Math 590 Homework 7 Fall 2018

Due: **Thursday**, November 29

1. **Greedy Ferries?**[1]

   Ranopolis is a sprawling city divided in half by the great Muddy River. The city has a number of ferries that shuttle cars across every day. Each ferry had three lanes for cars, and these lanes are of varying lengths (even within one ferry, the three lanes may not necessarily have the same length). When cars line up single-file at a ferry crossing, their lengths are measured automatically. Each car is then directed into one of the three lanes of the ferry. The cars can be parked bumper-to-bumper, with no space between them. *The objective is to park as many cars as possible on a ferry without overfilling any of the three lanes.* Notice that the cars must be processed in the order in which they arrived at the ferry crossing point and we can't skip one car in order to pack a later car (that violates fairness). **All lane lengths and car lengths are positive integers.** Let n denote the number of cars and let ℓ_1, ℓ_2, and ℓ_3 denote the lengths of the three lanes.

   (a) [2pts] Prof. I Lai was hired by the Ranopolis Ferry Transit Authority to design an algorithm for assigning cars to the ferry lanes. He proposed the following greedy algorithm:

   'Park each car in the lane that has the most remaining space, and break any ties arbitrarily.'

   Show a small counterexample where this greedy algorithm does NOT pack the most cars. You will need to give a sequence of car lengths, the three lane lengths, and then show what Prof. I Lai's algorithm would do and show that there exists a better solution for that same input.

   (b) [2pts] It turns out that this problem can be efficiently solved using an algorithmic approach we have already discussed in class. Briefly discuss what approach you would use, and how it might be implemented efficiently (you do not need to fully describe the algorithm, but instead indicate how you could approach this problem).

[1] Adapted from problem sets created by Harvey Mudd College CS Professor Ran Libeskind-Hadas.

2. **Rural Cell Phone Towers**

The cell phone service provider Vorizan has decided to invest in cell towers in rural areas. Here, a rural area can be treated as a long straight road with houses along it. Vorizan's objective is to deploy the least number of cell towers along the road so that every house will be within 3000 or fewer meters of a cell tower.

Your task is to develop a greedy algorithm that takes as input an array of n points on a line, representing the locations of n houses. Each point is the distance in meters from the left starting point of the road (the origin) and those points are in sorted order (increasing distance from the starting point). The algorithm must return a sorted array of positions at which to place the cell towers. That array must use the least number of cell towers possible. (Aside: this is a 1D variation of the set cover problem.)

(a) [2pts] Describe the simplest greedy algorithm that you can for this problem.

(b) [2pts] What is the worst-case asymptotic runtime of your algorithm? Give the best bound that you can and explain in a sentence or two.

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

3. **Greedy Party Planning!**[1]

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

'I'm throwing a company party! As you know, Millisoft 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 strong mathematical induction.

4. **Two Stacks can make a Queue**

A well implemented stack has two operations that it can support in $O(1)$ time: `Push` and `Pop`. It turns out that it is possible to implement an efficient queue using only two stacks. A queue has two supported operations:

- `Enqueue`: where a value is pushed into the rear of the queue.
- `Dequeue`: where a value is popped from the front of the queue.

(a) [2pts] Describe how a queue can be implemented using two stacks. (Hint: a more standard queue implementation typically tracks **two** values: front and rear. You have **two** stacks to use.)

(b) [2pts] Using the accounting method from class, prove that your stack-based queue has the property that any sequence of n operations (selected from `Enqueue` and `Dequeue`) takes a total of $O(n)$ time resulting in amortized $O(1)$ time for each of these operations!