Segment Tree

Enzo Vivallo

enzovivallo@estudiante.uc.cl

Motivación

Se te entrega una lista de enteros a de tamaño n ($1 \leq n \leq 10^5$), y cada elemento cumple $1 \leq a_i \leq 10^5$. Nos piden responder q ($1 \leq q \leq 10^5$) consultas del tipo:

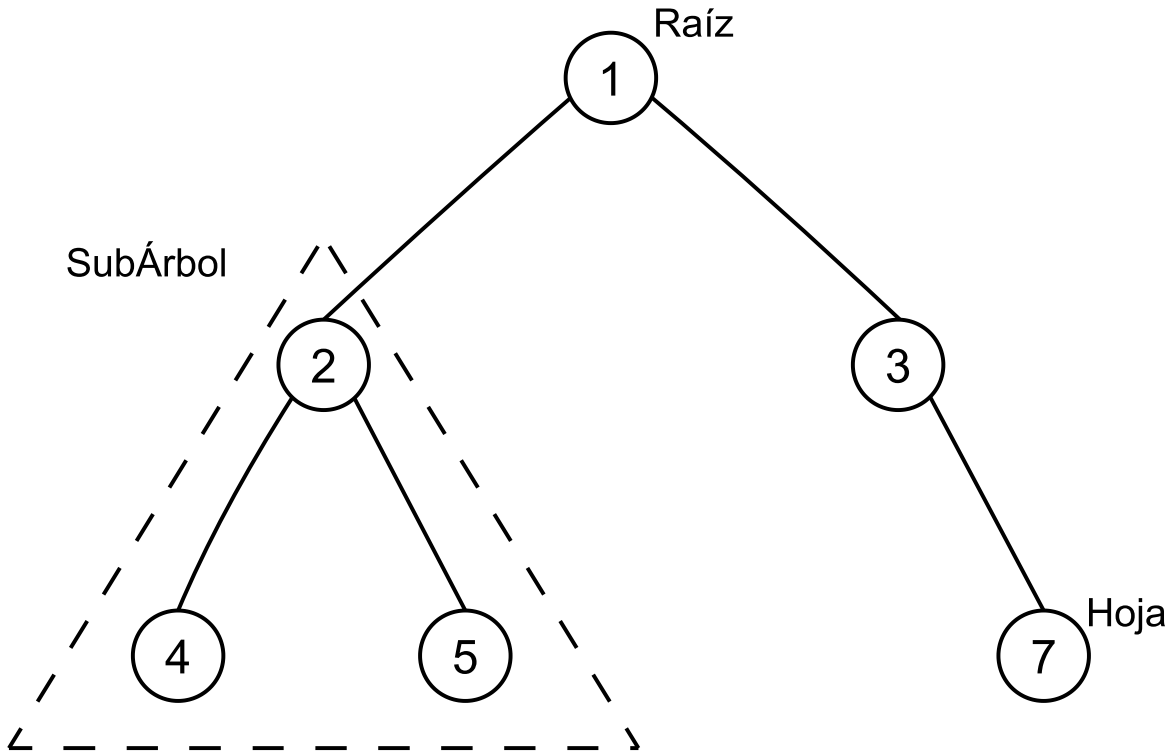
- ¿Cuál es el mínimo en el rango $[left, right]$?
- Cambia el valor del índice i por x

Segment Tree

- Estructura de datos que permite responder consultas de rango y actualizar elementos.
- Su construcción es $O(n)$ y cada consulta o actualización es $O(\log n)$.
- Se usa un *binary tree* con cada nodo representando un rango.

Binary Tree

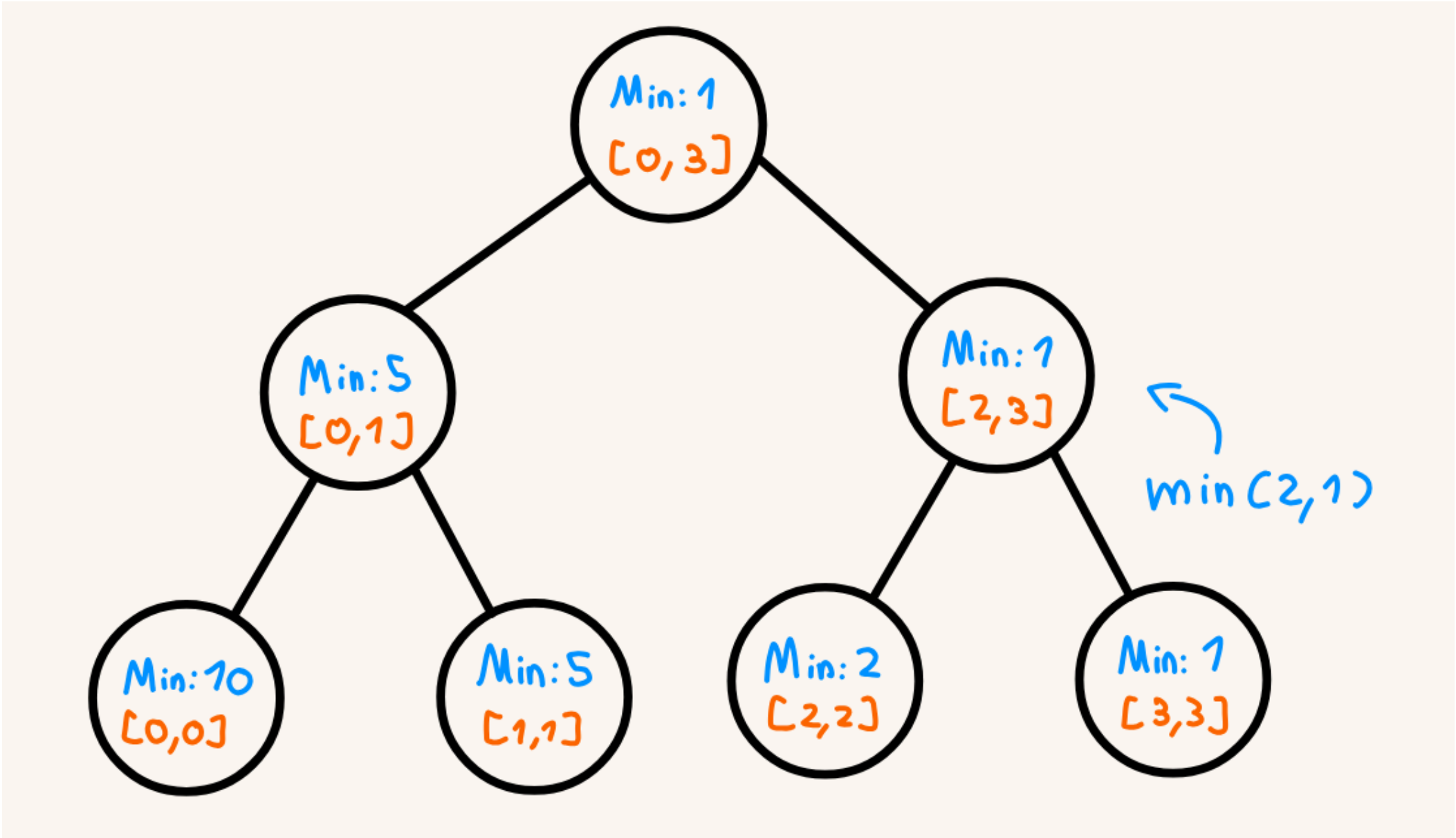
Un **Binary Tree** es un árbol donde cada nodo tiene a lo más 2 hijos y hay una raíz que es la única que no tiene un padre.



Armando un Segment Tree

Creamos un arbol binario donde cada nodo representa un rango de la lista de números. El nodo raíz representa el rango completo, y los nodos hijos representan un elemento de la lista.

Armando un Segment Tree



Armando un Segment Tree

Notamos que la operación debe cumplir que el padre de un nodo debe ser el resultado de aplicar la operación a los valores de sus hijos, a la función que hace esta operación solemos llamarla **merge**.

Para que la operación funcione en un **Segment Tree** debe poder calcularse consistentemente en un rango, es decir:

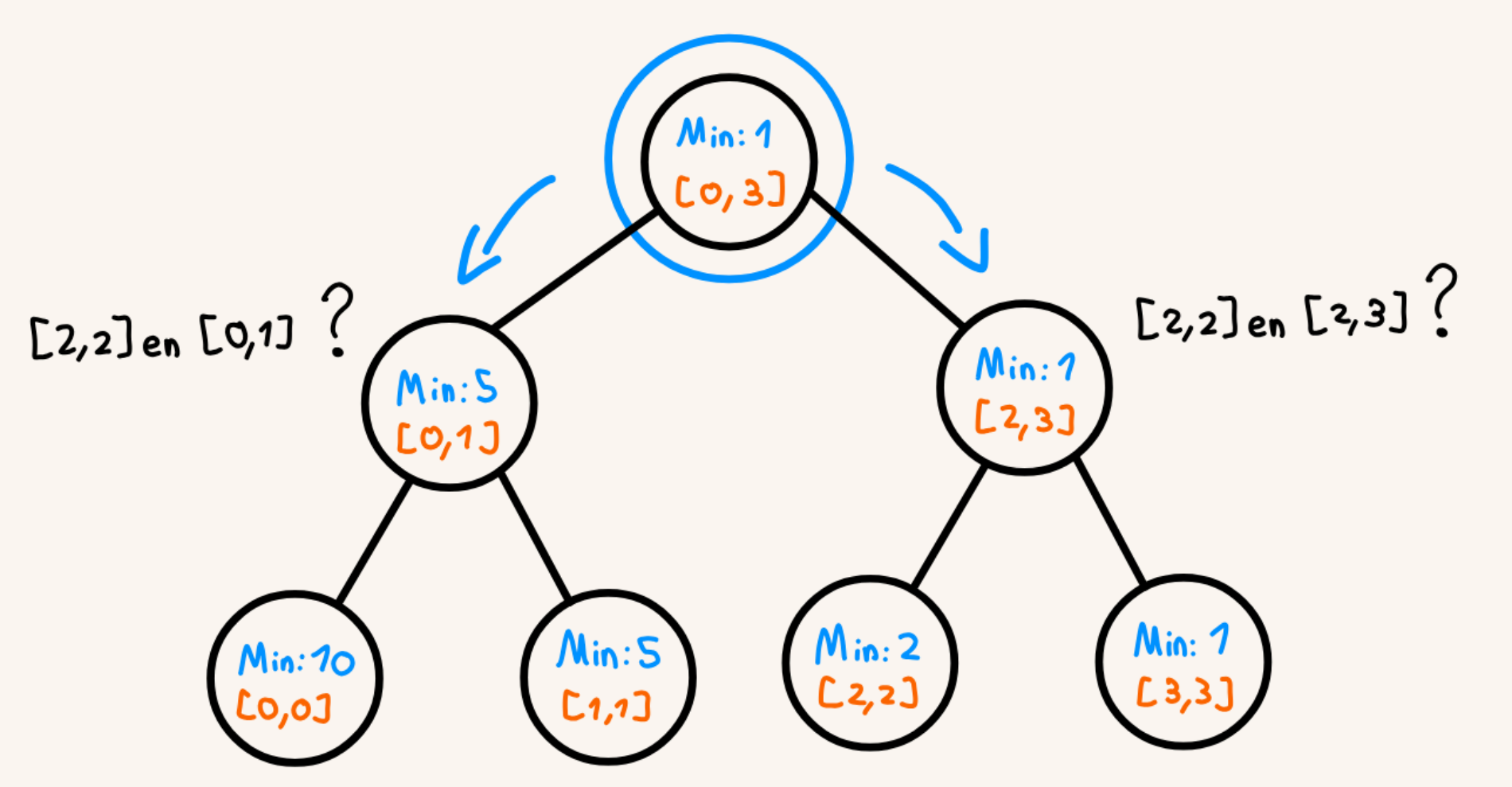
Siendo A la lista y $A[a \dots b]f$ la sub-lista del rango $[a, b]$ de la lista, y x un índice perteneciente a ese rango:

$$f(A[left \dots right]) = f(f(A[left \dots x]), f(A[x+1 \dots right]))$$

💡 Lo anterior no es asociatividad, pero si una operación es asociativa $a \circ b \circ c = (a \circ b) \circ c = a \circ (b \circ c)$, entonces sí cumple lo anterior.

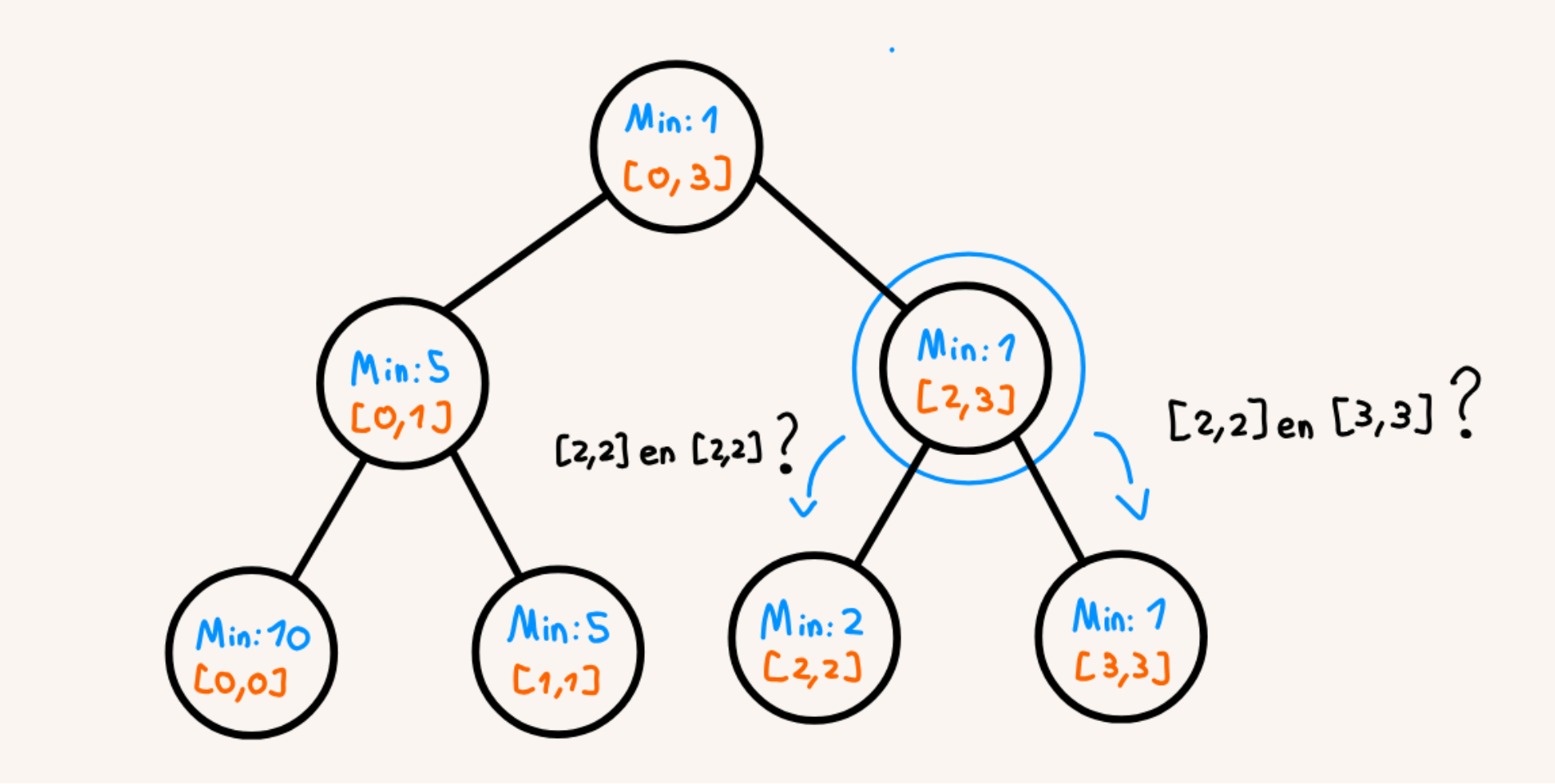
Consultas

Para $a = [10, 5, 2, 1]$ y la consulta de mínimo en el rango $[1, 3]$:

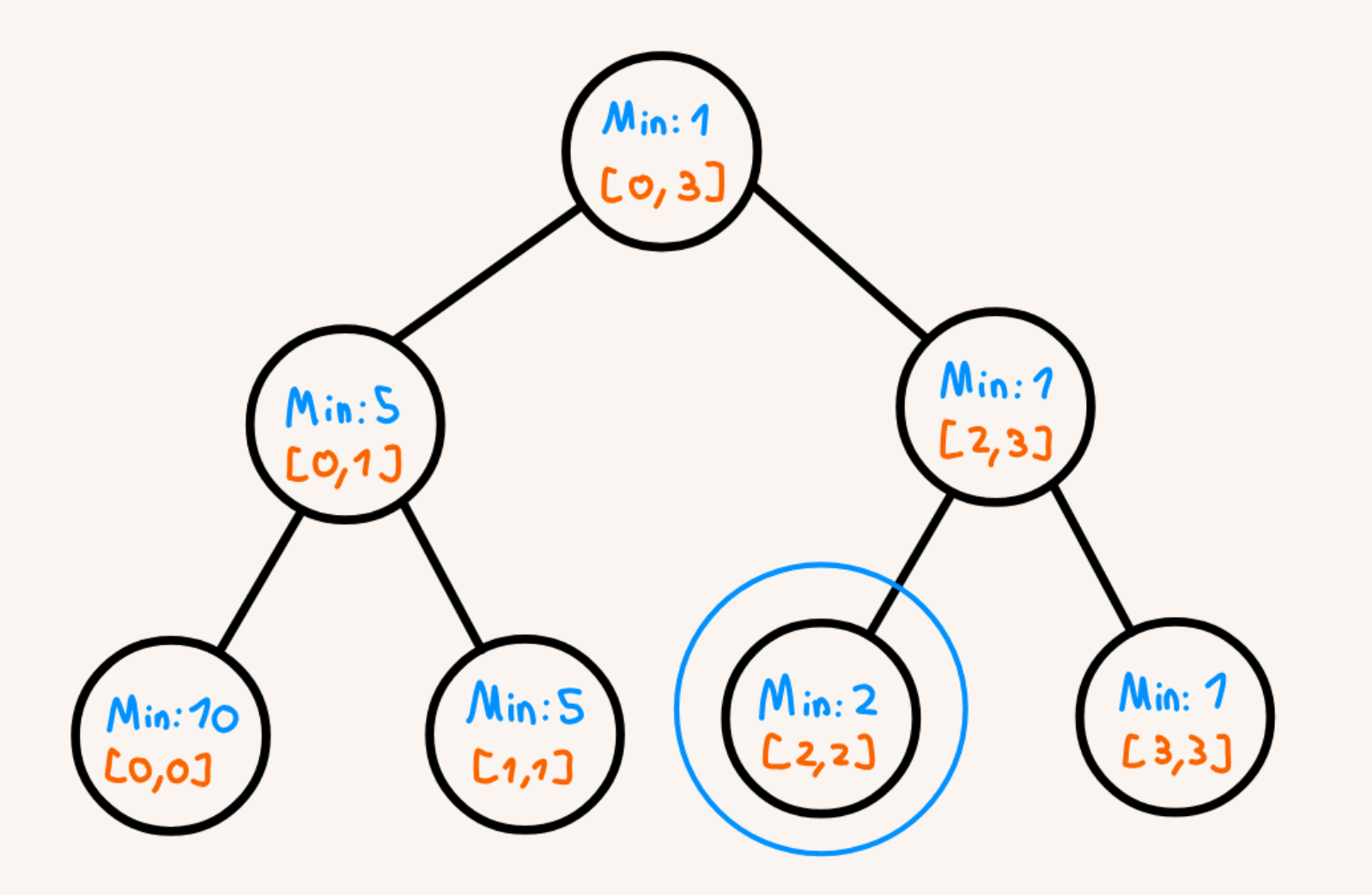


Consultas

Para $a = [10, 5, 2, 1]$ y la consulta de mínimo en el rango $[1, 3]$:

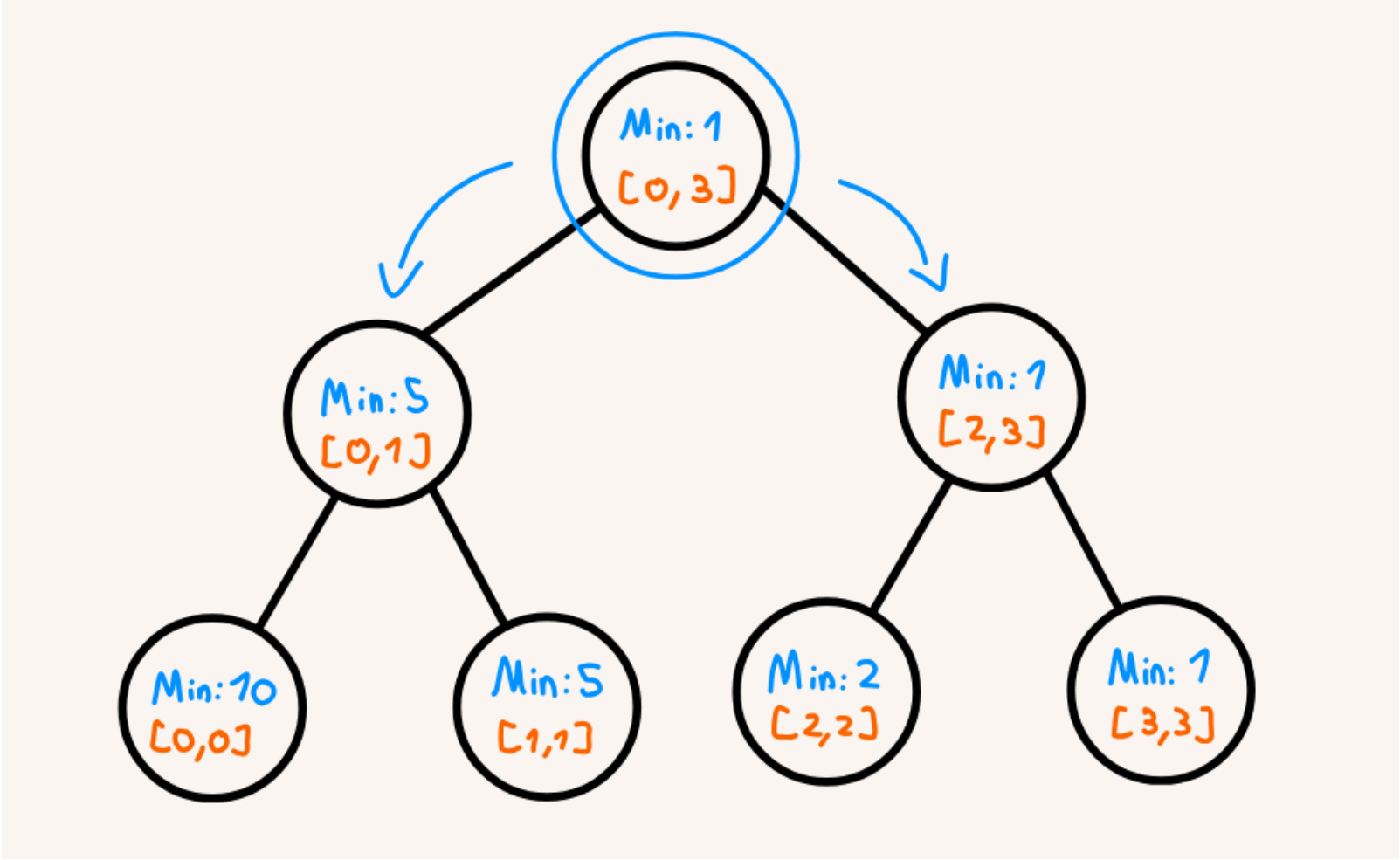


Entonces la consulta de mínimo en el rango $[1, 3]$ es igual a 2:



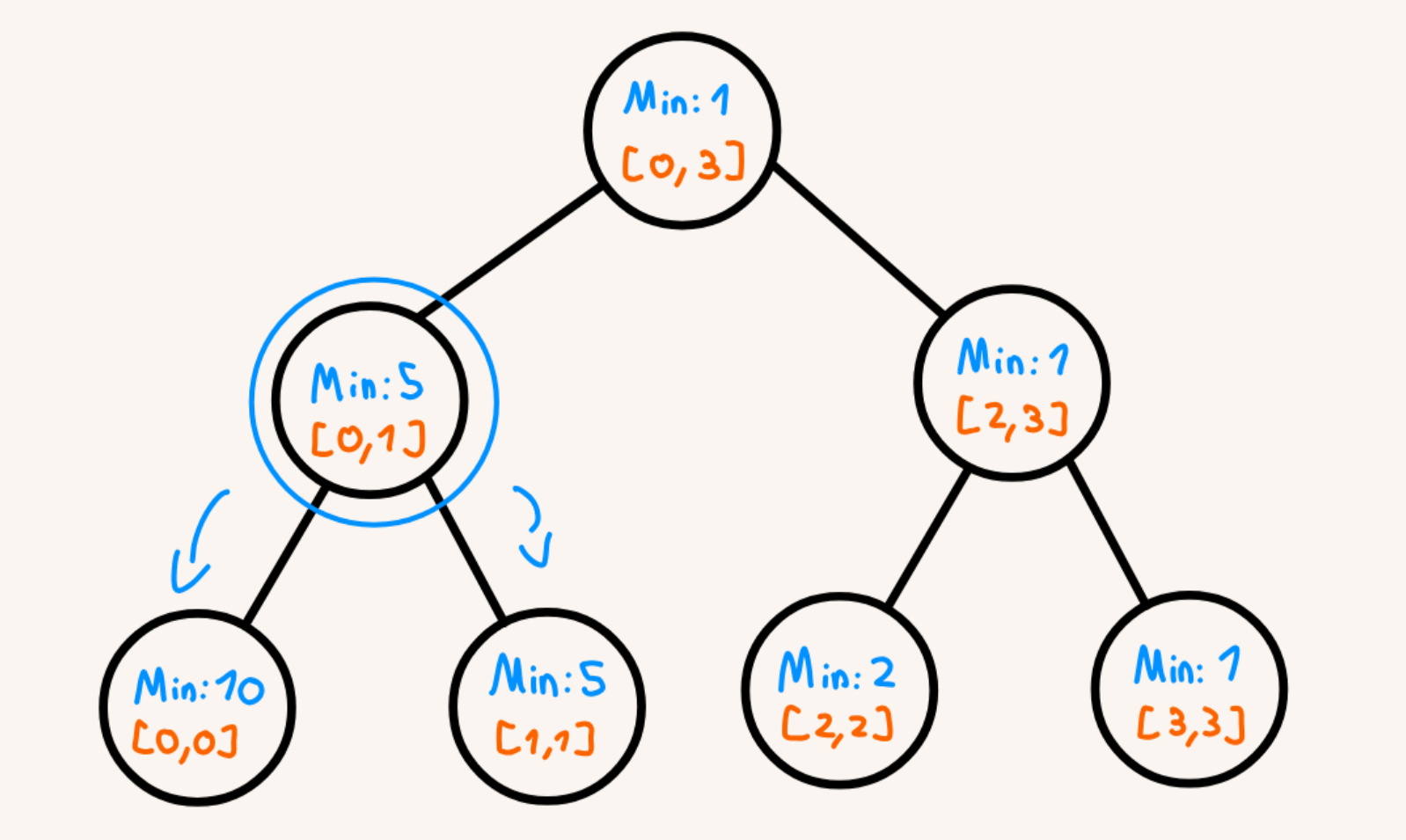
Actualización

Para $a = [10, 5, 2, 1]$ y actualización del índice 1 a 0:



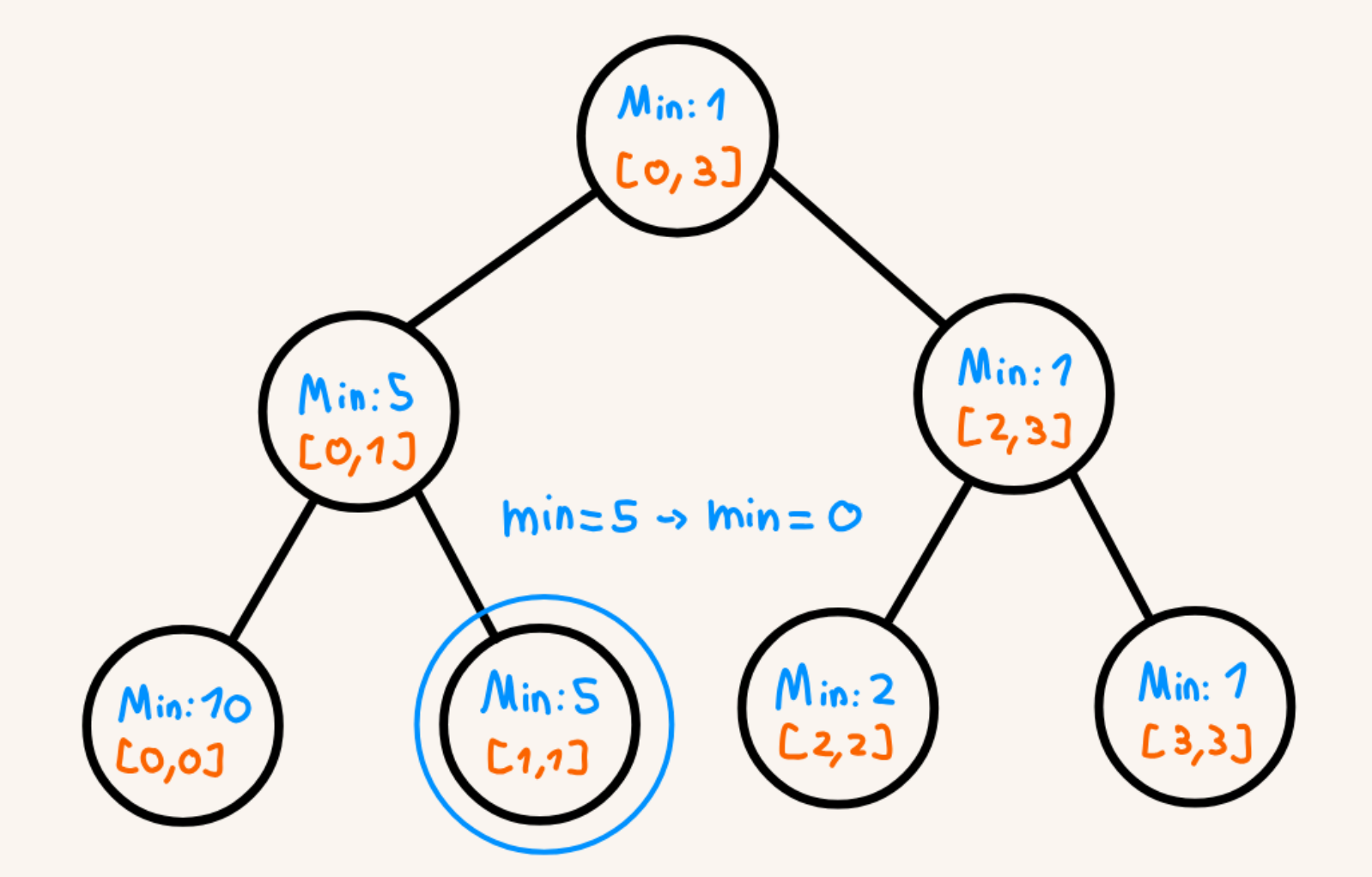
Actualización

Para $a = [10, 5, 2, 1]$ y actualización del índice 1 a 0:



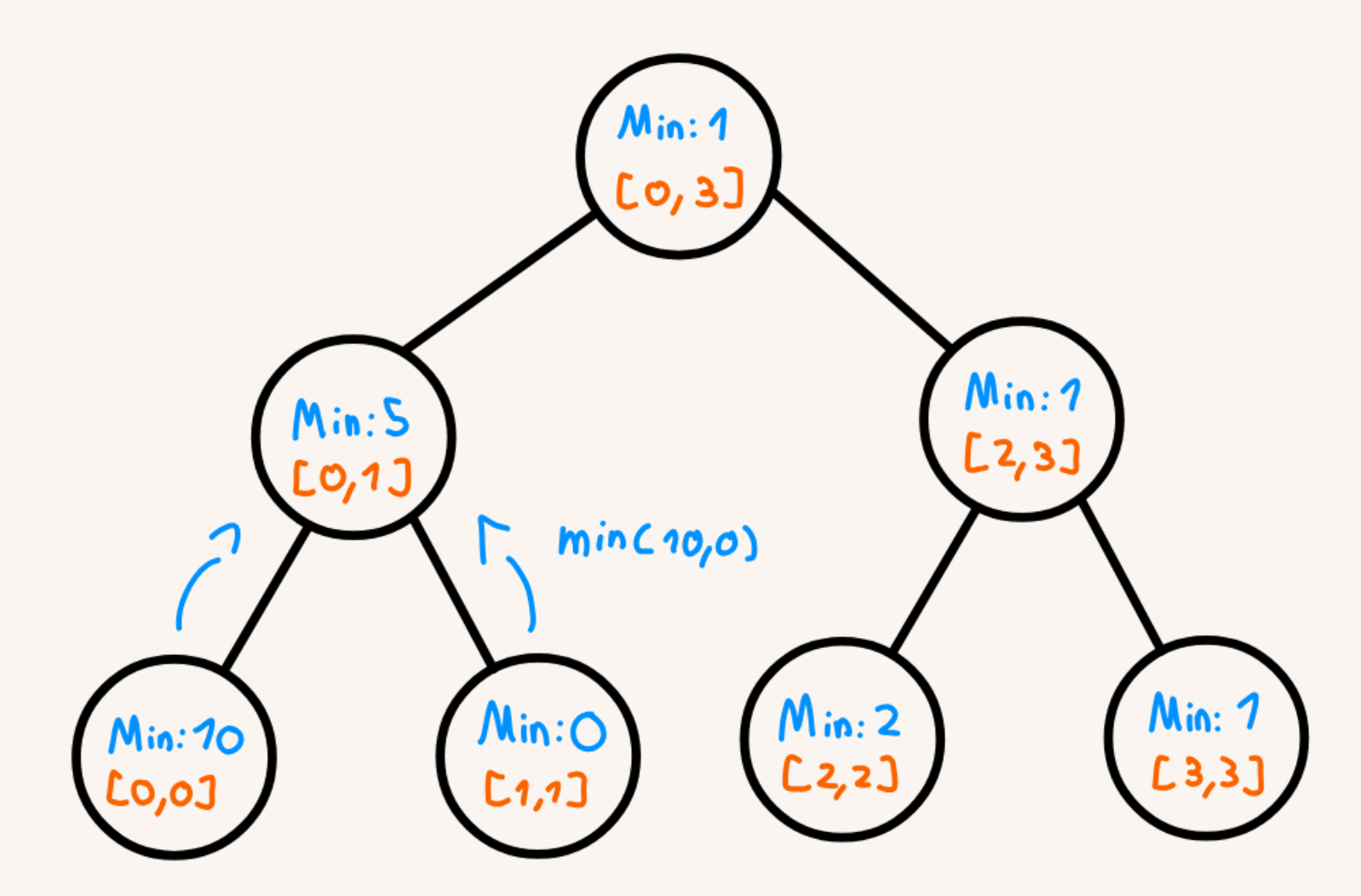
Actualización

Para $a = [10, 5, 2, 1]$ y actualización del índice 1 a 0:



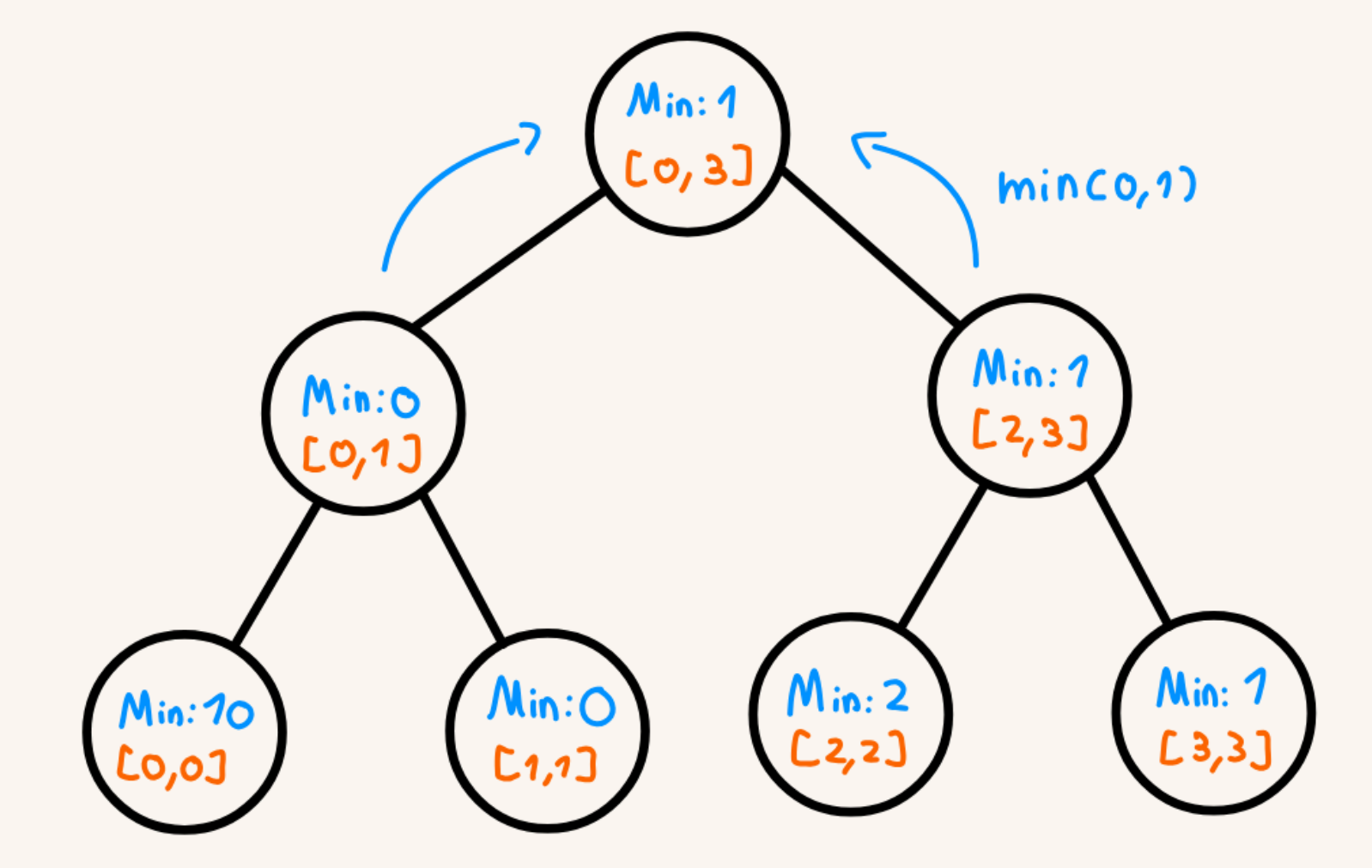
Actualización

Para $a = [10, 5, 2, 1]$ y actualización del índice 1 a 0:



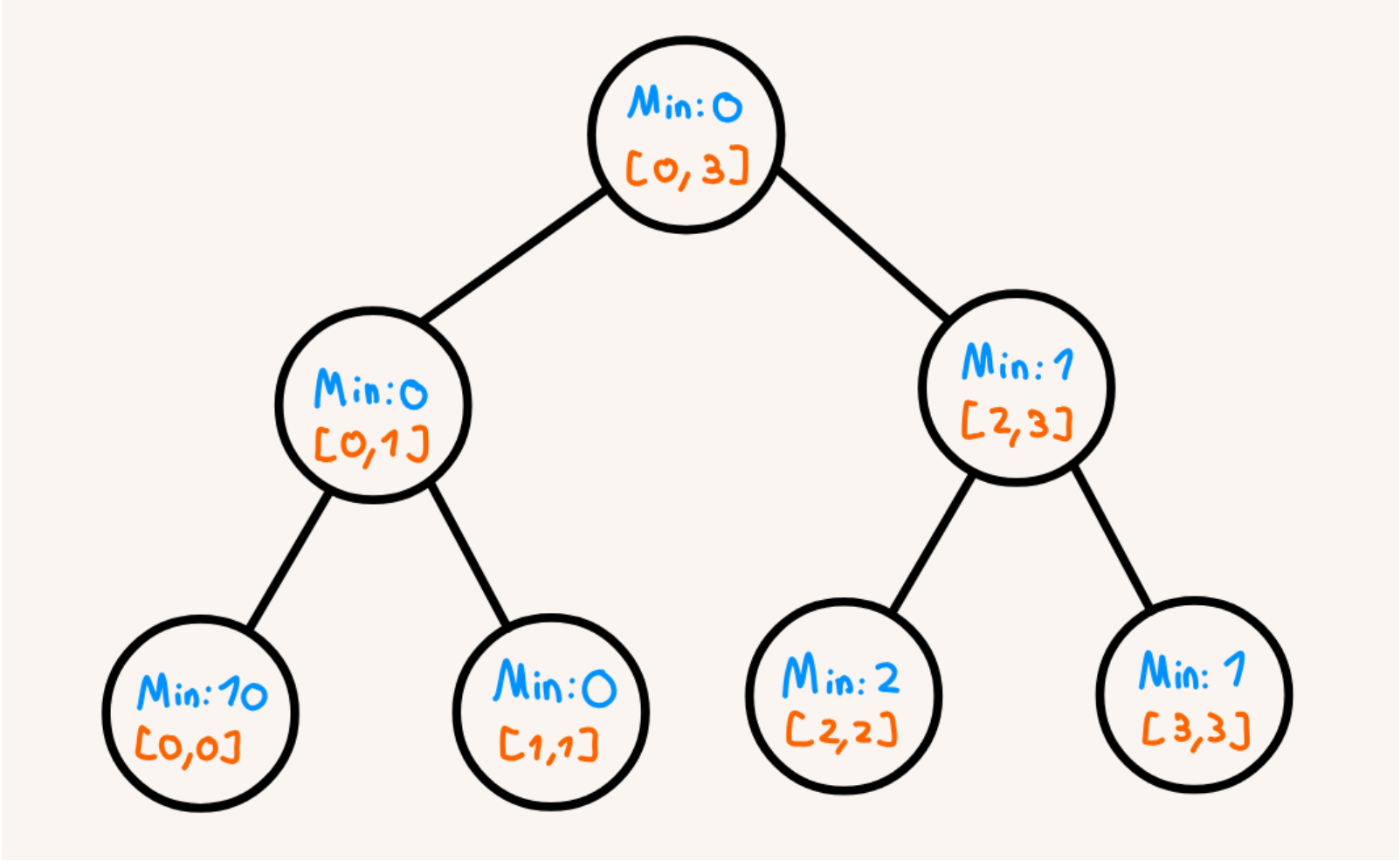
Actualización

Para $a = [10, 5, 2, 1]$ y actualización del índice 1 a 0:



Actualización

Para $a = [10, 5, 2, 1]$ y actualización del índice 1 a 0:



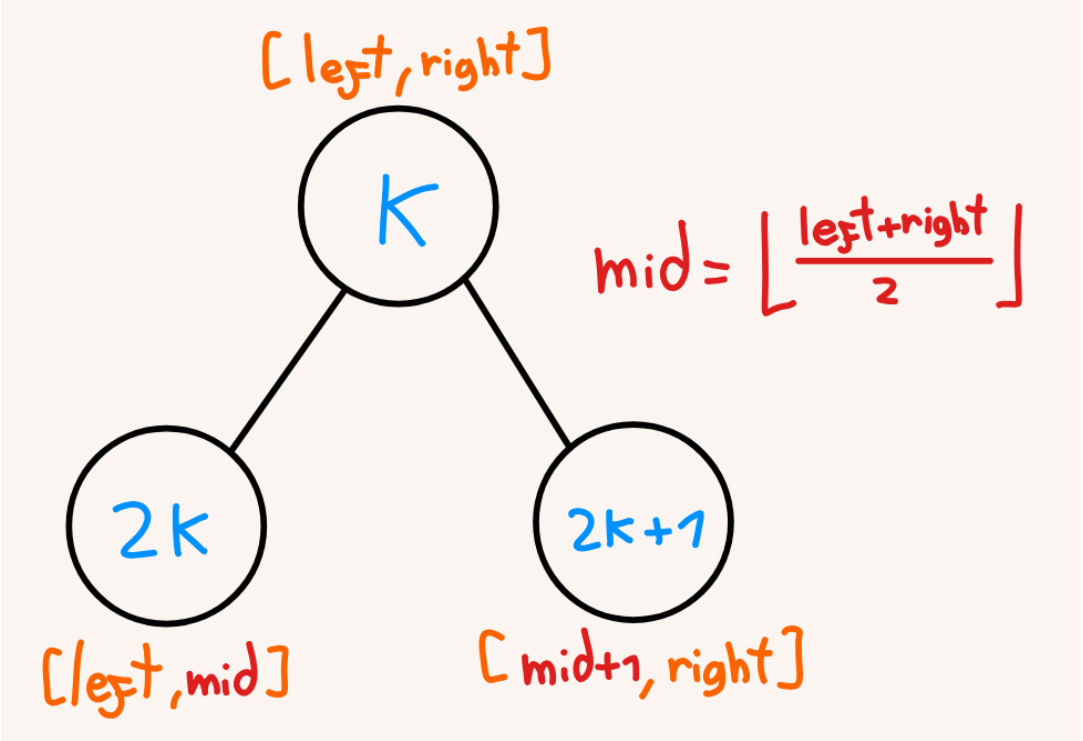
Implementación

- Creamos un arreglo de tamaño $4n$ para almacenar el árbol.
- La raíz corresponde al nodo 1 (el 0 no se usa) y rango $[0, n-1]$. Respecto a la raíz es posible crear el resto del árbol.

Implementación

El hijo izquierdo de un nodo con índice k es $2k$ y el derecho es $2k+1$ y el padre de k es $\left\lfloor \frac{k}{2} \right\rfloor$.

Cada rango $[left, right]$ en un nodo se separa en el hijo izquierdo $[left, mid]$ y el hijo derecho $[mid+1, right]$, donde $mid = \left\lfloor \frac{left+right}{2} \right\rfloor$.

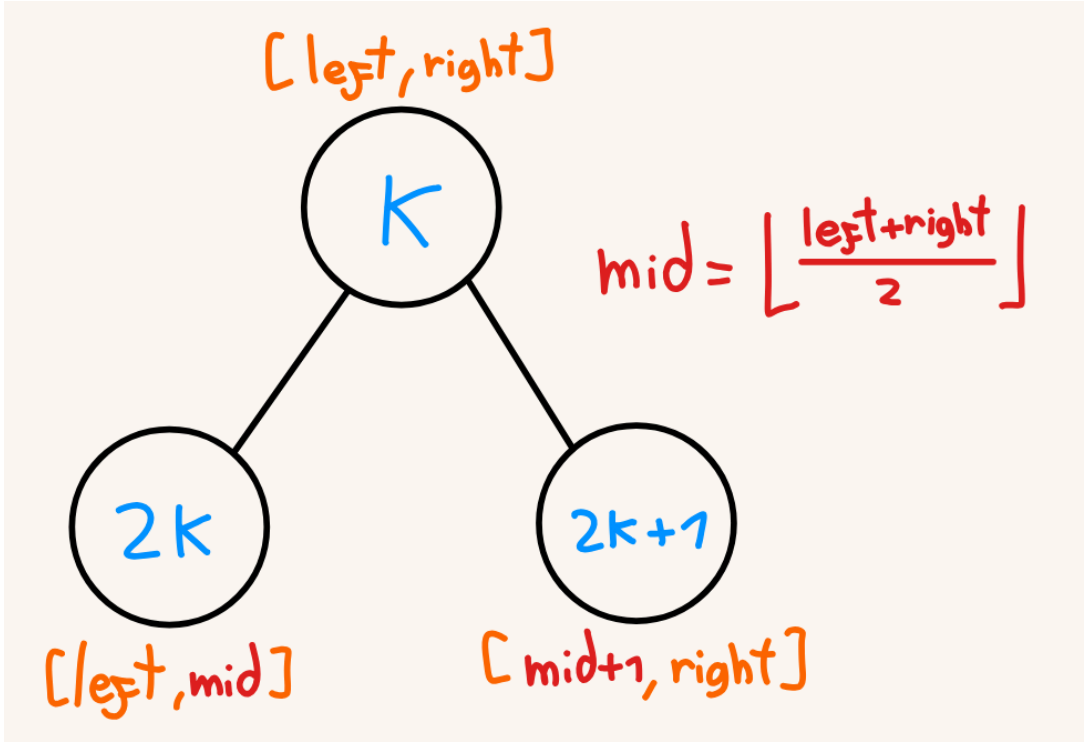


Implementación

Podemos crear el árbol recursivamente desde la raíz hacía abajo, hasta llegar a los nodos hoja que representan los elementos de la lista.

El caso base es si $left = right$, el nodo representa un elemento de la lista y se asigna el valor.

Luego por cada par de nodos hijos, se asigna el valor del nodo padre como la operación deseada sobre los valores de los hijos.



Tamaño del árbol

Si n es potencia de 2, el árbol binario es completo y tiene $2n - 1$ nodos en total.

Si n no es potencia de 2, el árbol binario no es completo, este podría tener hasta $2m - 1$ nodos, donde m es la potencia de 2 más cercana y mayor a n. Es fácil ver que para n la siguiente potencia de 2 puede ser hasta $2n$, por lo que el árbol puede tener hasta $4n - 1$ nodos.

$2 \cdot f(n) - 1 \leq 4n - 1$ con $f(n)$ la siguiente potencia de 2 de n.

💡 Usamos $4n$ porque consideramos la raíz como nodo 1 y el nodo 0 no se usa (porque se usa un nodo de más).

Complejidad

- Como dividimos el rango de a 2 en 2, el árbol binario tiene una altura de $\lfloor \log n \rfloor$.
- En la construcción del árbol creamos y recorremos hasta $4n$ nodos, por lo que la complejidad es $O(n)$.
- En las consultas si los 2 hijos de un nodo contienen un rango simplemente usamos el valor del nodo, en el caso contrario, hacemos una consulta recursiva en el hijo que contiene el rango. Entonces como dividimos el rango desde la raíz, por nivel solo podemos tener hasta 2 nodos visitados, entonces a los más pasariamos por $2 \log n$ nodos, por lo que la complejidad de una consulta es $O(\log n)$.
- En las actualizaciones aplica lo mismo, entonces la complejidad es $O(\log n)$.

Implementaciones

La implementación recursiva con un arreglo de tamaño $4n$ es la más clásica y es muy flexible, también existe la implementación iterativa que es más eficiente en memoria y en constante de tiempo pero es más díficil de modificar, se logra tener un arreglo de tamaño $2n$. Suele servir cuando nuestra solución entera es $O(n(\log n)^2)$ o más, ya que la constante de tiempo es menor.

🔗 Link de la implementación recursiva: miniurl.cl/mytaxb

🔗 Link de la implementación iterativa: miniurl.cl/61wmjr

Cómo usarlo

```
int merge(int a, int b) {
    return min(a, b);
}

int main() {
    vector<int> values = {10, 5, 2, 1, 5, 6};

    SegmentTree<int, merge> segmentTree(values);

    int result = segmentTree.query(1, 4); // Mínimo en el rango [1, 4]

    segmentTree.update(1, 0); // Cambia el valor del índice 1 a 0
}
```

Among the Tall

Te dan un arreglo A de N ($1 \leq N \leq 10^5$) personas, donde $A[i]$ es la altura de la persona en la posición i.

Se quiere calcular el promedio de altura de todas las personas para cada rango [1, i] que tengan altura mayor o igual a $A[i]$.

Debes imprimir ese promedio para cada posición, usando módulo 10^9+7.

🔗 Link del problema: codeforces.com/gym/105876/problem/A
🔗 Link del código solución: miniurl.cl/anrfcs

Pairs of Numbers

Tienes un arreglo de n ($1 \leq n \leq 3 * 10^5$) enteros positivos. Debes encontrar todos los segmentos $[l, r]$ del arreglo que cumplan estas condiciones:

- Existe un índice j dentro del rango $[l, r]$ tal que todos los elementos del segmento son divisibles por $a[j]$.
- Entre todos los segmentos que cumplen la condición anterior, queremos los que tengan el mayor largo posible ($r - l$).

🔗 Link del problema: codeforces.com/contest/359/problem/D
🔗 Link del código solución: miniurl.cl/e6st75

Mancala 2

Hay N ($1 \leq N \leq 2 * 10^5$) cajas numeradas del 0 al $N-1$. Inicialmente, la caja i contiene $A[i]$ pelotas y se entrega un arreglo B de M ($1 \leq M \leq 2 * 10^5$) enteros, donde cada $B[i]$ es el índice de una caja.

Takahashi realizará las siguientes operaciones para $i = 1, 2, \ldots, M$, en orden:

- Primero, define una variable C igual a 0.
- Luego, saca todas las pelotas de la caja $B[i]$ y las sostiene en la mano.
- Mientras tenga al menos una pelota en la mano, repite el siguiente proceso:
  - Aumenta el valor de C en 1.
  - Coloca una pelota en la caja $(B[i] + C) \mod N$.
- Al finalizar todas las operaciones, determina cuántas pelotas hay en cada caja.

Mancala 2

- Link del problema: atcoder.jp/contests/abc340/tasks/abc340_e
- Link del código solución: miniurl.cl/2kxq44

KQUERY

Dada una secuencia a de n $(1 \leq n \leq 30000)$ números en el rango $(1 \leq a[i] \leq 10^9)$ y q $(1 \leq q \leq 200000)$ consultas, cada consulta entrega un rango (i, j) y un valor k $(1 \leq k \leq 10^9)$. Tu tarea es contar cuántos elementos en el subarreglo $a[i]$ hasta $a[j]$ son mayores que k.

🔗 Link del problema: spoj.com/problems/KQUERY

🔗 Link del código solución: miniurl.cl/okyui2