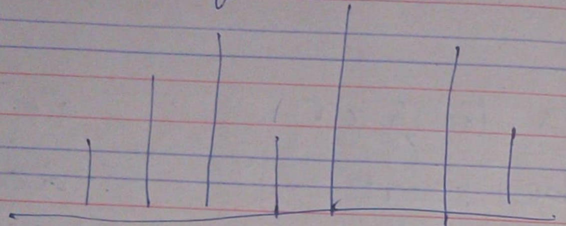
11. Container with Most water

Brute force



height [ar] = 0 1 2 3 4 5 6

```
                 maxCnt = 0
pseudcode :   for (int i = 0 -> n ) {
                  for (int j = i+1 -> n ) {
                      width = j - i
                      height = min(height[i], height[j])
                      maxCnt = max (maxCnt, height * width);
                  }
              }
                 return maxCnt;
```

∴ TC : $O(n^2)$ which is bad time complexity coz height.length = 10^5

so $n = 10^5$ => $O(10^{10})$ -> which will give a TLE.

SC : O(1) constant

optimal solⁿ.

Two - pointer approach ~~this~~ to

This will reduce TC to O(n) which is optimized solⁿ.

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

i ↑ ... ↑ j

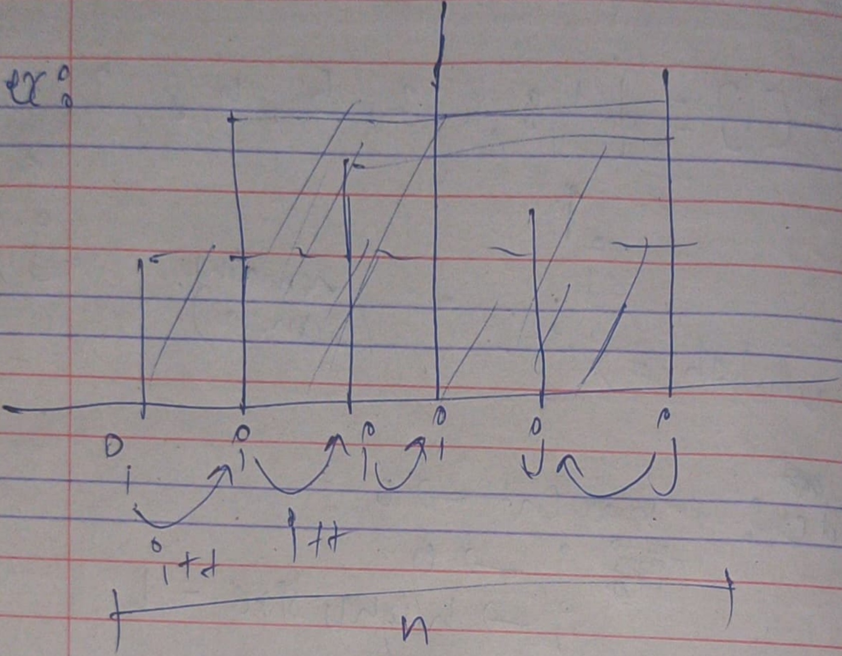
width => j - i (remain same)

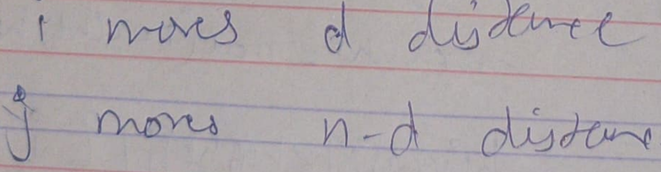
```
pseudcode :   maxCnt -> 0
              i -> 0
              j -> height.size - 1
              while (i < j) {
                  width -> j - i
                  height -> min (height[i], height[j])
                  maxCnt = max (maxCnt, height * width);
                  height[i] < height[j] ? i++ : j--
              }
              return maxCnt ;
```

· The two pointer placed at both end will only move forward (i) and backward (j) according to height of container

· Deciding factor of container will be minimum height of an edge

ex:





i moves d distance

j moves n-d distance

so $n-\cancel{d}+\cancel{d}=\underline{\underline{n}}$

so loop runs n times.

TC: O(n) ; SC: O(1)