ECE 590 Homework 7 Fall 2024

Due: Wednesday, December 4st

1. **The Index Problem**

   Let A be an array of n distinct integers where A is **already sorted** in ascending order. Our problem is to find an index i, $1 \leq i \leq n$, such that $A[i] = i$ or determine that no such i exists.

   Describe an algorithm for this problem with $O(\log n)$ worst case running time. You should give the algorithm (in clear English or in clear high-level pseudo-code) and briefly explain why the running time is $O(\log n)$ in the worst case.

2. **Party Planning!??**

   You've accepted a job as a senior algorithm designer at the company FacePage. One day, your boss Clark Hackerberg comes to you with the following problem:

   'I'm throwing a company party! As you know, FacePage has a hierarchical structure. You can think of it as a tree. The president, that's me, is at the root of the tree. Below the root are supervisors, below them are managers, below them are team leaders, etc., etc., until you get down to the leaves - the summer interns. The tree is not necessarily binary: some non-leaf nodes may have one "child", others two, and others even more. To make the party fun, I thought it would be best that we don't invite an employee along with their immediate boss (their parent in the tree). So how can I choose which employees to invite to guarantee the largest possible party?'

   In other words, your task is to take as input a tree representing the company hierarchy and compute the largest number of employees (nodes) that can be selected such that no two adjacent nodes (i.e., a node and its child) are chosen.

   (a) [2pts] Describe a *greedy algorithm* for this problem (in either pseudo-code or clear English). A greedy algorithm, in this case, is one that visits vertices one-by-one (in some order of your choosing) and makes binding decisions on whether or not to invite that vertex before moving on to the next vertex.

   (b) [2pts] What is the worst-case asymptotic runtime of your algorithm?

   (c) [2pts] Prove the correctness of your algorithm using induction.