CS 218, Winter 2024, **Due: 11:59pm, Thu 02/29, 2024**
Assignment #4 [1] **Basic Programming Due: 11:59pm, Fri 02/23, 2024**
Name: **Student ID:**

---

1 (2.5pts) Weight-balanced trees

1. (0.2pts)
   - A node u has w(u) nodes (including the node itself)
   - Its left node has w(lc(u) nodes including the node itself.
   - Same goes for the right node.
   - Now if we add w(lc(u)) and w(rc(u)), we get all the nodes under the root node u. Lets call it n1.
   - Now if I add n1 with the root node, then it becomes n1 + 1. Which is equal to w(u).
2. (0.5pts)
   - We can make a recursive relation with the height.
   - If there are n nodes in height h, then the maximum subtree is $1 - \alpha$ times the original number of nodes.
   - This will give $F(h-1) = (1-\alpha)F(h)$.
   - Solving this recursion (assuming $h >= 2$ we get $O(log(n)$ the height h.
3. (a) (0.3pts)
   (b) (0.4pts)
   (c) (0.7pts)
   (d) (0.4pts)

2 (1 pt) Spanning Trees

1. (0.2pts) A spanning tree for a graph is a subset of edges which form an undirected graph with no cycles such that all nodes are connected (can be more than one).
2. (0.1pts) It should contain $N - 1$ edges where N is the total number of nodes. It is because if we will add even a single edge then a cycle will be formed. If we will subtract even a single edge from $N - 1$ edges then not all nodes will be covered.
3. (0.7pts)
   - Sort the edges in decreasing order which takes nlogn time.
   - Create an empty answer set.
   - Take the first edge and add it into the answer set.
   - Take the next edge, see if it forms a cycle with the edges inside the answer set. If not then only add it.
   - If we get N - 1 edges then we return the answer set, if we do not and the total number of edges run out, then report the graph as disconnected hence no spanning tree possible. (Spanning forest is possible though!!)

[1] Some of the problems are adapted from existing problems from online sources. Thanks to the original authors.

- In order to detect the cycle we make an array of nodes, every time an edge if put inside the set, we mark the 2 nodes on the both side of the edge as visited.
- If a new edge comes, we check if both of the nodes this edge has on its side, are visited are not, if both are visited then its a cycle.
- We store the graph in the matrix form.
- This algorithm will give us n-1 edges with no cycle as that is how it id designed. So we definitely get a spanning tree.
- Since it is making the greedy choice of always choosing the maximum weight edge possible, hence it will give the maximum weight spanning tree.
- Therefore this algorithm is optimal.
- Sorting will take $O(ElogE$

3 Bonus Problems (4pts)

1. Another Dance Show.
2. Melody.
   - My approach I think is algorithmically right however implementation is causing problems.
   - We know the maximum difference between any two consecutive number cannot be greater than $2 * 88$. We store all these numbers as keys from $-2 * 88$ to $2 * 88$
   - If a tempo is repeating from sub array indexes [a, x..b] and [c, y..d] then we know that $c-a == d-b$ and also that $y - c = x - a$.
   - So we have to basically find all subarray having their continous difference, and finding the largest pair (length wise). We map the subarrays with the keys mentioned in point 1.
   - For e.g in the given example of n = 30 and input 25 27 30 34 39 45 52 60 69 79 69 60 52 45 39 34 30 26 22 18 82 78 74 70 66 67 64 60 65 80
   - We have key -4 having 2 subarrays from index 15-19 and index 20-24. Rest all subarrays have the size 2. For example subarray for key 5 has the size 2 from index 27-28.
   - So here the answer will be the length of subarray having the differnce -4 between each of the elements.
   - If we get the largest subarray of a corresponding key, lets say in the first example when n = 15 and input is 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
   - Here the largest subarray will be only one subarray having length 14 and key is 1;
   - Then return its length/2 which is 7 here.
   - The problem comes with case like n=20 and input 1 2 3 4 5 6 7 8 10 1 2 3 4 5 20 6 7 8 9 10
   - here we have 3 subarray with key 1, [1..8] and [10..14], [16..20].
   - Longest subarray is of length 8 but when I half it, it becomes 4 then my actually preference should be the other 2 arrays of length 5.
   - Implementing this will take O(nlogn).
3. Moving Stairs.