Assignment-3

Data Structures and Algorithms
Heaps and AVL Trees

Due Date: 16th March, 2025 11:59 PM

Important Notes

- You are only allowed to use **C** for this assignment.
- You are **not allowed** to use the inbuilt `qsort` function in C.
- You are not allowed to copy the exact code given in class. You are allowed to note the logic, but you need to implement the code by yourself.
- We are providing a boilerplate for AVL tree. This will **not** be considered for MOSS. You may still choose to implement it yourself.
- Plagiarism Policy. Please read this before you start the assignment.

TOTAL: [100 PTS]

1 The Chocolate Dispute [25 PTS]

1.1 Problem Statement

Aayush and Aniket are embroiled in a dispute over a collection of chocolates. Initially, Aayush possesses n chocolates, each laid out along the $X-axis$. The i^{th} chocolate covers a segment starting at position l_i and ending at position r_i (both inclusive). The size of a chocolate is defined as,

$$size = r_i - l_i + 1$$

Aniket chooses a point k on the $X-axis$. If there are multiple chocolates covering that point, Aayush, being greedy, will hand over the chocolate with the smallest size to Aniket. If no chocolate covers the point k, then Aniket receives nothing. Your task is to determine, for each query point, the size of the chocolate that Aniket will obtain. If he doesn't receive a chocolate, output -1.

1.2 Input Format

- The first line contains an integer n, the number of chocolates.
- Each of the next n lines contains two space-separated integers l_i and r_i, representing the starting and ending points of the i-th chocolate on the $X-axis$.
- The following line contains an integer q, the number of queries
- Each of the next q lines contains a single integer k, representing the query point on the $X-axis$.

1.3 Output Format

For each query, output a single integer — the size of the smallest chocolate covering the point k. If no chocolate covers the point, output -1.

1.4 Example Testcases

1.4.1 Example 1

Sample Input	Sample Output
3 1 6 2 5 3 4 3 2 3 7	4 2 -1

- For $k = 2$, both the first and second chocolates cover the point; however, the second chocolate has a smaller size (4) compared to the first (6), so the output is 4. For $k = 3$, all three chocolates cover the point, and the smallest among them is the third chocolate ($size = 2$). For $k = 7$, no chocolate covers the point, hence the output is -1.

1.5 Constraints

- $1 \leq n \leq 10^5$
- $1 \leq q \leq 10^5$
- $1 \leq l_i \leq r_i \leq 10^7$
- $1 \leq k \leq 10^7$

2 The Great Wall Reconstruction [40 PTS]

2.1 Problem Statement

A city is rebuilding its ancient Great Wall, which consists of multiple connected sections. Each section has a starting position, an ending position, and a specific height. Due to erosion and past reconstructions, some sections overlap, forming a continuous outer contour when viewed from a distance.

The government needs your help to determine the **visible outline** of the reconstructed wall, as seen from afar.

2.2 Input Format

- An integer n, representing the number of sections.
- n lines follow, each containing three integers: x_1, x_2, and h.

2.3 Output Format

The output should be a list of key points representing the visible outline of the reconstructed Great Wall. Each key point is a significant change in height along the structure.

- The list should be sorted in increasing order of the x-coordinate.
- Each key point should be represented as $[x, h]$, where:
  - x is the horizontal position.
  - h is the height of the wall at that position.
- The last key point should always have a height of 0, indicating the termination of the wall's visible structure.

2.4 Example Testcases

2.4.1 Example 1

Sample Input	**Sample Output**
5 2 9 10 3 7 15 5 12 12 15 20 10 19 24 8	2 10 3 15 7 12 12 0 15 10 20 8 24 0

- Figure A shows the given sections.
- Figure B depicts the visible outline.
- The marked points in Figure B represent key points in the final output.

Below is a visual representation of the input and expected output:

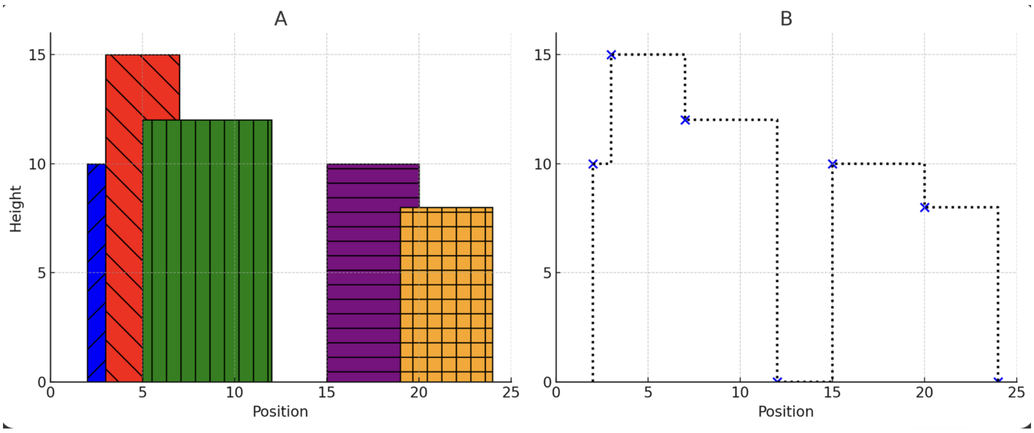

Figure A (left) represents the given sections, and Figure B (right) represents the visible outline.

2.4.2 Example 2

Sample Input	Sample Output
2 0 2 3 2 5 3	0 3 5 0

2.5 Constraints

- $1 \leq \text{sections.length} \leq 10^6$
- $0 \leq x_1 < x_2 \leq 2^{31} - 1$
- $1 \leq h \leq 2^{31} - 1$

3 The Canteen Cost Conundrum [35 PTS]

3.1 Problem Statement

With the rising prices in the college canteen, students are eager to find the most affordable way to assemble a meal.
Consider a dynamic menu where:

- New items are continuously added,
- Some items are removed, and
- Prices of certain items are updated.

A meal consists of k distinct food items or as many distinct items as available if there are fewer than k options.

3.2 Input Format

- The first line contains two integers n and k, representing the number of items initially on the menu and the number of items required to assemble a meal.
- The following line contains n space-separated integers which represent the prices of the items on the menu initially.
- The third line contains an integer u the number of updates to the menu.
- The following u lines contain updates of the form
  - $1\ x$: Add an item with price x to the menu
  - $2\ x$: Remove an item with price x from the menu. (It is guaranteed that there is an item with price x in the menu)
  - $3\ x\ y$: Update the price of an item with price x to y. (It is guaranteed that there is an item with price x in the menu)

3.3 Output Format

After every update operation output an integer on a separate line which is the minimum cost to assemble a meal.

3.4 Example Testcases

3.4.1 Example 1

Sample Input	Sample Output
8 3 1 2 3 4 5 6 7 8 4 1 9 1 2 2 1 3 2 1	6 5 7 6

- After adding an item of price 9 the cheapest meal costs $1 + 2 + 3 = 6$
- After adding an item of price 2 the cheapest meal costs $1 + 2 + 2 = 5$
- After removing an item of price 1 the cheapest meal costs $2 + 2 + 3 = 7$
- After changing the price of an item from 2 to 1 the cheapest meal costs $1 + 2 + 3 = 6$

3.4.2 Example 2

Sample Input	Sample Output

`4 5`
`1 2 2 3`
`3`
`3 2 3`
`1 1`
`3 3 2`
	`9`
	`10`
	`9`

- After changing the price of an item from 2 to 3 the cheapest meal costs $1+2+3+3=9$ (Note that there are only 4 items available)
- After adding an item costing 1 the price of the cheapest meal becomes $1+1+2+3+3=10$
- After updating the cost of an item from 3 to 2 the cost of the cheapest meal becomes $1+1+2+2+3=9$

3.5 Constraints

- $1 \leq n \leq 2 \times 10^5$
- $1 \leq k \leq 10^5$
- $1 \leq p \leq 10^9$
- $1 \leq u \leq 2 \times 10^5$

Good Luck, have fun coding!