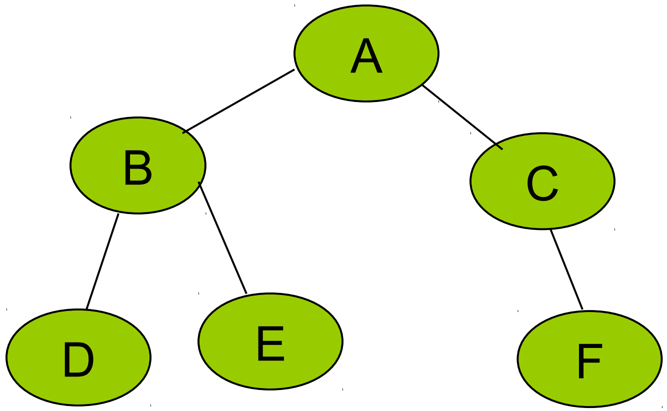
Minimum Spanning Tree

Presented by:
Hinal Lunagariya

TREE



- Connected acyclic graph
- Tree with n nodes contains exactly n-1 edges.

GRAPH

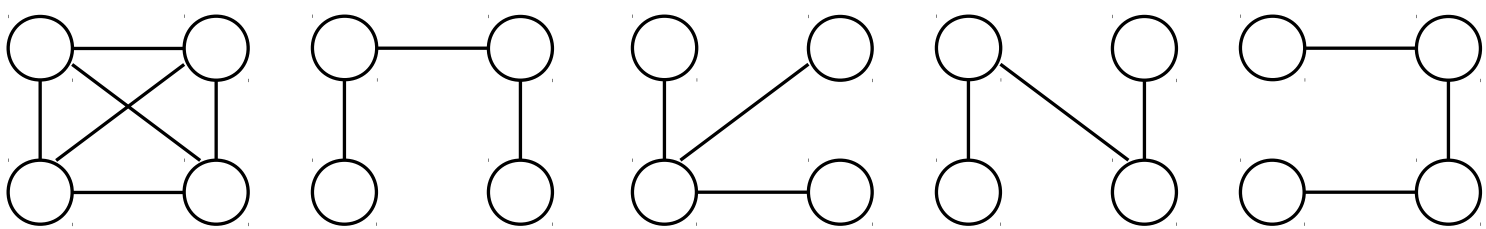
- Graph with n nodes contains less than or equal to n(n-1)/2 edges.

SPANNING TREE...

Suppose you have a connected undirected graph

- Connected: every node is reachable from every other node
- Undirected: edges do not have an associated direction

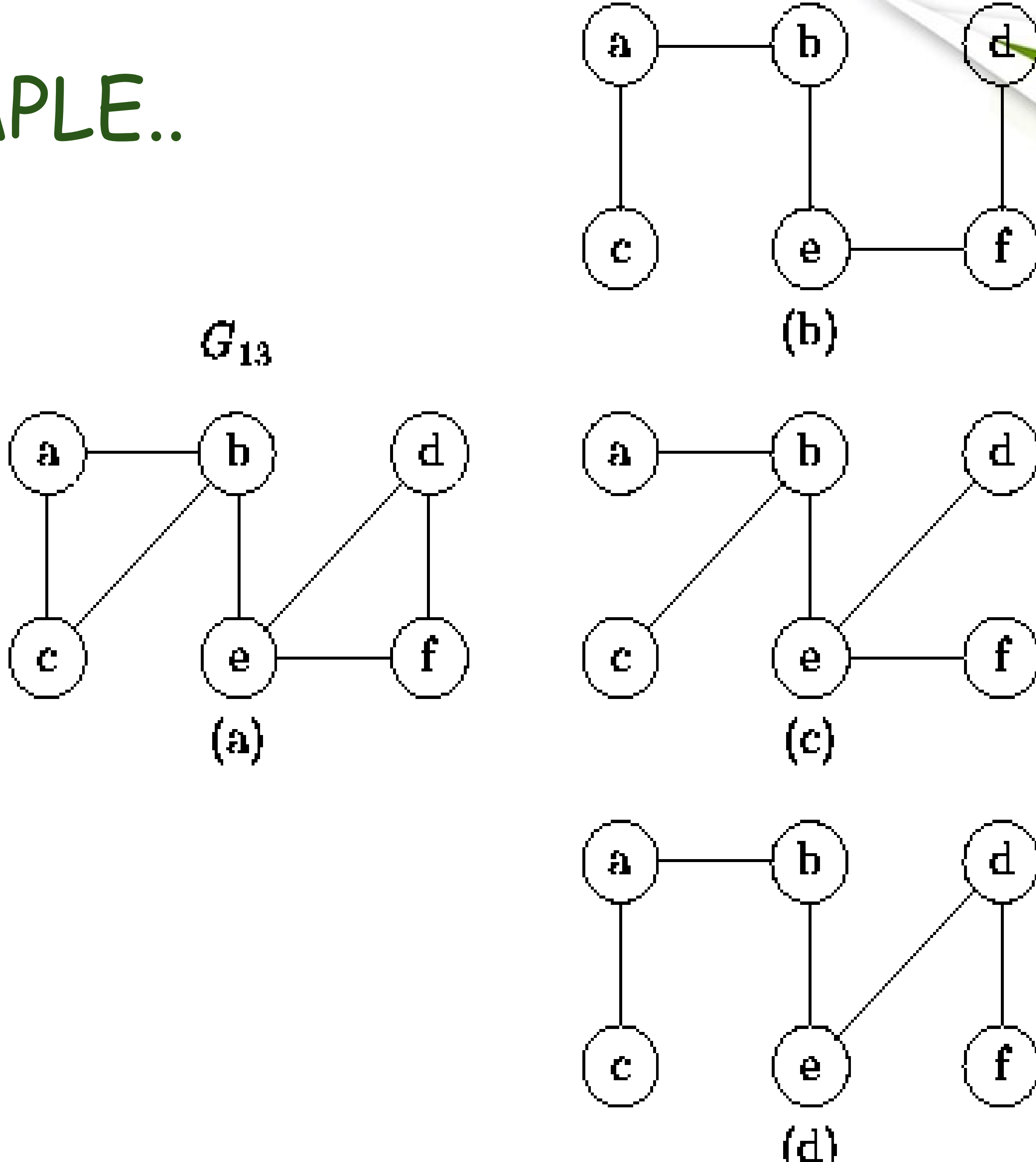
...then a spanning tree of the graph is a connected subgraph in which there are no cycles



A connected, undirected graph

Four of the spanning trees of the graph

EXAMPLE..



Minimizing costs

Suppose you want to supply a set of houses (say, in a new subdivision) with:

- electric power
- water
- sewage lines
- telephone lines

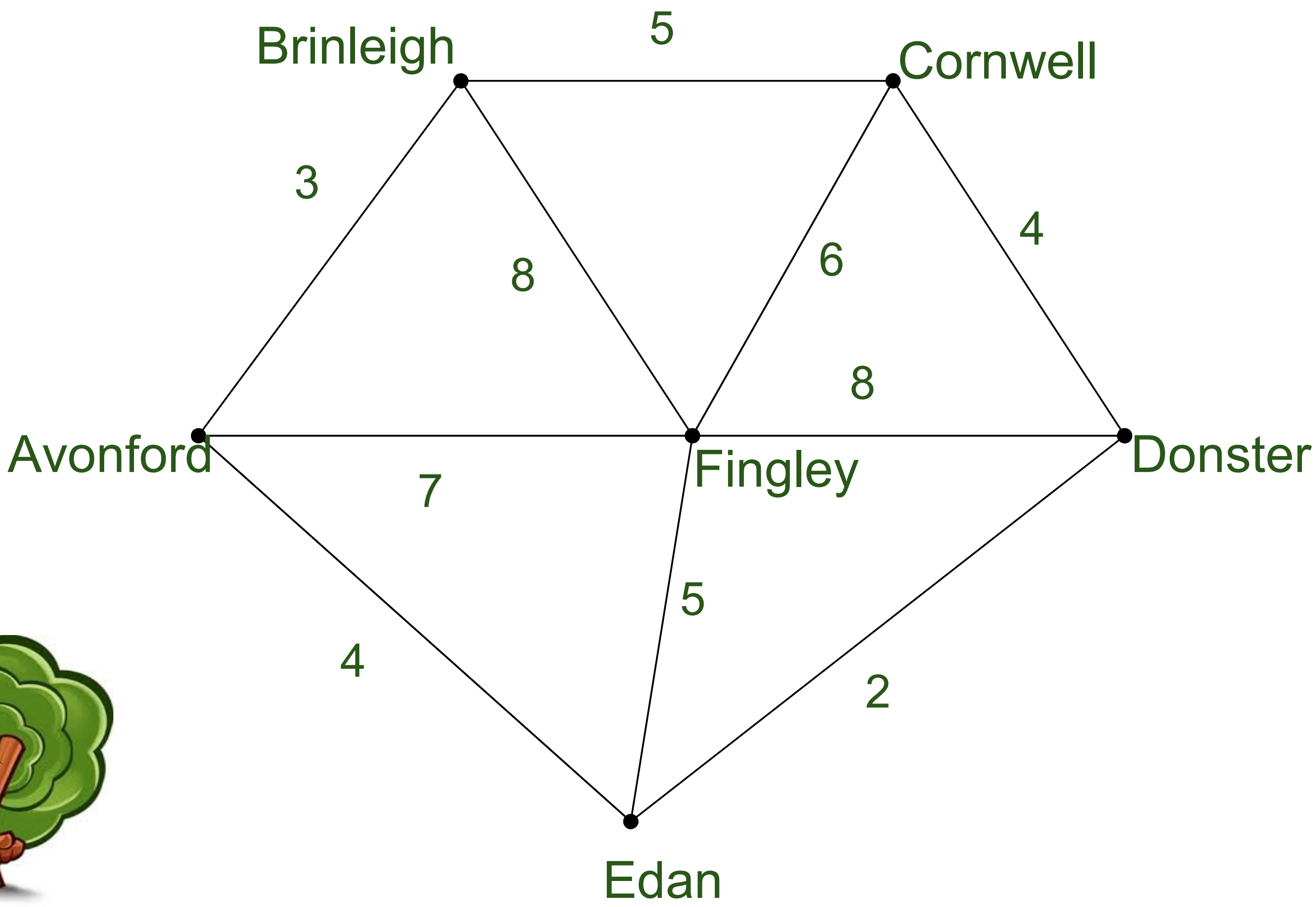
✓To keep costs down, you could connect these houses with a spanning tree (of, for example, power lines)

✓However, the houses are not all equal distances apart

✓To reduce costs even further, you could connect the houses with a *minimum-cost* spanning tree

Example

A cable company want to connect five villages to their network which currently extends to the market town of Avonford. What is the minimum length of cable needed?



MINIMUM SPANNING TREE

Let G = (N, A) be a connected, undirected graph where N is the set of nodes and A is the set of edges. Each edge has a given nonnegative length. The problem is to find a subset T of the edges of G such that all the nodes remain connected when only the edges in T are used, and the sum of the lengths of the edges in T is as small as possible possible. Since G is connected, at least one solution must exist.

Finding Spanning Trees

- There are two basic algorithms for finding minimum-cost spanning trees, and both are greedy algorithms

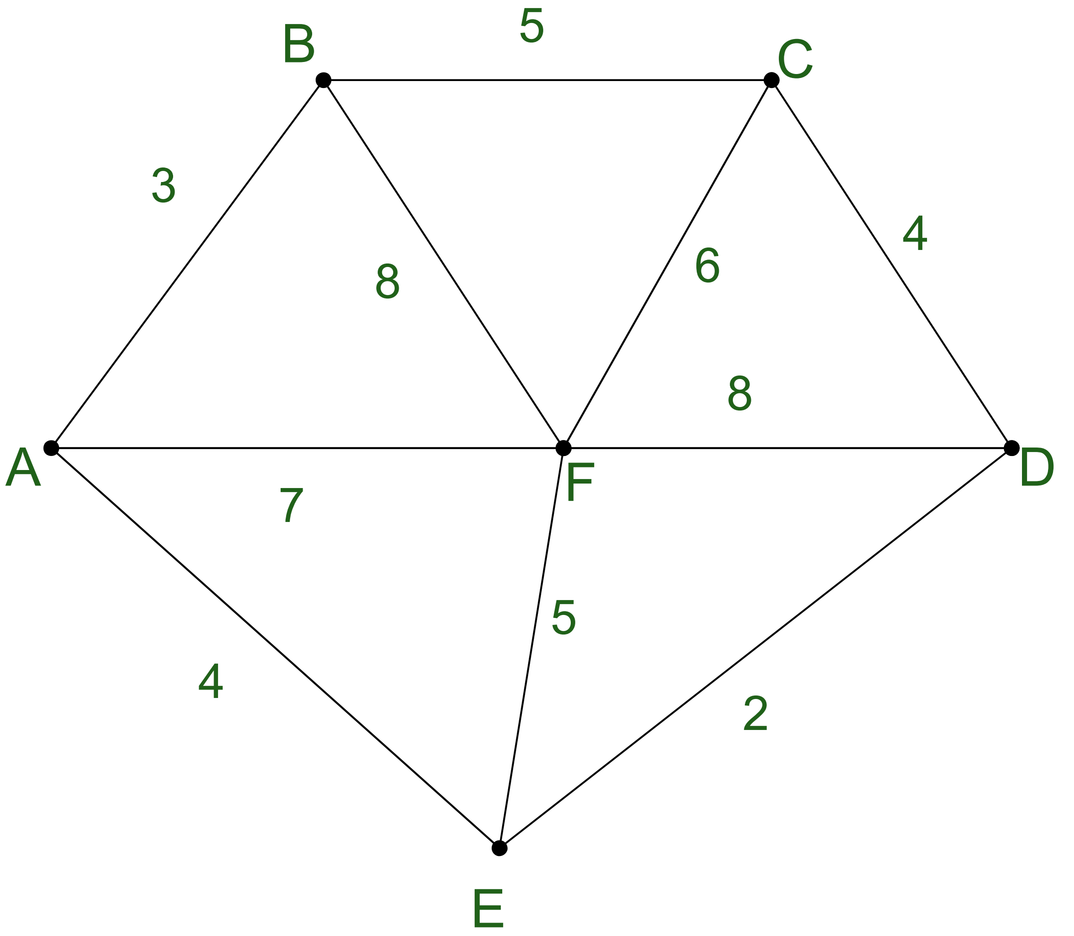
- **Kruskal's algorithm:**

  Created in 1957 by Joseph Kruskal

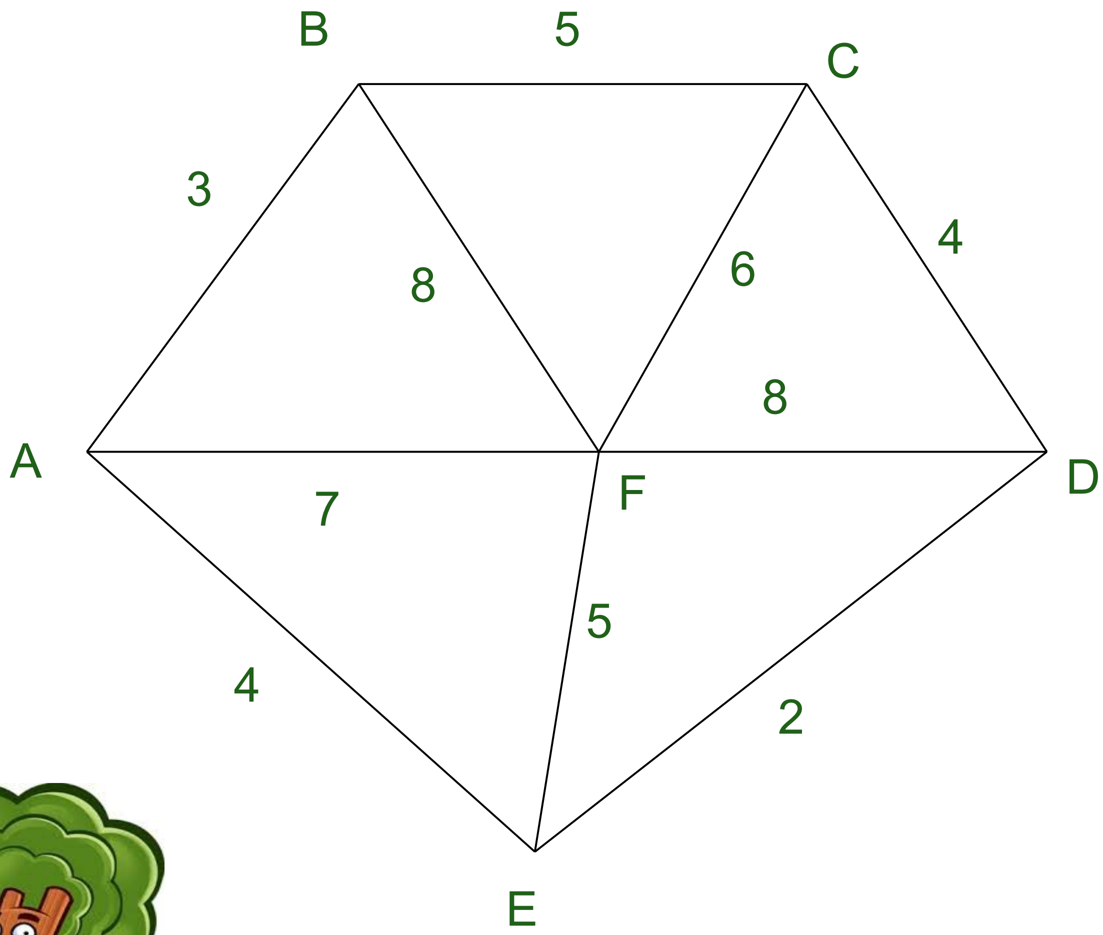
- **Prim's algorithm**

  Created by Robert C. Prim

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



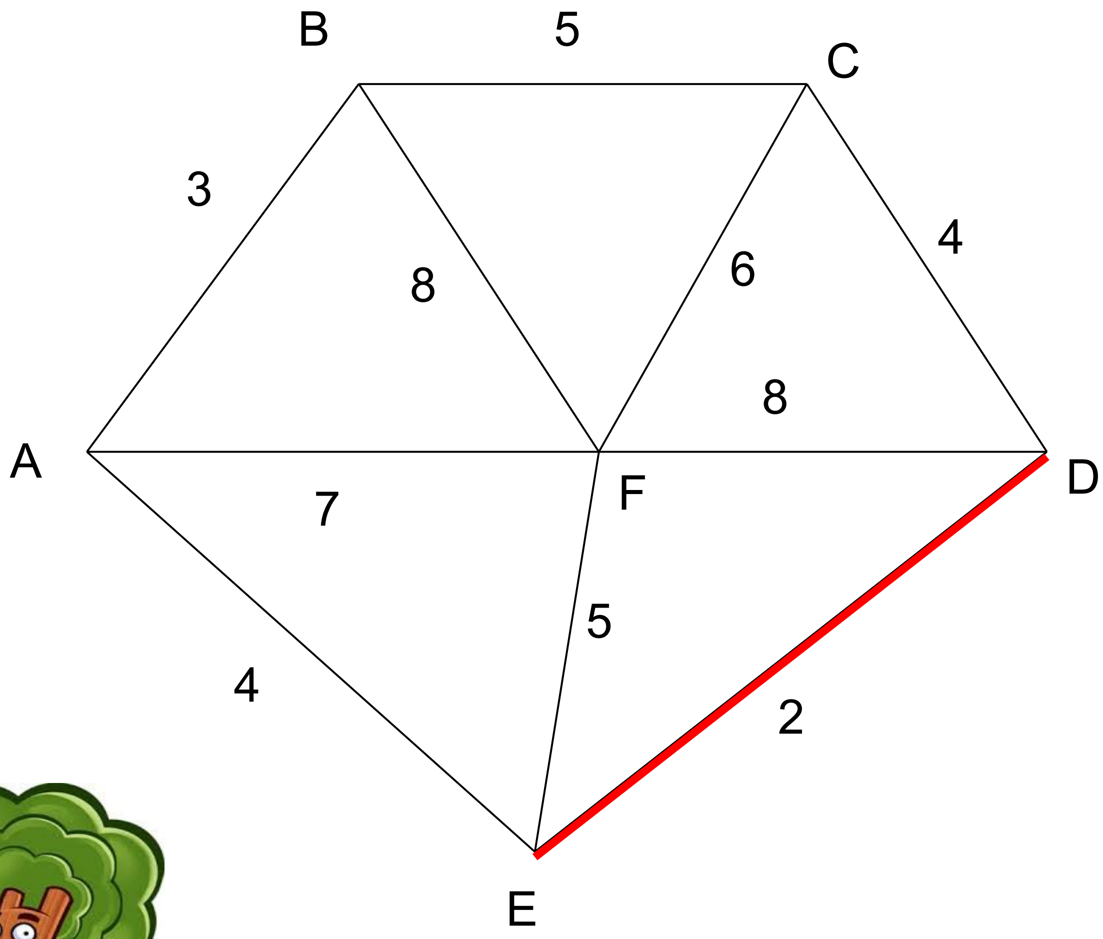
Kruskal's Algorithm



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

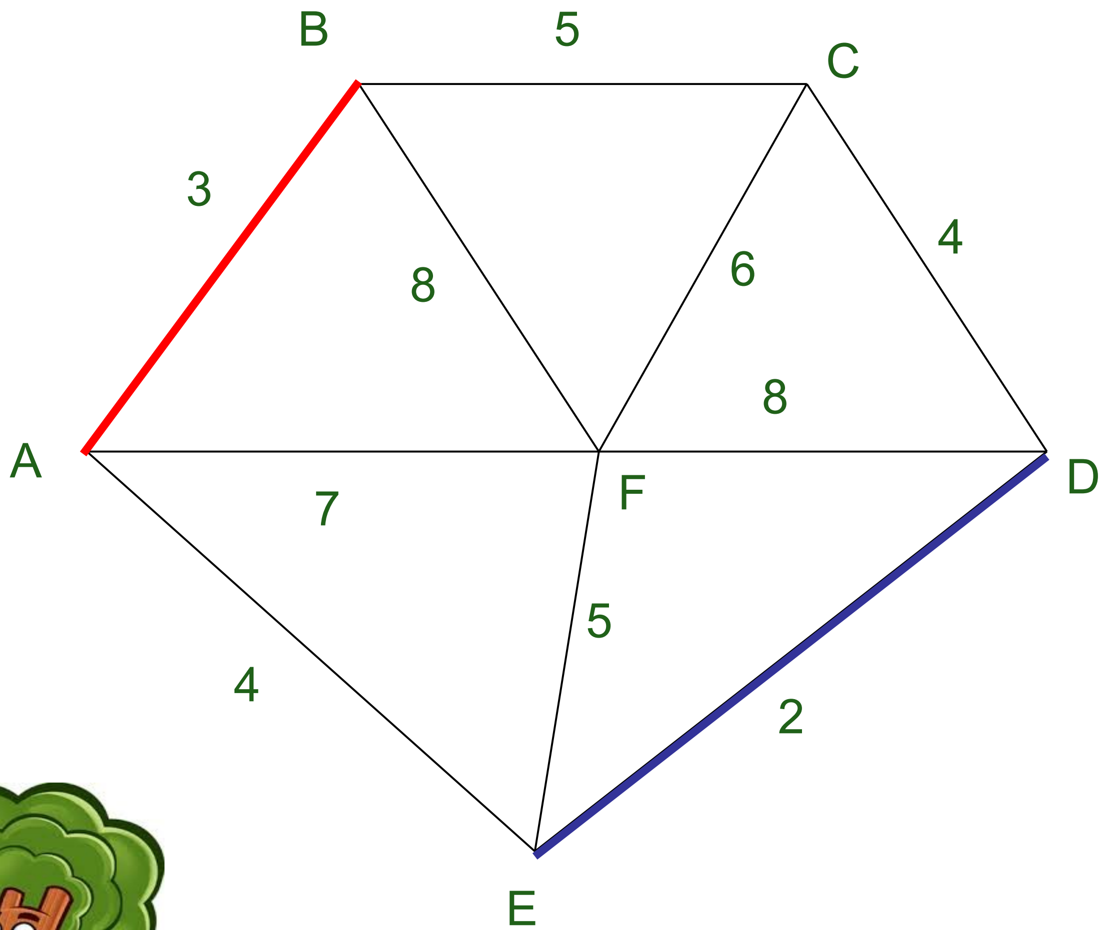
Kruskal's Algorithm



Select the shortest edge in the network

ED 2

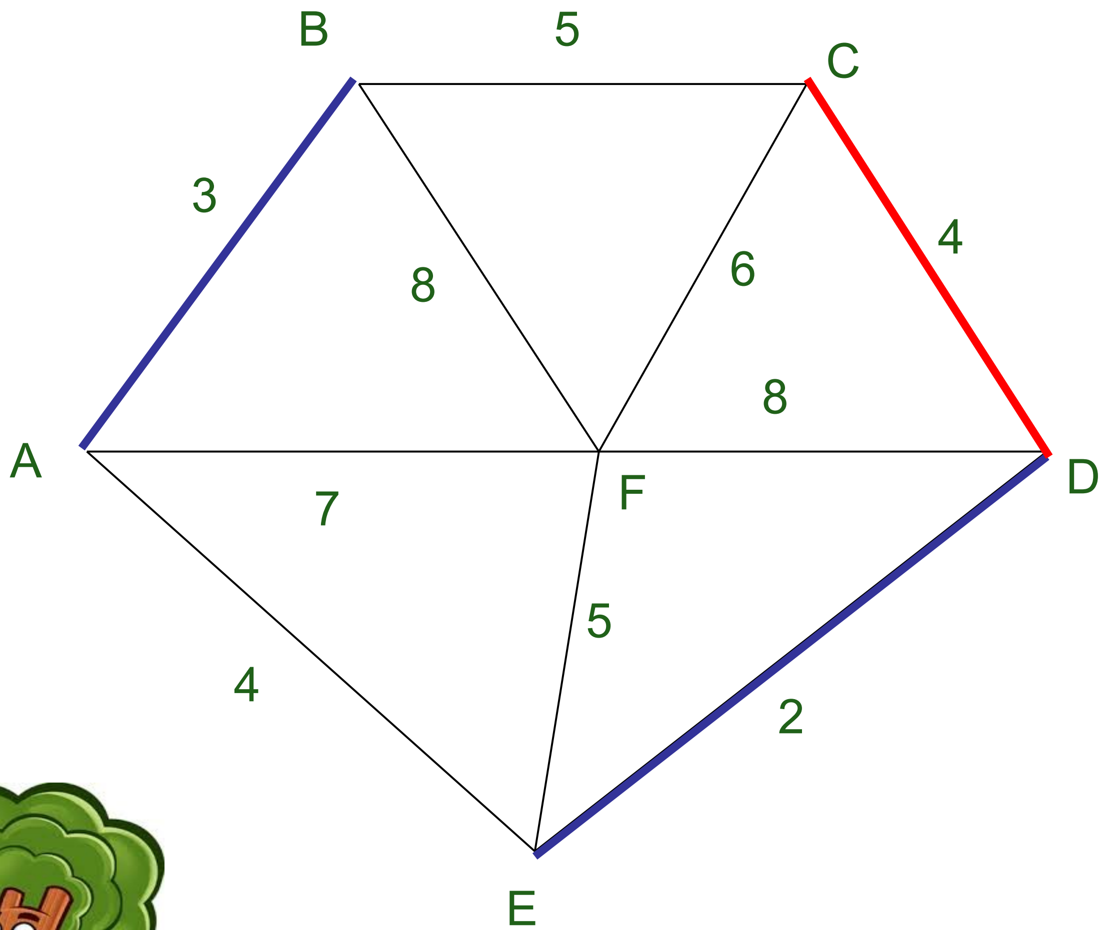
Kruskal's Algorithm



Select the next shortest edge which does not create a cycle

ED 2
AB 3

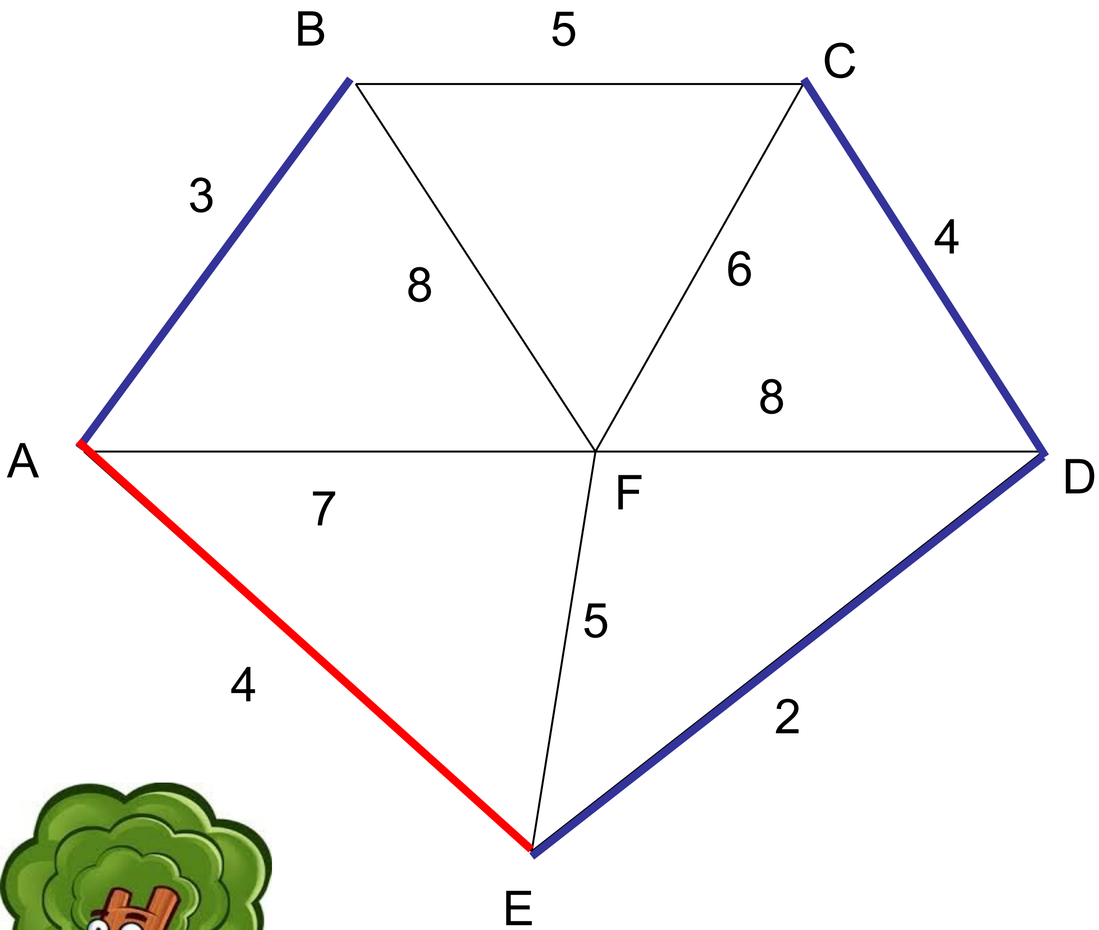
Kruskal's Algorithm



Select the next shortest
edge which does not create a cycle

ED 2
AB 3
CD 4 (or AE 4)

Kruskal's Algorithm



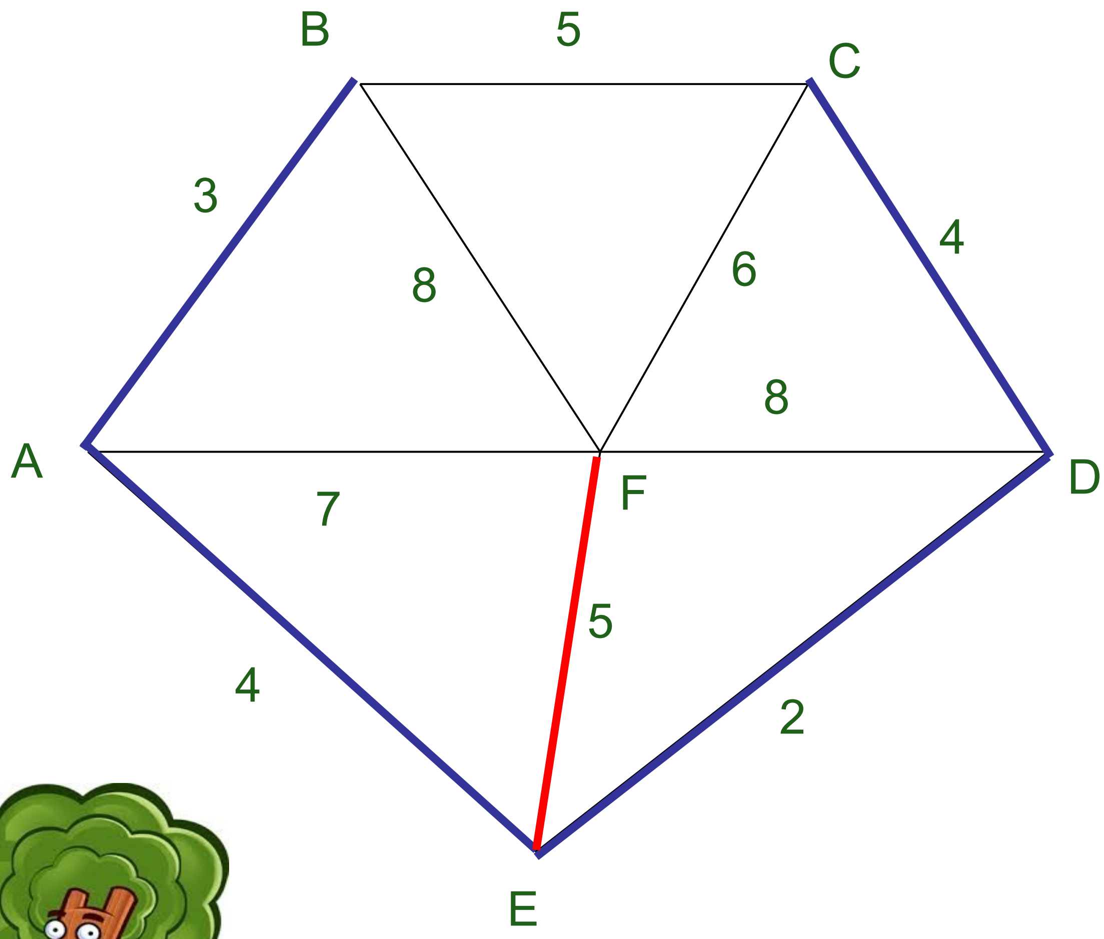
Select the next shortest edge which does not create a cycle

E

A

C

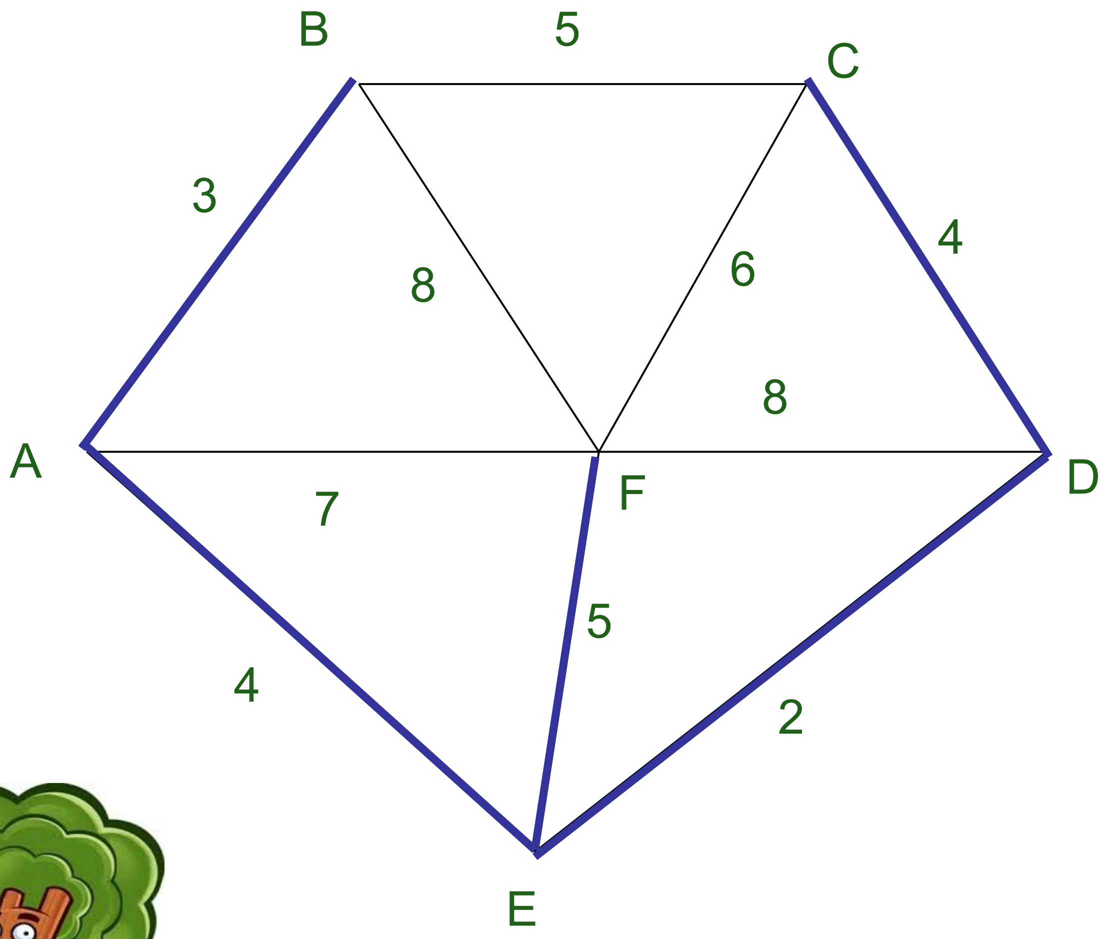
Kruskal's Algorithm



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

Algorithm

function Kruskal (G=(N,A): graph ; length : A→R^+):set of edges

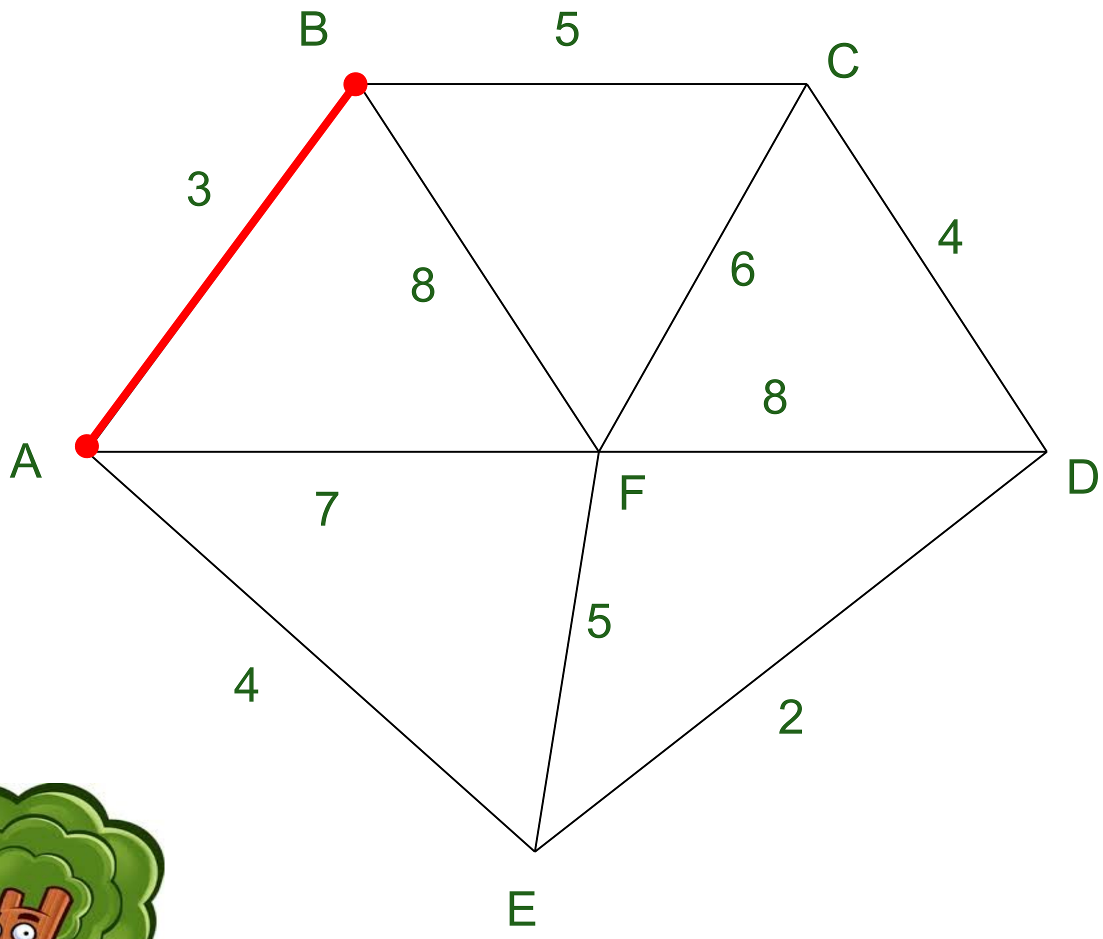
    {initialisation}
    sort A by increasing length
    N ← the number of nodes in N
    T ← Ø {will contain the edges of the minimum spanning tree}
    initialise n sets, each containing the different element of N
    {greedy loop}
    repeat
        e ← {u , v} ← shortest edge not yet considered
        ucomp ← find(u)
        vcomp ← find(v)
        if ucomp ≠ vcomp then
                merge(ucomp , vcomp)
                T ← T Ú {e}
    until T contains n-1 edges
    return T

Kruskal's Algorithm: complexity

Sorting loop:	O(a log n)
Initialization of components:	O(n)
Finding and merging:	O(a log n)

O(a log n)

Prim's Algorithm



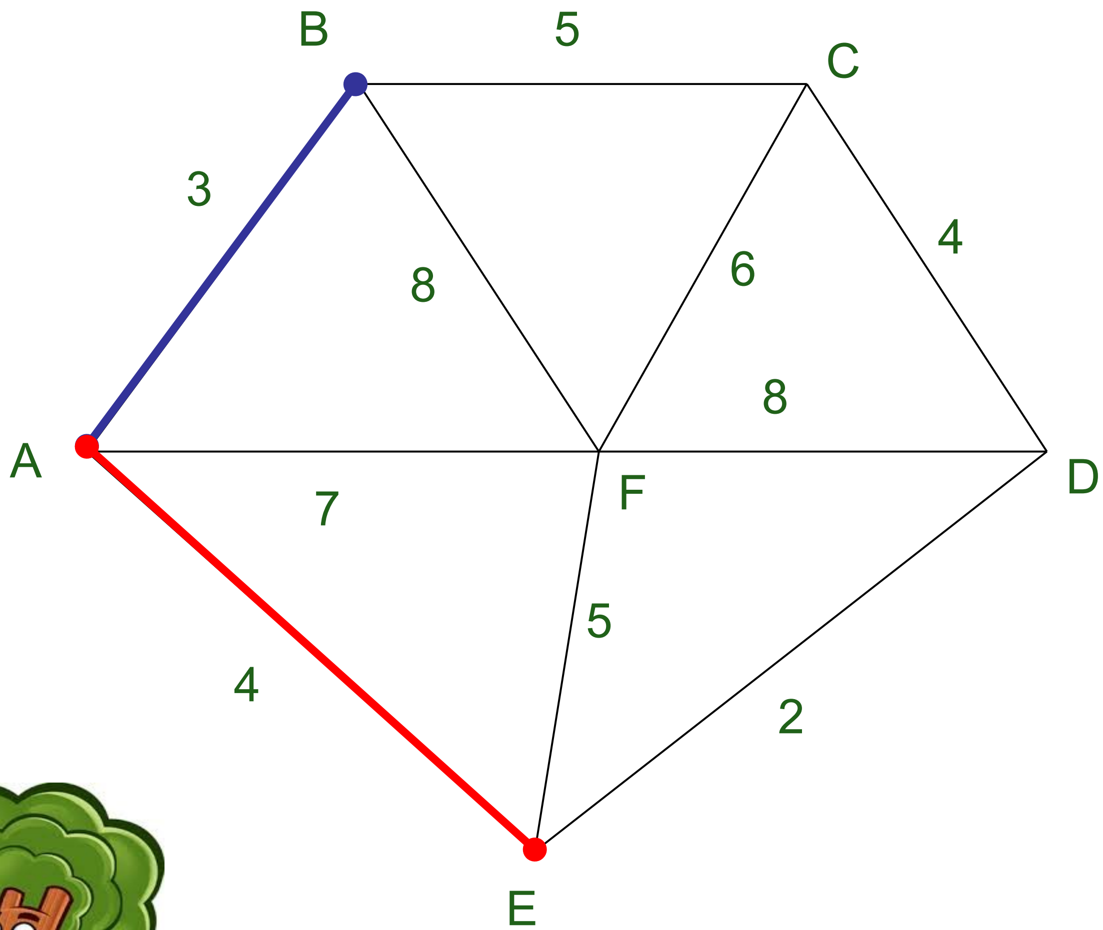
Select any vertex

A

Select the shortest edge connected to that vertex

AB 3

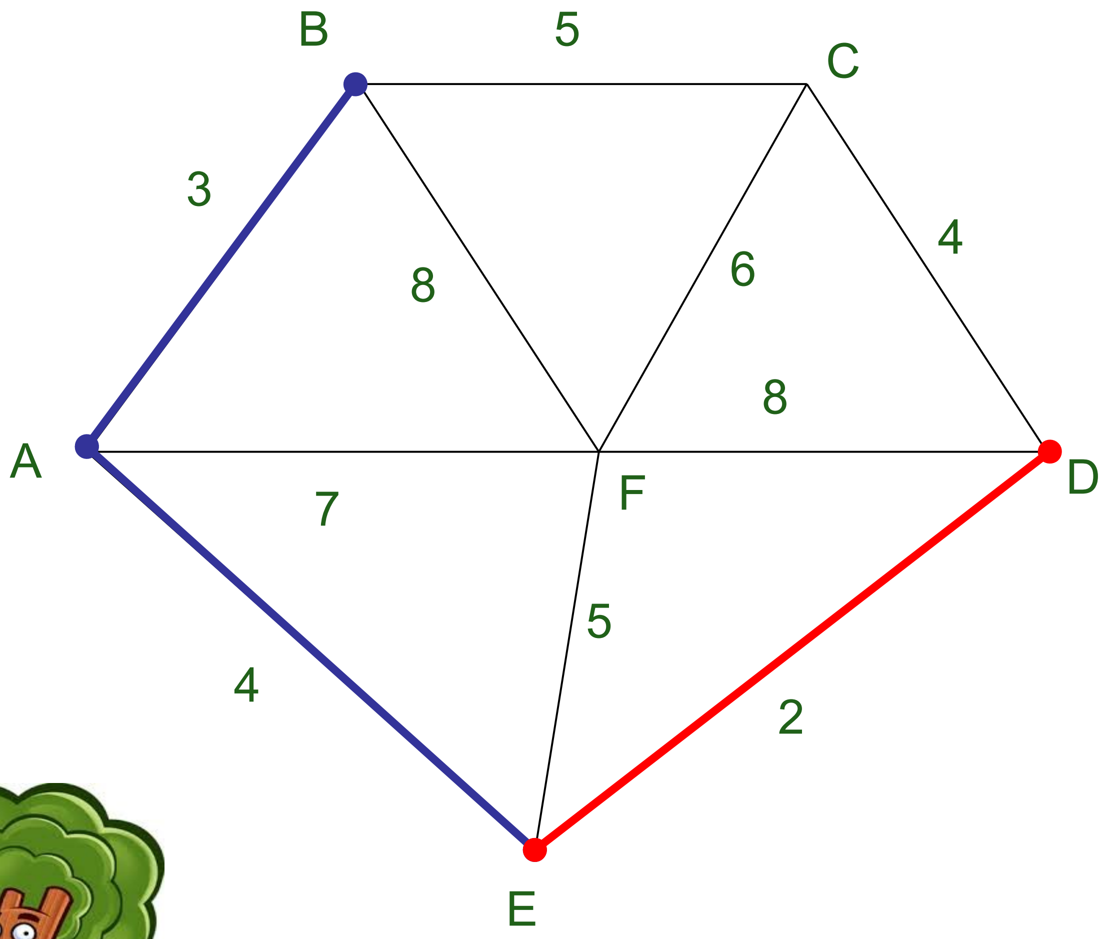
Prim's Algorithm



Select the shortest edge connected to any vertex already connected.

AE 4

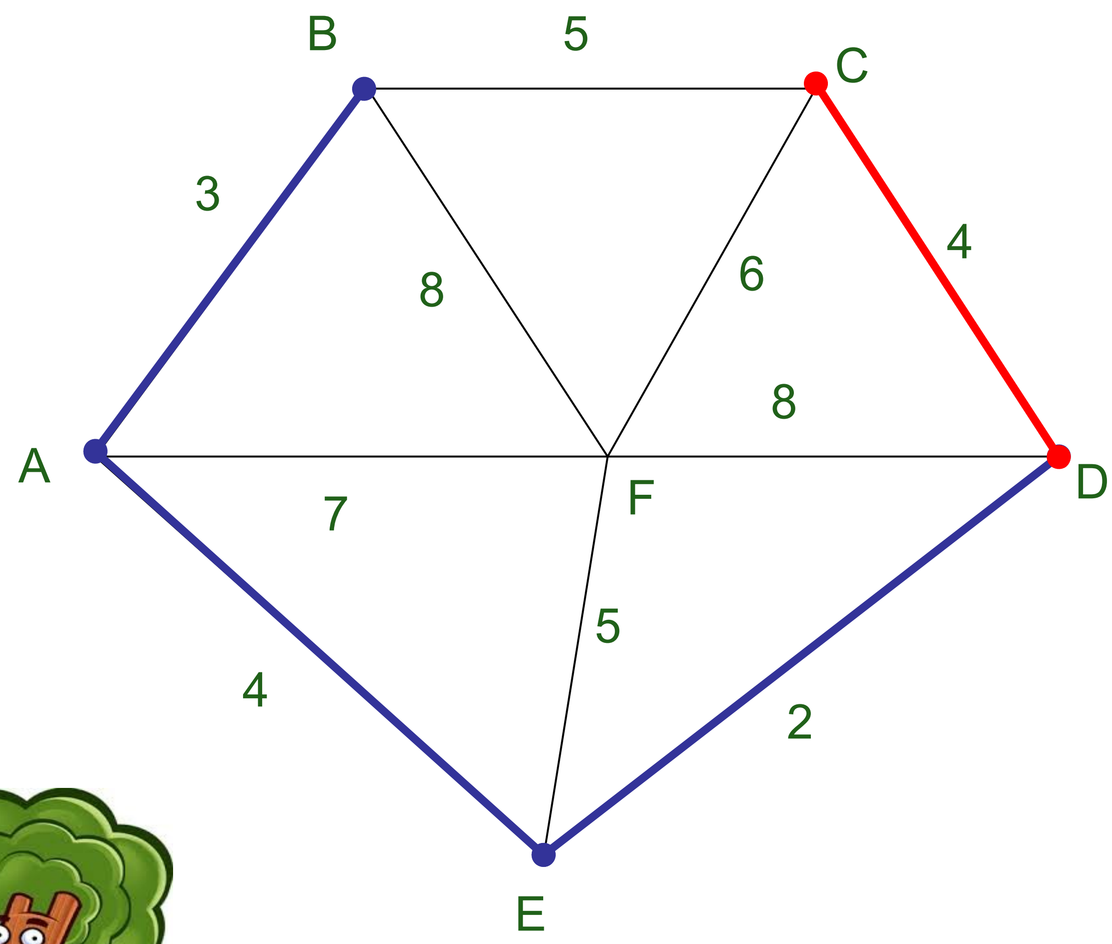
Prim's Algorithm



Select the shortest edge connected to any vertex already connected.

ED 2

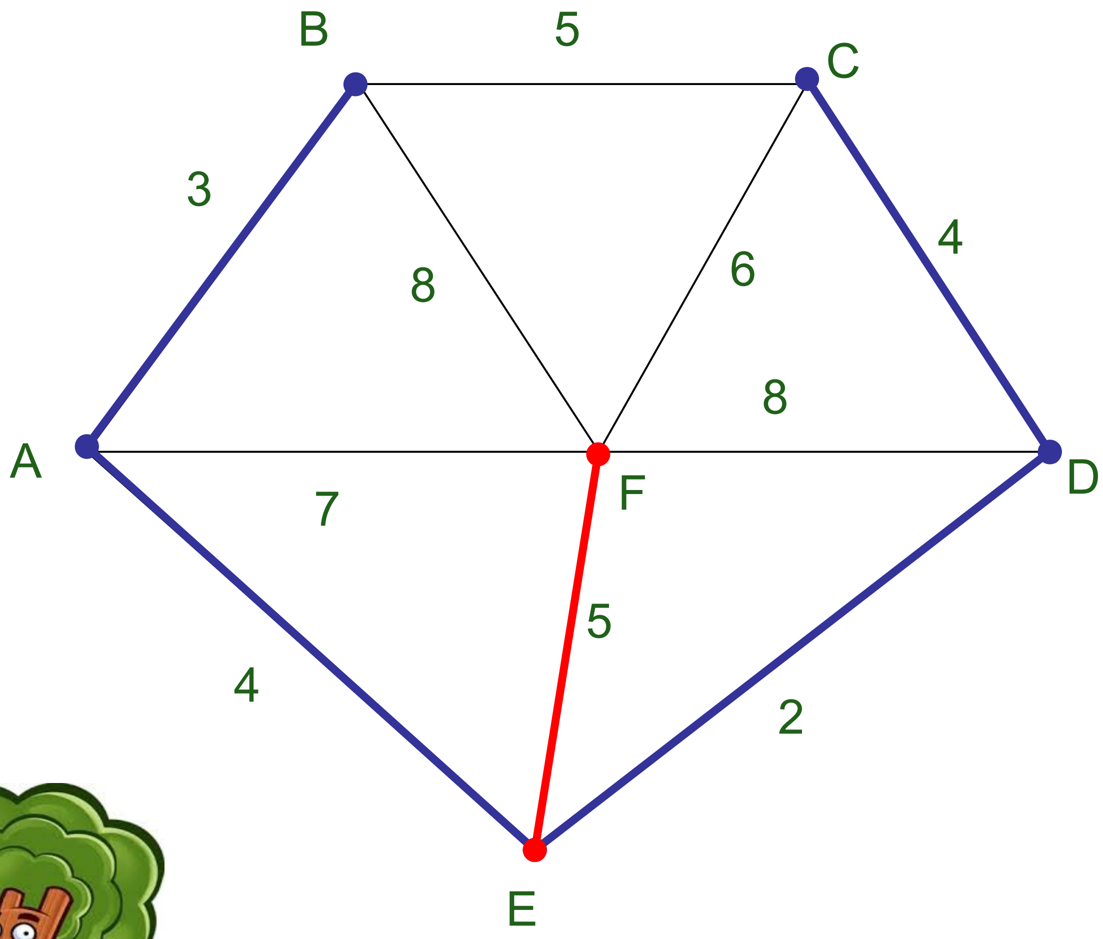
Prim's Algorithm



Select the shortest edge connected to any vertex already connected.

DC 4

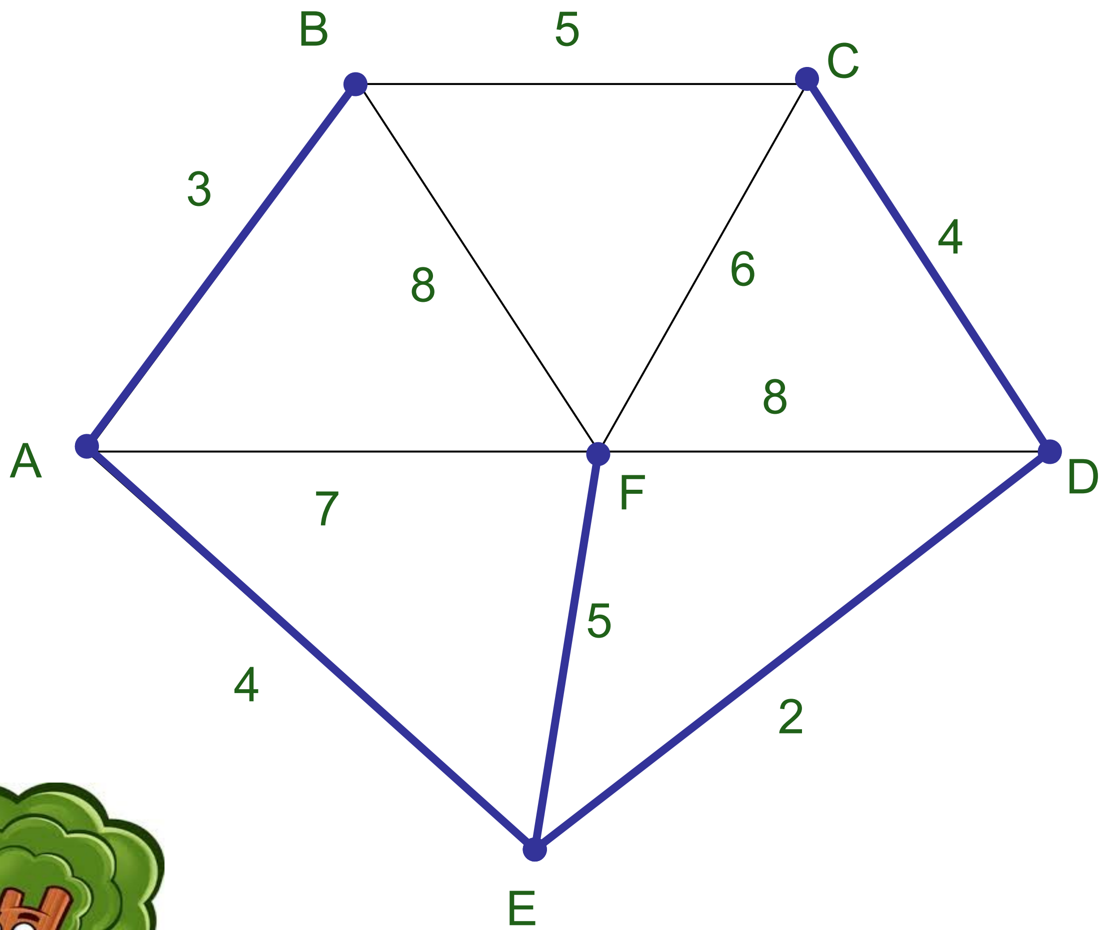
Prim's Algorithm



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

Prim's Algorithm

function Prim(G = <N,A>: graph ; length : A →R^+) : set of edges

{initialisation}

T ← Ø

B ← {an arbitrary member of N}

While B ≠ N do

find e = {u , v} of minimum length such

u € B and v € N \ B

T ←T ∪ {e}

B ← B ∪ {v}

Return T

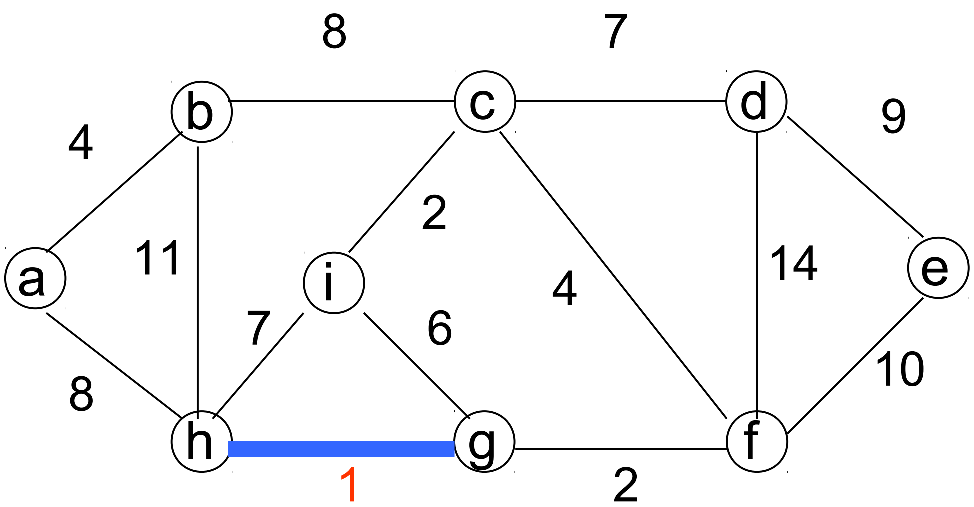
Complexity:

Outer loop: n-1 times

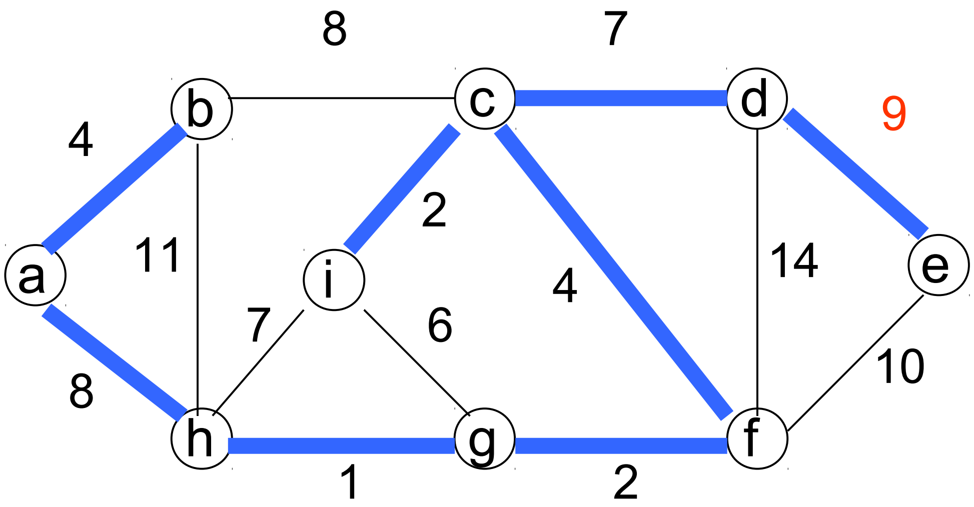
Inner loop: n times

$O(n^2)$

Example



Solution



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

Thank you!!