Module 3: Basic Written Problems

1. You are given a set of points $P = \{p_1, p_2, ..., p_n\}$ on the real line (you may assume these are given to you in sorted order). Describe an algorithm that determines the smallest set of unit-length closed intervals that contains all of the given points. For example, the points $\{0.9, 1.2, 1.3, 2.1, 3.0\}$ can be covered by $[0.7, 1.7]$ and $[2.0, 3.0]$. State the runtime of your algorithm.

2. Prove that your algorithm always returns the optimal result. Use a proof by induction.

3. It's your sister's birthday and you decide to pull a big prank on her. You plan to wrap several decreasingly sized presents inside of each other so that she receives a large present, but many boxes later, realizes the gift is actually quite small in size. You silly prankster you!! You have a bunch of boxes and want to figure out how many boxes deep the prank can possibly go!

   Develop an algorithm that accepts as input the set of boxes $B = \{b_1, b_2, ..., b_n\}$ and returns the maximum number of boxes that can be placed inside of one another. Each b_i contains fields for that boxes width, height, and depth respectively. Remember that boxes can be rotated in any way. Your algorithm should return the actual set of boxes that should be used for the prank. State the runtime of your algorithm.

4. Let's suppose we want to replicate a single file f over a collection of n servers $S_1, S_2, ..., S_n$. Each server has a cost of storing the file. Let c_i be the cost of storing f on server i. Let's also stipulate that $c_n = 0$ (to guarantee that the file can be stored on at least one server for free).

   Your problem is to figure what subset of the n servers you should store the file on in order to minimize cost (which we define in more detail now). Suppose the file is stored on a subset of server indices $L \subseteq [1, n]$ and that $n \in L$ (L cannot be empty). More formally, file f is on server i iff $i \in L$. When a user tries to access the file they start from some server S_j, and they attempt to access S_j, then S_{j+1} etc until they find the file (they will always find f on S_n in the worst case). We say this costs the user:

   $$u_j = \left(\min_{k \geq j : k \in L} k \right) - j$$

   For any particular set $L \subseteq [1, n]$, the total cost is thus:

   $$cost(L) = \sum_{i \in L} c_i + \sum_{j \in [1,n]} u_j$$

   First, provide a brute-force algorithm that solves this problem. What is the run-time and why? Then, provide a better algorithm that solves this problem.