//TIME:O(n)

//SPACE:O(1)

class Solution {

public int maxArea(int[] height) {

if(height==null || height.length==0)

return 0;

int max=0; int low=0;

int high=height.length - 1;

while(low<high){

max=Math.max(max,Math.min(height[low],height[high])\*(high-low));

if(height[low]<height[high]){

low++;

}

else{

high--;

}

}

return max;

}

}

//THREE SUM

//SPACE:0\O(N)

//TIME:0(N)

class Solution {

Map<Integer,Integer> hmap = new HashMap<>();

List<List<Integer>> result = new ArrayList<>();

public List<List<Integer>> threeSum(int[] nums) {

if(nums==null ||nums.length==0)

return result;

for(int i=0;i<nums.length;i++)

{

hmap.put(nums[i],i);

}

for(int i=0;i<nums.length;i++)

{

twosum(nums,-nums[i]);

}

return result;

}

public void twosum(int[] nums,int target)

{

for(int i=0;i<nums.length;i++){

int diff=target-nums[i];

if(hmap.containsKey(diff) && i!=hmap.get(diff) && i!=hmap.get(target))

result.add(diff,target);

result.add(hmap.containsValue(i));

}

}

}

//CONTAINER WITH MOST WATER

//TIME:O(N)

//SPACE:O(1)

class Solution {

public int maxArea(int[] height) {

if(height==null || height.length==0)

return 0;

int max=0; int low=0;

int high=height.length - 1;

while(low<high){

max=Math.max(max,Math.min(height[low],height[high])\*(high-low));

if(height[low]<height[high]){

low++;

}

else{

high--;

}

}

return max;

}

}