CS 161 (Stanford, Winter 2024) Section 4

1 Warm-up: Binary Search Trees vs Heaps

For each of the following, choose the corresponding data structure.

1. With this data structure you can efficiently find the element with key value 2020.

   (A) Red-black binary search trees (B) Heaps (C) Both (D) Neither

2. With this data structure you can efficiently find the smallest element.

   (A) Red-black binary search trees (B) Heaps (C) Both (D) Neither

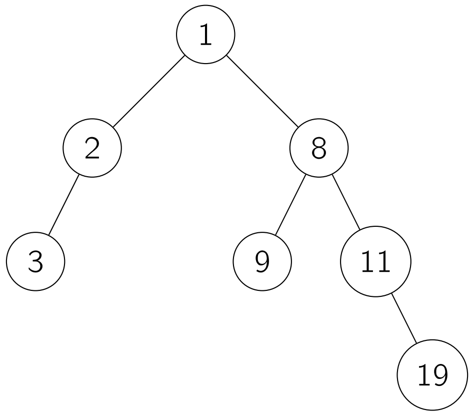
3. With this data structure you can efficiently find the median element.

   (A) Red-black binary search trees (B) Heaps (C) Both (D) Neither

4. This data structure is fast on average, but will be slow in the worst-case.

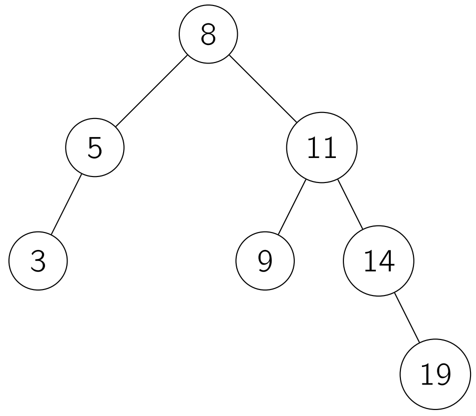
   (A) Red-black binary search trees (B) Heaps (C) Both (D) Neither

   For each of the following, choose the corresponding data structure.

5. 

   (A) Red-black binary search trees (B) Heaps (C) Both (D) Neither

6.



(A) Red-black binary search trees (B) Heaps (C) Both (D) Neither

2 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

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

1. 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.
2. Let $P(n)$ be the expected total path length of a randomly built binary search tree with n nodes. Show that $P(n) = \frac{1}{n}\sum_{i=0}^{n-1}(P(i) + P(n-i-1) + n - 1)$.
3. Show that $P(n) = O(n \log n)$. You may cite a result previously proven in the context of other topics covered in class.
4. 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 $O(n)$.

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

4 Grade inflation

Prof. Toon Ice used to grade on a $1, \ldots, 10$ basis. But to keep up with grade inflation, Prof. Ice decided to automatically update the grading rubric every year to output grades in the range $2^y + 1, \ldots, 2^y + 10$ (where y is the year). At the end of the quarter, Prof. Ice needs to report the sorted class grades to the registrar's office as soon as possible. Can you propose an appropriate algorithm?

Assumption: all the grades are integers.