Consider

Input Array:

arr = [3, 5, 2, 7, 4]

Prefix Sum Array:

prefix = [3, 8, 10, 17, 21]

Explanation:

- prefix[0]=arr[0]=3
- prefix[1]=prefix[0]+arr[1]=3+5=8
- prefix[2]=prefix[1]+arr[2]=8+2=10
- prefix[3]=prefix[2]+arr[3]=10+7=17
- prefix[4]=prefix[3]+arr[4]=17+4=21

```
#include <iostream>
#include <vector>
using namespace std;

vector<int> computePrefixSum(const vector<int>& arr) {
   int n = arr.size();
   vector<int> prefix(n);
   prefix[0] = arr[0];

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

   return prefix;
}

int main() {
   vector<int> arr = {3, 5, 2, 7, 4};
   vector<int> prefix = computePrefixSum(arr);

   cout << "Prefix Sum: ";
   for (int sum : prefix) {
      cout << sum << " ";
   }

   return 0;
}
```

Java

```
import java.util.Arrays;

public class PrefixSum {
```

```
    public static int[] computePrefixSum(int[] arr) {
        int n = arr.length;
        int[] prefix = new int[n];
        prefix[0] = arr[0];

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

        return prefix;
    }

    public static void main(String[] args) {
        int[] arr = {3, 5, 2, 7, 4};
        int[] prefix = computePrefixSum(arr);

        System.out.println("Prefix Sum: " + Arrays.toString(prefix));
    }
}
```

Applications

Applications

1. **Efficient Range Sum Queries**:
    - To compute the sum of elements from index l to r, use: sum[l to r]=prefix[r]−prefix[l−1]
    - If l=0 then sum[l to r]=prefix[r].
2. **Finding Subarray Sums**:
    - Quickly calculate the sum of any subarray in O(1) time after constructing the prefix sum array.
3. **Sliding Window Optimization**:
    - Often combined with the sliding window technique for certain optimizations.