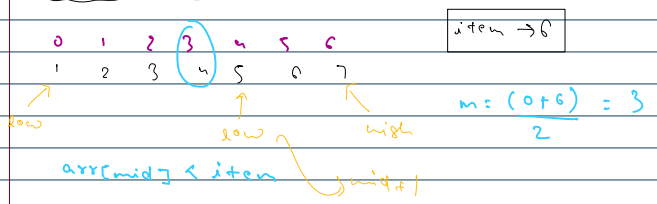
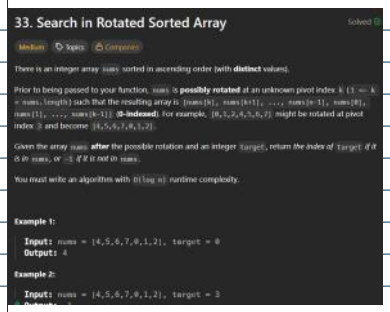
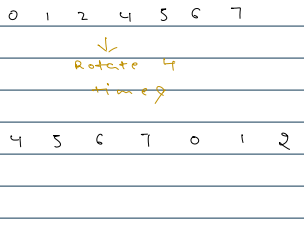
27 July 2025 10:09

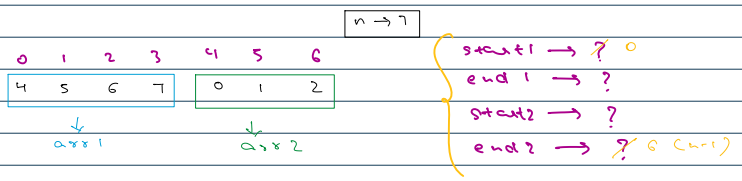
Binary Search

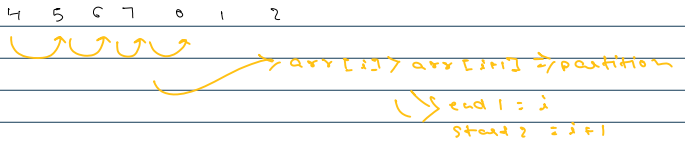


Search in Rotated Sorted Array



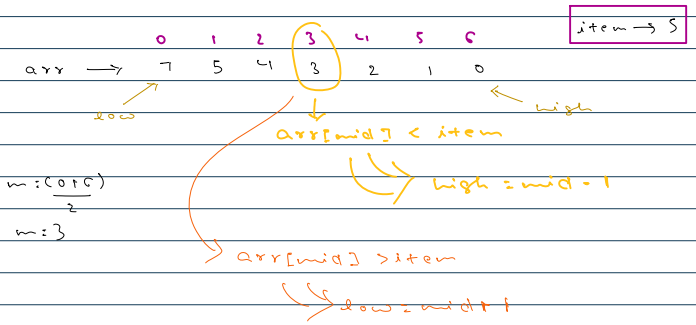







```
public class Main {
    public static void main(String[] args) {

        int[] arr = {2, 3, 4, 5, 7, 8, 11, 13, 13, 16, 17, 18, 22, 24};
        int item = 13;

        int startIndex = 5;
        int endIndex = 8;

        System.out.println(binarySearch(arr, item, startIndex, endIndex));
    }

    public static int binarySearch(int[] arr, int item, int low, int high){

        int n = arr.length;

        while(low<=high){
            int mid = (low+high)/2;
            if(arr[mid] == item){
                return mid;
            }
            else if(arr[mid]>item){
                high = mid - 1;
            }
            else {
                low = mid + 1;
            }
        }
        return -1;
    }
}
```

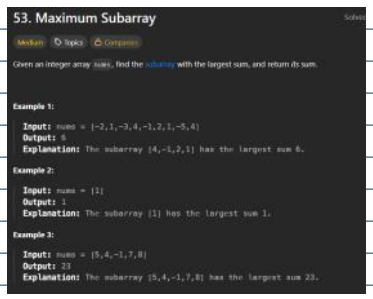
```
class Solution {
    public int search(int[] nums, int target) {

        int n = nums.length;

        int start1 = 0;
        int end1 = 0;

        int start2 = 0;
        int end2 = n-1;

        for(int i=0; i<n-1; i++){
            if(nums[i] > nums[i+1]){
                end1 = i;
                start2 = i+1;
            }
        }

        System.out.println(start1 + " " + end1 + " " + start2 + " " + end2);

        int ans = binarySearch(nums, target, start1, end1);

        if(ans == -1){
            ans = binarySearch(nums, target, start2, end2);
        }

        return ans;
    }

    public static int binarySearch(int[] arr, int item, int low, int high){

        int n = arr.length;

        while(low<=high){
            int mid = (low+high)/2;
            if(arr[mid] == item){
                return mid;
            }
            else if(arr[mid]>item){
                high = mid - 1;
            }
            else {
                low = mid + 1;
            }
        }
        return -1;
    }
}
```




```
public class Main {
    public static void main(String[] args) {

        int[] arr1 = {2, 3, 4, 5, 7, 8, 11, 13, 13, 16, 17, 18, 22, 24};
        int item1 = 13;

        int startIndex1 = 5;
        int endIndex1 = 8;
```

```
        System.out.println(binarySearchAscending(arr1, item1, startIndex1, endIndex1));

        int[] arr2 = {24, 22, 18, 17, 16, 13, 13, 11, 8, 7, 5, 4, 3, 2};
        int item2 = 13;

        int startIndex2 = 0;
        int endIndex2 = 13;

        System.out.println(binarySearchDescending(arr2, item2, startIndex2, endIndex2));

    }

    public static int binarySearchAscending(int[] arr, int item, int low, int high){

        int n = arr.length;

        while(low<=high){
            int mid = (low+high)/2;
            if(arr[mid] == item){
                return mid;
            }
            else if(arr[mid]>item){
                high = mid - 1;
            }
            else {
                low = mid + 1;
            }
        }
        return -1;
    }

    public static int binarySearchDescending(int[] arr, int item, int low, int high){

        int n = arr.length;

        while(low<=high){
            int mid = (low+high)/2;
            if(arr[mid] == item){
                return mid;
            }
            else if(arr[mid]>item){
                low = mid + 1;
            }
            else {
                high = mid - 1;
            }
        }
        return -1;
    }
}
```

Maximum Subarray



53. Maximum Subarray

Medium · Topics · Companies

Given an integer array nums, find the subarray with the largest sum, and return its sum.

Example 1:

Input: nums = [-2,1,-3,4,-1,2,1,-5,4]
Output: 6
Explanation: The subarray [4,-1,2,1] has the largest sum 6.

Example 2:

Input: nums = [1]
Output: 1
Explanation: The subarray [1] has the largest sum 1.

Example 3:

Input: nums = [5,4,-1,7,8]
Output: 23
Explanation: The subarray [5,4,-1,7,8] has the largest sum 23.

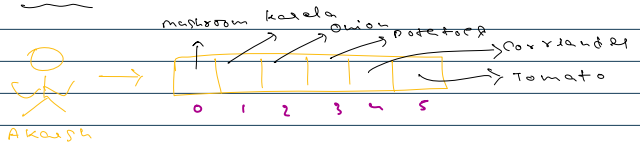
Subarray

It is a continuous part of an array.

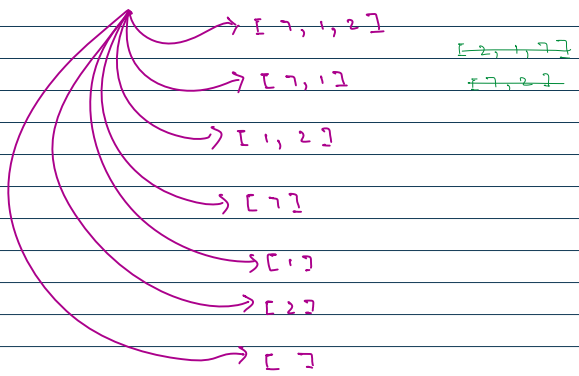
Story

It is a contiguous part of an array.

Story



arr → [7, 1, 2]



0	1	2	3	4	5
2	3	-7	4	-1	9

arr → [2, 3, -7, 4, -1, 9]

i : 0

2 → 2
2 3 → 5
2 3 -7 → -2
2 3 -7 4 → 2
2 3 -7 4 -1 → 1
2 3 -7 4 -1 9 → 10

i : 1

3 → 3
3 -7 → -4
3 -7 4 → 0
3 -7 4 -1 → -1
3 -7 4 -1 9 → 8

i : 2

-7 → -7
-7 4 → -3
-7 4 -1 → -4
-7 4 -1 9 → 5

i : 3

4 → 4
4 -1 → 3
4 -1 9 → 12

i : 4

-1 → -1
-1 9 → 8

i : 5

9 → 9

i : 0

2 → 2
2 3 → 5
2 3 -7 → -2
2 3 -7 4 → 2
2 3 -7 4 -1 → 1
2 3 -7 4 -1 9 → 10

```
public class Main {
    public static void main(String[] args) {

        int[] arr = {2, 3, -7, 4, -1, 9};
        int n = arr.length;

        int sum = 0;
        for(int i=0; i<n; i++){
            sum = sum + arr[i];
            System.out.print(sum + " ");
```

✗ Loop → 0 to n-1

```
        for(int i=0; i<n; i++){
            sum = sum + arr[i];
            System.out.print(sum + " ");
        }

        System.out.println();

        sum = 0;
        for(int i=1; i<n; i++){
            sum = sum + arr[i];
            System.out.print(sum + " ");
        }

        System.out.println();

        sum = 0;
        for(int i=2; i<n; i++){
            sum = sum + arr[i];
            System.out.print(sum + " ");
        }

        System.out.println();

        sum = 0;
        for(int i=3; i<n; i++){
            sum = sum + arr[i];
            System.out.print(sum + " ");
        }

        System.out.println();

        sum = 0;
        for(int i=4; i<n; i++){
            sum = sum + arr[i];
            System.out.print(sum + " ");
        }

        System.out.println();

        sum = 0;
        for(int i=5; i<n; i++){
            sum = sum + arr[i];
            System.out.print(sum + " ");
        }


    }


}
```

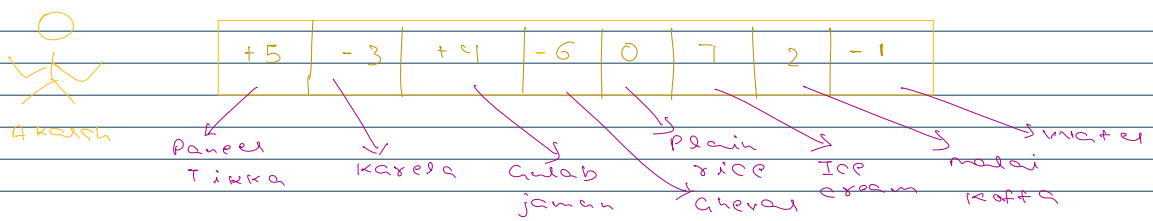
Loop i = 0 to n-1

||

for(int j=0 ; j<n ; j++) {

}

Kadane's Algo



Maximum possible score → ?

↳ [7, 2]

```java
class Solution {
    public int maxSubArray(int[] nums) {
        int n = nums.length;
        int ans = Integer.MIN_VALUE;
        int sum = 0;
        for(int j=0; j<n; j++){
            sum = sum + nums[j];
            ans = Math.max(ans, sum);

            if(sum < 0){
                sum = 0;
            }
        }
        return ans;
    }
}
```

Product of Array Except Self

238. Product of Array Except Self

Solved

Medium · Topics · Companies · Hint

Given an integer array `nums`, return *an array* `answer` *such that* `answer[i]` *is equal to the product of all the elements of* `nums` *except* `nums[i]`.

The product of any prefix or suffix of `nums` is **guaranteed** to fit in a **32-bit** integer.

You must write an algorithm that runs in O(n) time and without using the division operation.

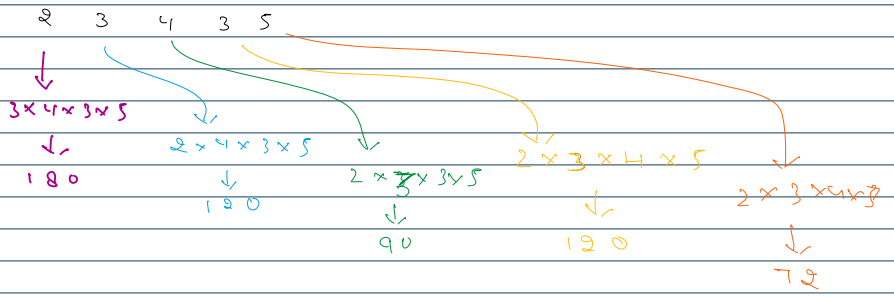
single loop → nested loop

Example 1:

```
Input: nums = [1,2,3,4]
Output: [24,12,8,6]
```

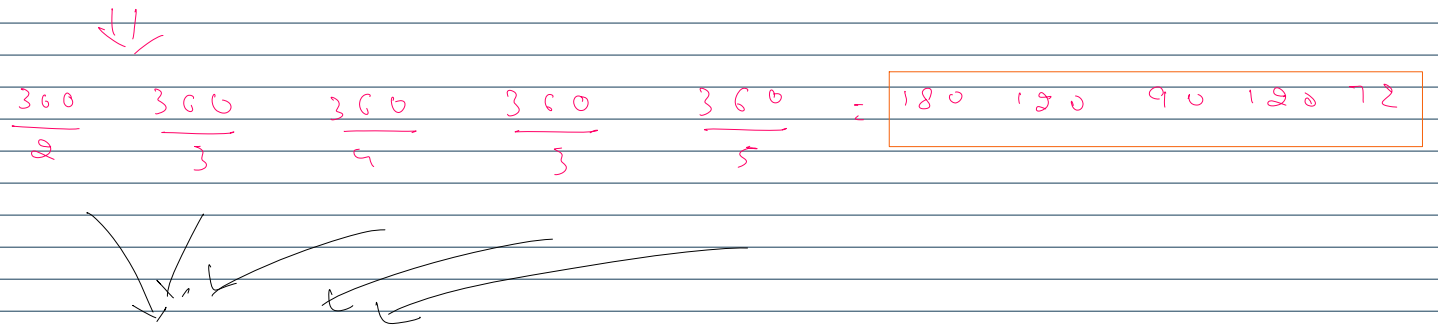
Example 2:

```
Input: nums = [-1,1,0,-3,3]
Output: [0,0,9,0,0]
```



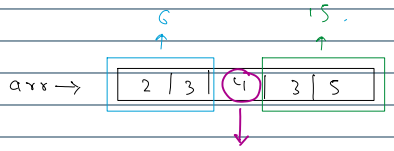
op → 180 120 90 120 72

2 3 4 3 5 ⇒ Product = 360



Product
arr[i]

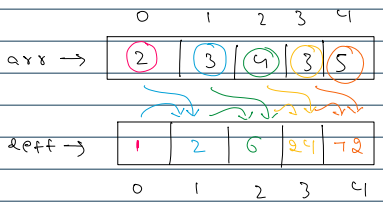
```
public class Main {

    public static void main(String[] args) {
        int[] arr = {2, 3, 4, 3, 5};
        // System.out.println(product(arr));

        int[] nums = productOfArrayExceptSelf(arr);

        for(int i=0; i<nums.length; i++){
            System.out.print(nums[i]+" ");
        }

    }

    public static int product(int[] arr){
        int prod = 1;
        int n = arr.length;
        for(int i=0; i<n; i++){
            prod = prod*arr[i];
        }
        return prod;
    }

    public static int[] productOfArrayExceptSelf(int[] arr){
        int prod = product(arr);
        int n = arr.length;
        int[] op = new int[n];

        for(int i=0;i<n;i++){
            op[i] = prod/arr[i];
        }

        return op;
    }


}
```

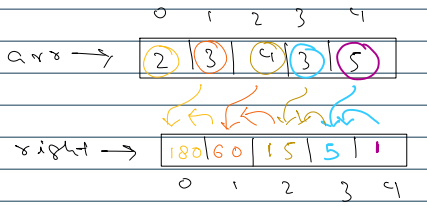


op[2] = product of left array × product of right array

= 6 × 15

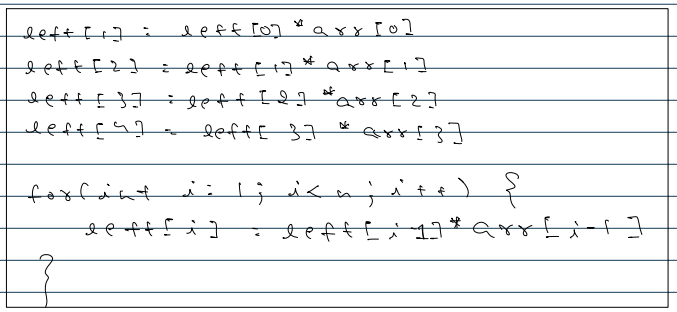
= 90

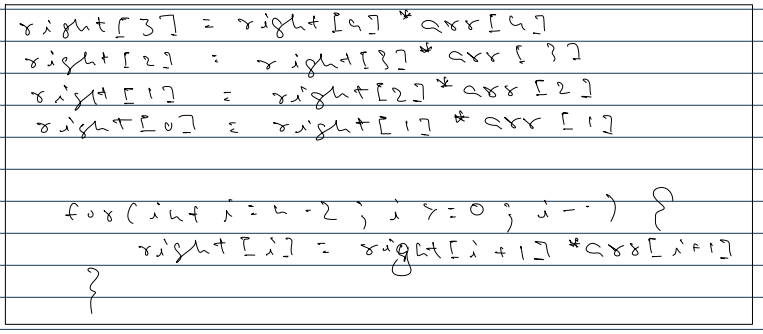


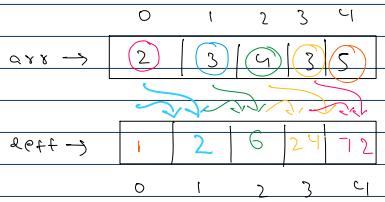


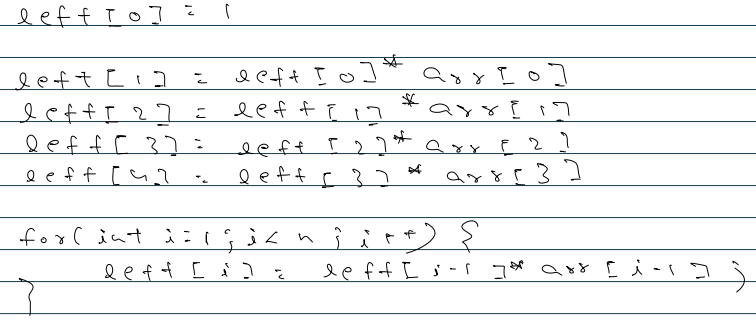
left[0] = 1 ×

right[n-1] = 1



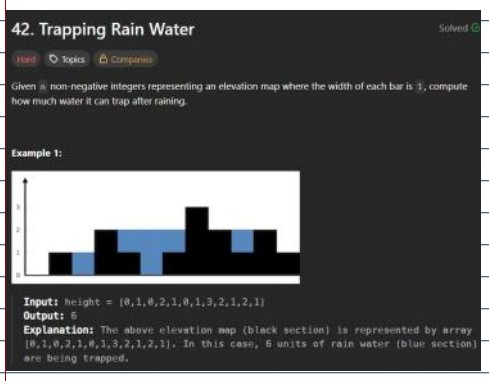


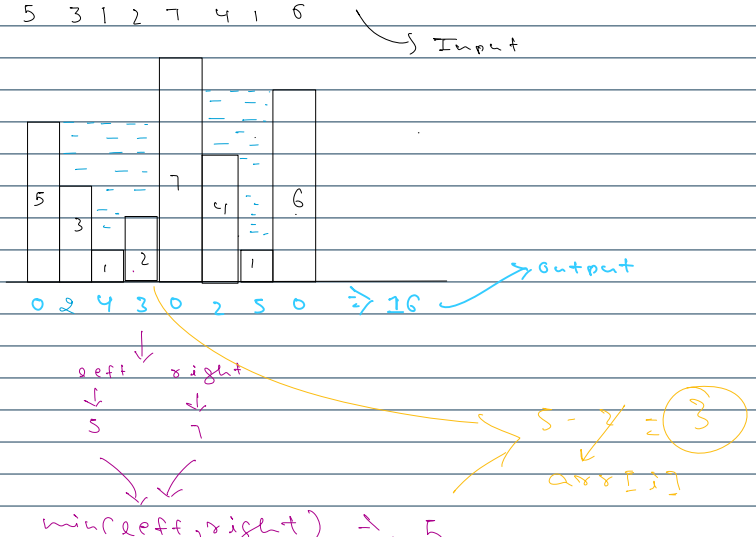


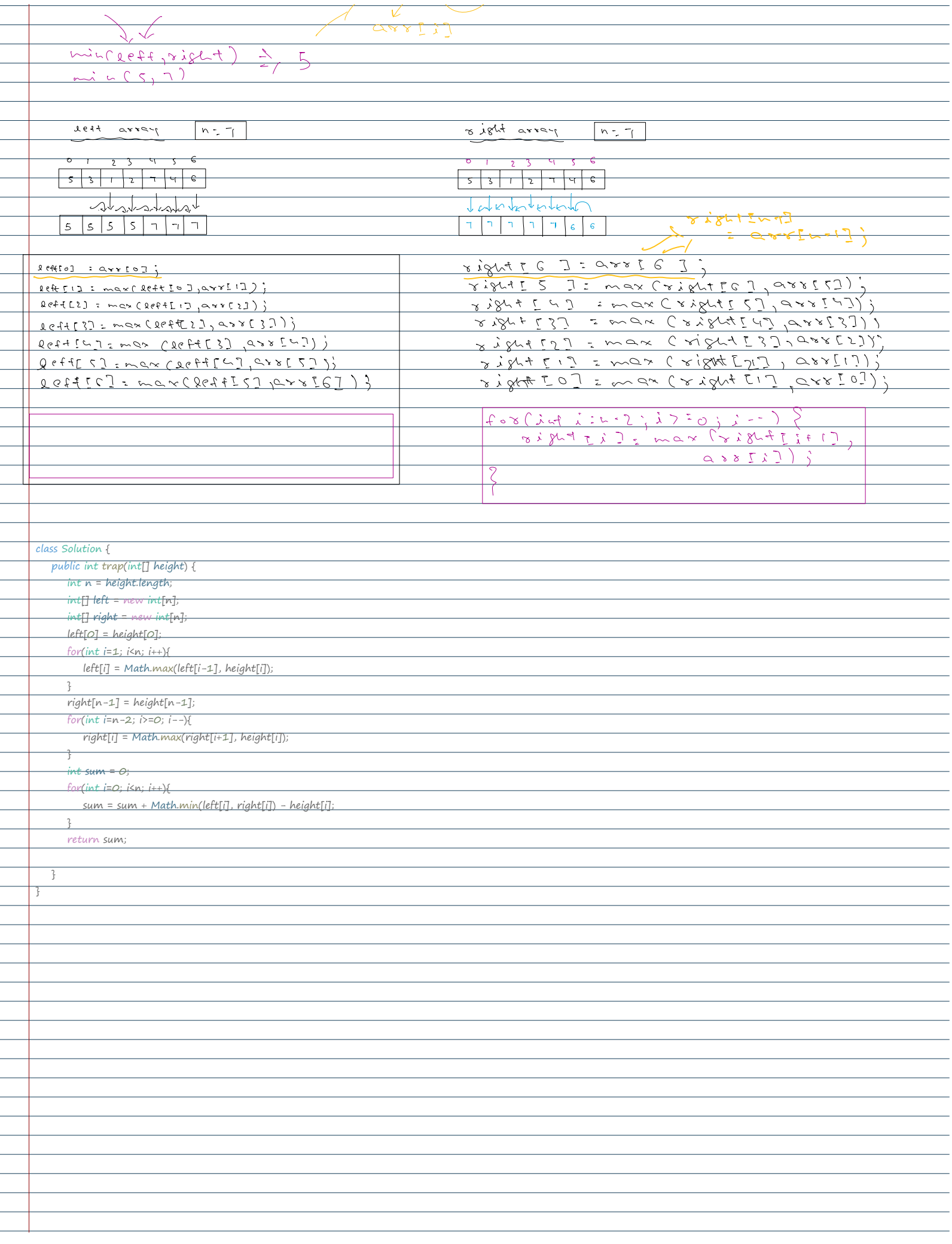


```
class Solution {
    public int[] productExceptSelf(int[] nums) {
        int n = nums.length;
        int[] left = new int[n];
        int[] right = new int[n];
        left[0] = 1;
        for(int i=1; i<n; i++){
            left[i] = left[i-1] * nums[i-1];
        }
        right[n-1] = 1;
        for(int i=n-2; i>=0; i--){
            right[i] = right[i+1] * nums[i+1];
        }
        for(int i=0; i<n; i++){
            nums[i] = left[i] * right[i];
        }
        return nums;

    }
}
```

Trapping Rain Water








```
class Solution {
    public int trap(int[] height) {
        int n = height.length;
        int[] left = new int[n];
        int[] right = new int[n];
        left[0] = height[0];
        for(int i=1; i<n; i++){
            left[i] = Math.max(left[i-1], height[i]);
        }
        right[n-1] = height[n-1];
        for(int i=n-2; i>=0; i--){
            right[i] = Math.max(right[i+1], height[i]);
        }
        int sum = 0;
        for(int i=0; i<n; i++){
            sum = sum + Math.min(left[i], right[i]) - height[i];
        }
        return sum;

    }
}
```