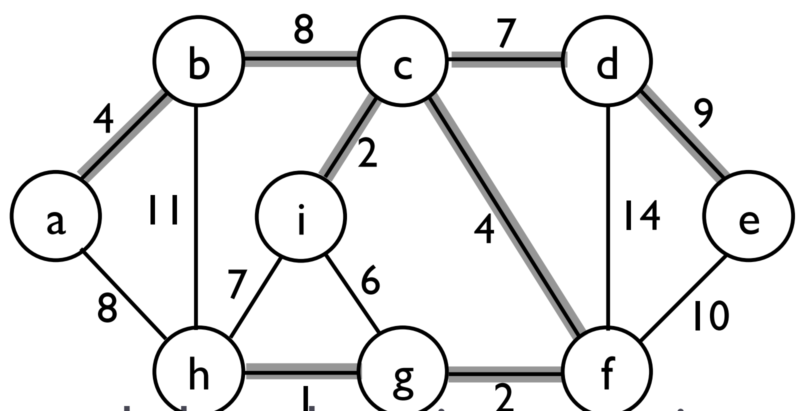
Graph Algorithms-III

Minimum Spanning Tree-Kruskal, Prim’s Algorithm

Minimum Spanning Trees

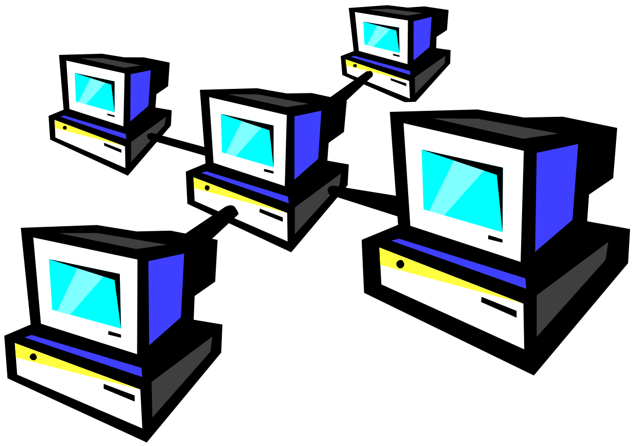
- Spanning Tree
  - A tree (i.e., connected, acyclic graph) which contains all the vertices of the graph
- Minimum Spanning Tree
  - Spanning tree with the **minimum sum of weights**

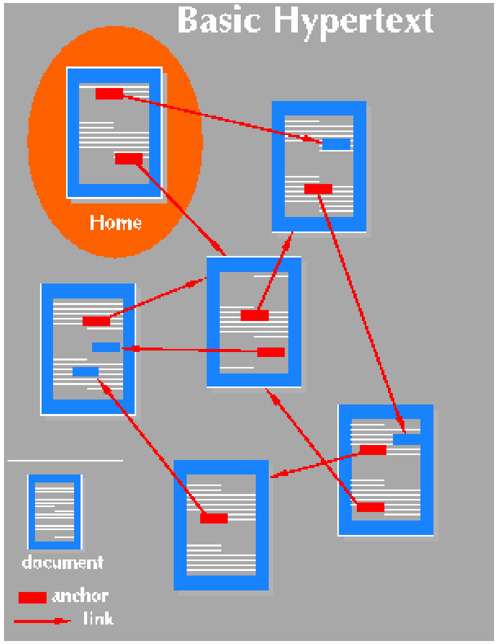


- Spanning forest
  - If a graph is not connected, then there is a spanning tree for each connected component of the graph

Applications of MST

- Find the least expensive way to connect a set of cities, terminals, computers, etc.

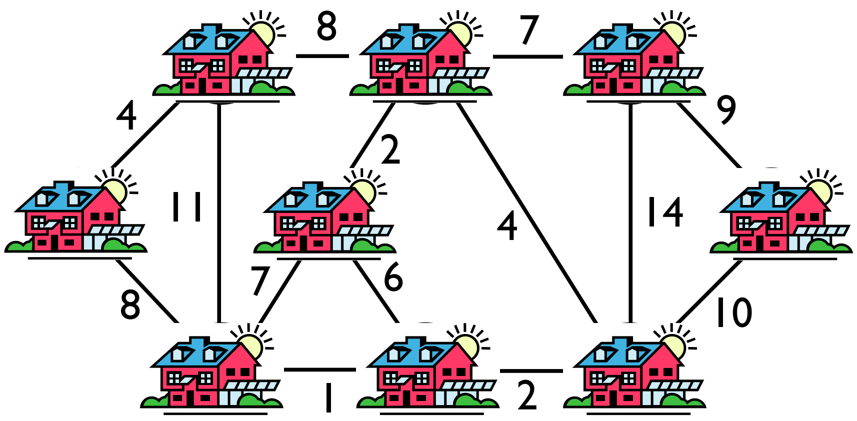




Example

Problem

- A town has a set of houses and a set of roads
- A road connects 2 and only 2 houses
- A road connecting houses u and v has a repair cost w(u, v)



Goal: Repair enough (and no more) roads such that:

1. Everyone stays connected

   i.e., can reach every house from all other houses

2. Total repair cost is minimum

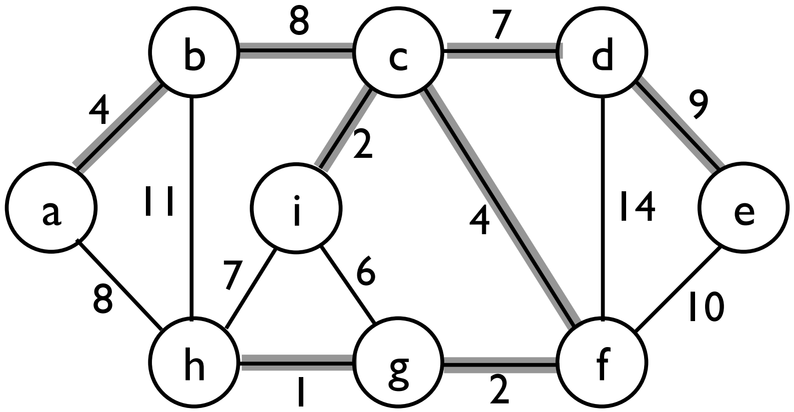
Properties of Minimum Spanning Trees

- Minimum spanning tree is **not** unique

- MST has no cycles – see why:
  - We can take out an edge of a cycle, and still have the vertices connected while reducing the cost
- # of edges in a MST:
  - |V| - 1

Growing a MST – Generic Approach

- Grow a set A of edges (initially empty)
- Incrementally add edges to A such that they would belong to a MST

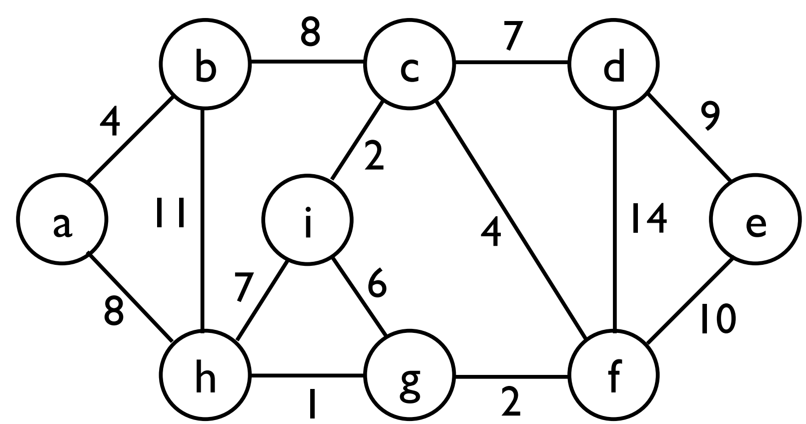


Idea: add only "safe" edges

– An edge (u, v) is **safe** for A if and only if $A \cup \{(u, v)\}$ is also a subset of **some** MST

Generic MST algorithm

1. $A \leftarrow \varnothing$
2. **while** A is not a spanning tree
3. **do** find an edge (u, v) that is safe for A
4. $A \leftarrow A \cup \{(u, v)\}$
5. **return** A



- How do we find safe edges?

Minimum Connector Algorithms

Kruskal's algorithm

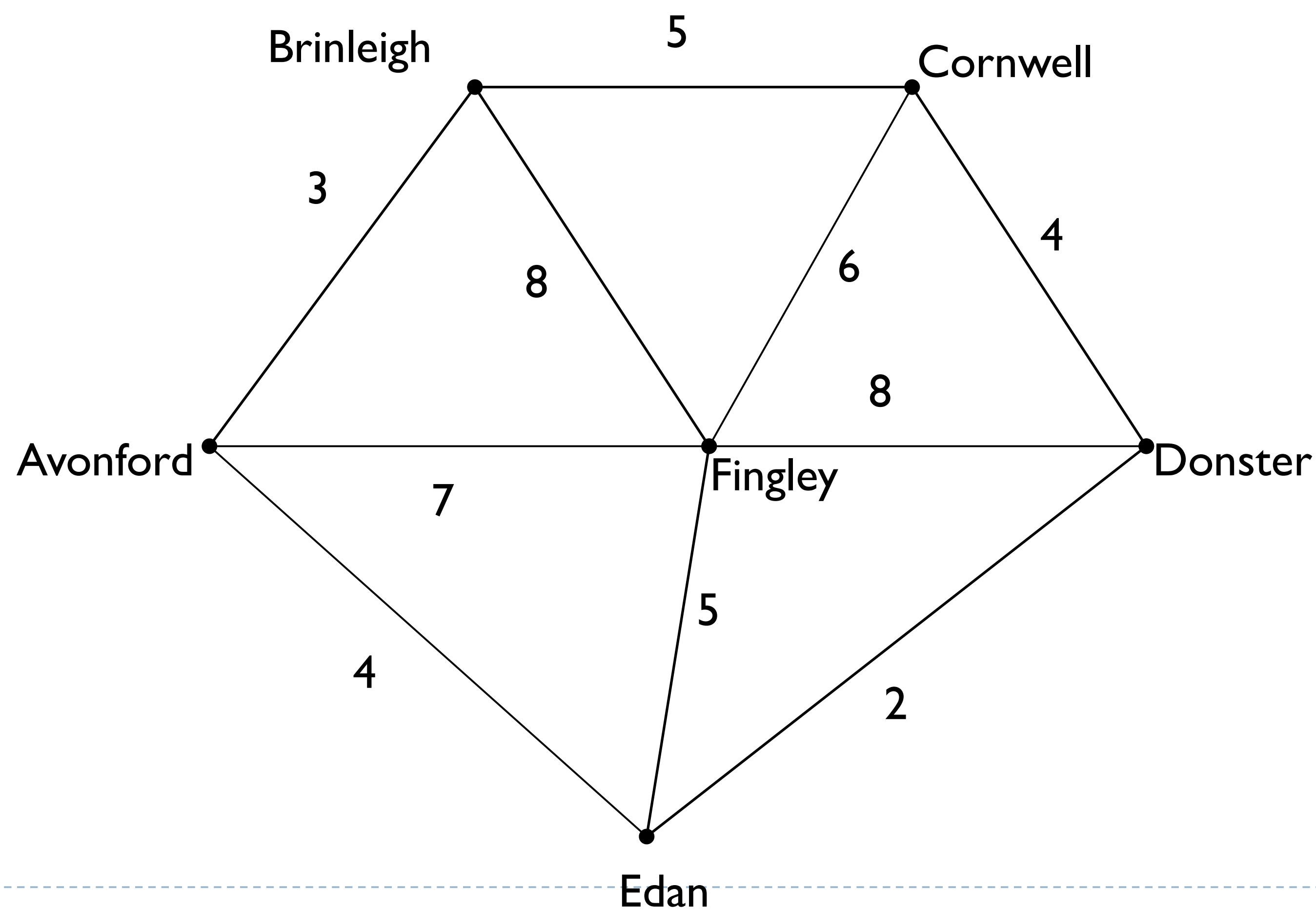
1. Select the shortest edge in a network
2. Select the next shortest edge which does not create a cycle
3. Repeat step 2 until all vertices have been connected

Prim's algorithm

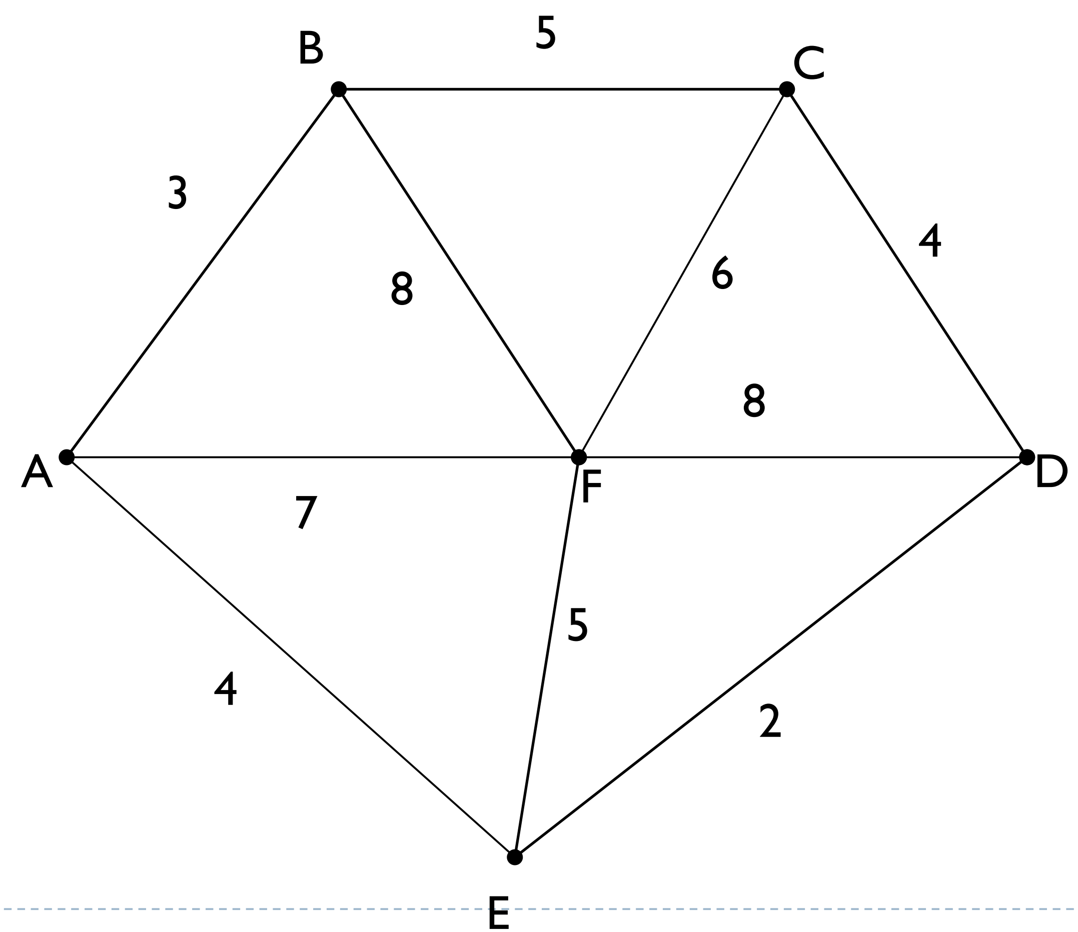
1. Select any vertex
2. Select the shortest edge connected to that vertex
3. Select the shortest edge connected to any vertex already connected
4. Repeat step 3 until all vertices have been connected

Example

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?



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



Kruskal Algorithm

Running time: O(V+ElgE+ElgV)=O(ElgE) or O(VlgE), if E<=V

```
MST-KRUSKAL(G, w)
A = ∅
for each vertex v ∈ G.V
    MAKE-SET(v)
sort the edges of G.E into nondecreasing order by weight w
for each edge (u, v) ∈ G.E, taken in nondecreasing order by weight
    if FIND-SET(u) ≠ FIND-SET(v)
        A = A ∪ {(u, v)}
        UNION(u, v)
return A
```

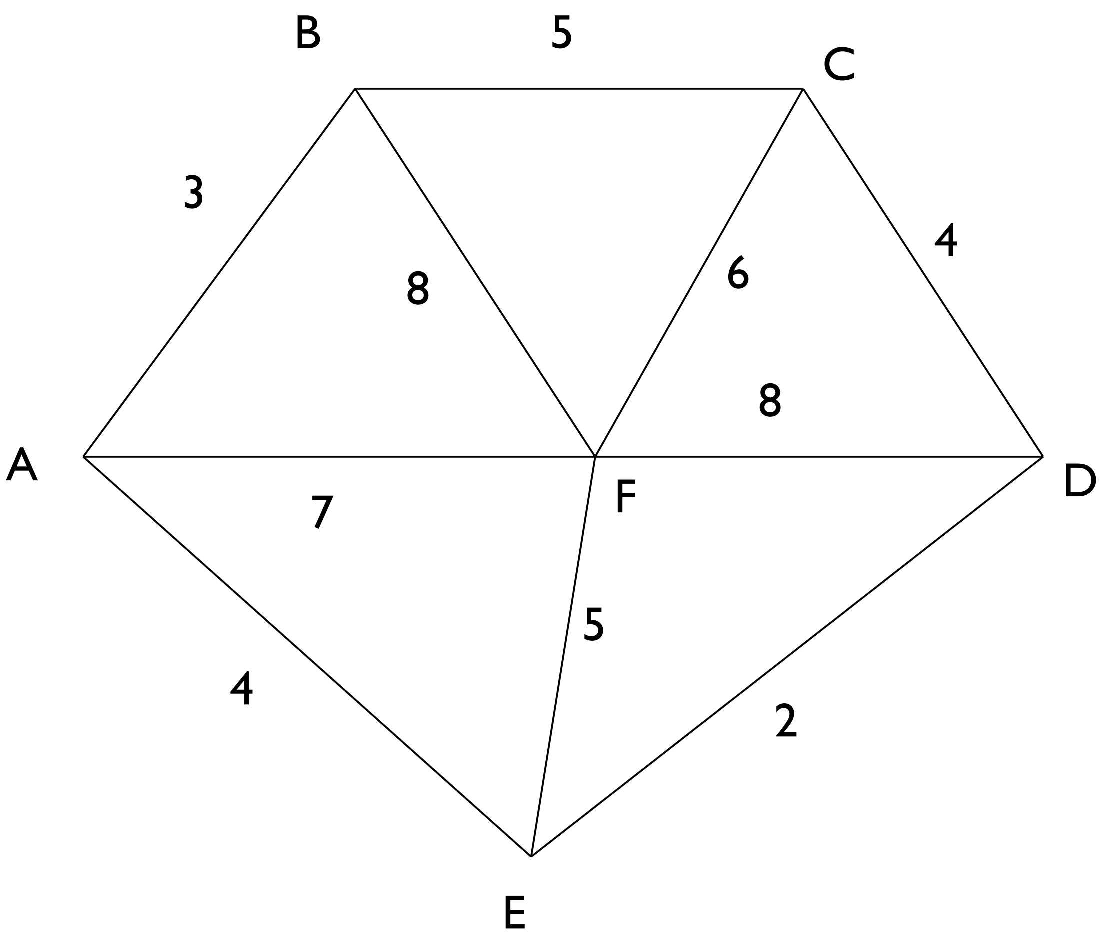
O(V)

O(ElgE)

O(E)

O(lgV)

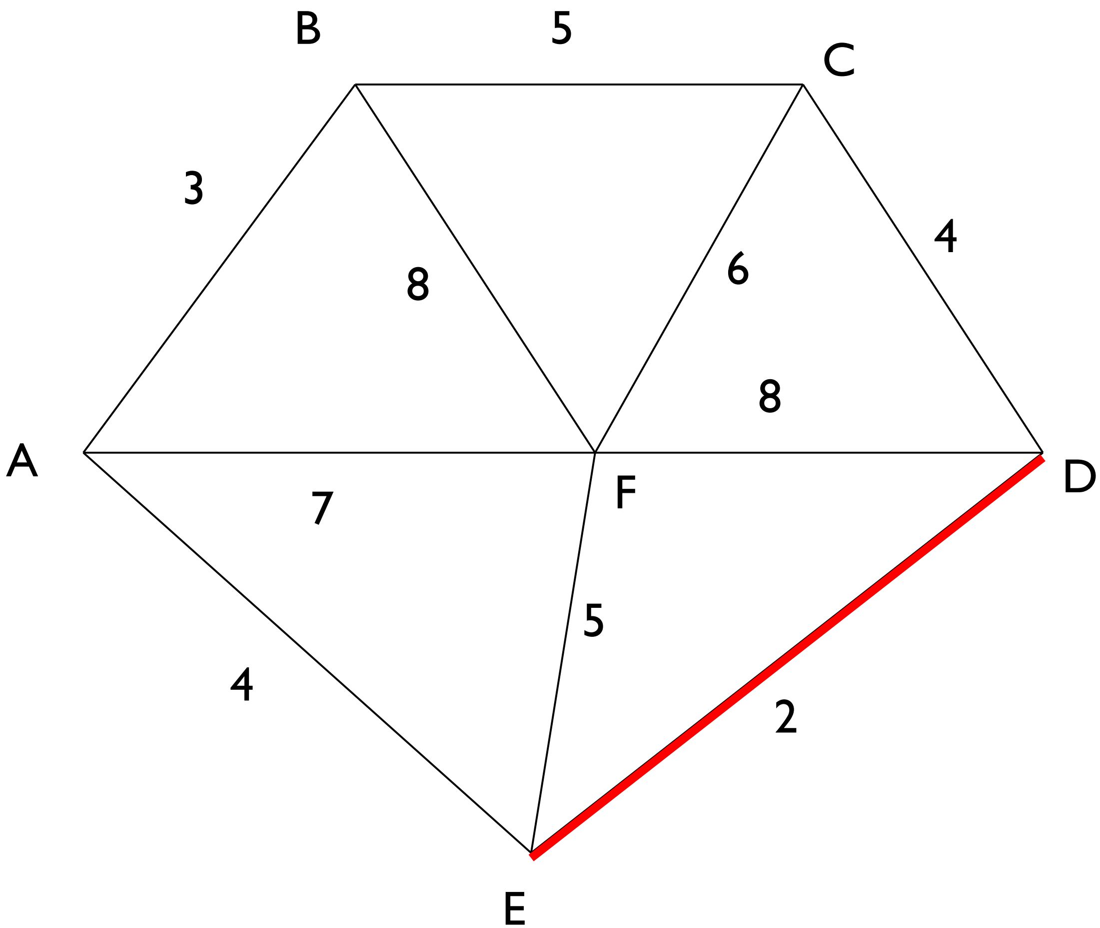
Kruskal's Algorithm



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
DF 8

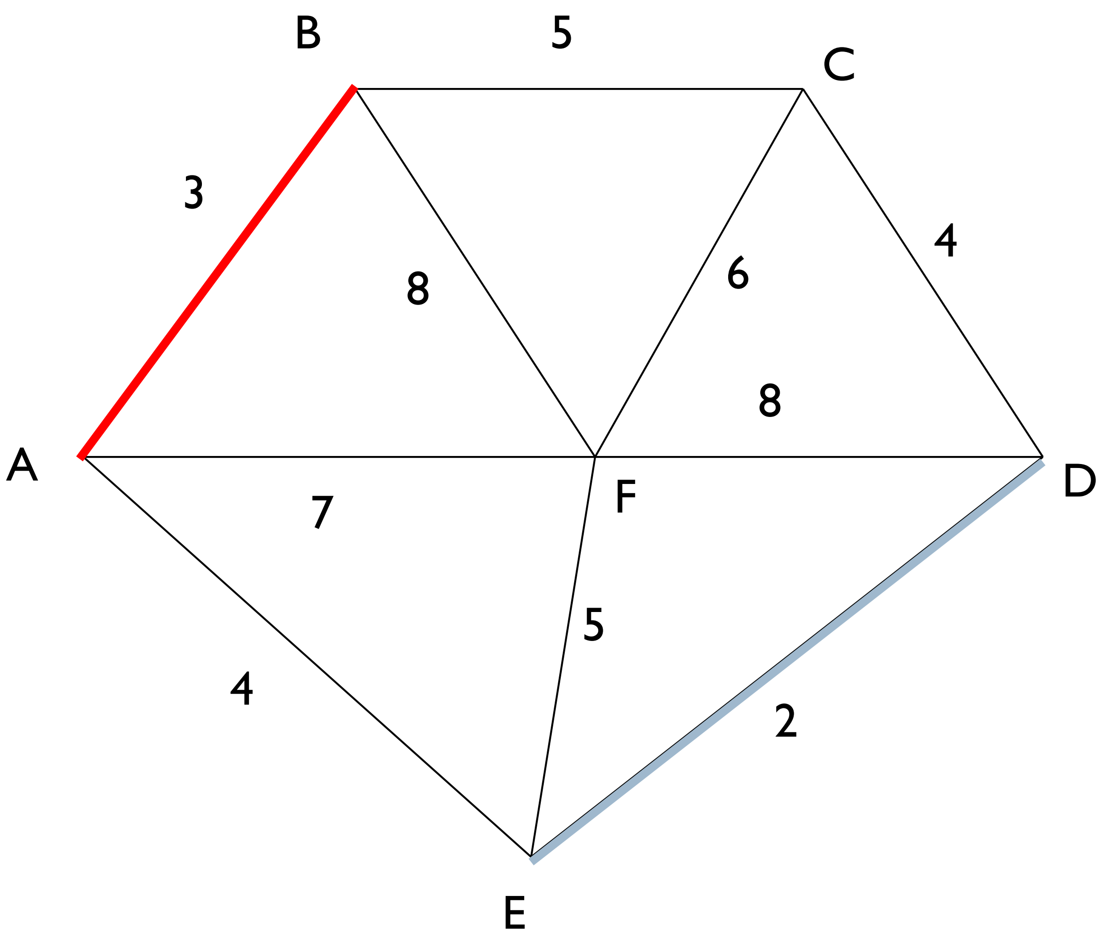
Kruskal's Algorithm



Select the shortest edge in the network

ED 2

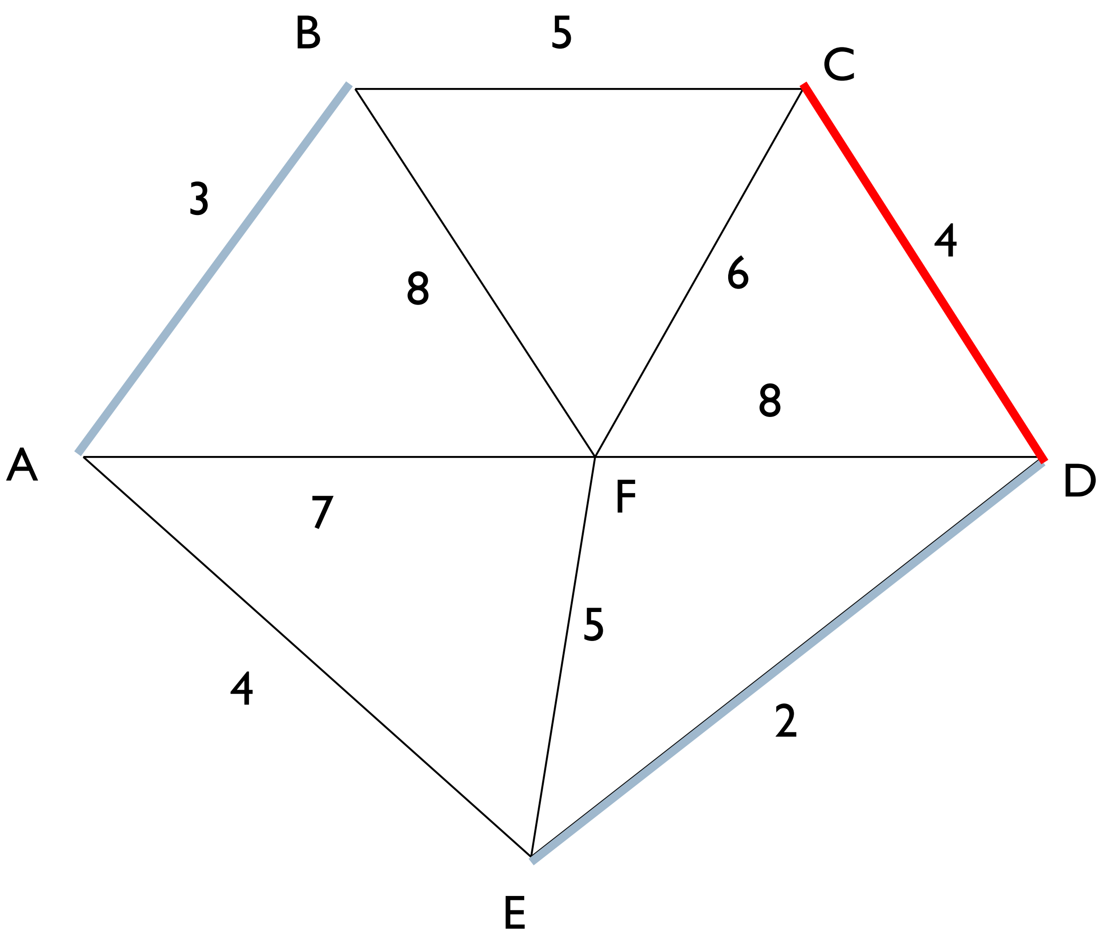
Kruskal's Algorithm



Select the next shortest edge which does not create a cycle

ED 2
AB 3

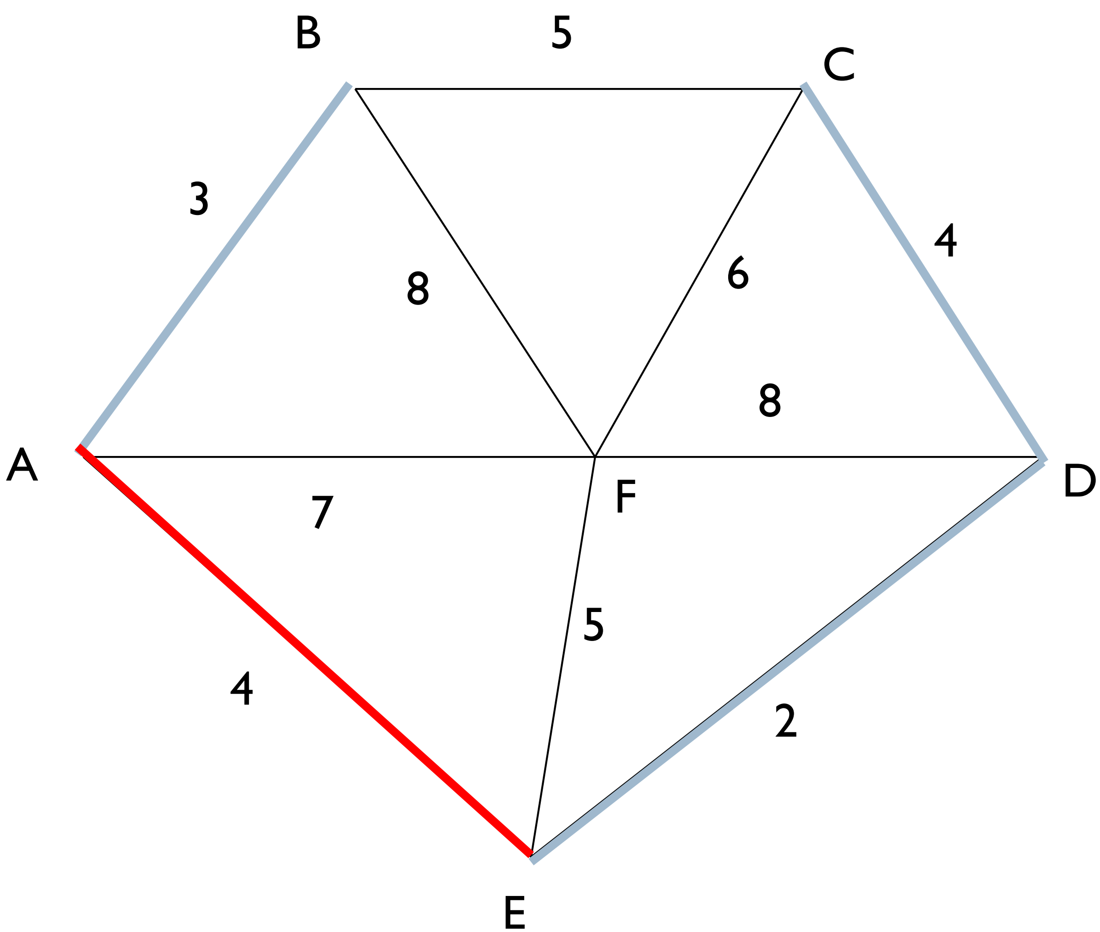
Kruskal's Algorithm



Select the next shortest edge which does not create a cycle

ED 2
AB 3
CD 4 (or AE 4)

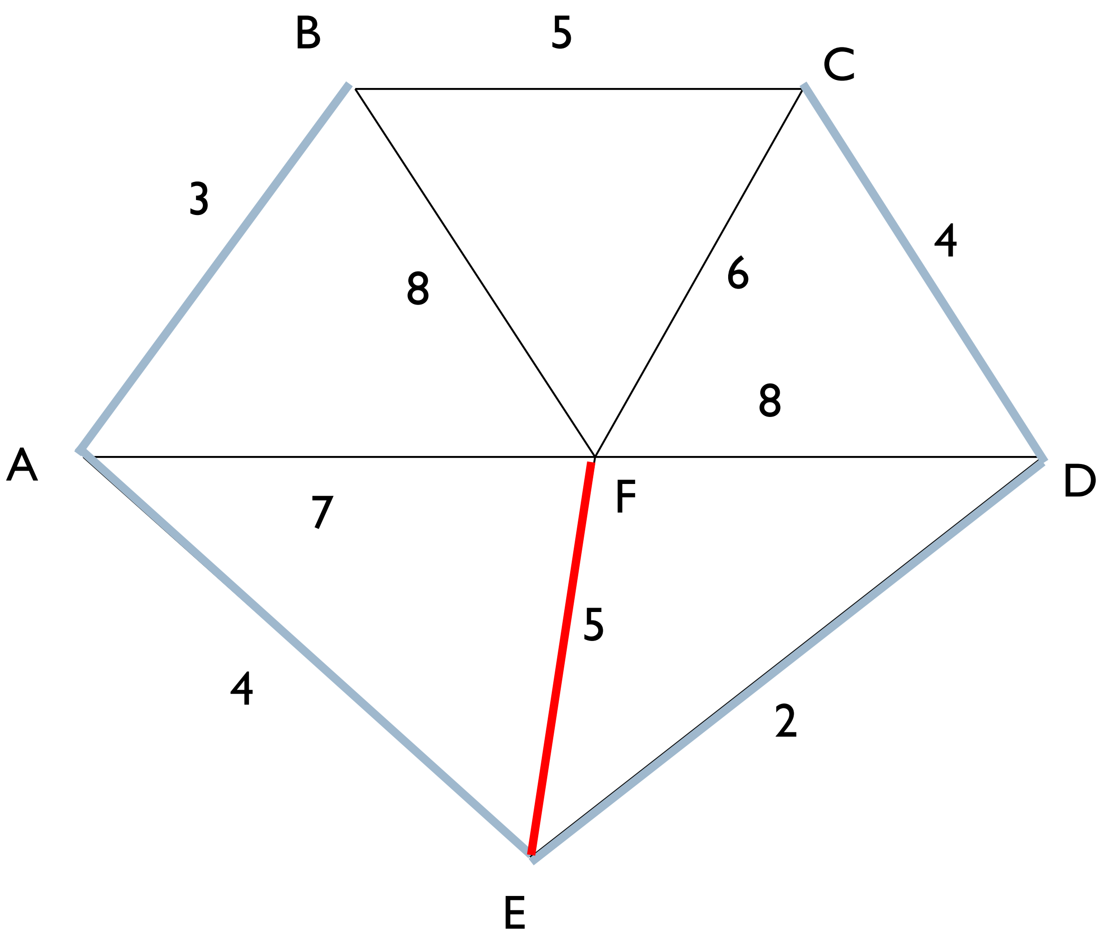
Kruskal's Algorithm



Select the next shortest edge which does not create a cycle

ED 2
AB 3
CD 4
AE 4

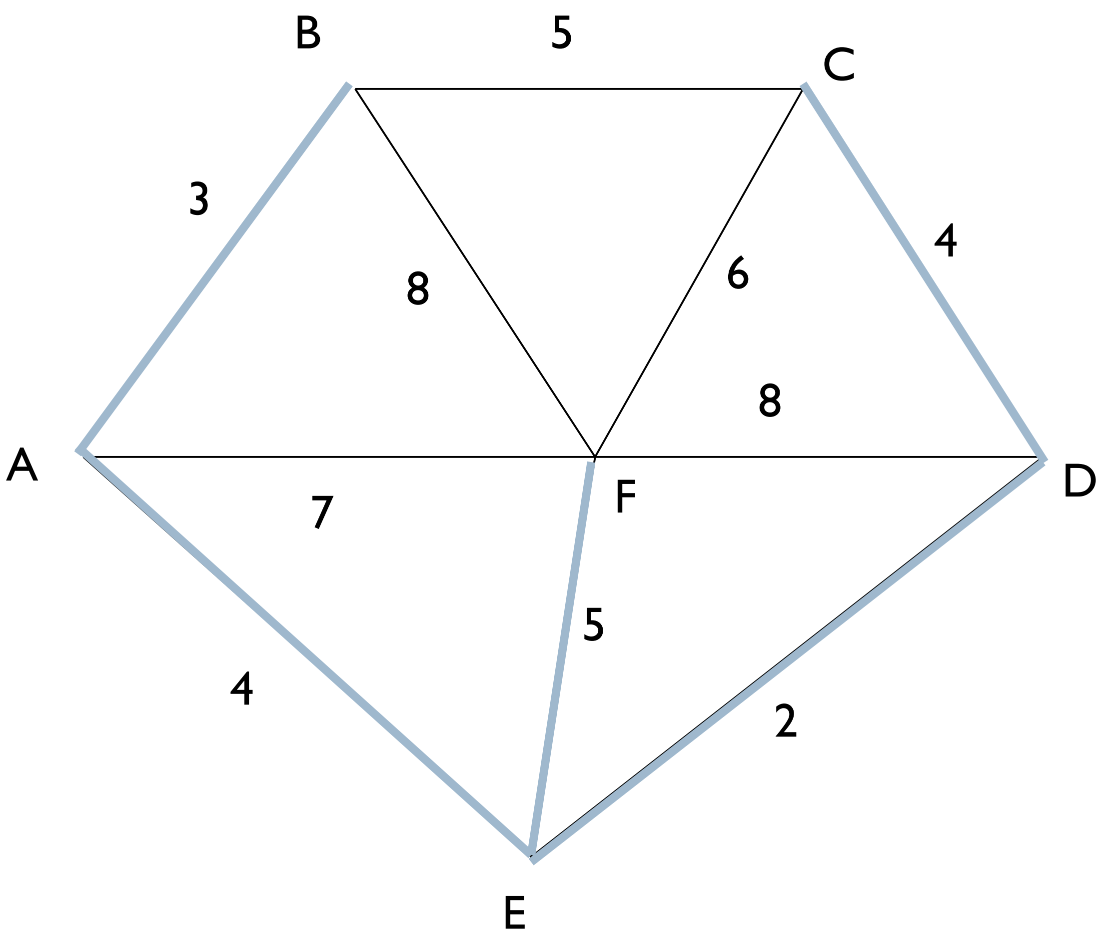
Kruskal's Algorithm



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



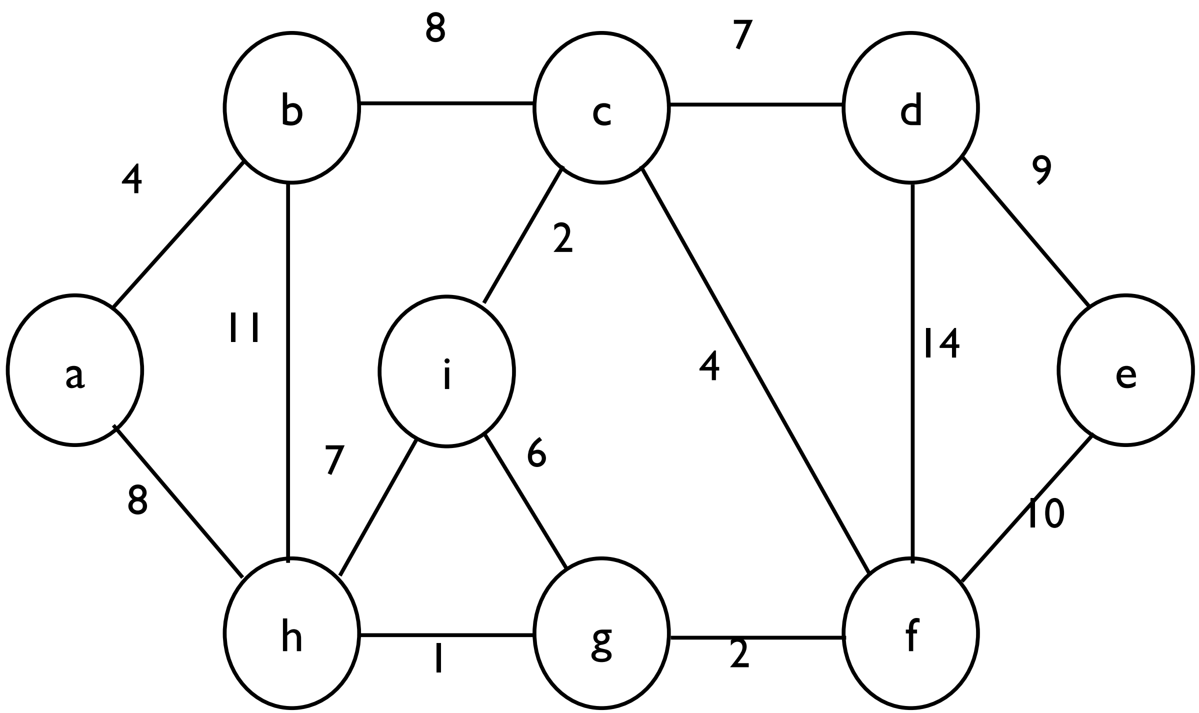
All vertices have been connected.

The solution is

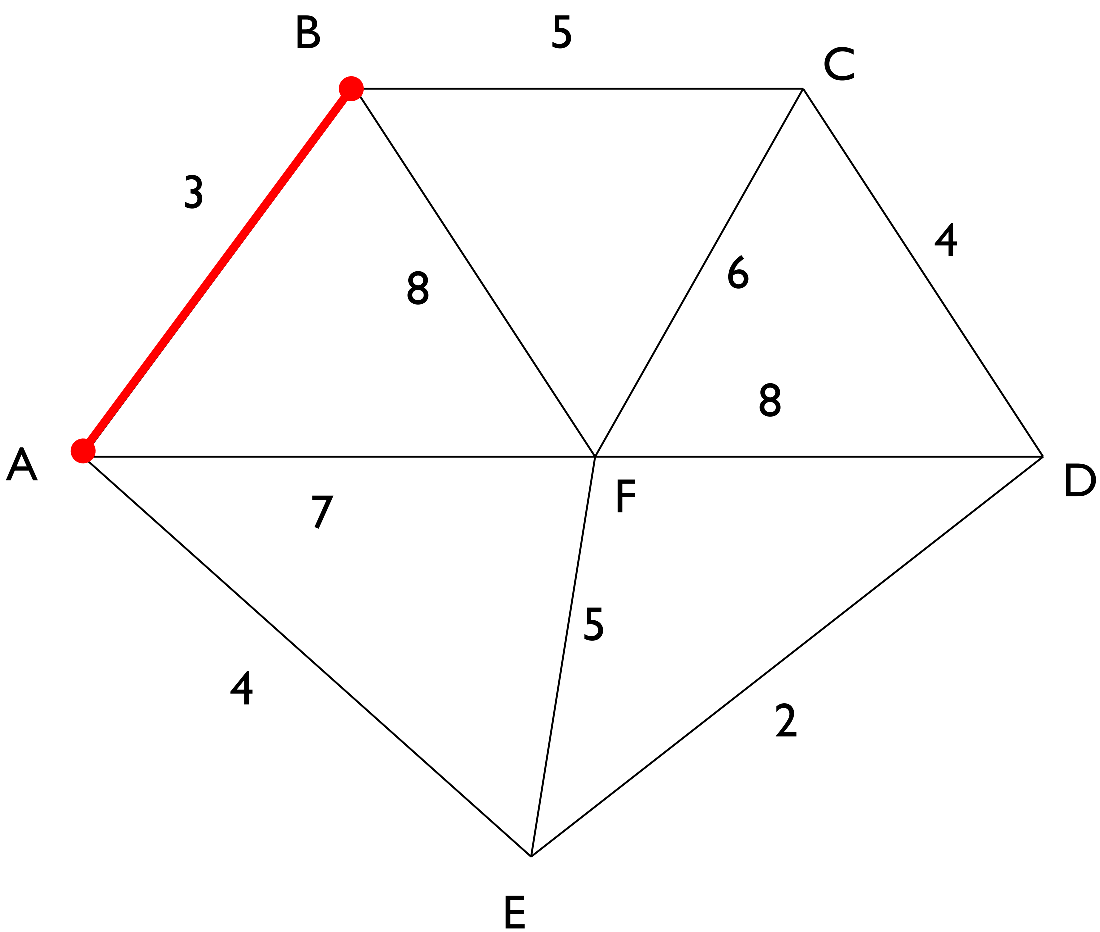
ED 2
AB 3
CD 4
AE 4
EF 5

Total weight of tree: 18

Find Spanning Tree using Kruskals Algorithm



Prim's Algorithm



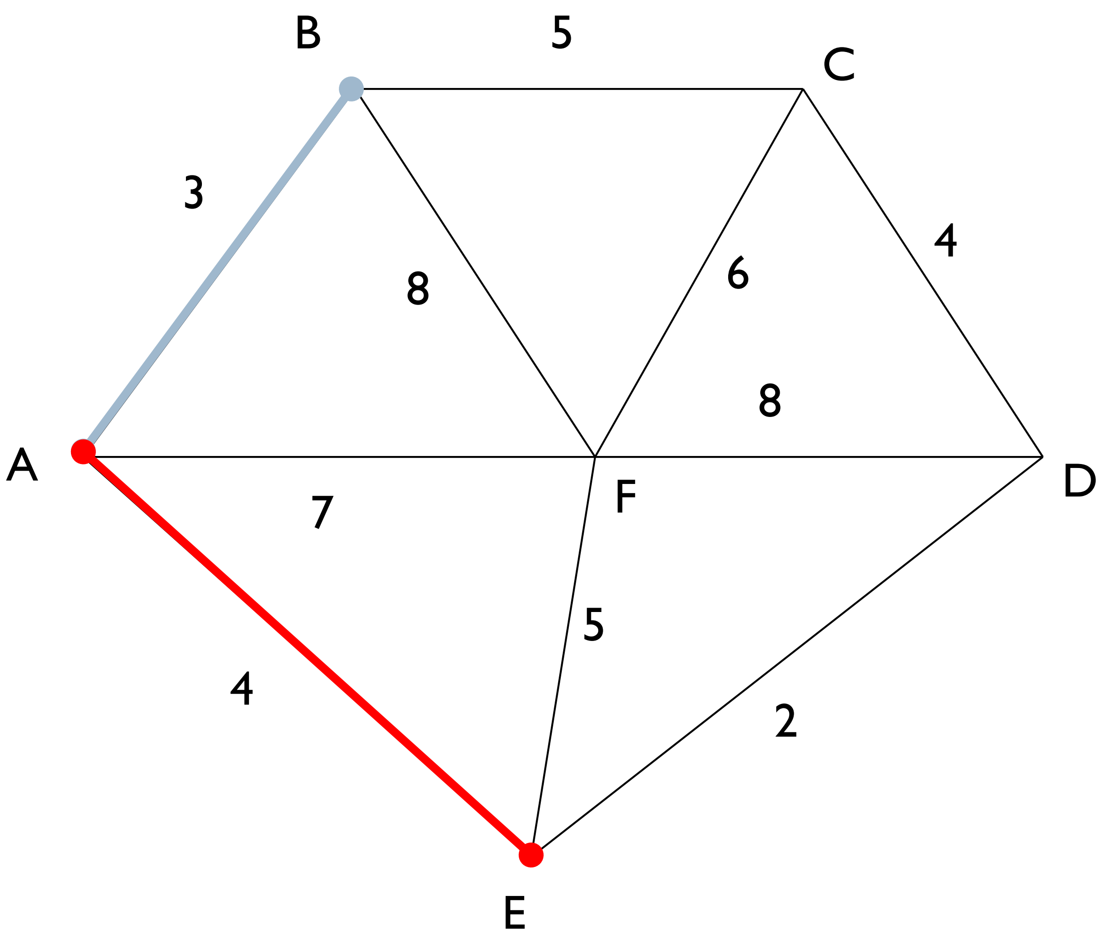
Select any vertex

A

Select the shortest edge connected to that vertex

AB 3

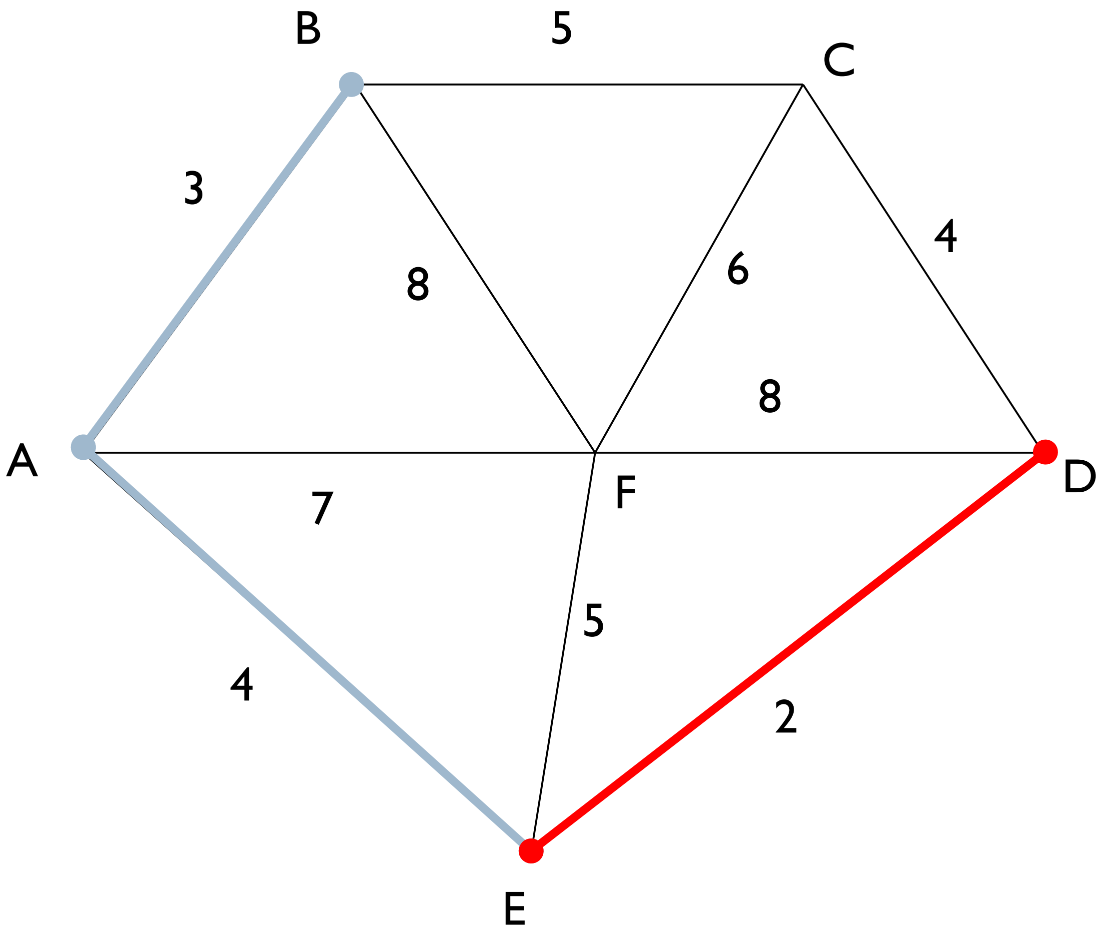
Prim's Algorithm



Select the shortest edge connected to any vertex already connected.

AE 4

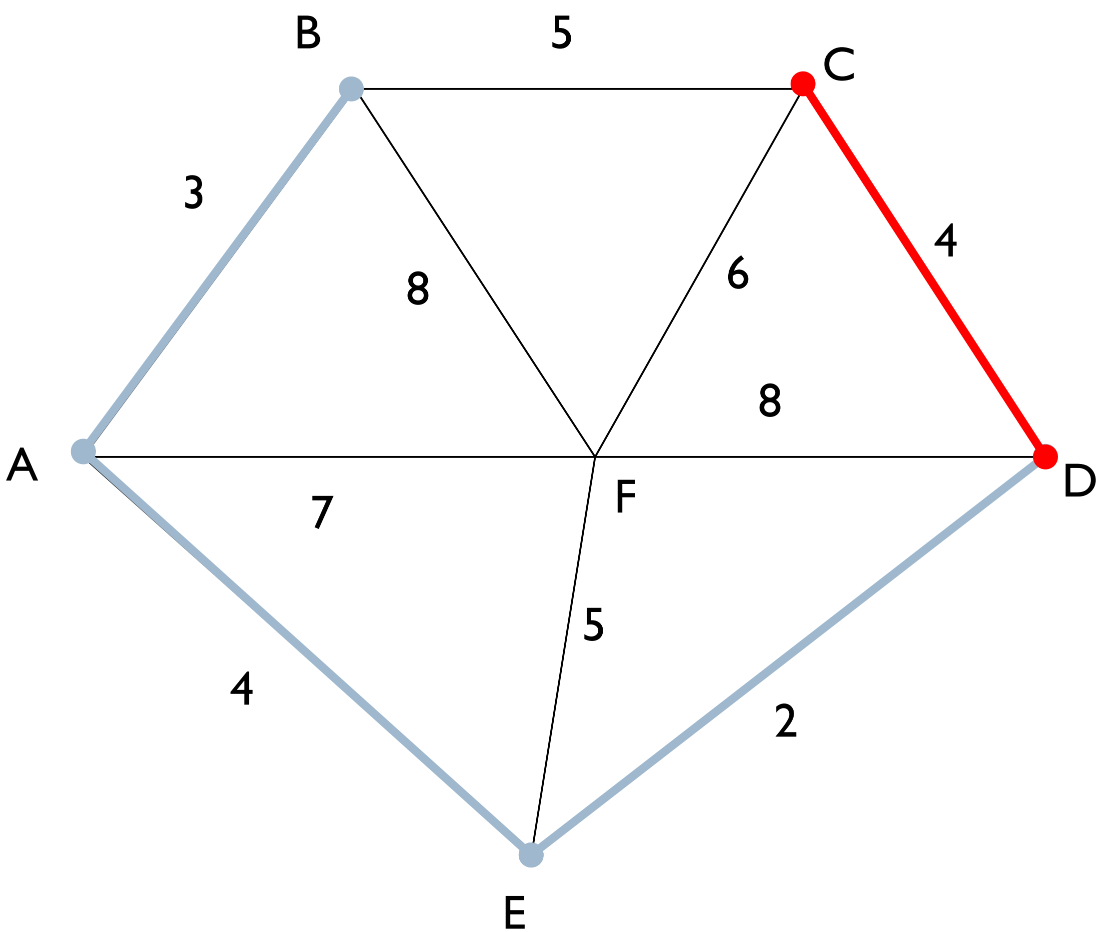
Prim's Algorithm



Select the shortest edge connected to any vertex already connected.

ED 2

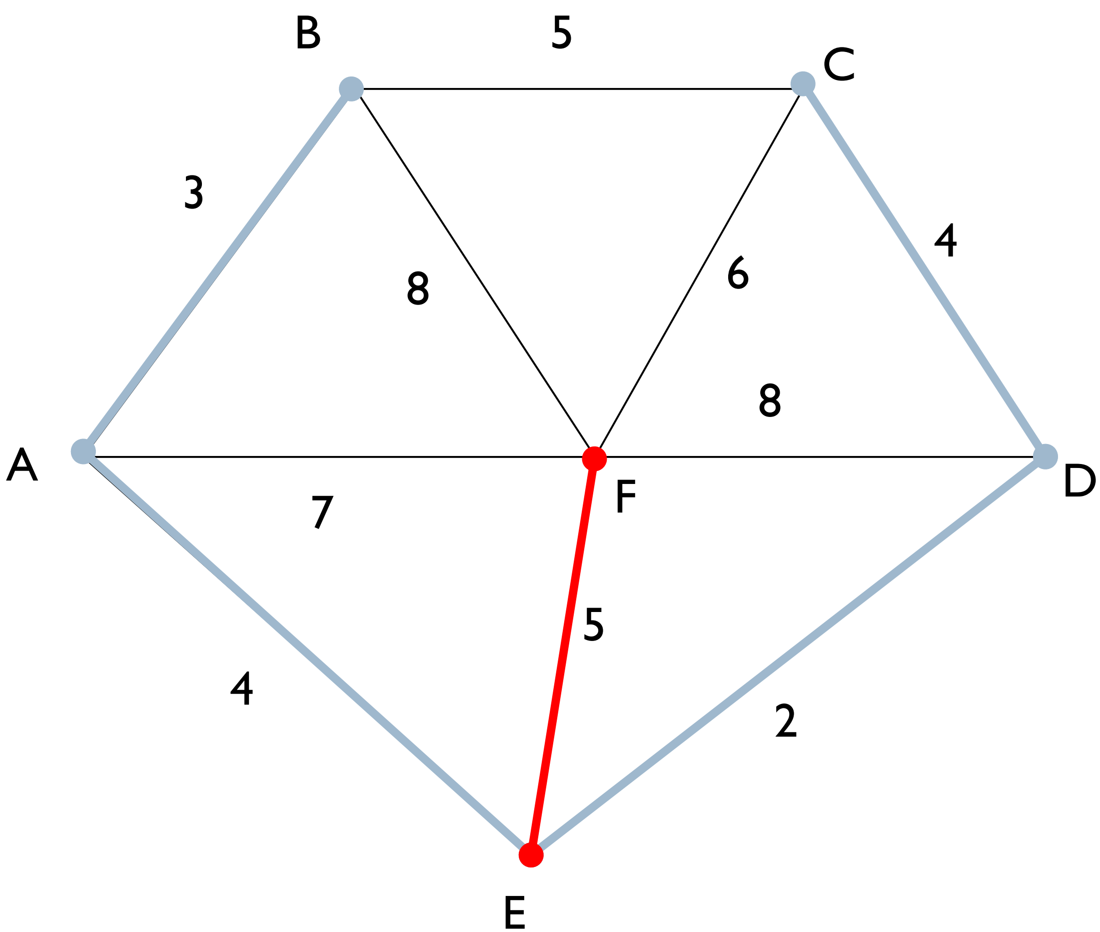
Prim's Algorithm



Select the shortest edge connected to any vertex already connected.

DC 4

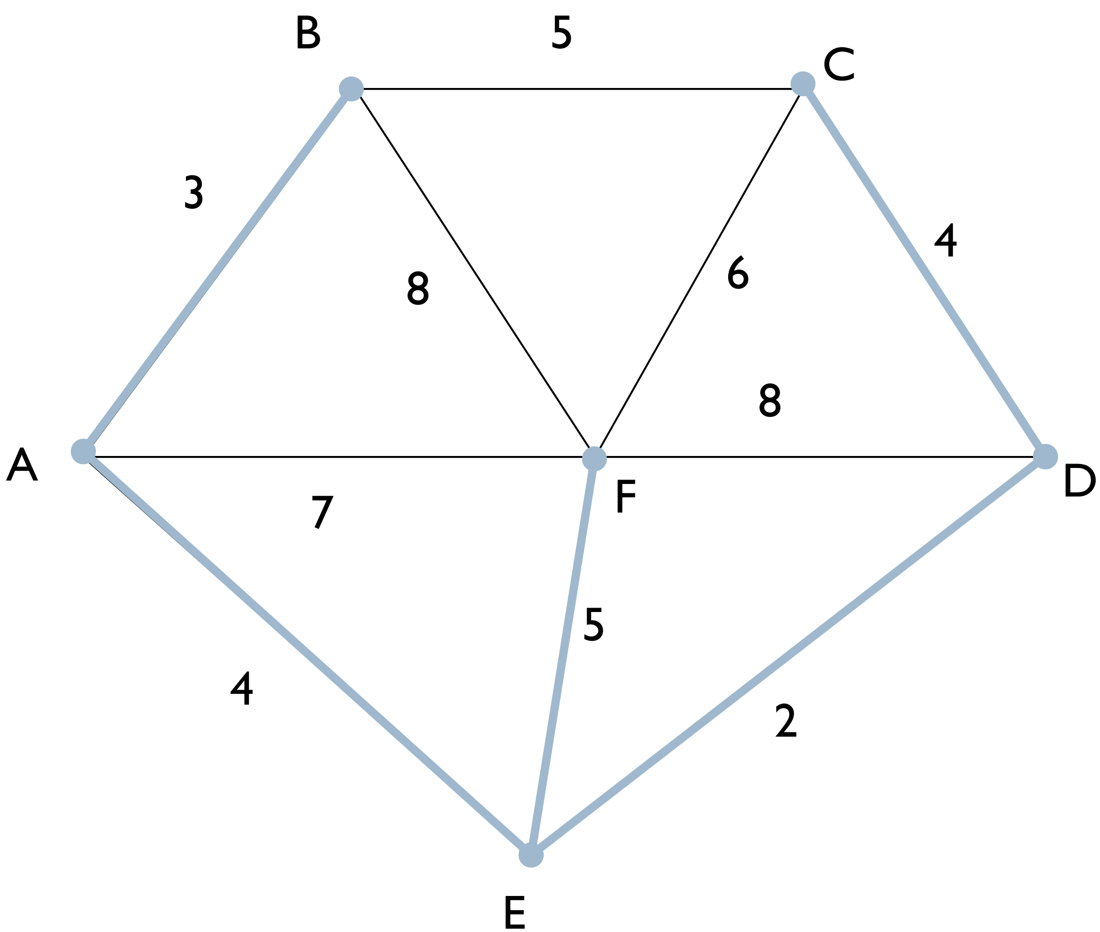
Prim's Algorithm



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

When the algorithm terminates, the min-priority queue Q is empty; the minimum spanning tree A for G is thus

$$A = \{(v, v.\pi) : v \in V - \{r\}\} .$$

Total time: $O(V\lg V + E\lg V) = O(E\lg V)$

MST-PRIM(G, w, r)

```
for each u ∈ G.V
        u.key = ∞
        u.π = NIL
r.key = 0
Q = G.V
while Q ≠ ∅
        u = EXTRACT-MIN(Q)
        for each v ∈ G.Adj[u]
                if v ∈ Q and w(u, v) < v.key
                        v.π = u
                        v.key = w(u, v)
```

$O(V)$ if Q is implemented as a min-heap

Takes $O(\lg V)$

Min-heap operations: $O(V\lg V)$

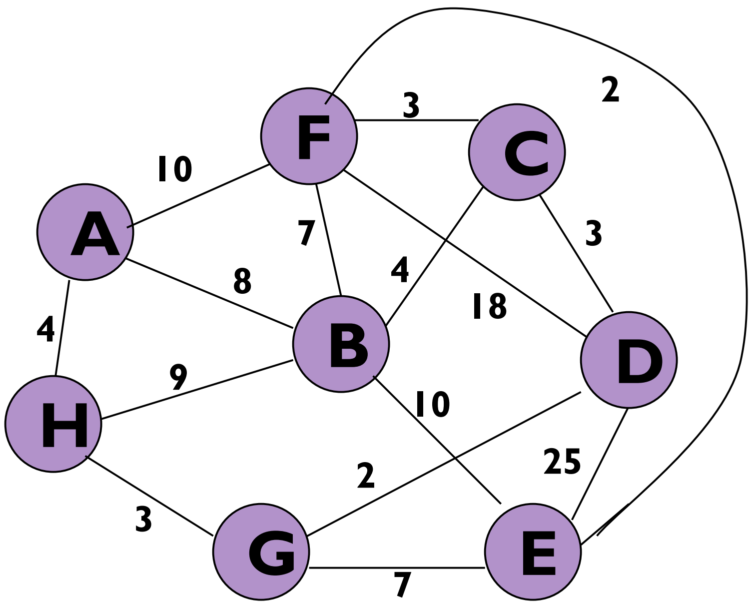
Executed O(E) times total

Constant

$O(E\lg V)$

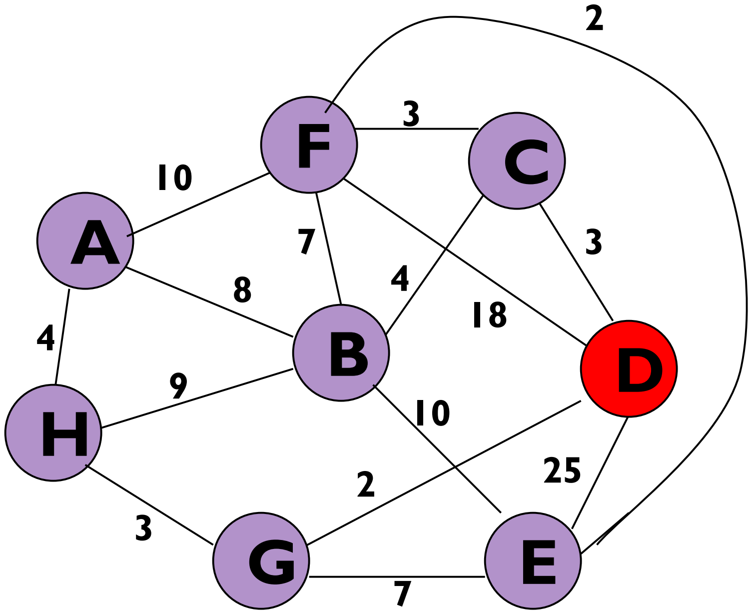
Takes $O(\lg V)$

Walk-Through



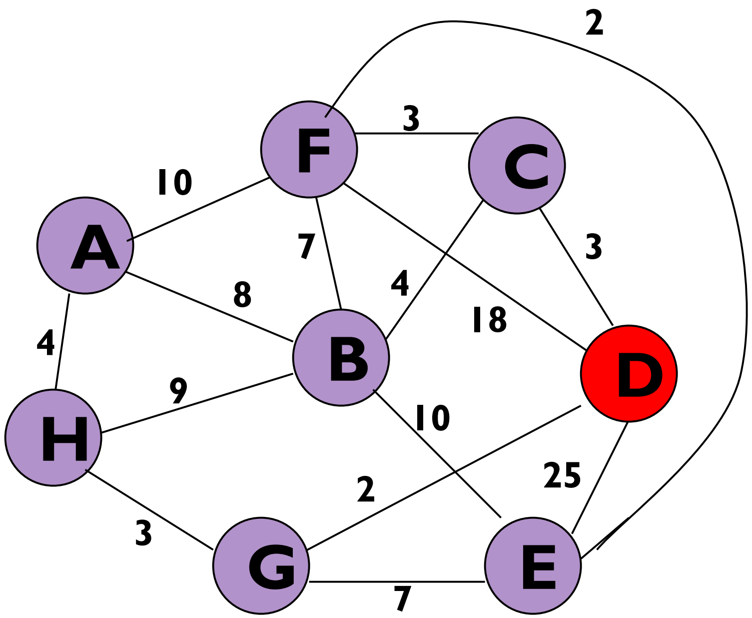
Initialize array

	r	k_v	π_v
A	–	∞	–
B	–	∞	–
C	–	∞	–
D	–	∞	–
E	–	∞	–
F	–	∞	–
G	–	∞	–
H	–	∞	–



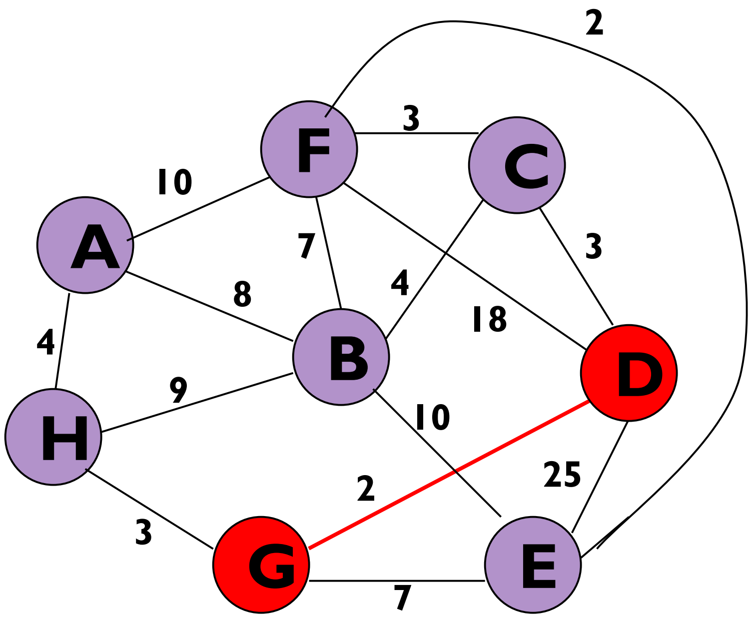
Start with any node, say D

	r	k_v	π_v
A			
B			
C			
D	T	0	–
E			
F			
G			
H			



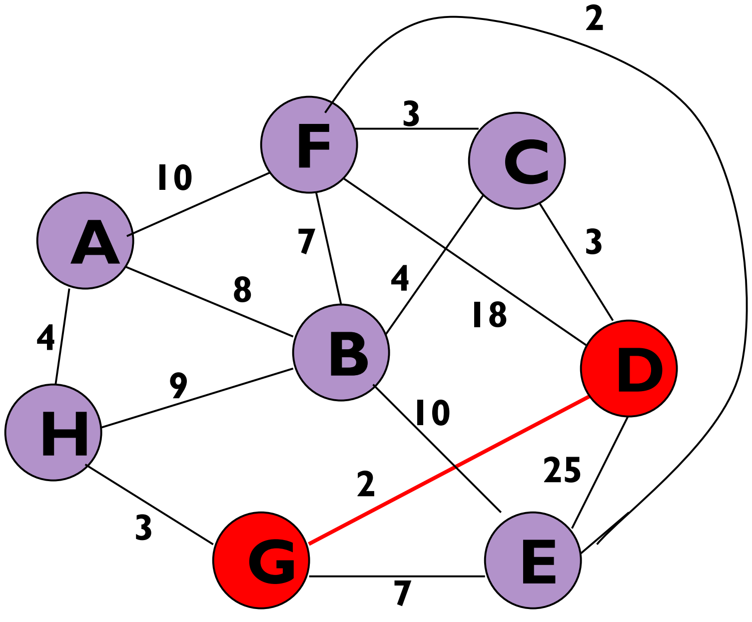
Update distances of adjacent, unselected nodes

	r	k_v	π_v
A			
B			
C		3	D
D	T	0	–
E		25	D
F		18	D
G		2	D
H			



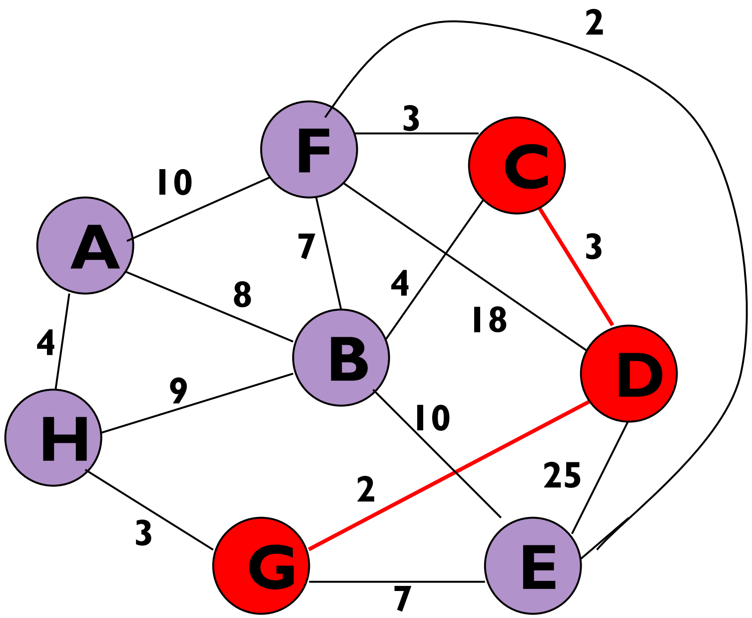
Select node with minimum distance

	r	k_v	π_v
A			
B			
C		3	D
D	T	0	–
E		25	D
F		18	D
G	T	2	D
H			



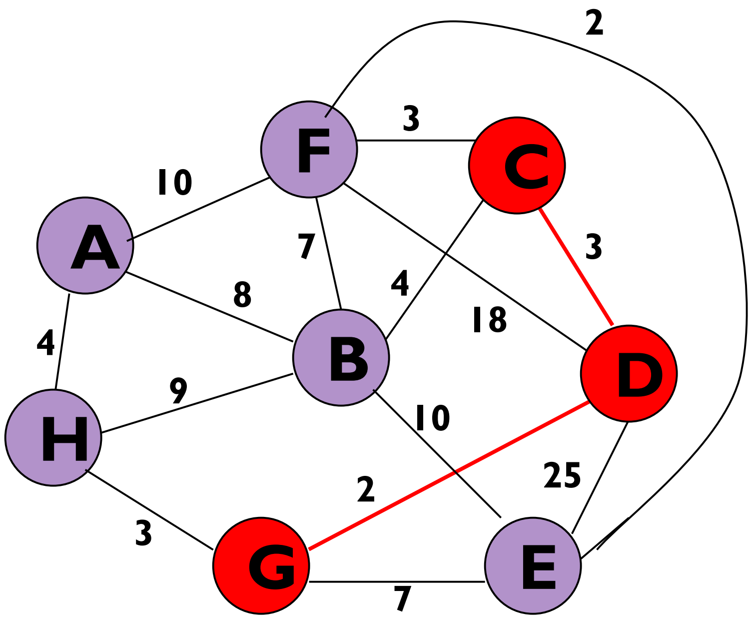
Update distances of adjacent, unselected nodes

	r	k_v	π_v
A			
B			
C		3	D
D	T	0	–
E		7	G
F		18	D
G	T	2	D
H		3	G



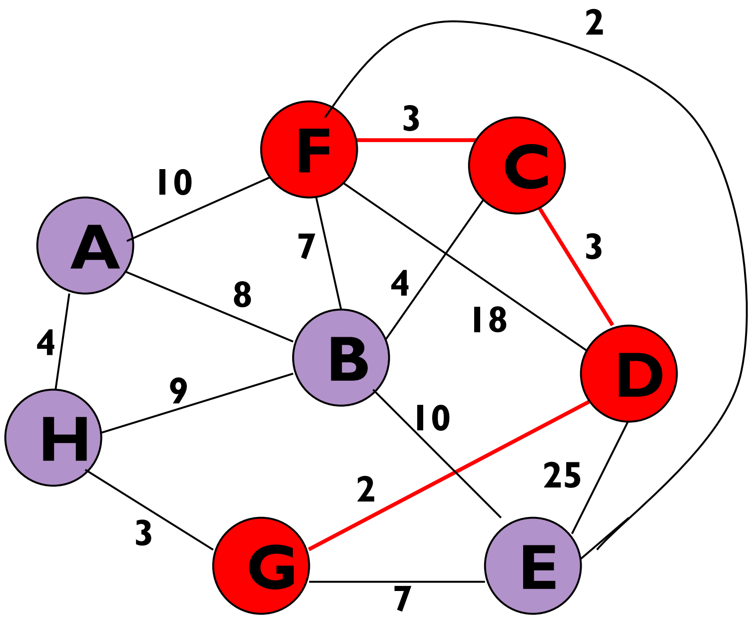
Select node with minimum distance

	r	k_v	π_v
A			
B			
C	T	3	D
D	T	0	–
E		7	G
F		18	D
G	T	2	D
H		3	G



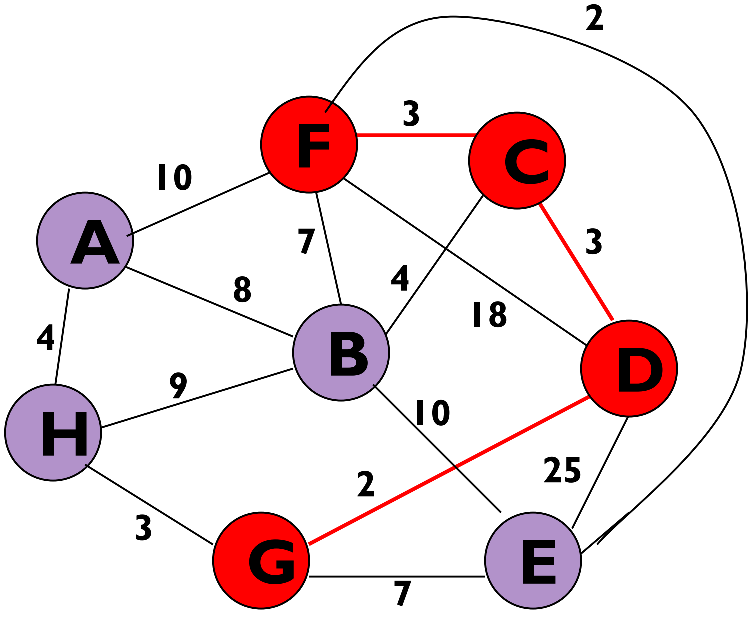
Update distances of adjacent, unselected nodes

	r	k_v	π_v
A			
B		4	C
C	T	3	D
D	T	0	–
E		7	G
F		3	C
G	T	2	D
H		3	G



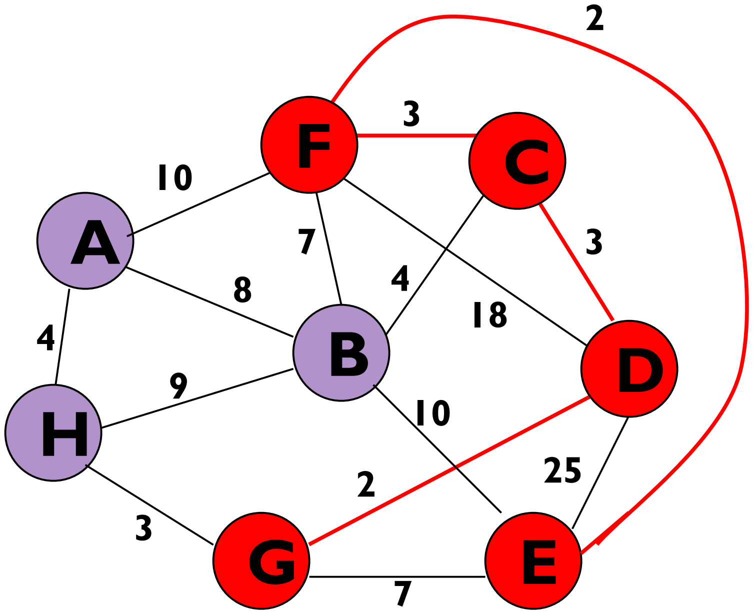
Select node with minimum distance

	r	k_v	π_v
A			
B		4	C
C	T	3	D
D	T	0	–
E		7	G
F	T	3	C
G	T	2	D
H		3	G



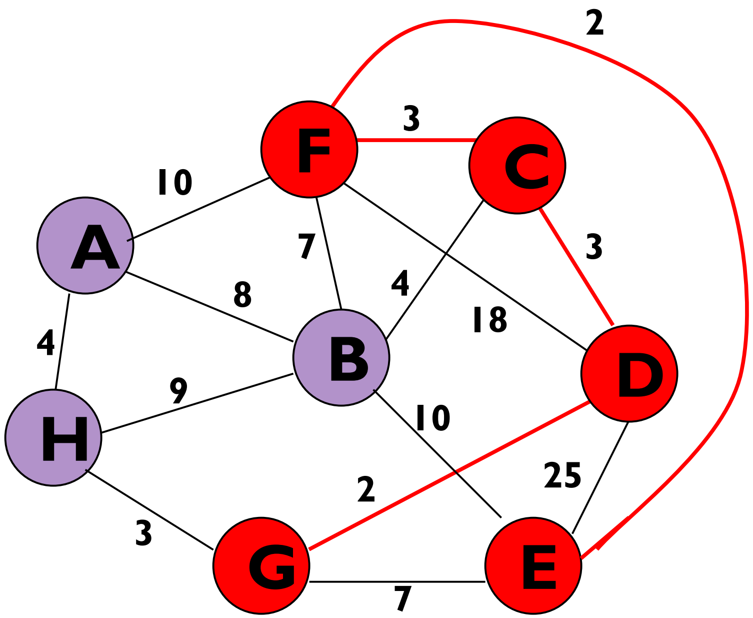
Update distances of adjacent, unselected nodes

	r	k_v	π_v
A		10	F
B		4	C
C	T	3	D
D	T	0	–
E		2	F
F	T	3	C
G	T	2	D
H		3	G



Select node with minimum distance

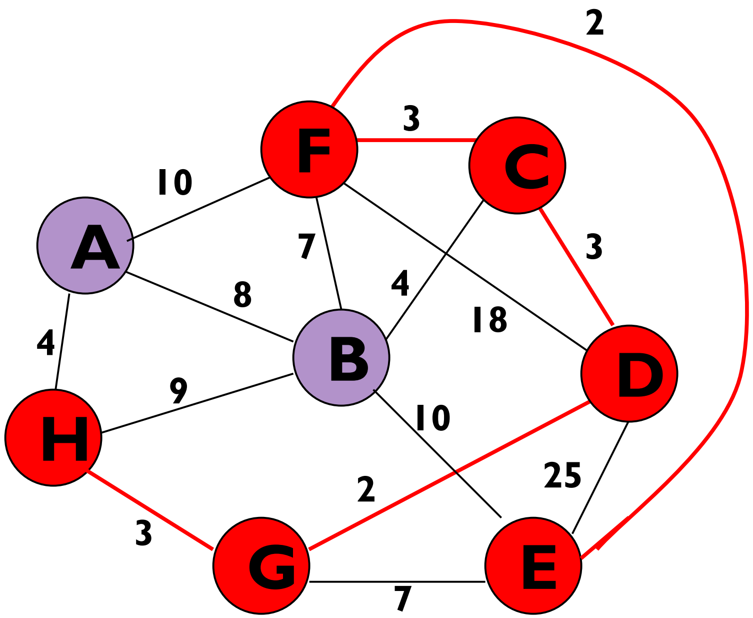
	r	k_v	π_v
A		10	F
B		4	C
C	T	3	D
D	T	0	–
E	T	2	F
F	T	3	C
G	T	2	D
H		3	G



Update distances of adjacent, unselected nodes

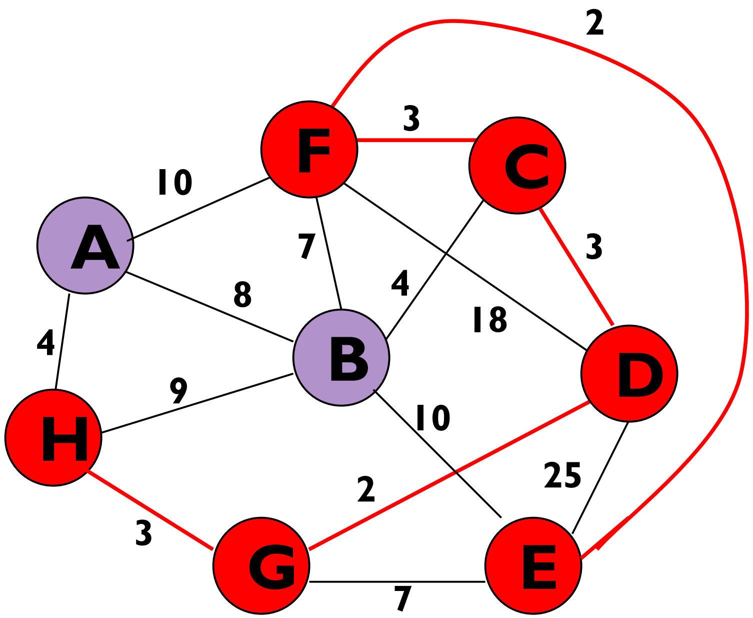
	r	k_v	π_v
A		10	F
B		4	C
C	T	3	D
D	T	0	–
E	T	2	F
F	T	3	C
G	T	2	D
H		3	G

Table entries unchanged



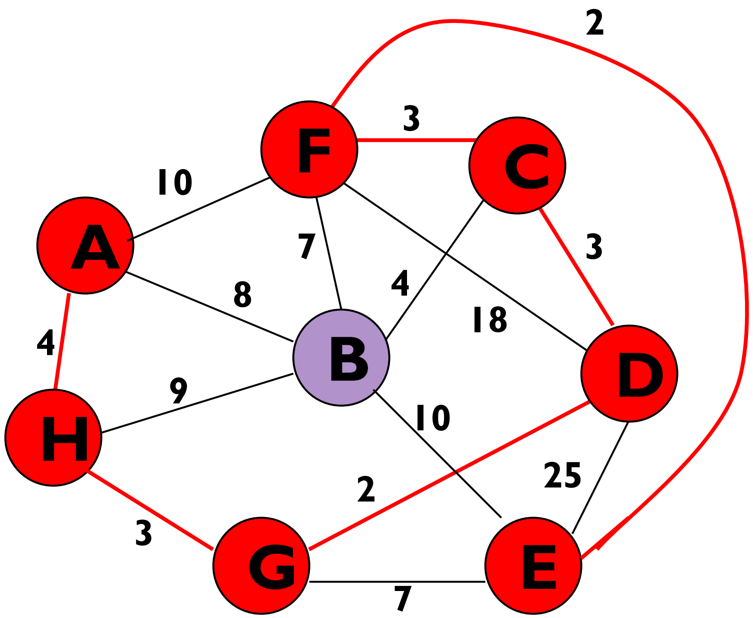
Select node with minimum distance

	r	k_v	π_v
A		10	F
B		4	C
C	T	3	D
D	T	0	–
E	T	2	F
F	T	3	C
G	T	2	D
H	T	3	G



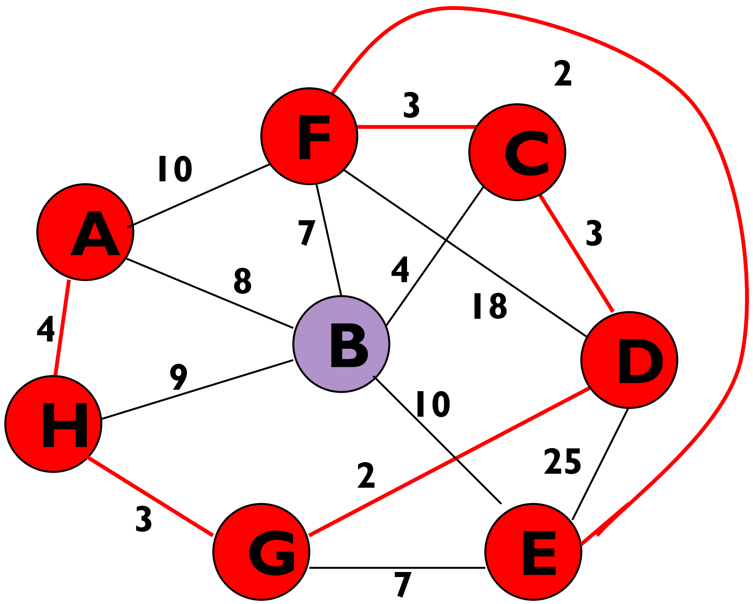
Update distances of adjacent, unselected nodes

	r	k_v	π_v
A		4	H
B		4	C
C	T	3	D
D	T	0	–
E	T	2	F
F	T	3	C
G	T	2	D
H	T	3	G



Select node with minimum distance

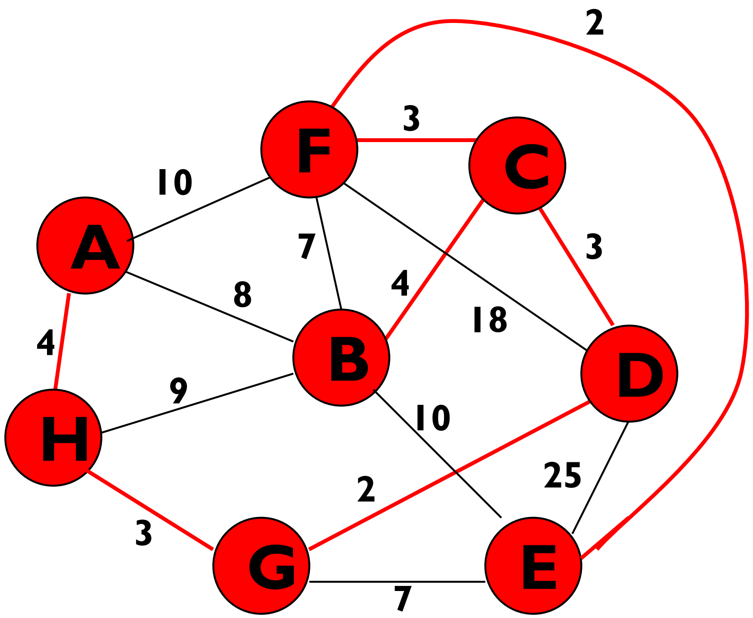
	r	k_v	π_v
A	T	4	H
B		4	C
C	T	3	D
D	T	0	–
E	T	2	F
F	T	3	C
G	T	2	D
H	T	3	G



Update distances of adjacent, unselected nodes

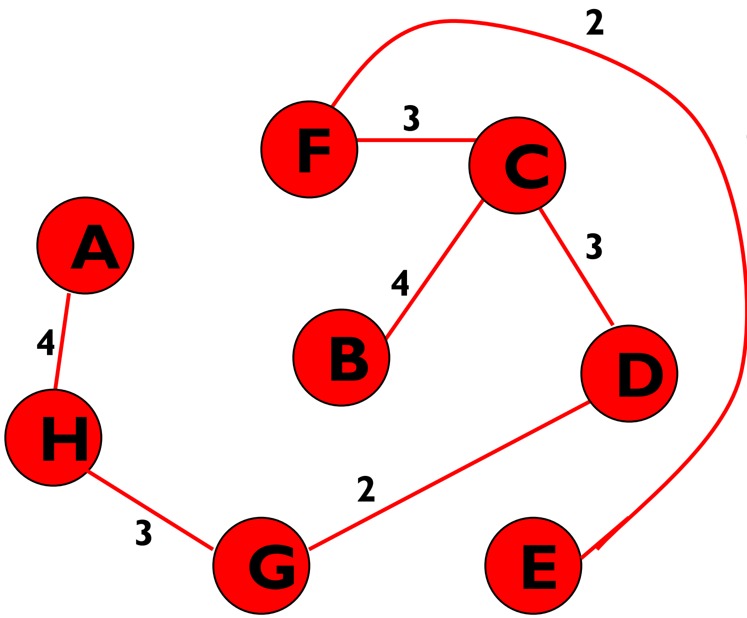
	r	k_v	π_v
A	T	4	H
B		4	C
C	T	3	D
D	T	0	–
E	T	2	F
F	T	3	C
G	T	2	D
H	T	3	G

Table entries unchanged



Select node with minimum distance

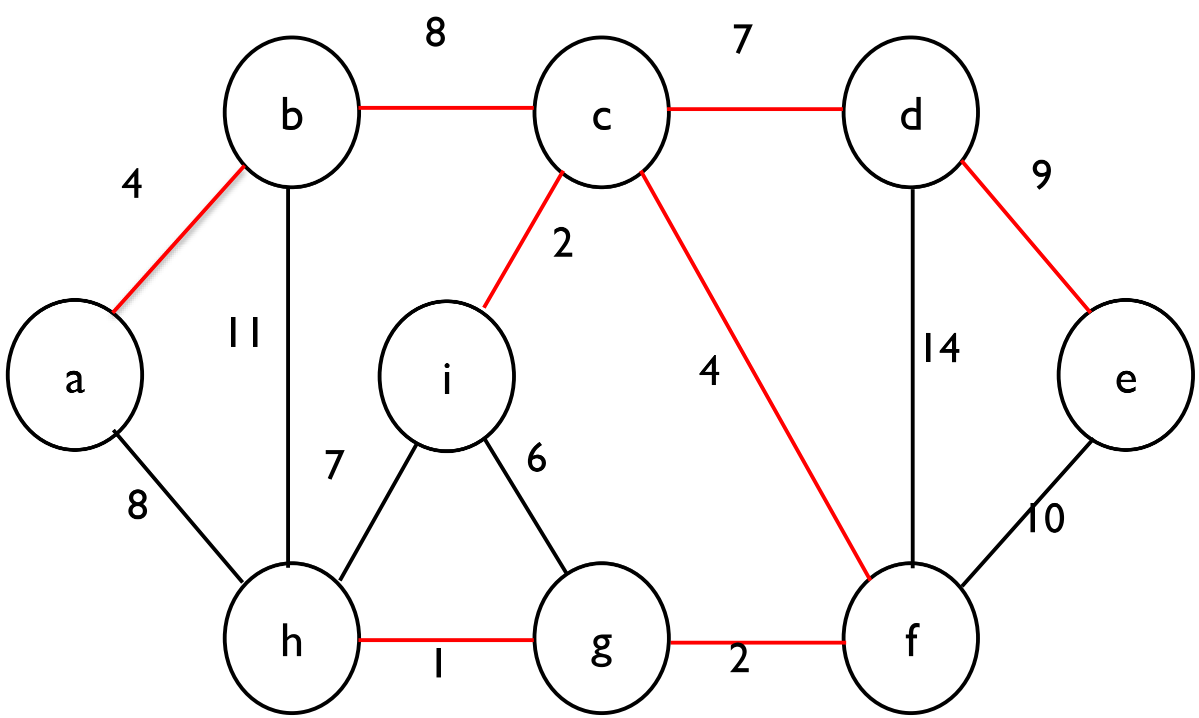
	r	k_v	π_v
A	T	4	H
B	T	4	C
C	T	3	D
D	T	0	–
E	T	2	F
F	T	3	C
G	T	2	D
H	T	3	G



Cost of Minimum Spanning Tree = Σ $\mathbf{d_v}$ = **21**

	r	k_v	π_v
A	T	4	H
B	T	4	C
C	T	3	D
D	T	0	–
E	T	2	F
F	T	3	C
G	T	2	D
H	T	3	G

Find Spanning Tree using Prims Algorithm



Some points to note

- 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.