Algorithm Challenge

The Sale Stock Engineering team wants to increase the amount of time Sistas spend at our application. For that purpose,
the team has just launched a mini-game at our application. To reward the players, the top scorer of each hour will be rewarded 10 SS points + bonus SS points. **The bonus points rewarded are the number of top scores in the previous hours that are lower than the top score in this hour**. (Each SS point is worth 1.000 IDR, good for buying presents for your significant other and/or your children :D). The following is an illustration.

- If the top scores of hour-1, hour-2, and hour-3 are 250, 1820, and 870,
- and if the top score of hour-4 = 1000,
- then 2 **bonus** points are rewarded because 250 and 870 are lower than 1000.
- The top scorer of hour-4 will receive (10 + 2) SS points.

Your VP of Engineering wants to find out the approximate cost of this initiative to Sale Stock and you have been asked to solve this problem! You will be given a set of simulated `scores`, which is an array of numbers representing the top scores, starting from hour-1. Tell him how many SS points we have given for free to all the winners by completing this function.

```
def totalCost(scores):
    answer = 0
    # write your solution here
    # .
    # .
    # .
    # .
    # .
    # .
    # .
    # .
    # .
    # .
    # .
    return answer
```

Example
For `scores = [250, 1820, 870, 1000, 2000]`,

- SS points awarded to the top scorer of hour-1 = 10 + 0 = 10 points
- SS points awarded to the top scorer of hour-2 = 10 + 1 = 11 points
- SS points awarded to the top scorer of hour-3 = 10 + 1 = 11 points
- SS points awarded to the top scorer of hour-4 = 10 + 2 = 12 points
- SS points awarded to the top scorer of hour-5= 10 + 4 = 14 points

The output should be `totalCost(scores) = 58`.

Details

- State and explain the time complexity of your algorithm.
- A brief explanation of how your code works would be nice.
- You may use external libraries such as Fenwick Tree written by other people, with proper citations.
- Execution time limit = 4 seconds (Python 3.6)
- Guaranteed input constraints:
  - `0 < len(scores) < 100000`
  - `0 < scores[i] < 10000`
- Sample test cases

```
scores = [1874, 1339, 5617, 8331, 5424, 9667]
totalCost(scores) = 72

scores = [1675, 4660, 7028, 8022, 2529, 3270, 2472, 420, 3024, 5501,
4647, 313, 3568, 4105, 7372, 8680, 1966, 3952, 5320, 7663,
2828, 5868, 286, 9149, 7979, 6050, 1070, 5388]
totalCost(scores) = 496

score = list(range(1, 11112)) * 9
totalCost(scores) = 2778472215
```