Crack a hack

Cloudy Day

Course:Algorithmic problem solving

Course code: 17ECSE309

By:
Abhijeet Ganapule
01FE15BEC003

1)Introduction: Cloudy day was a problem framed for Hourrank26,and the required knowledge for this problem is line sweep and greedy concepts.

Now coming to the problem, the main idea here is that any town covered by at least two clouds will never be cleared since we're only allowed to remove one cloud. Thus, we can safely ignore such towns and only consider towns covered by at most one cloud. In this simpler problem, a greedy approach will work: simply remove the cloud covering the most people.

The most time-consuming part is computing which towns are covered by at least two clouds. We will solve this using the line sweep technique.

For each cloud i , we will create two events, one at location yi-ri and the other at yi+ri+1. The first event corresponds to the event that i cloud is now active and will cover the upcoming cities (until it becomes inactive again). The second event corresponds to making cloud i inactive.

Similarly, we will create 1 event per town at location xi.

Now, imagine a vertical line sweeping from left to right. Each time this vertical line intercepts an event, we will process that event. When there are multiple events in the same position, we will process cloud events before the town events.

In addition, we will maintain a set containing the indices of all the active clouds. Making a cloud active or inactive correspond to inserting it to or deleting it from the active set.

Each time we encounter an event corresponding to a town,

- If the size of the active set is 0, then it means that the town isn't covered by any cloud and the residents of this city already see the sun.
- If the size of the active set is 1, then a unique cloud covers this town, specifically, the cloud corresponding to the unique index in our set.
- If the size of the active set is >1, then there are multiple clouds covering this town, and thus we can safely ignore it.

Thus for each cloud, we maintain a count of how many people will see the sun if this cloud is removed. We can then greedily choose to remove the cloud with the maximum such count.

The final answer will be the number of people who were already in a sunny town in the first place plus the number of people that will see the sun by removing the cloud from our greedy choice.

Code:

```
from collections import defaultdict


def maximumPeople(towns, cloud_start, cloud_end):
    towns = sorted(towns)
    cloud_start = sorted(cloud_start)
    cloud_end = sorted(cloud_end)

    cloud_start_i = 0
    cloud_end_i = 0
    clouds = set()

    d = defaultdict(int)
    free = 0
    for town_i in range(len(towns)):
        town_x = towns[town_i][0]
        while cloud_start_i < len(cloud_start) and
cloud_start[cloud_start_i][0] <= town_x:
            clouds.add(cloud_start[cloud_start_i][1])
            cloud_start_i += 1
        while cloud_end_i < len(cloud_end) and
cloud_end[cloud_end_i][0] < town_x:
            clouds.remove(cloud_end[cloud_end_i][1])
            cloud_end_i += 1
        if len(clouds) == 1:
```

```
            towns[town_i][2] = list(clouds)[0]
            d[list(clouds)[0]] += towns[town_i][1]
        elif len(clouds) == 0:
            free += towns[town_i][1]

    return max(d.values(), default=0) + free



if __name__ == "__main__":
    n = int(input().strip())
    p = [int(x) for x in input().strip().split()]
    x = [int(x) for x in input().strip().split()]
    towns = [[xi, pi, -1] for xi, pi in zip(x, p)]
    m = int(input().strip())
    y = [int(x) for x in input().strip().split()]
    r = [int(x) for x in input().strip().split()]
    cloud_start = [[y[i]-r[i], i] for i in range(m)]
    cloud_end = [[y[i]+r[i], i] for i in range(m)]
    result = maximumPeople(towns, cloud_start, cloud_end)
    print(result)
```

Explanation:

1. Create dictionary “town” containing the position of the town and population of that town.
2. Similarly dictionary of the clouds starting and ending point.
3. Sort all dictionaries created.

4. Now the depending on the range of clouds,if a town falls under it ,then it's population is not added.But it is appended into another dictionary.
5. If a town is free of cloud,then its added in a variable called free.
6. Now the town with maximum population and an cloud over it is added to the population sum obtained in free variable.

References:

1. https://stackoverflow.com/questions/13704860/zip-lists-in-python
2. https://stackoverflow.com/questions/5900578/how-does-collections-defaultdict-work
3. https://www.geeksforgeeks.org/python-dictionary/
4. https://www.geeksforgeeks.org/sets-in-python/
5. https://www.geeksforgeeks.org/sorted-function-python/