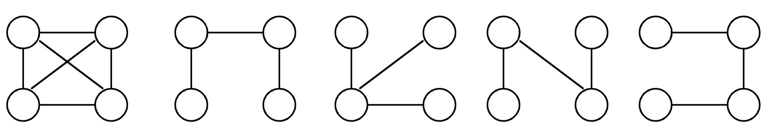
Spanning Trees

Spanning trees

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



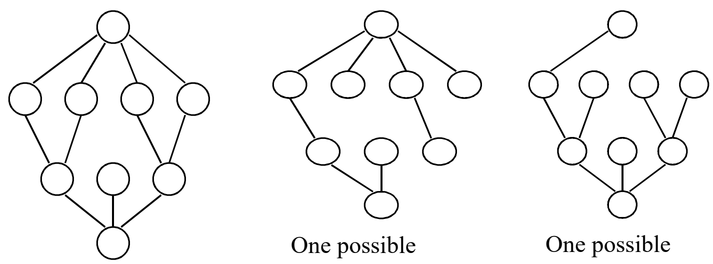
A connected, undirected graph

Four of the spanning trees of the graph

Finding a spanning tree

To find a spanning tree of a graph,

- pick an initial node and call it part of the spanning tree
- do a search from the initial node:
  - each time you find a node that is not in the spanning tree, add to the spanning tree both the new node *and* the edge you followed to get to it



An undirected graph

One possible result of a BFS starting from top

One possible result of a DFS starting from top

Given a graph G=(V,E), let T=(V, E') be a spanning tree of G. Here E' is a subset of E. How big is |E'|?

A. |E'| = |E|

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

C. |E'| = 2|V|

D. Depends on the graph G

E. None of the above

Given a connected undirected graph G=(V,E) and its spanning tree T=(V,E'), let (u,v) be an edge in E that is not in E'. If we add (u,v) to E', then

A. T will have a cycle
B. T will be a new spanning tree of G
C. T will no longer be connected
D. Ensure that T is connected
E. None of the above

Given a connected undirected graph G=(V,E) and its spanning tree T=(V,E'), let (u,v) be an edge in E'. If we remove (u,v) from E', then

A. T will have a cycle
B. T will be a new spanning tree of G
C. T will no longer be connected
D. Ensure that T is connected
E. None of the above

Properties of Spanning Trees

A connected undirected graph G can have more than one spanning tree.

All spanning trees of G have the same number of vertices and edges.

A spanning tree does not have any cycles.

Removing one edge from a spanning tree T will make T disconnected (a spanning tree is minimally connected).

Adding an edge to a spanning tree T will create a cycle in T (a spanning tree is maximally acyclic).

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

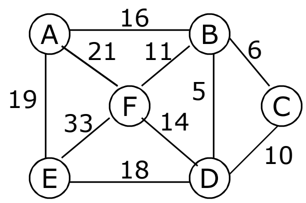
To reduce costs even further, you could connect the houses with a *minimum-cost* spanning tree

Minimum-cost spanning trees

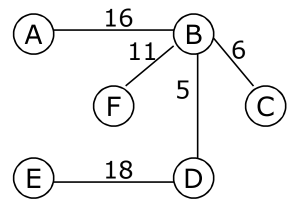
Suppose you have a connected undirected graph with a weight (or cost) associated with each edge

The cost of a spanning tree would be the sum of the costs of its edges

A minimum-cost spanning tree is a spanning tree that has the lowest cost



A connected, undirected graph



A minimum-cost spanning tree

Key Property of Minimum Spanning Trees

Let G=(V,E) be an undirected connected graph

Let X be a subset of V. Then V-X is the set of vertices in V that are not in X.

Let (u,v) be an edge where u is in X and v is in G-X. So (u,v) connects X to G-X.

Let (u,v) be the smallest edge connecting X to G-X.

Claim: (u, v) is in the minimum spanning tree of G

Proof by contradiction

Let T =(V,E’) be a spanning tree of G and the edge e=(u,v) is not in T.

Need to show that T is not a minimum spanning tree.

Since T is a spanning tree, it contains a unique path from u to v.

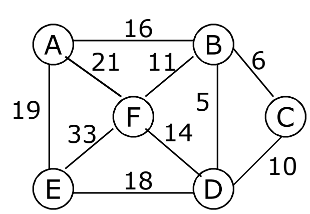
This path has to include another edge f, connecting X to G-X. If we add edge e to E’, then T will have a cycle: edges f and e will be part of that cycle. If we remove edge f, we will break the cycle and will have a tree again.

So We can construct another spanning tree, T’ = (V, E’- {f} U {e})

T’ has a lower weight. Therefore, T is not a minimum spanning tree. This means that e=(u,v) must be in the minimum spanning tree.

Which edge must be in the MST of the graph below

A. (D, C)
B. (F, D)
C. (B, D)
D. (A, E)
E. (E, F)



Finding spanning trees

Kruskal's algorithm: Start with *no* nodes or edges in the spanning tree, and repeatedly add the cheapest edge that does not create a cycle

- Here, we consider the spanning tree to consist of edges only

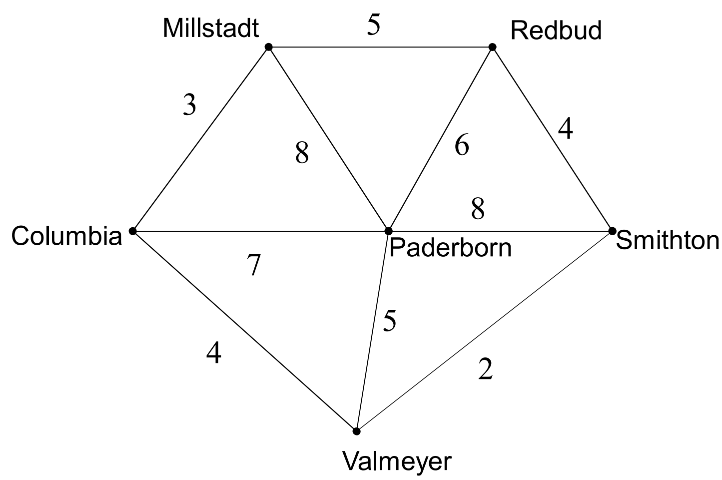
Prim's algorithm: Start with any *one node* in the spanning tree, and repeatedly add the cheapest edge, and the node it leads to, for which the node is not already in the spanning tree.

- Here, we consider the spanning tree to consist of both nodes and edges

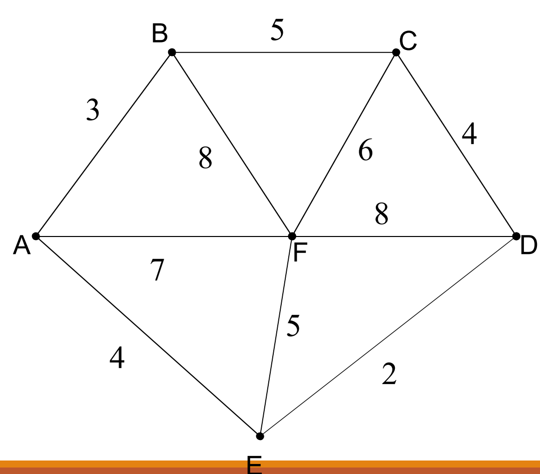
Kruskal's algorithm

```
KRUSKAL_MST(V, E)
 sort E in increasing order by weight
 let T = (V', E') initially be empty
 for each edge e in E, in the sorted
order:
  ◦ if endpoints of e are disconnected in T:
    ◦ Add e to E'
    ◦ Add endpoints of e to V'
 return T
```

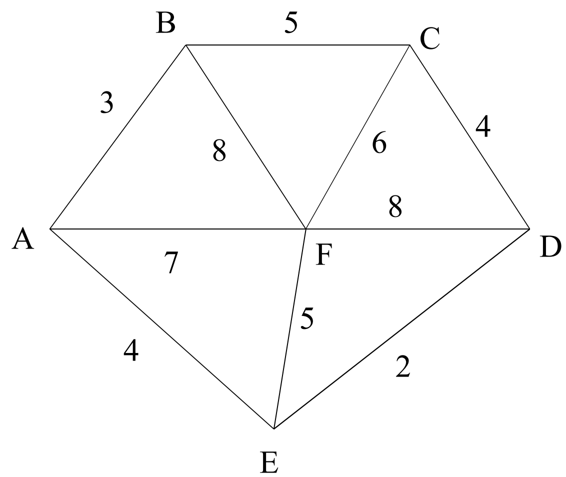
Example: A cable company want to connect five villages to their network which currently extends to the market town of Columbia. What is the minimum length of cable needed?



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



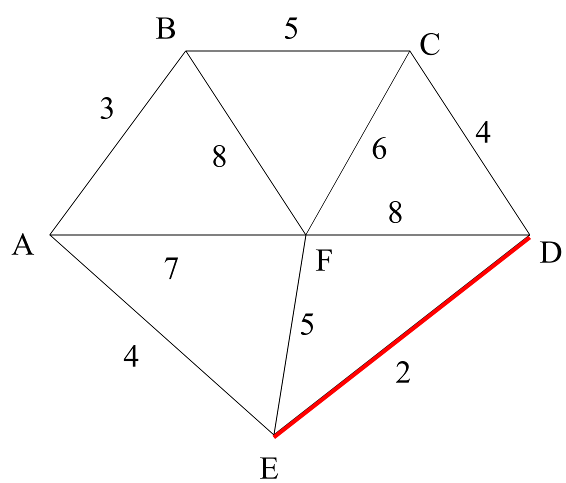
Kruskal's Algorithm



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

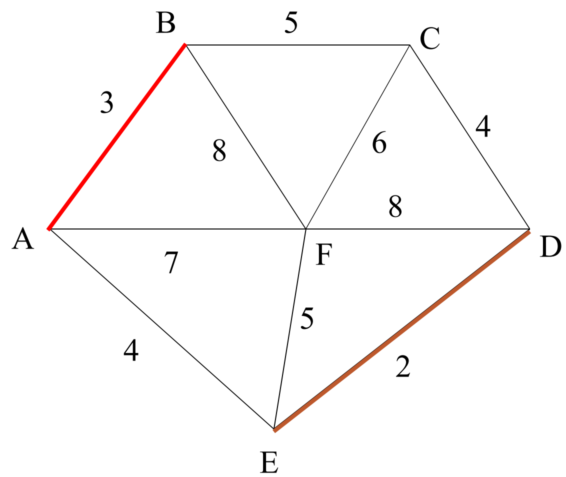
Kruskal's Algorithm



Select the shortest edge in the network

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

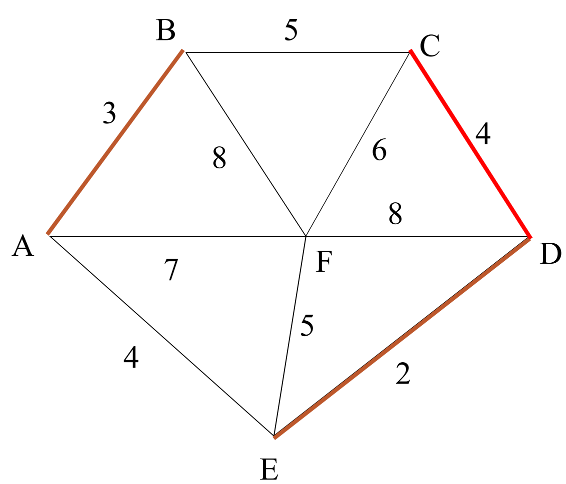
Kruskal's Algorithm



Select the next shortest edge which does not create a cycle

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

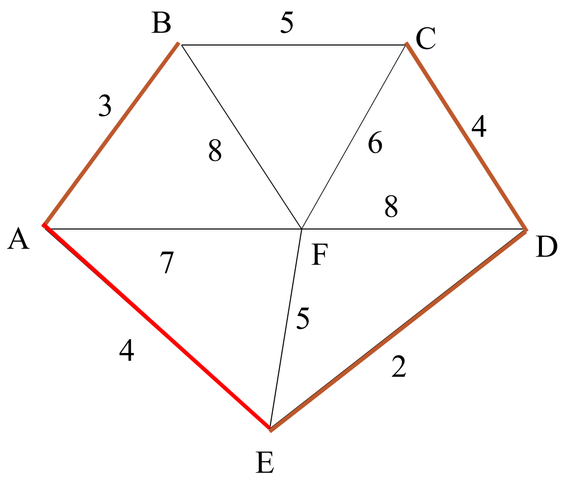
Kruskal's Algorithm



Select the next shortest edge which does not create a cycle

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

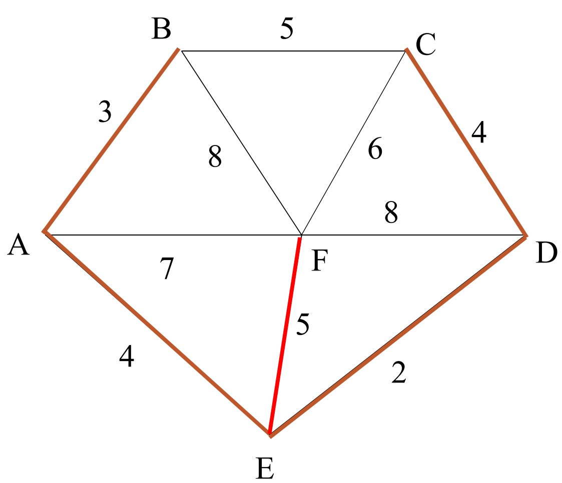
Kruskal's Algorithm



Select the next shortest edge which does not create a cycle

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

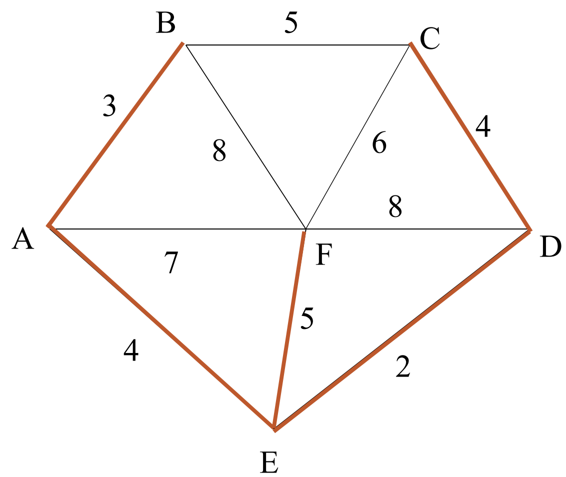
Kruskal's Algorithm



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
CF 6
AF 7
BF 8
CF 8

Kruskal's Algorithm



All vertices have been connected.

The solution is

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

Total weight of tree: 18

What is the asymptotic complexity of KRUSKAL_MST?

```
KRUSKAL_MST(V, E)
 sort E in increasing order by weight
 let T = (V', E') initially be empty
 for each edge e in E, in the sorted order:
 ◦ if endpoints of e are disconnected in T:
   ◦ Add e to E'
   ◦ Add endpoints of e to V'
 return T
```

A. Θ(|E| lg|E|)
B. Θ(|V| lg |V|)
C. Θ(|V| + |E|)
D. $\Theta(|V|^2)$
E. $\Theta(|E|^2)$