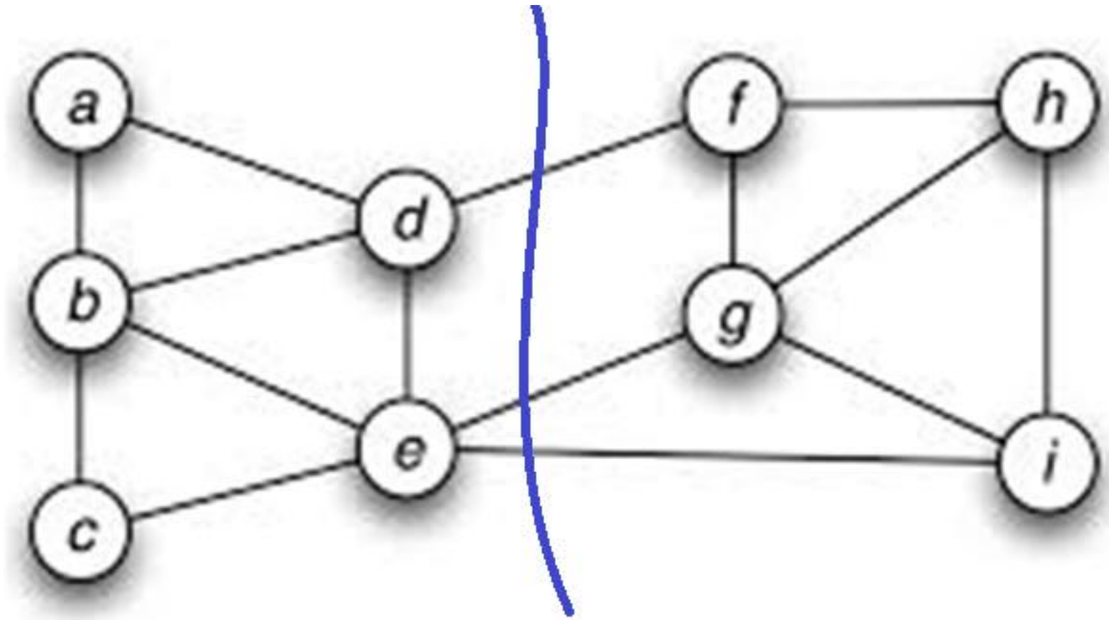
Minimum Spanning Tree

Definitions

- Given an undirected graph G = (V, E), a ***cut*** (S, V – S) is a partition of vertices into two disjoint sets S and V – S

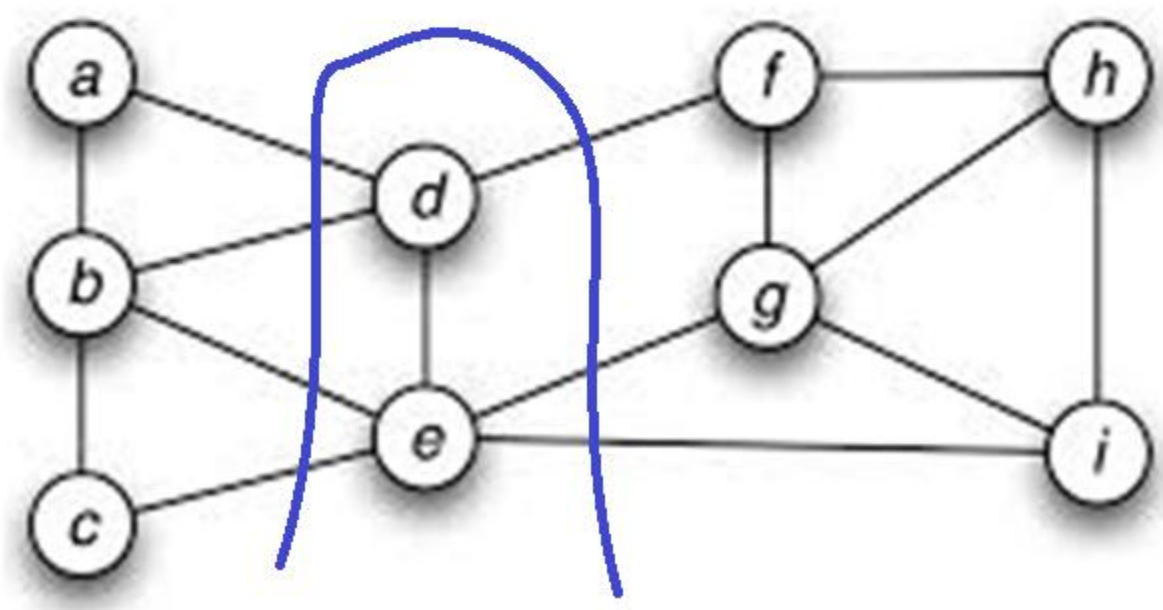


Example:

S = {a, b, c, d, e} and V – S = {f, g, h, i}

Definitions

- Given an undirected graph G = (V, E), a ***cut*** (S, V –S) is a partition of vertices into two disjoint sets S and V – S

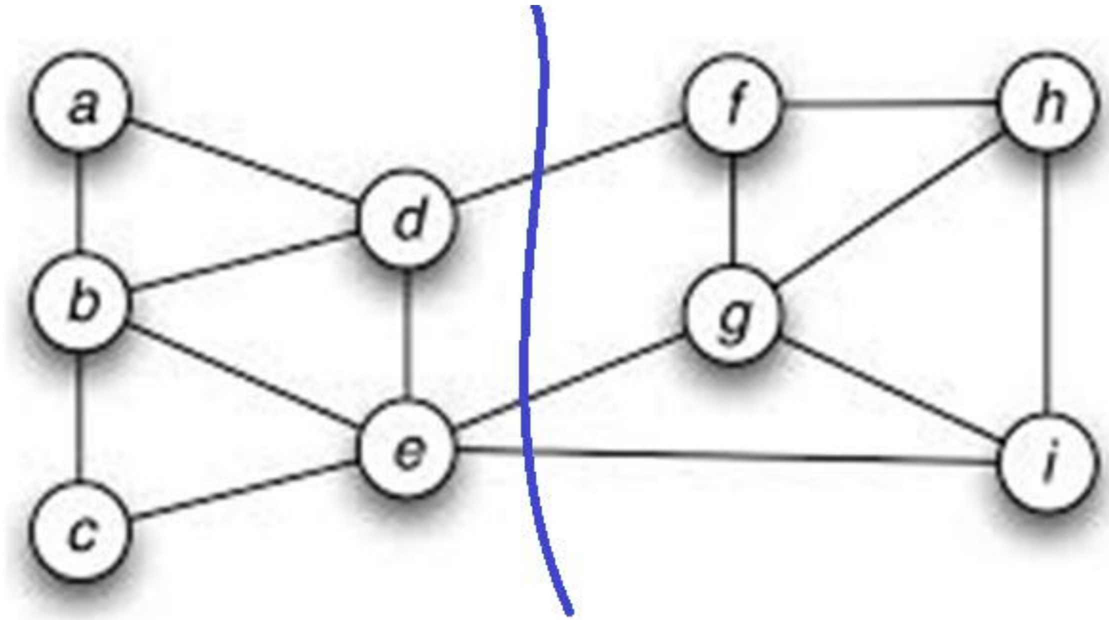


Example: A cut (S, V – S),

where S = {d, e} and V – S = {a, b, c, f, g, h, i}

Definitions

- Given an undirected graph G = (V, E) and a cut (S, V –S), a ***crossing edge of the cut*** is an edge (u, v) such that u $\in$ S and v $\in$ V – S

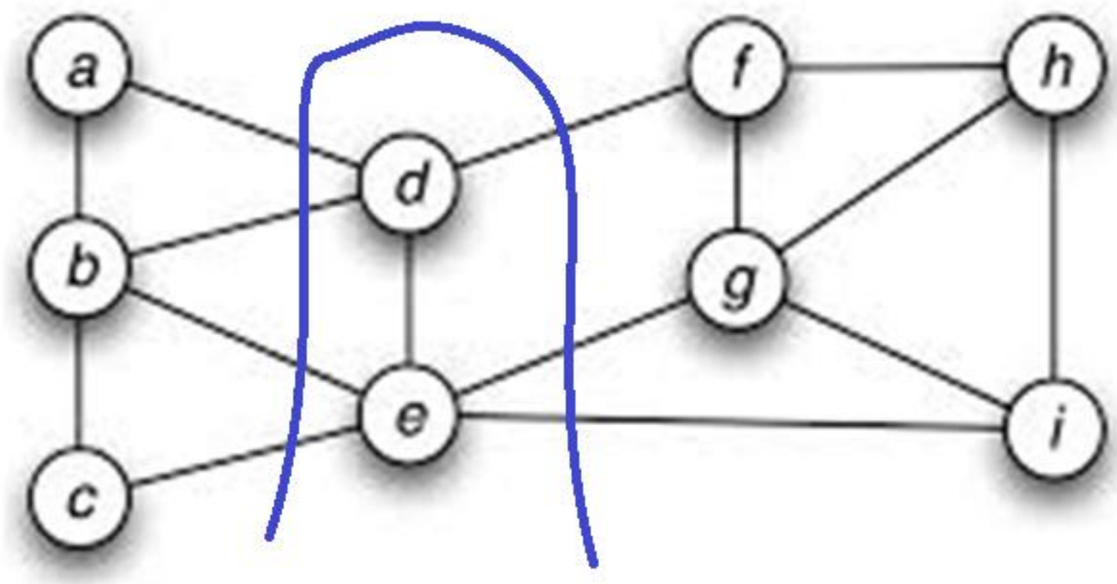


Example:

(d, f), (e, g) and (e, i) are ***crossing edges*** of (S, V-S), where S = {a, b, c, d, e} and V – S = {f, g, h, i}

Definitions

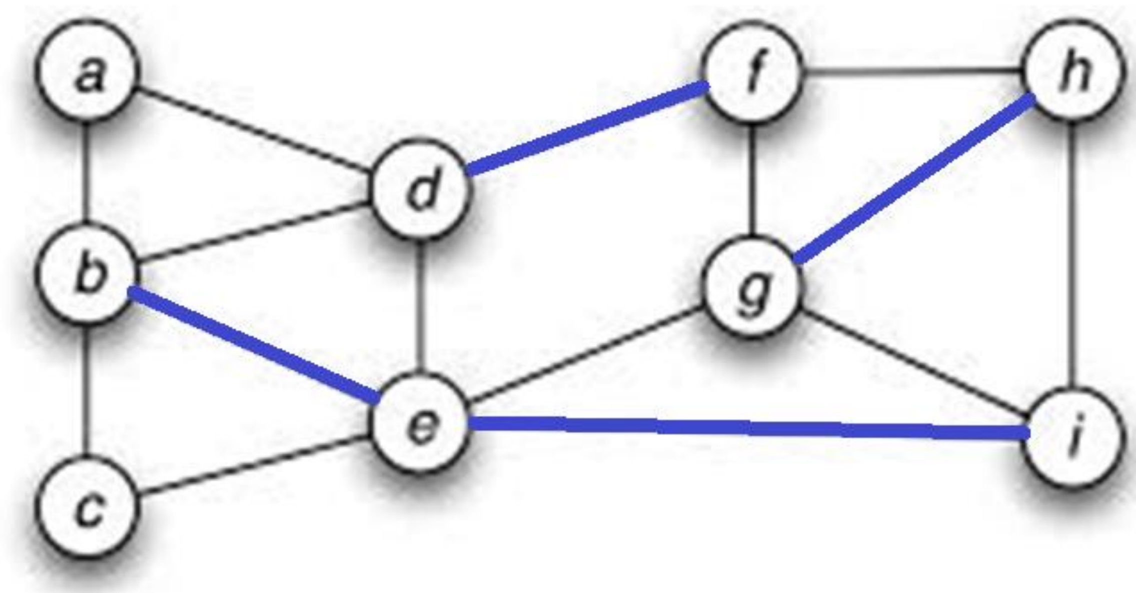
- Given an undirected graph G = (V, E) and a cut (S, V-S), an edge (u, v) in E ***crosses*** the cut if it is a crossing edge



Example: Edge (a, d) ***crosses the cut*** (S, V – S), where S = {d, e} and V – S = {a, b, c, f, g, h, i}

Definitions

- Given an undirected graph G = (V, E), a ***set of edges*** is a collection of edges in E

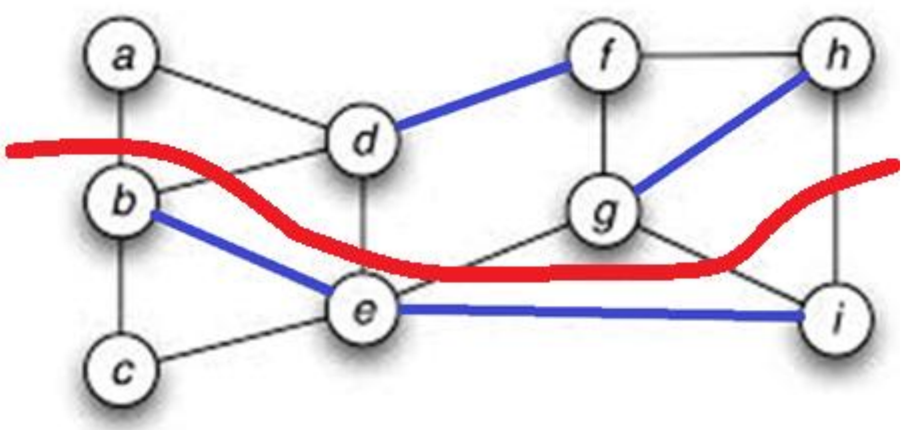


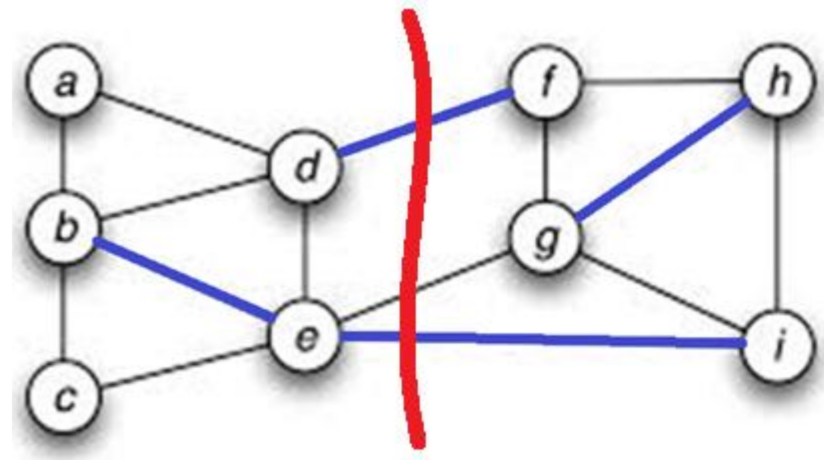
Example:

A = {(d, f), (g, h), (b, e), (e, i)} is a ***set of edges***

Definitions

- Given an undirected graph G = (V, E), a cut (S, V-S) and a set of edges A, the cut ***respects*** the set A if no edge in A crosses the cut



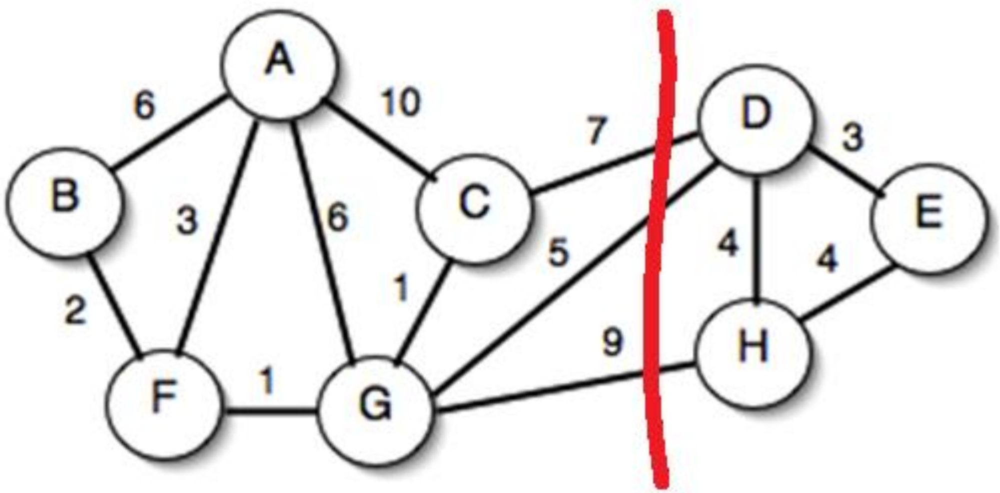


Example:

1. On the left, the cut ***respects*** A = {(d, f), (g, h), (b, e), (e, i)}
2. On the right, the cut ***does not respect*** A

Definitions

- Given a weighted undirected graph G = (V, E) and a cut (S, V-S), a ***light edge*** crossing the cut is a crossing edge with the smallest weight

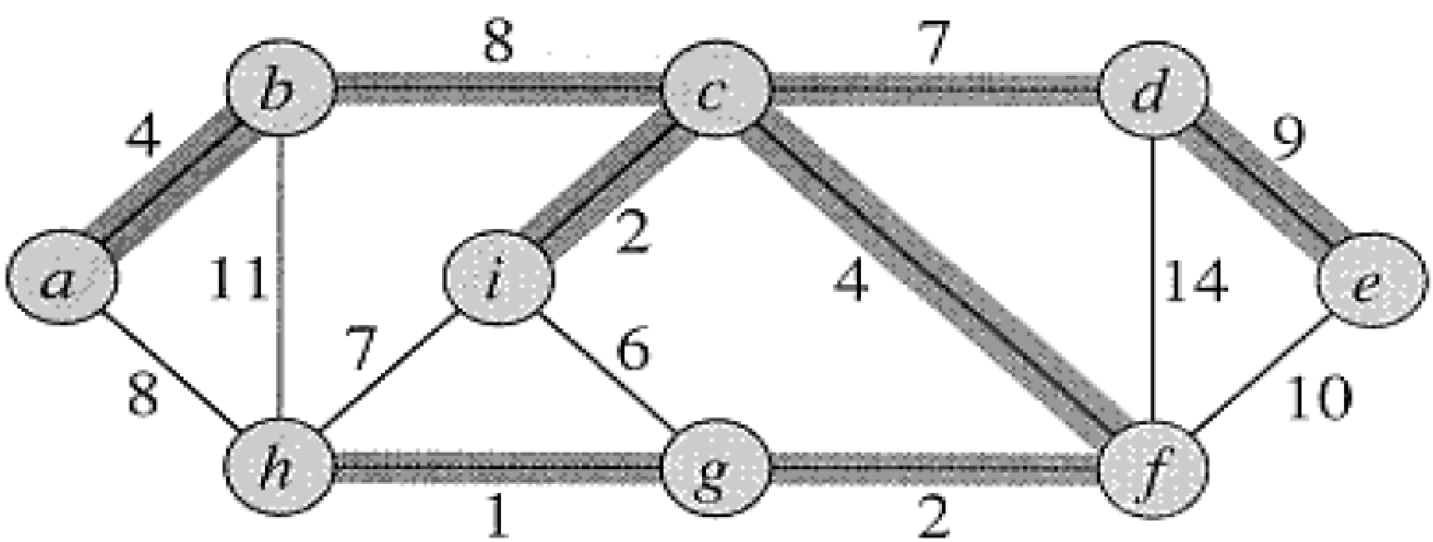


Example: Given a cut (S, V-S), where S = (D, E, H)

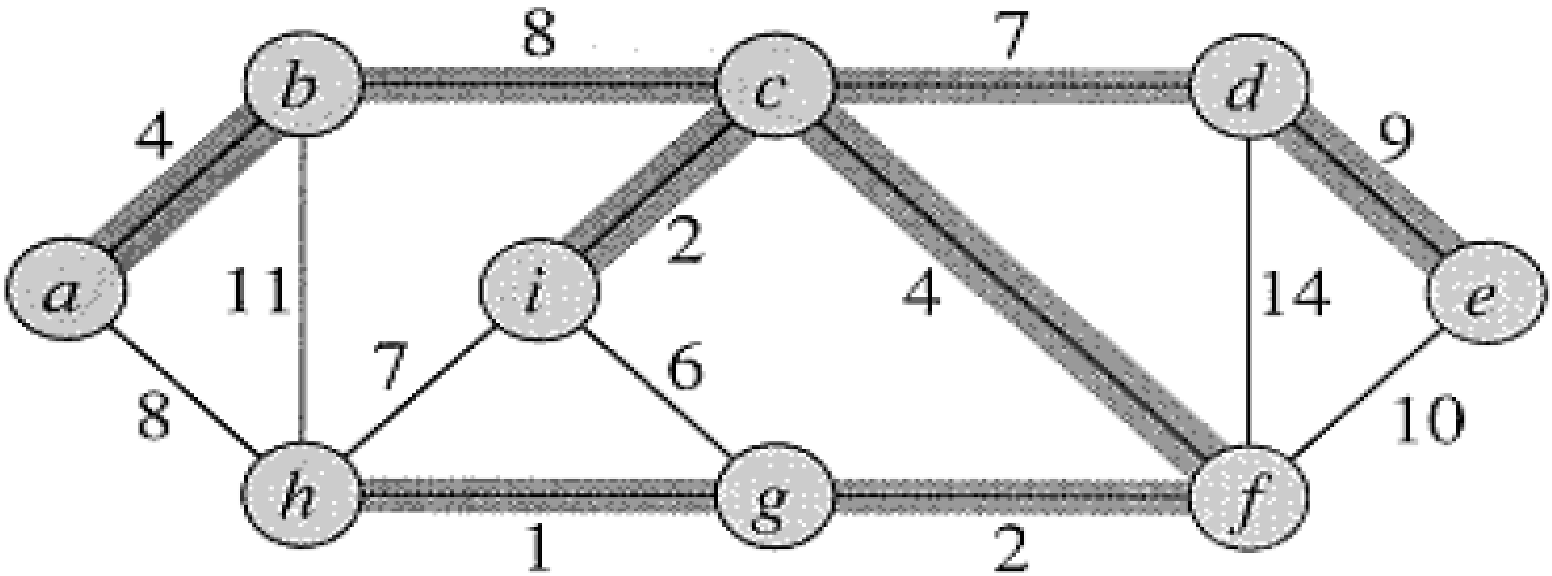
Edge (D, G) is a ***light edge*** because it is a crossing edge and has the smallest weight among all crossing edges

Definitions

- Given a weighted undirected graph $G = (V, E)$, a tree $T \subseteq E$ that spans all the vertices and whose total weight is minimum is called a ***minimum spanning tree***
- The total weight of a tree $T \subseteq E$ is the sum of weights of all edges in T



Minimum Spanning Tree, MST



$$w(T)=4+8+7+9+2+4+2+1=37$$

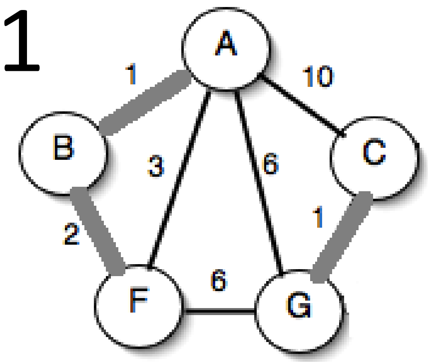
Note that the MST is not necessarily unique

For example, add (a,h), delete (b,c)

Minimum Spanning Tree, MST

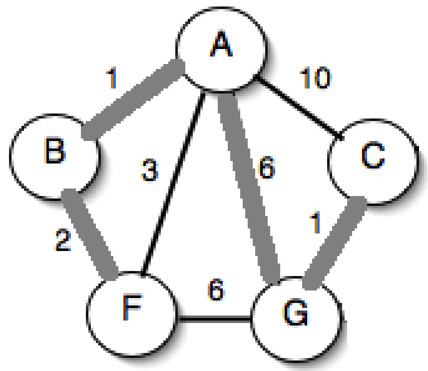
Identify a minimum spanning tree of the given graph

1



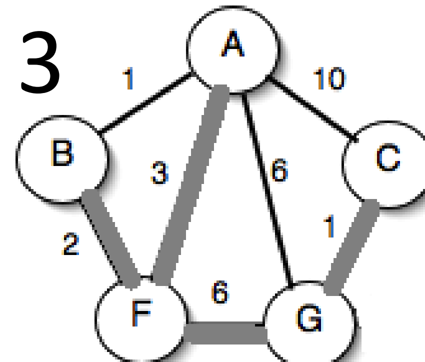
no

2



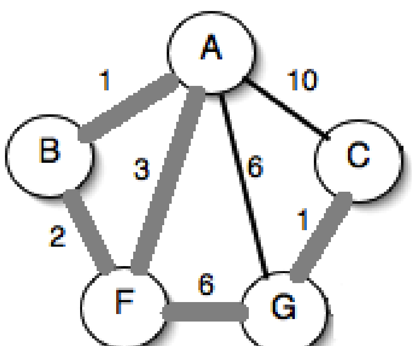
yes

3



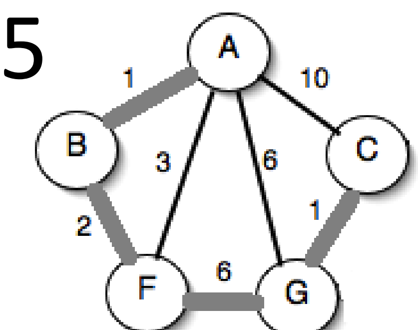
no

4



no

5



yes

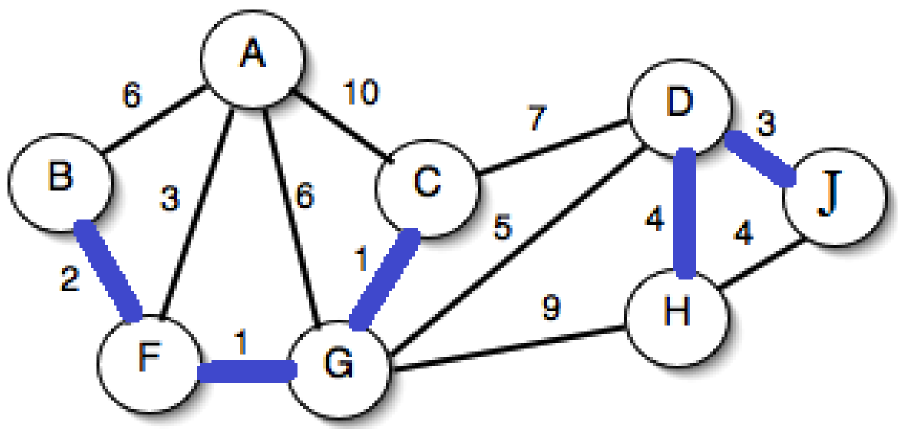
Generic MST algorithm

```
GENERIC-MST(G, w)
A ← Ø
while A does not form a spanning tree
      do find an edge (u, v) that is safe for A
            A ← A ∪ {(u, v)}
return A
```

Given a weighted undirected graph $G = (V, E)$ and a set of edges $A \subseteq T$, where T is a MST of G, an edge (u, v) in E is ***safe*** for A if the set of edges $A \cup (u, v)$ is a subset of some MST of G

Definitions

Given a weighted undirected graph G = (V, E) and a set of edges A $\subseteq$ T, where T is a MST of G, an edge (u, v) in E is ***safe*** for A if the set of edges A $\cup$ (u, v) is a subset of some MST of G



Example:

1. Edge (H, J) is ***not safe*** for A since the union of A and (H, J) contains a cycle and, hence, cannot be a subset of any MST
2. Edge (A, F) is ***safe*** for A

Theorem

Given:

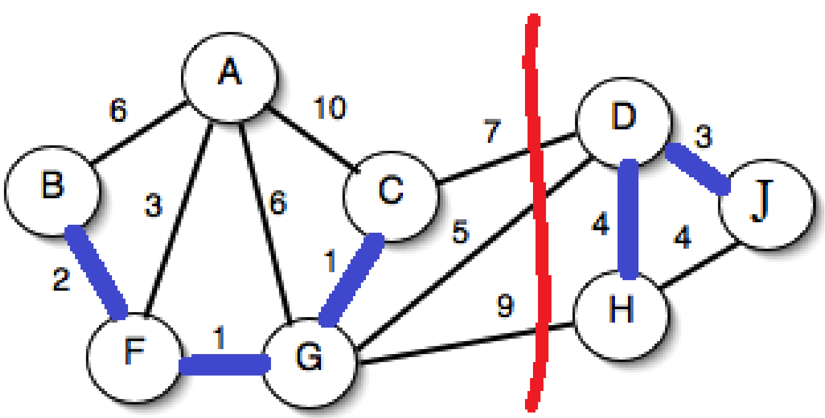
— Weighted undirected graph G = (V, E)

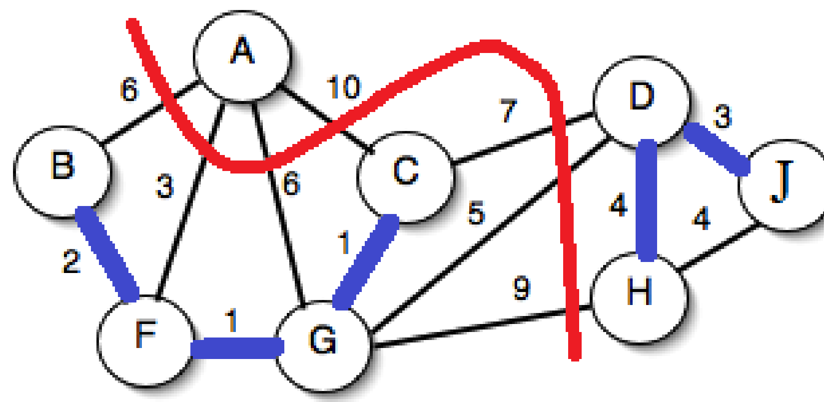
— Set of edges $A \subseteq T$, where T is a MST for G

— Cut (S, V-S) that respects A

— Light edge (u, v) crossing (S, V-S)

Then edge (u, v) is ***safe*** for A

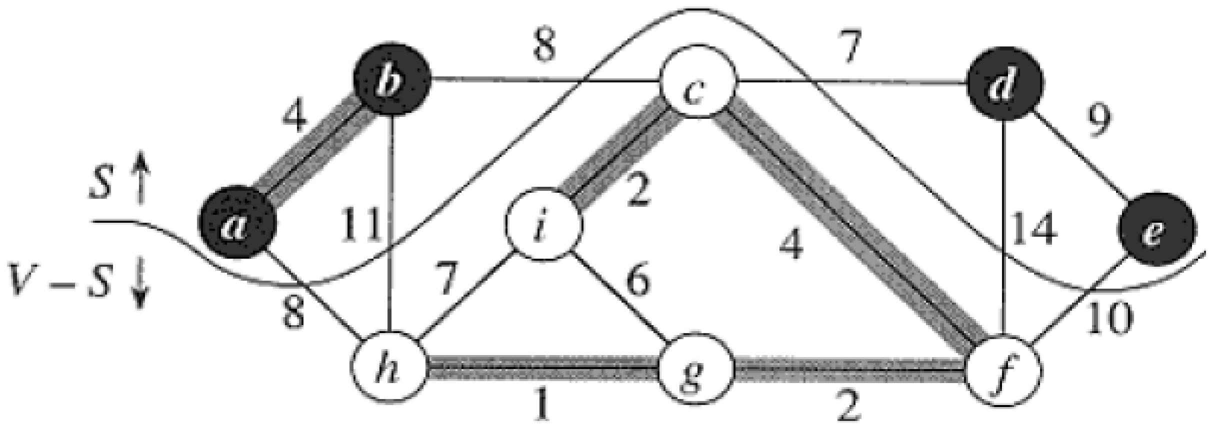


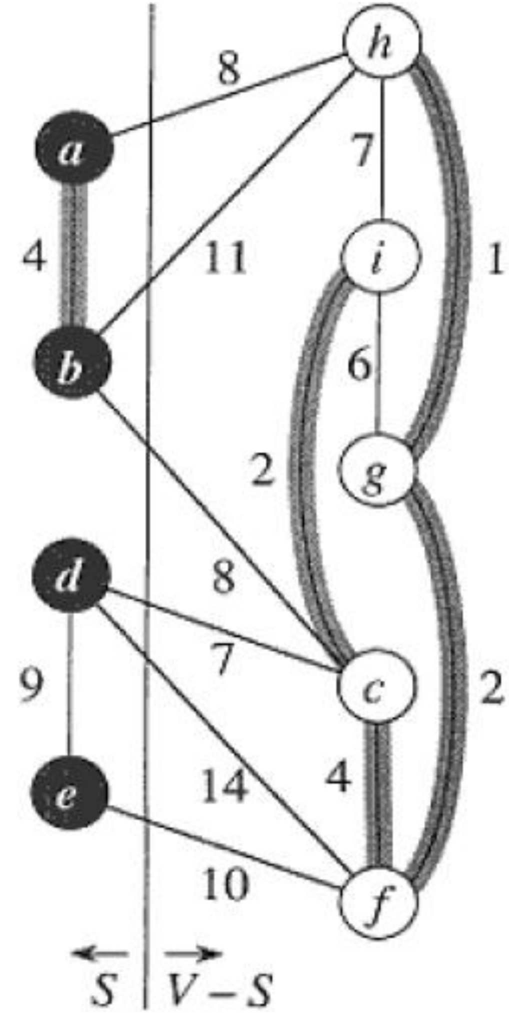


Example:

1. Edge (D, G) is ***safe*** for A
2. Edge (A, F) is ***safe*** for A

MST





Definitions

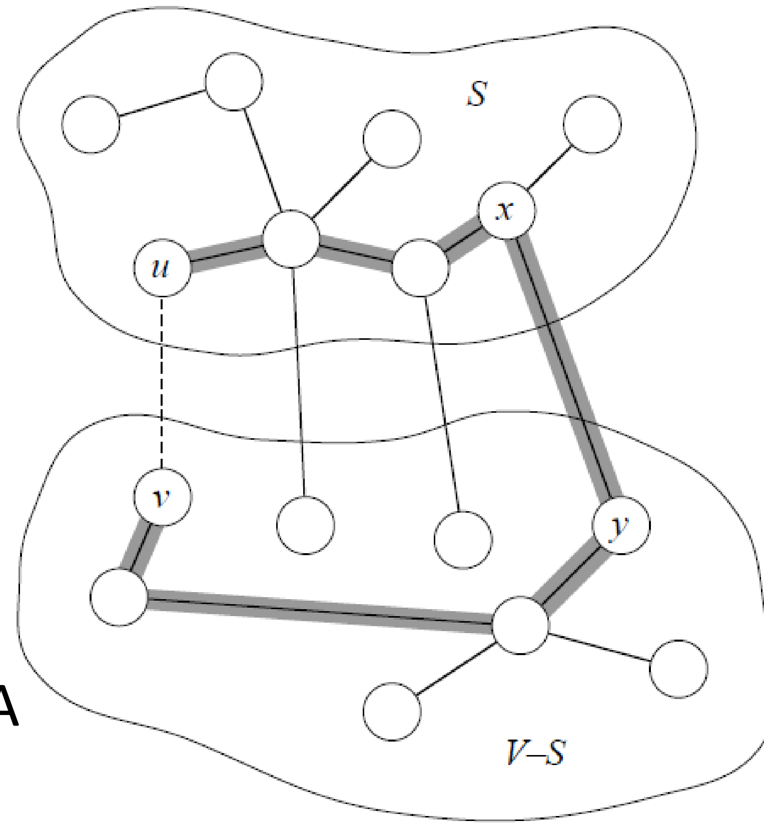
- Cut *(S, V-S):* a partition of *V*
- Crossing edge: one endpoint in *S* and the other in *V-S*
- A cut respects a set of *A* of edges if no edges in *A* crosses the cut
- A light edge crossing a partition if its weight is the minimum of any edge crossing the cut

Theorem. Let A be a subset of E that is included in some MST T of $G=(V,E)$. Let $(S, V\text{-}S)$ be any cut of G that respects A, and let (u, v) be a light edge crossing $(S, V\text{-}S)$. Then, edge (u, v) is safe for A.

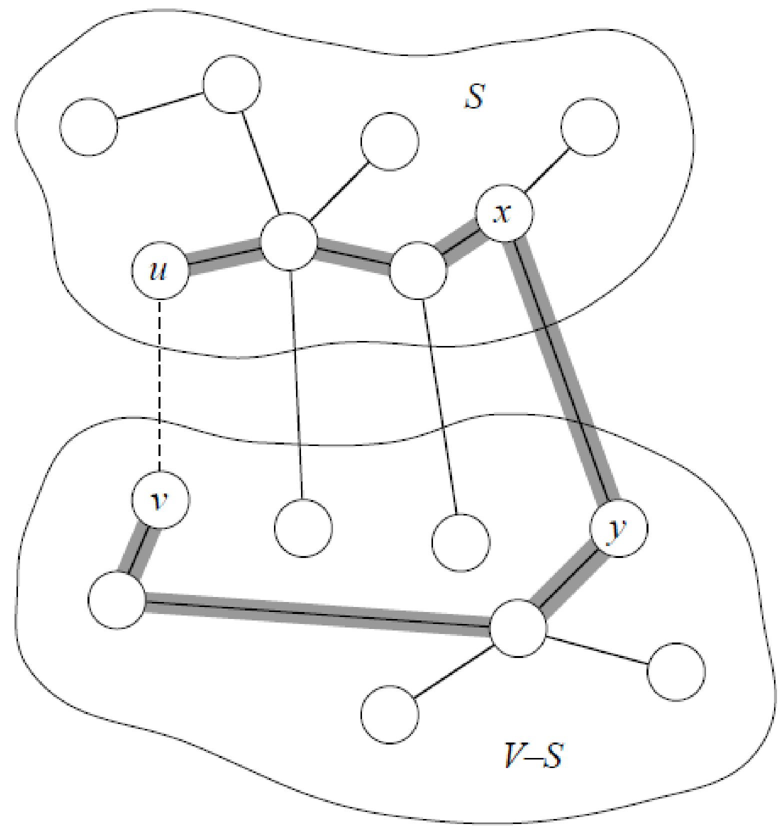
Case 1. T contains (u,v)

Case 2. T does not contain (u, v)

1. There is a path u to v in T
2. Adding (u,v) to T induces a cycle
3. Let (x,y) be crossing edge for (S, V-S) on this cycle
4. $w(u,v) \leq w(x, y)$ ((u,v) is light)
5. Let $T' = T - (x, y) \cup (u, v)$
6. T' is an MST containing all edges of A (cut respects A) and (u,v)
7. $W(T') = W(T) - w(x,y) + w(u,v) \leq W(T)$



Theorem. Let A be a subset of E that is included in some MST T of $G=(V,E)$. Let $(S, V\text{-}S)$ be any cut of G that respects A, and let (u, v) be a light edge crossing $(S, V\text{-}S)$. Then, edge (u, v) is safe for A.



MST

- Kruskal's algorithm
  - A is a forest
  - The safe edge added to A is always a least-weight edge in the graph that connects two distinct components
- Prim's algorithm
  - A is a single tree
  - The safe edge added to A is always a least-weight edge connecting the tree to a vertex not in the tree

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 (u, v) taken from the sorted list
      if FIND-SET(u) ≠ FIND-SET(v)
            A = A ∪ {(u, v)}
            UNION(u, v)
return A
```

Disjoint sets: tree representation

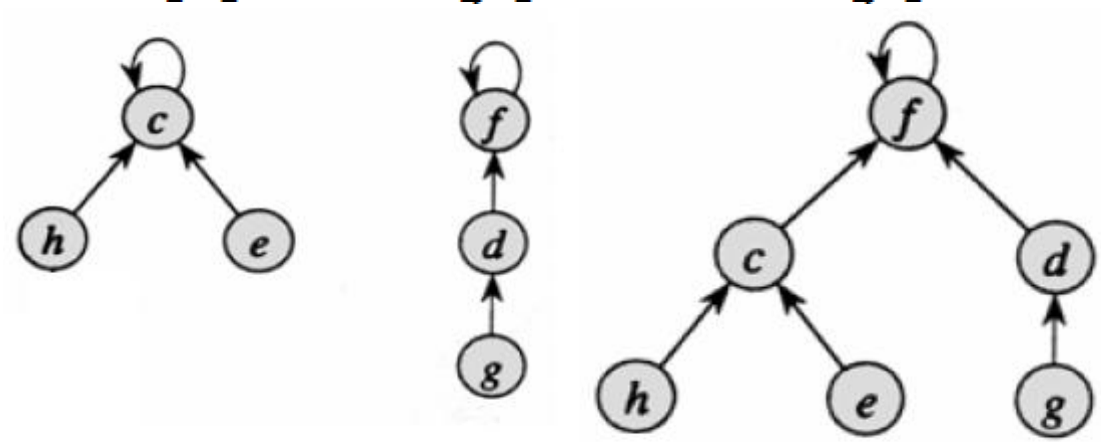
- In the worst case, no improvement over the linked list implementation
- Two heuristics allow us to achieve a running time almost linear in the total number of operations m (that is almost $O(1)$ amortized)
  - Union by rank
  - Path compression

Union by Rank

1. Keep ***rank*** for each node
2. Parent of a union is the root with higher rank

Union(x,y)
  Link(Find-Set(x), Find-Set(y))

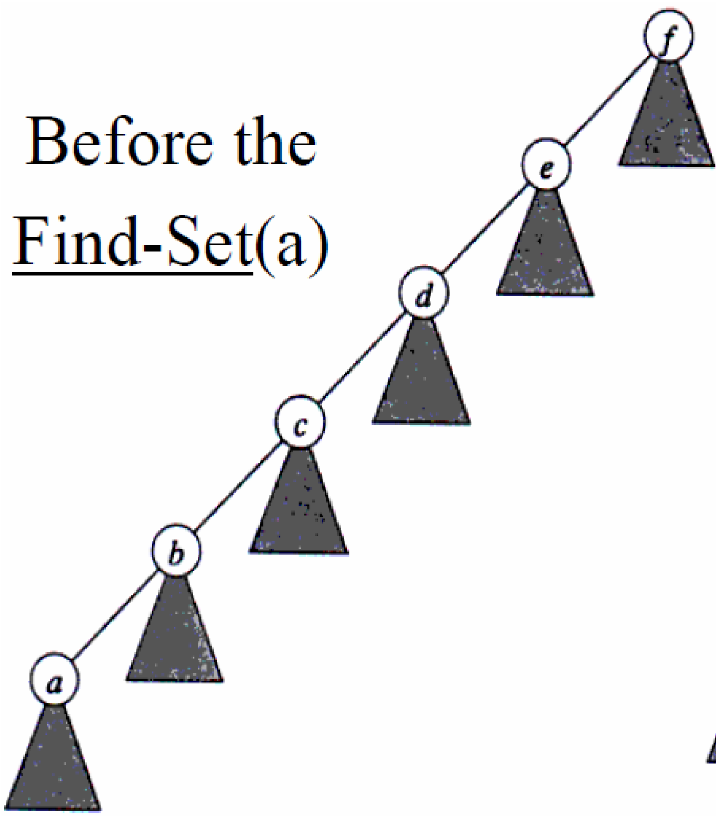
Link(x,y)
  if $rank[x] > rank[y]$ **then** $p[y] \leftarrow x$ /* x is the root */
  else $p[x] \leftarrow y$ /* y is the root */
    if $rank[x] = rank[y]$ **then** $rank[y] \leftarrow rank[y]+1$



Path Compression

- Speed up Union-Find operations by shortening the sub-tree paths to the root
- During a <u>Find-Set</u> operation, make each node on the find path point directly to the root

Before the <u>Find-Set</u>(a)

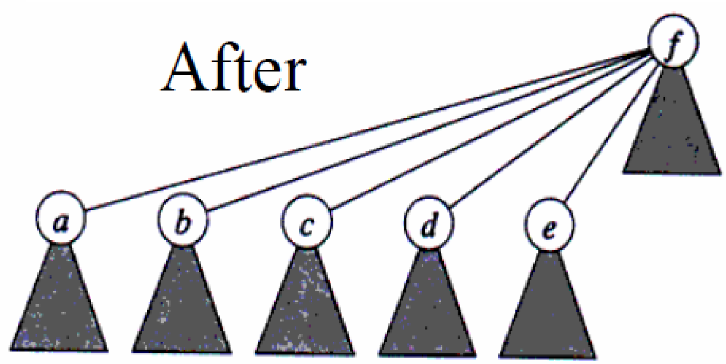


<u>Find-Set</u>(x)

if $x \neq p[x]$ **then** $p[x] \leftarrow$ <u>Find-Set</u>($p[x]$)

return $p[x]$

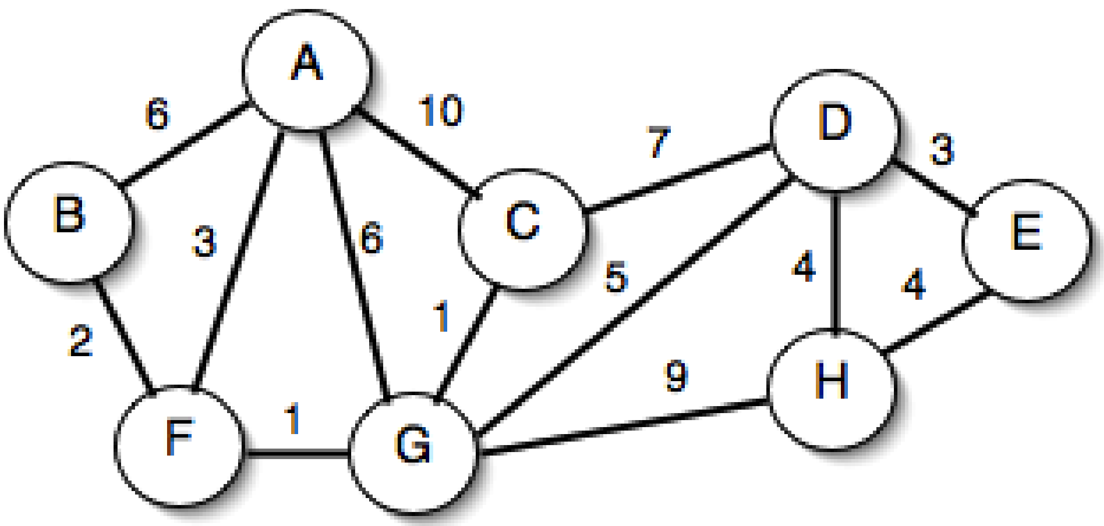
After



Run Kruskal's MST

KRUSKAL(G, w)

$A = \emptyset$
for each vertex $v \in G.V$
  MAKE-SET(v)
sort the edges by weight w
for each (u, v)
  if FIND-SET$(u) \neq$ FIND-SET(v)
    $A = A \cup \{(u, v)\}$
    UNION(u, v)
return A



```
KRUSKAL(G, w)
  A = ∅
  for each vertex v ∈ G.V
      MAKE-SET(v)
  sort the edges by weight w
  for each (u, v)
      if FIND-SET(u) ≠ FIND-SET(v)
          A = A ∪ {(u, v)}
          UNION(u, v)
  return A
```

Kruskal's running time

- m=edges, n=nodes
- Cost of creating the priority queue $O(m \log m)$ which is $O(m \log n)$
- $O(m)$ <u>Find-set</u> and <u>Union</u> and $O(n)$ Make-set, overall $O(m\,\alpha(n))$, $\alpha(n) \ll \log n$
- Overall running time is $O(m \log n)$

MST: Solve a Problem

- Given a weighted undirected graph G = (V, E) and already calculated MST T on G
- Design an efficient algorithm that calculates an MST on G after a new edge (u, v) is added to G on existing vertices u and v in V
- Prove the correctness of your algorithm
- Analyze time complexity

MST: Prove a Statement

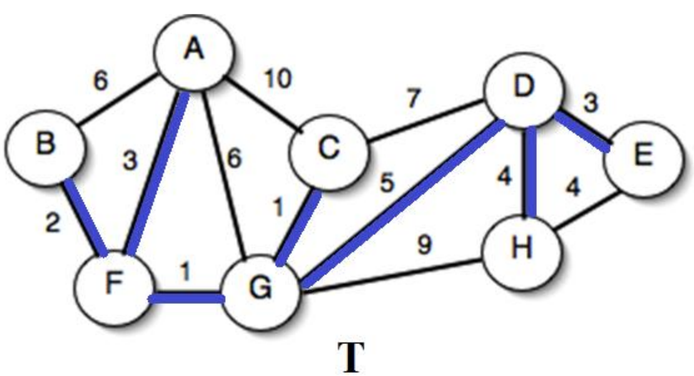
- Let (u, v) be a minimum-weight edge in a weighted connected graph G
- **Prove:** (u, v) belongs to some minimum spanning tree of G

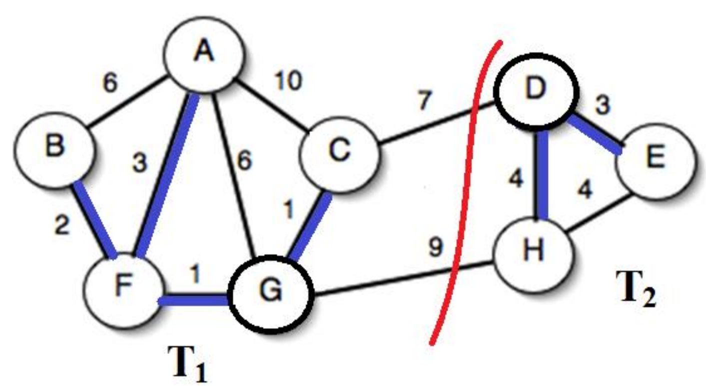
MST: Solve a Problem

- Given a weighted undirected graph G = (V, E) and already calculated MST T on G
- A new node ***v*** is added to G with ***k*** new edges incident at ***v*** and the other endpoints at vertices in G.
- Design an efficient algorithm that calculates an MST on the updated G
- Prove the correctness of your algorithm
- Analyze time complexity for the following case: k = O(1), k = O(V), E = O(V), E= O(V^2)

MST

- Let T be an MST of a weighted connected graph G and let (u, v) be an edge in T
- Removing (u, v) partitions T into two trees T_1 and T_2. Let (S, V-S) be a cut that respects T_1
- Let E_1 be the subset of non-crossing edges incident to S, and E_2 be the subset of non-crossing edges incident to V – S
- **Prove:** T_1 is an MST of $G_1 = (S, E_1)$ and T_2 is an MST of $G_2 = (V\text{-}S, E_2)$





Thank You