

**JAVA PRACTICE SHEET (02/02/2026 – 07/02/2026)**

**1.**

Given an array `arr[]` of positive integers, where each value represents the number of chocolates in a packet. Each packet can have a variable number of chocolates. There are `m` students, the task is to distribute chocolate packets among `m` students such that -

- i. Each student gets exactly one packet.
- ii. The difference between maximum number of chocolates given to a student and minimum number of chocolates given to a student is minimum and return that minimum possible difference.

Examples:

Input: `arr = [3, 4, 1, 9, 56, 7, 9, 12]`, `m = 5`

Output: 6

Explanation: The minimum difference between maximum chocolates and minimum chocolates is  $9 - 3 = 6$  by choosing following `m` packets :`[3, 4, 9, 7, 9]`.

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

Output: 2

Explanation: The minimum difference between maximum chocolates and minimum chocolates is  $4 - 2 = 2$  by choosing following `m` packets :`[3, 2, 4]`.

Input: `arr = [3, 4, 1, 9, 56]`, `m = 5`

Output: 55

Explanation: With 5 packets for 5 students, each student will receive one packet, so the difference is  $56 - 1 = 55$ .

**Sol**

Output Window

Compilation Results Custom Input Y.O.G.I. (AI Bot)

Problem Solved Successfully ✓

Test Cases Passed: 1112 / 1112

Attempts: Correct / Total: 1 / 1

Accuracy: 100%

Points Scored: 2 / 2

Your Total Score: 47 ↑

Time Taken: 0.83

Solve Next: Bubble Sort Floor in a Sorted Array Closest Triplet

Stay Ahead With: Build 21 Projects in 21 Days

```
1 // User function Template for Java
2 import java.util.*;
3 class Solution {
4     public int findMinDiff(ArrayList<Integer> arr, int m) {
5         // your code here
6         int n = arr.size();
7
8         if (m == 0 || m > n) return 0;
9
10        Collections.sort(arr);
11
12        int minDiff = Integer.MAX_VALUE;
13
14        for (int i = 0; i + m - 1 < n; i++) {
15            int diff = arr.get(i + m - 1) - arr.get(i);
16            minDiff = Math.min(minDiff, diff);
17        }
18
19        return minDiff;
20    }
21 }
22
23
24
25
26
27
```

Custom Input Compile & Run Submit

2.

Given a number  $x$  and an array of integers  $arr$ , find the smallest subarray with sum greater than the given value. If such a subarray do not exist return 0 in that case.

Examples:

Input:  $x = 51$ ,  $arr[] = [1, 4, 45, 6, 0, 19]$

Output: 3

Explanation: Minimum length subarray is  $[4, 45, 6]$

Input:  $x = 100$ ,  $arr[] = [1, 10, 5, 2, 7]$

Output: 0

Explanation: No subarray exist

Sol

**Output Window**

Compilation Results Custom Input Y.O.G.I. (AI Bot)

**Problem Solved Successfully**

Suggest Feedback

Test Cases Passed  
**1112 / 1112**

Attempts: Correct / Total  
**1 / 1**

Accuracy: 100%

Points Scored   
**2 / 2**

Your Total Score: 49

Time Taken  
**0.4**

**Solve Next**

Sorted subsequence of size 3 Array Duplicates Two Sum - Pair with Given Sum

**Stay Ahead With:**

**Build 21 Projects in 21 Days**  
Build real-world ML, Deep Learning & Gen AI projects

```

1 class Solution {
2     public static int smallestSubWithSum(int x, int[] arr) {
3         // Your code goes here
4         int n = arr.length;
5         int start = 0, sum = 0;
6         int minLen = Integer.MAX_VALUE;
7
8         for (int end = 0; end < n; end++) {
9             sum += arr[end];
10
11             while (sum > x) {
12                 minLen = Math.min(minLen, end - start + 1);
13                 sum -= arr[start];
14                 start++;
15             }
16
17             return (minLen == Integer.MAX_VALUE) ? 0 : minLen;
18         }
19     }
20 }
21
22
23

```

Custom Input Compile & Run Submit

3.

Given an array and a range a, b. The task is to partition the array around the range such that the array is divided into three parts.

- 1) All elements smaller than a come first.
- 2) All elements in range a to b come next.
- 3) All elements greater than b appear in the end.

The individual elements of three sets can appear in any order. You are required to return the modified array.

Note: The generated output is true if you modify the given array successfully.

Otherwise false.

Geeky Challenge: Solve this problem in  $O(n)$  time complexity.

Examples:

Input: arr[] = [1, 2, 3, 3, 4], a = 1, b = 2

Output: true

Explanation: One possible arrangement is: {1, 2, 3, 3, 4}. If you return a valid arrangement, output will be true.

Input: arr[] = [1, 4, 3, 6, 2, 1], a = 1, b = 3

Output: true

Explanation: One possible arrangement is: {1, 3, 2, 1, 4, 6}. If you return a valid

arrangement, output will be true.

## Sol

The screenshot shows a coding platform interface. On the left, the 'Output Window' is open, displaying 'Compilation Results' for a custom input 'Y.O.G.I. (AI Bot)'. It indicates the 'Problem Solved Successfully' with a green checkmark. Below this, it shows 'Test Cases Passed: 1111 / 1111', 'Attempts: Correct / Total: 1 / 1', 'Accuracy: 100%', 'Points Scored: 2 / 2', and 'Time Taken: 0.21'. There are also buttons for 'Solve Next' and 'Stay Ahead With: Build 21 Projects in 21 Days'. On the right, the code editor shows a C++ solution for a three-way partition problem. The code defines a class 'Solution' with a method 'threeWayPartition' that takes an array 'arr', a number 'a', and a number 'b'. It uses a while loop to partition the array around the range [a, b].

## 4.

Given an array `arr` and a number `k`. One can apply a swap operation on the array any number of times, i.e. choose any two index `i` and `j` ( $i < j$ ) and swap `arr[i]`, `arr[j]`. Find the minimum number of swaps required to bring all the numbers less than or equal to `k` together, i.e. make them a contiguous subarray.

Examples :

Input: `arr[] = [2, 1, 5, 6, 3]`, `k = 3`

Output: 1

Explanation: To bring elements 2, 1, 3 together, swap index 2 with 4 (0-based indexing), i.e. element `arr[2] = 5` with `arr[4] = 3` such that final array will be- `arr[] = [2, 1, 3, 6, 5]`

Input: `arr[] = [2, 7, 9, 5, 8, 7, 4]`, `k = 6`

Output: 2

Explanation: To bring elements 2, 5, 4 together, swap index 0 with 2 (0-based indexing) and index 4 with 6 (0-based indexing) such that final array will be- `arr[] = [9, 7, 2, 5, 4, 7, 8]`

Input: `arr[] = [2, 4, 5, 3, 6, 1, 8]`, `k = 6`

Output: 0

## Sol

The screenshot displays a coding platform interface. On the left, the 'Output Window' shows 'Compilation Results' for a custom input 'Y.O.G.J. (AI Bot)'. It indicates the problem was solved successfully with 1112/1112 test cases passed, 1/1 attempts, 100% accuracy, 4/4 points scored, and a time taken of 0.45 seconds. Below this, it suggests solving the next problem, 'Rearrange Array Alternately', and promotes a 'Build 21 Projects in 21 Days' challenge. On the right, the solution code for a Java function 'minSwap' is shown. The code aims to find the minimum number of swaps required to sort an array. It uses two arrays, 'good' and 'bad', to track elements in their correct or incorrect positions relative to a sorted state. The function returns the minimum of 'good' and 'bad' counts.

```
1 // User function Template for Java
2 class Solution {
3 // Function for finding maximum and value pair
4 int minSwap(int[] arr, int k) {
5 // Complete the function
6 // User function Template for Java
7 int n = arr.length;
8 int good = 0;
9 for (int i = 0; i < n; i++) {
10 if (arr[i] <= k) {
11 good++;
12 }
13 }
14 if (good == 0 || good == n) {
15 return 0;
16 }
17 int bad = 0;
18 for (int i = 0; i < good; i++) {
19 if (arr[i] > k) {
20 bad++;
21 }
22 }
23 int ans = bad;
24 int i = 0, j = good;
25 while (j < n) {
26 if (arr[i] > k) {
27 bad--;
28 }
29 if (arr[j] > k) {
30 bad++;
31 }
32 }
33 ans = Math.min(ans, bad);
34 i++;
35 j++;
36 }
37 return ans;
38 }
39 }
40 }
```

5.

Given an array `arr[]` of positive integers. Return true if all the array elements are palindrome otherwise, return false.

Examples:

Input: `arr[] = [111, 222, 333, 444, 555]`

Output: true

Explanation:

`arr[0] = 111`, which is a palindrome number.

`arr[1] = 222`, which is a palindrome number.

`arr[2] = 333`, which is a palindrome number.

`arr[3] = 444`, which is a palindrome number.

`arr[4] = 555`, which is a palindrome number.

As all numbers are palindrome so This will return true.

Input: `arr[] = [121, 131, 20]`

Output: false

Explanation: 20 is not a palindrome hence the output is false.

## Sol

**Output Window**

Compilation Results Custom Input Y.O.G.I. (AI Bot)

**Problem Solved Successfully** ✓ [Suggest Feedback](#)

Test Cases Passed: **1115 / 1115**

Attempts: Correct / Total: **1 / 1**  
Accuracy: 100%

Points Scored: **1 / 1**  
Your Total Score: 50 ↑

Time Taken: **0.12**

**Solve Next**

[Palindrome Array](#) [Smaller and Larger](#) [Find the left over element](#)

**Stay Ahead With:**

**Build 21 Projects in 21 Days**  
Build real-world ML, Deep Learning & Gen AI projects

```
1 /*Complete the Function below*/
2 class Solution {
3     public static boolean isPalinArray(int[] arr) {
4         // add code here.
5         for (int num : arr) {
6             if (!isPalindrome(num)) {
7                 return false;
8             }
9         }
10        return true;
11    }
12
13    private static boolean isPalindrome(int num) {
14        int original = num;
15        int reverse = 0;
16
17        while (num > 0) {
18            reverse = reverse * 10 + num % 10;
19            num /= 10;
20        }
21        return original == reverse;
22    }
23 }
24
25
```

Custom Input [Compile & Run](#) [Submit](#)

6.

Given an array `arr[]` of integers, calculate the median.

Examples:

Input: `arr[] = [90, 100, 78, 89, 67]`

Output: 89

Explanation: After sorting the array middle element is the median

Input: `arr[] = [56, 67, 30, 79]`

Output: 61.5

Explanation: In case of even number of elements, average of two middle elements is the median.

Input: `arr[] = [1, 2]`

Output: 1.5

Explanation: The average of both elements will result in 1.5.

## Sol

**Output Window**

Compilation Results Custom Input Y.O.G.I. (AI Bot)

**Problem Solved Successfully** ✓ [Suggest Feedback](#)

Test Cases Passed  
**1115 / 1115**

Attempts: Correct / Total  
**1 / 1**  
Accuracy: 100%

Points Scored  
**1 / 1**  
Your Total Score: 51 ↑

Time Taken  
**0.55**

**Solve Next**

[Multiply Array](#) [Mean of an Array](#) [Greatest of three numbers](#)

**Stay Ahead With:**

**Build 21 Projects in 21 Days**  
Build real-world ML, Deep Learning & Gen AI projects

```

1 import java.util.*;
2 class Solution {
3     public double findMedian(int[] arr) {
4         // Code here.
5         int n = arr.length;
6         Arrays.sort(arr);
7         if (n % 2 != 0) {
8             return arr[n / 2];
9         }
10        else {
11            return (arr[n / 2 - 1] + arr[n / 2]) / 2.0;
12        }
13    }
14 }

```

Custom Input [Compile & Run](#) [Submit](#)

7.

You are given a rectangular matrix `mat[][]` of size `n x m`, and your task is to return an array while traversing the matrix in spiral form.

Examples:

Input: `mat[][] = [[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]`

Explanation:

Input: `mat[][] = [[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: Applying same technique as shown above.

Input: `mat[][] = [[32, 44, 27, 23], [54, 28, 50, 62]]`

Output: `[32, 44, 27, 23, 62, 50, 28, 54]`

Explanation: Applying same technique as shown above, output will be `[32, 44, 27, 23, 62, 50, 28, 54]`.

**Sol**

Output Window

Compilation Results

Custom Input

Y.O.G.I. (AI Bot)

Problem Solved Successfully

Test Cases Passed

1115 / 1115

Attempts : Correct / Total

1 / 1

Accuracy : 100%

Points Scored

4 / 4

Your Total Score: 55

Time Taken

2.3

Solve Next

Find kth element of spiral matrix

Rotate by 90 degree

Reverse Spiral Form of Matrix

Stay Ahead With:

Build 21 Projects in 21 Days

Build real-world ML, Deep Learning & Gen AI projects

```

1 import java.util.*;
2 class Solution {
3     public ArrayList<Integer> spirallyTraverse(int[][] mat) {
4         // code here
5         ArrayList<Integer> result = new ArrayList<>();
6
7         int n = mat.length;
8         int m = mat[0].length;
9
10        int top = 0, bottom = n - 1;
11        int left = 0, right = m - 1;
12
13        while (top <= bottom && left <= right) {
14            for (int j = left; j <= right; j++) {
15                result.add(mat[top][j]);
16            }
17            top++;
18            for (int i = top; i <= bottom; i++) {
19                result.add(mat[i][right]);
20            }
21            right--;
22            if (top <= bottom) {
23                for (int j = right; j >= left; j--) {
24                    result.add(mat[bottom][j]);
25                }
26                bottom--;
27            }
28            if (left <= right) {
29                for (int i = bottom; i >= top; i--) {
30                    result.add(mat[i][left]);
31                }
32                left++;
33            }
34        }
35        return result;
36    }
37 }
38
39

```

Custom Input

Compile & Run

Submit

8.

You are given an  $m \times n$  integer matrix matrix with the following two properties:

- Each row is sorted in non-decreasing order.
- The first integer of each row is greater than the last integer of the previous row.

Given an integer target, return true if target is in matrix or false otherwise.

You must write a solution in  $O(\log(m * n))$  time complexity.

Example 1:

1	3	5	7
10	11	16	20
23	30	34	60

Input: matrix = [[1,3,5,7],[10,11,16,20],[23,30,34,60]], target = 3

Output: true

Example 2:



1	3	5	7
10	11	16	20
23	30	34	60

Input: matrix =  $[[1,3,5,7],[10,11,16,20],[23,30,34,60]]$ , target = 13

Output: false

**Sol**

The screenshot displays a LeetCode submission for the problem "Search in a Row-Wise Sorted Matrix". The submission is accepted, with 133/133 test cases passed. The user, Anurag Boral, submitted the solution on Feb 09, 2026, at 18:42. The performance metrics show a runtime of 0 ms (beats 100.00%) and memory usage of 44.26 MB (beats 6.85%). The solution is implemented in Java using a binary search approach on the flattened matrix.

```

1 class Solution {
2     public boolean searchMatrix(int[][] matrix, int target) {
3         int m = matrix.length;
4         int n = matrix[0].length;
5
6         int left = 0;
7         int right = m * n - 1;
8
9         while (left <= right) {
10
11             int mid = left + (right - left) / 2;
12
13             int row = mid / n;
14             int col = mid % n;
15
16             int value = matrix[row][col];
17
18             if (value == target) {
19                 return true;
20             }
21             else if (value < target) {
22                 left = mid + 1;
23             }
24             else {
25                 right = mid - 1;
26             }
27         }
28
29         return false;
30     }
31 }

```

9.

Given a row-wise sorted matrix  $mat[][]$  of size  $n*m$ , where the number of rows and columns is always odd. Return the median of the matrix.

Examples:

Input:  $mat[][] = [[1, 3, 5],$

$[2, 6, 9],$

$[3, 6, 9]]$

Output: 5

Explanation: Sorting matrix elements gives us [1, 2, 3, 3, 5, 6, 6, 9, 9]. Hence, 5 is median.

Input: `mat[][] = [[2, 4, 9],`

`[3, 6, 7],`

`[4, 7, 10]]`

Output: 6

Explanation: Sorting matrix elements gives us [2, 3, 4, 4, 6, 7, 7, 9, 10]. Hence, 6 is median.

Input: `mat = [[3], [4], [8]]`

Output: 4

Explanation: Sorting matrix elements gives us [3, 4, 8]. Hence, 4 is median.

### Median in a row-wise sorted Matrix

Difficulty: Medium Accuracy: 55.05% Submissions: 171K+ Points: 4

Given a **row-wise sorted** matrix `mat[][]` of size `n*m`, where the number of rows and columns is always **odd**. Return the **median** of the matrix.

**Examples:**

**Input:** `mat[][] = [[1, 3, 5], [2, 6, 9], [3, 6, 9]]`

**Output:** 5

**Explanation:** Sorting matrix elements gives us [1, 2, 3, 3, 5, 6, 6, 9, 9]. Hence, 5 is median.

**Input:** `mat[][] = [[2, 4, 9], [3, 6, 7], [4, 7, 10]]`

**Output:** 6

**Explanation:** Sorting matrix elements gives us [2, 3, 4, 4, 6, 7, 7, 9, 10]. Hence, 6 is median.

**Input:** `mat = [[3], [4], [8]]`

**Output:** 4

**Explanation:** Sorting matrix elements gives us [3, 4, 8]. Hence, 4 is median.

```
1 class Solution {
2     public int median(int[][] mat) {
3         // code here
4         int n = mat.length;
5         int m = mat[0].length;
6         int min = Integer.MAX_VALUE;
7         int max = Integer.MIN_VALUE;
8         for (int i = 0; i < n; i++) {
9             min = Math.min(min, mat[i][0]);
10            max = Math.max(max, mat[i][m - 1]);
11        }
12        int desired = (n * m) / 2;
13        while (min < max) {
14            int mid = min + (max - min) / 2;
15            int count = 0;
16
17            // Count elements <= mid
18            for (int i = 0; i < n; i++) {
19                count += upperBound(mat[i], mid);
20            }
21
22            if (count <= desired)
23                min = mid + 1;
24            else
25                max = mid;
26        }
27        return min;
28    }
29    private int upperBound(int[] row, int target) {
30        int low = 0, high = row.length;
31        while (low < high) {
32            int mid = (low + high) / 2;
33            if (row[mid] <= target)
34                low = mid + 1;
35            else
36                high = mid;
37        }
38        return low;
39    }
40 }
```

Custom Input Compile & Run Submit

## 10.

You are given a 2D binary array `arr[][]` consisting of only 1s and 0s. Each row of the array is sorted in non-decreasing order. Your task is to find and return the index of the first row that contains the maximum number of 1s. If no such row exists, return -1.

Note:

- The array follows 0-based indexing.
- The number of rows and columns in the array are denoted

by n and m respectively.

Examples:

Input: arr[][] = [[0,1,1,1], [0,0,1,1], [1,1,1,1], [0,0,0,0]]

Output: 2

Explanation: Row 2 contains the most number of 1s (4 1s). Hence, the output is 2.

Input: arr[][] = [[0,0], [1,1]]

Output: 1

Explanation: Row 1 contains the most number of 1s (2 1s). Hence, the output is 1.

Input: arr[][] = [[0,0], [0,0]]

Output: -1

Explanation: No row contains any 1s, so the output is -1.

**Sol**

Output Window

Compilation Results

Custom Input

Y.O.G.I. (AI Bot)

Problem Solved Successfully

Test Cases Passed

1111 / 1111

Attempts : Correct / Total

1 / 1

Accuracy : 100%

Points Scored

4 / 4

Your Total Score: 10

Time Taken

0.74

Solve Next

Max sum in the configuration

Boolean Matrix

Row with Minimum 1s

Stay Ahead With:

Build 21 Projects in 21 Days

Build real-world ML, Deep Learning & Gen AI projects

```
1 // User function Template for Java
2
3 class Solution {
4     public int rowWithMax1s(int arr[][]) {
5         // code here
6         // User function Template for Java
7         int n = arr.length;
8         int m = arr[0].length;
9
10        int maxRowIndex = -1;
11        int j = m - 1;
12
13        for (int i = 0; i < n; i++) {
14
15            while (j >= 0 && arr[i][j] == 1) {
16                j--;
17                maxRowIndex = i;
18            }
19
20        }
21
22        return maxRowIndex;
23    }
24 }
```

Custom Input

Compile & Run

Submit