CSE 40113: Algorithms
Homework 4

You may complete this homework in groups of 3 or less students. Note that the integrity policy applies: your group should write up your own work, although you're welcome to work on the problems in a larger group. If you have any questions, please re-read both the homework guidelines and the academic integrity policy carefully, and then come discuss any questions or concerns with me.

Required Problems

1. A local baker has come to you with the following problem. There are many orders that need to be filled, and for each item there is a baking time, b_i, and a subsequent cooling time, c_i. Each item must be baked and then cooled. Furthermore, only one item can be baked at any given time, since the oven is not large enough to fit two of the pans.

   The baker must decide on the order for baking the items. That is, the first item will be baked and then taken out of the oven to cool. The second item may be baked as soon as the first is taken out of the oven (that is, while that first item cools), and so on. The baker's goal is to minimize the overall time spent on the process (that is until all items have cooled). Your goal is to develop an efficient algorithm which produces a schedule whose completion time is as small as possible.

   To help you organize your thought, we suggest the following greedy rules:

   (a) Order the schedule based on b_i, from smallest baking time to largest baking time.

   (b) Order the schedule based on c_i, from largest cooling time to smallest cooling time.

   (c) Order the schedule based on the composite quantity $b_i + c_i$, ordered from smallest to largest.

   One of these greedy rules is guaranteed to produce the optimal schedule; the other rules are flawed. Your job is to figure out which is which! For each flawed rule, provide a relatively simple instance that demonstrates the non-optimality of the result. For the rule that is valid, give a proof that it guarantees an optimal completion time.

2. Suppose you are a simple shopkeeper living in a country with n different types of coins, with values $1 = c[1] < c[2] < \cdots < c[n]$. (In the U.S., $n = 6$ and the values are 1,5,10,25,50, and 100 cents, unless we get ride of the penny soon[1].) Your beloved and benevolent dictator, El Generalissimo, has decreed that whenever you give a customer change you must use the smallest possible number of coins, so as not to wear out the image of him so lovingly engraved on each coin.

[1] See https://www.npr.org/2025/02/10/nx-s1-5292082/trump-penny-mint-treasury

(a) In the U.S., there is a simple greedy algorithm that always results in the smallest number of coins: subtract the largest coin which is not too large and recursively give change for the reminder. El Generalissimo does not approve of American capitalist greed, however. Show that there is a set of coin values for which the greedy algorithm does not always give the smallest possible number of coins.

(b) Now suppose El Generalissimo decides to impose a currency system where the coin denominations are consecutive powers $b^0, b^1, b^2, \ldots, b^k$ of some integer $b \geq 2$. Prove that despite El Generalissimo's disapproval, the greedy algorithm described in part (a) does make optimal change in this system as well.

(Hint: You can start with powers of 2, if that makes things more concrete, but then be sure to justify why it generalizes for larger bases as well!)

3. You've been hired to store a sequence of n books on shelves in a library. The order of the books is fixed by the cataloging system and cannot be changed; each shelf must store a contiguous interval of the given sequence of books. You are given two arrays $H[1..n]$ and $T[1...n]$, where $H[i]$ and $T[i]$ are respectively the height and thickness of the i^{th} book in the sequence. All shelves in this library have the same length L; the total thickness of all books on any single shelf cannot exceed L.

(a) Suppose all the books have the same height h and the shelves have height larger than h, so every book fits on every shelf. Describe and analyze a greedy algorithm to store the books in as few shelves as possible. [Hint: The algorithm is obvious, but why is it correct?]

(b) That was a nice warmup, but now here's the real problem. In fact the books have different heights, but you can adjust the height of each shelf to match the tallest book on that shelf. (In particular, you can change the height of any empty shelf to zero.) Now your task is to store the books so that the sum of the heights of the shelves is as small as possible. Show that your greedy algorithm from part (a) does not always give the best solution to this problem. [Note: if you wanted to solve this optimally, you'd want to try dynamic programming!]

4. **Sample solved problem:**

Your friends are starting a security company that needs to obtain licenses for n different pieces of cryptographic software. Due to regulations, they can only obtain these licenses at the rate of at most one per month.

Each license is currently selling for a price of \$100. However, they are all becoming more expensive according to exponential growth curves: in particular, the cost of license j increases by a factor of $r_j > 1$ each month, where r_j is a given parameter. This means that if license j is purchased t months from now, it will cost $100 \cdot r_j^t$. We will assume that all the price growth rates are distinct; that is, $r_i \neq r_j$ for licenses $i \neq j$ (even though they start at the same price of \$100).

The question is: Given that the company can only buy at most one license a month, in which order should it buy the licenses so that the total amount of money it spends is as small as possible?

Give an algorithm that takes the n rates of price growth $r_1, r_2, \ldots, r_n$, and computes an order in which to buy the licenses so that the total amount of money spent is minimized. The running time of your algorithm should be polynomial in n.

Solution:

Two natural guesses for a good sequence would be to sort the r_i in decreasing order, or to sort them in increasing order. Faced with alternatives like this, it's perfectly reasonable to work out a small example and see if the example eliminates at least one of them. Here we could try $r_1 = 2, r_2 = 3$, and $r_3 = 4$. Buying the licenses in increasing order results in a total cost of

$$100(2 + 3^2 + 4^3) = 7,500,$$

while buying them in decreasing order results in a total cost of

$$100(4 + 3^2 + 2^3) = 2,100.$$

This tells us that increasing order is not the way to go. (On the other hand, it doesn't tell us immediately that decreasing order is the right answer, but our goal was just to eliminate one of the two options.)

Let's try proving that sorting the r_i in decreasing order in fact always gives the optimal solution. When a greedy algorithm works for problems like this, in which we put a set of things in an optimal order, we've seen in the text that it's often effective to try proving correctness using an exchange argument.

To do this here, let's suppose that there is an optimal solution O that differs from our solution S. (In other words, S consists of the licenses sorted in decreasing order.) So this optimal solution O must contain an inversion—that is, there must exist two neighboring months t and $t + 1$ such that the price increase rate of the license bought in month t (let us denote it by r_t) is less than that bought in month $t+1$ (similarly, we use r_{t+1} to denote this). Given that these must not be greedily ordered, we have $r_t < r_{t+1}$.

We claim that by exchanging these two purchases, we can strictly improve our optimal solution, which contradicts the assumption that O was optimal. Therefore if we succeed in showing this, we will successfully show that our algorithm is indeed the correct one.

Notice that if we swap these two purchases, the rest of the purchases are identically priced. In O, the amount paid during the two months involved in the swap is $100(r_t^t + r_{t+1}^{t+1})$. On

the other hand, if we swapped these two purchases, we would pay $100(r_{t+1}^t + r_t^{t+1})$. Since the constant 100 is common to both expressions, we want to show that the second term is less than the first one. So essentially, we want to show that $r_{t+1}^t + r_t^{t+1} < r_t^t + r_{t+1}^{t+1}$. If we rearrange to group like terms, we get the equivalent statement $r_t^{t+1} - r_t^t < r_{t+1}^{t+1} + r_{t+1}^t$, which is then equivalent to: $r_t(r_t - 1) < r_{t+1}^t(r_{t+1} - 1)$. This last inequality is true simply because $r_t > 1$ for all i, and because we started by assuming $r_t < r_{t+1}$, which implies the original desired inequality is also true since they are equivalent.

Our algorithm is then quite simple: sort by decreasing order and output the list. The algorithm's runtime is $O(n \log n)$ since that is the running time to sort.

Side Note: It's interesting to note that things become much less straightforward if we vary this question even a little. Suppose that instead of buying licenses whose prices increase, you're trying to sell off equipment whose cost is depreciating. Item i depreciates at a factor of $r_i < 1$ per month, starting from \$100, so if you sell it t months from now you will receive $100 \cdot r_i^t$. (In other words, the exponential rates are now less than 1, instead of greater than 1.) Ifyou can only sell one item per month, what is the optimal order in which to sell them? Here, it turns out that there are cases in which the optimal solution doesn't put the rates in either increasing or decreasing order (as in the input $\frac{3}{4}, \frac{1}{2}, \frac{1}{100}$).