Array Problems for Beginners

Here's a collection of array problems designed to help beginners build a strong foundation in array manipulation. Each problem includes a description, sample inputs and outputs, and explanations.

Problem 1: Find the Maximum Element

Description: Write a program that finds the largest element in an array.

Sample Input:

```
[3, 9, 5, 1, 7, 2]
```

Sample Output:

```
9
```

Explanation: Traverse through each element of the array and keep track of the maximum value seen so far. In this case, 9 is the largest element.

Problem 2: Find the Minimum Element

Description: Write a program that finds the smallest element in an array.

Sample Input:

```
[8, 4, 6, 2, 5, 10]
```

Sample Output:

```
2
```

Explanation: Similar to finding the maximum, traverse the array while tracking the minimum value encountered. Here, 2 is the smallest element.

Problem 3: Calculate the Sum of All Elements

Description: Write a program that calculates the sum of all elements in an array.

Sample Input:

```
[1, 2, 3, 4, 5]
```

Sample Output:

```
15
```

Explanation: Add each element in the array (1 + 2 + 3 + 4 + 5 = 15).

Problem 4: Calculate the Average of All Elements

Description: Write a program that calculates the average (mean) of all elements in an array.

Sample Input:

```
[2, 4, 6, 8, 10]
```

Sample Output:

```
6
```

Explanation: First calculate the sum of all elements (2 + 4 + 6 + 8 + 10 = 30), then divide by the number of elements (30 / 5 = 6).

Problem 5: Reverse an Array

Description: Write a program that reverses the elements of an array.

Sample Input:

```
[1, 2, 3, 4, 5]
```

Sample Output:

```
[5, 4, 3, 2, 1]
```

Explanation: Swap elements from the beginning and end, moving toward the center.

Problem 6: Count Even Numbers

Description: Write a program that counts the number of even elements in an array.

Sample Input:

```
[1, 2, 3, 4, 5, 6, 7, 8]
```

Sample Output:

```
4
```

Explanation: The even numbers in the array are 2, 4, 6, and 8, so there are 4 even numbers.

Problem 7: Find the Second Largest Element

Description: Write a program that finds the second largest element in an array.

Sample Input:

```
[12, 35, 1, 10, 34, 1]
```

Sample Output:

```
34
```

Explanation: The largest element is 35, and the second largest is 34.

Problem 8: Check if an Array is Sorted

Description: Write a program that determines whether an array is sorted in ascending order.

Sample Input 1:

```
[1, 2, 3, 4, 5]
```

Sample Output 1:

```
Yes, the array is sorted.
```

Sample Input 2:

```
[1, 3, 2, 5, 4]
```

Sample Output 2:

```
No, the array is not sorted.
```

Explanation: Check if each element is less than or equal to the next element. If any pair is out of order, the array is not sorted.

Problem 9: Remove Duplicates

Description: Write a program that removes duplicate elements from a sorted array and returns the new length.

Sample Input:

```
[1, 1, 2, 2, 3, 4, 5, 5]
```

Sample Output:

```
Array after removing duplicates: [1, 2, 3, 4, 5, _, _, _]
Length: 5
```

Explanation: Since the array is sorted, duplicates appear next to each other. Use two pointers to keep track of unique elements.

Problem 10: Find Missing Number

Description: Given an array containing n distinct numbers taken from 0 to n, find the missing number.

Sample Input:

```
[3, 0, 1, 4, 6, 2]
```

Sample Output:

```
5
```

Explanation: The array should contain numbers from 0 to 6, but 5 is missing.

Problem 11: Rotate Array to the Right

Description: Write a program that rotates the elements of an array to the right by k steps.

Sample Input:

```
Array: [1, 2, 3, 4, 5, 6, 7]
k: 3
```

Sample Output:

```
[5, 6, 7, 1, 2, 3, 4]
```

Explanation: After 3 rotations to the right, the original elements at indices 4, 5, and 6 are now at the beginning.

Problem 12: Merge Two Sorted Arrays

Description: Write a program that merges two sorted arrays into a new sorted array.

Sample Input:

```
Array 1: [1, 3, 5, 7]
Array 2: [2, 4, 6, 8]
```

Sample Output:

```
[1, 2, 3, 4, 5, 6, 7, 8]
```

Explanation: Compare elements from both arrays and insert them in order into a new array.

Problem 13: Find Element Pairs with Sum K

Description: Write a program that finds all pairs of elements in an array whose sum is equal to a given target value k.

Sample Input:

```
Array: [1, 5, 7, 1, 5, 9, 2, 6]
Target: 10
```

Sample Output:

```
Pairs: (1,9), (5,5), (1,9), (4,6)
```

Explanation: The pairs of indices with values that sum to 10 are: (0,5), (1,2), (3,5), and (2,7).

Problem 14: Find Leaders in an Array

Description: An element is a leader if it is greater than all elements to its right. Write a program to find all leaders in an array.

Sample Input:

```
[16, 17, 4, 3, 5, 2]
```

Sample Output:

```
Leaders: 17, 5, 2
```

Explanation: 17 is greater than all elements to its right. 5 is greater than all elements to its right. 2 is the last element and has no elements to its right, so it's a leader by default.

Problem 15: Find Majority Element

Description: Write a program to find the majority element in an array. A majority element appears more than n/2 times, where n is the array size.

Sample Input:

```
[3, 3, 4, 2, 4, 4, 2, 4, 4]
```

Sample Output:

```
Majority Element: 4
```

Explanation: The element 4 appears 5 times in an array of length 9, which is more than 9/2 = 4.5 times.

Problem 16: Array Intersection

Description: Write a program that finds the common elements between two arrays (the intersection).

Sample Input:

```
Array 1: [1, 2, 4, 5, 6]
Array 2: [2, 3, 5, 7]
```

Sample Output:

```
Intersection: [2, 5]
```

Explanation: The elements 2 and 5 appear in both arrays.

Problem 17: Check if Subarrays Sum to Zero

Description: Write a program to determine if an array has a subarray with a sum of zero.

Sample Input 1:

```
[4, 2, -3, 1, 6]
```

Sample Output 1:

```
Yes, subarray exists with sum 0
```

Explanation: The subarray [2, -3, 1] sums to zero.

Sample Input 2:

```
[4, 2, 5, 1, 6]
```

Sample Output 2:

```
No subarray exists with sum 0
```

Problem 18: Find the First Non-Repeating Element

Description: Write a program to find the first non-repeating element in an array.

Sample Input:

```
[9, 4, 9, 6, 7, 4]
```

Sample Output:

```
6
```

Explanation: 6 is the first element that doesn't appear more than once in the array.

Problem 19: Move All Zeros to End

Description: Write a program to move all zeros in an array to the end while maintaining the relative order of non-zero elements.

Sample Input:

```
[0, 1, 0, 3, 12]
```

Sample Output:

```
[1, 3, 12, 0, 0]
```

Explanation: All non-zero elements maintain their order, and all zeros are moved to the end.

Problem 20: Find Equilibrium Index

Description: Write a program to find the equilibrium index of an array. An equilibrium index is where the sum of elements to the left equals the sum of elements to the right.

Sample Input:

```
[-7, 1, 5, 2, -4, 3, 0]
```

Sample Output:

```
Equilibrium Index: 3
```

Explanation: At index 3, the sum of elements to the left (-7 + 1 + 5 = -1) equals the sum of elements to the right (-4 + 3 + 0 = -1).

Problem 21: Find Peak Element

Description: A peak element is an element that is greater than its neighbors. Write a program to find any peak element in an array.

Sample Input:

```
[1, 3, 20, 4, 1, 0]
```

Sample Output:

```
Peak Element: 20 at index 2
```

Explanation: 20 is greater than both its neighbors (3 and 4).

Problem 22: Replace Elements with Greatest on Right

Description: Replace each element of the array with the greatest element to its right. The last element should be replaced with -1.

Sample Input:

```
[17, 18, 5, 4, 6, 1]
```

Sample Output:

```
[18, 6, 6, 6, 1, -1]
```

Explanation:

- 17 → 18 (greatest element to its right)
- 18 → 6 (greatest element to its right)
- 5 → 6 (greatest element to its right)
- 4 → 6 (greatest element to its right)
- 6 → 1 (greatest element to its right)
- 1 → -1 (no elements to its right)

Problem 23: Find Maximum Difference

Description: Write a program to find the maximum difference between any two elements in an array such that the larger element appears after the smaller element.

Sample Input:

```
[2, 3, 10, 6, 4, 8, 1]
```

Sample Output:

```
Maximum Difference: 8
```

Explanation: The maximum difference is between 2 and 10, which is 8. Note that 10 appears after 2 in the array.

Problem 24: Segregate Even and Odd Numbers

Description: Write a program to segregate even and odd numbers in an array. The program should put all even numbers first, followed by all odd numbers.

Sample Input:

```
[12, 34, 45, 9, 8, 90, 3]
```

Sample Output:

```
[12, 34, 8, 90, 45, 9, 3]
```

Explanation: All even numbers (12, 34, 8, 90) appear before all odd numbers (45, 9, 3). The relative order within even and odd numbers doesn't matter.

Problem 25: Find the Smallest Missing Positive Integer

Description: Write a program to find the smallest missing positive integer in an array.

Sample Input:

```
[3, 4, -1, 1]
```

Sample Output:

```
2
```

Explanation: The array contains 1, 3, and 4 as positive integers. The smallest missing positive integer is 2.

Problem 26: Find Longest Consecutive Sequence

Description: Write a program to find the length of the longest consecutive elements sequence in an array.

Sample Input:

```
[100, 4, 200, 1, 3, 2]
```

Sample Output:

```
4
```

Explanation: The longest consecutive sequence is [1, 2, 3, 4], with a length of 4.

Problem 27: Trapping Rain Water

Description: Given an array representing the heights of bars, compute how much water can be trapped between the bars after raining.

Sample Input:

```
[0, 1, 0, 2, 1, 0, 1, 3, 2, 1, 2, 1]
```

Sample Output:

```
6
```

Explanation: Units of water trapped: 1 + 1 + 2 + 1 + 1 = 6

Problem 28: Buy and Sell Stock (Max Profit)

Description: Write a program to find the maximum profit by buying and selling a stock once. You have the daily prices of a stock.

Sample Input:

```
[7, 1, 5, 3, 6, 4]
```

Sample Output:

```
5
```

Explanation: Buy on day 2 (price = 1) and sell on day 5 (price = 6), profit = 6-1 = 5.

Problem 29: Product of Array Except Self

Description: Write a program that returns an array where each element is the product of all elements in the original array except the element at that index.

Sample Input:

```
[1, 2, 3, 4]
```

Sample Output:

```
[24, 12, 8, 6]
```

Explanation:

- At index 0: 2 * 3 * 4 = 24
- At index 1: 1 * 3 * 4 = 12
- At index 2: 1 * 2 * 4 = 8
- At index 3: 1 * 2 * 3 = 6

Problem 30: Find Max Sum Subarray (Kadane's Algorithm)

Description: Write a program to find the contiguous subarray with the largest sum.

Sample Input:

```
[-2, 1, -3, 4, -1, 2, 1, -5, 4]
```

Sample Output:

```
Maximum Subarray Sum: 6
Subarray: [4, -1, 2, 1]
```

Explanation: The subarray [4, -1, 2, 1] has the largest sum of 6.

Problem 31: Three Sum

Description: Given an array of integers, find all unique triplets in the array that sum to zero.

Sample Input:

```
[-1, 0, 1, 2, -1, -4]
```

Sample Output:

```
[[-1, -1, 2], [-1, 0, 1]]
```

Explanation: The triplets that sum to zero are [-1, -1, 2] and [-1, 0, 1].

Problem 32: Container With Most Water

Description: Given n non-negative integers where each represents a point at coordinate (i, height[i]), find two lines that together with the x-axis form a container that holds the most water.

Sample Input:

```
[1, 8, 6, 2, 5, 4, 8, 3, 7]
```

Sample Output:

```
49
```

Explanation: The maximum area is formed by the heights at indices 1 and 8, with heights 8 and 7, forming an area of min(8, 7) * (8 - 1) = 49.

Problem 33: Next Permutation

Description: Implement the next permutation, which rearranges numbers into the lexicographically next greater permutation.

Sample Input:

```
[1, 2, 3]
```

Sample Output:

```
[1, 3, 2]
```

Explanation: The next lexicographically greater permutation of [1, 2, 3] is [1, 3, 2].

Problem 34: Valid Mountain Array

Description: Given an array, determine if it is a valid mountain array. A mountain array is one that:

- Has length ≥ 3
- There exists some i with 0 < i < length - 1 such that:
  - arr[0] < arr[1] < ... < arr[i]
  - arr[i] > arr[i+1] > ... > arr[length-1]

Sample Input 1:

```
[0, 3, 2, 1]
```

Sample Output 1:

```
True
```

Sample Input 2:

```
[3, 5, 5]
```

Sample Output 2:

```
False
```

Explanation: In the first example, the array increases up to index 1 and then decreases, forming a mountain. In the second example, there is no strict increase followed by a strict decrease.

Problem 35: Wiggle Sort

Description: Given an unsorted array, sort it such that nums[0] < nums[1] > nums[2] < nums[3]...

Sample Input:

```
[3, 5, 2, 1, 6, 4]
```

Sample Output:

```
[1, 6, 2, 5, 3, 4]
```

Explanation: After sorting: 1 < 6 > 2 < 5 > 3 < 4

2D Array (Matrix) Problems for Beginners

Here's a collection of 2D array problems to help you practice matrix manipulation. Each problem includes a description, sample inputs and outputs, and explanations.

Problem 1: Matrix Traversal - Row-wise and Column-wise Sum

Description: Write a program to find the sum of each row and each column in a matrix.

Sample Input:

```
[
  [1, 2, 3],
  [4, 5, 6],
  [7, 8, 9]
]
```

Sample Output:

```
Row Sums: [6, 15, 24]
Column Sums: [12, 15, 18]
```

Explanation:

- Row 1 sum: 1 + 2 + 3 = 6
- Row 2 sum: 4 + 5 + 6 = 15
- Row 3 sum: 7 + 8 + 9 = 24
- Column 1 sum: 1 + 4 + 7 = 12
- Column 2 sum: 2 + 5 + 8 = 15
- Column 3 sum: 3 + 6 + 9 = 18

Problem 2: Matrix Diagonal Sum

Description: Write a program to find the sum of both diagonals of a square matrix.

Sample Input:

```
[
  [1, 2, 3],
  [4, 5, 6],
  [7, 8, 9]
]
```

Sample Output:

```
Primary Diagonal Sum: 15
Secondary Diagonal Sum: 15
```

Explanation:

- Primary diagonal elements: 1, 5, 9. Sum = 15
- Secondary diagonal elements: 3, 5, 7. Sum = 15

Problem 3: Matrix Transpose

Description: Write a program to find the transpose of a matrix. The transpose of a matrix is obtained by changing rows to columns and columns to rows.

Sample Input:

```
[
  [1, 2, 3],
  [4, 5, 6]
]
```

Sample Output:

```
[
  [1, 4],
  [2, 5],
  [3, 6]
]
```

Explanation:

- The rows of the original matrix become columns in the transpose.
- The columns of the original matrix become rows in the transpose.

Problem 4: Search in a Sorted Matrix

Description: Write a program to search for a target value in a matrix where each row and each column is sorted in ascending order.

Sample Input:

```
Matrix:
[
  [1,  4,  7, 11],
  [2,  5,  8, 12],
  [3,  6,  9, 16],
  [10, 13, 14, 17]
]
Target: 5
```

Sample Output:

```
Found at position (1, 1)
```

Explanation: The target value 5 is found at row 1, column 1 (0-indexed).

Problem 5: Rotate Matrix 90 Degrees Clockwise

Description: Write a program to rotate a square matrix 90 degrees clockwise.

Sample Input:

```
[
  [1, 2, 3],
  [4, 5, 6],
  [7, 8, 9]
]
```

Sample Output:

```
[
  [7, 4, 1],
  [8, 5, 2],
  [9, 6, 3]
]
```

Explanation: After rotating the matrix 90 degrees clockwise:

- First row becomes last column (read from top to bottom)
- Second row becomes second-to-last column (read from top to bottom)
- Third row becomes first column (read from top to bottom)

Problem 6: Count Negative Numbers in a Sorted Matrix

Description: Write a program to count negative numbers in a matrix where rows and columns are sorted.

Sample Input:

```
[
  [4,  3,  2, -1],
  [3,  2,  1, -1],
  [1,  1, -1, -2],
  [-1, -1, -2, -3]
]
```

Sample Output:

```
8
```

Explanation: There are 8 negative numbers in the matrix: -1 (2 occurrences), -2 (2 occurrences), and -3 (1 occurrence).

Problem 7: Find the Maximum Element in Each Row

Description: Write a program to find the maximum element in each row of a matrix.

Sample Input:

```
[
  [3, 7, 2, 9],
  [5, 1, 4, 8],
  [6, 2, 0, 3]
]
```

Sample Output:

```
[9, 8, 6]
```

Explanation:

- Maximum in row 1: 9
- Maximum in row 2: 8
- Maximum in row 3: 6

Problem 8: Print Matrix in Spiral Order

Description: Write a program to print elements of a matrix in spiral order.

Sample Input:

```
[
  [1, 2, 3, 4],
  [5, 6, 7, 8],
  [9, 10, 11, 12]
]
```

Sample Output:

```
[1, 2, 3, 4, 8, 12, 11, 10, 9, 5, 6, 7]
```

Explanation: The spiral order starts from the top-left corner and follows right → down → left → up, continuing inward.

Problem 9: Check if a Matrix is a Toeplitz Matrix

Description: A matrix is Toeplitz if every diagonal from top-left to bottom-right has the same elements. Write a program to check if a given matrix is a Toeplitz matrix.

Sample Input:

```
[
  [1, 2, 3, 4],
  [5, 1, 2, 3],
  [9, 5, 1, 2]
]
```

Sample Output:

```
True
```

Explanation: Every diagonal from top-left to bottom-right has the same elements:

- Diagonal 1: [1, 1, 1]
- Diagonal 2: [2, 2, 2]
- Diagonal 3: [3, 3]
- Diagonal 4: [4]
- Diagonal 5: [5, 5]
- Diagonal 6: [9]

Problem 10: Set Matrix Zeros

Description: Write a program such that if an element in an MxN matrix is 0, its entire row and column are set to 0.

Sample Input:

```
[
  [1, 1, 1],
  [1, 0, 1],
  [1, 1, 1]
]
```

Sample Output:

```
[
  [1, 0, 1],
  [0, 0, 0],
  [1, 0, 1]
]
```

Explanation: The element at position (1, 1) is 0, so its entire row and column are set to 0.

Problem 11: Island Perimeter

Description: You are given a 2D grid where 1 represents land and 0 represents water. Grid cells are connected horizontally and vertically. The grid is completely surrounded by water. Find the perimeter of the island (the total number of edges between land and water).

Sample Input:

```
[
  [0, 1, 0, 0],
  [1, 1, 1, 0],
  [0, 1, 0, 0],
  [1, 1, 0, 0]
]
```

Sample Output:

```
16
```

Explanation: The perimeter is the number of edges between land and water. Each land cell contributes 4 to the perimeter, minus 2 for each adjacent land cell (shared edge).

Problem 12: Find Saddle Point in Matrix

Description: A saddle point is an element in a matrix that is the minimum in its row and maximum in its column. Write a program to find saddle points in a matrix.

Sample Input:

```
[
  [1, 2, 3],
  [4, 5, 6],
  [7, 8, 9]
]
```

Sample Output:

```
No saddle point found
```

Sample Input 2:

```
[
  [3, 1, 7],
  [4, 2, 5],
  [9, 6, 8]
]
```

Sample Output 2:

```
Saddle point found at (1, 1) with value 2
```

Explanation: In the second example, 2 is minimum in its row (compared to 4 and 5) and maximum in its column (compared to 1).

Problem 13: Sum of Elements in a Given Rectangle

Description: Given a matrix and the coordinates of a rectangle (top-left and bottom-right), find the sum of all elements inside the rectangle.

Sample Input:

```
Matrix:
[
  [1, 2, 3, 4],
  [5, 6, 7, 8],
  [9, 10, 11, 12],
```

```
  [13, 14, 15, 16]
]
Top Left: (1, 1)
Bottom Right: (2, 2)
```

Sample Output:

```
34
```

Explanation: The rectangle contains the elements 6, 7, 10, and 11. The sum is 6 + 7 + 10 + 11 = 34.

Problem 14: Find the Maximum Sum Submatrix

Description: Write a program to find the submatrix with the maximum sum.

Sample Input:

```
[
  [1, 2, -1, -4, -20],
  [-8, -3, 4, 2, 1],
  [3, 8, 10, 1, 3],
  [-4, -1, 1, 7, -6]
]
```

Sample Output:

```
Maximum Sum: 29
Submatrix:
[
  [4, 2],
  [10, 1],
  [1, 7]
]
```

Explanation: The 3×2 submatrix from (1,2) to (3,3) has the maximum sum of 29.

Problem 15: Matrix Path with Obstacles

Description: Given a matrix with obstacles (1s) and empty spaces (0s), find the number of unique paths from the top-left corner to the bottom-right corner. You can only move right or down at each step.

Sample Input:

```
[
  [0, 0, 0],
  [0, 1, 0],
  [0, 0, 0]
]
```

Sample Output:

```
2
```

Explanation: There are two possible paths from top-left to bottom-right:

1. Right → Right → Down → Down
2. Down → Down → Right → Right The middle obstacle prevents the path: Right → Down → Right → Down.

Problem 16: Flood Fill Algorithm

Description: Implement a flood fill algorithm to fill connected, same-valued cells in a grid with a new value. It's similar to the "paint bucket fill" tool in paint programs.

Sample Input:

```
Image:
[
  [1, 1, 1],
  [1, 1, 0],
  [1, 0, 1]
]
Start Point: (1, 1)
New Color: 2
```

Sample Output:

```
[
  [2, 2, 2],
  [2, 2, 0],
  [2, 0, 1]
]
```

Explanation: Starting from position (1, 1) with value 1, all connected cells with value 1 are changed to value 2.

Problem 17: Count Islands in a Matrix

Description: Given a 2D grid where 1 represents land and 0 represents water, count the number of islands. An island is surrounded by water and is formed by connecting adjacent lands horizontally or vertically.

Sample Input:

```
[
  [1, 1, 0, 0, 0],
  [1, 1, 0, 0, 0],
  [0, 0, 1, 0, 0],
  [0, 0, 0, 1, 1]
]
```

Sample Output:

```
3
```

Explanation: There are three islands:

- Island 1: Top-left 2x2 grid of 1s
- Island 2: Single 1 in the middle
- Island 3: Bottom-right 2x1 grid of 1s

Problem 18: Check if a Matrix is a Magic Square

Description: A magic square is a square matrix where the sum of each row, each column, and both diagonals are equal. Write a program to check if a given matrix is a magic square.

Sample Input:

```
[
  [2, 7, 6],
  [9, 5, 1],
  [4, 3, 8]
]
```

Sample Output:

```
True
```

Explanation:

- Row sums: 15, 15, 15
- Column sums: 15, 15, 15
- Diagonal sums: 15, 15 Therefore, it's a magic square with magic sum 15.

Problem 19: Wave Traversal of Matrix

Description: Print the matrix in wave form, traversing from top to bottom for odd-indexed columns and bottom to top for even-indexed columns.

Sample Input:

```
[
  [1, 2, 3, 4],
  [5, 6, 7, 8],
  [9, 10, 11, 12]
]
```

Sample Output:

```
[1, 5, 9, 10, 6, 2, 3, 7, 11, 12, 8, 4]
```

Explanation:

- Column 0 (odd index): Top to bottom = 1, 5, 9
- Column 1 (even index): Bottom to top = 10, 6, 2
- Column 2 (odd index): Top to bottom = 3, 7, 11
- Column 3 (even index): Bottom to top = 12, 8, 4

Problem 20: Diagonal Traversal of Matrix

Description: Print the matrix in diagonal order, starting from the top-left element.

Sample Input:

```
[
  [1, 2, 3],
  [4, 5, 6],
  [7, 8, 9]
]
```

Sample Output:

```
[1, 2, 4, 7, 5, 3, 6, 8, 9]
```

Explanation: The elements are traversed in diagonal order:

- Diagonal 1: [1]

- Diagonal 2: [2, 4]
- Diagonal 3: [7, 5, 3]
- Diagonal 4: [6, 8]
- Diagonal 5: [9]