GUÍA DE APRENDIZAJE PARA CONCURSANTES ICPC Y IOI: SUMA DE PREFIJOS (*PREFIX SUM*)

---

1. Introducción

En varios problemas se nos da una colección de valores los cuales nunca van a actualizarse y sobre los cuales se van aplicar alguna operación o función conmutativa del tipo $(l, r,)$ siendo r y l los extremos del rango de las posiciones dentro de la colección para cuyos valores se le aplicará la operación. Para resolver este tipo de problemas vamos aplicar una estructura de datos para computar sumas de rangos.

2. Conocimientos previos

2.1. Arreglos y Matrices

Un arreglo o matriz es una colección ordenada de datos (tanto primitivos u objetos dependiendo del lenguaje). Los arreglos o matrices se emplean para almacenar múltiples valores en una sola variable, frente a las variables que sólo pueden almacenar un valor (por cada variable).

Estas estructuras de datos son adecuadas para situaciones en las que el acceso a los datos se realice de forma aleatoria e impredecible. Por el contrario, si los elementos pueden estar ordenados y se va a utilizar acceso secuencial sería más adecuado utilizar una lista, ya que esta estructura puede cambiar de tamaño fácilmente durante la ejecución de un programa, siendo esta última una estructura dinámica (al no tener un tamaño definido)

2.2. Función conmutativa

En matemáticas, la propiedad conmutativa o conmutatividad es una propiedad fundamental que tienen algunas operaciones según la cual el resultado de operar dos elementos no depende del orden en el que se toman. Esto se cumple en la adición y la multiplicación ordinarias: *el orden de los sumandos no altera la suma,o el orden de los factores no altera el producto*.

3. Desarrollo

Podemos fácilmente procesar las consultas de sumas en un arreglo estático construyendo un arreglo de suma en prefijos(prefix sum). Donde cada valor en el prefix array es igual a la suma en el arreglo original hasta esa posición. Por ejemplo tenemos que el valor en la posicion k es la función $sum_q(0, k)$.

Este prefix sum array puede ser construido en una complejidad $O(n)$.

Consideremos el siguiente ejemplo:

Indice	0	1	2	3	4	5	6	7
Valor	1	3	4	8	6	1	4	2

La suma de prefijo correspondiente al arreglo anterior es:

Indice	0	1	2	3	4	5	6	7
Valor	1	4	8	16	22	23	27	29

Ahora como el prefix sum array contiene todos los valores de $sum_q(0,k)$, entonces podemos calcular todos los valores de $sum_q(a,b)$ en tiempo $O(1)$ de la manera siguiente:

$$sum_q(a,b) = sum_q(0,b) - sum_q(0,a-1)$$

A continuación veamos un ejemplo de como funcionan estas consultas. Consideremos que queremos obtener la suma en el rango $[3,6]$.

Indice	0	1	2	3	4	5	6	7
Valor	1	3	4	8	6	1	4	2

En este caso $sum_q(3,6) = 8+6+1+4 = 19$. Esta suma puede ser calculada utilizando los dos valores del prefix sum array señalados.

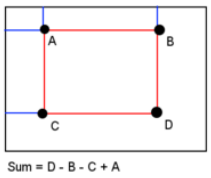
Indice	0	1	2	3	4	5	6	7
Valor	1	4	8	16	22	23	27	29

Entonces, $sum_q(3,6) = sum_q(0,6) - sum_q(0,2) = 27 - 8 = 19$.

3.1. Suma de prefijos en dos dimensiones

Tambien podemos generalizar la idea anterior para poder computar en más dimensiones. Por ejemplo podemos construir un prefix sum bidimensional que puede ser usado para calcular la suma de cualquier subarreglo rectangular en tiempo $O(1)$. Donde cada suma en tal arreglo corresponde al subarreglo que comienza en la esquina superior izquierda de la matriz y termina en la posicion de la matriz a donde accedemos.

El siguiente esquema ilustra la idea anterior:



Sum = D - B - C + A

La suma de prefijo se puede calcular de manera eficiente en un solo paso sobre la matriz, ya que el valor en el prefijo de suma en (i,j) es simplemente:

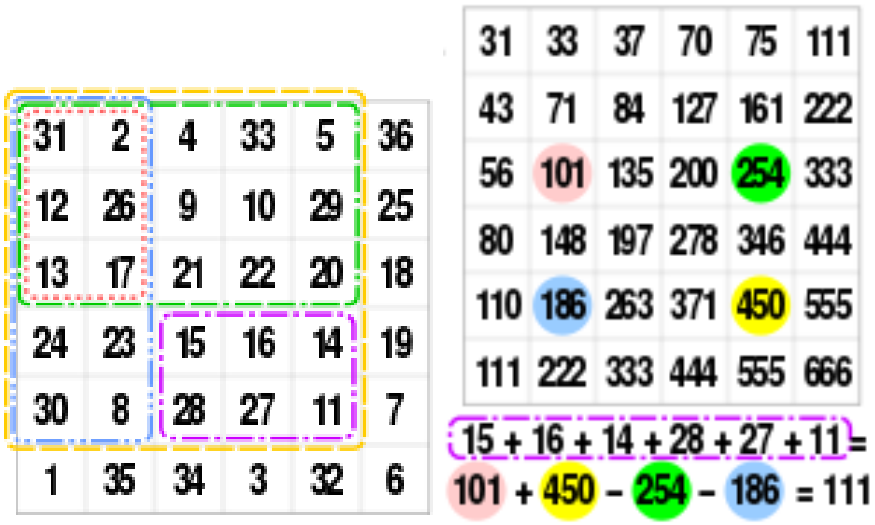
$$PS(i,j) = m(i,j) + PS(i,j-1) + PS(i-1,j) - PS(i-1,j-1)$$

Siendo SP el prefijo de suma , m la matriz con los valores originales y i , j indican una posición dentro de la matriz. Se bebe tener en cuenta que la matriz de suma de prefijo se calcula desde la esquina superior izquierda.

Una vez que se ha calculado la matriz de suma de prefijo , la evaluación de la suma sobre cualquier área rectangular requiere exactamente cuatro referencias de matriz, independientemente del tamaño del área. Es decir, tiene $A = (i_0, j_0)$, $B = (i_1, j_0)$, $C = (i_0, j_1)$ y $D = (i_1, j_1)$, la suma de $p(i,j)$ sobre el rectángulo formado por A, B, C y D es:

$$\sum_{i_0<i\leq i_1} \sum_{j_0<j\leq j_1} p(i,j) = PS(D) + PS(A) - PS(B) - PS(C)$$

La siguientes gráficas ilustran mejor la idea:



4. Implementación

4.1. C++

4.1.1. Suma de prefijo en una dimensión

```
#define ENDL '\n'
#define MAX_N 100010
using namespace std;

int n,nquerys,a,b;
int array[MAX_N],ps[MAX_N];

int main() {
   //leyendo los datos y construyendo la estructura
   cin>>n; ps[0]=0;
   for(int i=1;i<=n;i++){
      cin>>array[i];
      ps[i]=array[i]+ps[i-1];
   }

   //respondiendo la consultas
   cin>nquerys;
```

```
    for(int i=0;i<nquerys;i++){
        cin>>a>>b; //siempre a<=b
        cout<<ps[b]-ps[a-1]<<ENDL;
    }
    return 0;
}
```

4.1.2. Suma de prefijo en dos dimensiones

```
#define ENDL '\n'
#define MAX_N 100010
using namespace std;

int n,m,nquerys,I1,I2,J1,J2;
int matrix[MAX_N][MAX_N],ps[MAX_N][MAX_N];

int main() {
    //leyendo los datos y construyendo la estructura
    cin>>n>>m;

    fill(ps[0],ps[0]+MAX_N,0);
    for(int i=1;i<=n;i++){
        ps[i][0]=0;
        for(int j=1;j<=m;j++){
            cin>>matrix[i][j];
            ps[i][j]=matrix[i][j]+ps[i][j-1]+ps[i-1][j]-ps[i-1][j-1];
        }
    }
    //respondiendo la consultas
    cin>nquerys;

    for(int i=0;i<nquerys;i++){
        cin>>I1>>J1;//esquina superior izquierda
        cin>>I2>>J2;//esquina inferior derecha
        cout<<ps[I2][J2]+ps[I1-1][J1-1]-ps[I2][J1-1]-ps[I1-1][J2]<<ENDL;
    }
    return 0;
}
```

4.2. Java

4.2.1. Suma de prefijo en una dimensión

```
public static void main(String[] args) {
    int n, nquerys, a, b;
    Scanner in = new Scanner(System.in);
    // leyendo los datos y construyendo la estructura
```

```
    n = in.nextInt();
    int[] ps = new int[n + 1];
    int[] array = new int[n + 1];

    ps[0] = 0;
    for (int i = 1; i <= n; i++) {
        array[i] = in.nextInt();
        ps[i] = array[i] + ps[i - 1];
    }

    // respondiendo la consultas
    nquerys = in.nextInt();

    for (int i = 0; i < nquerys; i++) {
        // siempre a<=b
        a = in.nextInt();
        b = in.nextInt();
        System.out.println(ps[b] - ps[a - 1]);
    }
}
```

4.2.2. Suma de prefijo en dos dimensiones

```
public static void main(String[] args){
    int n, m, nquerys, I1, I2, J1, J2;
    Scanner in = new Scanner(System.in);
    // leyendo los datos y construyendo la estructura
    n = in.nextInt();
    m = in.nextInt();
    int[][] matrix = new int[n + 1][m + 1];
    int[][] ps = new int[n + 1][m + 1];

    Arrays.fill(ps[0], 0);
    for (int i = 1; i <= n; i++) {
        ps[i][0] = 0;
        for (int j = 1; j <= m; j++) {
            matrix[i][j] = in.nextInt();
            ps[i][j] = matrix[i][j]+ps[i][j-1]+ps[i-1][j]-ps[i-1][j-1];
        }
    }
    //respondiendo la consultas
    nquerys = in.nextInt();

    for (int i = 0; i < nquerys; i++) {
        // esquina superior izquierda
        I1 = in.nextInt(); J1 = in.nextInt();
        // esquina inferior derecha
        I2 = in.nextInt(); J2 = in.nextInt();
```

```
            System.out.println(ps[I2][J2]+ps[I1-1][J1-1]-ps[I2][J1-1]-ps[I1-1][J2]);
        }
    }
```

5. Aplicaciones

Entre las aplicaciones de las suma de prefijo podemos citar:

- **Encontrar si hay una subarray con 0 sumas:** dado un arreglo de números positivos y negativos, encuentrar si hay una subarrray (de tamaño al menos uno) con 0 suma.
- **Índice de equilibrio de un arreglo:** el índice de equilibrio de un arreglo es un índice tal que la suma de los elementos en los índices más bajos es igual a la suma de los elementos en los índices más altos.
- **Tamaño máximo del subarreglo, tal que todos los subarreglos de ese tamaño tengan una suma menor que k:** dado un arreglo de n enteros positivos y un entero positivo k, la tarea es encontrar el tamaño máximo del subarreglo tal que todos los subarreglos de ese tamaño tengan la suma de elementos menores que k.
- **Encuentre los números primos que se pueden escribir como la suma de la mayoría de los números primos consecutivos:** dado una serie de límites. Para cada límite, encuentre el número primo que se puede escribir como la suma de la mayoría de los primos consecutivos menores o iguales que el límite.
- **Intervalo más largo con la misma suma en dos arreglos binarios:** dados dos arreglos binarias, $arr1[]$ y $arr2[]$ del mismo tamaño n. Encuentre la longitud del tramo común más largo (i, j) donde $j \geq i$ tal que $arr1[i]+arr1[i+1]+\cdots+arr1[j] = arr2[i]+arr2[i+1]+\cdots+arr2[j]$.
- **Máxima suma de subarreglo módulo m:** dado un arreglo de n elementos y un entero m. La tarea es encontrar el valor máximo de la suma de su subarreglo módulo m, es decir, encontrar la suma de cada subarreglo módulo m e imprimir el valor máximo de esta operación de módulo.
- **Tamaño máximo del subarreglo, tal que todos los subarreglos de ese tamaño tengan una suma menor que k:** dado un arreglo de n enteros positivos y un entero positivo k, la tarea es encontrar el tamaño máximo del subarreglo tal que todos los subarreglos de ese tamaño tengan la suma de elementos menores que k.
- **Entero máximo ocurrido en n rangos:** dados n rangos de la forma L y R, la tarea es encontrar el entero máximo que ocurre en todos los rangos. Si sale más de uno de estos enteros, imprima el más pequeño.
- **Costo mínimo para adquirir todas las monedas con k monedas extra permitidas con cada moneda:** se le da una lista de N monedas de diferentes denominaciones. puede pagar una cantidad equivalente a cualquier moneda 1 y puede adquirir esa moneda. Además, una vez

que hayamos pagado una moneda, podremos elegir como máximo K monedas más y podremos adquirirlas gratis. La tarea es encontrar la cantidad mínima requerida para adquirir todas las N monedas por un valor dado de K.

- **Generador de números aleatorios en forma de distribución de probabilidad arbitraria:** dados n números, cada uno con cierta frecuencia de ocurrencia. Devuelve un número aleatorio con una probabilidad proporcional a su frecuencia de aparición.

6. Complejidad

La complejidad del prefijo se suma esta dado basicamente por las dos operaciones que realiza. La primera es la construción del prefijo de suma el cual tiene una complejidad de O(N) cuando es de una dimensión y O(NM) cuando es dos dimensiones siendo N y M las dimensiones de los arreglos o matrices sobre la cual se construyo. La otra operación es la consulta que independientemente de las dimensiones siempre será O(1).

7. Ejercicios propuestos

A continuación de una lista de ejercicios o problemas que se pueden resolver aplicando la técnica descrita en esta guía.

- DMOJ - Los Tablones de la Valla
- DMOJ - Campos de tabaco
- DMOJ - Cuadrado Máximo
- DMOJ - Rectángulo Maximo
- DMOJ - Tarea de impacto
- DMOJ - Breed Counting