Algorithms: CSE 202 — Homework 2

Problem 1: Nesting Boxes (CLRS)

A d-dimensional box with dimensions $(x_1, x_2, \ldots, x_d)$ *nests* within another box with dimensions $(y_1, y_2, \ldots, y_d)$ if there exists a permutation π on $\{1, 2, \ldots, d\}$ such that $x_{\pi(1)} < y_1, x_{\pi(2)} < y_2, \ldots, x_{\pi(d)} < y_d$.

1. Argue that the nesting relation is transitive.

2. Describe an efficient method to determine whether or not one d-dimensional box nests inside another.

3. Suppose that you are given a set of n d-dimensional boxes $\{B_1, B_2, \ldots, B_n\}$. Describe an efficient algorithm to determine the longest sequence $\langle B_{i_1}, B_{i_2}, \ldots, B_{i_k} \rangle$ of boxes such that B_{i_j} nests within $B_{i_{j+1}}$ for $j = 1, 2, \ldots, k-1$. Express the running time of your algorithm in terms of n and d.

Problem 2: Classes and rooms

You are given a list of classes C and a list of classrooms R. Each class c has a positive enrollment $E(c)$ and each room r has a positive integer capacity $S(r)$. You want to assign each class a room in a way that minimizes the total sizes (capacities) of rooms used. However, the capacity of the room assigned to a class must be at least the enrollment of the class. You cannot assign two classes to the same room. Design an efficient algorithm for assigning classes to rooms and prove the correctness of your algorithm.

Problem 3: Business plan

Consider the following problem. You are designing the business plan for a start-up company. You have identified n possible projects for your company, and for, $1 \leq i \leq n$, let $c_i > 0$ be the minimum capital required to start the project i and $p_i > 0$ be the profit after the project is completed. You also know your initial capital $C_0 > 0$. You want to perform at most k, $1 \leq k \leq n$, projects before the IPO and want to maximize your total capital at the IPO. Your company cannot perform the same project twice.

In other words, you want to pick a list of up to k distinct projects, $i_1, \ldots, i_{k'}$ with $k' \leq k$. Your *accumulated capital* after completing the project i_j will be $C_j = C_0 + \sum_{h=1}^{h=j} p_{i_h}$. The sequence must satisfy the constraint that you have sufficient capital to start the project i_{j+1} after completing the first j projects, i.e., $C_j \geq c_{i_{j+1}}$ for each $j = 0, \ldots, k'-1$. You want to maximize the final amount of capital, $C_{k'}$.

Problem 4: Shortest wireless path sequence (KT 6.14)

A large collection of mobile wireless devices can naturally form a network in which the devices are the nodes, and two devices x and y are connected by an edge if they are able to directly communicate with each other (e.g., by a short-range radio link). Such a network of wireless devices is a highly dynamic object, in which edges can appear and disappear over time as the devices move around. For instance, an edge (x, y) might disappear as x and y move far apart from each other and lose the ability to communicate directly.

In a network that changes over time, it is natural to look for efficient ways of *maintaining* a path between certain designated nodes. There are two opposing concerns in maintaining such a path: we want paths that are short, but we also do not want to have to change the path frequently as the network structure changes. (That is, we'd like a single path to continue working, if possible, even as the network gains and loses edges.) Here is a way we might model this problem.

Suppose we have a set of mobile nodes V, and at a particular point in time there is a set E_0 of edges among these nodes. As the nodes move, the set of edges changes from E_0 to E_1, then to E_2, then to E_3,

and so on, to an edge set E_b. For $i = 0, 1, 2, \ldots, b$, let G_i denote the graph (V, E_i). So if we were to watch the structure of the network on the nodes V as a "time lapse", it would look precisely like the sequence of graphs $G_0, G_1, G_2, \ldots, G_{b-1}, G_b$. We will assume that each of these graphs G_i is connected.

Now consider two particular nodes $s, t \in V$. For an s-t path P in one of the graphs G_i, we define the *length* of P to be simply the number of edges in P, and we denote this $\ell(P)$. Our goal is to produce a sequence of paths $P_0, P_1, \ldots, P_b$ so that for each i, P_i is an s-t path in G_i. We want the paths to be relatively short. We also do not want there to be too many *changes*–points at which the identity of the path switches. Formally, we define $changes(P_0, P_1, \ldots, P_b)$ to be the number of indices i ($0 \le i \le b-1$) for which $P_i \ne P_{i+1}$.

Fix a constant $K > 0$. We define the cost of the sequence of paths $P_0, P_1, \ldots, P_b$ to be

$$cost(P_0, P_1, \ldots, P_b) = \sum_{i=0}^{b} \ell(P_i) + K \cdot changes(P_0, P_1, \ldots, P_b).$$

1. Suppose it is possible to choose a single path P that is an s-t path in each of the graphs $G_0, G_1, \ldots, G_b$. Give a polynomial-time algorithm to find the shortest such path.

2. Give a polynomial-time algorithm to find a sequence of paths $P_0, P_1, \ldots, P_b$ of minimum cost, where P_i is an s-t path in G_i for $i = 0, 1, \ldots, b$.

Problem 5: Untangling signal superposition (KT 6.19)

You're consulting for a group of people (who would prefer not to be mentioned here by name) whose jobs consist of monitoring and analyzing electronic signals coming from ships in coastal Atlantic waters. They want a fast algorithm for a basic primitive that arises frequently: "untangling" a superposition of two known signals. Specifically, they're picturing a situation in which each of two ships is emitting a short sequence of 0s and 1s over and over, and they want to make sure that the signal they're hearing is simply an *interleaving* of these two emissions, with nothing extra added in.

This describes the whole problem; we can make it a little more explicit as follows. Given a string x consisting of 0s and 1s, we write x^k to denote k copies of x concatenated together. We say that a string x' is a *repetition* of x if it is a prefix of x^k for some number k. So $x' = 10110110110$ is a repetition of $x = 101$.

We say that a string s is an *interleaving* of x and y if its symbols can be partitioned into two (not necessarily contiguous) subsequences s' and s'', so that s' is a repetition of x and s'' is a repetition of y. (So each symbol in s must belong to exactly one of s' or s''.) For example, if $x = 101$ and $y = 00$, then $s = 100010101$ is an interleaving of x and y, since characters $1, 2, 5, 7, 8, 9$ form 101101–a repetition of x–and the remaining characters $3, 4, 6$ form 000–a repetition of y.

In terms of our application, x and y are the repeating sequences from the two ships, and s is the signal we're listening to: We want to make sure s "unravels" into simple repetitions of x and y. Give an efficient algorithm that takes strings s, x, and y and decides if s is an interleaving of x and y.