

475. Heaters

♥ Add to List ▾

Description (?tab=Description)

Submission (?tab=Submission)

Solutions (?tab=Solutions)

Total Accepted: **9146** Total Submissions: **30701** Difficulty: **Easy** Contributors: **neelamgehlot** (/neelamgehlot/)

📌 Notes

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 separately, 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 be 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.

Hide Company Tags

Google (/company/google/)

Hide Tags

Binary Search (/tag/binary-search/)

Have you met this question in a real interview? ☐ Yes ☐ No

🗨 Discuss (<https://discuss.leetcode.com/category/606>)

🎲 Pick One (/problems/random-one-question/)

📖 Editorial Solution


C++ ▾



```
1 class Solution {
2     public:
3
4         // two pointer linear solution + sorting O(nlogn)
5         int findRadius(vector<int>& houses, vector<int>& heaters) {
6             sort(heaters.begin(), heaters.end());
7             sort(houses.begin(), houses.end());
8             int i = 0, j = 0, res = 0;
9             while (i < houses.size()) {
10                 while (j < heaters.size() - 1 && abs(heaters[j+1] - houses[i]) <= abs(heaters[j] - houses[i])) {
11                     j++;
12                 }
13                 res = max(res, abs(heaters[j] - houses[i]));
14                 i++;
15             }
16             return res;
17         }
18
19         // My Working solution O(nlogn)
20         // New examples failed this solution
21         // int findRadius(vector<int>& houses, vector<int>& heaters) {
22         //     int res = 0;
23         //     sort(heaters.begin(), heaters.end());
24         //     for(auto house : houses) {
25         //         // search for the closest heater whose position is at least the current house's position
26         //         int j = 0;
27         //         while (j < heaters.size() && heaters[j] < house) {
28             j++;
29         }
30         if (j < heaters.size()) {
31             res = max(res, abs(heaters[j] - house));
32         }
33     }
34     return res;
35 }
```

✉ Send Feedback (<mailto:admin@leetcode.com?subject=Feedback>)

```
27 //      auto it = lower_bound(heaters.begin(), heaters.end(), house);
28 //      int mdist = it == heaters.end() ? abs(*heaters.rbegin() - house) : abs(*it - house);
29 //      if(it != heaters.begin()) {
30 //          --it;
31 //          mdist = min(mdist, abs(*it - house));
32 //      }
33
34 //      res = max(res, mdist);
35 //  }
36 //  return res;
37 // }
```

Custom Testcase ☐
Contribute Testcase 

Run Code

Submit Solution

[Frequently Asked Questions \(/faq/\)](#) | [Terms of Service \(/tos/\)](#)

[Privacy](#)

Copyright © 2017 LeetCode