475 Heaters

2018年4月12日 14:19

Winter is coming! Your first job during the contest is to design a standard heater with fixed warm radius to warm all the houses.
Now, you are given positions of houses and heaters on a horizontal line, find out minimum radius of heaters so that all houses could be covered by those heaters.
So, your input will be the positions of houses and heaters seperately, and your expected output will be the minimum radius standard of heaters.
Note:

1. Numbers of houses and heaters you are given are non -negative and will not exceed 25000.
2. Positions of houses and heaters you are given are non -negative and will not exceed 10^9.
3. As long as a house is in the heaters' warm radius range, it can be warmed.
4. All the heaters follow your radius standard and the warm radius will the same.

Example 1:
Input: [1,2,3],[2]
Output: 1
Explanation: The only heater was placed in the position 2, and if we use the radius 1 standard, then all the houses can be warmed.
Example 2:
Input: [1,2,3,4],[1,4]
Output: 1
Explanation: The two heater was placed in the position 1 and 4. We need to use radius 1 standard, then all the houses can be warmed.

来自 <https://leetcode.com/problems/heaters/description/>

Solution for Python3:

```python
class Solution1:
    def findRadius(self, houses, heaters):
        """
        :type houses: List[int]
        :type heaters: List[int]
        :rtype: int
        """
        heaters = sorted(heaters) + [float('inf')]
        p = r = 0
        for x in sorted(houses):
           while heaters[p+1] + heaters[p] <= 2 * x:
               p += 1
           r = max(r, abs(heaters[p] - x))
        return r

class Solution2:
    def findRadius(self, houses, heaters):
        """
        :type houses: List[int]
        :type heaters: List[int]
        :rtype: int
        """
        import bisect
        heaters.sort()
        return max(min(abs(house - heater) for i in [bisect.bisect(heaters, house)] for heater in heaters[i-(i>0):i+1]) for house in houses)
```

Solution for C++:

```cpp
class Solution {
public:
    int findRadius(vector<int>& houses, vector<int>& heaters) {
        sort(heaters.begin(), heaters.end());
        int res = INI_MIN;
        for (int house : houses) {
            auto index = lower_bound(heaters.begin(), heaters.end(), house);
            int dist1 = (index == heaters.begin()) ? INI_MIN : house - *(index - 1);
            int dist2 = (index == heaters.end()) ? INT_MAX : *(index) - house;
            res = max(res, min(dist1, dist2));
        }
        return res;
    }
};
```

Appendix:

Python bisect:二分查找插入模块 import bisect

1) bisect.bisect(T, a) 返回a在有序表T中应该插入的位置
2) bisect.insort(T, a) 将a插入到有序表T中
3) bisect.bisect_left(T,a) 返回a在有序表T中重复值a的左侧位置
4) bisect.bisect_right(T,a)
5) bisect.insort_left(T,a)
6) bisect.insort_right(T,a)