Math/CCS 103 Professor: Padraic Bartlett

Homework 1: Algorithms

Due Friday, week 1 *UCSB 2014*

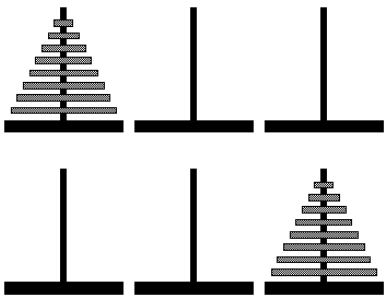
Homework Problems.

Pick **two** of the following **three** problems to solve!

1. The Towers of Hanoi is the following puzzle: Start with 3 rods. On one rod, place n disks with radii $1, 2, \dots n$, so that the disk with radius n is on the bottom, the disk with radius $n-1$ is on top of that disk, and so on/so forth.

   The goal of this puzzle is to move all of the disks from one rod to another rod, obeying the following rules:

   - You can move only one disk at a time.
   - Each move consists of taking the top disk off of some rod and placing it on another rod.
   - You cannot place a disk A on top of any disk B with radius smaller than A.

   Find a recursive algorithm for solving this puzzle! How long does it take to complete your solution? Suppose that you can perform a move once every second, and you can perform moves until the heat death of the universe (10^{100} years, say.) What is the largest puzzle you can solve?

2. Suppose you have a stack of n pancakes of different sizes. You want to sort these pancakes so that smaller pancakes are on top of larger pancakes!

   However, the only tool you have to do this is a spatula. The spatula can **flip** pancakes, as described here:

   - Suppose we have a stack of pancakes. Write this stack as an ordered list $(p_1, \dots p_n)$, where the first element in our list is at the bottom, the second is directly on top of the first, and so on/so forth.

- We can insert our spatula at any point in the stack. From there, we can **flip** all of the entries above where we put our spatula. For example, suppose we have the stack $(p_1, p_2, p_3, p_4, p_5, p_6)$. We could insert our spatula between pancakes p_3 and p_4, and flip the stack (p_4, p_5, p_6) to get the new arrangement

$$(p_1, p_2, p_3, p_6, p_5, p_4).$$

(Fun fact: the one and only research paper written by Bill Gates studied this problem.) Describe an algorithm to sort an arbitrary stack of n pancakes, using as few flips as possible. How many flips does your algorithm need, in the worst-case scenario?

3. Consider the following algorithm designed to sort a list of numbers:

Algorithm. Take as input some list $L = (l_1, \ldots l_n)$ of objects, along with some operation $\leq$ that can compare any two of these elements. Perform the following algorithm:

(a) Create a integer variable *loc* and a boolean variable *didSwap*.

(b) Set the variable *loc* to 1, and *didSwap* to $false$.

(c) While the variable *loc* is $< n$, perform the following steps:

i. If $l_{loc} > l_{loc+1}$, swap the two elements l_{loc}, l_{loc+1} in our list and set *didSwap* to true.

ii. Regardless of whether you swapped elements or not, add 1 to the variable *loc*, and go to 3.

(d) If *didSwap* is *true*, then go to 2.

(e) Otherwise, if *didSwap* is $false$, we went through our entire list and never found any pair of consecutive elements such that the second was larger than the first. Therefore, our list is sorted! Output our list.

What is the complexity (i.e. runtime) of this algorithm? (Assume that a list of n numbers is considered to have input size n.)