Home work 4

October 20, 2013 Himanshu Shah

1 Problem 2 - pg 189

(a) True.

When we square the values of all edges, they increase proportionately. Hence, the values that are smaller will remain smaller and those that are larger will still remain larger comparitively. In an MST only the edges with minimum weights are present. Hence,the MST will still have the same set of edges and will remain the same.

(b)True

Here also the same argument applies. As we are squaring the value of each of the edges, in the graph, the order of the edges in terms of their weights still remains unaffected. Moreover the graph being directed does not change anything as the direction of the paths still remains the same. Hence the path which that had the minimum cost between points s-t will still be the same.

2 Problem 20 - pg 199

(i) True

Proof: By definition, the minimum spanning tree is a spanning tree with total weight less than or equal to the weight of every other spanning tree. This also means that MST consists of those edges that have the minimum cost for connecting two nodes.Moreover, the weight of the edges in this case is the altitude. Also, the minimum spanning tree will not contain all the edges present in graph G unless the graph itself is a minimum spanning tree. Hence, the minimum spanning tree using edge weights is a minimum connected sub-graph.

Consider the following, let $G(V, E)$ be a graph with edges E and V vertices and Let $M(V', E')$ be a minimum spanning tree of such a graph. Now M consists of those edges that have the smallest weight as per its property. Here, the altitude of the path denotes the edge weight. Hence, M consists of those paths that has the smallest possible altitudes. Moreover, M being an MST will be connecting all the nodes. Hence we can say that an MST with respect to edge weight is also minimum altitude connected subgraph.

(ii) True

Proof: A minimum altitude connected subgraph, is one that connects nodes using the edges that have minimum weight(i.e the paths with minimum altitude). A minimum spanning tree of such a graph comprises of all such edges. Hence for any graph that is a minimum altitude connected subgraph can be built using only such edges present in an MST. Moreover, as we don't have edges with the same weights, the possibility of building a graph using edges that have the same weight but are not present in the MST is done away with.

3 Solve the following Recurrences and give a θ bound for each of them:

3.1 (a) $T(n) = 49T(n/25) + n^{3/2}logn$

Solution: The question is of the form $T(n) = aT(n/b) + f(n)$

where $a = 49$, $b = 25$, and $f(n) = n^{3/2}logn$. So, we will apply masters method and find K in the equation: $af(n/b) = Kf(n)$

$49n/25^{3/2}logn/25 = Kn^{3/2}logn$

$49(n^{3/2}/25^{3/2})log(n/25) = Kn^{3/2}logn$

$49(n^{3/2}/5^3)log(n/25) = Kn^{3/2}logn$

$49/5^3(n^{3/2})logn - log25 = Kn^{3/2}logn$

Hence $K < 1$ So, $T(n) = \theta(f(n))$

$T(n) = \theta(n^{3/2}logn)$

3.2 (b) $T(n) = T(n-1) + 2$

Solution:

$T(n) = T(n-1) + 2$ $T(n-1) = T(n-2) + 2*2$ $T(n-2) = T(n-3) + 2*3$ $T(n-3) = T(n-4) + 2*4$ ..... Hence, $T(n) = T(n-k) + 2*k$

To find the base case for this recurrence, $T(0) = c0 + 2*n$ Hence, we can say that $T(n) \in \theta(n)$.

3.3 (C) $T(n) = 2T(n-1) + 1$

Solution:

$T(n) = 2T(n-1) + 1$

$T(n) = T(n-1) + T(n-1) + 1$

$T(n-1) = T(n-2) + T(n-2) + 2 + 1$

$T(n-1) = T(n-2) + T(n-2) + (2*2) - 1$

$T(n-2) = T(n-3) + T(n-3) + (2*3) - 1$

$T(n-3) = T(n-4) + T(n-4) + (2*4) - 1$

.....

Hence, $T(n) = 2T(n-k) + (2*k) - 1$

Base case will be : $T(0) = c0 + (2*n) - 1$

Hence, we can say that $T(n) \in \theta(n)$.

3.4 (d) $T(n) = T(\sqrt{n}) + 1$

Solution:

Let P be a function defined for all m such that $P(m) = T(2^m)$

So, $P(m) = T(2^m)$

$= T(2^{m/2}) + 1$

$P(m) = P(m/2) + 1$

Hence, the equation is of the form $T(n) = aT(n/b) + f(n)$

where $a = 1$, $b = 2$, and $f(n) = 1$.

So, we will apply masters method and find K in the equation: $af(n/b) = Kf(n)$

$f(n/2) = K$

Hence K=1. So

$T(n) = \theta(f(n)log_b m)$

$T(n) = \theta(1log_2 m)$ $m = logn$ So, $T(n) = \theta(loglogn)$.

4 Give an Algorithm for the following:

4.1 (a)

Solution: We will use the notation V(x) to denote the *absolute* value of the element at position x in the array

Algorithm:

```
Step(i): Sort the entire set of elements in increasing order using quick sort.
Step(ii): Make two pointers i and j which denote the beginning and ending of the arra
Step(iii):

While(i!=j)
{
     if(V(i)>V(j))
       i++;
     else if(V(i)<V(j))
       j++;
     else
      {
         print("V(i) and V(j) have sum zero");
         flag = 1;
         break;
      }
}

//Check of flag is set or not
if(flag == 0)
   print("No such pairs found");
```

The running time of the algorithm = Time taken to sort the array plus the time taken to traverse all inputs.

Hence, $= O(nlogn) + O(n) = O(nlogn)$

```
Choose an arbitary Node as INITIAL_START_NODE.
Run DFS(Initial_Start_Node)
```

```
IF a back edge is detected.
[An already visited node is traversed again].
          Save the node as START_NODE and the node before it as END_NODE.
          Again run DFS(START_NODE) till END_NODE.
          At each step save the edge with maximum weight.
    Remove the edge with the maximum weight
    Again run DFS(Initial_Start_Node);
Else
  Exit normally.
```

5 Problem 26 - pg 202

Solution: In order to solve this question, we will be using graphs.

For each of the edges of the graph G, we will plot some initial points on an X-Y coordinate plane.

We shall do this so as to get an idea of how the edges are behaving at various times with respect to each other.

Now take the equation that overlaps the maximum with other graphs, and find the point at which it becomes minimum. Use the value in the equation of the edge and solve it to obtain t.

This is the time at which the value of the MST will be minimum.

The running time of this algorithm will be the product of the number of edges and the various time points considered.

Hence time of the algorithm will be polynomial to the number of nodes and edges.

Thus, proved.

6 Problem 27 - pg 202

Answer: True

$\mathcal{H}$ uses all the possible spanning tree permutations of graph G. So, if two spanning trees have completely different set of edges, there would always be other spanning trees that would differ by just one edge. Such trees would be neighbours and a collection of all such neighbours would eventually connect the two spanning trees even though they have completely different set of edges.

Proof: Consider a Graph G(V,E) with V vertices and E edges.

The graph G has multiple spanning trees denoted by $T^{'}, T^{''}, T^{'''}, ..., T^{n}$.

Now, $T^{'}(V, E^{'})$ has edges $E^{'}$ and vertices V and $T^{n}(V, E^{o})$ has edges E^{o} and vertices V such that $E^{'} \cap E^{n} = NULL$.

I will prove that even though $T^{'}$ and T^{n} do not share a single edge, they are still connnected in $\mathcal{H}$.

In order to do this we will connect $T^{'}$ to its neighbors $T^{"}$ who has only one edge that is different from $T^{'}$.

Similarly, we will perform the same operation and connect $T^{"}$ to its neighbors $T^{"'}$ who has only one edge that is different from $T^{"}$.

Now as $T^{'}$ is connected to $T^{"}$ and $T^{"}$ is connected to $T^{"'}$, we can say that $T^{'}$ is connected to $T^{"'}$.

If we continue to perform this operation, then we will notice that we will be getting closer and closer in terms of the number of different edges to T^{n}. Eventually there will be a connection from $T^{'}$ to a spanning tree who has just one edge different from T^{n} and in this way, we will get connected to T^{n}.

Hence, we can say that $\mathcal{H}$ will always be connected.

Thus proved.

7 Problem 31 - pg 204

Answer: (a)True

Let a path exists between A and B in H such that the path length is d_{uv}.

Now there exists an edge in G such that the length between A and B is l_{AB1}, such that $l_{AB1} < 3d_{uv}$. Hence as per the algorithm this edge won't be added to the H. So the length of the path will still remain d_{uv}, where as it is l_{AB1} in G.

Lets assume there exists another edge in G such that the length between A and B is l_{AB2}, such that $l_{AB2} > 3d_{uv}$. Hence as per the algorithm this edge will be added to H. So the length of the path will now become l_{AB2}.

Hence if the length of the edge under consideration is smaller than one third of the length of the existing path in H, then it will be added to the graph.

This will always keep the distance for paths less than at most three times the path length in G, because once it increases than thrice, it gets added to H and the cost reduces.

(b) We know that for a graph G with n-vertex and a girth> r the size of the graph is given as $n\lceil n^{2/(r-2)}\rceil$.

Hence for the graph H, the value of r is 4. This is because, in case there is a cycle then the length of the cycle would greater than or equal to $l_e + 3l_e$(4 edges). This is due to the constraint for the graph H.

Hence, the size of the graph lets say $f(n) = n\lceil n^{2/(4-2)}\rceil = n * \lceil n^{2/2}\rceil = n * \lceil n\rceil$

So in equation $\lim_{n\to\infty} f(n)/n^2$.

If we replace $f(n)$ by n^2, then $\lim_{n\to\infty} n\lceil n\rceil/n^2. = 0$.

Hence, Proved.