Problem 3414: Maximum Score of Non-overlapping Intervals

Problem Information

Difficulty: Hard
Acceptance Rate: 30.57%
Paid Only: No
Tags: Array, Binary Search, Dynamic Programming, Sorting

Problem Description

You are given a 2D integer array `intervals`, where `intervals[i] = [li, ri, weighti]`. Interval `i` starts at position `li` and ends at `ri`, and has a weight of `weighti`. You can choose _up to_ 4 **non-overlapping** intervals. The **score** of the chosen intervals is defined as the total sum of their weights.

Return the lexicographically smallest array of at most 4 indices from `intervals` with **maximum** score, representing your choice of non- overlapping intervals.

Two intervals are said to be **non-overlapping** if they do not share any points. In particular, intervals sharing a left or right boundary are considered overlapping.

Example 1:

Input: intervals = [[1,3,2],[4,5,2],[1,5,5],[6,9,3],[6,7,1],[8,9,1]]

Output: [2,3]

Explanation:

You can choose the intervals with indices 2, and 3 with respective weights of 5, and 3.

Example 2:

Input: intervals = [[5,8,1],[6,7,7],[4,7,3],[9,10,6],[7,8,2],[11,14,3],[3,5,5]]

Output: [1,3,5,6]

Explanation:

You can choose the intervals with indices 1, 3, 5, and 6 with respective weights of 7, 6, 3, and 5.

Constraints:

* `1 <= intevals.length <= 5 * 104` * `intervals[i].length == 3` * `intervals[i] = [li, ri, weighti]` * `1 <= li <= ri <= 109` * `1 <= weighti <= 109`

Code Snippets

C++:

```
class Solution {
public:
vector<int> maximumWeight(vector<vector<int>>& intervals) {

}
};
```

Java:

```
class Solution {
public int[] maximumWeight(List<List<Integer>> intervals) {

}
}
```

Python3:

```
class Solution:
def maximumWeight(self, intervals: List[List[int]]) -> List[int]:
```