

LABSHEET - 5

CS F211 - Data Structures and Algorithms
Academic Year: 2025-2026 (2nd Semester)

General Instructions for this Labsheet:

- Use of any built-in function like qsort(), bsearch(), sqrt(), etc. is not allowed.
- Questions 1, 2, 3, 6 can be done using any suitable method.
- Question 4 should be done using binary search. Using linear search is not allowed.
- Question 7 should be done using the Quick Select algorithm only.
- Question 5, 8, 9 should be done using Sorting only.
- Question 10 should be done using Radix Sort only.
- Use 1-based indexing whenever asked to print indices.

Question 1: Rectangle Sum

Problem Description:

You will be given a $M \times N$ matrix of positive integers, and 4 numbers: a_1, b_1, a_2, b_2 .

For the given matrix, print the 2D Prefix Sum Matrix from the top-left corner.

Also, print the sum of all the elements in the sub-rectangle defined by the top-left coordinate (a_1, b_1) and the bottom-right coordinate (a_2, b_2) using the calculated 2D Prefix Sum Matrix.

Note: a_1 and a_2 represent rows, while b_1 and b_2 represent columns.

Note: 1-based indexing has been followed for this question.

Input Format:

Line 1: Two integers M, N (Rows, Columns)

Next M lines: N integers each, representing the matrix rows

Next 1 Line: Four integers (a_1, b_1, a_2, b_2)

Output Format:

Print the 2D prefix matrix and the sum of the rectangle formed.

Sample Test Case 1:

Input:

3 3
1 3 5
2 4 6
3 4 1

1 2 3 3

Output:

1 4 9
3 10 21
6 17 29
Sum = 23

Sample Test Case 2:

Input:

4 4
5 1 9 11
2 4 8 10
7 14 3 8
12 6 4 5

2 1 4 2

Output:

5 6 15 26
7 12 29 50
14 33 53 82
26 51 75 109
Sum = 45

Question 2: Maximum Subarray with Equal Difference

Problem Description: Given an array of N integers. Return the longest subarray where the difference between every adjacent element is equal.

More formally, return the longest subarray $[l, r]$ ($1 \leq l < r \leq N$) such that the value of $a[i + 1] - a[i]$ for all i in the range $[l, r - 1]$ is the same.

If multiple such longest subarrays exist, return any one of them.

Note: 1-based indexing has been followed for this question.

Input Format:

Line 1: One integer N (Size of the array, $N \geq 2$)

Line 2: N integers (Array elements)

Output Format:

Two space-separated integers representing the starting and the ending indices of the subarray.

Sample Test Case 1:

Input:

3

1 3 6

Output:

1 2

Both 1 2 and 2 3 will be accepted as output.

Sample Test Case 2:

Input:

6

1 5 9 7 5 3

Output:

3 6

Question 3: Equal XOR on sides

Problem Description:

Given an array arr of N integers. An index is called XOR-Balanced if the xor value of all the elements on its left is equal to the xor value of all the elements on its right.

More formally, index i ($2 \leq i \leq N - 1$) is called XOR-Balanced if:

$$arr[1] \wedge \dots \wedge arr[i - 1] = (arr[i + 1] \wedge \dots \wedge arr[N])$$

Return all XOR-Balanced indices in the array.

Solve this using an efficient algorithm with $O(N)$ time complexity and $O(1)$ extra space.

Note: 1-based indexing has been followed for this question.

Input Format:

Line 1: One integer N (Size of the array, $N \geq 3$)

Line 2: N integers (Array elements)

Output Format:

All the XOR-Balanced indices

Sample Test Case 1:

Input:

8

2 3 4 2 3 2 6 2

Output:

4 6

Explanation: $(2 \wedge 3 \wedge 4) = (3 \wedge 2 \wedge 6 \wedge 2) = 5$, Hence, index 4 is a XOR-Balanced index

Similarly, $(2 \wedge 3 \wedge 4 \wedge 2 \wedge 3) = (6 \wedge 2) = 4$, Hence, index 6 is a XOR-Balanced index

Sample Test Case 2:

Input:

5

2 1 7 1 2

Output:

3

Question 4: Maximizing the Minimum Distance Between Towers

Problem Description:

You are a network engineer managing N signal towers placed along a straight highway. Each tower is located at a distinct integer position, and these positions are provided in strictly increasing order. To optimize the network's coverage while minimizing signal interference, you must activate exactly K towers from the available set.

Once the towers are selected, they maintain their original left-to-right order based on their positions. Your objective is to select the K towers such that the smallest distance between any two consecutive activated towers is as large as possible. Distance is only measured between towers you have chosen to activate, ignoring any inactive towers between them. You are essentially maximizing the minimum possible gap.

Input Format

- **Line 1:** Two space-separated integers: N (the total number of towers) and K (the number of towers to be activated) ($K \leq N$).
- **Line 2:** N space-separated distinct integers representing the positions of the towers, provided in strictly increasing order.

Output Format

- A single integer representing the **maximum possible minimum distance** between any two consecutive activated towers.

Sample Test Case 1

Input:

4 3
0 5 8 13

Output:

Explanation: You can activate towers at positions $\{0, 5, 13\}$. The gaps are $5 - 0 = 5$ and $13 - 5 = 8$. The minimum gap is **5**.

Sample Test Case 2

Input:

6 4
1 2 5 8 12 15

Output:

Explanation: A possible selection is $\{1, 5, 8, 12\}$, which gives gaps $\{4, 3, 4\}$. Here the min is 3.

Question 5: The Divisor-Based Ranker

Problem Description:

You are given an array of N distinct positive integers. Your task is to sort these integers based on the number of their divisors according to the following criteria:

1. Primary Rule: Sort in ascending order of the number of divisors (numbers with fewer divisors must appear first).
2. Tie-Breaking Rule: If two numbers have the same number of divisors, the larger numerical value must appear first.

Note: A divisor is a positive integer that divides the number exactly, including 1 and the number itself.

Input Format

- Line 1: An integer N , representing the number of elements in the array.
- Line 2: N space-separated distinct positive integers.

Output Format

- A single line containing the N integers, sorted according to the rules above.

Sample Case 1

Input:

4
6 10 2 4

Output:

2 4 10 6

Explanation:

2 has 2 divisors ($\{1, 2\}$)
4 has 3 divisors ($\{1, 2, 4\}$)
10 has 4 divisors ($\{1, 2, 5, 10\}$)
6 has 4 divisors ($\{1, 2, 3, 6\}$)
10 comes before 6 because $10 > 6$.

Sample Case 2

Input:

5
12 1 15 3 9

Output:

1 3 9 15 12

Explanation: Counts: 1(1), 3(2), 9(3), 15(4), 12(6).

Question 6: The Matrix Perimeter Erosion

Problem Description: You are given an $N \times M$ matrix of integers. A perimeter erosion transformation is applied to the matrix by rotating its elements layer by layer. A layer is defined as a complete rectangular boundary of the matrix. The outermost boundary is Layer 1, the next inner boundary is Layer 2, and so on. Each layer is rotated by exactly one position, following an alternating rotation pattern as:

- Layer 1 (outermost): Rotate clockwise
- Layer 2: Rotate counter-clockwise
- Layer 3: Rotate clockwise
- This alternating pattern continues for all valid layers

Rotation Rules:

- Clockwise rotation means each element moves to the next position along the perimeter in clockwise order like : Top row (left->right), right column (top->bottom), bottom row (right->left), left column (bottom->top).
- Counter-clockwise rotation means each element moves to the previous position along the perimeter.
- A layer must form a complete rectangle.
- If a layer consists of only one row or one column, it must not be rotated and should remain unchanged.

Your task is to output the matrix after applying all valid layer rotations. You may use a temporary one-dimensional array to store elements of a single layer, but you must not create another $N \times M$ matrix.

Input Format

Line 1: Two space-separated integers N and M

Next N lines: M space-separated integers representing the matrix

Output Format

Print the $N \times M$ matrix after applying the perimeter erosion transformation.

Sample Test Case 1

Input

```
4 4  
1 2 3 4  
5 6 7 8  
9 10 11 12  
13 14 15 16
```

Output

```
5 1 2 3  
9 7 11 4  
13 6 10 8  
14 15 16 12
```

Explanation:

- Layer 1 (outer): rotated clockwise
- Layer 2 (inner 2×2): rotated counter-clockwise

Sample Test Case 2

Input

```
3 5  
1 2 3 4 5  
6 7 8 9 10  
11 12 13 14 15
```

Output

```
6 1 2 3 4  
11 7 8 9 5  
12 13 14 15 10
```

Explanation:

- Layer 1 is rotated clockwise
- Inner layer is a single row -> remains unchanged

Question 7: Kth Largest Element

Problem Description: Given an integer array nums and an integer k, return the kth largest element in the array.

- Note that it is the kth largest element in the sorted order, not the kth distinct element.
- **Constraint:** You need to solve this using the Quick select algorithm.

Input Format:

- Line 1: N(Size of array).
- Line 2: N integers.
- Line 3: K.

Output Format:

- Print the kth largest element.

Sample Test Case 1:

Input:

```
6
3 2 1 5 6 4
2
```

Output:

```
5
```

(Explanation: Sorted array is [1, 2, 3, 4, 5, 6]. The 2nd largest is 5.)

Sample Test Case 2:

Input:

```
9
3 2 3 1 2 4 5 5 6
4
```

Output:

```
4
```

Question 8: The Conference Hall (Merge Intervals)

Problem Description: You are given a list of time intervals for meetings, where each interval has a start and an end time. Some meetings overlap (eg, 1:00-3:00 and 2:00-4:00). The facility manager wants to merge all overlapping meetings into consolidated blocks of time to determine when the hall is occupied.

- **Task:** Write a program to **merge all overlapping intervals**.

Input Format:

- Line 1: N (Number of intervals).
- Next N lines: Two integers per line (start end).

Output Format:

- Print the merged intervals in sorted order.

Sample Test Case 1:

Input:

```
4
1 3
2 6
8 10
15 18
```

Output:

```
1 6
8 10
15 18
```

Explanation:

- (1, 3) and (2, 6) overlap because 2 < 3. Merge to (1, 6).
- (8, 10) and (15, 18) do not overlap.

Sample Test Case 2:

Input:

```
2
1 4
4 5
```

Output:

```
1 5
```

Explanation: Intervals touching at the boundary are also considered overlapping).

Question 9: The Zero-Sum Triplets (3Sum)

Problem Description: You are given an integer array `nums` of size `N`. Your task is to find and print all unique triplets $[nums[i], nums[j], nums[k]]$ such that:

1. $i \neq j, i \neq k$, and $j \neq k$ (Indices are distinct)
2. $nums[i] + nums[j] + nums[k] == 0$ (The sum is zero)
- **Duplicates:** The result must not contain duplicate triplets (eg, if $[-1, 0, 1]$ found, do not print $[0, -1, 1]$ again).

Input Format:

- Line 1: `N`(Size of array).
- Line 2: `N` integers.

Output Format:

- Print each unique triplet on a new line.
- If no such triplets exist, print "No triplets found".

Sample Test Case 1:

Input:

6
-1 0 1 2 -1 -4

Output:

-1 -1 2
-1 0 1

Explanation:

- Sorted Array: $[-4, -1, -1, 0, 1, 2]$
- Fix first element -1: Search for pair summing to 1. Found $(-1, 2)$ and $(0, 1)$.

Sample Test Case 2:

Input:

3
0 1 1

Output:

No triplets found

Question 10: Radix Sort

Problem Description: Given an array of N non-negative integers. Sort the array using Radix Sort only. No other method will be accepted.

Input Format:

Line 1: One integer N (Size of the array, $N \geq 2$)

Line 2: N integers (Array elements)

Output Format:

Two space-separated integers representing the starting and the ending indices of the subarray.

Sample Test Case 1:

Input:

8

170 45 75 90 802 24 2 66

Output:

2 24 45 66 75 90 170 802

Sample Test Case 2:

Input:

6

43 8 43 7 9 8

Output:

7 8 8 9 43 43