cravo123 ★ 18 June 15, 2019 11:57 PM 2.4K VIEWS

12

Solution 1, first sort reversely by id and score, and then calculate average for each id, similar to Running Length Encoding.

```
class Solution:
    def highFive(self, items: List[List[int]]) -> List[List[int]]:
        items.sort(reverse=True)

        res = []
        curr = []
        idx = items[0][0]

        for i, val in items:
            if i == idx:
                if len(curr) < 5:
                    curr.append(val)
            else:
                res.append([idx, sum(curr) // len(curr)])
                curr = [val]
                idx = i

        res.append([idx, sum(curr) // len(curr)])

        res = res[::-1]

        return res
```

Solution 2, use priority queue for each id,

```
class Solution:
    def highFive(self, items: List[List[int]]) -> List[List[int]]:
        d = collections.defaultdict(list)

        for idx, val in items:
            heapq.heappush(d[idx], val)

            if len(d[idx]) > 5:
                heapq.heappop(d[idx])

        res = [[i, sum(d[i]) // len(d[i])] for i in sorted(d)]

        return res
```

Comments: 5 Best | Most Votes | Newest to Oldest | Oldest to Newest

Type comment here... (Markdown is supported)

Post

vsharda1 ★ 8 Last Edit: May 6, 2020 12:16 AM

@cravo123 ravo123 Nice solution but you can change the last line in the 2nd solution as we do not need sorted there.

```
res = [[i, sum(d[i]) // 5] for i in d]
```

1 Reply

WIZARDELF ★ 27 June 16, 2019 3:08 AM

which one has better performance?

0 Show 4 replies Reply

Xyzzy123 ★ 51 April 9, 2020 4:21 AM

My approach here:

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

    scores = collections.defaultdict(list)
```