# Homework #6
# Introduction to Algorithms/Algorithms 1
# 600.463
# Spring 2016

**Due on:** Thursday, March 24th, 11.59pm
**Submitted By:** Sindhuula Selvaraju
**JHED ID:** sselvar4

March 24, 2016

## 1 Problem 1 (25 points)

You are given a simple weighted graph $G = (V, E)$. Suppose that $e \in E$ is an edge that belongs to the cycle $C$ inside $G$, and has the maximum weight among other edges in $C$. Prove that minimum spanning tree of $G$ does not contain $e$.
**ANSWER:**
This is the Cycle property of MST and can be proved by contradiction.
We know: $G = (V, E)$, $e \in E$, cycle $C$ inside $G$. $e$ has maximum weight in $C$
To Prove: Minimum spanning tree(MST) of $G$ does not contain $e$
Proof by Contradiction:
Assume $e$ is present in the MST.
Assume that $e$ connects vertices $A$ and $B$
If we remove $e$ from the MST this should essentially form 2 sub-trees(Assume T' with V' vertices and T" with V-V' vertices) one which contains $A$ and other contains $B$.
A cut on a cycle would intersect at even number of edges, since the number of entry points into a graph for a cycle has to be the same as the number of exit points. Thus there will be another way to go from $A$ to $B$(the roundabout way. Imagine in a triangle XYZ instead of directly going from X to Y we first go from X to Z and then Z to Y).
There will be another edge $f$ in cycle $C$ which is the counterpart of e and will weigh less than e (since to form a cycle there needs to be atleast 3 edges) with

exactly one vertex in T'
Since the other vertex of $f$ will be in T" and
Weight($f$) $<$ Weight($e$)
So the MST formed by taking $f$ in place of $e$ will have a lower weight and the MST cannot have $f$ and $e$ both as it would result in a cycle which is not possible in an MST.
This contradicts our assumption that MST will have $e$ since there is a possibility of a lower weighted MST
Hence $e$ will not be in the MST
Hence Proved

## 2 Problem 2 (25 points)

You are given a simple weighted graph $G = (V, E)$ and its minimum spanning tree $T_{mst}$. Somebody added new edge $e$ to the graph $G$. Let's call new graph $G' = (V, E \cup e)$. Devise an algorithm which checks if $T_{mst}$ is also a minimum spanning tree for the graph $G'$ or not. Prove correctness of your algorithm and provide running time analysis.

You can assume that all edge weights in $G'$ are distinct. $T_{mst}$ and $G'$ are given to you as adjacency lists.

Your algorithm should work in $O(|V|)$ time for a full score. For the algorithm working in $O(|V|^{1+\varepsilon})$ time (for any positive $\varepsilon$) you will get 15 points.

**ANSWER:**

**Algorithm:** The main concept behind the algorithm is assuming that the new edge is added between $A$ and $B$, we need to check the current path from $A$ to $B$(or $B$ to $A$) in the $T_{mst}$. We can do this by iterating over the $T_{mst}$ using DFS with root as $A$ and traverse through $T_{mst}$ till we reach $B$. If we encounter any edge which has a greater value than the new edge $AB$ it means that the $T_{mst}$ is not the MST for the new graph. NOTE: Assume DFS Algorithm takes the $T_{mst}$, source and destination as input and returns weight of every edge in the path between between $A$ and $B$.

**Algorithm 1:** Finding if $T_{mst}$ is the MST for graph G' with added edge

**Input :** $T_{mst}$ for graph G, Vertices $A$ and $B$

**Output:** true if $T_{mst}$ will remain the same, false if $T_{mst}$ will change

```
CheckMST($T_{mst}$, $A$, $B$, )
{
weights[] = DFS($T_{mst}$, $A$, $B$)
for i in weights: {
        if (i > weight($AB$))
                return false;
}
return true;
}
```

**Proof of Correctness:** Since, only 1 edge is added to the graph to make G into G' and no vertices are added, this new edge will create a cycle because before connecting $A$ and $B$ a path must have already existed between $A$ and $B$ and adding the new path only closes the cycle. If this edge has the highest weight in the cycle formed it will not be part of the MST(as seen in Problem 1). On the other hand, if it does not have the highest weight then some other edge will have weight higher than this edge and will not be part of the MST. Since, this higher weighted edge is removed from the cycle a new MST will have to be formed. Since, all weights in G' are distinct one of the above 2 conditions will definitely occur. Since we know DFS is correct and complete if there exists a path between $A$ and $B$ it will definitely traverse it and since $T_{mst}$ definitely has a path between all vertices DFS will return the list of weights.

Hence Proved.

**Running Time Analysis:** Modified version of DFS will only expand vertices between $A$ and $B$ in $T_{mst}$. In the worst case DFS takes O($|V|$) The for loop will also atmost iterate over V Vertices and will take O($|V|$) comparisons. Hence, the running time is O($|V|$)