1. What is the Prefix Sum Pattern?
The **Prefix Sum** of an array is a technique where you create a new array (or use a data structure) to store cumulative sums of elements from the start up to each index. Formally, for an array arr[] of size n, its prefix sum array prefix[] is defined as:
$prefix[i]=arr[0]+arr[1]+\ldots+arr[i]$

for $0 \le i < n0$

2. Why use Prefix Sum?
Prefix sums help quickly compute the sum of elements in any contiguous subarray in **O(1)** time after an **O(n)** preprocessing step.

- **Without prefix sums**: Sum of elements from index i to j is computed by looping from i to j, which is **O(j - i + 1)**.
- **With prefix sums**: Sum of elements from index i to j is:

$sum(i,j)=prefix[j]-prefix[i-1]. \quad (i>0)$

or simply

$prefix[j] \quad (i=0)$

which is **O(1)**.

3. How to Compute Prefix Sum?
Given an array arr of size n:

```
int[] prefix = new int[n];
prefix[0] = arr[0];
for (int i = 1; i < n; i++) {
   prefix[i] = prefix[i - 1] + arr[i];
}
```

4. Where can Prefix Sum be applied?
Prefix sums are useful in many types of problems such as:

- **Range sum queries**: Quickly compute sum of elements in subarrays multiple times.
- **Number of elements in a range** satisfying a condition.
- **Finding subarrays with a certain sum**.
- **2D prefix sums** for matrix sub-rectangle sums.
- **Difference arrays** and range update queries.
- Problems involving **cumulative frequencies**, histograms, or quick summation checks.

5. How to identify problems where Prefix Sum applies?

Look for questions that:

- Ask for **sum of elements in a range/subarray** multiple times.
- Need **fast repeated sum queries** after an initial array is given.
- Need to **find number of subarrays** satisfying certain sum-related properties.
- Involve **checking sums quickly** without recalculating sums for overlapping parts.
- Deal with **prefix-based conditions**, such as count of elements or cumulative constraints.

6. Benefits of Prefix Sum Pattern

- Reduces repeated work in sum calculations.
- Transforms $O(n^2)$ range sum queries into $O(n)$ preprocessing + $O(1)$ query.
- Simplifies problem logic by leveraging precomputed cumulative data.
- Helps in solving problems related to subarray sums, histogram calculations, and range queries efficiently.

#	Problem Name	LeetCode Link
1	Range Sum Query - Immutable	https://leetcode.com/problems/range-sum-query-immutable/
2	Subarray Sum Equals K	https://leetcode.com/problems/subarray-sum-equals-k/
3	Maximum Size Subarray Sum Equals k	https://leetcode.com/problems/maximum-size-subarray-sum-equals-k/
4	Find Pivot Index	https://leetcode.com/problems/find-pivot-index/
5	Count Number of Nice Subarrays	https://leetcode.com/problems/count-number-of-nice-subarrays/
6	Binary Subarrays With Sum	https://leetcode.com/problems/binary-

		subarrays-with-sum/
7	Minimum Size Subarray Sum	https://leetcode.com/problems/minimum-size-subarray-sum/
8	Maximum Average Subarray I	https://leetcode.com/problems/maximum-average-subarray-i/
9	Number of Subarrays with Bounded Maximum	https://leetcode.com/problems/number-of-subarrays-with-bounded-maximum/
10	Prefix and Suffix Search	https://leetcode.com/problems/prefix-and-suffix-search/
11	Continuous Subarray Sum	https://leetcode.com/problems/continuous-subarray-sum/
12	Longest Subarray of 1's After Deleting One Element	https://leetcode.com/problems/longest-subarray-of-1s-after-deleting-one-element/
13	Number of Subarrays with Odd Sum	https://leetcode.com/problems/number-of-subarrays-with-odd-sum/
14	Longest Subarray With Sum at Most K	https://leetcode.com/problems/longest-subarray-with-sum-at-most-k/
15	Find the Longest	https://leetcode.com/

Balanced Substring of Parentheses	problems/longest-valid-parentheses/