🔥 5 Array Problem Patterns (with Similar question)

BY: coding_error1

🧠 1. Two Pointers Pattern

🔸 **Main Problem:** Find if a sorted array has a pair with a given sum
Input: `[1, 2, 4, 4], target = 8` → ✅ `true`

🔁 **Similar Questions:**

1. Find all unique pairs with a given sum
2. Check if array has a triplet with zero sum (use 2 pointers inside loop)
3. Remove duplicates from sorted array
4. Merge two sorted arrays

<details> <summary>💻 Code (Python, C++, Java)</summary>

Python

```python
def has_pair(arr, target):
    left, right = 0, len(arr) - 1
    while left < right:
        s = arr[left] + arr[right]
        if s == target:
            return True
        elif s < target:
            left += 1
        else:
            right -= 1
    return False
```

C++

```cpp
bool hasPair(vector<int>& arr, int target) {
    int left = 0, right = arr.size() - 1;
    while (left < right) {
        int sum = arr[left] + arr[right];
        if (sum == target) return true;
        else if (sum < target) left++;
        else right--;
    }
    return false;
}
```

Java

```java
public boolean hasPair(int[] arr, int target) {
    int left = 0, right = arr.length - 1;
    while (left < right) {
        int sum = arr[left] + arr[right];
        if (sum == target) return true;
        else if (sum < target) left++;
        else right--;
    }
    return false;
}
```

🔁 2. Sliding Window Pattern

🔸 **Main Problem:** Maximum sum of subarray of size k
Input: `[2, 1, 5, 1, 3, 2], k = 3` → ✅ `9`

🔁 **Similar Questions:**

1. Minimum size subarray sum ≥ target
2. Longest substring with K distinct characters
3. Count number of substrings of size k with distinct elements

<details> <summary> 💻 Code (Python, C++, Java)</summary>

Python

```python
def max_sum_subarray(arr, k):
    window_sum = sum(arr[:k])
    max_sum = window_sum
    for i in range(k, len(arr)):
        window_sum += arr[i] - arr[i - k]
        max_sum = max(max_sum, window_sum)
    return max_sum
```

C++

```cpp
int maxSumSubarray(vector<int>& arr, int k) {
    int windowSum = 0, maxSum = 0;
    for (int i = 0; i < k; i++) windowSum += arr[i];
    maxSum = windowSum;
    for (int i = k; i < arr.size(); i++) {
        windowSum += arr[i] - arr[i - k];
        maxSum = max(maxSum, windowSum);
    }
    return maxSum;
}
```

Java

```
java

public int maxSumSubarray(int[] arr, int k) {
    int windowSum = 0, maxSum = 0;
    for (int i = 0; i < k; i++) windowSum += arr[i];
    maxSum = windowSum;
    for (int i = k; i < arr.length; i++) {
        windowSum += arr[i] - arr[i - k];
        maxSum = Math.max(maxSum, windowSum);
    }
    return maxSum;
}
```

🧮 3. Prefix Sum Pattern

🔸 **Main Problem:** Get sum in range [i, j] quickly using prefix
Input: `[1, 2, 3, 4], i = 1, j = 3` → ✅ `9`

🔁 **Similar Questions:**

1. Subarray sum equal to K (using hashmap + prefix)
2. Count subarrays with even/odd sum
3. Number of ways to split array into 3 parts with equal sum

<details> <summary>💻 Code (Python, C++, Java)</summary>

Python

```
python

def prefix_sum(arr):
    prefix = [0] * len(arr)
    prefix[0] = arr[0]
    for i in range(1, len(arr)):
        prefix[i] = prefix[i - 1] + arr[i]
    return prefix

def range_sum(prefix, i, j):
    return prefix[j] - (prefix[i - 1] if i > 0 else 0)
```

C++

```
cpp

vector<int> prefixSum(vector<int>& arr) {
    vector<int> prefix(arr.size());
    prefix[0] = arr[0];
    for (int i = 1; i < arr.size(); i++)
        prefix[i] = prefix[i - 1] + arr[i];
    return prefix;
}
```

```
int rangeSum(vector<int>& prefix, int i, int j) {
    return prefix[j] - (i > 0 ? prefix[i - 1] : 0);
}
```

Java

```
java

public int[] prefixSum(int[] arr) {
    int[] prefix = new int[arr.length];
    prefix[0] = arr[0];
    for (int i = 1; i < arr.length; i++)
        prefix[i] = prefix[i - 1] + arr[i];
    return prefix;
}

public int rangeSum(int[] prefix, int i, int j) {
    return prefix[j] - (i > 0 ? prefix[i - 1] : 0);
}
```

📃 4. HashMap / Frequency Count

🔸 **Main Problem:** Find first duplicate
Input: `[1,2,3,4,2]` → ✅ `2`

🔁 **Similar Questions:**

1. Majority element (element occurring > n/2 times)
2. Find all elements occurring > n/3 times
3. First non-repeating element
4. Group anagrams (use map of sorted string to list)

<details> <summary> 💻 Code (Python, C++, Java)</summary>

Python

```
python

def first_duplicate(arr):
    seen = set()
    for num in arr:
        if num in seen:
            return num
        seen.add(num)
    return -1
```

C++

```
cpp

int firstDuplicate(vector<int>& arr) {
```

```
    unordered_set<int> seen;
    for (int num : arr) {
        if (seen.count(num)) return num;
        seen.insert(num);
    }
    return -1;
}
```

Java

```
java

public int firstDuplicate(int[] arr) {
    Set<Integer> seen = new HashSet<>();
    for (int num : arr) {
        if (seen.contains(num)) return num;
        seen.add(num);
    }
    return -1;
}
```

🔄 5. Cyclic Sort Pattern

🔸 **Main Problem:** Find missing number from 0 to n
Input: `[3, 0, 1]` → ✅ `2`

🔁 **Similar Questions:**

1. Find all missing numbers in array [1, n]
2. Find duplicate number in array
3. Set mismatch (one number missing and one duplicate)

<details> <summary>💻 Code (Python, C++, Java)</summary>

Python

```
python

def missing_number(arr):
    n = len(arr)
    return n * (n + 1) // 2 - sum(arr)
```

C++

```
cpp

int missingNumber(vector<int>& arr) {
    int n = arr.size();
    int sum = accumulate(arr.begin(), arr.end(), 0);
    return n * (n + 1) / 2 - sum;
}
```

Java

```
java

public int missingNumber(int[] arr) {
    int n = arr.length, sum = 0;
    for (int num : arr) sum += num;
    return n * (n + 1) / 2 - sum;
}
```
</details>