

DSA Programming Practice – 5,6 – KAVYA U – CSBS(III yr) – 2026 BATCH – 22CB024

1. Stock Buy and Sell – Max one Transaction Allowed :

Given an array `prices[]` of length `N`, representing the prices of the stocks on different days, the task is to find the maximum profit possible by buying and selling the stocks on different days when at most one transaction is allowed. Here one transaction means 1 buy + 1 Sell.

Note: Stock must be bought before being sold.

Input: `prices[] = {7, 10, 1, 3, 6, 9, 2}`

Output: 8

Explanation: Buy for price 1 and sell for price 9.

Input: `prices[] = {7, 6, 4, 3, 1}`

Output: 0

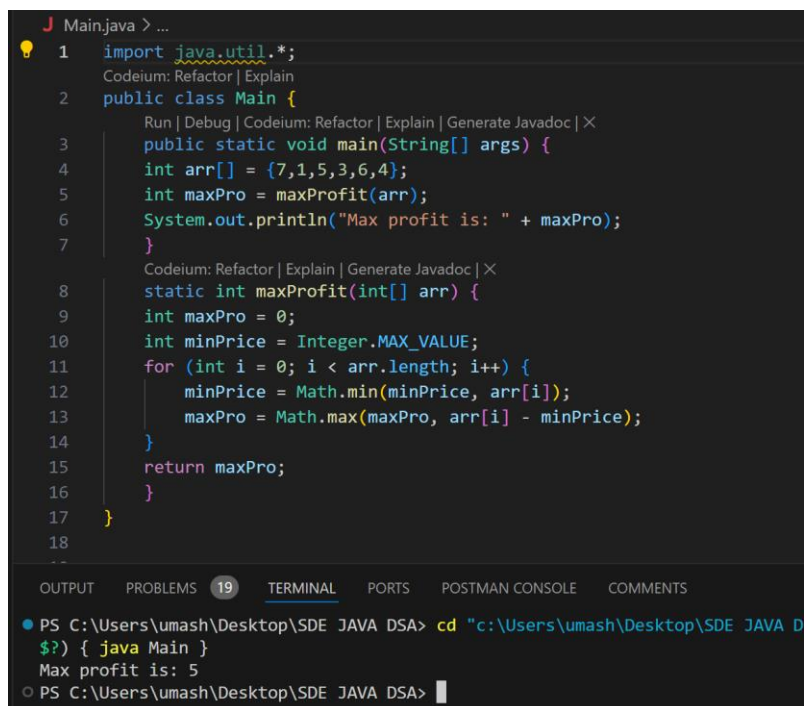
Explanation: Since the array is sorted in decreasing order, 0 profit can be made without making any transaction.

Input: `prices[] = {1, 3, 6, 9, 11}`

Output: 10

Explanation: Since the array is sorted in increasing order, we can make maximum profit by buying at `price[0]` and selling at `price[n-1]`.

Code and Output:



```
1 import java.util.*;
2 public class Main {
3     public static void main(String[] args) {
4         int arr[] = {7,1,5,3,6,4};
5         int maxPro = maxProfit(arr);
6         System.out.println("Max profit is: " + maxPro);
7     }
8     static int maxProfit(int[] arr) {
9         int maxPro = 0;
10        int minPrice = Integer.MAX_VALUE;
11        for (int i = 0; i < arr.length; i++) {
12            minPrice = Math.min(minPrice, arr[i]);
13            maxPro = Math.max(maxPro, arr[i] - minPrice);
14        }
15        return maxPro;
16    }
17 }
18
```

OUTPUT PROBLEMS 19 TERMINAL PORTS POSTMAN CONSOLE COMMENTS

```
PS C:\Users\umash\Desktop\SDE JAVA DSA> cd "c:\Users\umash\Desktop\SDE JAVA DSA"
$?) { java Main }
Max profit is: 5
PS C:\Users\umash\Desktop\SDE JAVA DSA>
```

Time Complexity: $O(N)$

2. Coin Change – Count Ways to Make Sum :

Given an integer array of coins[] of size N representing different types of denominations and an integer sum, the task is to count all combinations of coins to make a given value sum.

Note: Assume that you have an infinite supply of each type of coin.

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

Output: 4

Explanation: there are four solutions: {1, 1, 1, 1}, {1, 1, 2}, {2, 2} and {1, 3}

Input: sum = 10, coins[] = {2, 5, 3, 6}

Output: 5

Explanation: There are five solutions:

{2,2,2,2,2}, {2,2,3,3}, {2,2,6}, {2,3,5} and {5,5}

Input: sum = 10, coins[] = {10}

Output: 1

Explanation: The only is to pick 1 coin of value 10.

Input: sum = 5, coins[] = {4}

Output: 0

Code and Output:

```
coinchange2.java > coinchange2
1  import java.util.Arrays;
   Codeium: Refactor | Explain
2  class coinchange2 {
   Codeium: Refactor | Explain | Generate Javadoc | X
3      static long count(int coins[], int n, int sum)
4      {
5          int dp[] = new int[sum + 1];
6          dp[0] = 1;
7          for (int i = 0; i < n; i++)
8              for (int j = coins[i]; j <= sum; j++)
9                  dp[j] += dp[j - coins[i]];
10
11         return dp[sum];
12     }
   Run | Debug | Codeium: Refactor | Explain | Generate Javadoc | X
13     public static void main(String args[])
14     {
15         int coins[] = { 1, 2, 3 };
16         int n = coins.length;
17         int sum = 5;
18         System.out.println(count(coins, n, sum));
19     }
20 }
```

OUTPUT PROBLEMS 32 TERMINAL PORTS POSTMAN CONSOLE COMMENTS

```
● PS C:\Users\umash\Desktop\SDE JAVA DSA> cd "c:\Users\umash\Desktop\SDE
inchange2.java" ; if ($?) { java coinchange2 }
5
○ PS C:\Users\umash\Desktop\SDE JAVA DSA>
```

Time Complexity: $O(N \cdot \text{sum})$

3. Find first and last positions of an element in a sorted array :

Given a sorted array `arr[]` with possibly duplicate elements, the task is to find indexes of the first and last occurrences of an element `x` in the given array.

Input : `arr[] = {1, 3, 5, 5, 5, 5, 67, 123, 125}`, `x = 5`

Output : First Occurrence = 2

Last Occurrence = 5

Input : `arr[] = {1, 3, 5, 5, 5, 5, 7, 123, 125 }`, `x = 7`

Output : First Occurrence = 6

Last Occurrence = 6

Code and Output:

```
1  import java.util.ArrayList;
   Codeium: Refactor | Explain
2  public class index {
   Codeium: Refactor | Explain | Generate Javadoc | X
3      public static int first(ArrayList list, int x)
4      {
5          return list.indexOf(x);
6      }
   Codeium: Refactor | Explain | Generate Javadoc | X
7      public static int last(ArrayList list, int x)
8      {
9          return list.lastIndexOf(x);
10     }
   Run | Debug | Codeium: Refactor | Explain | Generate Javadoc | X
11     public static void main(String[] args)
12     {
13         int arr[] = { 1, 2, 2, 2, 2, 3, 4, 7, 8, 8 };
14         ArrayList<Integer> clist = new ArrayList<>();
15         for (int i : arr)
16             clist.add(i);
17         int x = 8;
18         System.out.println("First Occurrence = "
19             + first(clist, x));
20         System.out.println("Last Occurrence = "
21             + last(clist, x));
22     }
23 }
```

OUTPUT PROBLEMS 30 TERMINAL PORTS POSTMAN CONSOLE COMMENTS

```
PS C:\Users\umash\Desktop\SDE JAVA DSA> cd "c:\Users\umash\Desktop\SDE JAVA"
($?) { java index }
First Occurrence = 8
Last Occurrence = 9
PS C:\Users\umash\Desktop\SDE JAVA DSA>
```

Time Complexity: $O(N)$

4. Find the transition point in a binary array :

Given a sorted array containing only numbers 0 and 1, the task is to find the transition point efficiently. The transition point is the point where "0" ends and "1" begins.

Input: 0 0 0 1 1

Output: 3

Explanation: Index of first 1 is 3

Input: 0 0 0 0 1 1 1 1

Output: 4

Explanation: Index of first 1 is 4

Code and Output:

```
1  class transitionpoint {
2      static int findTransitionPoint(int arr[], int n)
3      {
4          int lb = 0, ub = n - 1;
5          while (lb <= ub) {
6              int mid = (lb + ub) / 2;
7              if (arr[mid] == 0)
8                  lb = mid + 1;
9              else if (arr[mid] == 1) {
10                 if (mid == 0
11                     || (mid > 0 &&
12                         arr[mid - 1] == 0)) {
13                     return mid;
14                 }
15                 ub = mid - 1;
16             }
17         }
18         return -1;
19     }
20
21     public static void main(String args[])
22     {
23         int arr[] = { 0, 0, 0, 0, 1, 1 };
24         int point = findTransitionPoint(arr, arr.length);
25         System.out.println(
26             point >= 0 ? "Transition point is " + point
27             : "There is no transition point");
28     }
29 }
```

Run | Debug | Codeium: Refactor | Explain | Generate Javadoc | X

31 TERMINAL

PS C:\Users\umash\Desktop\SDE JAVA DSA> cd "c:\Users\umash\Desktop\SDE JAVA DSA\"
ansitionpoint.java } ; if (\$?) { java transitionpoint }
Transition point is 4

Time Complexity: $O(\log N)$

5. Find the first repeating element in an array of integers :

Given an array of integers arr[], The task is to find the index of first repeating element in it i.e. the element that occurs more than once and whose index of the first occurrence is the smallest.

Input: arr[] = {10, 5, 3, 4, 3, 5, 6}

Output: 5

Explanation: 5 is the first element that repeats

Input: arr[] = {6, 10, 5, 4, 9, 120, 4, 6, 10}

Output: 6

Explanation: 6 is the first element that repeats

Code and Output:

```
Codeium: Refactor | Explain
2  class duplicateno {
    Codeium: Refactor | Explain | Generate Javadoc | X
3      static void printFirstRepeating(int arr[])
4      {
5          int min = -1;
6          HashSet<Integer> set = new HashSet<>();
7          for (int i = arr.length - 1; i >= 0; i--) {
8              if (set.contains(arr[i]))
9                  min = i;
10             set.add(arr[i]);
11         }
12         if (min != -1)
13             System.out.println(
14                 "The first repeating element is "
15                 + arr[min]);
16         else
17             System.out.println(
18                 x:"There are no repeating elements");
19     }
    Run | Debug | Codeium: Refactor | Explain | Generate Javadoc | X
20     public static void main(String[] args)
21         throws java.lang.Exception
22     {
23         int arr[] = { 10, 5, 3, 4, 3, 5, 6 };
24         printFirstRepeating(arr);
25     }
26 }

OUTPUT  PROBLEMS 30  TERMINAL  PORTS  POSTMAN CONSOLE  COMMENTS
● PS C:\Users\umash\Desktop\SDE JAVA DSA> cd "c:\Users\umash\Desktop\SDE JAVA DSA\
plicateno.java" ; if ($?) { java duplicateno }
The first repeating element is 5
○ PS C:\Users\umash\Desktop\SDE JAVA DSA>
```

Time Complexity: $O(N)$

6. Remove duplicates from sorted array :

Given a sorted array `arr[]` of size `N`, the task is to remove the duplicate elements from the array. We need keep order of the remaining distinct elements as it was in the original array.

Input: `arr[] = {2, 2, 2, 2, 2}`

Output: `arr[] = {2}`

Explanation: All the elements are 2, So only keep one instance of 2.

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

Output: `arr[] = {1, 2, 3, 4, 5}`

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

Output : `arr[] = {1, 2, 3}`

Explanation : No change as all elements are distinct

Code and Output:

```
removeduplicates.java > removeduplicates > main(String[])
Codeium: Refactor | Explain
1 class removeduplicates {
Codeium: Refactor | Explain | Generate Javadoc | X
2     static int removeDuplicates(int[] arr) {
3         int n = arr.length;
4         if (n <= 1)
5             return n;
6         int idx = 1;
7         for (int i = 1; i < n; i++) {
8             if (arr[i] != arr[i - 1]) {
9                 arr[idx++] = arr[i];
10            }
11        }
12        return idx;
13    }
Run | Debug | Codeium: Refactor | Explain | Generate Javadoc | X
14    public static void main(String[] args) {
15        int[] arr = {1, 2, 2, 3, 4, 4, 4, 5, 5};
16        int newSize = removeDuplicates(arr);
17        for (int i = 0; i < newSize; i++) {
18            System.out.print(arr[i] + " ");
19        }
20    }
21 }
22

OUTPUT  PROBLEMS 30  TERMINAL  PORTS  POSTMAN CONSOLE  COMMENTS
● PS C:\Users\umash\Desktop\SDE JAVA DSA> cd "c:\Users\umash\Desktop\SDE JAVA DSA"
removeduplicates.java } ; if ($?) { java removeduplicates }
1 2 3 4 5
○ PS C:\Users\umash\Desktop\SDE JAVA DSA>
```

Time Complexity: $O(N)$

7. Maximum Index:

Given an array `arr[]` of N positive integers. The task is to find the maximum of $j - i$ subjected to the constraint of `arr[i] <= arr[j]`.

Input: {34, 8, 10, 3, 2, 80, 30, 33, 1}

Output: 6 ($j = 7, i = 1$)

Input: {9, 2, 3, 4, 5, 6, 7, 8, 18, 0}

Output: 8 ($j = 8, i = 0$)

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

Output: 5 ($j = 5, i = 0$)

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

Output: 0

Code and Output:

```
1  import java.io.*;
2  import java.util.*;
Codeium: Refactor | Explain
3  class maxindex{
Codeium: Refactor | Explain | Generate Javadoc | X
4  static int maxIndexDiff(ArrayList<Integer> arr, int n)
5  {
6      Map<Integer,
7      ArrayList<Integer>> hashmap = new HashMap<Integer, ArrayList<Integer>>();
8      for(int i = 0; i < n; i++)
9      {
10         if(hashmap.containsKey(arr.get(i)))
11         {
12             hashmap.get(arr.get(i)).add(i);
13         }
14         else
15         {
16             hashmap.put(arr.get(i), new ArrayList<Integer>());
17             hashmap.get(arr.get(i)).add(i);
18         }
19     }
20     Collections.sort(arr);
21     Collections.sort(arr);
22     int maxDiff = Integer.MIN_VALUE;
23     int temp = n;
24     for(int i = 0; i < n; i++)
25     {
26         if (temp > hashmap.get(arr.get(i)).get(index:0))
27         {
28             temp = hashmap.get(arr.get(i)).get(index:0);
29         }
30         maxDiff = Math.max(maxDiff,
31         hashmap.get(arr.get(i)).get(
32         hashmap.get(arr.get(i)).size() - 1) - temp);
33     }
34     return maxDiff;
35 }
Run | Debug | Codeium: Refactor | Explain | Generate Javadoc | X
36 public static void main(String[] args)
37 {
38     int n = 9;
39     ArrayList<Integer> arr = new ArrayList<Integer>(
40     Arrays.asList(...a:34, 8, 10, 3, 2, 80, 30, 33, 1));
41     int ans = maxIndexDiff(arr, n);
42     System.out.println("The maxIndexDiff is : " + ans);
43 }
44 }
45
```

OUTPUT PROBLEMS 31 TERMINAL PORTS POSTMAN CONSOLE COMMENTS

The maxIndexDiff is : 6

PS C:\Users\umash\Desktop\SDE JAVA DSA>

Time Complexity: $O(\log nN)$

8. Sort an array in wave form :

Given an unsorted array of integers, sort the array into a wave array. An array $arr[0..n-1]$ is sorted in wave form if: $arr[0] \geq arr[1] \leq arr[2] \geq arr[3] \leq arr[4] \geq \dots$

Input: $arr[] = \{10, 5, 6, 3, 2, 20, 100, 80\}$

Output: $arr[] = \{10, 5, 6, 2, 20, 3, 100, 80\}$

Explanation:

here you can see $\{10, 5, 6, 2, 20, 3, 100, 80\}$ first element is larger than the second and the same thing is repeated again and again. large element – small element-large element -small element and so on .it can be small element-larger element – small element-large element -small element too. all you need to maintain is the up-down fashion which represents a wave. there can be multiple answers.

Input: $arr[] = \{20, 10, 8, 6, 4, 2\}$

Output: $arr[] = \{20, 8, 10, 4, 6, 2\}$

Code and Output:

```
1  import java.util.*;
   Codeium: Refactor | Explain
2  class wavesort{
   Codeium: Refactor | Explain | Generate Javadoc | X
3      void swap(int arr[], int a, int b)
4      {
5          int temp = arr[a];
6          arr[a] = arr[b];
7          arr[b] = temp;
8      }
   Codeium: Refactor | Explain | Generate Javadoc | X
9      void sortInWave(int arr[], int n)
10     {
11         Arrays.sort(arr);
12         for (int i=0; i<n-1; i += 2)
13             swap(arr, i, i+1);
14     }
   Run | Debug | Codeium: Refactor | Explain | Generate Javadoc | X
15     public static void main(String args[])
16     {
17         wavesort ob = new wavesort();
18         int arr[] = {10, 90, 49, 2, 1, 5, 23};
19         int n = arr.length;
20         ob.sortInWave(arr, n);
21         for (int i : arr)
22             System.out.print(i + " ");
23     }
24 }
25 }
```

OUTPUT PROBLEMS 31 TERMINAL PORTS POSTMAN CONSOLE COMMENTS

```
2 1 10 5 49 23 90
PS C:\Users\umash\Desktop\SDE JAVA DSA>
```

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

9. Bubble Sort:

Code and Output:

```
bubblesort.java > bubblesort > printArray(int[], int)
1  import java.io.*;
   Codeium: Refactor | Explain
2  class bubblesort {
   Codeium: Refactor | Explain | Generate Javadoc | X
3      static void bubbleSort(int arr[], int n){
4          int i, j, temp;
5          boolean swapped;
6          for (i = 0; i < n - 1; i++) {
7              swapped = false;
8              for (j = 0; j < n - i - 1; j++) {
9                  if (arr[j] > arr[j + 1]) {
10                     temp = arr[j];
11                     arr[j] = arr[j + 1];
12                     arr[j + 1] = temp;
13                     swapped = true;
14                 }
15             }
16             if (swapped == false)
17                 break;
18         }
19     }
   Codeium: Refactor | Explain | Generate Javadoc | X
20     static void printArray(int arr[], int size){
21         int i;
22         for (i = 0; i < size; i++)
23             System.out.print(arr[i] + " ");
24         System.out.println();
25     }
   ⚡
26
27     Run | Debug | Codeium: Refactor | Explain | Generate Javadoc | X
28     public static void main(String args[]){
29         int arr[] = { 64, 34, 25, 12, 22, 11, 90 };
30         int n = arr.length;
31         bubbleSort(arr, n);
32         System.out.println(x:"Sorted array: ");
33         printArray(arr, n);
34     }
35
36     OUTPUT  PROBLEMS 34 TERMINAL PORTS POSTMAN CONSOLE COMMENTS
37
38     Sorted array:
39     11 12 22 25 34 64 90
40     PS D:\SDE JAVA DSA>
```

Time Complexity: $O(N^2)$

10. Quick Sort:

Code and Output:

```
1  import java.util.Arrays;
   Codeium: Refactor | Explain
2  class quicksort {
   Codeium: Refactor | Explain | Generate Javadoc | X
3      static int partition(int[] arr, int low, int high) {
4          int pivot = arr[high];
5          int i = low - 1;
6          for (int j = low; j <= high - 1; j++) {
7              if (arr[j] < pivot) {
8                  i++;
9                  swap(arr, i, j);
10             }
11         }
12         swap(arr, i + 1, high);
13         return i + 1;
14     }
   Codeium: Refactor | Explain | Generate Javadoc | X
15     static void swap(int[] arr, int i, int j) {
16         int temp = arr[i];
17         arr[i] = arr[j];
18         arr[j] = temp;
19     }
   Codeium: Refactor | Explain | Generate Javadoc | X
20     static void quickSort(int[] arr, int low, int high) {
21         if (low < high) {
22             int pi = partition(arr, low, high);
23             quickSort(arr, low, pi - 1);
24             quickSort(arr, pi + 1, high);
25         }
26     }
```

```
Run | Debug | Codeium: Refactor | Explain | Generate Javadoc | X
28     public static void main(String[] args) {
29         int[] arr = {10, 7, 8, 9, 1, 5};
30         int n = arr.length;
31
32         quickSort(arr, low:0, n - 1);
33
34         for (int val : arr) {
35             System.out.print(val + " ");
36         }
37     }
38 }
```

OUTPUT PROBLEMS 35 TERMINAL PORTS POSTMAN CONSOLE COMMENTS

```
1 5 7 8 9 10
PS D:\SDE JAVA DSA>
```

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

11. Non-repeating Character:

Given a string *s* of lowercase English letters, the task is to find the first non-repeating character. If there is no such character, return '\$'.

Input: *s* = "geeksforgeeks"

Output: 'f'

Explanation: 'f' is the first character in the string which does not repeat.

Input: *s* = "racecar"

Output: 'e'

Explanation: 'e' is the only character in the string which does not repeat.

Input: "aabbccc"

Output: '\$'

Explanation: All the characters in the given string are repeating.

Code Output:

```
1  import java.util.Arrays;
   Codeium: Refactor | Explain
2  class nonrepeatchar {
3      static final int MAX_CHAR = 26;
   Codeium: Refactor | Explain | Generate Javadoc | X
4      static char nonRepeatingChar(String s) {
5          int[] freq = new int[MAX_CHAR];
6          for (char c : s.toCharArray())
7              freq[c - 'a']++;
8          for (int i = 0; i < s.length(); ++i) {
9              if (freq[s.charAt(i) - 'a'] == 1)
10                 return s.charAt(i);
11          }
12          return '$';
13     }
   Run | Debug | Codeium: Refactor | Explain | Generate Javadoc | X
14     public static void main(String[] args) {
15         String s = "racecar";
16
17         System.out.println(nonRepeatingChar(s));
18     }
19 }
20
```

OUTPUT PROBLEMS 36 TERMINAL PORTS POSTMAN CONSOLE COMMENTS

```
● PS D:\SDE JAVA DSA> cd "d:\SDE JAVA DSA\" ; if ($?) { javac nonrepeatch
epeatchar }
e
○ PS D:\SDE JAVA DSA>
```

Time Complexity: O(N)

12. Edit Distance :

Given two strings s1 and s2 of lengths m and n respectively and below operations that can be performed on s1. Find the minimum number of edits (operations) to convert 's1' into 's2'.

- Insert: Insert any character before or after any index of s1
- Remove: Remove a character of s1
- Replace: Replace a character at any index of s1 with some other character.

Note: All of the above operations are of equal cost.

Input: s1 = "geek", s2 = "gesek"

Output: 1

Explanation: We can convert s1 into s2 by inserting a 's' between two consecutive 'e' in s2.

Input: s1 = "cat", s2 = "cut"

Output: 1

Explanation: We can convert s1 into s2 by replacing 'a' with 'u'.

Input: s1 = "sunday", s2 = "saturday"

Output: 3

Explanation: Last three and first characters are same. We basically need to convert "un" to "atur". This can be done using below three operations. Replace 'n' with 'r', insert t, insert a

Code and Output:

```
Codeium: Refactor | Explain
1 public class editdist {
  Codeium: Refactor | Explain | Generate Javadoc | X
2 private static int minDisRec(String s1, String s2, int m, int n, int[][] memo) {
3     if (m == 0) return n;
4     if (n == 0) return m;
5     if (memo[m][n] != -1) return memo[m][n];
6     if (s1.charAt(m - 1) == s2.charAt(n - 1)) {
7         memo[m][n] = minDisRec(s1, s2, m - 1, n - 1, memo);
8     } else {
9         int insert = minDisRec(s1, s2, m, n - 1, memo);
10        int remove = minDisRec(s1, s2, m - 1, n, memo);
11        int replace = minDisRec(s1, s2, m - 1, n - 1, memo);
12        memo[m][n] = 1 + Math.min(insert, Math.min(remove, replace));
13    }
14    return memo[m][n];
15 }
  Codeium: Refactor | Explain | Generate Javadoc | X
16 public static int minDis(String s1, String s2) {
17     int m = s1.length(), n = s2.length();
18     int[][] memo = new int[m + 1][n + 1];
19     for (int i = 0; i <= m; i++) {
20         for (int j = 0; j <= n; j++) {
21             memo[i][j] = -1;
22         }
23     }
24     return minDisRec(s1, s2, m, n, memo);
25 }
  Codeium: Refactor | Explain | Generate Javadoc | X
26 public static void main(String[] args) {
27     String s1 = "GEEXSFRGEEKKS";
28     String s2 = "GEEKSFORGEEKS";
29     System.out.println(minDis(s1, s2));
30 }
31 }
```

OUTPUT PROBLEMS 36 TERMINAL PORTS POSTMAN CONSOLE COMMENTS

3

Time Complexity: $O(M*N)$

13. [Form the Largest Number:](#)

Given an array of strings arr[] of length n, where every string representing a non-negative integers, the task is to arrange them in a manner such that after concatenating them in order, it results in the largest possible number. Since the result may be very large, return it as a string.

Input: n = 5, arr[] = {"3", "30", "34", "5", "9"}

Output: "9534330"

Explanation: Given numbers are {"3", "30", "34", "5", "9"}, the arrangement "9534330" gives the largest value.

Input: n = 4, arr[] = {"54", "546", "548", "60"}

Output: "6054854654"

Explanation: Given numbers are {"54", "546", "548", "60"}, the arrangement "6054854654" gives the largest value.

Code and Output:

```
1  import java.util.Arrays;
2  import java.util.Comparator;
   Codeium: Refactor | Explain
3  public class largestnum {
   Codeium: Refactor | Explain | Generate Javadoc | X
4      public static String largestNumber(String[] arr)
5      {
6          Comparator<String> myCompare
7              = (X, Y) -> (X + Y).compareTo(Y + X);
8          Arrays.sort(arr, myCompare.reversed());
9          if (arr[0].equals(anObject:"0")) {
10             return "0";
11         }
12         StringBuilder result = new StringBuilder();
13         for (String num : arr) {
14             result.append(num);
15         }
16         return result.toString();
17     }
   Run | Debug | Codeium: Refactor | Explain | Generate Javadoc | X
18     public static void main(String[] args)
19     {
20         String[] arr1 = { "3", "30", "34", "5", "9" };
21         System.out.println(
22             largestNumber(arr1));
23     }
24 }
25
```

OUTPUT PROBLEMS 36 TERMINAL PORTS POSTMAN CONSOLE COMMENTS

9534330

PS D:\SDE JAVA DSA>

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

14. Merge Sort:

Code and Output:

```

2  class mergesort {
3      static void merge(int arr[], int l, int m, int r)
4      {
5          int n1 = m - l + 1;
6          int n2 = r - m;
7          int L[] = new int[n1];
8          int R[] = new int[n2];
9          for (int i = 0; i < n1; ++i)
10             L[i] = arr[l + i];
11          for (int j = 0; j < n2; ++j)
12             R[j] = arr[m + 1 + j];
13          int i = 0, j = 0;
14          int k = l;
15          while (i < n1 && j < n2) {
16              if (L[i] <= R[j]) {
17                  arr[k] = L[i];
18                  i++;
19              }
20              else {
21                  arr[k] = R[j];
22                  j++;
23              }
24              k++;
25          }
26          while (i < n1) {
27              arr[k] = L[i];
28              i++;
29              k++;
30          }
31          while (j < n2) {
32              arr[k] = R[j];
33              j++;
34              k++;
35          }
36      }

```

```

37  static void sort(int arr[], int l, int r)
38  {
39      if (l < r) {
40          int m = l + (r - l) / 2;
41          sort(arr, l, m);
42          sort(arr, m + 1, r);
43          merge(arr, l, m, r);
44      }
45  }

```

```

46  static void printArray(int arr[])
47  {
48      int n = arr.length;
49      for (int i = 0; i < n; ++i)
50          System.out.print(arr[i] + " ");
51      System.out.println();
52  }

```

```

53  public static void main(String args[])
54  {
55      int arr[] = { 12, 11, 13, 5, 6, 7 };
56      System.out.println(x:"Given array is");
57      printArray(arr);
58      sort(arr, l:0, arr.length - 1);
59      System.out.println(x:"\nSorted array is");
60      printArray(arr);
61  }

```

OUTPUT PROBLEMS 38 TERMINAL PORTS POSTMAN CONSOLE COMMENTS

Sorted array is
5 6 7 11 12 13

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

15. Ternary Search:

Code and Output:

```
class ternarysearch {
    Codeium: Refactor | Explain | Generate Javadoc | X
    static int ternarySearch(int l, int r, int key, int ar[])
    {
        if (r >= 1) {
            int mid1 = l + (r - 1) / 3;
            int mid2 = r - (r - 1) / 3;
            if (ar[mid1] == key) {
                return mid1;
            }
            if (ar[mid2] == key) {
                return mid2;
            }
            if (key < ar[mid1]) {
                return ternarySearch(l, mid1 - 1, key, ar);
            }
            else if (key > ar[mid2]) {
                return ternarySearch(mid2 + 1, r, key, ar);
            }
            else {
                return ternarySearch(mid1 + 1, mid2 - 1, key, ar);
            }
        }
        return -1;
    }
}
```

```
25 public static void main(String args[])
26 {
27     int l, r, p, key;
28     int ar[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
29     l = 0;
30     r = 9;
31     key = 5;
32     p = ternarySearch(l, r, key, ar);
33     System.out.println("Index of " + key + " is " + p);
34     key = 8;
35     p = ternarySearch(l, r, key, ar);
36     System.out.println("Index of " + key + " is " + p);
37 }
38 }
```

OUTPUT PROBLEMS 38 TERMINAL PORTS POSTMAN CONSOLE COMMENTS

Index of 8 is 7

PS D:\SDE JAVA DSA>

Time Complexity : $O(2 * \log_3 n)$

16. Interpolation Search:

Code and Output:

```
2  class interpolationsearch {
3      public static int interpolationSearch(int arr[], int lo,int hi, int x)
4      {
5          int pos;
6          if (lo <= hi && x >= arr[lo] && x <= arr[hi]) {
7              pos = lo
8                  + ((hi - lo) / (arr[hi] - arr[lo]))
9                  * (x - arr[lo]);
10             if (arr[pos] == x)
11                 return pos;
12             if (arr[pos] < x)
13                 return interpolationSearch(arr, pos + 1, hi, x);
14             if (arr[pos] > x)
15                 return interpolationSearch(arr, lo, pos - 1, x);
16         }
17         return -1;
18     }
19     Run | Debug | Codeium: Refactor | Explain | Generate Javadoc | X
20     public static void main(String[] args)
21     {
22         int arr[] = { 10, 12, 13, 16, 18, 19, 20, 21, 22, 23, 24, 33, 35, 42, 47 };
23         int n = arr.length;
24         int x = 18;
25         int index = interpolationSearch(arr, lo:0, n - 1, x);
26         if (index != -1)
27             System.out.println("Element found at index "+ index);
28         else
29             System.out.println(x:"Element not found.");
30     }
31 }
```

OUTPUT PROBLEMS 39 TERMINAL PORTS POSTMAN CONSOLE COMMENTS

Element found at index 4
PS D:\SDE JAVA DSA>

Time Complexity : $O(\log_2(\log_2 n))$