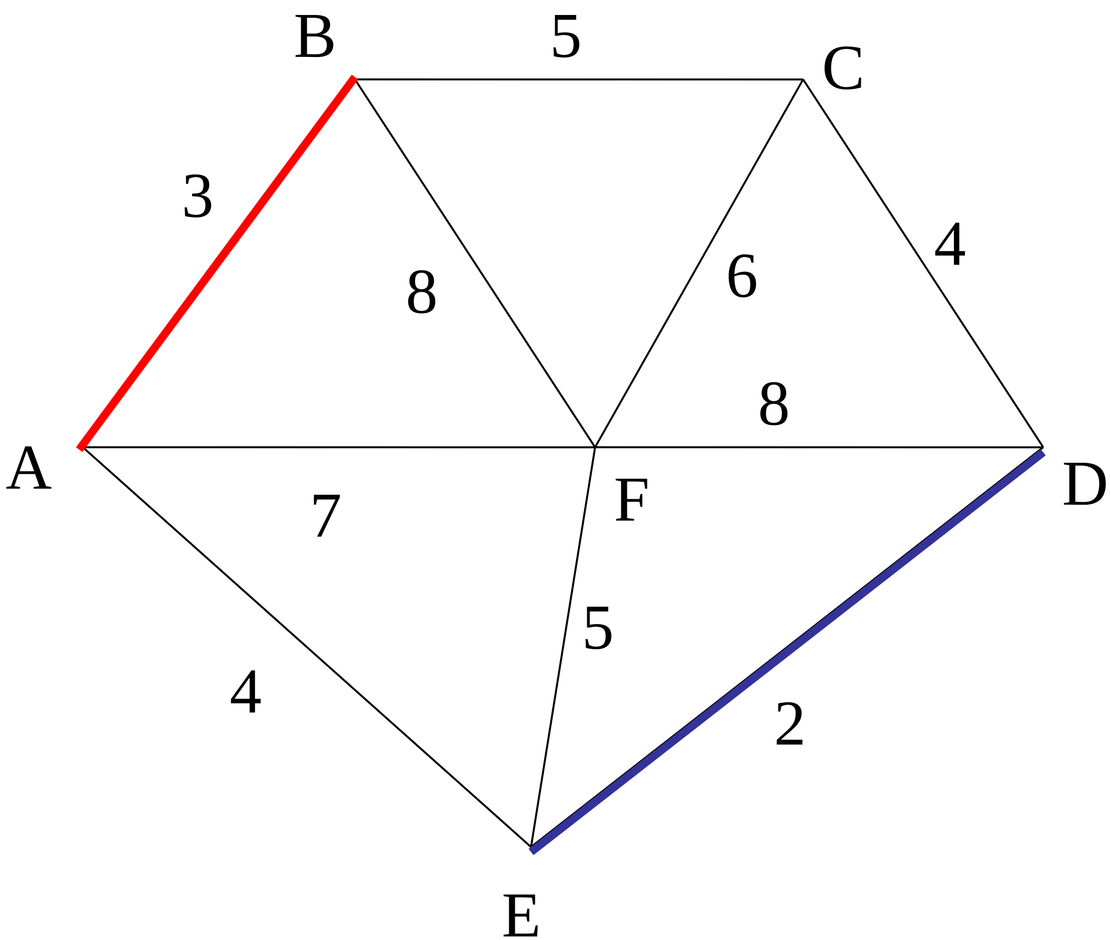


Select the shortest edge in the network

ED 2

Example

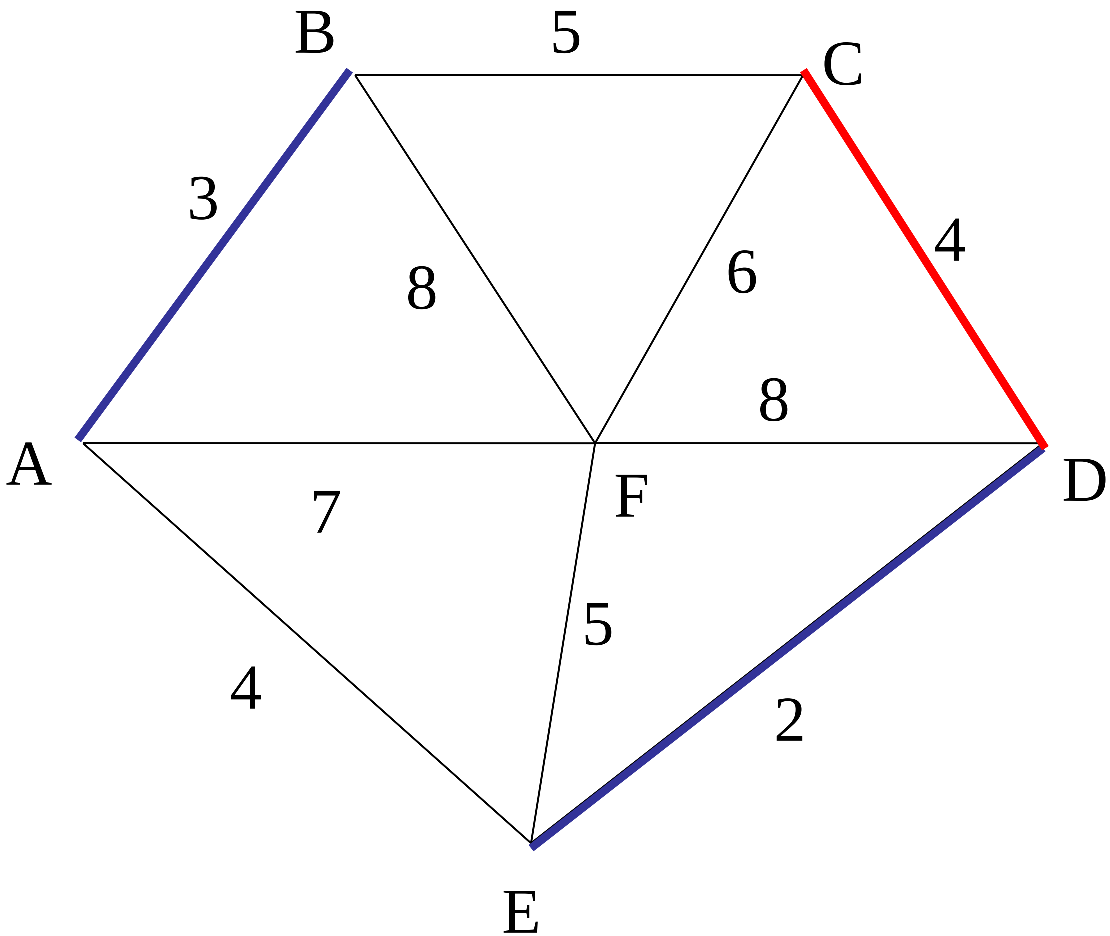
Kruskal's Algorithm



Select the next shortest edge which does not create a cycle

ED 2
AB 3

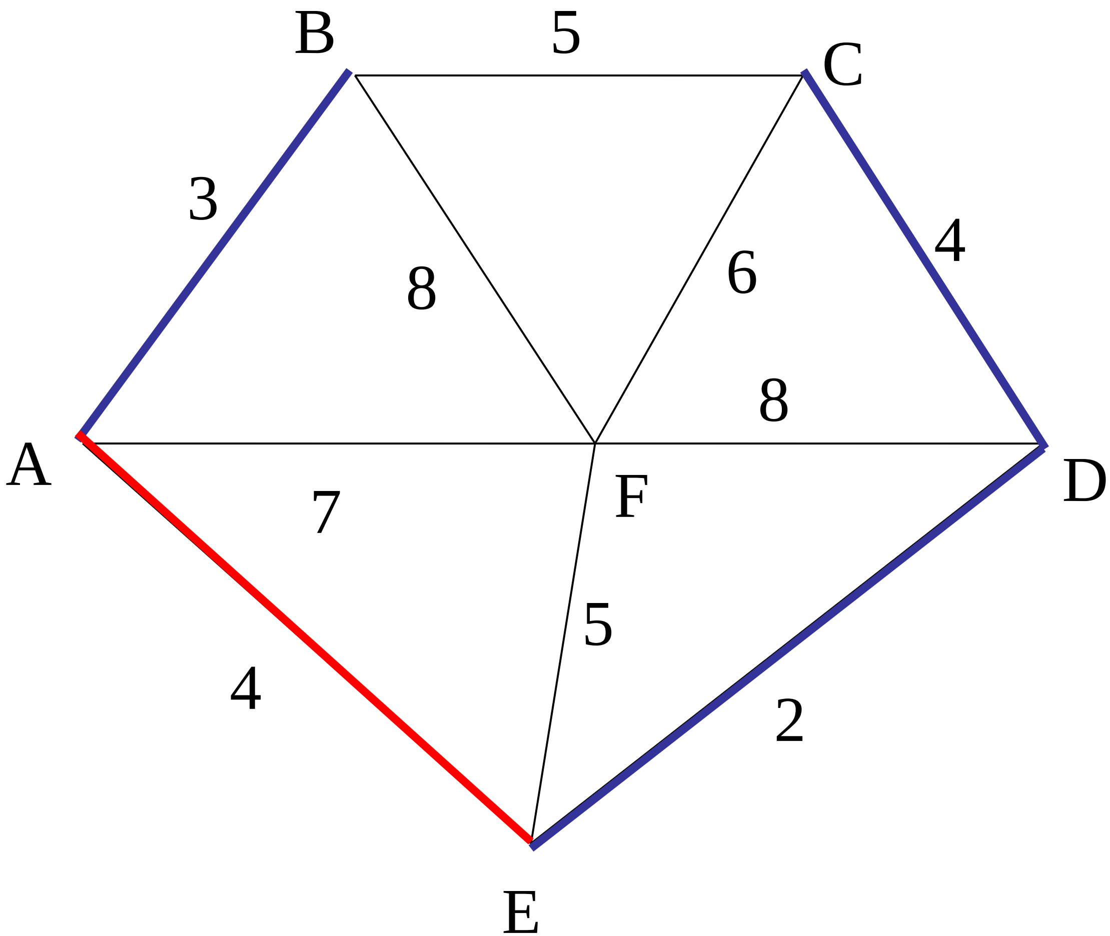
Kruskal's Algorithm



Select the next shortest edge which does not create a cycle

ED 2
AB 3
CD 4 (or AE 4)

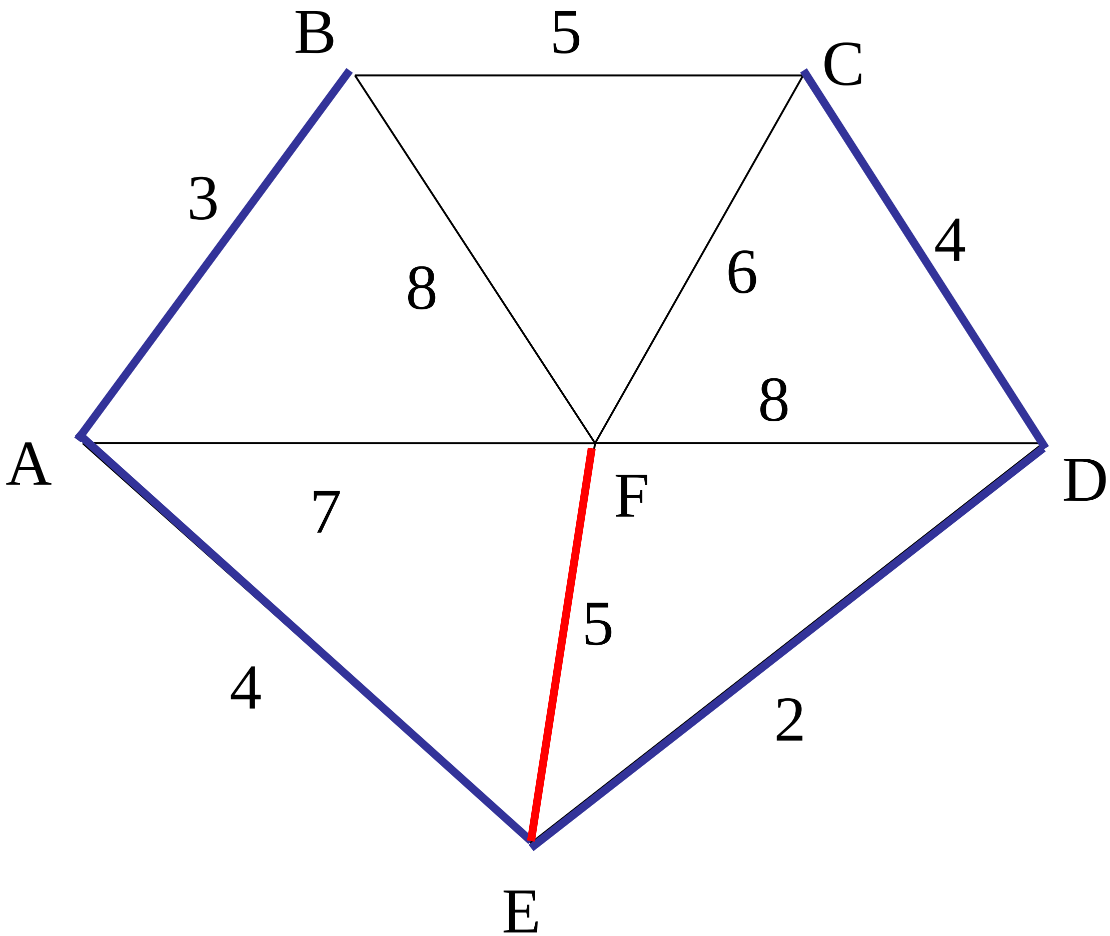
Kruskal's Algorithm



Select the next shortest edge which does not create a cycle

ED 2
AB 3
CD 4
AE 4

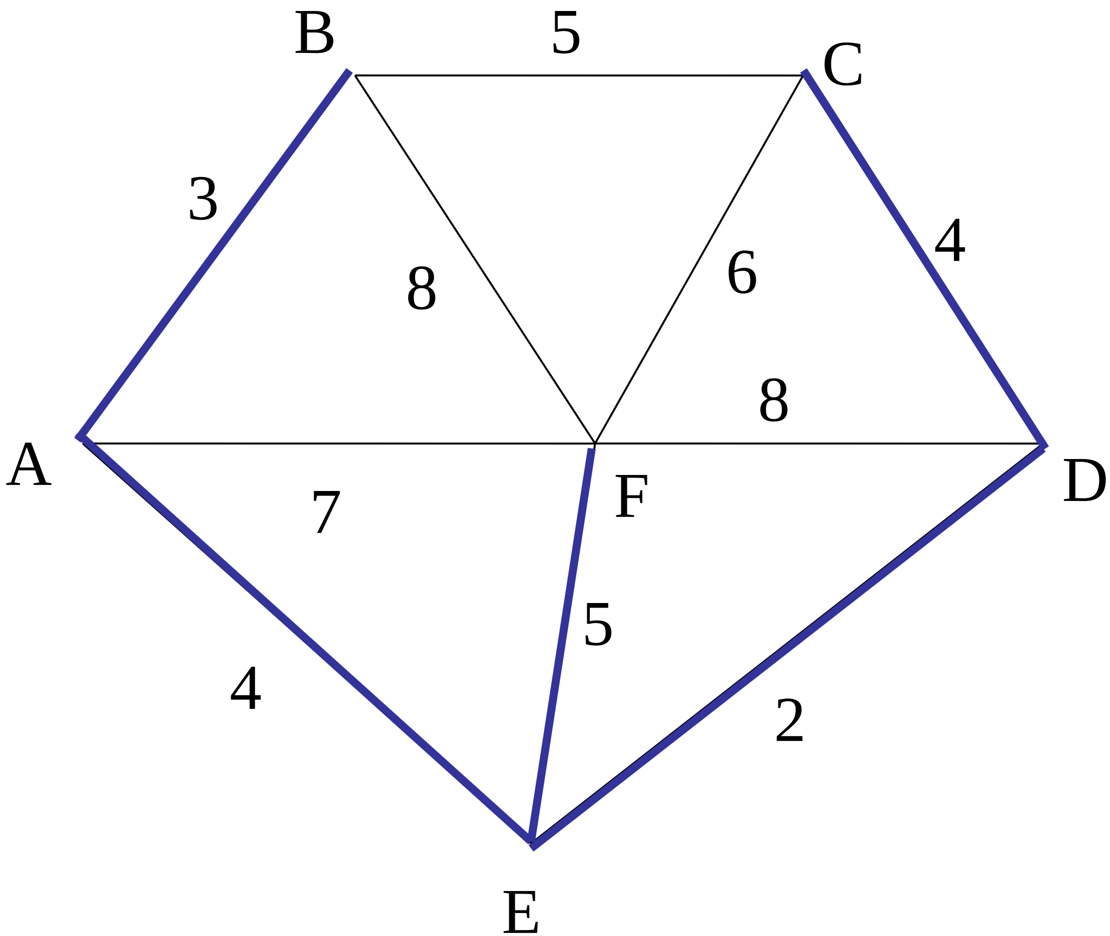
Kruskal's Algorithm



Select the next shortest edge which does not create a cycle

ED 2
AB 3
CD 4
AE 4
BC 5 – forms a cycle
EF 5

Kruskal's Algorithm



All vertices have been connected.

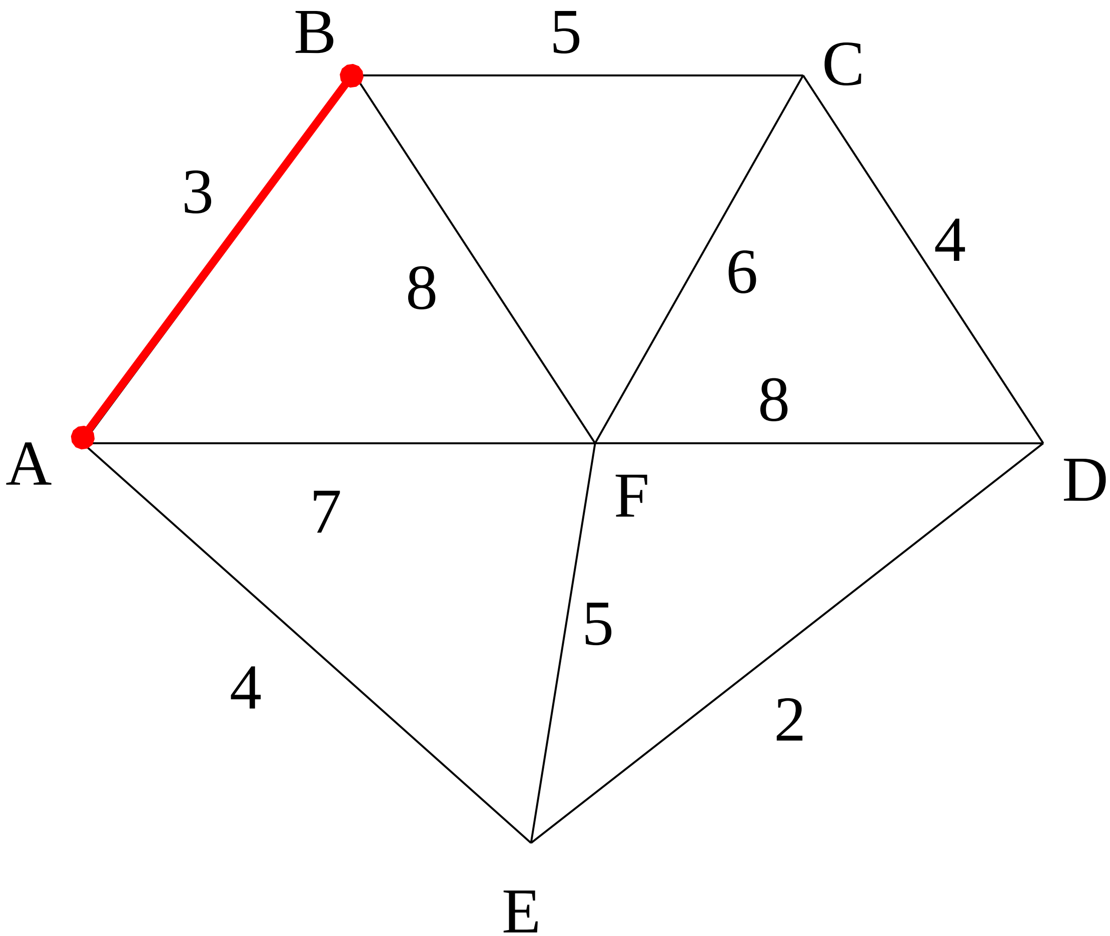
The solution is

ED 2
AB 3
CD 4
AE 4
EF 5

Total weight of tree: 18

Example

Prim's Algorithm



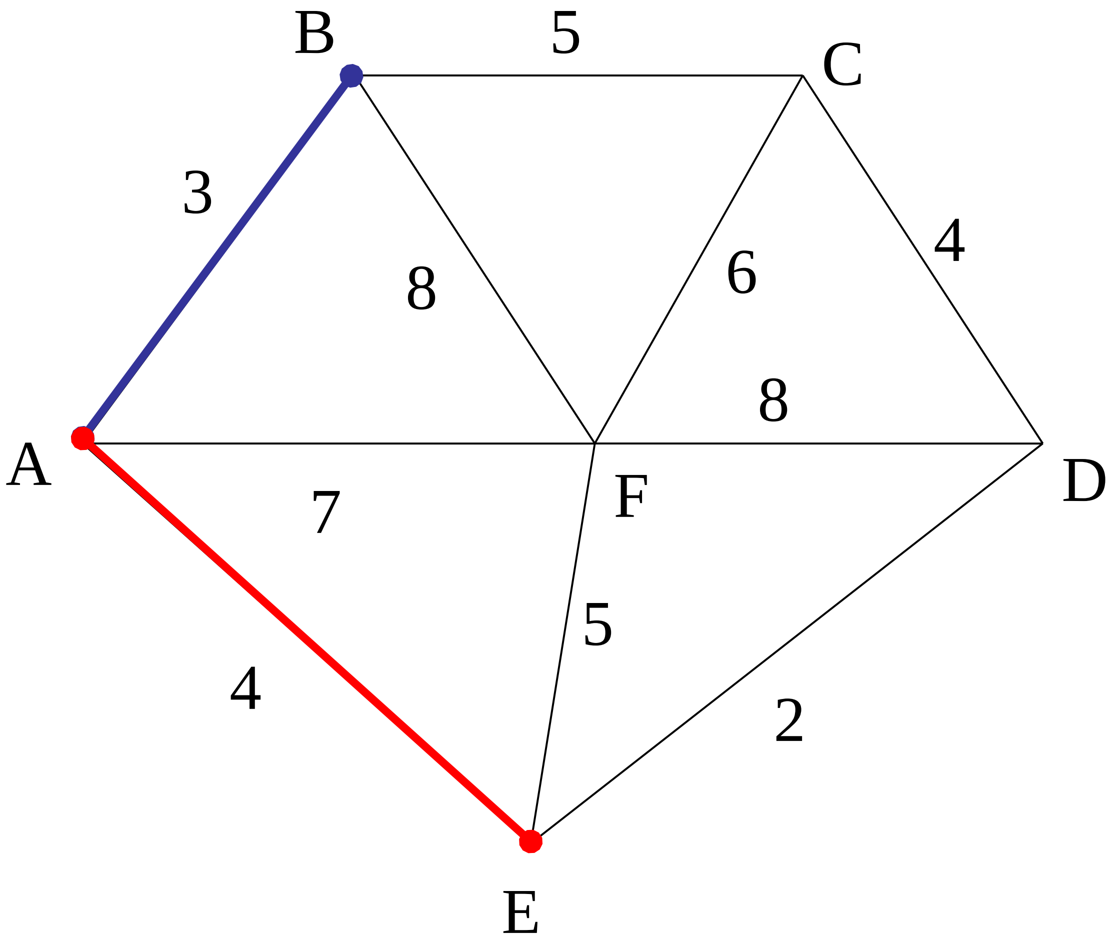
Select any vertex

A

Select the shortest edge connected to that vertex

AB 3

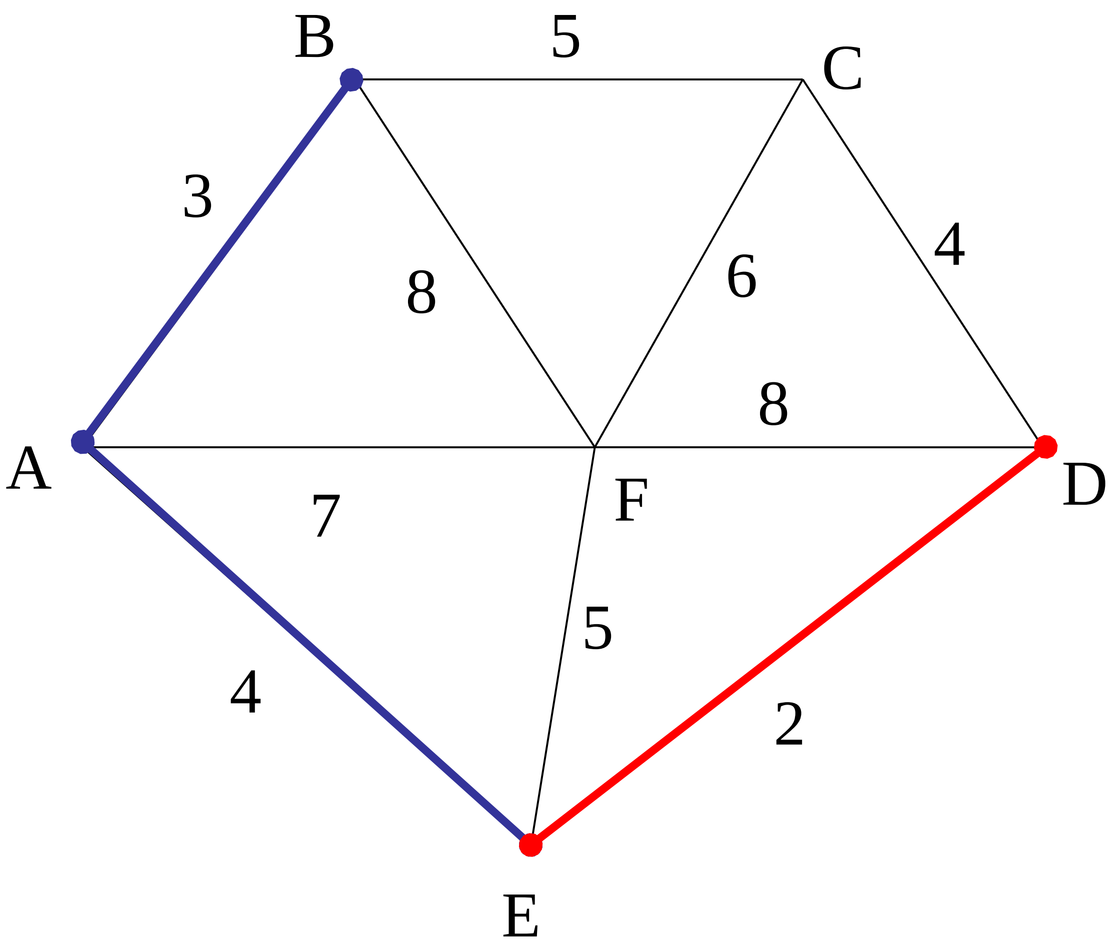
Prim's Algorithm



Select the shortest edge connected to any vertex already connected.

AE 4

Prim's Algorithm

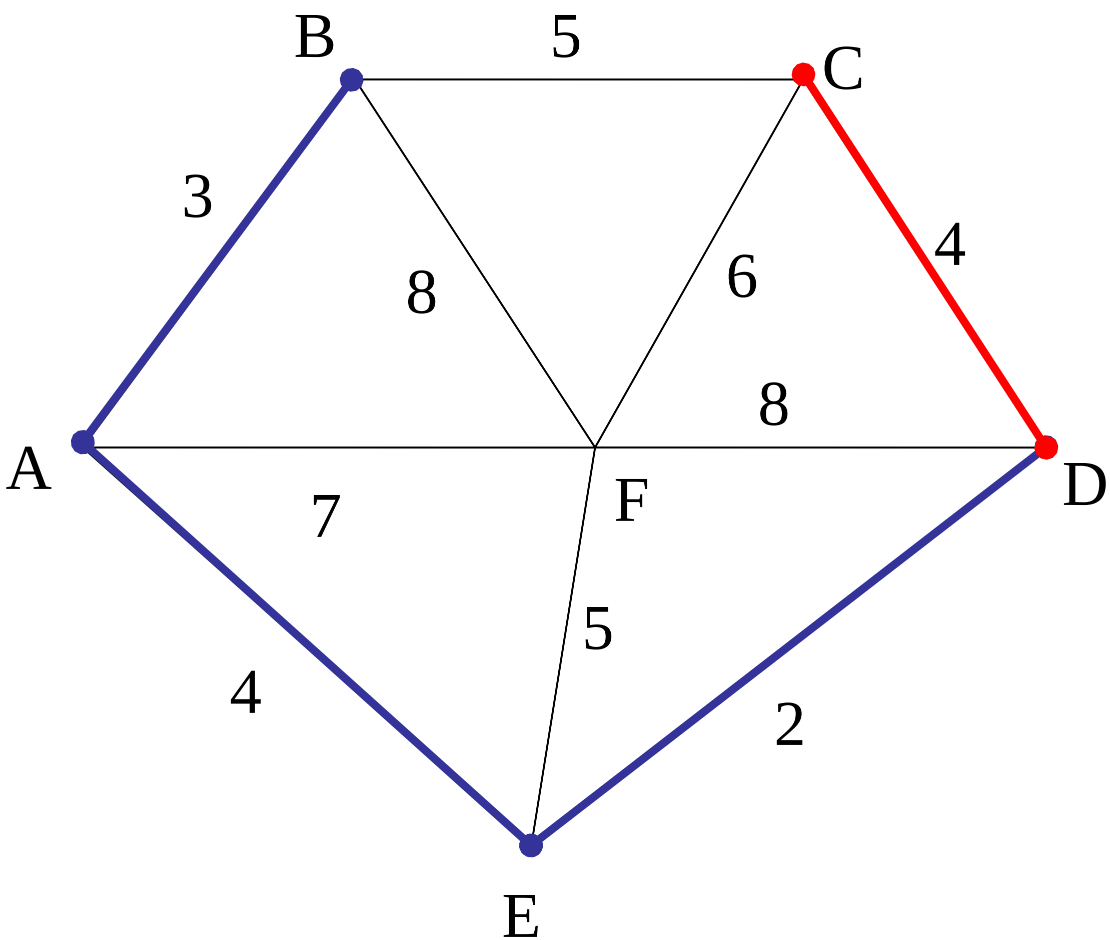


Select the shortest edge connected to any vertex already connected.

ED 2

Example

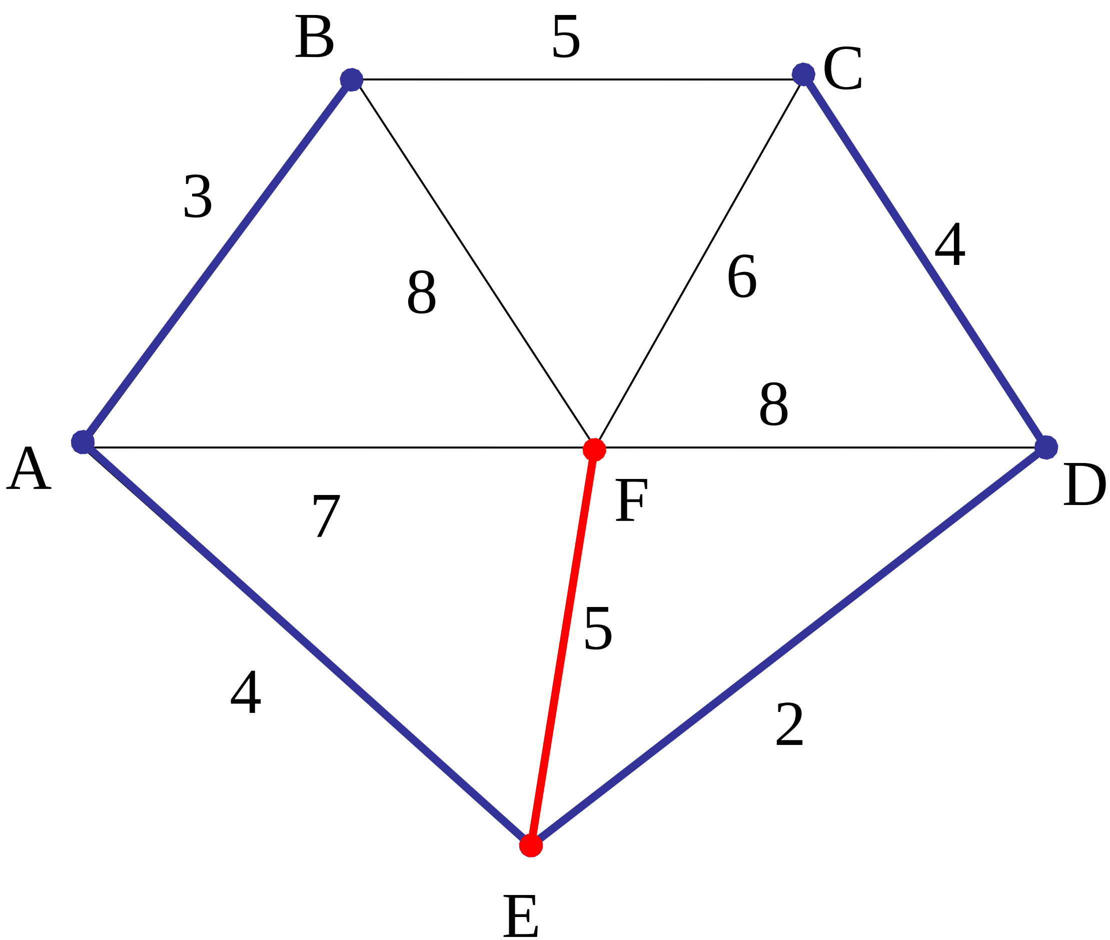
Prim's Algorithm



Select the shortest edge connected to any vertex already connected.

DC 4

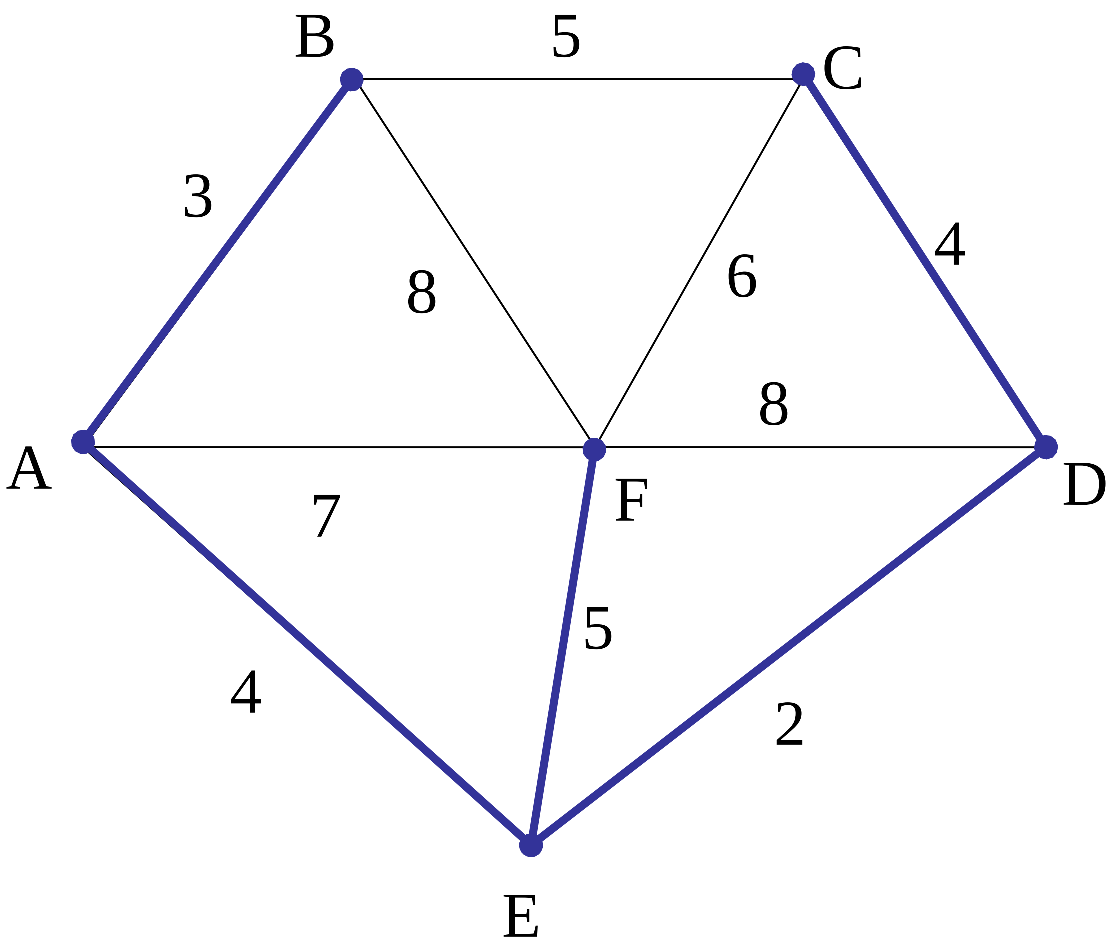
Prim's Algorithm



Select the shortest edge connected to any vertex already connected.

EF 5

Prim's Algorithm



All vertices have been connected.

The solution is

AB 3
AE 4
ED 2
DC 4
EF 5

Total weight of tree: 18

Notes

- Both algorithms will always give solutions with the same length.
- They will usually select edges in a different order – you must show this in your workings.
- Occasionally they will use different edges – this may happen when you have to choose between edges with the same length. In this case there is more than one minimum connector for the network.

Summary

1. Introduction
2. Applications of Trees
3. Tree Traversal
4. Spanning Trees
5. Minimum Spanning Trees