[CSE3081(2반)] 알고리즘 설계와 분석

2020학년도 2학기

강의자료

(2020.11.26 목요일)

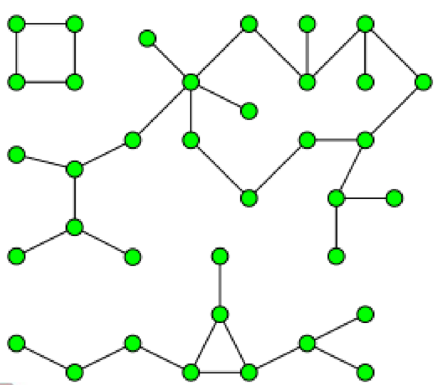
서강대학교 공과대학 컴퓨터공학과
임 인 성 교수

[주제 6]

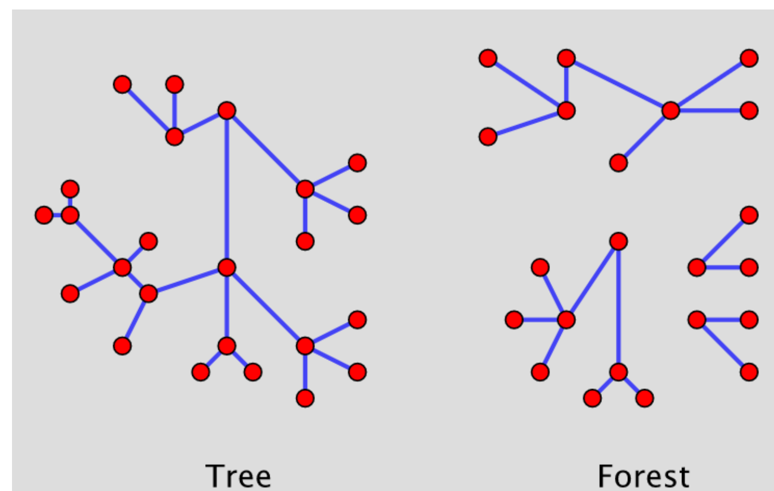
Graph Algorithms

Minimum Spanning Trees

- **Tree**
  - A tree is a connected graph T that contains no cycle.
  - Other equivalent statements (T = (V, E) where |V| = n)
    - T contains no cycles, and has n-1 edges.
    - T is connected, and has n-1 edges.
    - Any two vertices of T are connected by exactly one path.
    - T contains no cycles, but the addition of any new edge creates exactly one cycle.
- **Forest**
  - A forest is a graph with no cycles.

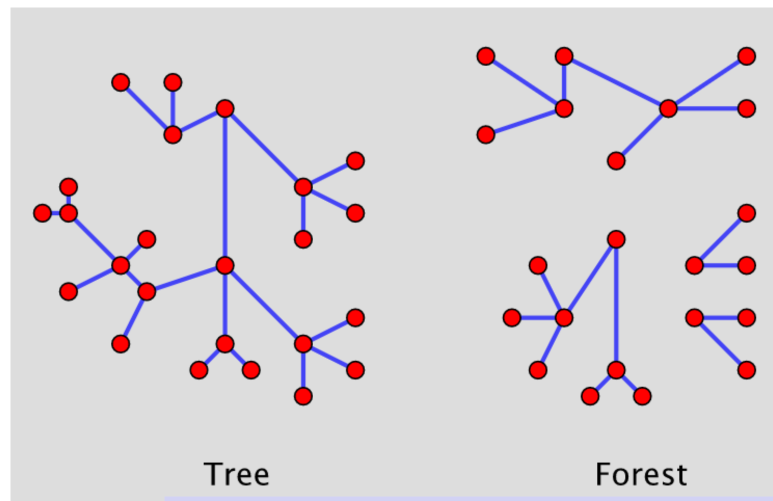


https://en.wikipedia.org/wiki



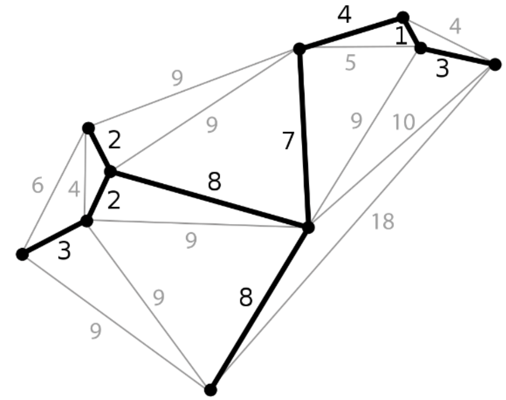
https://www.mathreference.org/

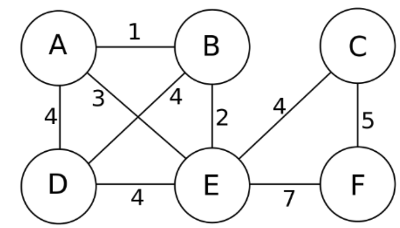
- **Buy-Two-Get-One-Free Theorem**
  - For a graph G = (V, E) with n vertices, any two of the following three properties imply the third one:
    ① G is connected.
    ② G is acyclic.
    ③ G has n-1 edges.



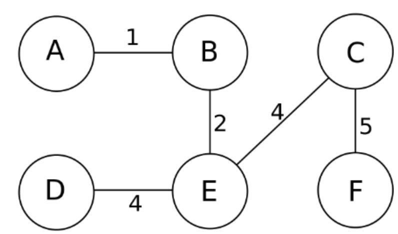
https://www.mathreference.org/

- **Minimum spanning tree**
  - A *spanning tree* for a graph G = (V, E) is a tree that contains all the vertices of G.
  - The *cost* of a spanning tree of a weighted graph G = (V, E) is the sum of the weights of the edges in the spanning tree.
  - A *minimum spanning tree* for a weighted graph G = (V, E) is a spanning tree of least cost.





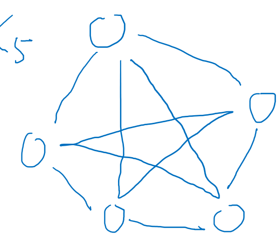
- **Problem**
  - Given a weighted graph G = (V, E), find a minimum spanning tree of G.

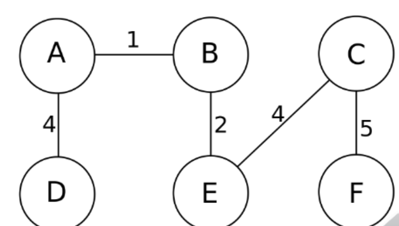


complete graph

- **A naïve approach**
  - Examine all the spanning trees of G, and take one having least cost.
  - ✓ There are n^{n-2} spanning trees in K_n!

K_5





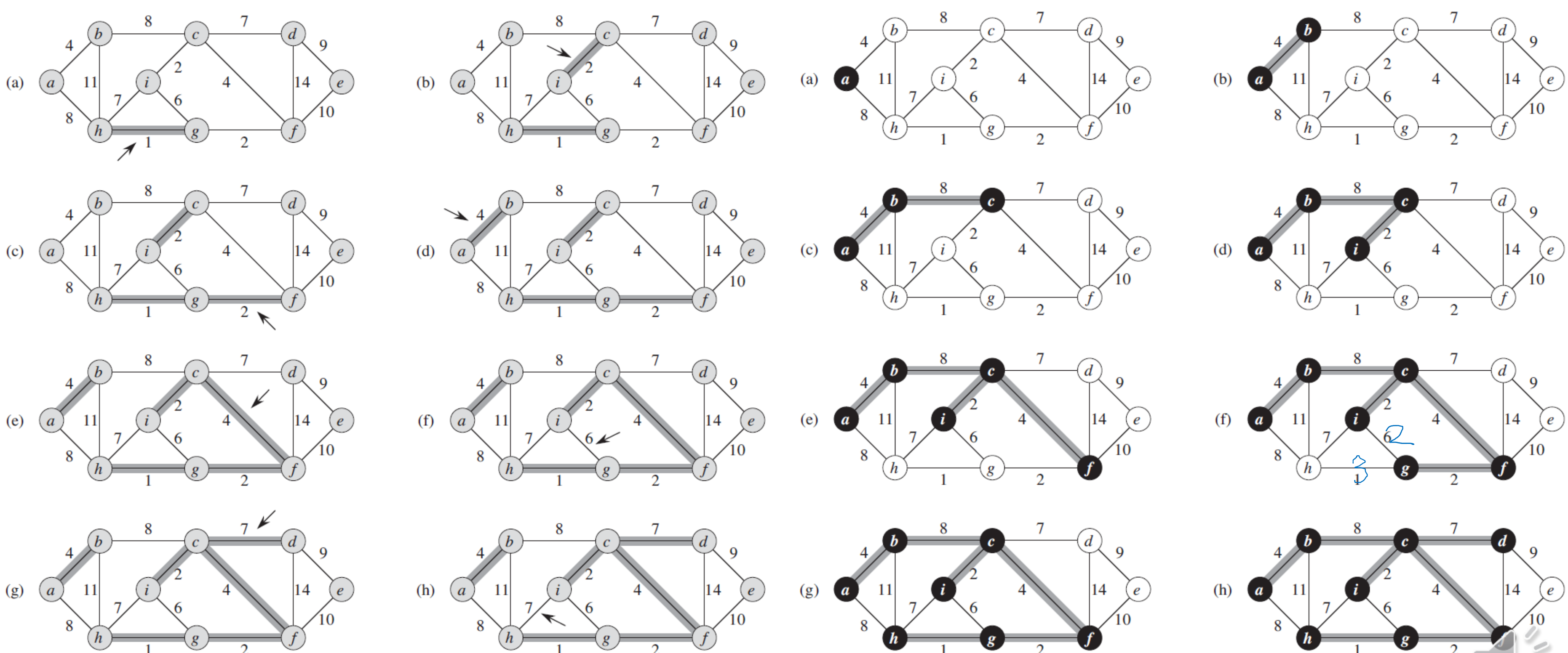
https://en.wikipedia.org/wiki/Minimum_spanning_tree

Kruskal's Algorithm vs Prim's Algorithm (Greedy!)

- **Kruskal's algorithm:** In each step, find and add **an edge of the least possible weight** that connects any two trees in the (current) forest.

- **Prim's algorithm:** In each step, find and add **an edge of the least possible weight** that connects the (current) tree to a non-tree vertex.

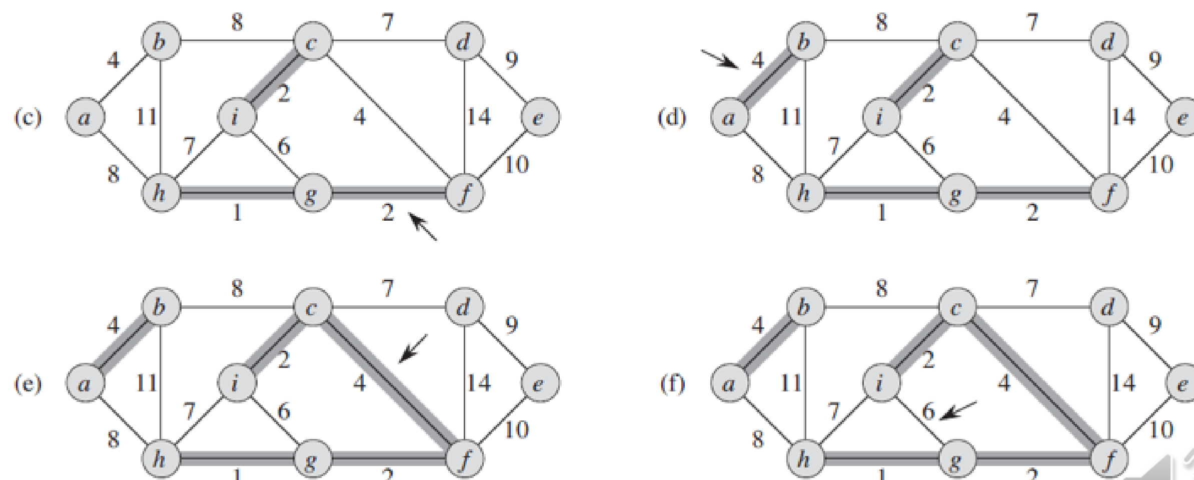
Courtesy of T. Cormen et al.



Generic MST Algorithm and its Correctness

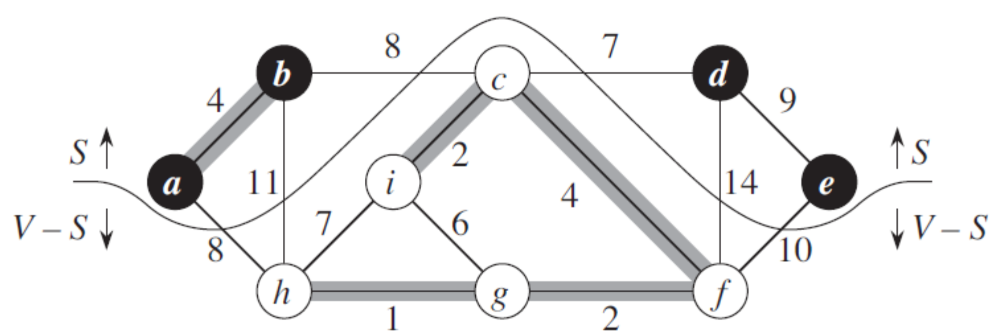
- **Generic algorithm for a graph G = (V, E) with a weight function w**
  - For an edge set A that is a subset of some MST, an edge (u, v) is called **a safe edge for A** if A ∪ {(u, v)} is also a subset of some MST.
  - **Loop invariant for a set of edges A**
    - *Prior to each iteration, A is a subset of some minimum spanning tree.*

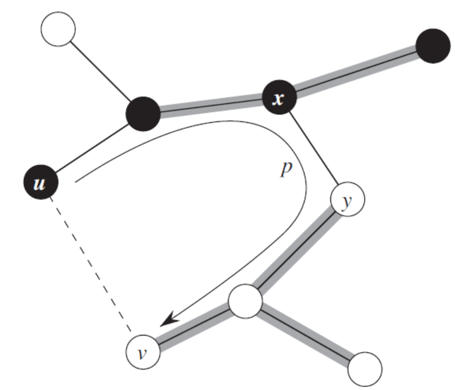
```
Generic-MST(G) {
    A := empty; // A: a set of edges of G
    While (A does not form a spanning tree) {
        Find and edge (u, v) that is safe for A;
        A := A ∪ {(u, v)};
    }
}
```



- **Some definitions**
  - A **cut** (S, V-S) of G is a partition of V.
  - An edge (u, v) of G **crosses a cut (S, V-S)** if u ∊ S and v ∊ V-S → **cut-set**.
  - A cut **respects** a set A of edges if no edge in A crosses the cut.
  - An edge is a **light edge crossing a cut** if its weight is the minimum of any edge crossing the cut.

Courtesy of T. Cormen et al.





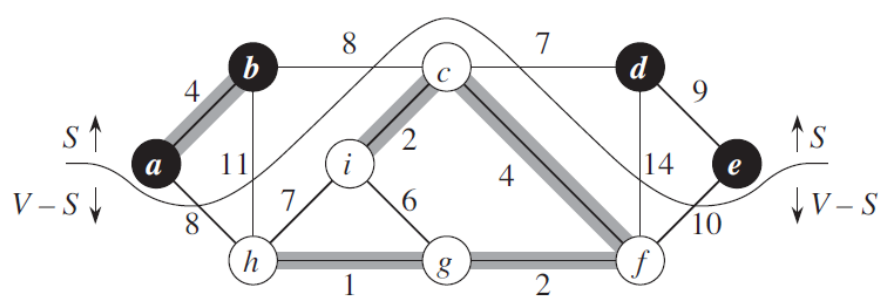
Cut Property

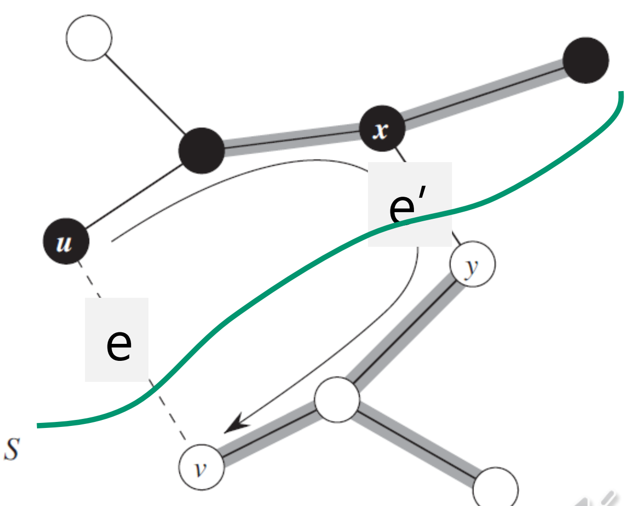
https://en.wikipedia.org/wiki/Minimum_spanning_tree

For any cut C of the graph, if the weight of an edge e in the cut-set of C is strictly smaller than the weights of all other edges of the cut-set of C, then this edge belongs to all MSTs of the graph.

Proof: Assume that there is an MST T that does not contain e. Adding e to T will produce a cycle, that crosses the cut once at e and crosses back at another edge e' . Deleting e' we get a spanning tree T∖{e'}∪{e} of strictly smaller weight than T. This contradicts the assumption that T was a MST.

- ✓ By a similar argument, if more than one edge is of minimum weight across a cut, then each such edge is contained in some minimum spanning tree.

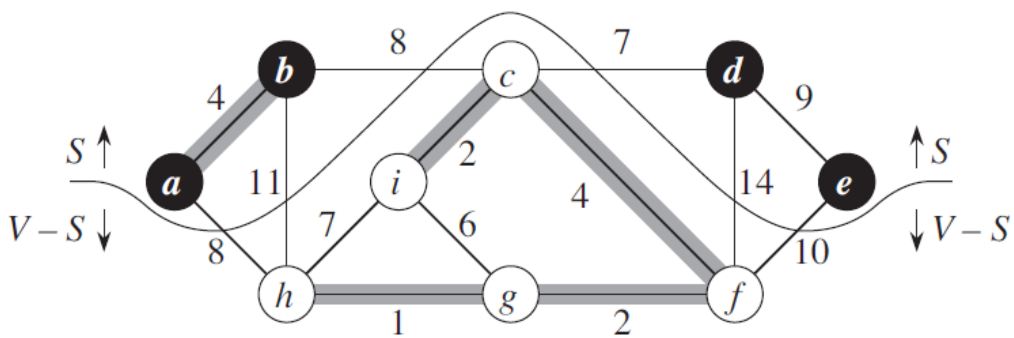


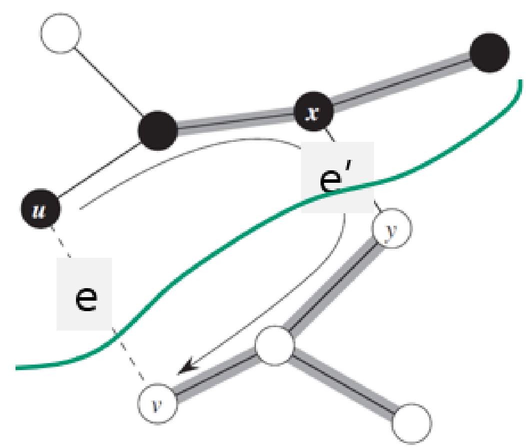


```
Generic-MST(G) {
    A := empty; // A: a set of edges of G
    While (A does not form a spanning tree) {
        Find and edge (u, v) that is safe for A;
        A := A ∪ { (u, v) };
    }
}
```

Loop invariant for the set A

- *Prior to each iteration, A is a subset of some minimum spanning tree.*





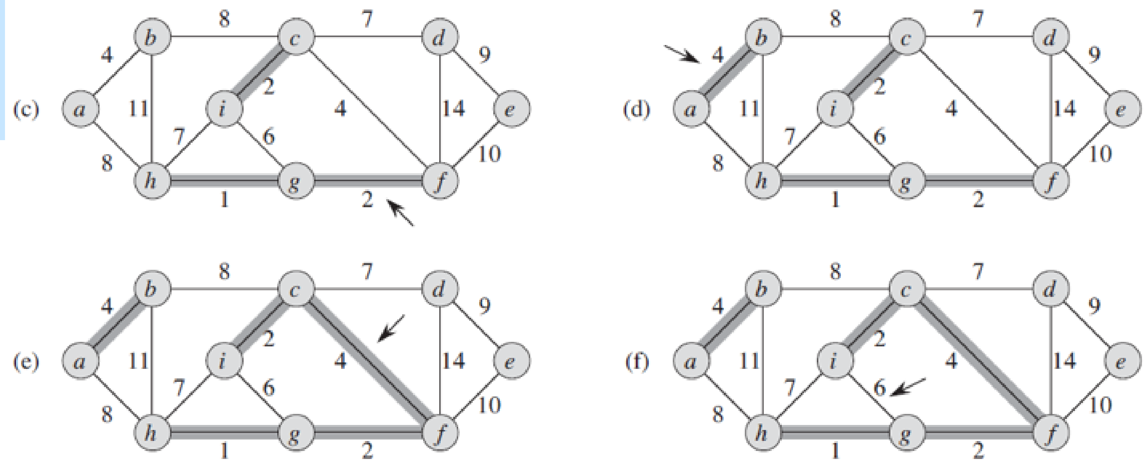
Theorem

Let G = (V, E) be a connected, undirected graph with a real-valued weight function w defined on E. Let **A** be a set of E that is included in some minimum spanning tree for G, let **(S, V-S)** be any cut of G that respects A, and let **(u, v)** be a light edge crossing (S, V-S). Then, **edge (u, v) is safe for A**.

Selection of Next Edge: Kruskal's Algorithm

Generic-MST(G) {
  A := empty; // **A: a set of edges of G**
  While (A does not form a spanning tree) {
    Find and edge **(u, v)** that is **safe for A**;
    A := A ∪ { **(u, v)** };
  }
}

In each step, find and add **an edge of the least possible weight** that connects any two trees in the (current) forest.



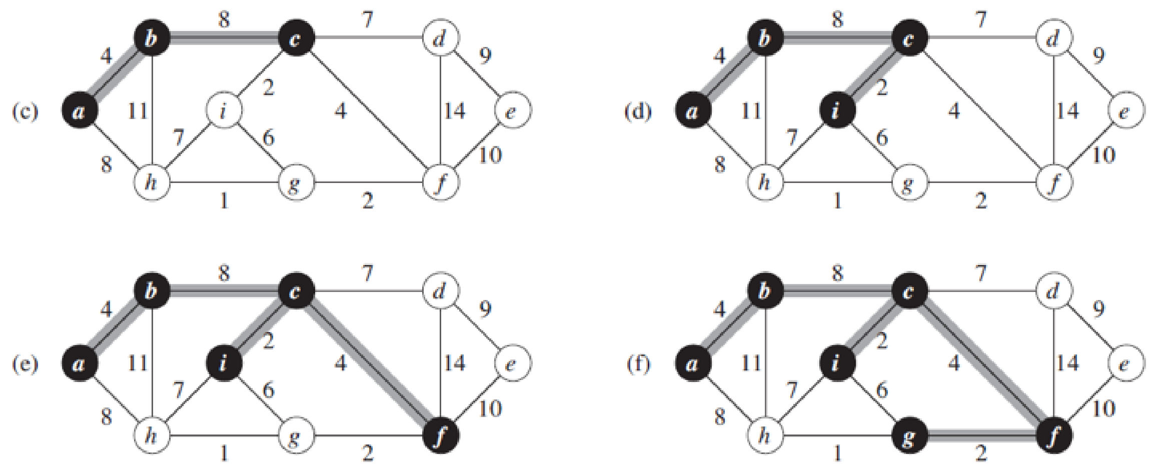
Theorem

Let G = (V, E) be a connected, undirected graph with a real-valued weight function w defined on E. Let **A** be a set of E that is included in some minimum spanning tree for G, let **(S, V-S)** be any cut of G that respects A, and let **(u, v)** be a light edge crossing (S, V-S). Then, **edge (u, v) is safe for A**.

Selection of Next Edge: Prim's Algorithm

Generic-MST(G) {
  A := empty; // **A: a set of edges of G**
  While (A does not form a spanning tree) {
    Find and edge **(u, v)** that is **safe for A**;
    A := A ∪ { **(u, v)** };
  }
}

In each step, find and add **an edge of the least possible weight** that connects the (current) tree to a non-tree vertex.



Theorem

Let G = (V, E) be a connected, undirected graph with a real-valued weight function w defined on E. Let **A** be a set of E that is included in some minimum spanning tree for G, let **(S, V-S)** be any cut of G that respects A, and let **(u, v)** be a light edge crossing (S, V-S). Then, **edge (u, v) is safe for A**.