CS1340: DISCRETE STRUCTURES II

PRACTICE QUESTIONS IV– ANSWERS

(1) Suppose 1000 people enter a chess tournament. Use a rooted tree model of the tournament to determine how many games must be played to determine a champion, if a player is eliminated after one loss and games are played until only one entrant has not lost. (Assume there are no ties.)

Proof: The tournament can be modelled using a full 2-ary tree/binary tree. There are going to be 1000 leaves as there are 1000 people. Each internal vertex will represent the winner of the game played by its children. The root will be the final winner. We have $m = 2$, $l = 1000$. Therefore,

$$\begin{aligned} i &= \frac{l-1}{m-1} \\ &= \frac{1000-1}{2-1} \\ &= 999. \end{aligned}$$

We have 999 internal vertices, so 999 games must be played.

(2) Every tree $T = (V, E)$ with $|V| \geq 2$ has at least two vertices that have degree 1.

Proof: Every tree has $n-1$ edges, so the the sum of the degrees of all the vertices of any tree have to be $2(n-1)$. But if there are fewer than two vertices of degree one, then the sum of the degrees of all the vertices must be at least $2n-1$, and contradicts previous statement.

(3) Prove that a graph with distinct edge weights has a unique minimum spanning tree.

Proof: If T_1 and T_2 are distinct MSTs then consider the edge of min weight among all the edges that are contained in exactly one of T_1 or T_2. We assume that e_1 is the edge and it is contained in T_1. Now T_2 with e_1 in it will introduce a cycle and one of the edges in T_2 is not in T_1 since T_1 does not contain a cycle.

Since e_2 is an edge different from e_1 and is not in both the trees this implies $wt(e_1) < wt(e_2)$. Consider $T = T_2 \cup \{e_1\} \setminus \{e_2\}$. It is a spanning tree (Show!) and total weight of T is smaller than T_2 but this contradicts the fact that T_2 is a MST.

(4) Let x and y be two nodes of a binary tree B. Prove that x is an ancestor of y iff x stands before y in the pre-order traversal of B and x stands after y in the post-order traversal of B.

Proof: If x is an ancestor of y then in a preorder traversal of B since x will be visited before y since every node is visited before its children/descendants are visited. x will stand after y in the postorder traversal since the node is visited only after the all the descendants/given by its subtrees are visited.

Now to show converse: That is x stands before y in the preorder traversal of B and x stands after y in the postorder traversal of B. Let us assume that x is not an ancestor of y. There can be two cases here:

(a) y is an ancestor of x: but if it was then that would mean in preorder traversal y would have been visited before x, contradiction

(b) y is not an ancestor of x. Since either of them is not ancestor of the other, this implies neither x nor y is the root. So there is atleast a common ancestor. Let the lowest common ancestor (lca) be a - that is the ancestor you encounter on the paths from x to root and y to root.

 (i) Both x and y are in the same subtree of T of a - since a is the lca either x or y is the root r of a subtree- or else that root would have been the lca. But that means either x is ancestor of y or vice-versa.

 (ii) x is in the left subtree of a and y is in the right subtree of a. But that means in post-order traversal x will appear before y since the left subtrees are exhausted before right subtree.

 (iii) x is in the right subtree of a and y is in left subtree of a then in that case y would be visited before x in preorder traversal and that is a contradiction.

Therefore x is not an ancestor of y leads to a contradiction when we assume that x stands before y in the preorder traversal of B and x stands after y in the postorder traversal of B. So the negation shoudl be true - x is an ancestor of y.

(5) Suppose you are given a weighted undirected graph G and its MST T. Give an algorithm that finds the second MST - that is the algorithm should return the spanning tree of G with smallest total weight except of T.

Proof: The MST and the second MST will differ exactly by an edge. The algorithm finds that edge.

Algorithm:

1. Set $wt(T) = \infty$, $E_{new} = -1$ and $E_{odd} = -1$.
2. For (every edge e that is not in the tree T), do:
 (a) Add e to T (and therefore we end up with a cycle).
 (b) Find k the edge with maximum weight in the cycle but $k \neq e$.
 (c) Remove k from the cycle.
 (d) Let $\delta = wt(e) - wt(k)$.
 (e) If $\delta < wt(T)$ then $wt(T) = \delta$ and $E_{new} = e$ and $E_{old} = k$.
3. The second MST $T' = (T \setminus E_{old}) \cup E_{new}$.