FEBRERO 4

SEGMENT TREE

① PROBLEMA DE MOTIVACIÓN

Dado un arreglo de N elementos, cada uno de ellos un entero, se desea responder Q consultas del tipo ¿Cuál es la suma del rango $[i,j]$ es igual a la suma de $a_i + a_{i+1} + \dots + a_{j-1} + a_j$ $(i \le j)$

$$\begin{bmatrix} 1 \le N \le 10^5 \\ 1 \le Q \le 10^5 \end{bmatrix}$$

Ej) $N = 7$

	0	1	2	3	4	5	6
v	3	2	1	9	-1	3	-2

$Q[2,4] = 9$ $\quad Q[4,4] = -1$

$Q[0,3] = 15$

a) Solución 1: Fuerza Bruta

* Hacer la suma $\mathcal{O}(N)$
* Hay Q consultas

$\Big\} \Rightarrow \mathcal{O}(QN)$

b) Solucion 2: Suma de Prefijos

	0	1	2	3	4	5	6
v	3	2	1	9	-1	3	-2

preproceso ↓

	0	1	2	3	4	5	6
acum	3	5	6	15	14	17	15

acum [j] guardará la suma de $a_0 + a_1 + \dots + a_j$

↳ $Q[0,j]$

¿ $Q[i,j]$?

$$\begin{cases} \text{acum}[j] & \text{si } i = 0 \\ \text{acum}[j] - \text{acum}[i-1] & \text{si } i > 0 \end{cases}$$

	0	1	2	3	4	5	6
v	3	2	1	9	-1	3	-2

	0	1	2	3	4	5	6
acum	3	5	6	15	14	17	15

$Q[2,4]$
$=$
$\text{acum}[4] - \text{acum}[1]$
$=$
$14 - 5 = 9$

Complejidad

Preproceso $\mathcal{O}(N)$

Por Consulta $\mathcal{O}(1)$

Total Consultas $\mathcal{O}(Q)$

Total $\mathcal{O}(N+Q)$

Observaciones.

1) La operación tiene que ser asociativa

2) La operación debe tener inverso → El neutro es único

(SI) SUMA $+(a,b)$ ¿NEUTRO? $+(a,0) = a$

¿INVERSO? $+(a,-a) = 0$

(NO) MENOR $\min(a,b)$ ¿NEUTRO? $\min(a,\infty) = a$

¿INVERSO? $\min(a,?) = \infty$

No Existe

(NO) GCD $\gcd(a,b)$ ¿NEUTRO? $\gcd(a,0) = a$ ... $\gcd(a,a) = a$

¿INVERSO? $\gcd(a,?) = 0$ (=> <=)

3) No funciona si las consultas incluyen una actualización de datos

	0	1	2	3	4	5	6
v	3	2	1	9	-1	3	-2
acum	3	5	6	15	14	17	15

pos val

update(3, 6)

↓

$O(N)$

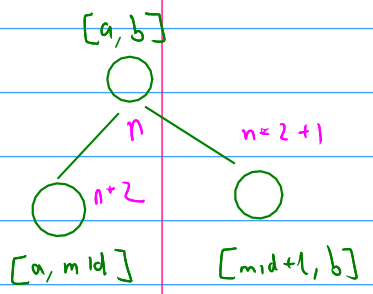
	0	1	2	3	4	5	6
v	3	2	1	6	-1	3	-2
acum	3	5	6	12	11	14	12

② Segment Tree (Árbol de Segmentos)

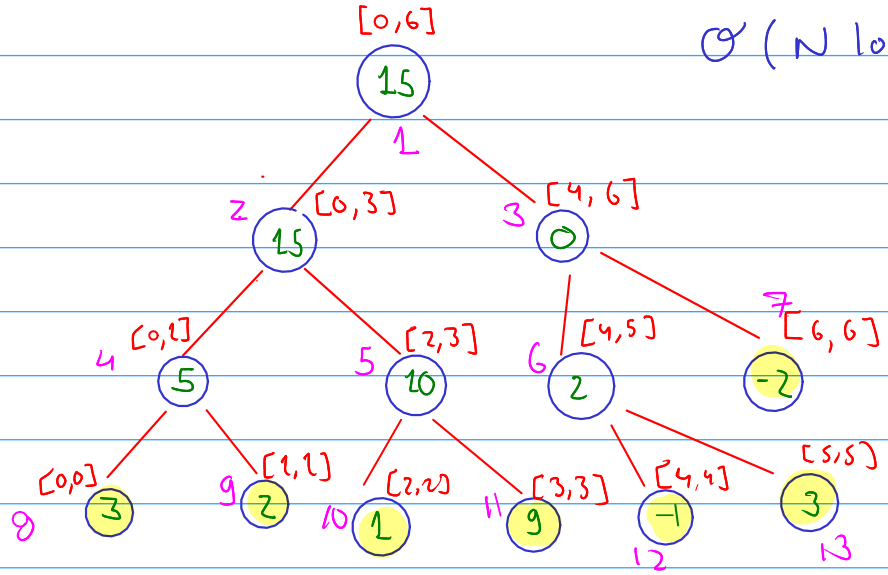
	0	1	2	3	4	5	6
v	3	2	1	9	-1	3	-2

a) Idea

* Construcción



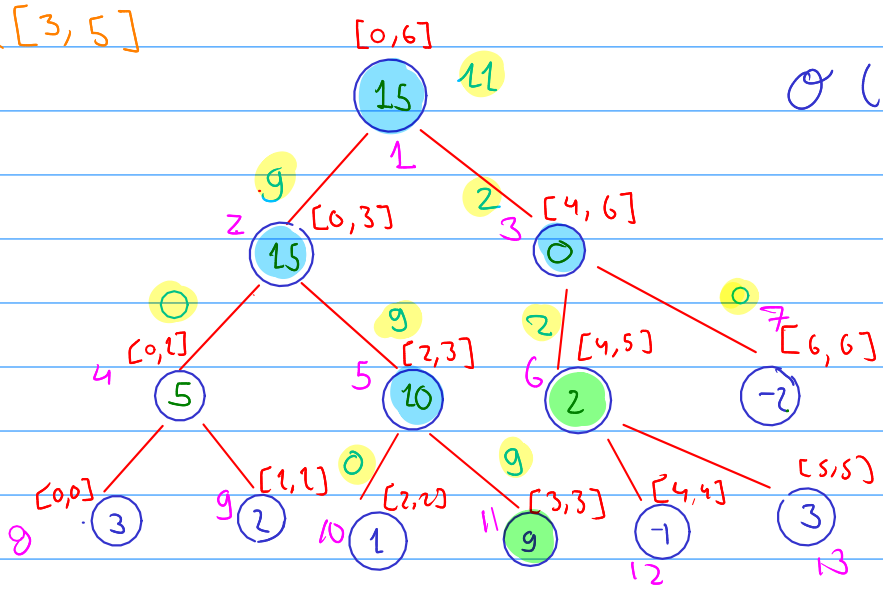
$mid = \frac{a+b}{2}$



$\mathcal{O}(N \log N)$

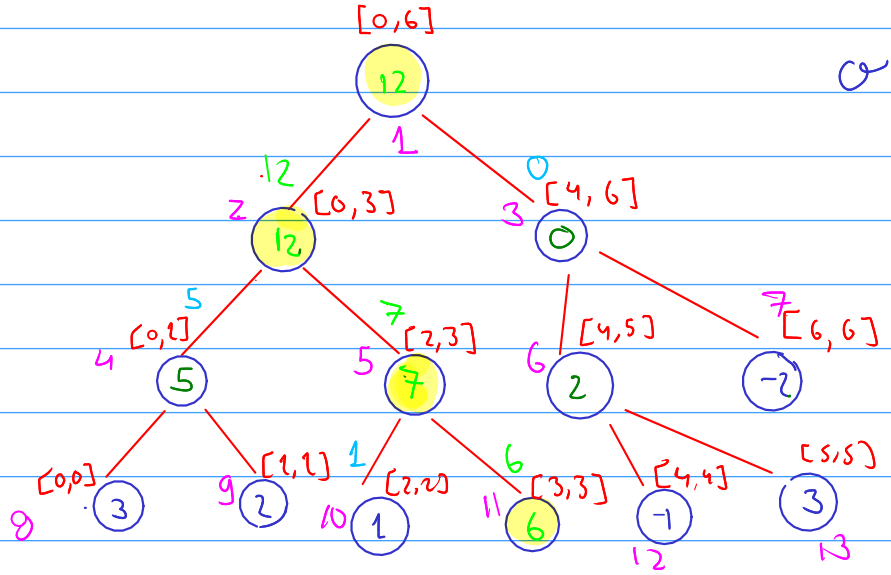
* Consulta

Q [3, 5]



$\mathcal{O}(\log N)$

* Actualización update (3, 6)
pos val

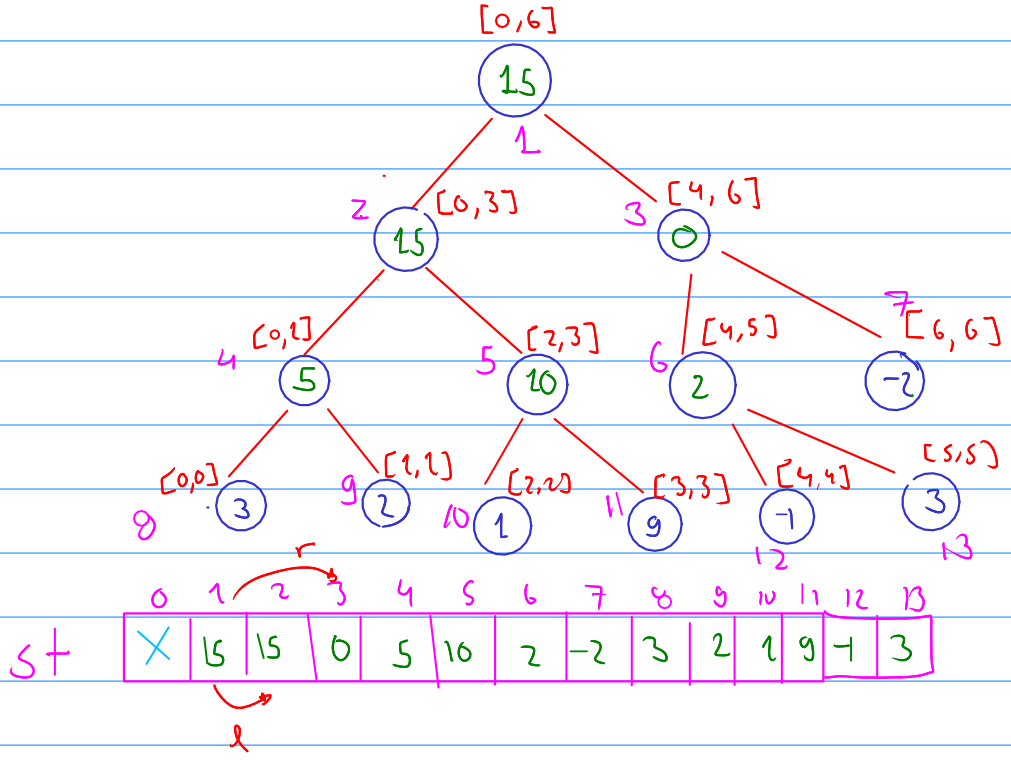


$O(\log N)$

Tiempo total $O(N \log N + Q \log N)$

$O((N+Q) \log N)$

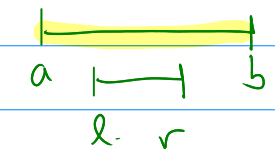
Implementación



* Construcción

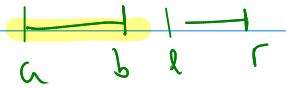
```cpp
void build(int nodo,int l,int r){
    if(l==r){
        st[nodo] = v[l];
        return;
    }
    int mid = (l+r)/2;
    // Generamos nodo izquierda
    build(nodo*2,l,mid);
    // Generamos nodo derecha
    build(nodo*2+1,mid+1,r);
    // Construimos el nodo
    st[nodo] = st[nodo*2] + st[nodo*2+1];
}
```

* Consulta



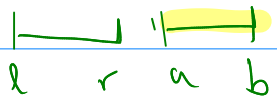
Caso Base : retornamos la información de todo el rango $[l,r]$

$[l,r] \subseteq [a,b]$



Caso Base : retornamos el elemento neutro

$[a,b] \cap [l,r] = \emptyset$



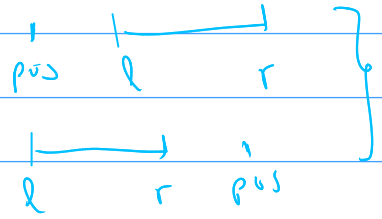

```cpp
int query(int nodo,int l,int r,int a,int b){
    if(a<=l && r<=b) return st[nodo];
    if(b<l || r<a) return 0; // Elemento Neutro de la operacion
    int mid = (l+r)/2;
    // Respuesta del izquierdo
    int r1 = query(nodo*2,l,mid,a,b);
    // Respueta del derecho
    int r2 = query(nodo*2+1,mid+1,r,a,b);
    // Retornamos la union de ambas respuestas
    return r1 + r2;
}
```

* Actualización (pos, val)

Caso Base

pos = l = r → Llegar al nodo que quiero actualizar
st[nodo] = val;

Caso Base



Estoy fuera del rango y acabo la búsqueda.

```
void update(int nodo, int l, int r, int pos, int val){
    if(l==r and l==pos){
        st[nodo] = val;
        return;
    }
    if(pos<l || r<pos) return;
    int mid = (l+r)/2;
    // Actualizamos nodo izquierda
    update(nodo*2,l,mid,pos,val);
    // Actualizamos nodo derecha
    update(nodo*2+1,mid+1,r,pos,val);
    // Actualizamos el nodo
    st[nodo] = st[nodo*2] + st[nodo*2+1];
}
```