

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

**1.**

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 LeetCode problem editor interface. On the right, there is a code editor window containing Java code for the 'threeWayPartition' function. The code uses three pointers (low, mid, high) to partition an array around a value 'a'. It swaps elements to ensure all elements less than or equal to 'a' are on the left, and all elements greater than 'b' are on the right. The code editor has line numbers from 1 to 32.

```

1 class Solution {
2     // Function to partition the array around the range such
3     // that the array is divided into three parts.
4     public void threeWayPartition(int arr[], int a, int b) {
5         // code here
6         int low = 0;
7         int mid = 0;
8         int high = arr.length - 1;
9
10        while (mid <= high) {
11            if (arr[mid] < a) {
12                int temp = arr[low];
13                arr[low] = arr[mid];
14                arr[mid] = temp;
15
16                low++;
17                mid++;
18            }
19            else if (arr[mid] > b) {
20                int temp = arr[mid];
21                arr[mid] = arr[high];
22                arr[high] = temp;
23
24                high--;
25            }
26            else {
27                mid++;
28            }
29        }
30    }
31 }
32

```

On the left, there is a summary section with the following details:

- Output Window**
- Compilation Results**: Custom Input, Y.O.G.I. (AI Bot)
- Problem Solved Successfully** (green checkmark)
- Test Cases Passed**: 1111 / 1111
- Attempts : Correct / Total**: 1 / 1 (Accuracy: 100%)
- Points Scored**: 2 / 2
- Your Total Score**: 2 ↗
- Solve Next**: Wave Array, Sort by Absolute Difference, Convert an array to reduced form
- Stay Ahead With:** Build 21 Projects in 21 Days (Build real-world ML, Deep Learning & Gen AI projects)

2.

Given an array  $\text{arr}$  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  $\text{arr}[i]$ ,  $\text{arr}[j]$ . 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:  $\text{arr}[] = [2, 1, 5, 6, 3]$ ,  $k = 3$

Output: 1

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

Input:  $\text{arr}[] = [2, 7, 9, 5, 8, 7, 4]$ ,  $k = 6$

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-  $\text{arr}[] = [9, 7, 2, 5, 4, 7, 8]$

Input:  $\text{arr}[] = [2, 4, 5, 3, 6, 1, 8]$ ,  $k = 6$

Output: 0

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 4 / 4 Time Taken 0.45

Your Total Score: 6 ↑

Solve Next

Rearrange Array Alternately Count Number Subarray Inversions

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

Custom Input Compile & Run Submit

### 3.

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

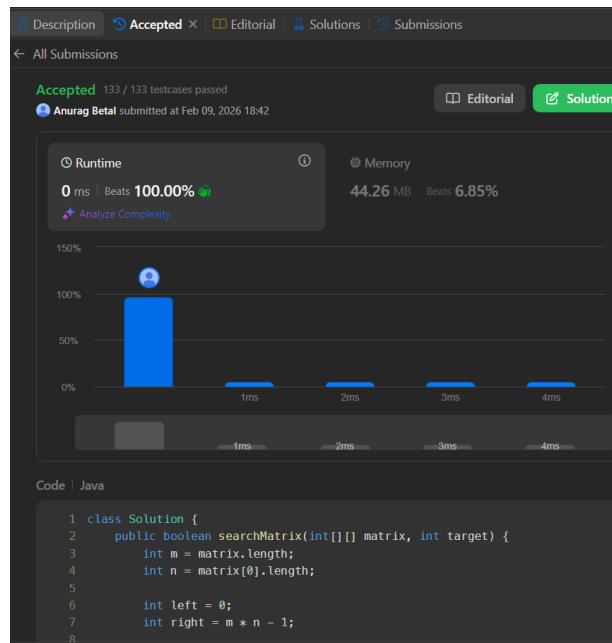
Example 2:

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 shows a LeetCode submission page. At the top, it says "Accepted" with 133 / 133 testcases passed by Anurag Betal at Feb 09, 2026 18:42. Below this, there are two performance metrics: "Runtime" (0 ms | Beats 100.00%) and "Memory" (44.26 MB | Beats 6.85%). A bar chart visualizes the runtime distribution. On the right, the code is displayed in Java:

```

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             } else if (value < target) {
21                 left = mid + 1;
22             } else {
23                 right = mid - 1;
24             }
25         }
26
27         return false;
28     }
29 }
30
31

```

## 4.

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 ✓ Suggest Feedback

Test Cases Passed <b>1111 / 1111</b>	Attempts : Correct / Total <b>1 / 1</b> Accuracy : 100%
Points Scored <span>ⓘ</span> <b>4 / 4</b>	Time Taken <b>0.74</b>
Your Total Score: 10 <span>↗</span>	

**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

Custom Input   Compile & Run   Submit

```
1 // User function Template for Java
2
3 * class Solution {
4     public int rowWithMaxis(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            while (j >= 0 && arr[i][j] == 1) {
15                j--;
16                maxRowIndex = i;
17            }
18        }
19
20    }
21
22    return maxRowIndex;
23 }
24 }
```