

Container With Most Water

11. Container With Most Water

Medium

Topics

Companies

Hint

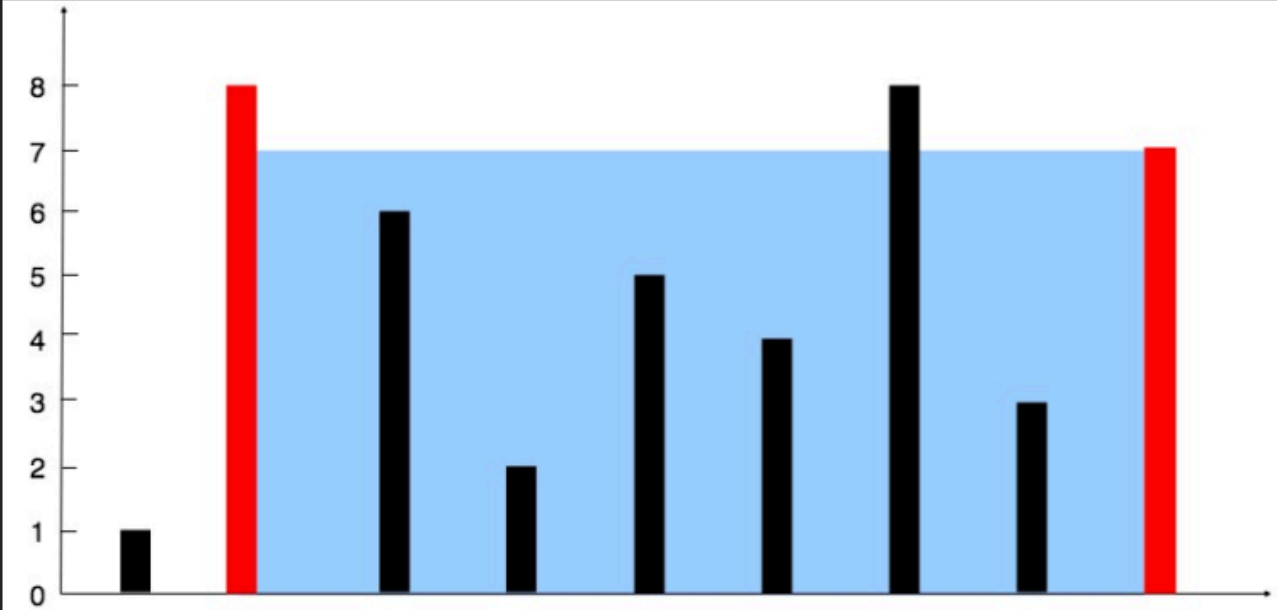
You are given an integer array `height` of length `n`. There are `n` vertical lines drawn such that the two endpoints of the `ith` line are `(i, 0)` and `(i, height[i])`.

Find two lines that together with the x-axis form a container, such that the container contains the most water.

Return *the maximum amount of water a container can store*.

Notice that you may not slant the container.

Example 1:



Input: `height = [1,8,6,2,5,4,8,3,7]`
Output: 49
Explanation: The above vertical lines are represented by array `[1,8,6,2,5,4,8,3,7]`. In this case, the max area of water (blue section) the container can contain is 49.

Example 2:

Input: height = [1,1]

Output: 1

Constraints:

- $n == \text{height.length}$
- $2 \leq n \leq 10^5$
- $0 \leq \text{height}[i] \leq 10^4$

Approach: Brute Force

D&A Problems

Array: Container with most water (LeetCode-11)

height = [1, 8, 6, 2, 5, 4, 8, 3, 7]

Return max amount of water container can store

Ans: { pick 2 line, left & right to form a container }

Max area = $\frac{\text{length} \times \text{height}}{\text{length}}$

Container ka left & right height or else chota wala (1 ki chote height tk hi water store krega)

Think: brute force (all possible containers)

$i \rightarrow 0 \text{ to } (\text{arr.size}() - 1)$
 $j \rightarrow i+1 \text{ to } \text{arr.size}()$ } \Rightarrow all possible containers

picking small height $\rightarrow \text{min_h} = \min(\text{arr}[i], \text{arr}[j])$



area = $\text{min_h} * (j - i)$

max_area = $\max(\text{max_area}, \text{area})$ initialize max = 0

Return max_area

Time Complexity: $O(n^2)$

</> Code

C++   Auto

```
1  class Solution {
2  public:
3      int maxArea(vector<int>& height) {
4          int maxWater = 0;
5          int n = height.size();
6          for(int i=0;i<n-1;i++){
7              for(int j=i+1;j<n;j++){
8                  int h = min(height[i],height[j]); //min height
9                  int w = j-i; // width
10                 int currWater = h*w;
11                 maxWater=max(maxWater,currWater);
12             }
13         }
14         return maxWater;
15     }
16 }
```

Saved

☒ Testcase |  Test Result

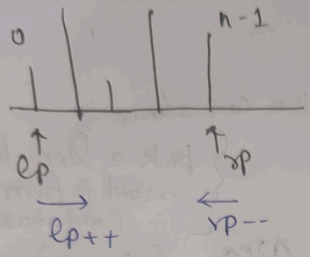
Accepted Runtime: 0 ms

• Case 1

• Case 2

Approach: Optimal

Optimal approach (2 pointer)



approach:

agr left pointer (lp) ka height min h right pointer (rp) se
 $\Rightarrow lp++$

agr rp ka height min h lp k height se $\Rightarrow rp--$

$lp = 0, rp = n-1$

while (lp < rp) {

$w = rp - lp$ // width

$h = \min[ht[lp], ht[rp]]$ // height

currWt = $w * h$

maxWater = $\max(\maxWater, currWt)$

$ht[lp] < ht[rp] ? lp++ : rp--;$

}

because uska reverse krnge
toh area kmesaa decrease
hi hoga since area is
driven by chota height bar

</> Code

C++ Auto

```
1 class Solution {
2 public:
3     int maxArea(vector<int>& height) {
4         int maxWater = 0;
5         int n = height.size();
6         int lp = 0; int rp = n-1;
7         while(lp < rp){
8             int w = rp - lp; //width
9             int h = min(height[lp], height[rp]);
10            int currWater = w * h;
11            maxWater = max(maxWater, currWater);
12            height[lp] < height[rp] ? lp++ : rp--;
13        }
14        return maxWater;
15    }
16 }
```

Saved

☒ Testcase | [> Test Result](#)

Accepted Runtime: 0 ms

• Case 1

• Case 2