< Back | Python - heap of closest bike to each worker

yorkshire ★ 684 Last Edit: June 3, 2019 4:51 AM 6.4K VIEWS

63

For each worker, create a sorted list of distances to each bike. The elements of the list are tuples (distance, worker, bike).
For each worker, add the tuple with the shortest distance to the heap.
Until each worker has a bike, pop the smallest distance from the heap.
If this bike is not used, update the result for this worker, else add the next closest tuple for this worker to the heap.

```
def assignBikes(self, workers, bikes):
    distances = []     # distances[worker] is tuple of (distance, worker, bike) for each bike
    for i, (x, y) in enumerate(workers):
        distances.append([])
        for j, (x_b, y_b) in enumerate(bikes):
            distance = abs(x - x_b) + abs(y - y_b)
            distances[-1].append((distance, i, j))
        distances[-1].sort(reverse = True)  # reverse so we can pop the smallest distance

    result = [None] * len(workers)
    used_bikes = set()
    queue = [distances[i].pop() for i in range(len(workers))]  # smallest distance for each worker
    heapq.heapify(queue)

    while len(used_bikes) < len(workers):
        _, worker, bike = heapq.heappop(queue)
        if bike not in used_bikes:
            result[worker] = bike
            used_bikes.add(bike)
        else:
            heapq.heappush(queue, distances[worker].pop())  # bike used, add next closest bike

    return result
```

Comments: 11 Best Most Votes Newest to Oldest Oldest to Newest

Type comment here... (Markdown is supported)

Post

HarveyW ★ 198 July 8, 2019 3:23 AM

nice code, especially the last while condition.

4 Reply

puppy101puppy ★ 10 July 19, 2019 10:23 AM

Thought this is perfect match problem, then looked at this solution for 10 min and realized the problem is different...
BTW, does anyone know how to solve an alternation of the problem where we want to minimize the [sum of all matched bike to person distances]?

2 Show 2 replies Reply

aunbr ★ 155 June 2, 2019 7:01 PM

O(WBlog(WB) + Wlog(WB)) ?

2 Show 3 replies Reply

zhaolzhang ★ 2 April 5, 2020 1:13 AM

Followed your idea, just made it easier to read.

```
def assignBikes(self, workers: List[List[int]], bikes: List[List[int]]) -> List[int]:
    sorted_work = collections.defaultdict(list)
    for i, worker in enumerate(workers):
        for j, bike in enumerate(bikes):
            bisect.insort(sorted_work[i], (abs(worker[0] - bike[0]) + abs(worker[1] - bike[1]), i, j))

    heap = []
    used = set()
    res = [None] * len(workers)
```

Read More

1 Reply

chyeahhiahaha ★ 1 July 26, 2019 11:50 PM

when updating values to "result", why you use the "while len(used_bikes) < len(workers)" instead of checking the "result" has no None value anymore

0 Show 1 reply Reply

dourojege ★ 293 July 15, 2019 11:55 PM

nice solution!
reconstructing 1 ascending linkedlist from N ascending linkedlists

0 Reply

vijay20 ★ 3 April 22, 2020 10:27 AM

We don't need a heap for this since we are only popping from the heap. We can simply use a sorted list.

0 Reply

merkle_tree ★ 3 March 26, 2020 5:42 AM

Beautiful solution. Is there any way we can optimize space / time here?

0 Reply

soccerlack ★ 0 November 18, 2019 12:03 AM

How are we sure if we choose the pair with the smallest worker index when there are same distances from different workers to a bike?

0 Show 1 reply Reply

gw980531 ★ 7 August 30, 2019 10:22 AM

Any idea why the following solution got TLE? Do they not have similar run time?

```
class Solution(object):
    def assignBikes(self, workers, bikes):
        """
        :type workers: List[List[int]]
        :type bikes: List[List[int]]
        :rtype: List[int]
        """
        distances = []
        for i in range(len(workers)):
```

Read More

0 Show 3 replies Reply