CS240E Tutorial, Feb 28, 2025

1. Which asymptotic relationships hold between $g(n) = n \log n$ and the recursively defined function $f(n)$ with $f(1) = 0$ and $f(n) = n + \frac{2}{n} \sum_{i=1}^{n-1} f(i)$?

2. Let T be a binary heap of size $n = 2^k$ for some integer k. Assume that T is stored as a tree. Explain how to build a binomial heap that stores the same items in $O(\log n)$ time.

3. Consider the pseudocode on the right. What is the expected run-time?

Algorithm 1: *mystery*(int n)

if $n \leq 1$ **then return**
if *random*$(2) = 0$ **then** *mystery*$(\lfloor \frac{n}{2} \rfloor)$
else *mystery*$(n-1)$

4. Let T be an AVL-tree. Assume that you have just now inserted a new key k into T, and let z be the node that stores k. Show that z has no grandchildren.

5. In the array below, *interpolation-search*(70) finds the item at the first probe that uses the formula. What is the smallest possible integer at XX?

0	1	2	3	4	5	6	7	8	9	10
XX	57	64	70	72	77	81	92	93	99	100

6. Show that for *any* deterministic meldable heap H with n nodes, the operation *insert*(k) has **worst-case** run-time $O(\log n)$ if we use the child with the not-larger size for breaking ties during merge.

7. You are given n rectangles $R_0, \ldots, R_{n-1}$ that lie on an $n \times n$-grid, i.e., the coordinates of all four corners of each rectangle are integers in $\{1, \ldots, n\}$.

   Describe an algorithm that sorts the rectangles by increasing area in $O(n)$ time.

8. Why is a binary search tree (without any rebalancing) not oblivious?

9. We have seen *many* variants of binary search trees in class, each with some advantages and disadvantages with respect to run-time and/or number of rotations. Create one more variant that satisfies the following constraints:

   - Key-value pairs are stored in a binary search tree. Nodes may store additional information.
   - The space is $O(n)$ at all times, where n is the number of currently stored key-value pairs.
   - *search* has $O(\log n)$ worst-case run-time.
   - *insert* has $O(\log n)$ worst-case and uses at most 1 (single or double) rotation.
   - *delete* has $O(\log n)$ amortized time and uses no rotations.