Greedy Python.

alexZhang ★ 18 Last Edit: June 2, 2019 6:08 AM 601 VIEWS

13

For each number, we have two choice: floor or ceil.
So the minimum sum we could get is sum each floored price.
And the maximum sum we could get is sum each ceiled price.
If target is larger or equal than the minimum one and small or equal to the maximum one, we could get such value. Otherwise we need to return a "-1".

If we can get target, we can do floor operation to each value first.
Now we get a minimum value.
To get the target, we need select (target - minimum) numbers of price and do ceil operation on it instead of floor operation so that we can get target.(ceil(p) - floor(p) = 1)
We want to minimum the total change (which is sum of the abs change by doing operation(floor | ceil) to each price).
So we can:

1. Calculate the change made by floor operation and ceil operation for each price.
2. Calculate the reduce in the change by flip the floor operation to ceil operation.
3. Sort them in des order.
4. For the first (target - minimum) changes, we flip floor operation to ceil.(As we gain more reduces to the total changes by doing ceil operation to these prices.)
5. For the rest changes, we select floor operation.
6. Return the sum of all changes.

```
class Solution:
    def minimizeError(self, prices: List[str], target: int) -> str:
        diff = []
        low, high = 0, 0
        for i, p in enumerate(map(float, prices)):
            f, c = math.floor(p), math.ceil(p)
            low, high = low + f, high + c
            fDiff, cDiff = p - f, c - p
            diff.append((fDiff - cDiff, fDiff, cDiff))
        if not low <= target <= high:
            return "-1"
        ceilN = target - low
        diff = sorted(diff, reverse=True)
        return "{:.3f}".format(sum([d[2] for d in diff[:ceilN]]) + sum([d[1] for d in diff[ceilN:]]))
```

Comments: 1 Sort: Best · Most Votes · Newest to Oldest · Oldest to Newest

Type comment here... (Markdown is supported)

Post

nachiyer ★ 7 August 19, 2019 6:23 AM

Awesome and intuitive solution.

Basically, if we floor every value in prices, target - (amount of prices summed if all were floored) is number of elements we need to ceil.
fDiff - cDiff measures how expensive it is to floor compared to ceil the price, and it is our heuristic for which prices we want to ceil/floor. If fDiff - cDiff is big, it means it's expensive to floor, so let's ceil it instead.

Also, instead of putting fDiff - cDiff, if we put cDiff - fDiff then we don't need to reverse the list when we sort diff. cDiff - fDiff would mean that the smaller it is, then it's cheaper to ceil than to diff this price.

2 Reply