CS 344: Design and Analysis of Computer Algorithms **Rutgers: Spring 2021**

Greedy Algorithms Practice

February 24, 2021

Instructor: Sepehr Assadi

Problem 1. You are hired by a company for consultation on selling or buying a certain share of stock. You are given an accurate prediction of the price of this stock for the next n days as an array $P[1:n]$ where $P[i]$ is the price in the i-th day. Your company can only have a single share of stock at any given day. Every day, you can decide whether to sell the share (if you already have one), buy a single share (if your company has no share at the moment), or do nothing this day. You are allowed to this as many times as you like throughout the course of these n days; for instance, you can buy a share at day 1, do nothing for days 2 and 3, sell it at day 4, and do nothing for days 5 to 10, buy a share at day 11 and so on and so forth. However, you cannot buy *and* sell the share in the same exact day.

Your goal is to find a strategy for buying and selling these shares so as to maximize the profit of your company; here, the profit is the the total prices you received on the days you sold the share minus the prices you paid for buying the shares on the other days. Design a greedy algorithm fo find the maximum profit for this problem with worst-case runtime of $O(n)$. You can assume the company always has the money to buy a share in any given day (if you already do not have one), and that you start with no share initially.

Example. A couple of examples for this problem:

- *Input:* $n = 6$ and $P = [7, 1, 5, 3, 6, 4]$

  Output: 7

  Explanation: Buy on day 2 (price = 1) and sell on day 3 (price = 5), then buy on day 4 (price = 3) and sell on day 5 (price = 6); the total profit will be $5 + 6 - (1 + 3) = 7$.

- *Input:* $n = 5$ and $P = [1, 2, 3, 4, 5]$

  Output: 4

  Explanation: Buy on day 1 (price = 1) and sell on day 5 (price = 5); the total profit will be $= 5 - 1 = 4$. Note that you cannot buy on day 1 and 2 simultaneously and sell them later as it requires you to hold more than one share; you must sell the share before you can buy it again.

- *Input:* $n = 5$ and $P = [7, 6, 4, 3, 1]$

  Output: 0

  Explanation: In this case, no transaction is done (since price only drops); the total profit will be 0.

Problem 2. Given an array $A[1:n]$ of n distinct integers, our goal is to find three different numbers from the array to form a triangle of largest perimeter whose sides are equal to these three numbers. Recall that any set of positive integers can form a triangle as long as sum of every two of them is larger than the third one, and that the perimeter of a triangle is simply the sum of length of its sides.

Design a greedy algorithm for this problem with worst-case running time of $O(n \log n)$.

Example. A couple of examples for this problem:

- *Input:* $n = 4$ and $A = [4, 6, 3, 2]$;

  Output: $4, 6, 3$ with perimeter 13;

  Explanation: We can form a triangle with numbers $3, 4, 6$ and since these are the largest three numbers of the array, clearly this triangle has the largest perimeter.

- *Input:* $n = 5$ and $A = [8, 1, 3, 2, 4]$;

  Output: $4, 3, 2$ with perimeter 9;

  Explanation: We cannot pick 8 as part of our triangle because it does not form a triangle with any of the remaining numbers. This leaves $4, 3, 2$ as the largest number and since they can also form a triangle, we can pick them.

- *Input:* $n = 4$ and $A = [4, 6, 2, 1]$

  Output: 'there is no triangle'

  Explanation: There is simply no triangle with sides chosen from A.

Problem 3. You are given a set of n (closed) intervals on a line:

$$[a_1, b_1], [a_2, b_2], ..., [a_n, b_n].$$

Design an $O(n \log n)$ time greedy algorithm to select the *minimum* number of points on the line between $[\min_i a_i, \max_j b_j]$ such that any input interval contains at least one of the chosen points.

Example: If the following 5 intervals are given to you:

$$[2, 5], [3, 9], [2.5, 9.5], [4, 8], [7, 9],$$

then a correct answer is: $\{5, 9\}$ (the first four intervals contain number 5 and the last contains number 9; we also definitely need two points since $[2, 5]$ and $[7, 9]$ are disjoint and no single point can take care of both of them at the same time).

Problem 4. You are given a set of n (closed) intervals on a line:

$$[a_1, b_1], [a_2, b_2], ..., [a_n, b_n].$$

Design an $O(n \log n)$ time greedy algorithm to *color* these intervals with *minimum* number of possible colors so that any two intervals that are *not* disjoint are colored differently.

Example: If the following 5 intervals are given to you:

$$[2, 5], [3, 6], [6, 8], [4, 9], [7, 9],$$

then a correct answer is 3 colors with one possible coloring

$$[2, 5] : 1\ ,\ [3, 6] : 2\ ,\ [4, 9] : 3\ ,\ [6, 8] : 1\ ,\ [7 : 9] : 2.$$

Note that we definitely need at least 3 colors because intervals $[2, 5], [3, 6], [4, 9]$ are all intersecting with each other and thus need 3 different colors.