

## Ranjith R 22IT085 Day 1 DSA Practice

### 1) Maximum Subarray Sum – Kadane's Algorithm:

Given an array arr[], the task is to find the subarray that has the maximum sum and return its sum.

Input: arr[] = {2, 3, -8, 7, -1, 2, 3}

Output: 11

Explanation: The subarray {7, -1, 2, 3} has the largest sum 11.

Input: arr[] = {-2, -4}

Output: -2

Explanation: The subarray {-2} has the largest sum -2.

Input: arr[] = {5, 4, 1, 7, 8}

Output: 25

### Solution:

#### Kadane's Algorithm:

```
package JavaPractice;

import java.util.*;

public class Array {

    public static int maxSubArray(int[] arr) {

        int max=arr[0];

        int res=arr[0];

        for(int i=1;i<arr.length;i++) {

            max=Math.max(arr[i],max+arr[i]);

            res=Math.max(res,max);

        }

        return res;
    }
}
```

```

    }

    public static void main(String[] args) {

        int[] arr={-2, -4};

        System.out.println(maxSubArray(arr));

        int[] arr1= {2, 3, -8, 7, -1, 2, 3};

        System.out.println(maxSubArray(arr1));

    }

}

```

Time Complexity:  $O(n)$

## Output:

The screenshot shows the Eclipse IDE with the following components:

- Package Explorer:** Shows a project named 'JavaPractice' with a source folder 'src' containing 'Array.java'.
- Editor:** Displays the code for 'Array.java'. The code defines a class 'Array' with a method 'maxSubArray' and a 'main' method. The 'main' method calls 'maxSubArray' on two arrays: {-2, -4} and {2, 3, -8, 7, -1, 2, 3}.
- Console:** Shows the output of the program, which is '-2' followed by '11' on separate lines.

```

1 package JavaPractice;
2 import java.util.*;
3 public class Array {
4     public static int maxSubArray(int[] arr) {
5         int max=arr[0];
6         int res=arr[0];
7         for(int i=1;i<arr.length;i++) {
8             max=Math.max(arr[i],max+arr[i]);
9             res=Math.max(res,max);
10        }
11        return res;
12    }
13    public static void main(String[] args) {
14        int[] arr={-2, -4};
15        System.out.println(maxSubArray(arr));
16        int[] arr1= {2, 3, -8, 7, -1, 2, 3};
17        System.out.println(maxSubArray(arr1));
18    }
19 }
20

```

Console Output:

```

<terminated> Array [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (9 Nov 2024, 9:18:13 am – 9:18:14 am) [pid: 6488]
-2
11

```

## 2)Maximum Product Subarray

Given an integer array, the task is to find the maximum product of any subarray.

Input: arr[] = {-2, 6, -3, -10, 0, 2}

Output: 180

Explanation: The subarray with maximum product is {6, -3, -10} with product = 6 \* (-3) \* (-10) = 180

Input: arr[] = {-1, -3, -10, 0, 60}

Output: 60

Explanation: The subarray with maximum product is {60}.

### Solution:

```
package JavaPractice;

import java.util.*;

public class Array {

    public static int maxSubArray(int[] nums) {
        int max=nums[0];
        int min=nums[0];
        int res=nums[0];
        for(int i=1;i<nums.length;i++){
            int temp=max(nums[i],nums[i]*max,nums[i]*min);
            min=min(nums[i],nums[i]*max,nums[i]*min);
            max=temp;
            res=Math.max(res,max);
        }
        return res;
    }

    public static int max(int a,int b,int c){
        return Math.max(a,Math.max(b,c));
    }

    public static int min(int a,int b,int c){
```

```

        return Math.min(a,Math.min(b,c));
    }

    public static void main(String[] args) {

        int[] arr= {-1, -3, -10, 0, 60};

        System.out.println(maxSubArray(arr));

        int[] arr1= {-2, 6, -3, -10, 0, 2};

        System.out.println(maxSubArray(arr1));

    }

}

```

Time Complexity:O(n)

## OUTPUT:

The screenshot shows an IDE with a project named 'JavaPractice'. The 'src' folder contains 'Array.java'. The code in 'Array.java' is as follows:

```

1 package JavaPractice;
2 import java.util.*;
3 public class Array {
4     public static int maxSubArray(int[] nums) {
5         int max=nums[0];
6         int min=nums[0];
7         int res=nums[0];
8         for(int i=1;i<nums.length;i++){
9             int temp=max(nums[i],nums[i]*max,nums[i]*min);
10            min=min(nums[i],nums[i]*max,nums[i]*min);
11            max=temp;
12            res=Math.max(res,max);
13        }
14        return res;
15    }
16    public static int max(int a,int b,int c){
17        return Math.max(a,Math.max(b,c));
18    }
19    public static int min(int a,int b,int c){
20        return Math.min(a,Math.min(b,c));
21    }
22    public static void main(String[] args) {
23        int[] arr= {-1, -3, -10, 0, 60};
24        System.out.println(maxSubArray(arr));
25        int[] arr1= {-2, 6, -3, -10, 0, 2};
26        System.out.println(maxSubArray(arr1));
27    }
28 }
29

```

The output window at the bottom shows the following output:

```

60
180

```

### 3)Search in a sorted and rotated Array

Given a sorted and rotated array arr[] of n distinct elements, the task is to find the index of given

key in the array. If the key is not present in the array, return -1.

Input : arr[] = {4, 5, 6, 7, 0, 1, 2}, key = 0

Output : 4

Input : arr[] = { 4, 5, 6, 7, 0, 1, 2 }, key = 3

Output : -1

Input : arr[] = {50, 10, 20, 30, 40}, key = 10

Output : 1

#### Solution:

```
package JavaPractice;

import java.util.*;

public class Array {

    public static int search(int[] nums, int target) {

        int low=0;

        int high=nums.length-1;

        while(low<=high){

            int mid=(low+high)/2;

            if(nums[mid]==target){

                return mid;

            }

            if(nums[low]<=nums[mid]){

                if(target<nums[mid] && target>=nums[low]){

                    high=mid-1;

                }

            }

            else{

                low=mid+1;

            }

        }

        return -1;

    }

}
```

```

        }
        else{
            low=mid+1;
        }
    }
    else{
        if(target<=nums[high] &&  nums[mid]<target){
            low=mid+1;
        }
        else{
            high=mid-1;
        }
    }
}
return -1;
}

public static void main(String[] args) {
    int[] arr= {4, 5, 6, 7, 0, 1, 2};
    System.out.println(search(arr,0));
    int[] arr1={ 4, 5, 6, 7, 0, 1, 2};
    System.out.println(search(arr1,3));
}
}

```

Time Complexity: $O(\log n)$

**OUTPUT:**

```
4= public static int search(int[] nums, int target) {
5   int low=0;
6   int high=nums.length-1;
7   while(low<=high){
8       int mid=(low+high)/2;
9       if(nums[mid]==target){
10          return mid;
11      }
12      if(nums[low]<=nums[mid]){
13          if(target<nums[mid] && target>=nums[low]){
14              high=mid-1;
15          }
16          else{
17              low=mid+1;
18          }
19      }
20      else{
21          if(target<=nums[high] && nums[mid]<target){
22              low=mid+1;
23          }
24          else{
25              high=mid-1;
26          }
27      }
28  }
29  return -1;
30 }
31= public static void main(String[] args) {
32     int[] arr= {4, 5, 6, 7, 0, 1, 2};
33     System.out.println(search(arr,0));
34     int[] arr1={ 4, 5, 6, 7, 0, 1, 2};
35     System.out.println(search(arr1,3));
36 }
37 }
38 }

<terminated> Array [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (9 Nov 2024, 11:02:29 am - 11:02:29 am) [pid: 16600]
4
-1
```

## 4)Container With Most Water

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

Input: arr = [1, 5, 4, 3]

Output: 6

Explanation:

5 and 3 are distance 2 apart. So the size of the base = 2.

Height of container =  $\min(5, 3) = 3$ . So total area =  $3 * 2 = 6$

Input: arr = [3, 1, 2, 4, 5]

Output: 12

Explanation:

5 and 3 are distance 4 apart. So the size of the base = 4.

Height of container =  $\min(5, 3) = 3$ . So total area =  $4 * 3 = 12$

### Solution:

```
package JavaPractice;

import java.util.*;

public class Array {

    public static int maxArea(int[] height) {
        int ans=0;
        int first=0;
        int last=height.length-1;
        while(first<last){
            int a=Math.min(height[first],height[last]);
            int area=a*(last-first);
            ans=Math.max(area,ans);
            if(height[first]<=height[last]){
                first++;
            }
            else{
                last--;
            }
        }
    }
}
```



```

        return ans;
    }

    public static void main(String[] args) {

        int[] arr= {1, 5, 4, 3};

        System.out.println(maxArea(arr));

        int[] arr1={ 4, 5, 6, 7, 0, 1, 2};

        System.out.println(maxArea(arr1));

    }
}

```

Time Complexity:  $O(n)$

## OUTPUT

The screenshot shows the Eclipse IDE with the following code in 'Array.java':

```

1 package JavaPractice;
2 import java.util.*;
3 public class Array {
4     public static int maxArea(int[] height) {
5         int ans=0;
6         int first=0;
7         int last=height.length-1;
8         while(first<last){
9
10             int a=Math.min(height[first],height[last]);
11             int area=a*(last-first);
12             ans=Math.max(area,ans);
13
14             if(height[first]<=height[last]){
15                 first++;
16             }
17             else{
18                 last--;
19             }
20         }
21         return ans;
22     }
23     public static void main(String[] args) {
24         int[] arr= {1, 5, 4, 3};
25         System.out.println(maxArea(arr));
26         int[] arr1={ 4, 5, 6, 7, 0, 1, 2};
27         System.out.println(maxArea(arr1));
28     }
29 }
30

```

The console output at the bottom shows:

```

6
12

```

## 5)Find the Factorial of a large number

Input: 100

Output:

933262154439441526816992388562667004907159682643816214685929638952175999932299  
156089414639761565182862536979208272237582511852109168640000000000000000000000  
00

Input: 50

Output: 30414093201713378043612608166064768844377641568960512000000000000

## SOLUTION

```
package JavaPractice;

import java.math.BigInteger;
import java.util.*;

public class Array {

    public static BigInteger Fact(int n) {

        BigInteger ans=BigInteger.valueOf(1);

        for(int i=2;i<=n;i++) {

            ans=ans.multiply(BigInteger.valueOf(i));

        }

        return ans;

    }

    public static void main(String[] args) {

        System.out.println(Fact(50));

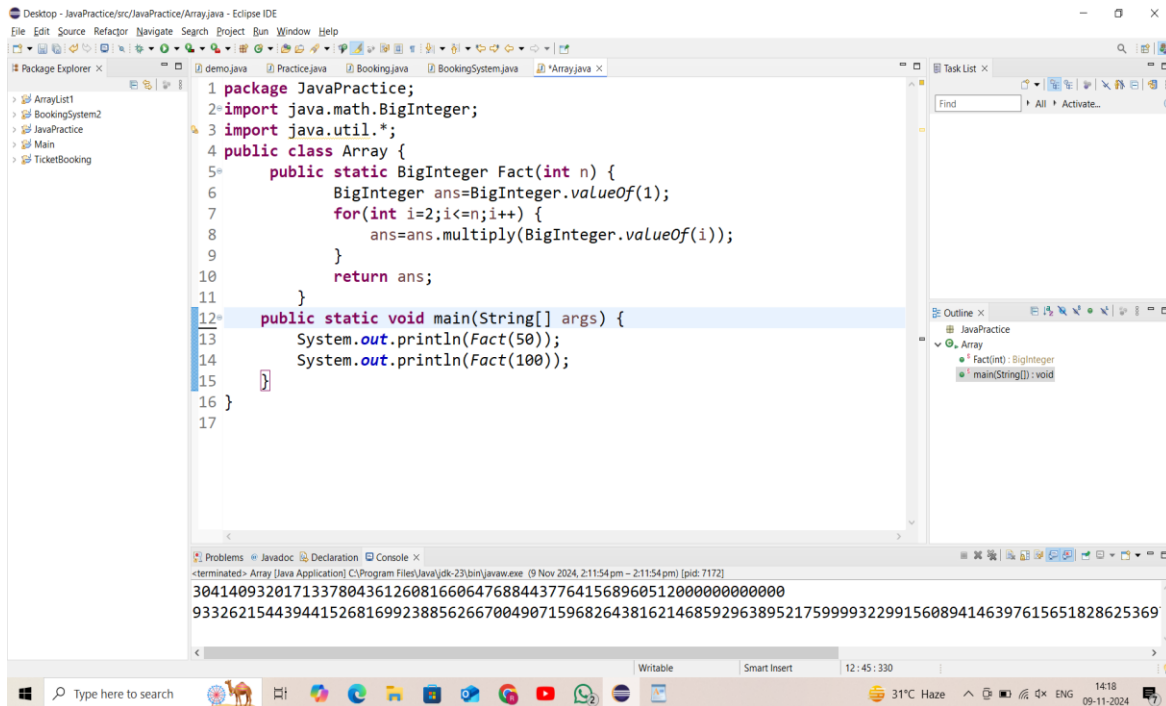
        System.out.println(Fact(100));

    }

}
```

Time Complexity:O(n)

## OUTPUT:



The screenshot shows the Eclipse IDE with a Java project named 'JavaPractice'. The main editor displays the following code:

```
1 package JavaPractice;
2 import java.math.BigInteger;
3 import java.util.*;
4 public class Array {
5     public static BigInteger Fact(int n) {
6         BigInteger ans=BigInteger.valueOf(1);
7         for(int i=2;i<=n;i++) {
8             ans=ans.multiply(BigInteger.valueOf(i));
9         }
10        return ans;
11    }
12    public static void main(String[] args) {
13        System.out.println(Fact(50));
14        System.out.println(Fact(100));
15    }
16 }
17
```

The console output at the bottom shows the execution results:

```
<terminated> Array [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (9 Nov 2024, 2:11:54 pm - 2:11:54 pm) [pid: 7172]
30414093201713378043612608166064768844377641568960512000000000000000
9332621544394415268169923885626670049071596826438162146859296389521759999322991560894146397615651828625369
```

## 6)Trapping Rainwater Problem

states that given an array of n non-negative integers `arr[]` representing an elevation map where the width of each bar is 1, compute how much water it can trap after rain.

Input: `arr[] = {3, 0, 1, 0, 4, 0, 2}`

Output: 10

Explanation: The expected rainwater to be trapped is shown in the above image.

Input: `arr[] = {3, 0, 2, 0, 4}`

Output: 7

Explanation: We trap  $0 + 3 + 1 + 3 + 0 = 7$  units.

Input: arr[] = {1, 2, 3, 4}

Output: 0

Explanation : We cannot trap water as there is no height bound on both sides

Input: arr[] = {10, 9, 0, 5}

Output: 5

Explanation : We trap  $0 + 0 + 5 + 0 = 5$

## SOLUTION:

```
package JavaPractice;

import java.util.*;

public class Array {

    public static int trap(int[] h) {

        int start = 0;

        int end = h.length - 1;

        int maxLeft = h[start];

        int maxRight = h[end];

        int w = 0;

        while (start < end) {

            if (maxLeft < maxRight) {

                start++;

                maxLeft = Math.max(maxLeft, h[start]);

                w += maxLeft - h[start];

            } else {

                end--;

            }

        }

        return w;

    }

}
```

```

        maxRight = Math.max(maxRight, h[end]);

        w += maxRight - h[end];
    }
}

return w;
}

public static void main(String[] args) {
    System.out.println(trap(new int[]{3, 0, 1, 0, 4, 0, 2}));
    System.out.println(trap(new int[]{1, 2, 3, 4}));
}
}

```

Time Complexity:O(n)

**OUTPUT:**

```
1 package JavaPractice;
2 import java.math.BigInteger;
3
4 public class Array {
5     public static int trap(int[] h) {
6         int start = 0;
7         int end = h.length - 1;
8         int maxLeft = h[start];
9         int maxRight = h[end];
10        int w = 0;
11        while (start < end) {
12            if (maxLeft < maxRight) {
13                start++;
14                maxLeft = Math.max(maxLeft, h[start]);
15                w += maxLeft - h[start];
16            } else {
17                end--;
18                maxRight = Math.max(maxRight, h[end]);
19                w += maxRight - h[end];
20            }
21        }
22        return w;
23    }
24    public static void main(String[] args) {
25        System.out.println(trap(new int[]{3, 0, 1, 0, 4, 0, 2}));
26        System.out.println(trap(new int[]{1, 2, 3, 4}));
27    }
28 }
```

Problems Javadoc Declaration Console X

<terminated> Array [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (9 Nov 2024, 9:38:09 pm – 9:38:10 pm) [pid: 8660]

10  
0

7)

## Chocolate Distribution Problem

Given an array `arr[]` of  $n$  integers where `arr[i]` represents the number of chocolates in  $i$ th packet. Each packet can have a variable number of chocolates. There are  $m$  students, the task is to distribute chocolate packets such that: Each student gets exactly one packet.

The difference between the maximum and minimum number of chocolates in the packets given to the students is minimized.

Input: `arr[] = {7, 3, 2, 4, 9, 12, 56}`,  $m = 3$

Output: 2

Explanation: If we distribute chocolate packets `{3, 2, 4}`, we will get the minimum difference, that is 2.

Input: `arr[] = {7, 3, 2, 4, 9, 12, 56}`,  $m = 5$

Output: 7

Explanation: If we distribute chocolate packets {3, 2, 4, 9, 7}, we will get the minimum difference, that is  $9 - 2 = 7$ .

**Solution:**

```
package JavaPractice;

import java.util.*;

public class Array {

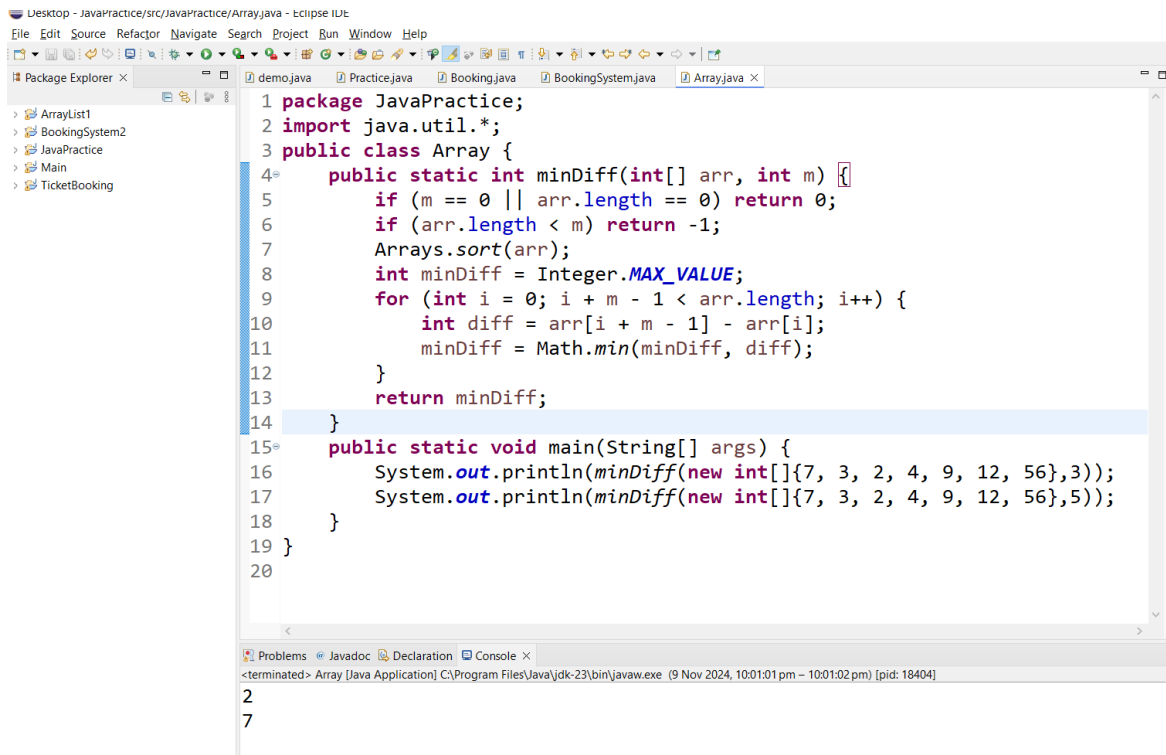
    public static int minDiff(int[] arr, int m) {
        if (m == 0 || arr.length == 0) return 0;
        if (arr.length < m) return -1;
        Arrays.sort(arr);
        int minDiff = Integer.MAX_VALUE;
        for (int i = 0; i + m - 1 < arr.length; i++) {
            int diff = arr[i + m - 1] - arr[i];
            minDiff = Math.min(minDiff, diff);
        }
        return minDiff;
    }

    public static void main(String[] args) {
        System.out.println(minDiff(new int[]{7, 3, 2, 4, 9, 12, 56},3));
        System.out.println(minDiff(new int[]{7, 3, 2, 4, 9, 12, 56},5));
    }
}
```

```
}
```

Time Complexity:  $O(n \log n)$

## OUTPUT:



```
1 package JavaPractice;
2 import java.util.*;
3 public class Array {
4     public static int minDiff(int[] arr, int m) {
5         if (m == 0 || arr.length == 0) return 0;
6         if (arr.length < m) return -1;
7         Arrays.sort(arr);
8         int minDiff = Integer.MAX_VALUE;
9         for (int i = 0; i + m - 1 < arr.length; i++) {
10             int diff = arr[i + m - 1] - arr[i];
11             minDiff = Math.min(minDiff, diff);
12         }
13         return minDiff;
14     }
15     public static void main(String[] args) {
16         System.out.println(minDiff(new int[]{7, 3, 2, 4, 9, 12, 56},3));
17         System.out.println(minDiff(new int[]{7, 3, 2, 4, 9, 12, 56},5));
18     }
19 }
20
```

2  
7

## 8) Merge Overlapping Intervals

Given an array of time intervals where  $arr[i] = [start_i, end_i]$ , the task is to merge all the overlapping intervals into one and output the result which should have only mutually exclusive intervals.

Input:  $arr[] = [[1, 3], [2, 4], [6, 8], [9, 10]]$

Output:  $[[1, 4], [6, 8], [9, 10]]$

Explanation: In the given intervals, we have only two overlapping intervals  $[1, 3]$  and  $[2, 4]$ .



Therefore, we will merge these two and return [[1, 4]], [6, 8], [9, 10]].

Input: arr[] = [[7, 8], [1, 5], [2, 4], [4, 6]]

Output: [[1, 6], [7, 8]]

Explanation: We will merge the overlapping intervals [[1, 5], [2, 4], [4, 6]] into a single interval [1, 6]

### SOLUTION:

```
package JavaPractice;

import java.util.*;

public class Array {

    public int[][] merge(int[][] intervals) {

        Arrays.sort(intervals, (a, b) -> Integer.compare(a[0],
b[0]));

        List<int[]> res = new ArrayList<>();

        int[] p = intervals[0];

        for (int i = 1; i < intervals.length; i++) {

            int[] cur = intervals[i];

            if (cur[0] <= p[1]) {

                p[1] = Math.max(p[1], cur[1]);

            } else {

                res.add(p);

                p = cur;

            }

        }

        res.add(p);

        return res.toArray(new int[res.size()][]);

    }

}
```

```

    }

    public static void main(String[] args) {
        Array arr = new Array();

        int[][] intervals = {{1, 3}, {2, 4}, {6, 8}, {9, 10}};

        int[][] result = arr.merge(intervals);

        System.out.println("Merged intervals:");

        for (int[] interval : result) {
            System.out.println(Arrays.toString(interval));
        }
    }
}

```

**Time Complexity:**  $O(n \log n)$

**OUTPUT:**

```

3 public class Array {
4
5     public int[][] merge(int[][] intervals) {
6         Arrays.sort(intervals, (a, b) -> Integer.compare(a[0], b[0]));
7         List<int[]> res = new ArrayList<>();
8         int[] p = intervals[0];
9         for (int i = 1; i < intervals.length; i++) {
10             int[] cur = intervals[i];
11             if (cur[0] <= p[1]) {
12                 p[1] = Math.max(p[1], cur[1]);
13             } else {
14                 res.add(p);
15                 p = cur;
16             }
17         }
18         res.add(p);
19         return res.toArray(new int[res.size()][]);
20     }
21
22     public static void main(String[] args) {
23         Array arr = new Array();
24         int[][] intervals = {{1, 3}, {2, 4}, {6, 8}, {9, 10}};
25         int[][] result = arr.merge(intervals);
26         System.out.println("Merged intervals:");
27         for (int[] interval : result) {
28             System.out.println(Arrays.toString(interval));
29         }
30     }
31 }
32

```

Problems Javadoc Declaration Console

<terminated> Array [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (9 Nov 2024, 10:19:11 pm - 10:19:12 pm) [pid: 6636]

```

[1, 4]
[6, 8]
[9, 10]

```

## 9. A Boolean Matrix Question

Given a boolean matrix  $mat[M][N]$  of size  $M \times N$ , modify it such that if a matrix cell  $mat[i][j]$  is 1 (or true) then make all the cells of  $i$ th row and  $j$ th column as 1.

Input:  $\{\{1, 0\}, \{0, 0\}\}$

Output:  $\{\{1, 1\}, \{1, 0\}\}$

Input:  $\{\{0, 0, 0\}, \{0, 0, 1\}\}$

Output:  $\{\{0, 0, 1\}, \{1, 1, 1\}\}$

Input:  $\{\{1, 0, 0, 1\},$

$\{0, 0, 1, 0\},$

$\{0, 0, 0, 0\}\}$

Output:  $\{\{1, 1, 1, 1\},$

$\{1, 1, 1, 1\},$

{1, 0, 1, 1}}

## SOLUTION:

```
package JavaPractice;

import java.util.*;

public class Array {

    public static void modifyMatrix(int[][] arr) {

        int row = arr.length;

        int col = arr[0].length;

        int[] rowFlags = new int[row];

        int[] colFlags = new int[col];

        for (int i = 0; i < row; i++) {

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

                if (arr[i][j] == 1) {

                    rowFlags[i] = 1;

                    colFlags[j] = 1;

                }

            }

        }

        for (int i = 0; i < row; i++) {

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

                if (rowFlags[i] == 1 || colFlags[j] == 1) {

                    arr[i][j] = 1;

                }

            }

        }

    }

}
```

```

        }
    }
}

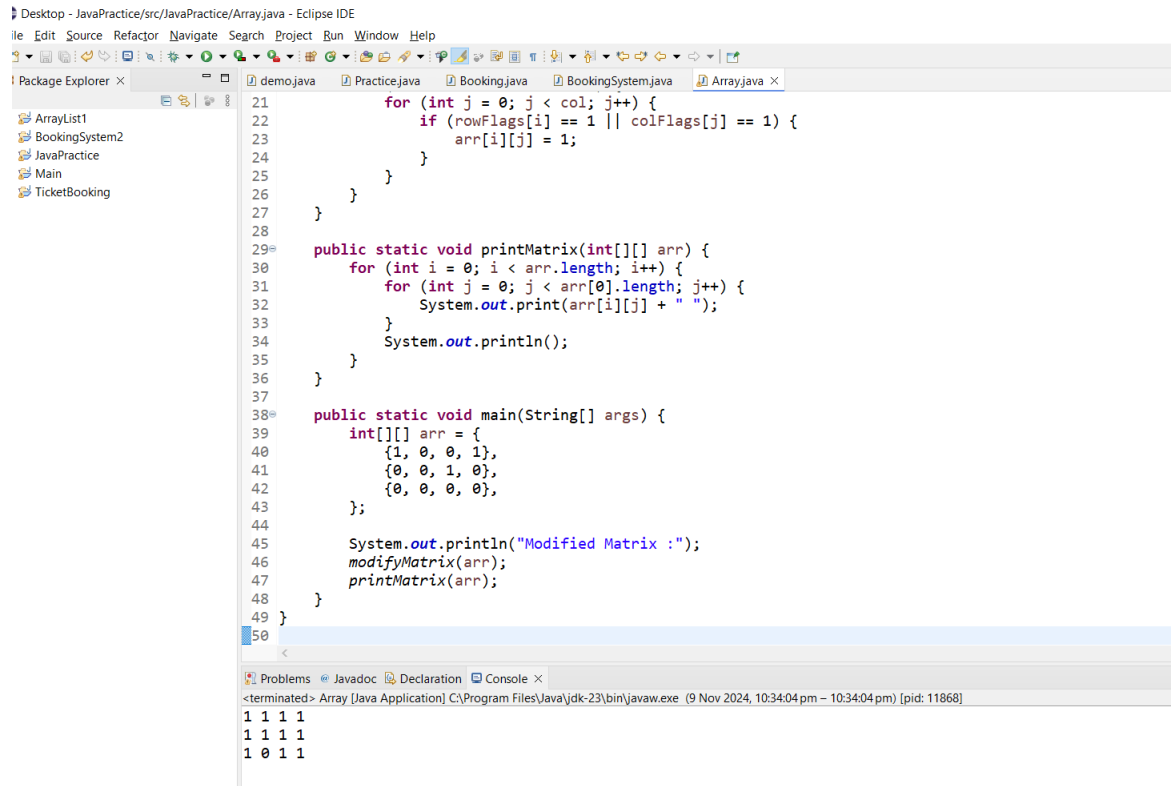
public static void printMatrix(int[][] arr) {
    for (int i = 0; i < arr.length; i++) {
        for (int j = 0; j < arr[0].length; j++) {
            System.out.print(arr[i][j] + " ");
        }
        System.out.println();
    }
}

public static void main(String[] args) {
    int[][] arr = {
        {1, 0, 0, 1},
        {0, 0, 1, 0},
        {0, 0, 0, 0},
    };
    System.out.println("Modified Matrix :");
    modifyMatrix(arr);
    printMatrix(arr);
}
}

```

**Time Complexity:**  $O(n*m)$

## OUTPUT:



```
Desktop - JavaPractice/src/JavaPractice/Array.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer x demo.java Practice.java Booking.java BookingSystem.java Array.java x
ArrayList1
BookingSystem2
JavaPractice
Main
TicketBooking

21         for (int j = 0; j < col; j++) {
22             if (rowFlags[i] == 1 || colFlags[j] == 1) {
23                 arr[i][j] = 1;
24             }
25         }
26     }
27 }
28
29 public static void printMatrix(int[][] arr) {
30     for (int i = 0; i < arr.length; i++) {
31         for (int j = 0; j < arr[0].length; j++) {
32             System.out.print(arr[i][j] + " ");
33         }
34         System.out.println();
35     }
36 }
37
38 public static void main(String[] args) {
39     int[][] arr = {
40         {1, 0, 0, 1},
41         {0, 0, 1, 0},
42         {0, 0, 0, 0},
43     };
44
45     System.out.println("Modified Matrix :");
46     modifyMatrix(arr);
47     printMatrix(arr);
48 }
49 }
50

Problems Javadoc Declaration Console x
<terminated> Array [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (9 Nov 2024, 10:34:04 pm - 10:34:04 pm) [pid: 11868]
1 1 1 1
1 1 1 1
1 0 1 1
```

## 10. Print a given matrix in spiral form

Given an m x n matrix, the task is to print all elements of the matrix in spiral form.

Input: matrix = { {1, 2, 3, 4},

{5, 6, 7, 8},

{9, 10, 11, 12},

{13, 14, 15, 16 } }

Output: 1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10

Input: matrix = { {1, 2, 3, 4, 5, 6},

{7, 8, 9, 10, 11, 12},

{13, 14, 15, 16, 17, 18} }

Output: 1 2 3 4 5 6 12 18 17 16 15 14 13 7 8 9 10 11

Explanation: The output is matrix in spiral format.

## SOLUTION:

```
package JavaPractice;

import java.util.ArrayList;
import java.util.List;

class Array {

    public List<Integer> spiralOrder(int[][] matrix) {

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

        int top = 0;
        int bot = matrix.length - 1;
        int left = 0;
        int right = matrix[0].length - 1;
        while (top <= bot && left <= right) {
            for (int i = left; i <= right; i++) {
                list.add(matrix[top][i]);
            }
            top++;
            for (int i = top; i <= bot; i++) {
                list.add(matrix[i][right]);
            }
            right--;
            if (top <= bot) {
                for (int i = right; i >= left; i--) {
                    list.add(matrix[bot][i]);
                }
            }
        }
    }
}
```

```

        }
        bot--;
    }
    if (left <= right) {
        for (int i = bot; i >= top; i--) {
            list.add(matrix[i][left]);
        }
        left++;
    }
}

return list;
}

public static void main(String[] args) {
    Array arr = new Array();
    int[][] matrix1 = {
        {1, 2, 3, 4},
        {5, 6, 7, 8},
        {9, 10, 11, 12},
        {13, 14, 15, 16}
    };
    System.out.println("Spiral Order for matrix1:");
    List<Integer> result1 = arr.spiralOrder(matrix1);
    for (int num : result1) {
        System.out.print(num + " ");
    }
}

```



```

        System.out.println();

        int[][] matrix2 = {
            {1, 2, 3, 4, 5, 6},
            {7, 8, 9, 10, 11, 12},
            {13, 14, 15, 16, 17, 18}
        };

        System.out.println("Spiral Order for matrix2:");
        List<Integer> result2 = arr.spiralOrder(matrix2);
        for (int num : result2) {
            System.out.print(num + " ");
        }
        System.out.println();
    }
}

package JavaPractice;

import java.util.Stack;

public class Array {

    public static String isBalanced(String s) {
        Stack<Character> stack = new Stack<>();

        for (int i = 0; i < s.length(); i++) {
            char ch = s.charAt(i);

```

```

        if (ch == '(') {
            stack.push(ch);
        } else if (ch == ')') {
            if (stack.isEmpty()) {
                return "Not Balanced";
            }
            stack.pop();
        }
    }

    return stack.isEmpty() ? "Balanced" : "Not Balanced";
}

public static void main(String[] args) {

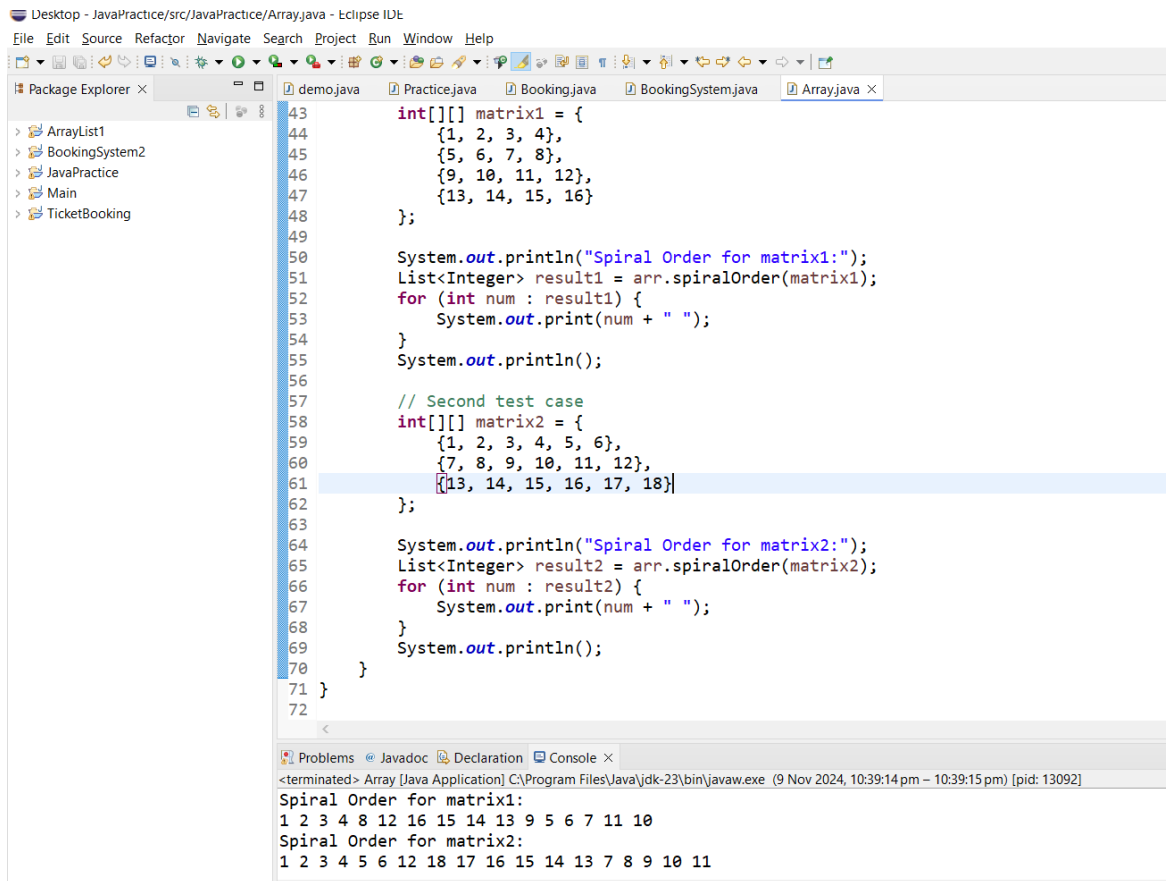
    String str1 = "((()))()()";
    System.out.println("Input: " + str1);
    System.out.println("Output: " + isBalanced(str1));

    String str2 = "()()()()";
    System.out.println("Input: " + str2);
    System.out.println("Output: " + isBalanced(str2));
}
}

```

**Time Complexity:**  $O(n*m)$

## OUTPUT:



```
Desktop - JavaPractice/src/JavaPractice/Array.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer x
  > ArrayList1
  > BookingSystem2
  > JavaPractice
  > Main
  > TicketBooking

demo.java Practice.java Booking.java BookingSystem.java Array.java x
43     int[][] matrix1 = {
44         {1, 2, 3, 4},
45         {5, 6, 7, 8},
46         {9, 10, 11, 12},
47         {13, 14, 15, 16}
48     };
49
50     System.out.println("Spiral Order for matrix1:");
51     List<Integer> result1 = arr.spiralOrder(matrix1);
52     for (int num : result1) {
53         System.out.print(num + " ");
54     }
55     System.out.println();
56
57     // Second test case
58     int[][] matrix2 = {
59         {1, 2, 3, 4, 5, 6},
60         {7, 8, 9, 10, 11, 12},
61         {13, 14, 15, 16, 17, 18}
62     };
63
64     System.out.println("Spiral Order for matrix2:");
65     List<Integer> result2 = arr.spiralOrder(matrix2);
66     for (int num : result2) {
67         System.out.print(num + " ");
68     }
69     System.out.println();
70 }
71 }
72 }

Problems Javadoc Declaration Console x
<terminated> Array [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (9 Nov 2024, 10:39:14 pm - 10:39:15 pm) [pid: 13092]
Spiral Order for matrix1:
1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10
Spiral Order for matrix2:
1 2 3 4 5 6 12 18 17 16 15 14 13 7 8 9 10 11
```

### 13. Check if given Parentheses expression is balanced or not

Given a string str of length N, consisting of „(, „, and „), only, the task is to check whether it is balanced or not.

Input: str = “((()))00”

Output: Balanced

Input: str = “0)()((0)”

Output: Not Balanced

## SOLUTION:

```
package JavaPractice;

import java.util.Stack;

public class Array {

    public static String isBalanced(String s) {
        Stack<Character> stack = new Stack<>();

        for (int i = 0; i < s.length(); i++) {
            char ch = s.charAt(i);

            if (ch == '(') {
                stack.push(ch);
            } else if (ch == ')') {
                if (stack.isEmpty()) {
                    return "Not Balanced";
                }
                stack.pop();
            }
        }

        return stack.isEmpty() ? "Balanced" : "Not Balanced";
    }

    public static void main(String[] args) {
        String str1 = "((()))()()";

        System.out.println("Input: " + str1);

        System.out.println("Output: " + isBalanced(str1));
    }
}
```

```

String str2 = "()((()))";

System.out.println("Input: " + str2);

System.out.println("Output: " + isBalanced(str2));

}

}

```

**Time Complexity:  $O(n)$**

## OUTPUT:

```

1 package JavaPractice;
2 import java.util.Stack;
3
4 public class Array {
5
6     public static String isBalanced(String s) {
7         Stack<Character> stack = new Stack<>();
8
9         for (int i = 0; i < s.length(); i++) {
10             char ch = s.charAt(i);
11
12             if (ch == '(') {
13                 stack.push(ch);
14             } else if (ch == ')') {
15                 if (stack.isEmpty()) {
16                     return "Not Balanced";
17                 }
18                 stack.pop();
19             }
20         }
21
22         return stack.isEmpty() ? "Balanced" : "Not Balanced";
23     }
24
25     public static void main(String[] args) {
26
27         String str1 = "((( )))()";
28         System.out.println("Input: " + str1);
29         System.out.println("Output: " + isBalanced(str1));
30
31     }
32 }

```

Problems Javadoc Declaration Console ×

<terminated> Array [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (9 Nov 2024, 10:45:09 pm – 10:45:10 pm) [pid: 16876]

```

Input: ((( )))()
Output: Balanced
Input: ()(( ))
Output: Not Balanced

```

## 14. Check if two Strings are Anagrams of each other

Given two strings s1 and s2 consisting of lowercase characters, the task is to check whether the two given strings are anagrams of each other or not. An anagram of a

string is another string that contains the same characters, only the order of characters can be different.

Input: s1 = “geeks” s2 = “kseeg”

Output: true

Explanation: Both the string have same characters with same frequency. So, they are anagrams.

Input: s1 = “allergy” s2 = “allergic”

Output: false

Explanation: Characters in both the strings are not same. s1 has extra character „y“ and s2 has extra characters „i“ and „c“, so they are not anagrams.

Input: s1 = “g”, s2 = “g”

Output: true

Explanation: Characters in both the strings are same, so they are anagrams

## **SOLUTION:**

```
package JavaPractice;
```

```
import java.util.Arrays;
```

```
public class Array {
```

```
    public boolean isAnagram(String s, String t) {
```

```
        char[] sarr = s.toCharArray();
```

```
        char[] tarr = t.toCharArray();
```

```
        Arrays.sort(sarr);
```

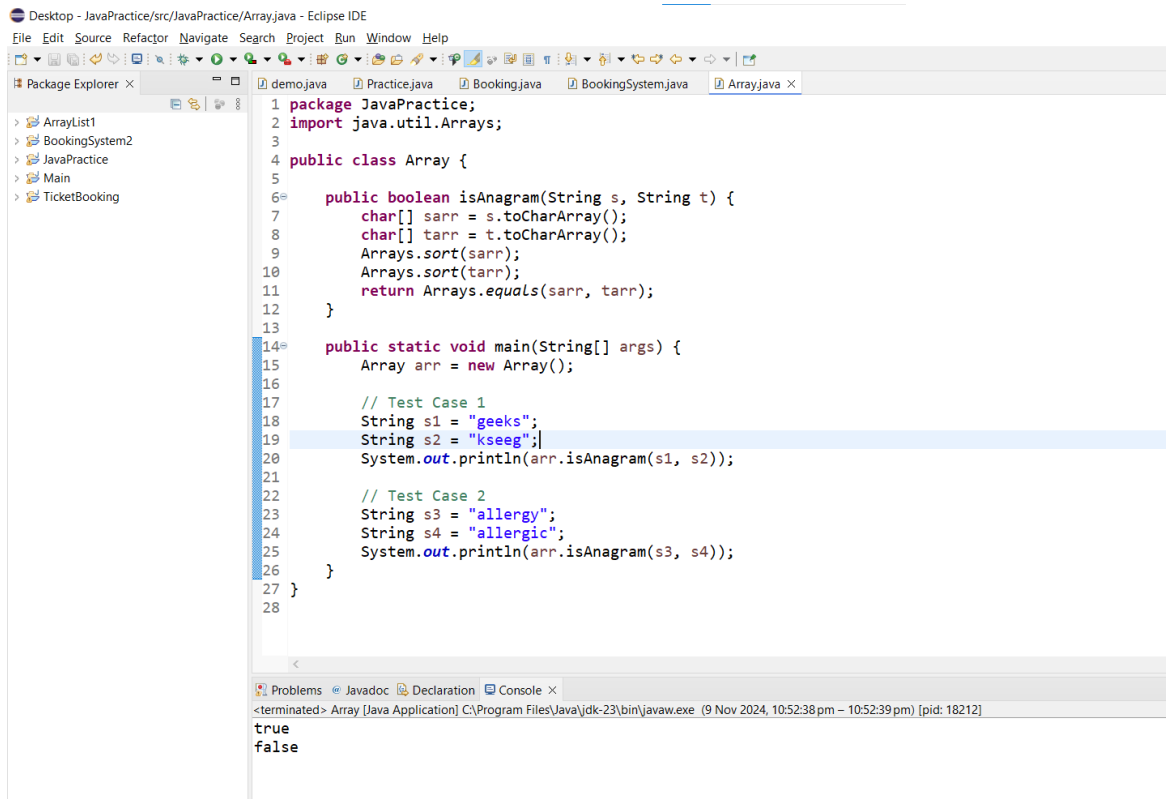
```
        Arrays.sort(tarr);
```

```
        return Arrays.equals(sarr, tarr);
```

```
}  
  
public static void main(String[] args) {  
    Array arr = new Array();  
  
    // Test Case 1  
  
    String s1 = "geeks";  
    String s2 = "kseeg";  
    System.out.println(arr.isAnagram(s1, s2));  
  
    // Test Case 2  
  
    String s3 = "allergy";  
    String s4 = "allergic";  
    System.out.println(arr.isAnagram(s3, s4));  
}  
}
```

**Time Complexity:**  $O(n)$

**OUTPUT:**



The screenshot shows the Eclipse IDE with a Java project named 'JavaPractice'. The 'Package Explorer' on the left shows the project structure. The main editor displays the code for 'Array.java'. The code defines a package 'JavaPractice', imports 'java.util.Arrays', and defines a public class 'Array'. It contains a method 'isAnagram' that takes two strings 's' and 't', converts them to character arrays, sorts them, and returns 'true' if they are equal. The 'main' method tests this with two cases: 'geeks' and 'kseeg' (returns true) and 'allergy' and 'allergic' (returns false). The console at the bottom shows the output: 'true' and 'false'.

```
1 package JavaPractice;
2 import java.util.Arrays;
3
4 public class Array {
5
6     public boolean isAnagram(String s, String t) {
7         char[] sarr = s.toCharArray();
8         char[] tarr = t.toCharArray();
9         Arrays.sort(sarr);
10        Arrays.sort(tarr);
11        return Arrays.equals(sarr, tarr);
12    }
13
14    public static void main(String[] args) {
15        Array arr = new Array();
16
17        // Test Case 1
18        String s1 = "geeks";
19        String s2 = "kseeg";
20        System.out.println(arr.isAnagram(s1, s2));
21
22        // Test Case 2
23        String s3 = "allergy";
24        String s4 = "allergic";
25        System.out.println(arr.isAnagram(s3, s4));
26    }
27 }
28
```

Problems Javadoc Declaration Console  
<terminated> Array [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (9 Nov 2024, 10:52:38 pm - 10:52:39 pm) [pid: 18212]  
true  
false

## 15.Longest Palindromic Substring

Given a string str, the task is to find the longest substring which is a palindrome. If there are multiple answers, then return the first appearing substring.

Input: str = “forgeeksskeegfor”

Output: “geeksskeeg”

Explanation: There are several possible palindromic substrings like “kssk”, “ss”, “eeksskee” etc. But the substring “geeksskeeg” is the longest among all.

### SOLUTION:

```
package JavaPractice;

class Array {

    public String longestPalindrome(String s) {
```



```

    if (s == null || s.length() == 0) {
        return "";
    }

    int start = 0;
    int end = 0;

    for (int i = 0; i < s.length(); i++) {
        int odd = expandAroundCenter(s, i, i);
        int even = expandAroundCenter(s, i, i + 1);
        int max_len = Math.max(odd, even);

        if (max_len > end - start) {
            start = i - (max_len - 1) / 2;
            end = i + max_len / 2;
        }
    }

    return s.substring(start, end + 1);
}

private int expandAroundCenter(String s, int left, int right) {
    while (left >= 0 && right < s.length() && s.charAt(left) ==
s.charAt(right)) {
        left--;
        right++;
    }

    return right - left - 1;
}

public static void main(String[] args) {

```

```

        Array array = new Array();

        String str1 = "Geeks";

        System.out.println("Output: " +
array.longestPalindrome(str1));

        String str2 = "";

        System.out.println("Output: " +
array.longestPalindrome(str2));

    }

}

```

**Time Complexity:  $O(n)$**

**OUTPUT:**

```

Desktop - JavaPractice/src/JavaPractice/Array.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
  ArrayList1
  BookingSystem2
  JavaPractice
  Main
  TicketBooking
demo.java Practice.java Booking.java BookingSystem.java Array.java
16         if (max_len > end - start) {
17             start = i - (max_len - 1) / 2;
18             end = i + max_len / 2;
19         }
20     }
21
22     return s.substring(start, end + 1);
23 }
24
25 private int expandAroundCenter(String s, int left, int right) {
26     while (left >= 0 && right < s.length() && s.charAt(left) == s.charAt(right)) {
27         left--;
28         right++;
29     }
30     return right - left - 1;
31 }
32
33
34
35 public static void main(String[] args) {
36     Array array = new Array();
37
38     String str1 = "Geeks";
39     System.out.println("Output: " + array.longestPalindrome(str1));
40
41     String str2 = "";
42     System.out.println("Output: " + array.longestPalindrome(str2));
43 }
44 }
45
Problems Javadoc Declaration Console
<terminated> Array [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (10 Nov 2024, 9:24:22 am - 9:24:23 am) [pid: 8260]
Output: ee
Output:

```

## 16.Longest Common Prefix using Sorting

Given an array of strings `arr[]`. The task is to return the longest common prefix among each and every strings present in the array. If there's no prefix common in all the strings, return "-1".

Input: arr[] = ["geeksforgeeks", "geeks", "geek", "geezer"]

Output: gee

Explanation: "gee" is the longest common prefix in all the given strings.

Input: arr[] = ["hello", "world"]

Output: -1

Explanation: There's no common prefix in the given strings.

### SOLUTION:

```
package JavaPractice;

import java.util.Arrays;

class Array {

    public String longestCommonPrefix(String[] strs) {

        if (strs == null || strs.length == 0) {

            return "-1";

        }

        StringBuilder ans = new StringBuilder();

        Arrays.sort(strs);

        for (int i = 0; i < strs[0].length(); i++) {

            if (strs[0].charAt(i) != strs[strs.length - 1].charAt(i))

            {

                ans.append(strs[0].charAt(i));

            } else {

                break;

            }

        }

    }

}
```

```

    }

    return ans.length() > 0 ? ans.toString() : "-1";
}

public static void main(String[] args) {
    Array array = new Array();

    String[] str1 = {"geeksforgeeks", "geeks", "geek", "geezer"};

    System.out.println("Output: " +
array.longestCommonPrefix(str1));

    String[] str2 = {"hello", "world"};

    System.out.println("Output: " +
array.longestCommonPrefix(str2));
}
}

```

**Time Complexity:**  $O(n \log n + m)$

**OUTPUT:**

```
File Edit Source Refactor Navigate Search Project Run Window Help
demo.java Practice.java Booking.java BookingSystem.java Array.java x
Package Explorer x
> ArrayList1
> BookingSystem2
> JavaPractice
> Main
> TicketBooking
5 public String longestCommonPrefix(String[] strs) {
6     if (strs == null || strs.length == 0) {
7         return "-1";
8     }
9
10    StringBuilder ans = new StringBuilder();
11    Arrays.sort(strs);
12
13    for (int i = 0; i < strs[0].length(); i++) {
14        if (strs[0].charAt(i) != strs[strs.length - 1].charAt(i)) {
15            ans.append(strs[0].charAt(i));
16        } else {
17            break;
18        }
19    }
20
21    return ans.length() > 0 ? ans.toString() : "-1";
22 }
23
24 public static void main(String[] args) {
25     Array array = new Array();
26
27     String[] strs1 = {"geeksforgeeks", "geeks", "geek", "geezer"};
28     System.out.println("Output: " + array.longestCommonPrefix(strs1));
29
30     String[] strs2 = {"hello", "world"};
31     System.out.println("Output: " + array.longestCommonPrefix(strs2));
32 }
33 }
34
Problems Javadoc Declaration Console x
<terminated> Array [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (10 Nov 2024, 9:31:16 am - 9:31:17 am) [pid: 5236]
Output: gee
Output: -1
```

## 17. Delete middle element of a stack

Given a stack with push(), pop(), and empty() operations, The task is to delete the middle element of it without using any additional data structure.

Input : Stack[] = [1, 2, 3, 4, 5]

Output : Stack[] = [1, 2, 4, 5]

Input : Stack[] = [1, 2, 3, 4, 5, 6]

Output : Stack[] = [1, 2, 4, 5, 6]

### SOLUTION:

```
package JavaPractice;

import java.util.Stack;

class Array {
```

```
public void deleteMiddle(Stack<Integer> stack, int current) {  
    if (stack.isEmpty()) {  
        return;  
    }  
    int middle = (stack.size() / 2) + 1;  
    deleteMiddleHelper(stack, middle, current);  
}  
  
private void deleteMiddleHelper(Stack<Integer> stack, int middle,  
int current) {  
    if (current == middle) {  
        stack.pop();  
        return;  
    }  
    int top = stack.pop();  
    deleteMiddleHelper(stack, middle, current + 1);  
    stack.push(top);  
}  
  
public static void main(String[] args) {  
    Array array = new Array();  
    Stack<Integer> stack1 = new Stack<>();  
    stack1.push(1);  
    stack1.push(2);  
    stack1.push(3);  
    stack1.push(4);  
    stack1.push(5);  
}
```

```

        array.deleteMiddle(stack1, 1);

        System.out.println("After deleting middle element from stack1:
" + stack1);

        Stack<Integer> stack2 = new Stack<>();

        stack2.push(1);

        stack2.push(2);

        stack2.push(3);

        stack2.push(4);

        stack2.push(5);

        stack2.push(6);

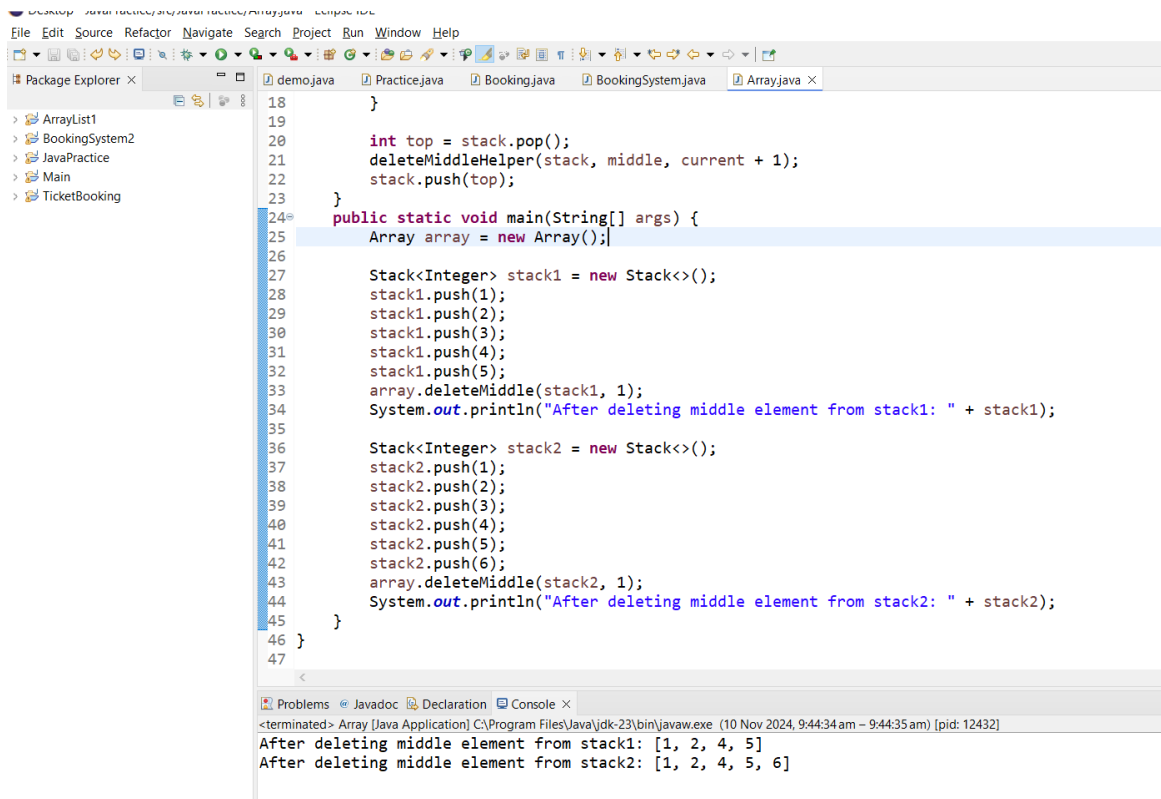
        array.deleteMiddle(stack2, 1);

        System.out.println("After deleting middle element from stack2:
" + stack2);
    }
}

```

**Time Complexity:**  $O(n)$

**OUTPUT:**



```
18 }
19
20 int top = stack.pop();
21 deleteMiddleHelper(stack, middle, current + 1);
22 stack.push(top);
23 }
24 public static void main(String[] args) {
25     Array array = new Array();
26
27     Stack<Integer> stack1 = new Stack<>();
28     stack1.push(1);
29     stack1.push(2);
30     stack1.push(3);
31     stack1.push(4);
32     stack1.push(5);
33     array.deleteMiddle(stack1, 1);
34     System.out.println("After deleting middle element from stack1: " + stack1);
35
36     Stack<Integer> stack2 = new Stack<>();
37     stack2.push(1);
38     stack2.push(2);
39     stack2.push(3);
40     stack2.push(4);
41     stack2.push(5);
42     stack2.push(6);
43     array.deleteMiddle(stack2, 1);
44     System.out.println("After deleting middle element from stack2: " + stack2);
45 }
46 }
47
```

Problems Javadoc Declaration Console ×

<terminated> Array [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (10 Nov 2024, 9:44:34 am - 9:44:35 am) [pid: 12432]

After deleting middle element from stack1: [1, 2, 4, 5]

After deleting middle element from stack2: [1, 2, 4, 5, 6]

## 18)Next Greater Element (NGE) for every element in given Array

Given an array, print the Next Greater Element (NGE) for every element.

Note: The Next greater Element for an element x is the first greater element on the right side of x in the array. Elements for which no greater element exist, consider the next greater element as -1.

Input: arr[] = [ 4 , 5 , 2 , 25 ]

Output: 4 -> 5

5 -> 25

2 -> 25

25 -> -1

Explanation: Except 25 every element has an element greater than them present on the right side



Input: arr[] = [ 13 , 7, 6 , 12 ]

Output: 13 -> -1

7 -> 12

6 -> 12

12 -> -1

Explanation: 13 and 12 don't have any element greater than them present on the right side

## SOLUTION:

```
package JavaPractice;

import java.util.Arrays;
import java.util.ArrayDeque;
import java.util.Deque;

class Array {

    public int[] nextGreaterElements(int[] nums) {

        int[] res = new int[nums.length];

        int n = nums.length;

        Arrays.fill(res, -1);

        Deque<Integer> s = new ArrayDeque<>();

        for (int i = 0; i < 2 * nums.length; i++) {

            while (!s.isEmpty() && nums[i % n] > nums[s.peek()]) {

                res[s.pop()] = nums[i % n];

            }

        }

    }

}
```

```

        s.push(i % n);
    }

    return res;
}

public static void main(String[] args) {
    Array array = new Array();

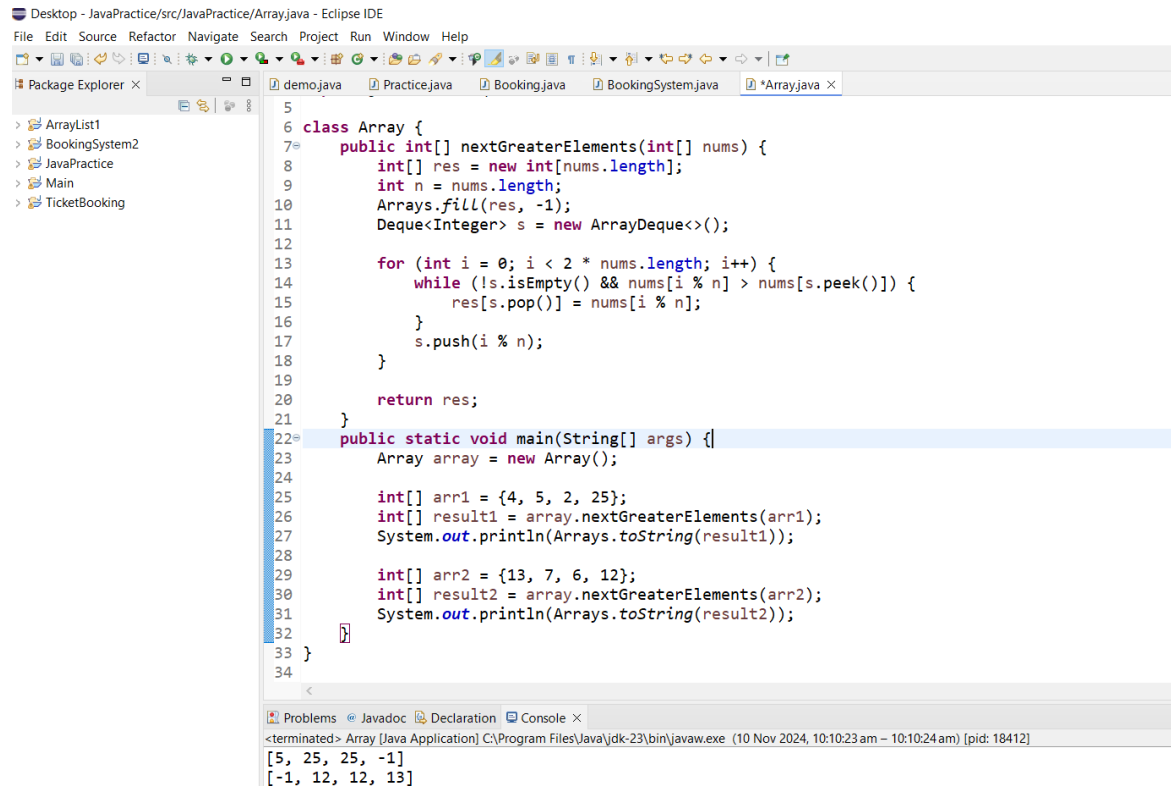
    int[] arr1 = {4, 5, 2, 25};
    int[] result1 = array.nextGreaterElements(arr1);
    System.out.println(Arrays.toString(result1));

    int[] arr2 = {13, 7, 6, 12};
    int[] result2 = array.nextGreaterElements(arr2);
    System.out.println(Arrays.toString(result2));
}
}

```

Time Complexity:  $O(n)$

**OUTPUT:**



```
5
6 class Array {
7     public int[] nextGreaterElements(int[] nums) {
8         int[] res = new int[nums.length];
9         int n = nums.length;
10        Arrays.fill(res, -1);
11        Deque<Integer> s = new ArrayDeque<>();
12
13        for (int i = 0; i < 2 * nums.length; i++) {
14            while (!s.isEmpty() && nums[i % n] > nums[s.peek()]) {
15                res[s.pop()] = nums[i % n];
16            }
17            s.push(i % n);
18        }
19
20        return res;
21    }
22    public static void main(String[] args) {
23        Array array = new Array();
24
25        int[] arr1 = {4, 5, 2, 25};
26        int[] result1 = array.nextGreaterElements(arr1);
27        System.out.println(Arrays.toString(result1));
28
29        int[] arr2 = {13, 7, 6, 12};
30        int[] result2 = array.nextGreaterElements(arr2);
31        System.out.println(Arrays.toString(result2));
32    }
33 }
34
```

Problems Javadoc Declaration Console ×

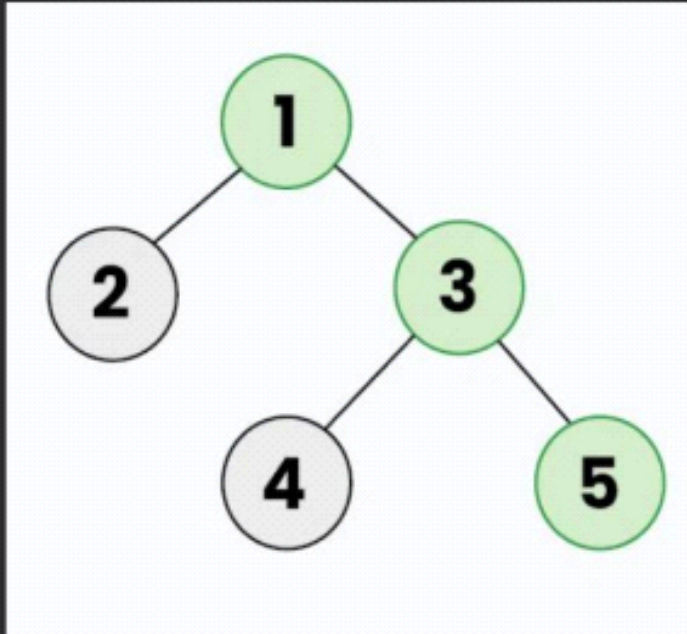
<terminated> Array [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (10 Nov 2024, 10:10:23 am – 10:10:24 am) [pid: 18412]

[5, 25, 25, -1]  
[-1, 12, 12, 13]

## 19. Print Right View of a Binary Tree

Given a Binary Tree, the task is to print the Right view of it. The right view of a Binary Tree is a set of rightmost nodes for every level.

*Example 1: The **Green** colored nodes (1, 3, 5) represents the Right view in the below Binary tree.*



### SOLUTION:

```
package JavaPractice;

import java.util.*;

class TreeNode {
    int val;
    TreeNode left;
    TreeNode right;
    TreeNode(int x) {
        val = x;
        left = null;
        right = null;
    }
}
```

```
    }  
}
```

```
class Array {  
    public List<Integer> rightSideView(TreeNode root) {  
        List<Integer> result = new ArrayList<Integer>();  
        rightView(root, result, 0);  
        return result;  
    }  
    public void rightView(TreeNode curr, List<Integer> result, int  
currDepth)  
    {  
        if(curr == null) {  
            return;  
        }  
        if(currDepth == result.size()) {  
            result.add(curr.val);  
        }  
        rightView(curr.right, result, currDepth + 1);  
        rightView(curr.left, result, currDepth + 1);  
    }  
    public static void main(String[] args)  
    {  
        TreeNode root = new TreeNode(1);  
        root.left = new TreeNode(2);  
    }  
}
```

```
root.right = new TreeNode(3);
root.right.right = new TreeNode(5);
root.right.left = new TreeNode(4);
Array solution = new Array();
List<Integer> rightViewList = solution.rightSideView(root);
System.out.println("Right View:");
for (Integer val : rightViewList)
{
    System.out.print(val + " ");
}
}
```

**Time Complexity:**  $O(n)$

**OUTPUT:**

```
23
24 public void rightView(TreeNode curr, List<Integer> result, int currDepth)
25 {
26     if(curr == null) {
27         return;
28     }
29     if(currDepth == result.size()) {
30         result.add(curr.val);
31     }
32     rightView(curr.right, result, currDepth + 1);
33     rightView(curr.left, result, currDepth + 1);
34 }
35
36 public static void main(String[] args)
37 {
38     TreeNode root = new TreeNode(1);
39     root.left = new TreeNode(2);
40     root.right = new TreeNode(3);
41     root.right.right = new TreeNode(5);
42     root.right.left = new TreeNode(4);
43     Array solution = new Array();
44     List<Integer> rightViewList = solution.rightSideView(root);
45     System.out.println("Right View:");
46     for (Integer val : rightViewList)
47     {
48         System.out.print(val + " ");
49     }
50 }
51 }
52
```

Problems Javadoc Declaration Console x

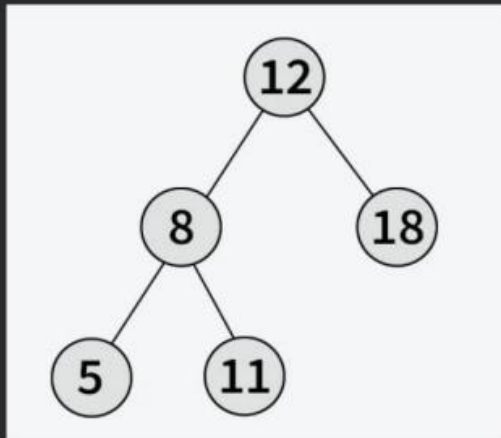
<terminated> Array [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (10 Nov 2024, 10:12:47 am - 10:12:47 am) [pid: 11144]

Right View:  
1 3 5

## 20)Maximum Depth or Height of Binary Tree

Given a binary tree, the task is to find the maximum depth or height of the tree. The height of the Tree is the number of vertices in the tree from the root to the deepest node

*Example 1: The height of the below binary tree is 3.*



**SOLUTION:**

```
package JavaPractice;
```

```
class Node {  
    int val;  
    Node left;  
    Node right;  
  
    Node(int x) {  
        val = x;  
        left = null;  
        right = null;  
    }  
}
```

```
class Array {
```



```

public int maxDepth(Node root) {
    if (root == null) {
        return 0;
    }

    int lh = maxDepth(root.left);
    int rh = maxDepth(root.right);
    return 1 + Math.max(lh, rh);
}

public static void main(String[] args) {
    Node root = new Node(12);
    root.left = new Node(8);
    root.right = new Node(18);
    root.left.left = new Node(5);
    root.left.right = new Node(11);

    Array solution = new Array();
    int maxDepth = solution.maxDepth(root);

    System.out.println("Maximum Depth:" + maxDepth);
}
}

```

**Time Complexity:**  $O(n)$

**OUTPUT:**

Desktop - javanovice\src\javanovice\main.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer ×

- > ArrayList1
- > BookingSystem2
- > JavaPractice
- > Main
- > TicketBooking

```
9      val = x;
10     left = null;
11     right = null;
12 }
13 }
14
15 class Array {
16     public int maxDepth(Node root) {
17         if (root == null) {
18             return 0;
19         }
20         int lh = maxDepth(root.left);
21         int rh = maxDepth(root.right);
22         return 1 + Math.max(lh, rh);
23     }
24
25     public static void main(String[] args) {
26         Node root = new Node(12);
27         root.left = new Node(8);
28         root.right = new Node(18);
29         root.left.left = new Node(5);
30         root.left.right = new Node(11);
31
32         Array solution = new Array();
33         int maxDepth = solution.maxDepth(root);
34
35         System.out.println("Maximum Depth:" + maxDepth);
36     }
37 }
38
```

Problems Javadoc Declaration Console ×

<terminated> Array [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (10 Nov 2024, 10:19:51 am - 10:19:51 am) [pid: 11028]

Maximum Depth:3