Table of Contents

1. Introduction

Prefix sum is a technique used in computer science and mathematics to efficiently compute and store the cumulative sum of elements in an array. The prefix sum array is derived from the input array in such a way that each element in the prefix sum array represents the sum of all elements up to that particular index in the original array.

Definition: A prefix sum is an array derived from the input array such that the element at each index in the prefix sum array represents the sum of all elements up to that index in the original array.

Formula: Given an input array `arr` and its corresponding prefix sum array `prefix_sum`, the relationship is defined as follows:

prefix_sum$[i]$ = $\sum_{j=0}^{i} arr[j]$

2. Algorithm

1. **Initialization:**
    - Create a new array `prefix_sum` of the same length as the input array.
    - Initialize the first element of `prefix_sum` as the same as the first element of the input array.
2. **Compute Prefix Sum:**
    - Iterate through the input array from the second element onward.
    - At each index (i), update `prefix_sum[i]` as the sum of `prefix_sum[i-1]` and `arr[i]`.
3. **Result:**
    - The `prefix_sum` array now holds the cumulative sum of the elements up to each index in the original array.

3. Example Usage

Scenario:

Consider a scenario where you need to efficiently calculate the sum of elements within a specific range ([l, r]) multiple times for a given array. The prefix sum can be precomputed to achieve this efficiently.

Implementation:

```
def calculate_prefix_sum(arr):
    n = len(arr)
    prefix_sum = [0] * n
    prefix_sum[0] = arr[0]

    for i in range(1, n):
        prefix_sum[i] = prefix_sum[i-1] + arr[i]

    return prefix_sum

# Example Usage:
input_array = [1, 2, 3, 4, 5]
prefix_sum_array = calculate_prefix_sum(input_array)

# Now, prefix_sum_array = [1, 3, 6, 10, 15]
```

Now, with the `prefix_sum` array, you can efficiently calculate the sum of elements within a specific range ([l, r]) by simply subtracting `prefix_sum[l-1]` from `prefix_sum[r]`.

4. Advantages

Advantages of the prefix sum technique:

1. **Efficient Range Sum Queries:**
    - Prefix sum allows for constant-time computation of the sum of elements within a specific range, making it particularly useful in scenarios where range queries are frequent.
2. **Optimized Time Complexity:**

- By precomputing prefix sums, certain computations that involve cumulative sums can be optimized, leading to improved time complexity.

3. **Space Complexity Trade-off:**
    - While the prefix sum introduces additional space complexity to store the cumulative sums, it can result in significant time savings for repeated sum calculations, making it a space-efficient trade-off.