Question 2: Range Sum Queries using Prefix Sum Array

Problem

Write a program to find the sum of elements in a given range [L, R] using a prefix sum array.

Algorithm

1. Build the prefix sum array P from the original array A:
   - P[0] = A[0]
   - P[i] = P[i-1] + A[i] for i > 0
2. To find the sum of elements from index L to R (inclusive):
   - If L is 0: return P[R]
   - Otherwise: return P[R] - P[L-1]

Program Implementation

cpp

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

vector<int> buildPrefixSum(const vector<int>& arr) {
    int n = arr.size();
    vector<int> prefixSum(n);

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

    return prefixSum;
}

int rangeSum(const vector<int>& prefixSum, int L, int R) {
    if (L == 0) {
        return prefixSum[R];
    } else {
        return prefixSum[R] - prefixSum[L-1];
    }
}

int main() {
    vector<int> arr = {2, 4, 5, 1, 6, 8, 3, 9};
    vector<int> prefixSum = buildPrefixSum(arr);

    int L1 = 1, R1 = 4;
    cout << "Sum of elements from index " << L1 << " to " << R1 << ": "
         << rangeSum(prefixSum, L1, R1) << endl;

    int L2 = 0, R2 = 7;
    cout << "Sum of elements from index " << L2 << " to " << R2 << ": "
         << rangeSum(prefixSum, L2, R2) << endl;

    int L3 = 3, R3 = 6;
    cout << "Sum of elements from index " << L3 << " to " << R3 << ": "
         << rangeSum(prefixSum, L3, R3) << endl;

    return 0;
}
```

Time Complexity

- Building the prefix sum array: O(n) where n is the size of the array
- Each range sum query: O(1) - constant time

Space Complexity

- O(n) for storing the prefix sum array

Example Explanation

Let's trace through an example using the array [2, 4, 5, 1, 6, 8, 3, 9]:

1. First, build the prefix sum array: Prefix Sum: [2, 6, 11, 12, 18, 26, 29, 38]
2. For the range [1, 4] (second element to fifth element):
   - Sum = P[4] - P[0] = 18 - 2 = 16
   - Verification: 4 + 5 + 1 + 6 = 16
3. For the range [0, 7] (entire array):
   - Sum = P[7] = 38
   - Verification: 2 + 4 + 5 + 1 + 6 + 8 + 3 + 9 = 38
4. For the range [3, 6]:
   - Sum = P[6] - P[2] = 29 - 11 = 18
   - Verification: 1 + 6 + 8 + 3 = 18

This demonstrates how prefix sum arrays allow us to calculate range sums in constant time after the O(n) preprocessing.