< Back | Python binary search solution

CodeLenin ★ 47 Last Edit: September 7, 2019 10:11 PM 799 VIEWS

8

Basic idea is to group all indexes by color.
And for each index in the queries, do binary search on the color group.
There are 3 possibilites

1. the index is before the list(list of indexes for a color group)
2. the index is after the list
3. the index is in the middle of the list

```
class Solution:
    def shortestDistanceColor(self, colors: List[int], queries: List[List[int]]) -> List[int]:
        maps = collections.defaultdict(list)
        # group all indexes by color
        for i, v in enumerate(colors):
            maps[v].append(i)

        ans = []
        for i, c in queries:
            if c in maps:
                # search where the element can be inserted
                index = bisect.bisect_left(maps[c], i)
                if index == 0:
                    ans.append(abs(i-maps[c][0]))
                elif index >= len(maps[c]):
                    ans.append(i-maps[c][-1])
                else:
                    ans.append(min(abs(i-maps[c][index-1]), abs(maps[c][index]-i)))
            else:
                ans.append(-1)

        return ans
```

Comments: 4 Best Most Votes Newest to Oldest Oldest to Newest

Type comment here... (Markdown is supported)

Post

Keraju ★ 0 January 16, 2020 4:28 PM

Good solution. Easy to understand.:-)

0 Reply

carlsagan21 ★ 2 December 7, 2019 3:16 PM

So this is Q * logN?
Left/Right search is Q + N.
But in real world the performance of these approaches depends on how inputs are formed.

0 Show 1 reply Reply

Stocke777 ★ 25 September 7, 2019 11:00 PM

Great solution, thnx

0 Reply

lmm333 ★ 7 September 7, 2019 10:19 PM

Nice idea!

0 Reply