





Climbing the Leaderboard ☆

50.11 more points to get your next star!

Rank: **17364** | Points: **2149.89/2200**



①

Problem Submissions Leaderboard Editorial △

Alice is playing an arcade game and wants to climb to the top of the leaderboard and wants to track her ranking. The game uses Dense Ranking, so its leaderboard works like this:

- $\bullet\,\,$ The player with the highest score is ranked number 1 on the leaderboard.
- Players who have equal scores receive the same ranking number, and the next player(s) receive the immediately following ranking number.

For example, the four players on the leaderboard have high scores of 100, 90, 90, and 80. Those players will have ranks 1, 2, 2, and 3, respectively. If Alice's scores are 70, 80 and 105, her rankings after each game are 4^{th} , 3^{rd} and 1^{st} .

Function Description

Complete the climbingLeaderboard function in the editor below. It should return an integer array where each element res[j] represents Alice's rank after the j^{th} game.

climbingLeaderboard has the following parameter(s):

- scores: an array of integers that represent leaderboard scores
- alice: an array of integers that represent Alice's scores

Input Format

The first line contains an integer n, the number of players on the leaderboard.

The next line contains n space-separated integers scores[i], the leaderboard scores in decreasing order.

The next line contains an integer, m, denoting the number games Alice plays.

The last line contains m space-separated integers alice[j], the game scores.

Constraints

- $1 \le n \le 2 \times 10^5$
- $1 \le m \le 2 \times 10^5$
- $0 \le scores[i] \le 10^9$ for $0 \le i < n$
- $0 \le alice[j] \le 10^9$ for $0 \le j < m$
- The existing leaderboard, *scores*, is in descending order.
- Alice's scores, *alice*, are in ascending order.

Subtask

For **60%** of the maximum score:

- $1 \le n \le 200$
- $1 \le m \le 200$

Output Format

Print m integers. The j^{th} integer should indicate Alice's rank after playing the j^{th} game.



100 100 50 40 40 20 10

Array: scores

7 100 100 50 40 40 20 10 4 5 25 50 120

5 25 50 120

Array: alice

Sample Output 1

6

4

2

1

Explanation 1

Alice starts playing with **7** players already on the leaderboard, which looks like this:

Rank	Name	Score
1	Emma	100
1	David	100
2	Caroline	50
3	Ritika	40
3	Tom	40
4	Heraldo	20
5	Riley	10

After Alice finishes game $\mathbf{0}$, her score is $\mathbf{5}$ and her ranking is $\mathbf{6}$:



Harm	Hamo	00010
1	Emma	100
1	David	100
2	Caroline	50
3	Ritika	40
3	Tom	40
4	Heraldo	20
5	Riley	10
6	Alice	5

After Alice finishes game 1, her score is 25 and her ranking is 4:

Rank	Name	Score
1	Emma	100
1	David	100
2	Caroline	50
3	Ritika	40
3	Tom	40
4	Alice	25
5	Heraldo	20
6	Riley	10

After Alice finishes game 2, her score is 50 and her ranking is tied with Caroline at 2:

Rank	Name	Score
1	Emma	100
1	David	100
2	Caroline	50
2	Alice	50
3	Ritika	40
3	Tom	40
4	Heraldo	20
5	Riley	10

After Alice finishes game **3**, her score is **120** and her ranking is **1**:





Harm	Hamo	00010
1	Alice	120
2	Emma	100
2	David	100
3	Caroline	50
4	Ritika	40
4	Tom	40
5	Heraldo	20
6	Riley	10

Sample Input 2

100 90 90 80 75 60

Array: scores

50 65 77 90 102

Array: alice

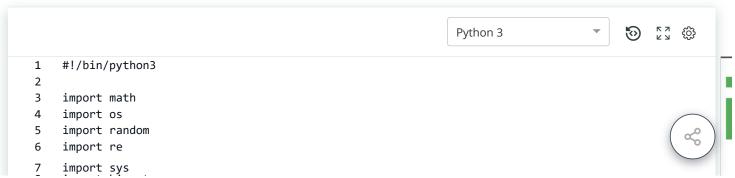
Sample Output 2

6

5

4 2

1



Copy Download

6 100 90 90 80 75 60 5

50 65 77 90 102

```
8
     import bisect
 9
10
     def remove_duplicates_from_scores(scores):
         unique scores = []
11
12
         last_score = None
13
         for score in scores:
14
             if (score != last_score):
15
                 unique_scores.append(score)
16
                 last_score = score
17
18
         #print ("unique {}".format(unique_scores))
19
         return unique_scores
20
21
     def find_alice_rank(scores, last_rank, alice_score):
22
         rank = last_rank
23
         for score in scores[:last_rank]:
             if alice_score >= score:
24
25
                  return rank
26
             rank += 1
27
         return rank
28
29
     # Complete the climbingLeaderboard function below.
     def climbingLeaderboard(scores, alice):
30
31
         result = []
32
33
         scores = remove_duplicates_from_scores(scores);
34
35
         rank = 0
36
         for alice_score in alice:
             rank = find_alice_rank(scores, rank, alice_score)
37
38
             result.append(rank)
                                                                                            Line: 1 Col: 1
```

Run Code

Submit Code

Contest Calendar | Blog | Scoring | Environment | FAQ | About Us | Support | Careers | Terms Of Service | Privacy Policy | Request a Feature

