CS 317, Assignment 3

Due on 22 October in class

This assignment is all about divide and conquer algorithms, even if this is not explicitly stated in the questions below.

1. You are given a sorted sequence of integers $A_{1\ldots n}$ that has been circularly shifted k positions to the right. For example, $\langle 35, 42, 5, 15, 27, 29\rangle$ is a sorted sequence that has been circularly shifted $k = 2$ positions, while $\langle 27, 29, 35, 42, 5, 15\rangle$ has been shifted $k = 4$ positions.

   Design an algorithm that finds the index of the largest element in A in $O(\log n)$ time. Analyze the running time of your algorithm and provide a proof of correctness.

2. We want to sort a stack of n pancakes by size so that the largest pancake is at the bottom and the smallest at the top. For convenience we index the stack from 1 to n, with index 1 at the top and n at the bottom. We are allowed only two operations: FindLargest(n) returns the index of the largest pancake in an n-pancake stack, and Flip(k) flips the stack of pancakes $1 \ldots k$ so that the k-th pancake becomes the first, the $k-1$-st pancake becomes the second, and so on (just like inserting a spatula immediately below index k and using it to flip the stack of pancakes above it).

   (a) Design a linear-time algorithm to sort an arbitrary stack of n pancakes. Analyze the running time of your algorithm and provide a proof of correctness.

   (b) Discuss the optimality of your algorithm. (Hint: describe a stack of n pancakes that requires $\Omega(n)$ flips to sort.)

   (c) Suppose now that one side of each pancake is burned. Describe a linear-time algorithm that sorts an arbitrary stack of n pancakes so that the burned side of each pancake faces down. Analyze the running time of your algorithm and provide a proof of correctness.

3. Let T be a binary tree with n vertices. Deleting any vertex v splits T into at most three subtrees, one rooted at the left child of v (if any), the second rooted at the right child of v (if any), and the third containing the parent of v (if any). We call v a central vertex if each of these smaller trees has at most $n/2$ vertices.

   (a) Show that every binary tree had a central vertex.

   (b) Describe an algorithm to find a central vertex in an arbitrary given binary tree. Analyze the running time of your algorithm and provide a proof of correctness.

Make sure you review the submission guidelines posted on the course's Web site before submitting. Note in particular (last warning) that the only acceptable ways to describe an algorithm are pseudo-code (preferred) or actual code. Textual descriptions in particular are not acceptable.