Estructura de datos “Segment Tree”

Agustín Santiago Gutiérrez

Olimpíada Informática Argentina 2015

Contenidos

Contenidos

Información acumulada en rangos

- Supongamos que tenemos un arreglo v de n posiciones. Una pregunta natural que podemos hacernos es: ¿Cuál es la suma de los elementos en el rango $[i, j)$?
- Podemos notar $suma(i, j)$ a dicha suma. Así tendríamos por ejemplo:

$$\begin{array}{rcccccccccccccc}
i & : & 0 & 1 & 2 & & 3 & & 4 & & 5 & & 6 & & \\
v[i] & : & 2 & 6 & 2 & & -4 & & 6 & & 4 & & 3 & & \\
suma(2,5) & = & & & 2 & + & (-4) & + & 6 & & & & & = & 4 \\
suma(3,7) & = & & & & & (-4) & + & 6 & + & 4 & + & 3 & = & 9 \\
suma(5,5) & = & & & & & & & & & & & & = & 0
\end{array}$$

Notar que los índices del arreglo van de 0 a $n-1$, y una pregunta $[i, j)$ válida tiene $0 \leq i \leq j \leq n$

Solución trivial

- La solución más natural al problema anterior es realizar una iteración (`for`) del rango $[i, j)$, calculando la suma total y devolviéndola ante cada pregunta.
- La complejidad de esta solución es $O(Q \cdot n)$, siendo Q la cantidad de preguntas que nos interesa responder.
- El problema de esta solución es que es costosa en tiempo de ejecución cuando nos interesa hacer muchas preguntas sobre el mismo arreglo.

Solución mejorada (tabla aditiva)

- Esta solución puede mejorarse mediante el cómputo de una *tabla aditiva*
- Definimos un nuevo arreglo V de $n+1$ posiciones, con $V[i] = suma(0,i) = \sum_{j=0}^{i-1} v[j]$
- La clave está en notar que $suma(i,j) = V[j] - V[i]$
- El arreglo V puede computarse acumulando en una sola iteración, en $O(n)$
- Luego cada pregunta podemos responderla en $O(1)$, comparado a $O(n)$ que costaba en la solución trivial.

Ejemplo (tabla aditiva)

```
// Cálculo de V:
V[0] = 0
for i = 1 to n
    V[i] = V[i-1] + v[i-1]
```

i	:	0	1	2	3	4	5	6	7
$v[i]$	:	2	6	2	-4	6	4	3	
$V[i]$	:	0	2	8	10	6	12	16	19

- $suma(2, 5) = V[5] - V[2] = 12 - 8 = 4$
- $suma(3, 7) = V[7] - V[3] = 19 - 10 = 9$
- $suma(5, 5) = V[5] - V[5] = 12 - 12 = 0$

Contenidos

Complicaciones para la tabla aditiva

La tabla aditiva es un recurso muy útil pero hay dos complicaciones comunes que no nos permiten utilizarla:

- Cuando se realizan modificaciones al arreglo v entre preguntas, habría que recalcular V cada vez, y ya no ganamos nada.
- Si en lugar de la suma queremos saber el mínimo o el máximo valor en el rango, la tabla no sirve (no podemos "restar").

La estructura de *Segment Tree* permite resolver ambos inconvenientes (y otros más que no hemos mencionado).

Contenidos

Estructura : Descripción

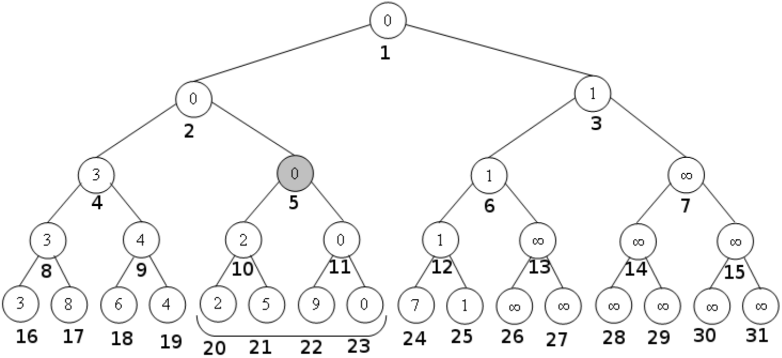
- Para trabajar con el segment tree asumiremos que n es potencia de 2. De no serlo, basta extender el arreglo v con a lo sumo n elementos adicionales para que sea potencia de 2.
- Utilizaremos un árbol binario completo de n hojas, codificado en un arreglo A de $2n$ elementos:
- A_1 guardará la raíz del árbol.
- Para cada i, los dos hijos del elemento A_i serán A_{2i} y A_{2i+1}.
- El padre de un elemento i será `i/2`, salvo en el caso de la raíz.

Estructura : Descripción (cont)

- Las hojas tendran índices en $[n, 2n)$.
- La idea será que cada nodo interno del árbol (guardado en un elemento del arreglo A) almacene el mínimo (o la suma, o la información que corresponda) entre sus dos hijos.
- Las hojas contendrán en todo momento los elementos del arreglo v.

Estructura : Dibujo

Este segment tree guarda mínimos (la misma idea funciona para máximos, para la suma, etc), para un arreglo inicial de 10 elementos:



Estructura : Inicializacion

- Llenamos $A_n, \cdots, A_{2n-1}$ con los elementos del arreglo v.
- Para calcular los nodos internos, hacemos simplemente:

$$A_i = \min(A_{2i}, A_{2i+1})$$

- Notar que se debe recorrer i en forma descendente para no utilizar valores aún no calculados.
- El proceso de inicialización toma tiempo $O(n)$, y la estructura utiliza $O(n)$ memoria.

Estructura : Modificaciones

- Para modificar el arreglo, utilizaremos nuevamente la fórmula:

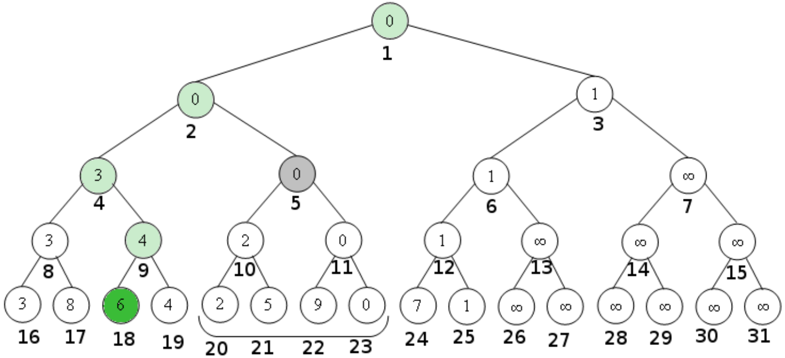
$$A_i = \min(A_{2i}, A_{2i+1})$$

- Notemos que si cambiamos el valor de v_i por x, debemos modificar A_{n+i} haciéndolo valer x, y recalcular los nodos internos del árbol...
- Pero sólo se ven afectados los ancestros de A_{n+i}.
- Luego es posible modificar un valor de v en tiempo $O(\lg n)$, recalculando sucesivamente los padres desde A_{n+i} hasta llegar a la raíz.

Estructura : Nodos que hay que modificar

```
A[n+i] = nuevoValor
x      = (n + i) / 2
mientras x >= 1
    A[x] = min(A[2*x], A[2*x+1])
    x    = x / 2
```

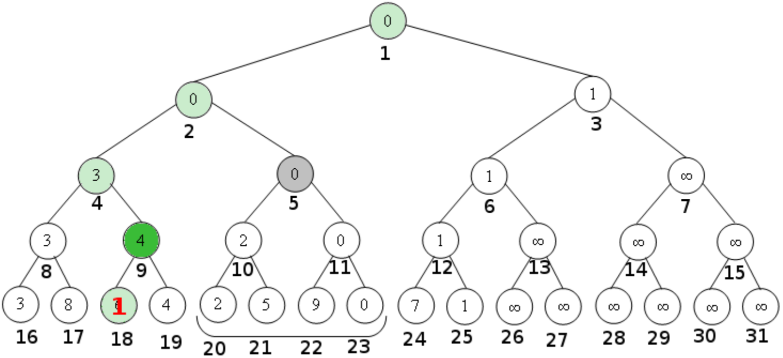
Si se modifica $v_2 = 1$:



Estructura : Nodos que hay que modificar

```
A[n+i] = nuevoValor
x      = (n + i) / 2
mientras x >= 1
    A[x] = min(A[2*x], A[2*x+1])
    x    = x / 2
```

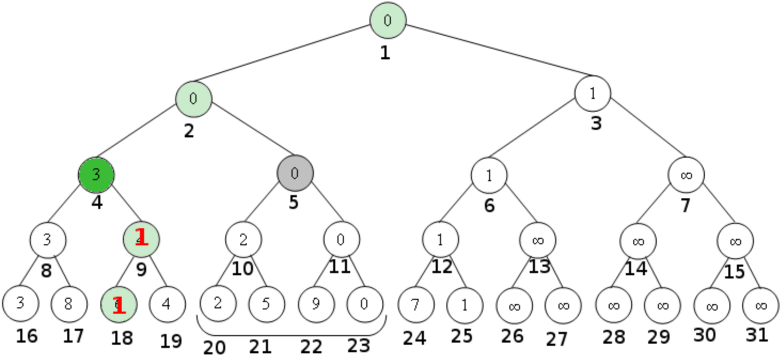
Si se modifica $v_2 = 1$:



Estructura : Nodos que hay que modificar

```
A[n+i] = nuevoValor
x      = (n + i) / 2
mientras x >= 1
    A[x] = min(A[2*x], A[2*x+1])
    x    = x / 2
```

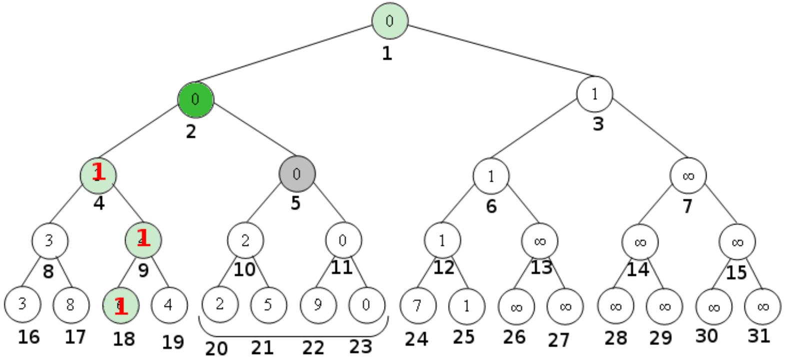
Si se modifica $v_2 = 1$:



Estructura : Nodos que hay que modificar

```
A[n+i] = nuevoValor
x      = (n + i) / 2
mientras x >= 1
    A[x] = min(A[2*x], A[2*x+1])
    x    = x / 2
```

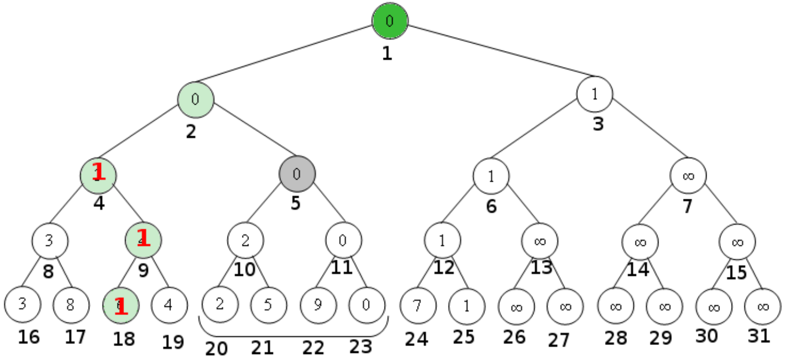
Si se modifica $v_2 = 1$:



Estructura : Nodos que hay que modificar

```
A[n+i] = nuevoValor
x      = (n + i) / 2
mientras x >= 1
    A[x] = min(A[2*x], A[2*x+1])
    x    = x / 2
```

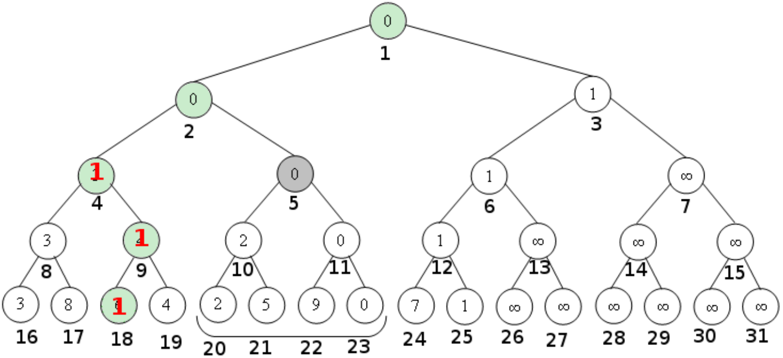
Si se modifica $v_2 = 1$:



Estructura : Nodos que hay que modificar

```
A[n+i] = nuevoValor
x      = (n + i) / 2
mientras x >= 1
    A[x] = min(A[2*x], A[2*x+1])
    x    = x / 2
```

Si se modifica $v_2 = 1$:



Estructura : Queries

Para encontrar el mínimo en un rango $[i, j)$, exploramos los nodos bajando desde la raíz recursivamente:

- Si el intervalo del nodo actual está totalmente contenido en el rango que nos interesa, devolvemos directamente el valor guardado en el nodo.
- Si el intervalo del nodo actual es disjunto con el rango que nos interesa, devolvemos $+\infty$ (o cero si trabajamos con la suma, etc).
- En otro caso (hay parte adentro y parte afuera), propagamos la pregunta a los dos hijos del nodo.

Estructura : Queries (cont)

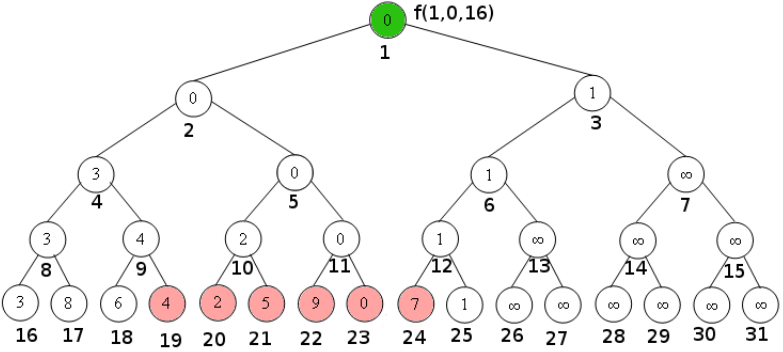
Tomaremos $f(k, l, r)$ como "El resultado de consultar por el rango $[i, j)$ desde el nodo número k, cuyo intervalo es el $[l, r)$ "

$$f(k, l, r) = \begin{cases} A_k & \text{si } i \leq l < r \leq j \quad \text{(contenido)} \\ +\infty & \text{si } r \leq i \text{ o } l \geq j \quad \text{(disjunto)} \\ \min(f(2k, l, \frac{l+r}{2}), f(2k+1, \frac{l+r}{2}, r)) & \text{sino} \end{cases}$$

- La respuesta vendrá dada por $f(1, 0, n)$ (notar que la función recursiva usa los valores i y j)
- La complejidad temporal de un query es $O(\lg n)$, ya que se exploran a lo más 4 nodos por nivel.

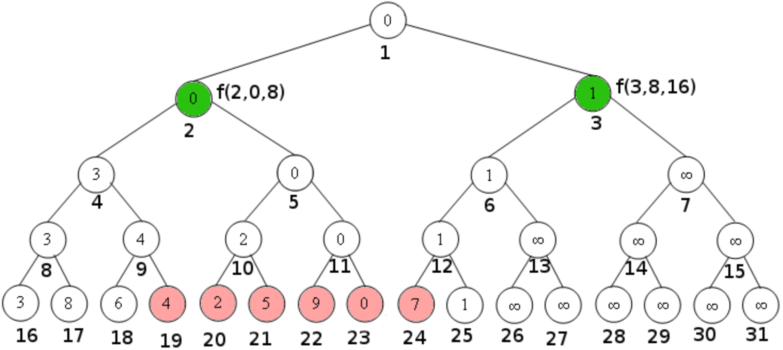
Estructura : Nodos consultados en una query

Consideremos la consulta que busca el mínimo en el rango $[3, 9)$:



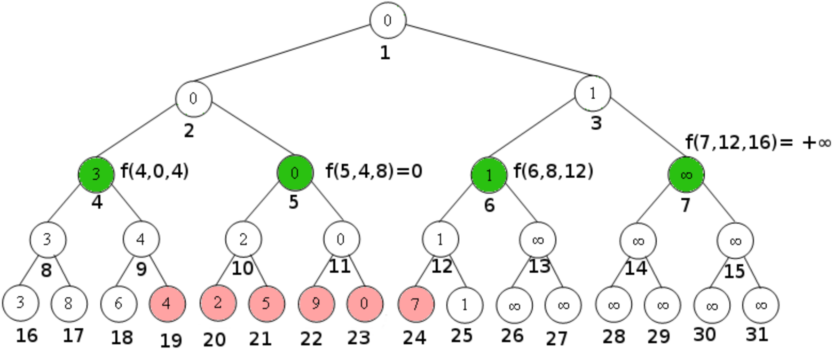
Estructura : Nodos consultados en una query

Consideremos la consulta que busca el mínimo en el rango $[3, 9)$:



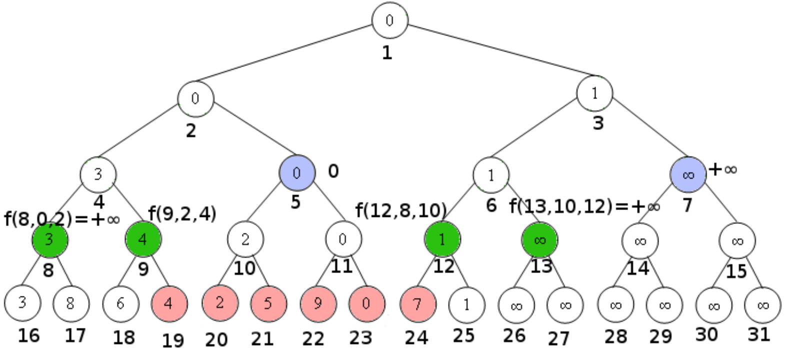
Estructura : Nodos consultados en una query

Consideremos la consulta que busca el mínimo en el rango $[3, 9)$:



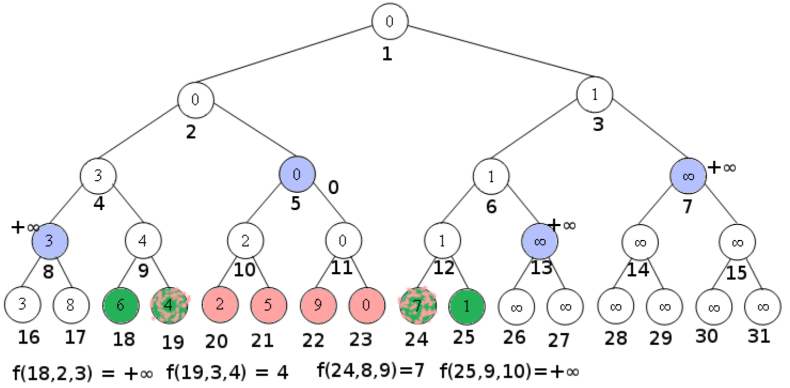
Estructura : Nodos consultados en una query

Consideremos la consulta que busca el mínimo en el rango $[3,9)$:



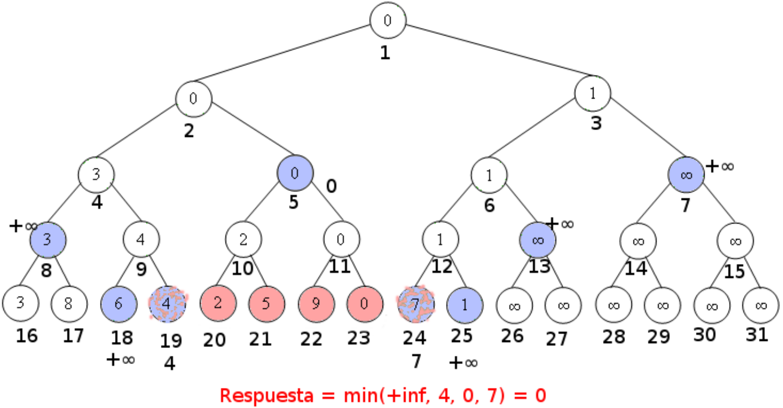
Estructura : Nodos consultados en una query

Consideremos la consulta que busca el mínimo en el rango $[3, 9)$:



Estructura : Nodos consultados en una query

Consideremos la consulta que busca el mínimo en el rango $[3, 9)$:



Contenidos

Problema sumo (Certamen de selección 2010):

- En el problema se dan $L \leq 100000$ luchadores de sumo, cada uno con dos atributos altura y peso entre 0 y 1000000 (no hay dos luchadores iguales).
- Un luchador domina a otro si le gana estrictamente en uno de los atributos, y no es peor en el otro ($\geq$ y $>$).
- Se pide dar para cada luchador, la cantidad de luchadores a los que domina.

Problema sumo (Certamen de selección 2010):

- Observación: Si ordenamos a los luchadores por altura, desempatando por peso, los dominados quedan siempre a la izquierda.
- Más precisamente, el luchador i dominará exactamente a los $j < i$ tales $peso(j) \leq peso(i)$.
- Es decir, que si vamos “agregando” los pesos de los luchadores en orden, la respuesta es en cada momento la cantidad de elementos agregados que son menores o iguales que $peso(i)$.

Problema sumo (Certamen de selección 2010):

- Si en un arreglo v guardamos en el lugar x la cantidad de luchadores agregados actualmente con peso x...

Problema sumo (Certamen de selección 2010):

- Si en un arreglo v guardamos en el lugar x la cantidad de luchadores agregados actualmente con peso x...
- La cantidad que queremos en todo momento es la suma de los valores de v entre 0 y $peso(i)$
- Como el arreglo se va modificando (se agregan luchadores), no sirve una tablita aditiva.
- Pero podemos utilizar un Segment Tree para obtener las sumas deseadas y actualizar en $O(\lg n)$

Contenidos

Tarea

- http://www.spoj.pl/problems/KGSS/