CS 161 Winter 2020 Section 4

January 30-31, 2020

1 More Sorting!

We are given an unsorted array A with n numbers between 1 and M where M is a large but constant positive integer. We want to find if there exist two elements of the array that are within T of one another.

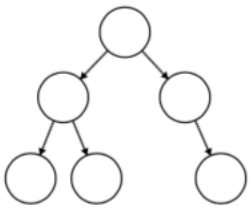
(a) Design a simple algorithm that solves this in $O(n^2)$.

(b) Design a simple algorithm that solves this in $O(n \log n)$.

(c) How could you solve this in $O(n)$? (Hint: modify bucket sort.)

2 Uniqueness of BST Structure

You are given a binary tree structure with n nodes and a set of n distinct keys (numbers). Prove or disprove: There is exactly one way to assign keys to the given tree structure such that the resulting tree is a valid binary search tree. **Example**: You are given the binary tree drawn below and the set of keys 1, 2, 3, 4, 5, 6. The question asks whether there is exactly one way to assign the keys to nodes such that the tree will be a binary search tree. (If you prove the statement, it should be for any input and not just this example.)



3 Approximate Sorting

(Note: This is a more difficult question from a previous offering's homework)

We say an array A of n distinct numbers is **k-approximately sorted** if the following property holds: for each $i = 1, 2, ..., n$, if $A[i]$ is the j-th smallest number in the sequence, then $|i - j| \leq k$. In other words, each number is at most k positions away from its actual sorted position. A k-approximate sorting algorithm takes a sequence of distinct numbers as input and produces a sequence that is k-approximately sorted.

(a) Show that given any n distinct numbers, the number of permutations of those numbers that are $\sqrt{n}$-approximately sorted is $O(n^{cn})$, for some $c < 1$. In other words, show that there are $O(n^{cn})$ possible permutations of the numbers such that each element lands within $\sqrt{n}$ distance of its sorted positions.

(b) Use part (a) to find a lower bound on the number of leaf nodes in the decision tree for any comparison-based $\sqrt{n}$-approximate sorting algorithm, and prove that any such $\sqrt{n}$-approximate sorting algorithm must have worst case complexity $\Omega(n \log n)$.

(c) Let $k > 1$ be an arbitrary constant. Can we construct an $\frac{n}{k}$-approximate sorting algorithm that does better than $O(nlogn)$? (Hint: the SELECT algorithm might help you here).