题目：

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 u
se radius 1 standard, then all the houses can be warmed.
```

1.时间：O（max(N*M, NLOGN + MLOGM）；空间：O（1）　　->>错误 ,缺乏状态记录

```
class Solution {
public:
    int findRadius(vector<int>& houses, vector<int>& heaters) {
        if (houses.empty() || heaters.empty()) return 0;
        std::sort(houses.begin(), houses.end());
        std::sort(heaters.begin(), heaters.end());
        int leftIndex = 0, rightIndex = heaters.size() - 1;
        int res = 0;
        for (int i = 0; i < houses.size(); ++i){

            int dist = std::numeric_limits<int>::max();
            while (leftIndex < heaters.size() && heaters[leftIndex] <= houses[i]){
                leftIndex++;
            }
            if(leftIndex > 0 || heaters[leftIndex] <= houses[i])
                dist = std::min(dist, houses[i] - heaters[leftIndex]);
            while (rightIndex >= leftIndex && heaters[rightIndex] >= houses[i]){
                rightIndex--;
            }
            if(rightIndex < heaters.size() - 1 || heaters[rightIndex] >= houses[i])
```

```
                dist = std::min(dist, heaters[rightIndex] - houses[i]);

            res = std::max(res, dist);

            leftIndex = 0;
            rightIndex = heaters.size() - 1;
        }
        return res;
    }
};
```

2.时间：O（ ）；空间：O（N+M）

```
class Solution {
public:
    int findRadius(vector<int>& houses, vector<int>& heaters) {
        if (houses.empty() || heaters.empty()) return 0;

        std::sort(houses.begin(), houses.end());
        std::sort(heaters.begin(), heaters.end());

        std::vector<int> left(houses.size(), 0);
        std::vector<int> right(houses.size(), 0);
```

```
int leftIndex = 0, rightIndex = heaters.size() - 1;
for (int i = 0; i < houses.size(); ++i){
    int dist = std::numeric_limits<int>::max();
    while (leftIndex < heaters.size() && heaters[leftIndex] <= houses[i]){
        left[i] = leftIndex + 1;    /* 用 0 区分有没有找到 */
        leftIndex++;
    }
    leftIndex = left[i] == 0 ? 0 : left[i] - 1;
    while (rightIndex >= leftIndex && houses[i] <= heaters[rightIndex]){
        right[i] = rightIndex + 1;
        rightIndex--;
    }
    rightIndex = heaters.size() - 1;
}

int res = 0;
for (int i = 0; i < houses.size(); ++i){
    int dist = std::numeric_limits<int>::max();
    if (left[i] != 0){
        //dist = std::min(dist, houses[i] - heaters[left[i] - 1]);
        dist = houses[i] - heaters[left[i] - 1];
    }
```

```
            if (right[i] != 0){
                dist = std::min(dist, heaters[right[i] - 1] - houses[i]);
            }
            res = std::max(res, dist);
        }
        return res;
    }
};
```

3.时间：O（NLOGN）；空间：O（1）

```
class Solution {
public:
    int findRadius(vector<int>& houses, vector<int>& heaters) {
        if (houses.empty() || heaters.empty()) return 0;
        std::sort(houses.begin(), houses.end());
        std::sort(heaters.begin(), heaters.end());
        int res = 0;
        for (int i = 0; i < houses.size(); ++i){
            int index = binarySearch(heaters, houses[i]);
            /*int distOne = 0;
            int distTwo = 0;
            std::cout << heaters[index] << " " << houses[i] << std::endl;*/
            if (heaters[index] < houses[i]){
```

```
                int distOne = houses[i] - heaters[index];
                int distTwo = (index + 1 < heaters.size()) ?
                    (heaters[index + 1] - houses[i]) :
std::numeric_limits<int>::max();
                res = std::max(res, std::min(distOne, distTwo));
            } else if(heaters[index] > houses[i]){
                int distOne = (index - 1 >= 0) ?
                    houses[i] - heaters[index - 1] : std::numeric_limits<int>::max();
                int distTwo = heaters[index] - houses[i];
                res = std::max(res, std::min(distOne, distTwo));
            }
        }
        return res;
    }
private:
    int binarySearch(const std::vector<int>& heaters, int num){
        int lower = 0, upper = heaters.size() - 1;
        while (lower < upper){
            int mid = lower + ((upper - lower) >> 1);
            if (heaters[mid] == num) return mid;
            else if (heaters[mid] < num) lower = mid + 1;
            else   upper = mid - 1;
```

```
            }

            return lower;

        }

};
```