Jade Yu Cheng
ICS 311
Homework 7
Sep 18, 2008

Question for lecture 8

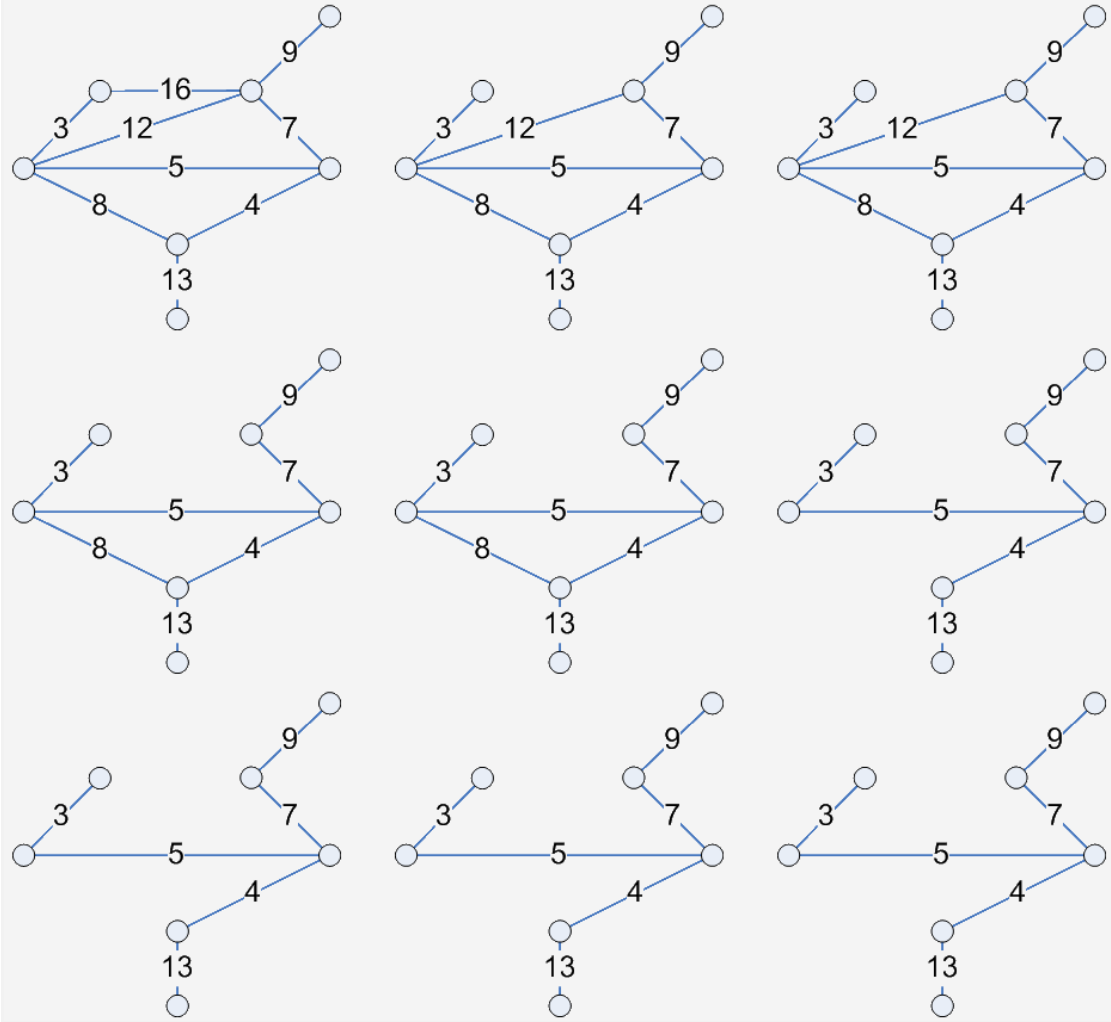
Problem 23-4 on p. 578

**Alternative minimum-spanning-tree algorithms**
In this problem, we give pseudocode for three different algorithms. Each one takes a graph as input and returns a set of edges T. For each algorithm, you must either prove that T is a minimum spanning tree or prove that T is not a minimum spanning tree. Also describe the most efficient implementation of each algorithm, whether or not it computes a minimum spanning tree.

a.

```
MAYBE-MST-A(G, w) {
1     sort the edges into nonincreasing order of edge weights w;
2     T ← E;
3       for each edge e, taken in nonincreasing order by weight {
4          do if T – {e} is a connected graph
5             then T ← T – {e};
6       }
7     return T;
8   }
```

**Answer:** The first algorithm is a MST algorithm. The procedure is shown as the graph below. We get a MST with a total edge weight of $3+5+7+4+13+9=41$. This algorithm implements a greedy algorithm. The greedy function is to find and delete the next most weighted edge, such that after the deletion the graph is still a connected graph.




Proof:

Consider, we have two subparts of the final MST, $T_1$ and $T_2$. According to the Optimal Substructure Theorem, the subtree $T_1$ with $G_1(V_1, E_1)$ and subtree $T_2$ with $G_2(V_2, E_2)$ are MSTs as well. There exist vertices $u' \in V_1$, $u \in V_1$ and $v' \in V_2$, $v \in V_2$.

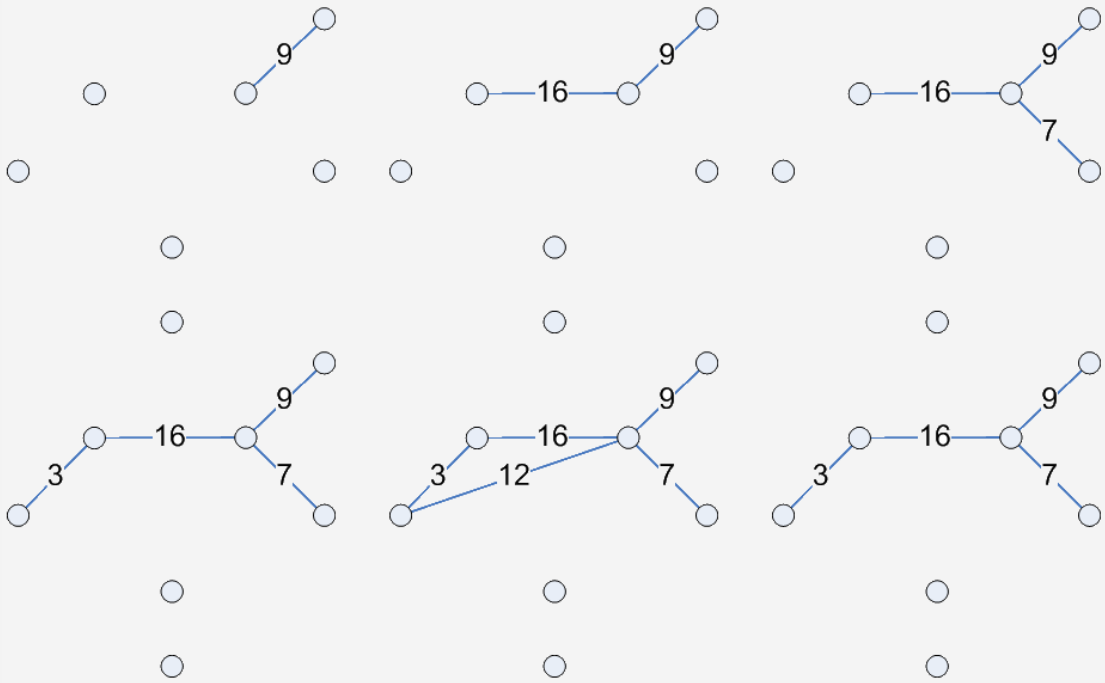
Suppose, we have edge $(u', v') \notin T$ that gives a shorter path to connect subtree $T_1$ and $T_2$. Both $(u', v') \notin T$ and $(u, v) \in T$ meet the second requirement in the algorithm. That is after the deletion of either $(u', v') \notin T$ or $(u, v) \in T$, the rest of the graph is still a connected graph because it is two MSTs connected through either $(u', v') \notin T$ or $(u, v) \in T$ depending on which one is deleted. Therefore the decision of which one to delete is determined by the weight of these two edges. If $(u', v') \notin T$ is lighter than $(u, v) \in T$, the algorithm would have picked $(u', v') \notin T$ instead of $(u, v) \in T$.

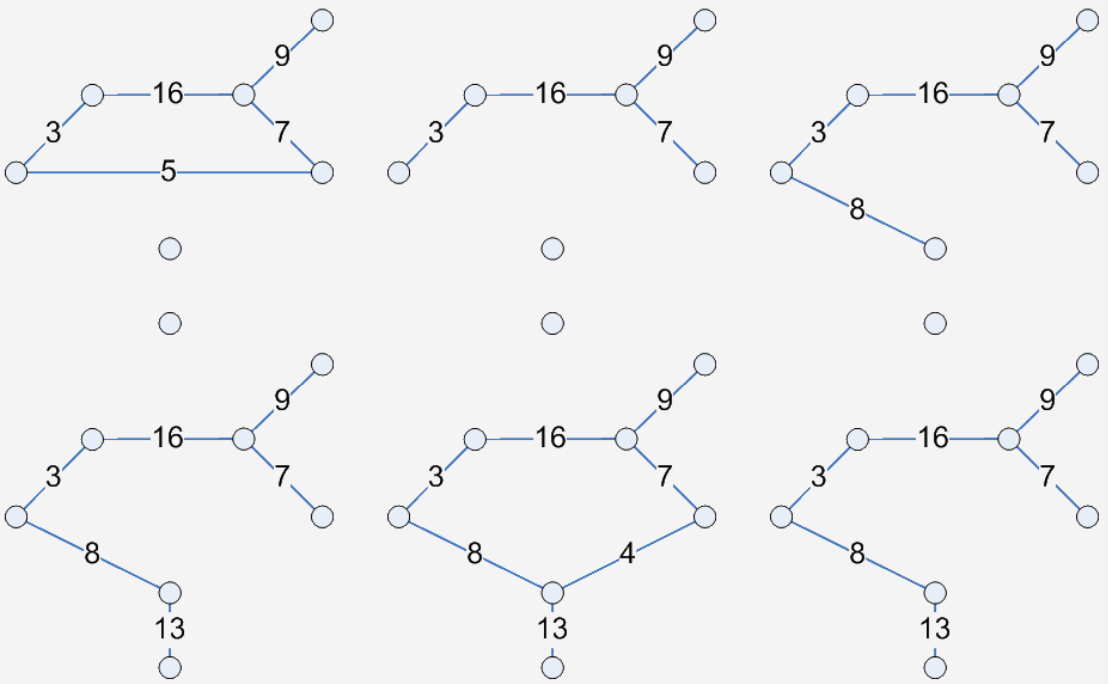
Therefore, MAYBE-MST-A(G, w) is a MST algorithm.

b.

```
MAYBE-MST-B(G, w) {
1     T ← ∅;
2       for each edge e, taken in arbitrary order {
3         do if T ∪ {e}
4           then T ← T ∪ {e};
5       }
6     return T;
7   }
```

**Answer:** The second algorithm is not a MST algorithm. The operations of the pseudocode are shown as below. We get a tree structure with a total weight of $16+3+7+8+13+9=56$. Obviously, $56>41$. This is not a MST algorithm.






Proof:

Consider we have two subparts of the final MST, $T_1$ and $T_2$. According to the Optimal Substructure Theorem, the subtree $T_1$ with $G_1(V_1, E_1)$ and subtree $T_2$ with $G_2(V_2, E_2)$ are MSTs as well. There exist vertices $u' \in V_1$, $u \in V_1$ and $v' \in V_2$, $v \in V_2$.

Suppose we have edges $(u', v') \notin T$ and $(u, v) \in T$. There are two different cases to consider after this assumption:

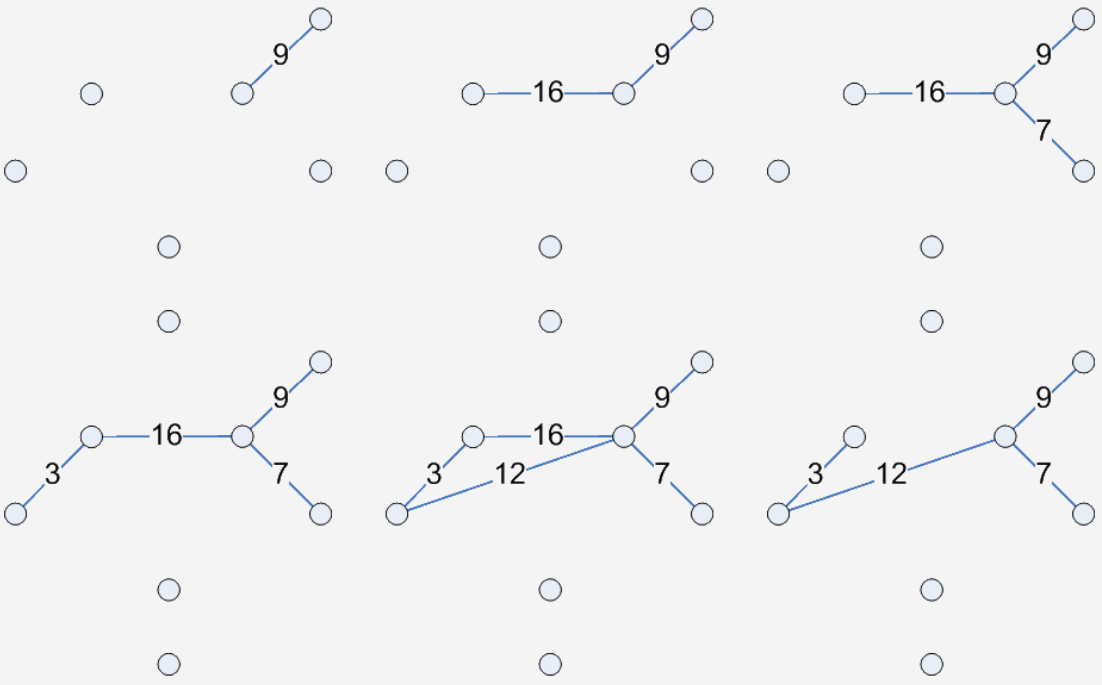
1. Edge $(u', v') \notin T$ and edge $(u, v) \in T$ do not form a circle when both of them exist at the same time. If we delete both of them at the same time, the rest of the graph would become three individual MSTs. This is a confliction with the presupposition that there are only two MSTs before the operation of adding either $(u', v') \notin T$ or $(u, v) \in T$ into the picture. Therefore this case will never happen.
2. Edge $(u', v') \notin T$ and edge $(u, v) \in T$ form a circle when both of them exist at the same time. Based on the algorithm, the last inserted one between $(u', v') \notin T$ and $(u, v) \in T$ would be deleted, leaving the first inserted one. At the same time, the algorithm also says the order of the insertion is random. Therefore, there is no guarantee that the earlier inserted ones are also the lighter ones.

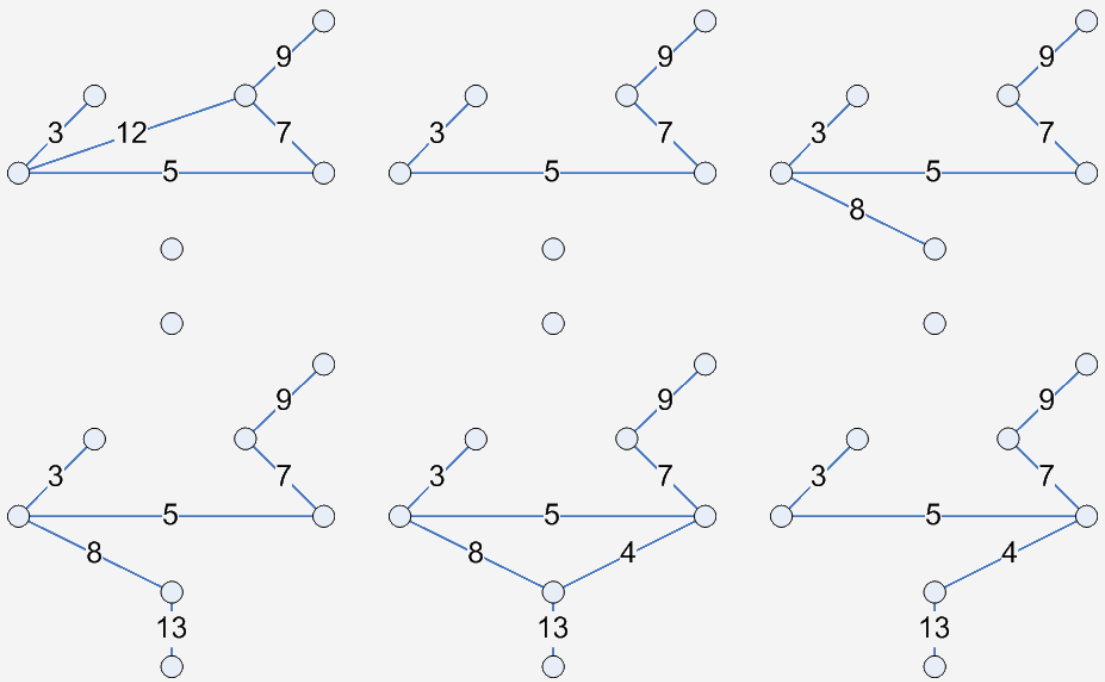
Therefore, MAYBE-MST-B(G, w) is not a MST algorithm.

c.

```
MAYBE-MST-C(G, w) {
1       T ← ∅;
3         for each edge e, taken in arbitrary order by weight {
4           do T ← T ∪ {e};
5             if T has a cycle c {
6               Then let e' be a maximum-weight edge on c;
7               T ← T – {e};
8             }
9         }
10      return T;
11    }
```

**Answer:** The third algorithm is a MST algorithm. The procedure is shown as the graph below. We get a MST with a total weight of $3+5+7+4+13+9=41$.
The algorithm also implements a greedy algorithm. The greedy function is to find the most weighted edge in the current existing circle and delete that edge. If no circle currently exists, continue adding a random edge until the next circle appears.






Proof:

Consider we have two subparts of the final MST, $T_1$ and $T_2$. According to the Optimal Substructure Theorem, the subtree $T_1$ with $G_1(V_1, E_1)$ and subtree $T_2$ with $G_2(V_2, E_2)$ are MSTs as well. There exist vertices $u' \in V_1$, $u \in V_1$ and $v' \in V_2$, $v \in V_2$.

Suppose we have edge $(u', v') \notin T$ that gives a shorter path to connect subtrees $T_1$ and $T_2$. There are two different cases to consider after this assumption:

1. Edge $(u', v') \notin T$ and edge $(u, v) \in T$ do not form a circle when both of them exist at the same time. If we delete both of them at the same time, the rest of the graph would become three individual MSTs. This is a confliction with the presupposition that there are only two MSTs before the operation of adding either $(u', v') \notin T$ or $(u, v) \in T$ into the picture. Therefore this case will never happen.
2. Edge $(u', v') \notin T$ and edge $(u, v) \in T$ form a circle when both of them exist at the same time. Based on the algorithm, the more weighted one between $(u', v') \notin T$ and $(u, v) \in T$ would be deleted, leaving the lighter one. If $(u', v') \notin T$ gives a shorter path to connect subtrees $T_1$ and $T_2$, $(u', v') \notin T$ would have been chosen instead of $(u, v) \in T$.

Therefore, MAYBE-MST-C(G, w) is a MST algorithm.