

VIJAY M

22IT125

DAY-4

CODING PRACTICES AND PROBLEM

1. Stock Buy and Sell

Time Complexity : $O(n)$

Solution:

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

```
class GfG {
```

```
    static int maxProfitRec(int[] price, int start, int end) {  
        int res = 0;
```

```
        for (int i = start; i < end; i++) {  
            for (int j = i + 1; j <= end; j++) {  
                if (price[j] > price[i]) {  
                    int curr = (price[j] - price[i]) +  
                        maxProfitRec(price, start, i - 1) +  
                        maxProfitRec(price, j + 1, end);  
                    res = Math.max(res, curr);  
                }  
            }  
        }
```

```
        return res;  
    }
```

```
    static int maximumProfit(int[] prices) {  
        return maxProfitRec(prices, 0, prices.length - 1);  
    }
```

```

public static void main(String[] args) {
    int[] prices = {100, 180, 260, 310, 40, 535, 695};
    System.out.println(maximumProfit(prices));
}
}

```

Output:

The screenshot shows an IDE with a project named 'TRIPILET'. The 'src' folder contains 'Main.java'. The code in 'Main.java' defines a class 'GfG' with a static method 'maxProfitRec' that takes an array of prices and start/end indices. It uses a recursive approach to find the maximum profit. The 'Run' tab at the bottom shows the command executed: 'C:\Users\sanja\jdk\openjdk-23.0.1\bin\java.exe "-javaagent:C:\Users\sanja\AppData\Local\JetBrains\IntelliJ IDEA 2024.2.4\lib\idea_rt.jar=57307:C:\Users\sanja\AppData\Local\Je 865'. The output is 'Process finished with exit code 0'.

```

1 import java.util.Arrays;
2
3 class GfG {
4
5     static int maxProfitRec(int[] price, int start, int end) {
6         int res = 0;
7
8         for (int i = start; i < end; i++) {
9             for (int j = i + 1; j <= end; j++) {
10                 if (price[j] > price[i]) {
11                     int curr = (price[j] - price[i]) +
12                         maxProfitRec(price, start, end: i - 1) +
13                         maxProfitRec(price, start: j + 1, end);
14                     res = Math.max(res, curr);
15                 }
16             }
17         }
18         return res;
19     }
20 }

```

2.Coin Change

Time Complexity : $O(\text{sum} * n)$

Solution:

```

class GfG {

    static int countRecur(int[] coins, int n, int sum) {

        if (sum == 0) return 1;
    }
}

```

```
if (sum < 0 || n == 0) return 0;
```

```
return countRecur(coins, n, sum - coins[n - 1]) +  
    countRecur(coins, n - 1, sum);
```

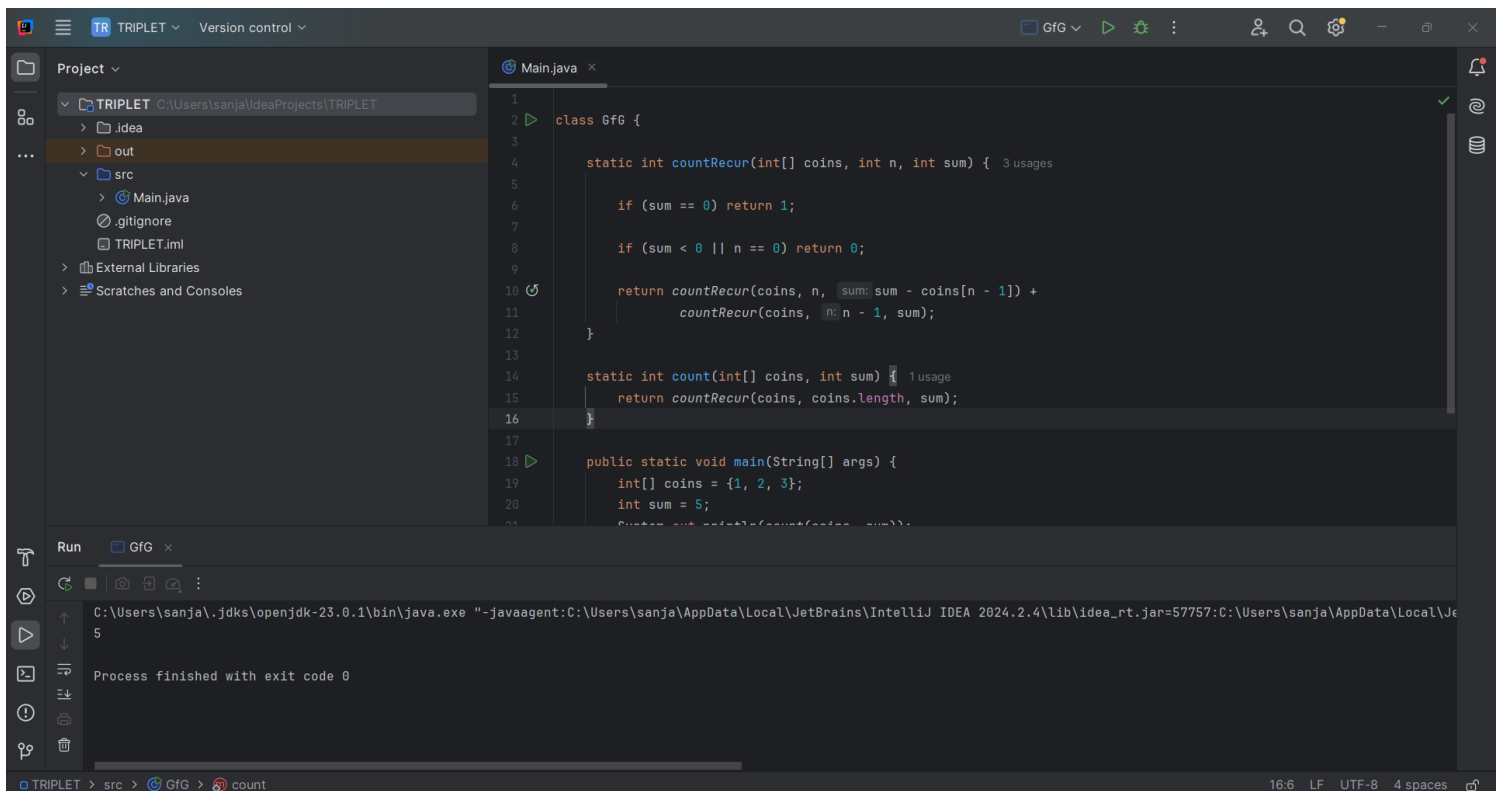
```
}
```

```
static int count(int[] coins, int sum) {  
    return countRecur(coins, coins.length, sum);  
}
```

```
public static void main(String[] args) {  
    int[] coins = {1, 2, 3};  
    int sum = 5;  
    System.out.println(count(coins, sum));  
}
```

```
}
```

Output:



```
1  
2 class Gfg {  
3  
4     static int countRecur(int[] coins, int n, int sum) { 3 usages  
5  
6         if (sum == 0) return 1;  
7  
8         if (sum < 0 || n == 0) return 0;  
9  
10        return countRecur(coins, n, sum - coins[n - 1]) +  
11            countRecur(coins, n - 1, sum);  
12    }  
13  
14    static int count(int[] coins, int sum) 1 usage  
15    {  
16        return countRecur(coins, coins.length, sum);  
17    }  
18  
19    public static void main(String[] args) {  
20        int[] coins = {1, 2, 3};  
21        int sum = 5;  
22        System.out.println(count(coins, sum));  
23    }  
24 }
```

Run Gfg x

C:\Users\sanja\.jdk\openjdk-23.0.1\bin\java.exe "-javaagent:C:\Users\sanja\AppData\Local\JetBrains\IntelliJ IDEA 2024.2.4\lib\idea_rt.jar=57757:C:\Users\sanja\AppData\Local\Je
5

Process finished with exit code 0

TRIPLET > src > Gfg > count 16:6 LF UTF-8 4 spaces

3.First and Last Occurance

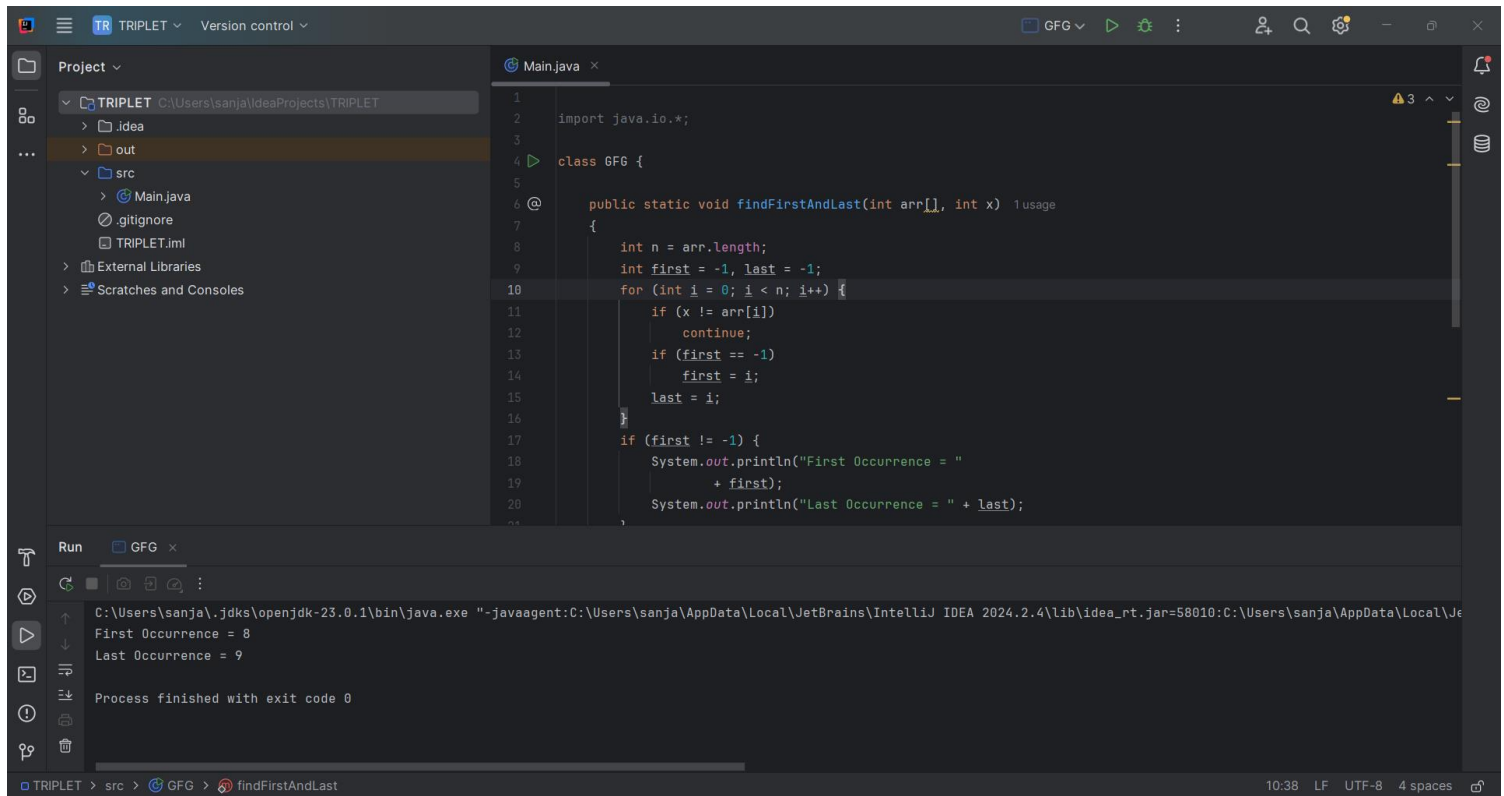
Time Complexity : $O(n)$

Solution:

```
import java.io.*;
class GFG {
    public static void findFirstAndLast(int arr[], int x)
    {
        int n = arr.length;
        int first = -1, last = -1;
        for (int i = 0; i < n; i++) {
            if (x != arr[i])
                continue;
            if (first == -1)
                first = i;
            last = i;
        }
        if (first != -1) {
            System.out.println("First Occurrence = "
                               + first);
            System.out.println("Last Occurrence = " + last);
        }
        else
            System.out.println("Not Found");
    }

    public static void main(String[] args)
    {
        int arr[] = { 1, 2, 2, 2, 2, 3, 4, 7, 8, 8 };
        int x = 8;
        findFirstAndLast(arr, x);
    }
}
```

Output:



The screenshot shows an IDE with a project named 'TRIPLET'. The 'Main.java' file is open, containing the following code:

```
1
2 import java.io.*;
3
4 class GFG {
5
6     public static void findFirstAndLast(int arr[], int x) {usage
7     {
8         int n = arr.length;
9         int first = -1, last = -1;
10        for (int i = 0; i < n; i++) {
11            if (x != arr[i])
12                continue;
13            if (first == -1)
14                first = i;
15            last = i;
16        }
17        if (first != -1) {
18            System.out.println("First Occurrence = "
19                               + first);
20            System.out.println("Last Occurrence = " + last);
21        }
22    }
23 }
```

The 'Run' tab shows the output of the program:

```
C:\Users\sanja\.jdk\openjdk-23.0.1\bin\java.exe "-javaagent:C:\Users\sanja\AppData\Local\JetBrains\IntelliJ IDEA 2024.2.4\lib\idea_rt.jar=58010:C:\Users\sanja\AppData\Local\J...
First Occurrence = 8
Last Occurrence = 9
Process finished with exit code 0
```

4.First and Transition Point

Time Complexity : $O(n)$

Solution:

```
import java.util.*;
```

```
class GFG
```

```
{
```

```
static int findTransitionPoint(int arr[], int n)
```

```
{
```

```
for(int i = 0; i < n ; i++)
```

```
    if(arr[i] == 1)
```

```
        return i;
```

```
return -1;
```

```

    }

    public static void main (String[] args)
    {
        int arr[] = {0, 0, 0, 0, 1, 1};
        int n = arr.length;

        int point = findTransitionPoint(arr, n);

        if (point >= 0)
            System.out.print("Transition point is " + point);
        else
            System.out.print("There is no transition point");
    }
}

```

Output:

The screenshot shows an IDE window with a project named 'TRIPLET'. The 'Main.java' file is open, showing the following code:

```

1  import java.util.*;
2
3
4  class GFG
5  {
6
7      static int findTransitionPoint(int arr[], int n) 1 usage
8      {
9
10         for(int i = 0; i < n ; i++)
11             if(arr[i] == 1)
12                 return i;
13
14         return -1;
15     }
16
17     public static void main (String[] args)
18     {
19         int arr[] = {0, 0, 0, 0, 1, 1};
20         int n = arr.length;
21     }
22 }

```

The 'Run' tab at the bottom shows the execution output:

```

C:\Users\sanja\.jdk\openjdk-23.0.1\bin\java.exe "-javaagent:C:\Users\sanja\AppData\Local\JetBrains\IntelliJ IDEA 2024.2.4\lib\idea_rt.jar=58248:C:\Users\sanja\AppData\Local\Je
Transition point is 4
Process finished with exit code 0

```

The status bar at the bottom indicates the file path: 'TRIPLET > src > GFG > findTransitionPoint' and the time: '10:36'.

5.First Repeating Element

Time Complexity: $O(n)$

Solution:

```
import java.util.*;
```

```
class GfG {
```

```
    static int firstRepeatingElement(int[] arr, int n) {
```

```
        for (int i = 0; i < n; i++) {  
            for (int j = i + 1; j < n; j++) {
```

```
                if (arr[i] == arr[j]) {  
                    return i;
```

```
                }
```

```
            }
```

```
        }
```

```
        return -1;
```

```
    }
```

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

```
        int[] arr = { 10, 5, 3, 4, 3, 5, 6 };
```

```
        int n = arr.length;
```

```
        int index = firstRepeatingElement(arr, n);
```

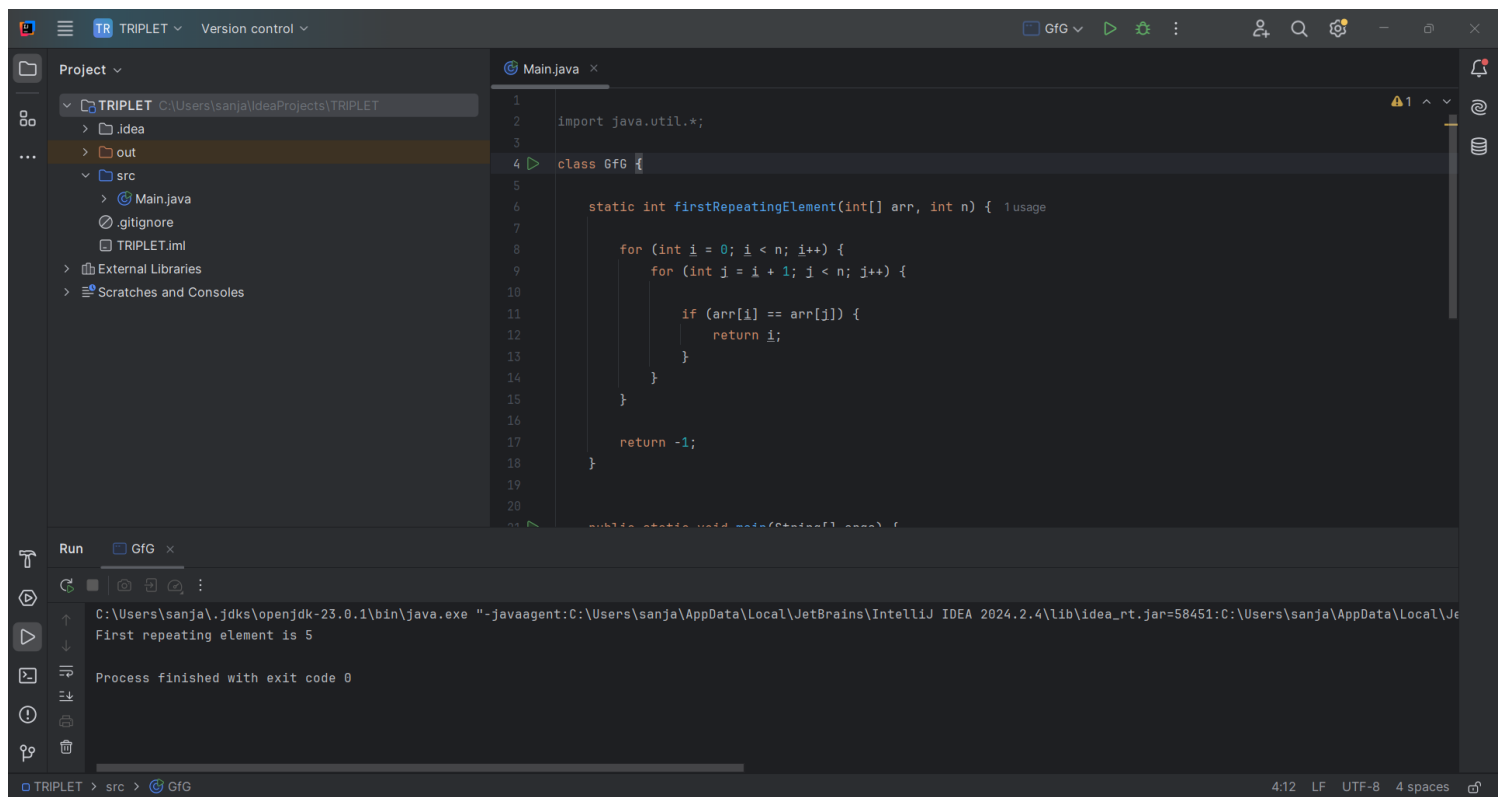
```
        if (index == -1) {
```

```
            System.out.println("No repeating element found!");
```

```
        }
```

```
else {
    System.out.println("First repeating element is " + arr[index]);
}
```

Output:



6.Remove Duplicates sorted array

Time Complexity : $O(n)$

Solution:

```
import java.util.HashSet;
```

```
class GfG {
```

```
static int removeDuplicates(int[] arr) {
```



```
HashSet<Integer> s = new HashSet<>();
```

```
int idx = 0;
```

```
for (int i = 0; i < arr.length; i++) {  
    if (!s.contains(arr[i])) {  
        s.add(arr[i]);  
        arr[idx++] = arr[i];  
    }  
}
```

```
return idx;
```

```
}
```

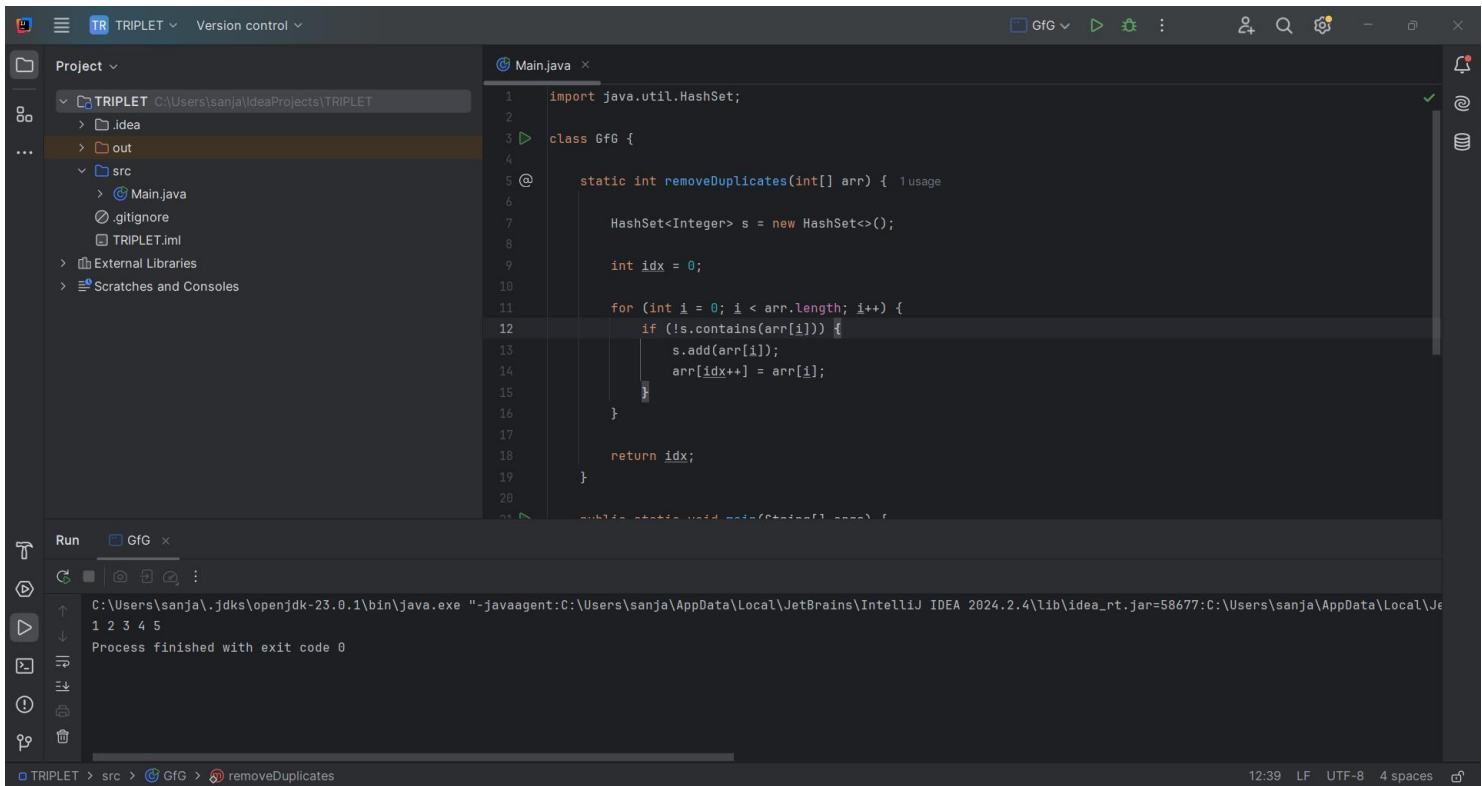
```
public static void main(String[] args) {  
    int[] arr = {1, 2, 2, 3, 4, 4, 4, 5, 5};  
    int newSize = removeDuplicates(arr);
```

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

```
}
```

```
}
```

Output:



```
1 import java.util.HashSet;
2
3 class GfG {
4
5     @
6     static int removeDuplicates(int[] arr) { 1 usage
7
8         HashSet<Integer> s = new HashSet<>();
9
10        int idx = 0;
11
12        for (int i = 0; i < arr.length; i++) {
13            if (!s.contains(arr[i])) {
14                s.add(arr[i]);
15                arr[idx++] = arr[i];
16            }
17        }
18
19        return idx;
20    }
21
22    public static void main(String[] args) {
23        int arr[] = {1, 2, 3, 4, 5};
24        int n = arr.length;
25        removeDuplicates(arr);
26        for (int i = 0; i < n; i++) {
27            System.out.print(arr[i] + " ");
28        }
29        System.out.println();
30    }
31 }
```

Run GfG x

```
C:\Users\sanja\.jdk\openjdk-23.0.1\bin\java.exe "-javaagent:C:\Users\sanja\AppData\Local\JetBrains\IntelliJ IDEA 2024.2.4\lib\idea_rt.jar=58677:C:\Users\sanja\AppData\Local\J
1 2 3 4 5
Process finished with exit code 0
```

6.Maximum Index

Time Complexity : $O(n^2)$

Solution:

```
public class FindMaximum {
```

```
int maxIndexDiff(int arr[], int n)
```

```
{
```

```
int maxDiff = -1;
```

```
int i, j;
```

```
for (i = 0; i < n; ++i) {
```

```
    for (j = n - 1; j > i; --j) {
```

```
        if (arr[j] > arr[i] && maxDiff < (j - i))
```

```
            maxDiff = j - i;
```

```

    }
}

return maxDiff;
}

public static void main(String[] args)
{
    FindMaximum max = new FindMaximum();
    int arr[] = { 9, 2, 3, 4, 5, 6, 7, 8, 18, 0 };
    int n = arr.length;
    int maxDiff = max.maxIndexDiff(arr, n);
    System.out.println(maxDiff);
}
}

```

Output:

The screenshot shows an IDE with the following components:

- Project View:** Shows a project named 'TRIPLET' with a source folder 'src' containing 'FindMaximum.java'.
- Code Editor:** Displays the code for 'FindMaximum.java'. The code defines a class 'FindMaximum' with a method 'maxIndexDiff' that iterates through an array to find the maximum difference between the index of the maximum element and the index of the minimum element. The 'main' method calls this function on the array {9, 2, 3, 4, 5, 6, 7, 8, 18, 0}.
- Run Console:** Shows the command used to run the program: `C:\Users\sanja\jdk\openjdk-23.0.1\bin\java.exe -javaagent:C:\Users\sanja\AppData\Local\JetBrains\IntelliJ IDEA 2024.2.4\lib\idea_rt.jar=58928:C:\Users\sanja\AppData\Local\Je`. The output is 'Process finished with exit code 0'.
- Status Bar:** Shows the current file is 'maxIndexDiff' in the 'src' folder of the 'TRIPLET' project.

7.Wave Array

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

Solution:

```
import java.util.*;

class SortWave
{

    void swap(int arr[], int a, int b)
    {
        int temp = arr[a];
        arr[a] = arr[b];
        arr[b] = temp;
    }

    void sortInWave(int arr[], int n)
    {

        Arrays.sort(arr);

        for (int i=0; i<n-1; i += 2)
            swap(arr, i, i+1);
    }

    public static void main(String args[])
    {
        SortWave ob = new SortWave();
        int arr[] = {10, 90, 49, 2, 1, 5, 23};
        int n = arr.length;
        ob.sortInWave(arr, n);
        for (int i : arr)
            System.out.print(i + " ");
    }
}
```

Output:

