Documentatuon for 475. Heaters

1. **Problem Statement**

   Given the positions of houses and heaters on a 1D line, return the minimum standard radius of heaters such that every house is within at least one heater's warm radius.

   - All heaters have the same fixed radius.
   - A house is warmed if it lies within the heater's radius.

   🔢 Constraints:

   - 1 <= houses.length, heaters.length <= 30,000
   - 1 <= houses[i], heaters[i] <= 10^9

2. **Intuition**

   The idea is simple:

   - Every house must be within the range of at least one heater.
   - For each house, find its nearest heater and measure the distance.
   - The maximum of these distances is the minimum radius required to cover all houses.

3. **Key Observations**

   - Sorting both houses and heaters helps in efficient searching.
   - For each house, finding the closest heater is key.
   - Binary search is ideal since the heaters array is sorted.

4. **Approach**

- Sort both houses and heaters arrays.
- Use bisect_left (binary search) to find the insertion point of each house in the heater array.
- Compute the minimum distance to the nearest heater for each house.
- Track the maximum distance across all houses.
- Return this value as the minimum radius required.

5. **Edge Cases**

- House exactly at the heater position → distance = 0
- All heaters to the left/right of all houses → only use nearest end
- Multiple houses at the same position
- Duplicate heater positions

6. **Complexity Analysis**

 Time Complexity:

- Sorting houses: O(N log N)
- Sorting heaters: O(M log M)
- Binary search for each house: O(N log M)

Total: O((N + M) log M)

 Space Complexity:

- O(1) extra space (in-place sorting)
- No extra data structures used beyond input

7. Alternative Approaches

1. Two Pointers Method

- Use two pointers to walk through sorted houses and heaters.
- At each house, move heater pointer forward only if the next heater is closer.
- More cache-friendly and often faster in practice.
- Time: O(N + M) after sorting

2. Brute Force

- For each house, compute distance to every heater.
- Time: O(N * M) → Too slow for large inputs.

8. Test Cases

Houses	Heaters	Output	Explanation
[1, 2, 3]	[2]	1	All houses within radius 1 of heater 2
[1, 2, 3, 4]	[1, 4]	1	House 2 and 3 covered by 1 and 4
[1, 5]	[2]	3	House 5 is 3 units away
[10, 20, 30]	[15]	15	30 is 15 units from heater
[1, 3, 5, 7]	[2, 6]	2	All covered within radius 2

9. Final Thoughts

- Binary search provides a clean and fast solution for locating nearest heaters.
- Always sort both input lists to guarantee correct and efficient processing.
- Edge case handling is critical when all heaters are on one side.

For large input sizes, avoid brute force. This problem is a classic example of how search + sorting can solve seemingly complex problems efficiently.