

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 page for "All Possible Splits". The status bar indicates "Solved Successfully" with a timestamp of "2023-02-02 11:11:11". The submission details show "Accepted" status, "1 / 1" test cases passed, and "Accuracy: 100%". The code editor contains the following Java code:

```
1 // Solution
2 // 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     int i = 0;
6     int mid = 0;
7     int high = arr.length - 1;
8
9     while (i <= high) {
10         if (arr[i] <= a) {
11             swap(arr, i, mid);
12             i++;
13             mid++;
14         } else if (arr[i] >= b) {
15             swap(arr, i, high);
16             high--;
17         } else {
18             i++;
19         }
20     }
21 }
```

2.

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

```
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             ans = Math.min(ans, bad);
33             i++;
34             j++;
35         }
36     }
37 }
38 }
```

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

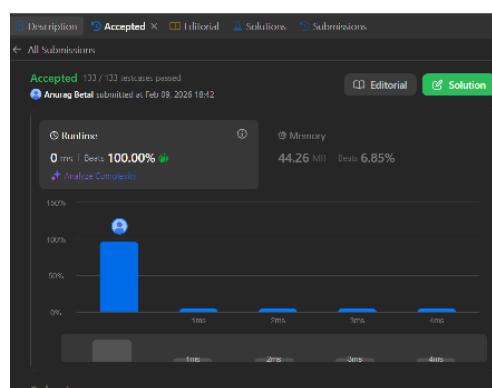
Example 2:

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



```

Description Accepted ✅ Accepted ✕ Editorial Solutions Submissions
< All Submissions
Accepted 100 / 100 testcases passed
Anurag Betal submitted at Feb 09, 2026 10:42
Runtime 0 ms | Beats 100.00% Memory 44.26 MB | Beats 6.85%
Analyze Complexity
Code 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 = n * m - 1;
8
9         while (left <= right) {
10             int mid = left + (right - left) / 2;
11
12             int row = mid / n;
13             int col = mid % n;
14
15             int value = matrix[row][col];
16
17             if (value == target) {
18                 return true;
19             } else if (value < target) {
20                 left = mid + 1;
21             } else {
22                 right = mid - 1;
23             }
24         }
25
26         return false;
27     }
28 }

```

Testcase Test Result

4.

You are given a 2D binary array arr $[\cdot]$ 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 $[\cdot] = [[0,1,1,1], [0,0,1,1], [1,1,1,1], [0,0,0,0]]$

Output: 2

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

Input: arr $[\cdot] = [[0,0], [1,1]]$

Output: 1

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

Input: arr $[\cdot] = [[0,0], [0,0]]$

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

Time Taken 0.74

Your Total Score: 10 ↗

Snipped Feedback

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

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