1086. High Five

Easy 589 96 Add to List Share

Given a list of the scores of different students, `items`, where `items[i] = [ID_i, score_i]` represents one score from a student with `ID_i`, calculate each student's **top five average**.

Return *the answer as an array of pairs* `result`, *where* `result[j] = [ID_j, topFiveAverage_j]` *represents the student with* `ID_j` *and their* ***top five average****. Sort* `result` *by* `ID_j` *in* ***increasing order****.*

A student's **top five average** is calculated by taking the sum of their top five scores and dividing it by `5` using **integer division**.

Example 1:

```
Input: items = [[1,91],[1,92],[2,93],[2,97],[1,60],[2,77],[1,65],[1,87],[1,100],[2,100],[2,76]]
Output: [[1,87],[2,88]]
Explanation:
The student with ID = 1 got scores 91, 92, 60, 65, 87, and 100. Their top five average is (100 + 92 + 91 + 87 + 65) / 5 = 87.
The student with ID = 2 got scores 93, 97, 77, 100, and 76. Their top five average is (100 + 97 + 93 + 77 + 76) / 5 = 88.6, but with integer division their average converts to 88.
```

Example 2:

```
Input: items = [[1,100],[7,100],[1,100],[7,100],[1,100],[7,100],[1,100],[7,100],[1,100],[7,100]]
Output: [[1,100],[7,100]]
```

Constraints:

- `1 <= items.length <= 1000`
- `items[i].length == 2`
- `1 <= ID_i <= 1000`
- `0 <= score_i <= 100`
- For each `ID_i`, there will be **at least** five scores.

Accepted 74,045 | Submissions 97,635

Seen this question in a real interview before? Yes No

Companies

0 ~ 6 months 6 months ~ 1 year 1 year ~ 2 years

Goldman Sachs | 21

Related Topics

Array Hash Table Sorting

Hide Hint 1

How can we solve the problem if we have just one student?

Hide Hint 2

Given an student sort their grades and get the top 5 average.

Hide Hint 3

Generalize the idea to do it for many students.

Java Autocomplete

```java
class Solution {

    private int K;

    public int[][] highFive(int[][] items) {
        this.K = 5;
        Arrays.sort(
                items,
                new Comparator<int[]>() {
                    @Override
                    public int compare(int[] a, int[] b) {
                        if (a[0] != b[0])
                            // item with lower id goes first
                            return a[0] - b[0];
                        // in case of tie for ids, item with higher score goes first
                        return b[1] - a[1];
                    }
                });
        List<int[]> solution = new ArrayList<>();
        int n = items.length;
        int i = 0;
        while (i < n) {
            int id = items[i][0];
            int sum = 0;
            // obtain total using the top 5 scores
            for (int k = i; k < i + this.K; ++k)
                sum += items[k][1];
            // ignore all the other scores for the same id
            while (i < n && items[i][0] == id)
                i++;
            solution.add(new int[] {id, sum / this.K});
        }
        int[][] solutionArray = new int[solution.size()][];
        return solution.toArray(solutionArray);
    }
}


class Solution {
    private int K;

    public int[][] highFive(int[][] items) {
        this.K = 5;
        TreeMap<Integer, Queue<Integer>> allScores = new TreeMap<>();
        for (int[] item : items) {
            int id = item[0];
            int score = item[1];
            if (!allScores.containsKey(id))
                // max heap
                allScores.put(id, new PriorityQueue<>((a,b) -> b - a));
            // Add score to the max heap
            allScores.get(id).add(score);
        }

```