Algorithms Laboratory (CS29203)
Assignment 3: Greedy Algorithms
Department of CSE, IIT Kharagpur

31st August 2023

Question-1 (50 points)

In last summer, the CSE department has purchased a powerful workstation that can run unlimited number of programs at the same time. You can switch on the workstation when you need to run a program, and can switch it off when it is idle. Since the workstation consumes a lot of power and CSE department wants to cut down the electricity bill, the goal is to optimize the power consumption for the workstation.

Let us say that each program that is run on the workstation has a starting time, ending time, and a specific duration. Say that these are represented as a 2D integer array programs[i] = [start_i, end_i, duration_i], which means that the i-th program runs for duration_i seconds, (which need not be continuous) within the time period start_i to end_i (both inclusive). Note that start time, end time and duration, all are integers.

Now given a list of programs in the above format, your task is to find the minimum time during which the workstation should be switched on to complete all the programs. The workstation may be turned on only when it needs to run a task. You can also turn it off if it is idle. You have to implement a greedy strategy to solve this problem.

For example, let us consider the following array: programs = $[[2,3,1],[4,5,1],[1,5,2]]$. Then the minimum time during which the workstation should be switched on to complete all the programs is 2. This can be explianed by the following logic. The first program can be run in the inclusive time range $[2,2]$. The second program can be run in the inclusive time range $[5,5]$. The third program can be run in the two inclusive time ranges $[2,2]$ and $[5,5]$.

Example:

```
(Input) programs = [[1,3,2],[2,5,3],[5,6,2]]
(Output) 4

Explanation:

- The first program can be run in the inclusive time range [2, 3].
- The second program can be run in the inclusive time ranges [2, 3] and [5, 5].
- The third program can be run in the two inclusive time range [5, 6].

So the workstation will be on for a total of 4 seconds.
```

Question-2 (50 points)

You are playing an alphabet game with your friend. One of you start (initiator) with a proposing a string of alphabets, while the other remains empty. The rule of the game is to apply the following two operations until both you and your friend run out of alphabets:

- The person who initiates the game with a string of alphabets, removes the first character of the string and gives it to the other person (the receiver). That person will append this character to his/her set of alphabet(s). Note that the initiator can supply characters in this way for multiple times until he decides to stop.
- The receiver deletes the last character of the string that he has, and writes that alphabet on a piece of paper. Note that the receiver can perform this operation multiple times until he decides to stop.

The above operations can be performed as many times as one wants.

Now the goal of the game is to find the final string that is written on the paper to have the property that among all possible combinations of strings that can be generated (i.e. *can* be written on the paper) by performing the above operations, it comes first in the alphabetical order.

Let's take an example that the initiator string is i = "bdda". At this moment the receiver string is empty, i.e. r = " ", and so is the content of the paper, i.e., p = " ". Now in order to achieve the goal, the final string in the paper should start with alphabet "a". In order to do this, we perform the first operation four times, and the second operation once. At this moment, the alphabet "a" is written on the paper. Now since the initiator string is empty, we do not have any other choice but perform the second operation three more times, thus giving the final answer "addb". Notice that among all stings that can be generated (i.e. *can* be written on the paper) from i = "bdda" by performing the two operations, "addb" will come first in the alphabetical order.

Your task is to develop and implement a greedy strategy to solve the above problem.

Example 1:

```
(Input) i = "bac"
(Output) "abc"

Explanation:

- Perform first operation twice: i = "c", r = "ba", p = "".
- Perform second operation twice i = "c", r = "", p = "ab".
- Perform first operation once:  i = "", r = "c", p = "ab".
- Perform second operation once: i = "", r = "", p = "abc".
```

Example 2:

```
(Input) i = "cbaabc"
(Output) "aabbcc"

Explanation:
- Perform First operation 3 times
- Perform Second operation once
- Perform First operation once
- Perform Second operation 2 times
- Perform First operation once
- Perform Second operation once
- Perform First operation once
- Perform Second operation 2 times
```