TALLER PROGCOMP: TRACK EDD

SEGMENT TREE LAZY

Gabriel Carmona Tabja

Universidad Técnica Federico Santa María,
Università di Pisa

May 6, 2024

Part I

Range Query con Update en Rango

RMQ - Range Sum Query con Update en Rango

Contextualización

Tenemos un arreglo de tamaño N, este arreglo posee números. Dentro de este programa existen dos tipos de consultas:

1. Determinar la suma en el rango i y j, donde $i \leq j$
2. Aumentar los valores entre las posiciones i y j por k, donde $i \leq j$

Solución con SegmentTree

- Determinar la suma en el rango: $O(\log n)$
- Hacer un update por cada posición entre el rango: $O(n \log n)$

Lazy Propagation

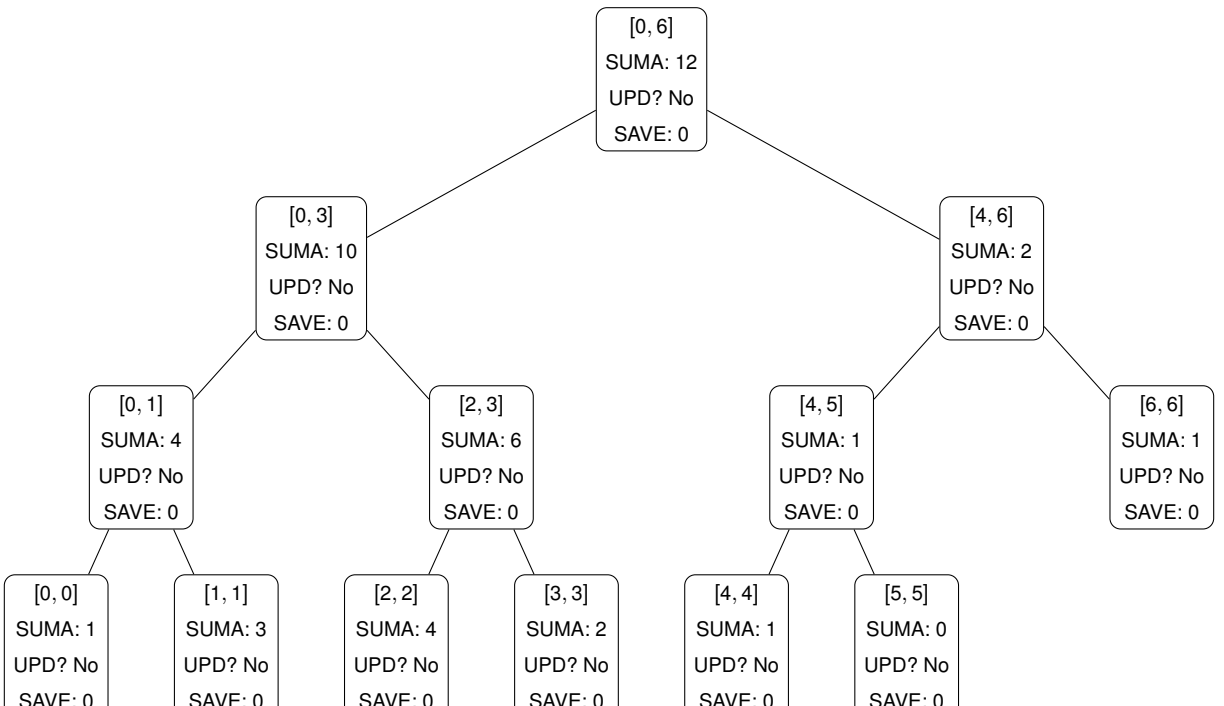
Idea

Un algoritmo *Lazy* es un técnica que dice:

Si no lo necesito ahora, lo recordaré para hacerlo después.

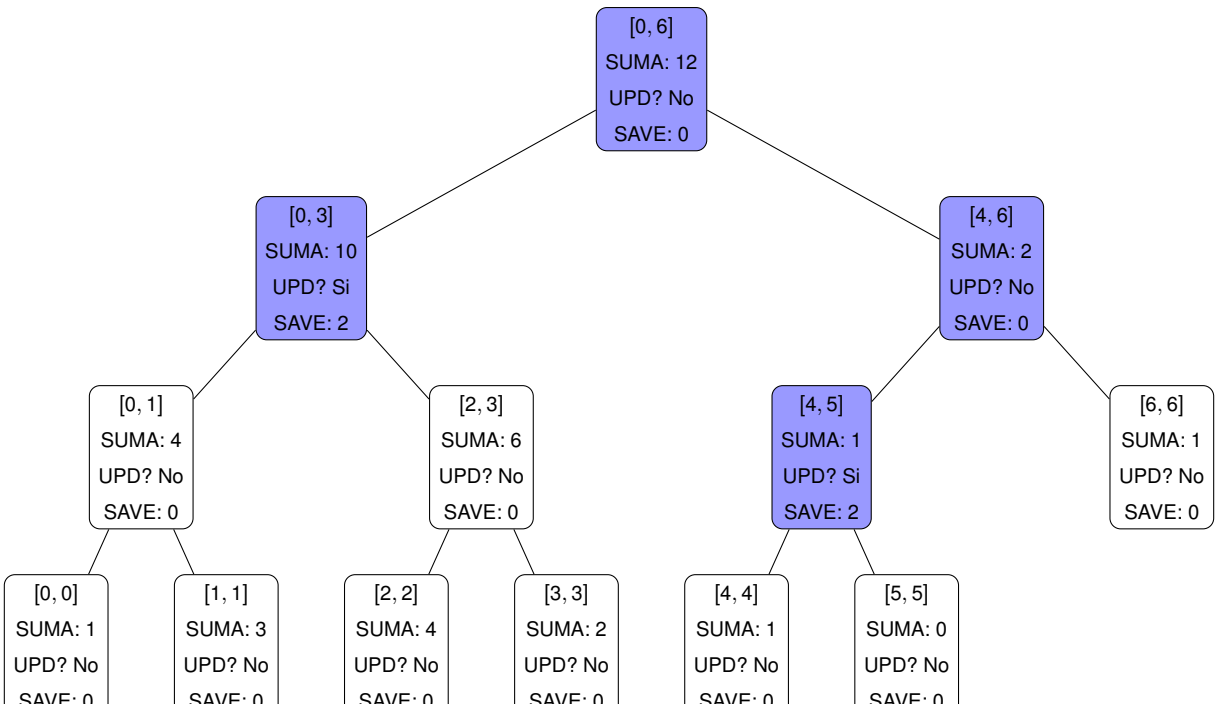
Segment Tree con Lazy Progation

Digamos el siguiente arreglo, $A = \{1, 3, 4, 2, 1, 0, 1\}$



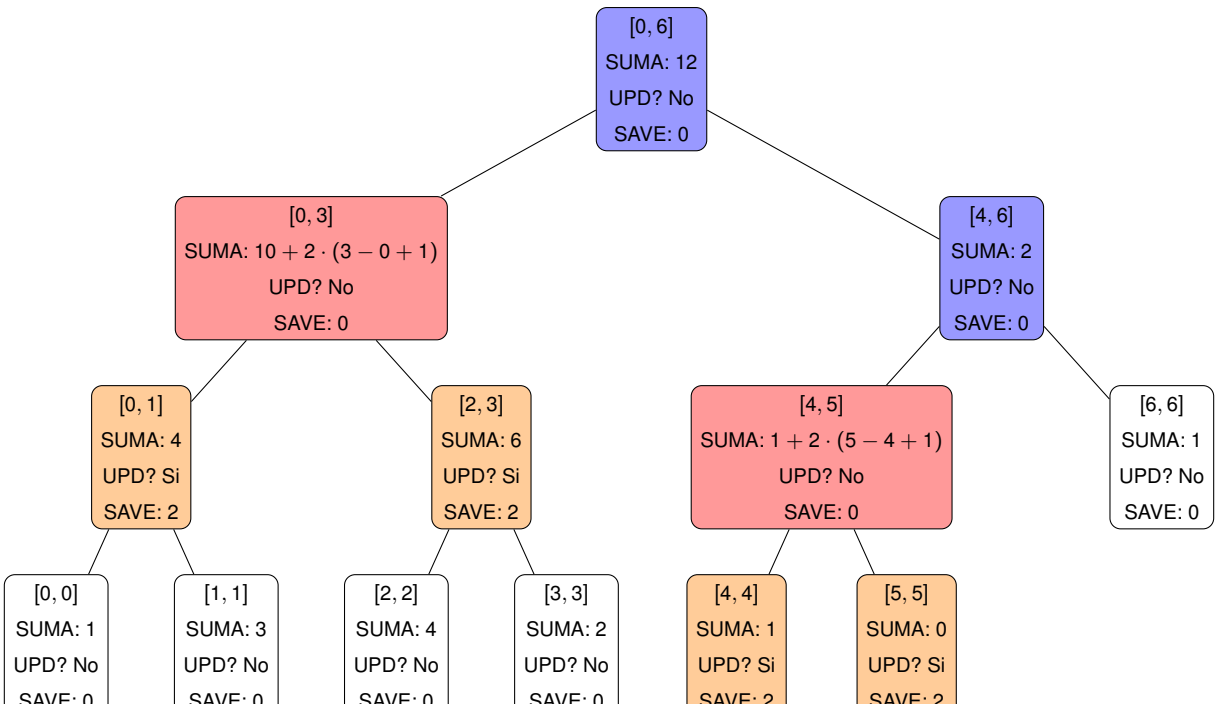
SEGMENT TREE CON LAZY PROGATION

A cada valor del rango 0 a 5 se aumentará en 2.



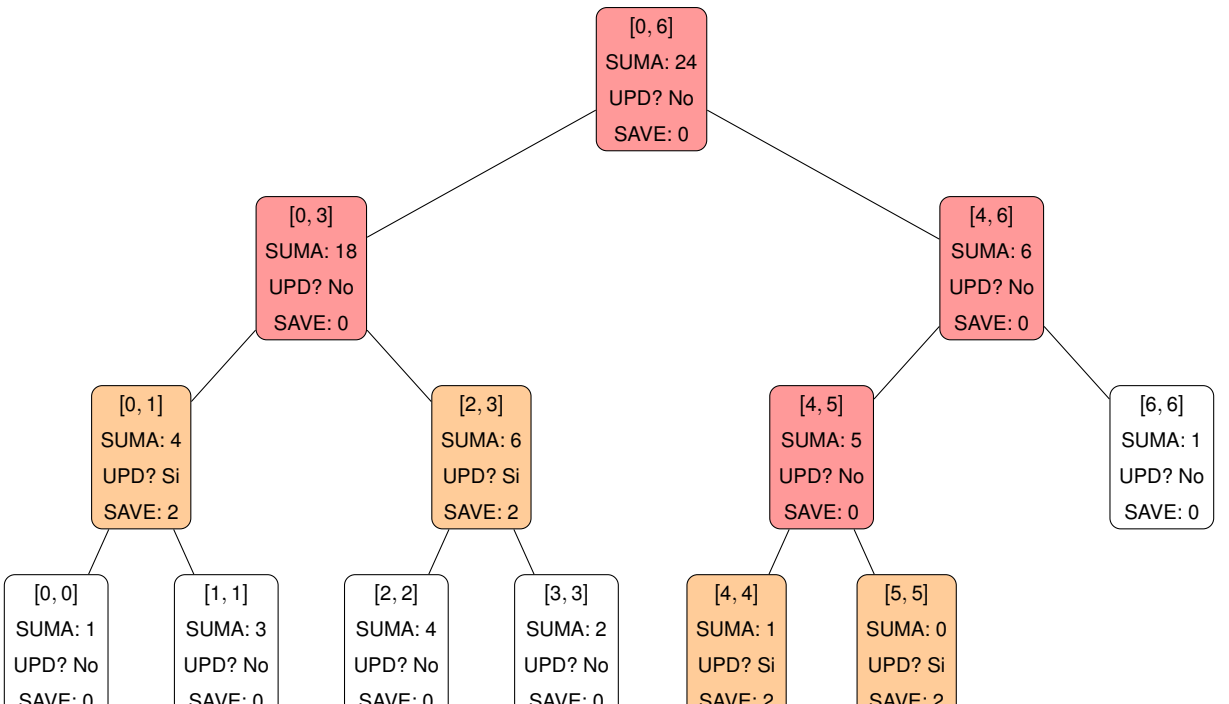
Segment Tree con Lazy Progation

Cada valor del rango 0 a 5 se aumentará en 2.



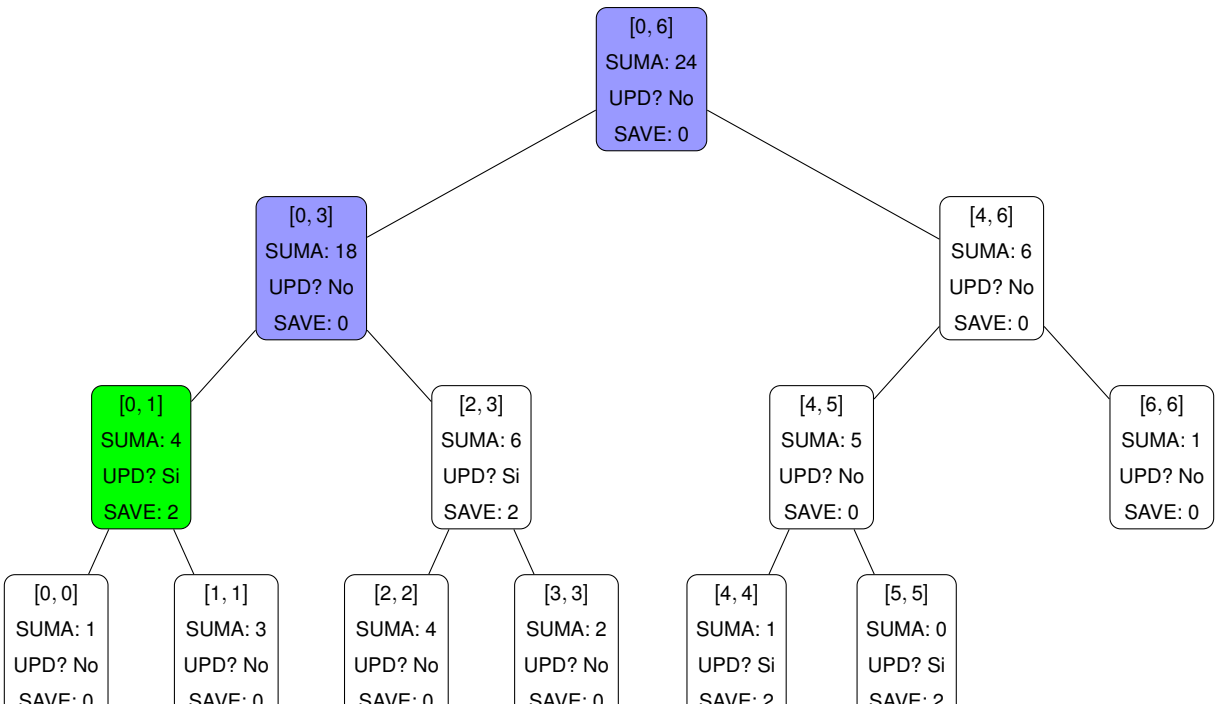
Segment Tree con Lazy Progation

Cada valor del rango 0 a 5 se aumentará en 2.



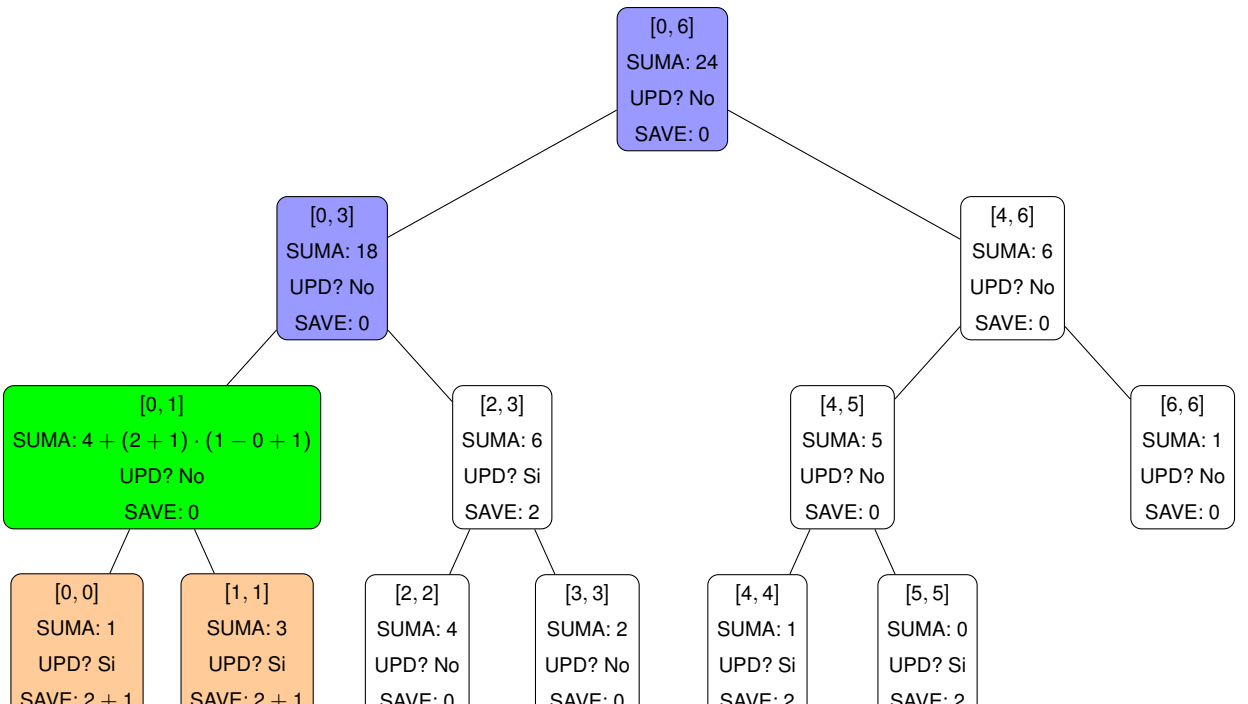
Segment Tree con Lazy Progation

Cada valor del rango 0 a 1 se aumentará en 1.



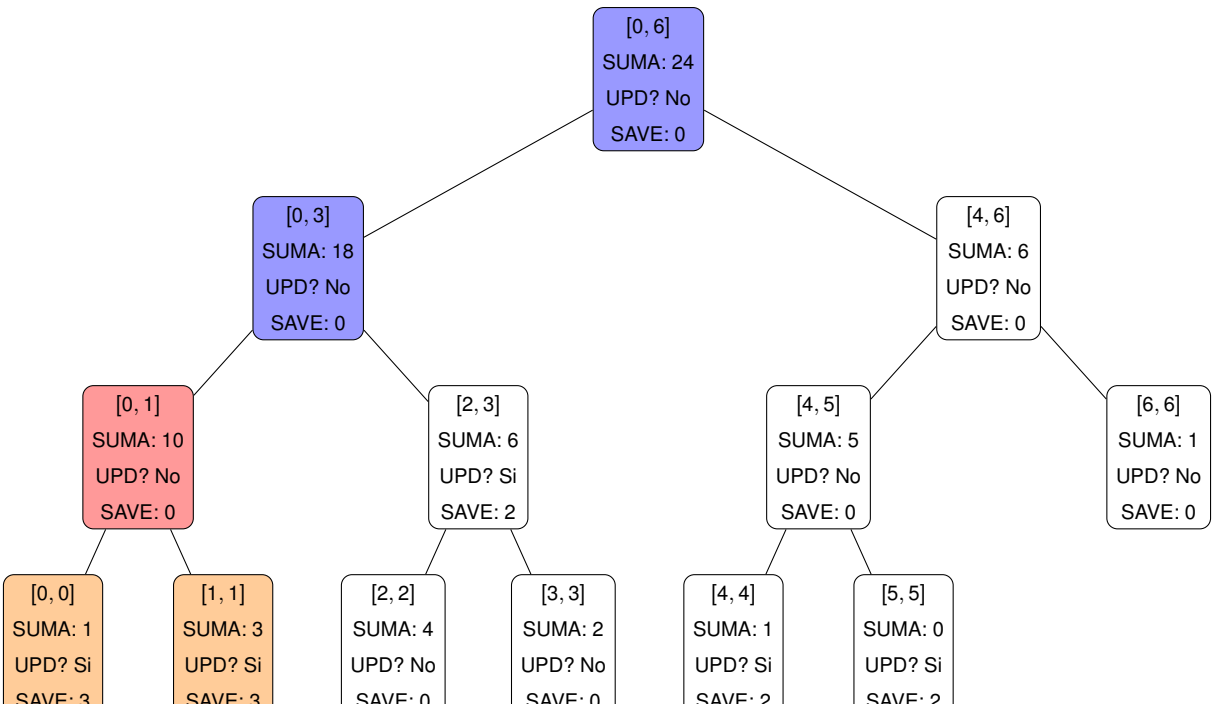
SEGMENT TREE CON LAZY PROGATION

Cada valor del rango 0 a 1 se aumentará en 1.



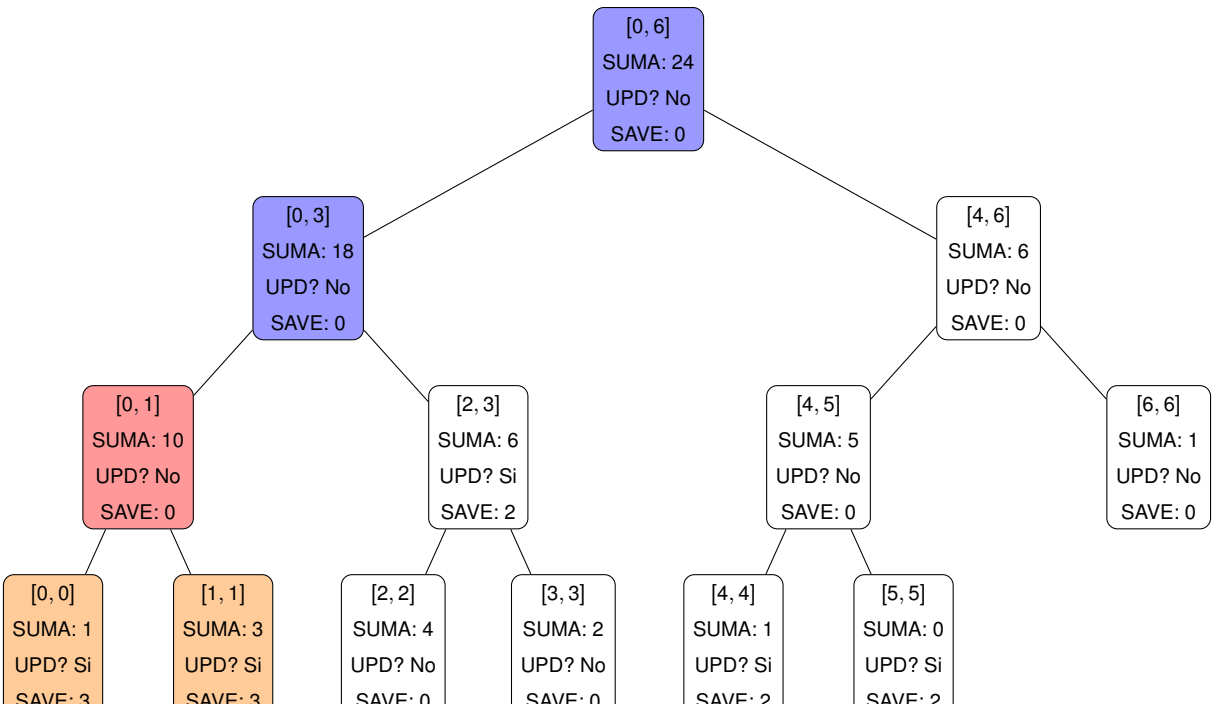
Segment Tree con Lazy Progation

Cada valor del rango 0 a 1 se aumentará en 1.



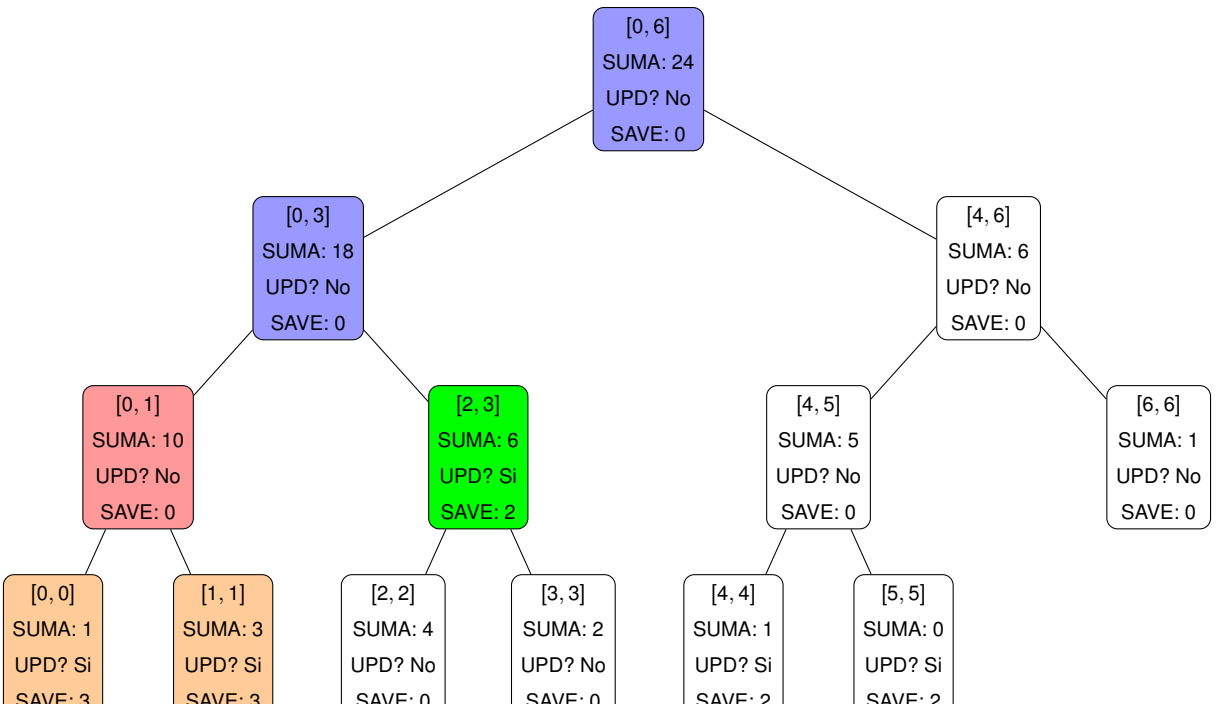
Segment Tree con Lazy Progation

Cada valor del rango 0 a 1 se aumentará en 1.



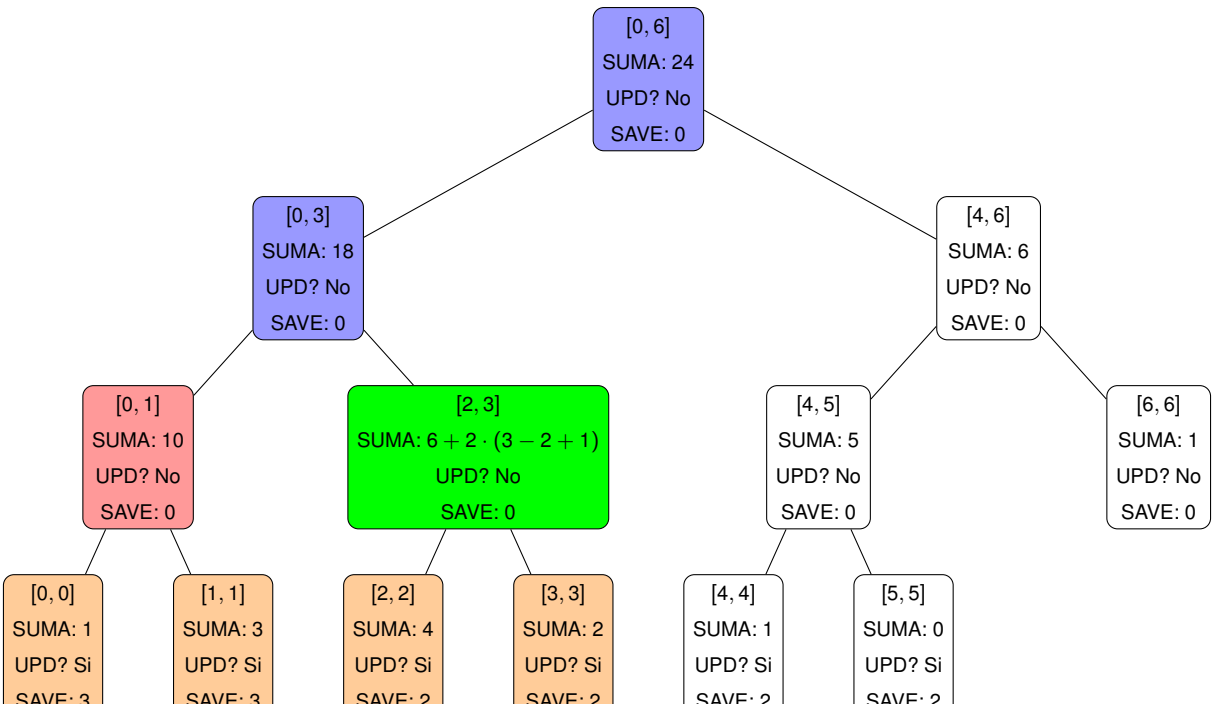
Segment Tree con Lazy Progation

Cada valor del rango 0 a 1 se aumentará en 1.



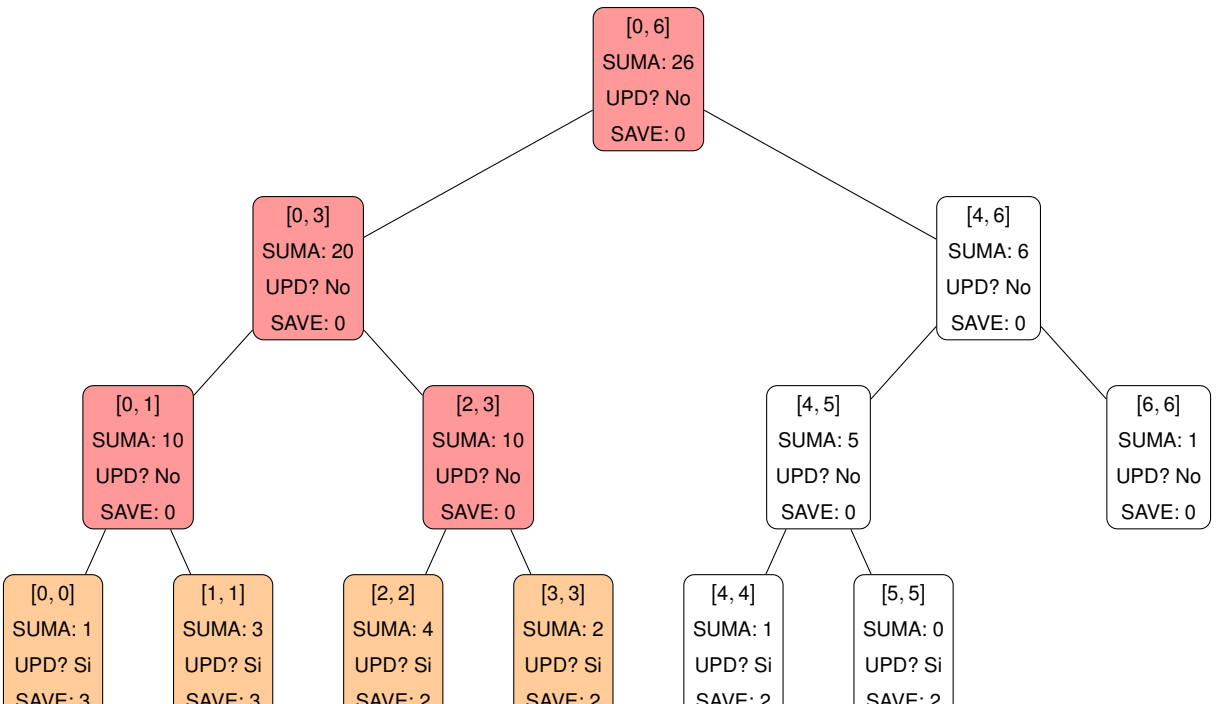
Segment Tree con Lazy Progation

Cada valor del rango 0 a 1 se aumentará en 1.



Segment Tree con Lazy Progation

Cada valor del rango 0 a 1 se aumentará en 1.



Complejidad y Código

La complejidad de hacer update en rango es $O(\log n)$ **amortizado**.

```
template<
  class T1, // la respuesta a la consulta
  class T2, // los valores guardados debido al lazy
  T1 merge(T1, T1),
  void pushUpd(T2&, T2&, int, int, int, int), // empuja el update de un nodo a otro
  void applyUpd(T2&, T1&, int, int)           // aplica la actualizacion guardada en en el
      nodo
>
struct SegmentTreeLazy
```

Ejemplo: Complejidad y Código

La complejidad de hacer update en rango es $O(\log n)$ **amortizado**.

```
long long merge(long long a, long long b){
  return a + b;
}
void pushUpd(long long &u1, long long &u2, int l1, int r1, int l2, int r2){
  u2 += u1;
}
void applyUpd(long long &u, long long &v, int l, int r){
  v += (r - l + 1) * u;
}

int main() {
  // se asume que aqui se setea todo lo previo para el problema
  SegmentTreeLazy<long long, long long, merge, pushUpd, applyUpd> ST(nums);

  cout << ST.query(i, j) << "\n";
  ST.update(i, j, k);
  return 0;
}
```

El código completo para usarlo se encuentra en Segment Tree Lazy[1].

[1] Este código fue realizado por Javier Oliva

REFERENCES I