CS 161 Winter 2021 Section 4

February 2021

1 More Sorting!

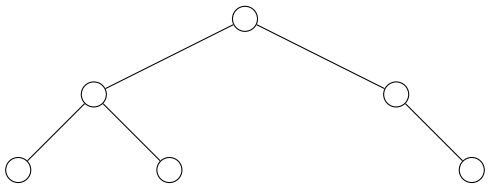
We are given an unsorted array A with n numbers between 1 and M where M is a large but constant positive integer. We want to find if there exist two elements of the array that are within T of one another.

1. Design a simple algorithm that solves this in $O(n^2)$.
2. Design a simple algorithm that solves this in $O(n \log n)$.
3. 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 Randomly Built BSTs

In this problem, we prove that the average depth of a node in a randomly built binary search tree with n nodes is $O(\log n)$. A *randomly built binary search tree* with n nodes is one that arises from inserting the n keys in random order into an initially empty tree, where each of the $n!$ permutations of the input keys is equally likely.

Let $d(x, T)$ be the depth of node x in a binary tree T (the depth of the root is 0). Then, the average depth of a node in a binary tree T with n nodes is

$$\frac{1}{n} \sum_{x \in T} d(x, T).$$

a. Let the *total path length* $P(T)$ of a binary tree T be defined as the sum of the depths of all nodes in T, so the average depth of a node in T with n nodes is equal to $\frac{1}{n}P(T)$. Show that $P(T) = P(T_L) + P(T_R) + n - 1$, where T_L and T_R are the left and right subtrees of T, respectively.

b. Let $P(n)$ be the expected total path length of a randomly built binary search tree with n nodes. Show that $P(n) = \frac{1}{n}\sum_{i=0}^{n-1}(P(i) + P(n-i-1) + n - 1)$.

c. Show that $P(n) = O(n \log n)$. You may cite a result previously proven in the context of other topics covered in class.

d. Design a sorting algorithm based on randomly building a binary search tree. Show that its (expected) running time is $O(n \log n)$. Assume that a random permutation of n keys can be generated in time $O(n)$.