4. Prefix Sum (Suma de Prefijos)

La técnica de **Prefix Sum (Suma de Prefijos)** es una técnica algorítmica fundamental utilizada para procesar de manera eficiente consultas de suma sobre rangos (subarrays o sublistas) en una estructura de datos lineal. Su objetivo principal es reducir el tiempo de cálculo de estas sumas de $\mathcal{O}(N)$ por cada consulta a $\mathcal{O}(1)$, después de un preprocesamiento inicial de $\mathcal{O}(N)$.

—

0.1. ¿Qué es la Técnica de Prefix Sum?

La idea central de Prefix Sum es crear un array auxiliar (llamado array de suma de prefijos o `prefixSumArray`) donde cada elemento almacena la suma acumulada de los elementos originales desde el inicio hasta esa posición.

Si tienes un array original $arr = [a_0, a_1, a_2, \ldots, a_{N-1}]$, su array de suma de prefijos `prefixSumArray` se construye de la siguiente manera:

- `prefixSumArray[0] = arr[0]`
- `prefixSumArray[i] = arr[i] + prefixSumArray[i-1]` para $i > 0$

Esto significa que `prefixSumArray[i]` contiene la suma de todos los elementos desde `arr[0]` hasta `arr[i]`.

—

0.2. ¿Cómo Funciona la Técnica de Prefix Sum?

Una vez que el `prefixSumArray` ha sido construido, calcular la suma de cualquier rango $[L, R]$ (desde el índice L hasta el índice R, ambos inclusive) se vuelve trivial y muy rápido:

- La suma de $arr[L \ldots R]$ es simplemente `prefixSumArray[R] - prefixSumArray[L-1]`.
- Para el caso especial donde $L = 0$, la suma es simplemente `prefixSumArray[R]`. Para simplificar la fórmula, a menudo se inicializa `prefixSumArray[0]` como 0 (o se usa un array 1-indexado y `prefixSumArray[0]` es 0, y `prefixSumArray[i]` es la suma hasta `arr[i-1]`), de modo que `prefixSumArray[R] - prefixSumArray[L-1]` siempre funcione.

Visualización:
`arr = [2, 4, 1, 3, 5]`
`prefixSumArray = [2, 6, 7, 10, 15]`
Para calcular la suma del rango $[1, 3]$ (es decir, $arr[1] + arr[2] + arr[3] = 4 + 1 + 3 = 8$):
`prefixSumArray[3] - prefixSumArray[1-1] = prefixSumArray[3] - prefixSumArray[0] = 10 - 2 = 8`

—

0.3. Ejemplo Clásico de Prefix Sum en C++

Veamos un ejemplo de cómo implementar y usar la técnica de Prefix Sum.

Suma de Rangos en un Array

Dado un array de números, precalcula las sumas de prefijo y luego responde eficientemente a múltiples consultas de suma de rango.

Listing 1: Cálculo y uso del array de suma de prefijos

```cpp
#include <iostream>
#include <vector>
#include <numeric> // No es estrictamente necesario, pero
      til  para otros escenarios

using namespace std; // Incluyendo namespace std como se
    solicit

// Funci n para construir el array de suma de prefijos
vector<int> buildPrefixSumArray(const vector<int>& arr) {
    int n = arr.size();
    vector<int> prefixSum(n);

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

// Funci n para obtener la suma de un rango [L, R]
int getRangeSum(const vector<int>& prefixSum, int L, int
    R) {
    if (L < 0 || R >= prefixSum.size() || L > R) {
        // Manejo de errores o caso inv lido
        cerr << "Rango inv lido." << endl;
        return 0; // O lanzar una excepci n
    }

    if (L == 0) {
        return prefixSum[R];
    } else {
        return prefixSum[R] - prefixSum[L-1];
    }
}

int main() {
    vector<int> original_array = {10, 20, 30, 40, 50};
```

```
38
39      cout << "Array original: ";
40      for (int x : original_array) {
41          cout << x << " ";
42      }
43      cout << endl;
44
45      // Construir el array de suma de prefijos
46      vector<int> prefix_sum_array = buildPrefixSumArray(
            original_array);
47
48      cout << "Array de Suma de Prefijos: ";
49      for (int x : prefix_sum_array) {
50          cout << x << " ";
51      }
52      cout << endl;
53
54      // Consultas de suma de rango
55      cout << "\nSuma del rango [0, 2]: " << getRangeSum(
            prefix_sum_array, 0, 2) << endl;
56      cout << "Suma del rango [1, 3]: " << getRangeSum(
            prefix_sum_array, 1, 3) << endl;
57      cout << "Suma del rango [3, 4]: " << getRangeSum(
            prefix_sum_array, 3, 4) << endl;
58      cout << "Suma del rango [0, 4]: " << getRangeSum(
            prefix_sum_array, 0, 4) << endl;
59
60      return 0;
61  }
```

0.4. Análisis de Eficiencia de Prefix Sum

- **Preprocesamiento (Construcción del array de suma de prefijos):**
  - **Complejidad Temporal:** $\mathcal{O}(N)$, donde N es el número de elementos en el array original. Se necesita una única pasada para calcular todas las sumas acumuladas.
  - **Complejidad Espacial:** $\mathcal{O}(N)$, ya que se crea un nuevo array del mismo tamaño que el original para almacenar las sumas de prefijo.
- **Consultas de Suma de Rango:**
  - **Complejidad Temporal:** $\mathcal{O}(1)$ por cada consulta. Una vez que el array de suma de prefijos está construido, cualquier suma de rango se puede obtener con solo una o dos operaciones de resta/acceso a array.

- **Complejidad Espacial:** $\mathcal{O}(1)$ adicional por consulta (sin contar el `prefixSumArray` ya creado).

La principal ventaja de Prefix Sum radica en su capacidad para responder a un gran número de consultas de rango de manera extremadamente rápida después de un costo inicial razonable.

—

0.5. Aplicaciones Comunes de Prefix Sum

La técnica de Prefix Sum es la base para resolver muchos problemas de manera eficiente, no solo sumas directas, sino también para:

- Encontrar el subarray con la suma máxima/mínima.
- Problemas que involucran promedios de rangos.
- Balancear arrays o verificar propiedades de suma.
- En problemas 2D (matrices), se extiende a la **Suma de Prefijos 2D** o **Área de Suma Acumulativa** para calcular sumas de submatrices en $\mathcal{O}(1)$.
- Optimizar ciertas aplicaciones de la técnica de Sliding Window donde la suma de la ventana es una métrica clave.