COL351 Holi2023: Tutorial Problem Set 1

1. Given a list L of n integers and a positive integer k, our goal is to output the k largest integers in L (in an arbitrary order). Assume $k \leq n$. Design an $O(n \log k)$ time algorithm for doing this. You are advised to write your algorithm in plain English, clearly stating the data structures used.

2. Given a min-heap H of n integers and a positive integer k, our goal is to output the k smallest integers in H in ascending order (without modifying the heap). Assume $k \leq n$. Design an algorithm for doing this. Improve your algorithm to achieve a running time of $O(k \log k)$.

3. Let $n(h)$ denote the minimum number of nodes in an AVL tree of height h. For example, $n(0) = 0$, $n(1) = 1$, $n(2) = 2$, $n(3) = 4$, and so on. Derive a recurrence relation for $n(h)$ and solve it. Hence, prove that the height of an AVL tree with n nodes is $O(\log n)$.

4. Let T be a binary search tree. Suppose you insert a new key k into T in the standard manner, so that k now resides in a leaf. Prove that if T was an AVL tree before insertion, then after insertion, every height-unbalanced node in the resulting tree must lie on the path from the root to the leaf containing k. Further, let ℓ be the lowest unbalanced node (that is, ℓ is a descendant of all unbalanced nodes). Prove that even if we only balance ℓ using rotations, the whole tree becomes balanced.

5. Let T be a binary search tree. Suppose you delete a node x of T having at most one child in the standard manner. Prove that if T was an AVL tree before deletion, then after deletion, the resulting tree can contain at most one unbalanced node. Prove that if such an unbalanced node exists, then it must be an ancestor of x in the original tree.

6. Prove that the number of connected components of any graph $G = (V, E)$ is at least $|V| - |E|$.

7. Prove that a graph is a tree if and only if for any two vertices u and v, the graph contains a unique path from u to v.

8. **[Cut property of trees]** Let $G = (V, E)$ be a tree and $e \in E$ be an arbitrary edge. Prove that $G' = (V, E \setminus \{e\})$ has exactly 2 connected components. Further, prove that if u and v are vertices in different connected components of G', then the graph $G'' = (V, (E \setminus \{e\}) \cup \{\{u, v\}\})$ is also a tree.

9. Let $G = (V, E)$ be any connected graph which satisfies the above cut property, that is, for every $e \in E$, the graph $G' = (V, E \setminus \{e\})$ has exactly 2 connected components. Prove that G is a tree.

10. **[Cycle property of trees]** Let $G = (V, E)$ be a tree and u, v be vertices such that $\{u, v\} \notin E$. Prove that $G' = (V, E \cup \{\{u, v\}\})$ has exactly 1 cycle (modulo the choice of the initial vertex and the direction of traversal). Further, prove that if e is any edge in this cycle, then the graph $G'' = (V, (E \cup \{\{u, v\}\}) \setminus \{e\})$ is also a tree.

11. Let $G = (V, E)$ be any acyclic graph which satisfies the above cycle property, that is, for every two vertices u, v such that $\{u, v\} \notin E$, the graph $G' = (V, E \cup \{u, v\})$ has exactly 1 cycle (modulo the choice of the initial vertex and the direction of traversal). Prove that G is a tree.

12. A *spanning tree* of a graph G is a tree whose vertex set is the vertex set of G, and which is a subgraph of G. Prove that a graph is connected if and only if it has a spanning tree.

13. The distance between two vertices of a connected undirected graph to be the length of a shortest path between those vertices (for example, the distance between a vertex and itself is 0, the distance between adjacent vertices is 1, and so on). The *diameter* of a graph is defined as the maximum, over all pairs u, v of its vertices, of the distance between u and v. Let G be a connected undirected graph. Perform a breadth-first-traversal of G starting from some vertex s, and let T be the resulting breadth-first-traversal-tree. Suppose T has exactly $h+1$ levels $L_0 \ldots, L_h$, where $L_0 = \{s\}$. Prove that the diameter of G is at least h and at most $2h$.

14. Let G be a connected undirected graph on n vertices such that the diameter of G is more than $n/2$. Prove that there exists a vertex v in G such that deletion of v (and all edges incident on v) results in a disconnected graph.

15. Recall the out-adjacency list representation of directed graphs, where we have one linked list for every vertex v, where the linked list contains the out-neighbors of v. Given a directed acyclic graph G in its out-adjacency list representation and a vertex t of G, we would like to compute, for every vertex v of G, the number of directed paths from v to t. Note that the number such paths could be exponential in the number of vertices, so a brute-force counting is too inefficient.

    Complete the following algorithm to compute an array `pathCount` indexed by the vertex set of G so that at the end of the run of the algorithm, `pathCount[v]` equals the number of directed paths from v to t in G. Each blank must be filled up with an expression that can be computed in $O(1)$ time. `findPathCounts(G,t):`

    (a) Compute a topological sort of `G`.

    (b) For each vertex `v` in the reverse of the topological sort order:

        2.1. If `v=t` then `pathCount[v]` $\leftarrow$ _____, else `pathCount[v]` $\leftarrow$ _____.

        2.2. For each out-neighbor `u` of `v`: `pathCount[v]` $\leftarrow$ _____.

    (c) Return `pathCount`.

    Prove that the above algorithm runs in time $O(n+m)$, where n is the number of vertices and m is the number of edges in G. Why is it necessary to iterate over the vertex set in reverse topological sort order in the algorithm of part (a)? Suppose that for every vertex v, instead of counting the number of paths from v to t, you wanted to find the length of the longest path from v to t. How will you modify the above algorithm to achieve the same running time?