CS 430 Lecture 26 Activities

Opening Questions

1. What is the difference between a tree and a graph? All trees are graphs, but not all graphs are trees. In a tree, there is only 1 path between any 2 vertices. Trees are also acyclic since no item can have multiple parents. If trees have $|V|$ nodes, then there are $|V| - 1$ edges.

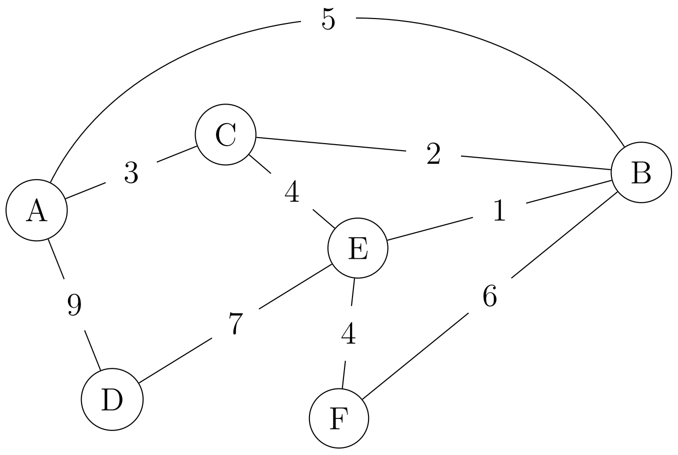
2. Give a recursive definition for a tree. Base case: single node with no children. A tree is . . . a node pointing to other trees with no cycles.

3. In a weighted undirected graph, what is the difference between a minimum spanning tree and a shortest path in a graph? A minimum spanning tree is made up of the edges of minimum total weight that still span (connect) all vertices in the graph. This does not assure the minimum path between all vertices. A shortest path is the minimum path distance between two vertices.

4. Since the shortest paths contain the shortest sub-paths (optimal substructure), name an algorithmic approach that we might try to find a shortest path in a graph. Greedy algorithms and Dynamic Programming.

Minimum Spanning Trees (MST)

1. Give a definition of a Minimum Spanning Tree, and find an MST of the below graph. An MST is a graph that spans over all vertices with no cycles. $|V| - 1$ edges.



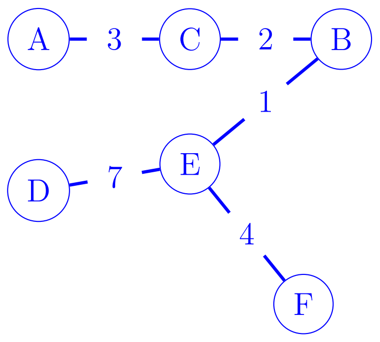


Figure 26.1: One spanning tree of Figure 1.. Total weight: 22.

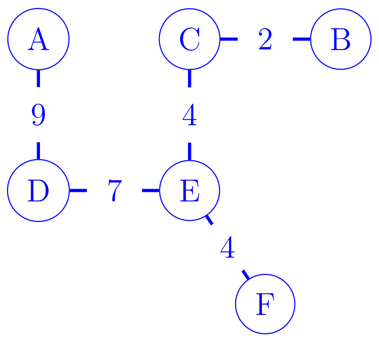


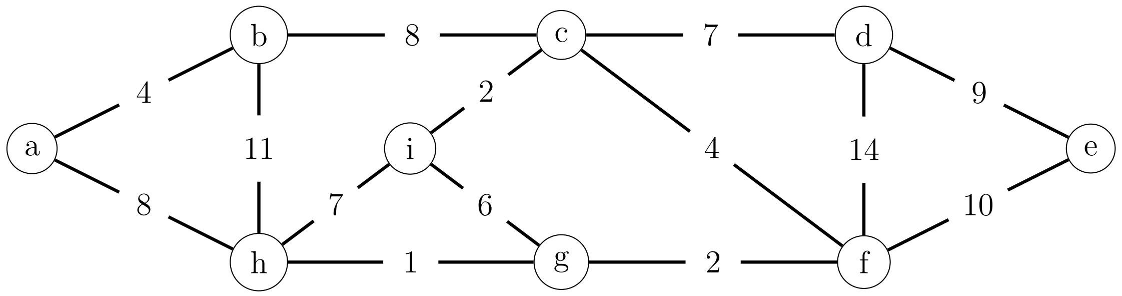
Figure 26.2: Another spanning tree of Figure 1.. Total Weight: 26.

2. Prove a Minimum Spanning Tree has optimal substructure. Pick any subsets of adjacent vertices in an optimal MST. Those edges that connect that subset of vertices in the MST must also be MST.

3. What are some possible greedy approaches to find a Minimum Spanning Tree? Prove correct or show counterexample.

   - Grow min edges first; no cycles.
   - Prune/remove max edge, but stay connected.
   - Creating edges from visited nodes to unvisited nodes.
   - **Prim**: Min edge from visited vertex set to unvisited vertex set.
   - **Kruskal**: Pick min edge that its vertices are not already in the connected component.

4. Demonstrate your MST algorithm on the following graph and write pseudocode.



Node	**Visited?**
a	
b	
c	Visited
d	
e	
f	
g	
h	
i	

Started at c.

Algorithm 26.1 Prim's Algorithm (MST)

```
function MST-PRIM($G$, W, $r$)
    for all $u \in G.V$ do
        $u.key \leftarrow \infty$
        $u.\pi \leftarrow$ NIL
    end for
    $r.key \leftarrow 0$
    $Q \leftarrow G.B$
    while $Q \neq \emptyset$ do
        $u \leftarrow$ EXTRACT-MIN($Q$)
        for all $v \in G.Adj[u]$ do
            if $v \in Q$ and W$(u, v) < v.key$ then
                $v.\pi \leftarrow u$
                $v.key \leftarrow$ W$(u, v)$
            end if
        end for
    end while
end function
```

Algorithm 26.2 Kruskal's Algorithm (MST)

```
function MST-KRUSKAL($G$, $w$)
    $A \leftarrow \emptyset$
    for all vertex $v \in V[G]$ do
        MAKE-SET($v$)
    end for
    sort the edges of $E$ into increasing
    for all edge $(u, v) \in E$, taken in increasing order do
        if FIND-SET($u$) $\neq$ FIND-SET($v$) then
            $A \leftarrow A \cup \{(u, v)\}$
            UNION($u$,$v$)
        end if
    end for
end function
```

Demonstration of Prim (Deleted): http://en.wikipedia.org/wiki/File:Prim-algorithm-animation-2.gif
Demonstration of Kruskal: https://www.cs.usfca.edu/~galles/visualization/Kruskal.html

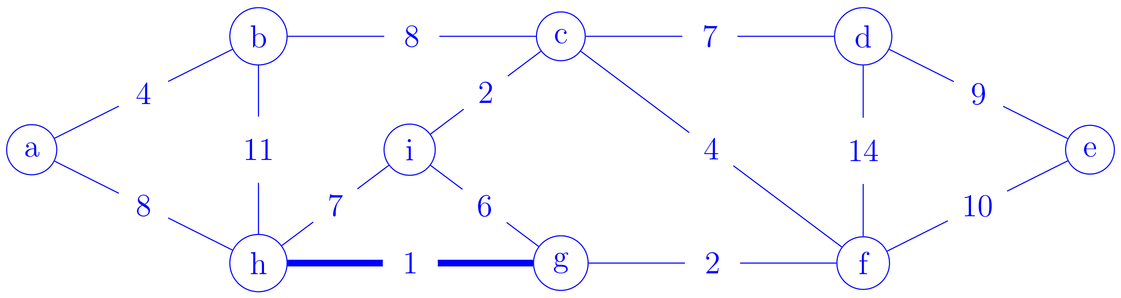


Figure 26.3: Connected vertices h and g with an edge with weight 3.

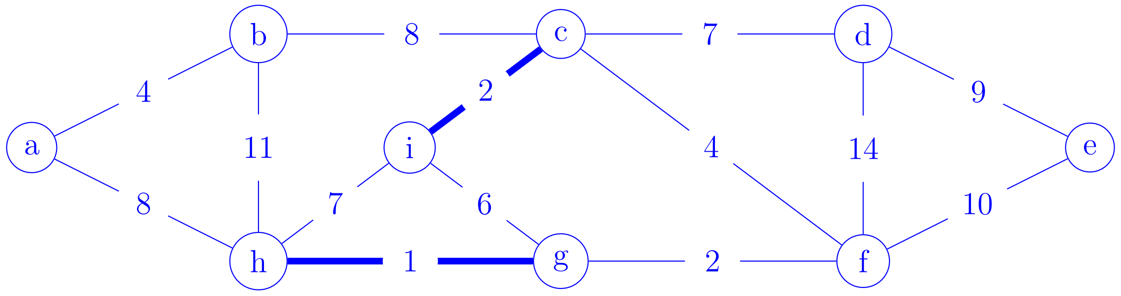


Figure 26.4: Connected vertices c and i with an edge with weight 3.

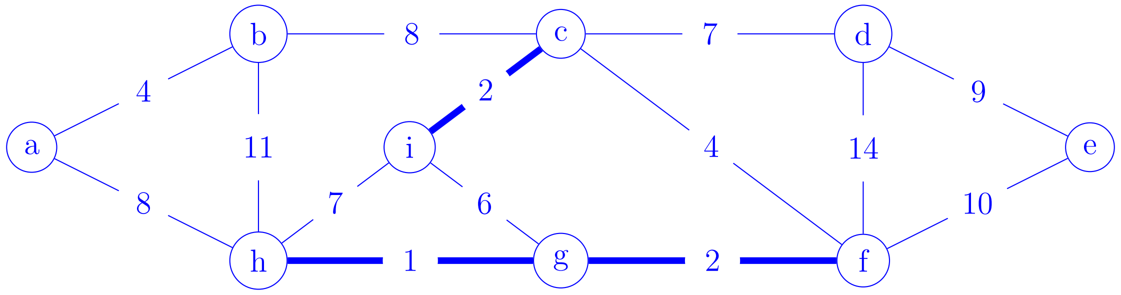


Figure 26.5: Connected vertices g and f with an edge with weight 3.

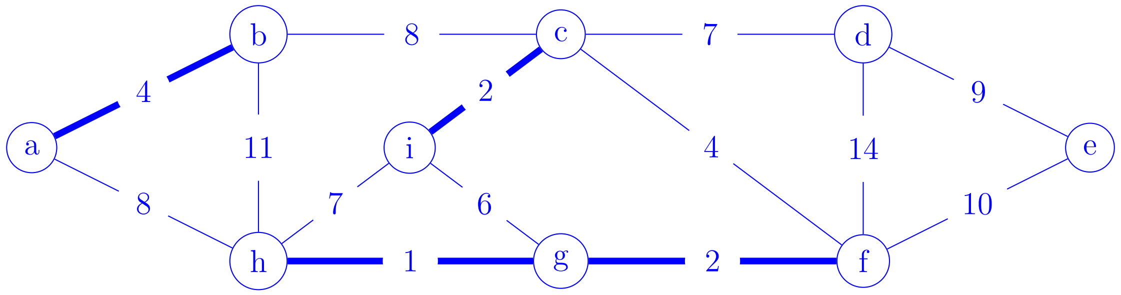


Figure 26.6: Connected vertices a and b with an edge with weight 3.

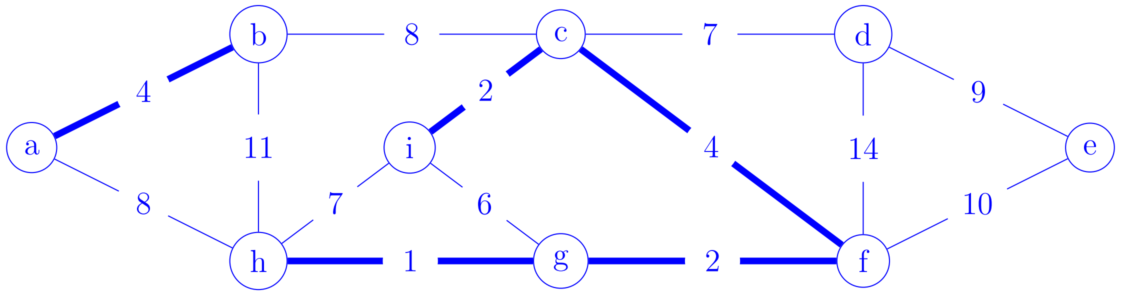


Figure 26.7: Connected vertices c and f with an edge with weight 3.

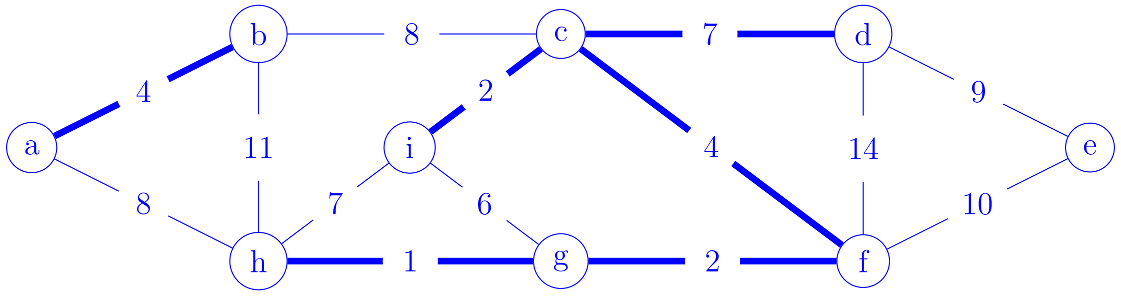


Figure 26.8: Connected vertices c and d with an edge with weight 3.

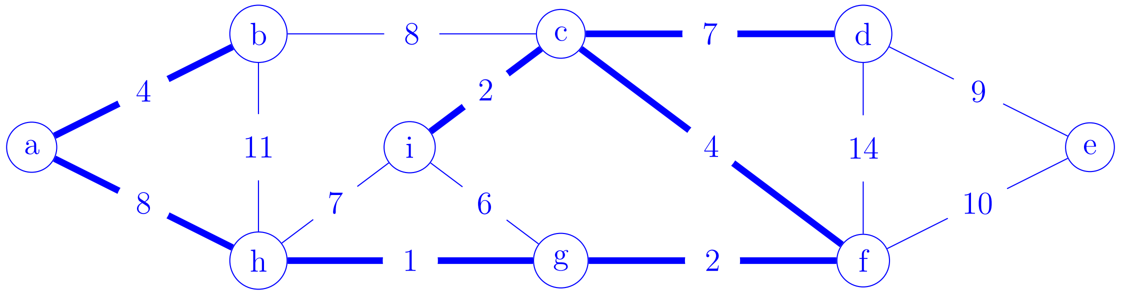


Figure 26.9: Connected vertices a and h with an edge with weight 3.

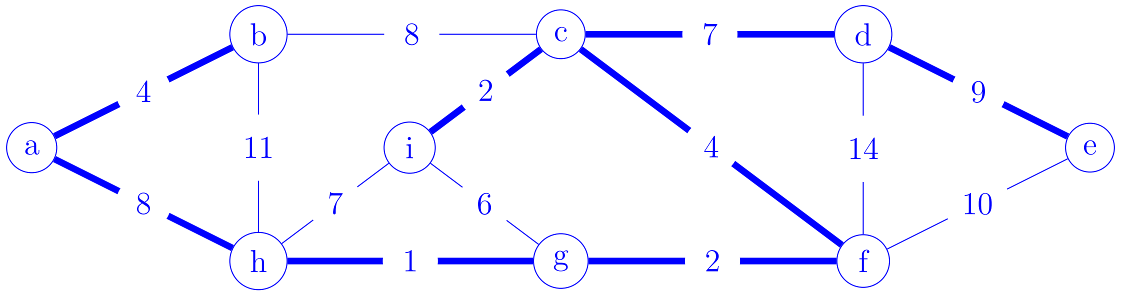


Figure 26.10: Connected vertices d and e with an edge with weight 3.