2021-2022, 1st Semester

Problem Set 2
Due: 11-09, 20:00

Please answer each of the following problems.
Note: For all problems, if you include pseudocode in your solution, please also include a brief description of what the pseudocode does.

1. Suppose that p is an unknown value, $0 < p < 1$. Suppose that you can call a function `randP` which returns `true` with probability p and returns `false` with probability $1 - p$. Every call to `randP` is independent. You have no way to generate random numbers except through `randP`. (20 points)

   (a) (8 pts) Describe an algorithm—using `randP`—that returns `true` with probability 1/2 and `false` with probability 1/2. Your algorithm should in expectation, use $\frac{1}{p(1-p)}$ calls to `randP` (in other words, run $\frac{1}{p(1-p)}$ times of function `randP` in expectation). Your algorithm **does not** have access to the value of p, and **does not** have access to any source of randomness other than calls to `randP`. [**We are expecting pseudocode, and a short description of what the algorithm does.**]

   Hints: (i) Your algorithm does not have to compute p, or an approximation to it. (ii) Notice that in the worst case, your algorithm may use more calls to `randP`, possibly even infinitely many.

   (b) (6 pts) Formally prove that your algorithm runs using expected $\frac{1}{p(1-p)}$ calls to `randP`. [**We are expecting a mathematical calculation of the expected value of the total number of calls to `randP`.**]

   (c) (6 pts) Informally argue that your algorithm returns `true` with probability 1/2 and `false` with probability 1/2. [**We are expecting an informal justification of why the algorithm returns `true` with probability** 1/2 **and `false` with probability** 1/2**. Your argument should convince the reader that you are correct, but does not have to be a formal proof.**]

2. Suppose that A and B are sorted arrays of length n in ascending order, and that all numbers in the arrays are distinct. (20 points)

   (a) (8 pts) Design an algorithm to find the median of all $2n$ numbers in $O(\log n)$ time. For our purposes, we define the median of the $2n$ numbers as the nth smallest number in the $2n$ values. [**We are expecting: pseudocode, and a description of the algorithm.**]

   (b) (6 pts) Informally argue that your algorithm correctly finds the median of all $2n$ numbers. [**We are expecting a short (paragraph or two) argument that will convince the reader why your algorithm works correctly.**]

   (c) (6 pts) Prove that your algorithm runs in time $O(\log(n))$ time. [**We are expecting a formal proof.**]

3. **Sorting in Linear Time** (20 points) In lecture, we discussed how we can sort an array of n integers in the range $[0, k)$ in $O(n)$ time. In this question, we'll show that we can actually expand the range of our values without incurring too much cost.

   (a) (12 points) Given an array A of n pairs of integers (a_1, b_1),(a_2, b_2), ..., (a_n, b_n), where $a_i, b_i \in [0, n)$ for each i. We say two pairs (a_i, b_i), (a_j, b_j) are in lexicographic order if and only if one of the following holds:

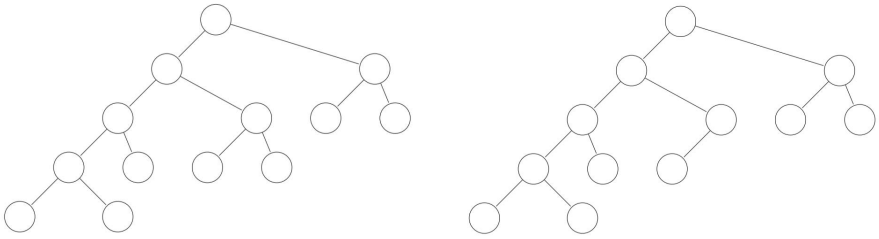
   - $a_i < a_j$
   - $a_i = a_j$ and $b_i \leq b_j$

   (e.g. the ordered pairs {(1, 2), (1, 2)}, {(1, 2), (1, 3)}, and {(1, 2), (2, 1)} are in lexicographic order, while the ordered pairs {(1, 2), (1, 1)} and {(2, 1), (1, 2)} are not.) Give an algorithm that sorts A according to lexicographic order in $O(n)$ time. **Formally prove** the correctness and runtime of your algorithm.

   (b) (8 points) **Informally explain** how part (a) can be used to sort n integers in the range $[0, n^2)$ in $O(n)$ time.

4. Binary Search Trees (20 points) This problem requires you to draw trees. To cut down on the amount of drawing you have to do, you will only be required to show the final tree for full credit. This means you will only have to draw (at most) five trees total: one for each part of (a) that you answer yes to, and one for each part of (b). If you want you may draw your trees by hand and scan them into your PDF.

   (a) (10 points) For each of the following unlabeled binary trees in Figure 1, state whether or not it can be the structure of a red-black tree. If yes, label the nodes red or black. If no, state which of the red-black tree's four properties cannot be satisfied and why.



(a) Tree 1 (5 points) (b) Tree 2 (5 points)

Figure 1: Red-Black Trees?

   (b) (10 points) Perform the following operations on this binary search tree, and draw the final result for each part. Each part of this problem is independent of each other – the operations in part 2 must be performed on the original tree, and not on the output of part 1.

      (i) (4 points) Use the Insert procedure in the lecture notes to insert 14 into the tree in Figure 2.

      (ii) (3 points) Use the Delete procedure in the lecture notes to delete 6 from this tree.

      (iii) (3 points) Use the Delete procedure in the lecture notes to delete 15 from the tree.

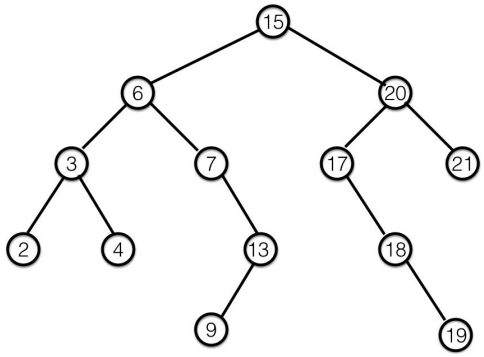


Figure 2: Red-Black Trees?

5. (10 points) Let T be a Red-Black tree with root x as shown in Figure 3. Let T_L be the subtree rooted at x's left child, and let T_R be the subtree rooted at x's right child. Decide if each of the following statements are true or false. If it is true, give a proof. If it is false, give a counter-example.

   (a) (5 points) In the set-up in Figure 3, we must have

$$|T_L| \geq \frac{|T|}{2} - 1 \qquad \text{and} \qquad |T_R| \geq \frac{|T|}{2} - 1$$

   where $|T|$ denotes the number of nodes in T (including the root, not including NIL nodes).

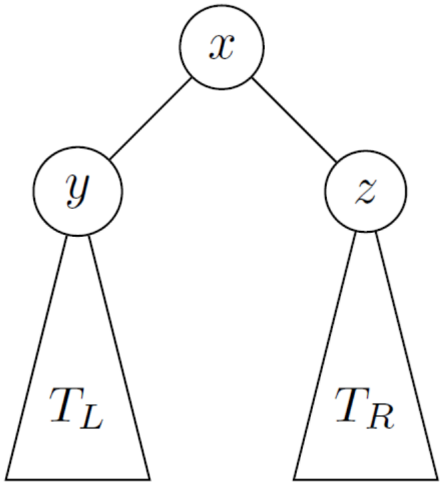


Figure 3: Height of Red-Black Tree

(b) (5 points) In the set-up in Figure 3, we must have

$$|T_L| \geq \frac{\sqrt{|T|}}{2} - 1 \qquad \text{and} \qquad |T_R| \geq \frac{\sqrt{|T|}}{2} - 1$$

where $|T|$ denotes the number of nodes in T (including the root, not including NIL nodes).

[We are expecting: For each, either a rigorous proof, or an explicit counter-example.]

[HINT: You may use a claim that we proved in class.]

6. (10 points) Suppose we use a hash function h to hash n distinct keys into an array T of length m. Assuming simple uniform hashing, what is the expected number of keys that does not collide with any other keys? More precisely, what is the expected cardinality of $\{l\colon \forall k \neq l \text{ and } h(k) \neq h(l)\}$? What is the maximum cardinality of the set? **[We are expecting: a detailed proof.]**