475. Heaters

Winter is coming! During the contest, your first job is to design a standard heater with a fixed warm radius to warm all the houses.

Every house can be warmed, as long as the house is within the heater's warm radius range.

Given the positions of `houses` and `heaters` on a horizontal line, return *the minimum radius standard of heaters so that those heaters could cover all houses.*

Notice that all the `heaters` follow your radius standard, and the warm radius will the same.

Example 1:

```
Input: houses = [1,2,3], heaters = [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: houses = [1,2,3,4], heaters = [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.
```

Example 3:

```
Input: houses = [1,5], heaters = [2]
Output: 3
```

Constraints:

- `1 <= houses.length, heaters.length <= 3 * 10<sup>4</sup>`
- `1 <= houses[i], heaters[i] <= 10<sup>9</sup>`

```python
class Solution:
    def findRadius(self, houses: List[int], heaters: List[int]) -> int:
        if len(houses)==1:
            return abs(min(heaters)-houses[0])
        heaters.sort()
        ans = []
```

```
        for ele in houses:
            ceil = self.getCeil(heaters,ele)
            floor = self.getFloor(heaters,ele)
            if ceil!=-1 and floor!=-1:
                temp = min(ele-floor,ceil-ele)
                ans.append(temp)
            elif ceil==-1:
                ans.append(ele-floor)
            elif floor==-1:
                ans.append(ceil-ele)

        return max(ans)
        # print(ceil,floor)
        # print(ans)

    def getCeil(self,arr,target):
        res = -1
        lo = 0
        hi = len(arr)-1
        while lo<=hi:
            mid = (lo+hi)//2
            if arr[mid]==target:
                return arr[mid]
            elif arr[mid]>target:
                res = arr[mid]
                hi = mid-1
            elif arr[mid]<target:
                lo = mid+1
        return res

    def getFloor(self,arr,target):
        res = -1
        lo = 0
        hi = len(arr)-1
        while lo<=hi:
            mid = (lo+hi)//2
            if arr[mid]==target:
                return arr[mid]
            elif arr[mid]>target:
                hi = mid-1
            elif arr[mid]<=target:
                res = arr[mid]
```

```
                lo = mid+1
        return res
```