Problem 1086: High Five

Problem Information

Difficulty: Easy
Acceptance Rate: 0.00%
Paid Only: No

Problem Description

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.

Code Snippets

C++:

```
class Solution {
public:
vector<vector<int>> highFive(vector<vector<int>>& items) {

}
};
```

Java:

```
class Solution {
public int[][] highFive(int[][] items) {

}
}
```

Python3:

```
class Solution:
def highFive(self, items: List[List[int]]) -> List[List[int]]:
```

Python:

```
class Solution(object):
def highFive(self, items):
"""
:type items: List[List[int]]
:rtype: List[List[int]]
"""
```

JavaScript:

```
/**
* @param {number[][]} items
* @return {number[][]}
*/
var highFive = function(items) {

};
```

TypeScript:

```
function highFive(items: number[][]): number[][] {

};
```

C#:

```
public class Solution {
public int[][] HighFive(int[][] items) {

}
}
```

C:

```
/**
* Return an array of arrays of size *returnSize.
* The sizes of the arrays are returned as *returnColumnSizes array.
* Note: Both returned array and *columnSizes array must be malloced, assume
caller calls free().
*/
int** highFive(int** items, int itemsSize, int* itemsColSize, int*
returnSize, int** returnColumnSizes) {

}
```

Go:

```
func highFive(items [][]int) [][]int {

}
```

Kotlin:

```
class Solution {
fun highFive(items: Array<IntArray>): Array<IntArray> {

}
}
```

Swift:

```
class Solution {
func highFive(_ items: [[Int]]) -> [[Int]] {

}
}
```

Rust:

```
impl Solution {
pub fn high_five(items: Vec<Vec<i32>>) -> Vec<Vec<i32>> {

}
}
```

Ruby:

```
# @param {Integer[][]} items
# @return {Integer[][]}
def high_five(items)

end
```

PHP:

```
class Solution {

/**
* @param Integer[][] $items
* @return Integer[][]
*/
function highFive($items) {

}
}
```

Dart:

```
class Solution {
List<List<int>> highFive(List<List<int>> items) {

}
}
```

Scala:

```
object Solution {
def highFive(items: Array[Array[Int]]): Array[Array[Int]] = {

}
}
```

Elixir:

```
defmodule Solution do
@spec high_five(items :: [[integer]]) :: [[integer]]
def high_five(items) do

end
end
```

Erlang:

```
-spec high_five(Items :: [[integer()]]) -> [[integer()]].
high_five(Items) ->
.
```

Racket:

```
(define/contract (high-five items)
(-> (listof (listof exact-integer?)) (listof (listof exact-integer?)))
)
```

Solutions

C++ Solution:

```
/*
* Problem: High Five
* Difficulty: Easy
* Tags: array, hash, sort, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/
```

```
class Solution {
public:
vector<vector<int>> highFive(vector<vector<int>>& items) {

}
};
```

Java Solution:

```
/**
* Problem: High Five
* Difficulty: Easy
* Tags: array, hash, sort, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public int[][] highFive(int[][] items) {

}
}
```

Python3 Solution:

```
"""
Problem: High Five
Difficulty: Easy
Tags: array, hash, sort, queue, heap

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

class Solution:
def highFive(self, items: List[List[int]]) -> List[List[int]]:
# TODO: Implement optimized solution
```

```
pass
```

Python Solution:

```
class Solution(object):
def highFive(self, items):
"""
:type items: List[List[int]]
:rtype: List[List[int]]
"""
```

JavaScript Solution:

```
/**
* Problem: High Five
* Difficulty: Easy
* Tags: array, hash, sort, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
* @param {number[][]} items
* @return {number[][]}
*/
var highFive = function(items) {

};
```

TypeScript Solution:

```
/**
* Problem: High Five
* Difficulty: Easy
* Tags: array, hash, sort, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
```

```
*/

function highFive(items: number[][]): number[][] {

};
```

C# Solution:

```
/*
* Problem: High Five
* Difficulty: Easy
* Tags: array, hash, sort, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public int[][] HighFive(int[][] items) {

}
}
```

C Solution:

```
/*
* Problem: High Five
* Difficulty: Easy
* Tags: array, hash, sort, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
* Return an array of arrays of size *returnSize.
* The sizes of the arrays are returned as *returnColumnSizes array.
* Note: Both returned array and *columnSizes array must be malloced, assume
caller calls free().
```

```
*/
int** highFive(int** items, int itemsSize, int* itemsColSize, int*
returnSize, int** returnColumnSizes) {

}
```

Go Solution:

```
// Problem: High Five
// Difficulty: Easy
// Tags: array, hash, sort, queue, heap
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func highFive(items [][]int) [][]int {

}
```

Kotlin Solution:

```
class Solution {
fun highFive(items: Array<IntArray>): Array<IntArray> {

}
}
```

Swift Solution:

```
class Solution {
func highFive(_ items: [[Int]]) -> [[Int]] {

}
}
```

Rust Solution:

```
// Problem: High Five
// Difficulty: Easy
// Tags: array, hash, sort, queue, heap
```

```
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

impl Solution {
pub fn high_five(items: Vec<Vec<i32>>) -> Vec<Vec<i32>> {

}
}
```

Ruby Solution:

```
# @param {Integer[][]} items
# @return {Integer[][]}
def high_five(items)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[][] $items
* @return Integer[][]
*/
function highFive($items) {

}
}
```

Dart Solution:

```
class Solution {
List<List<int>> highFive(List<List<int>> items) {

}
}
```

Scala Solution:

```
object Solution {
def highFive(items: Array[Array[Int]]): Array[Array[Int]] = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec high_five(items :: [[integer]]) :: [[integer]]
def high_five(items) do

end
end
```

Erlang Solution:

```
-spec high_five(Items :: [[integer()]]) -> [[integer()]].
high_five(Items) ->
.
```

Racket Solution:

```
(define/contract (high-five items)
(-> (listof (listof exact-integer?)) (listof (listof exact-integer?)))
)
```