Relative Ranks

Given scores of **N** athletes, find their relative ranks and the people with the top three highest scores, who will be awarded medals: "Gold Medal", "Silver Medal" and "Bronze Medal".

Example 1:

```
Input: [5, 4, 3, 2, 1]
Output: ["Gold Medal", "Silver Medal", "Bronze Medal", "4", "5"]
Explanation: The first three athletes got the top three highest scores, so they got
 "Gold Medal", "Silver Medal" and "Bronze Medal".
For the left two athletes, you just need to output their relative ranks according t
o their scores.
```

Note:

1. N is a positive integer and won't exceed 10,000.
2. All the scores of athletes are guaranteed to be unique.

Solution 1

Basically this question is to find out the `score` -> `ranking` mapping. The easiest way is to sort those `scores` in `nums`. But we will lose their original order. We can create (`score`, `original index`) pairs and sort them by `score` decreasingly. Then we will have `score` -> `ranking` (new index) mapping and we can use `original index` to create the result.

Time complexity: O(NlgN). Space complexity: O(N). N is the number of scores.

```
Example:

nums[i]    : [10, 3, 8, 9, 4]
pair[i][0] : [10, 3, 8, 9, 4]
pair[i][1] : [ 0, 1, 2, 3, 4]

After sort:
pair[i][0] : [10, 9, 8, 4, 3]
pair[i][1] : [ 0, 3, 2, 4, 1]
```

```
public class Solution {
    public String[] findRelativeRanks(int[] nums) {
        int[][] pair = new int[nums.length][2];

        for (int i = 0; i < nums.length; i++) {
            pair[i][0] = nums[i];
            pair[i][1] = i;
        }

        Arrays.sort(pair, (a, b) -> (b[0] - a[0]));

        String[] result = new String[nums.length];

        for (int i = 0; i < nums.length; i++) {
            if (i == 0) {
                result[pair[i][1]] = "Gold Medal";
            }
            else if (i == 1) {
                result[pair[i][1]] = "Silver Medal";
            }
            else if (i == 2) {
                result[pair[i][1]] = "Bronze Medal";
            }
            else {
                result[pair[i][1]] = (i + 1) + "";
            }
        }

        return result;
    }
}
```

Also we can use an one dimension array. This will save a little bit space but space

complexity is still O(n).

```
public class Solution {
    public String[] findRelativeRanks(int[] nums) {
        Integer[] index = new Integer[nums.length];

        for (int i = 0; i < nums.length; i++) {
            index[i] = i;
        }

        Arrays.sort(index, (a, b) -> (nums[b] - nums[a]));

        String[] result = new String[nums.length];

        for (int i = 0; i < nums.length; i++) {
            if (i == 0) {
                result[index[i]] = "Gold Medal";
            }
            else if (i == 1) {
                result[index[i]] = "Silver Medal";
            }
            else if (i == 2) {
                result[index[i]] = "Bronze Medal";
            }
            else {
                result[index[i]] = (i + 1) + "";
            }
        }

        return result;
    }
}
```

written by shawngao original link here

Solution 2

Use a dictionary mapping scores to ranks.

```
def findRelativeRanks(self, nums):
    sort = sorted(nums)[::-1]
    rank = ["Gold Medal", "Silver Medal", "Bronze Medal"] + map(str, range(4, len
(nums) + 1))
    return map(dict(zip(sort, rank)).get, nums)
```

written by StefanPochmann original link here

Solution 3

```
class Solution {
public:
    vector<string> findRelativeRanks(vector<int>& nums) {
        vector<int> rank;
        for(int i=0; i<nums.size(); ++i) rank.push_back(i);

        sort(rank.begin(), rank.end(), [&](int a, int b){return nums[a] > nums[b]
;});
        vector<string> ranks(nums.size());

        for(int i=3; i<nums.size(); ++i){
            ranks[rank[i]] = std::to_string(i+1);
        }

        if(nums.size() > 0) ranks[rank[0]] = "Gold Medal";
        if(nums.size() > 1) ranks[rank[1]] = "Silver Medal";
        if(nums.size() > 2) ranks[rank[2]] = "Bronze Medal";

        return ranks;
    }
};
```

written by kevin36 original link here