Segment Tree

Profs. Andre Gustavo, Salles Magalhaes

Segment Tree - somas de intervalos

- Dado um array de números, como achar a soma dos valores em um intervalo?
- Solução fácil e eficiente: prefix-sum
- Para permitir atualizações: segment tree

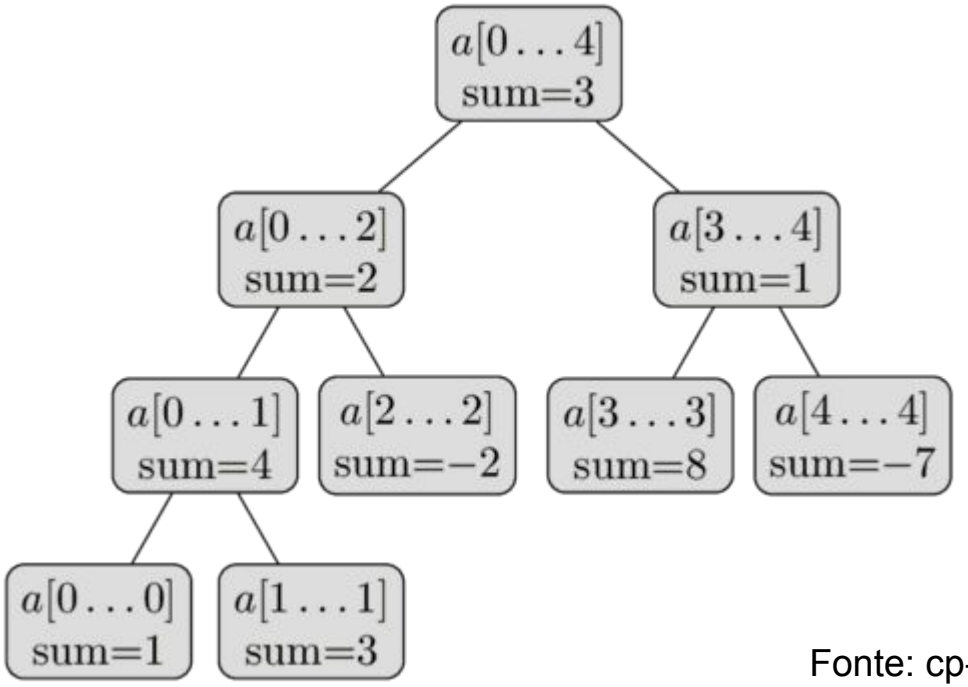
Array	Values =	8	7	3	9	5	1	10
A	Indices =	0	1	2	3	4	5	6

Segment Tree

- Utilizada também para RMQ (Range Minimum Queries) e vários outros tipos de consultas
- Permite consultas **dinâmicas**
- Exemplos:
  - Achar menor valor entre posições 2 e 5 de um array
  - Soma dos valores entre as posições 3 e 10 (fácil com prefix-sum, mas seg-tree → dinâmico)

Segment Tree - somas de intervalos

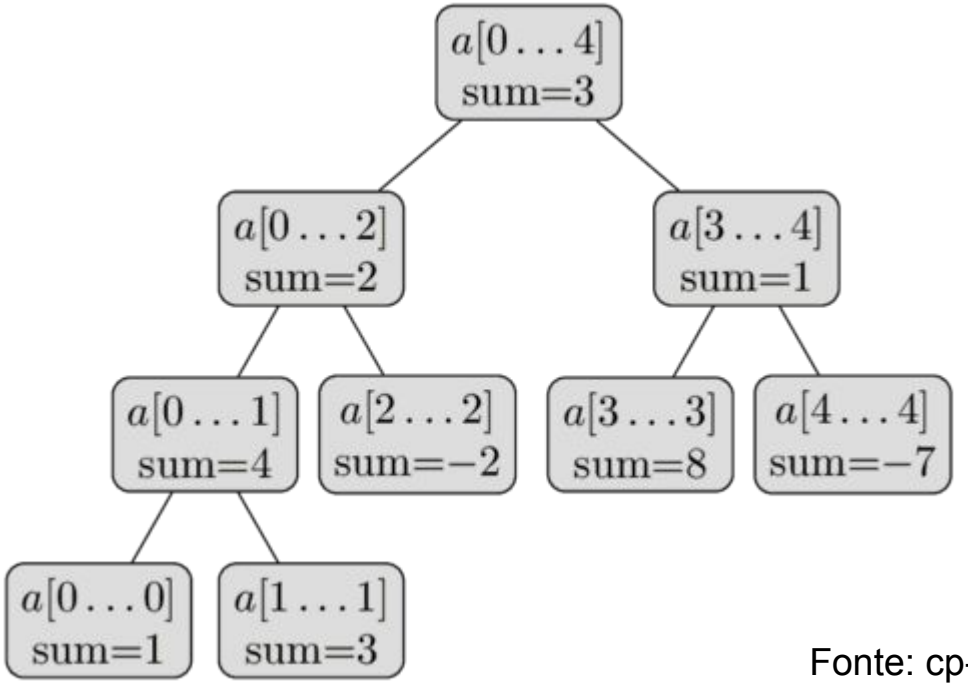
- Material baseado no tutorial do cp-algorithms: https://cp-algorithms.com/data_structures/segment_tree.html#structure-of-the-segment-tree
- Decisões ao criar seg-tree:
  - **Valor** a ser armazenado nos nodos. Exemplo: soma dos elementos em um segmento, índice do menor elemento, índice do maior, etc.
  - Como fazer **merge** de segmentos. Exemplo: soma de dois segmentos.



Fonte: cp-algorithms

Segment Tree - soma de valores a intervalos

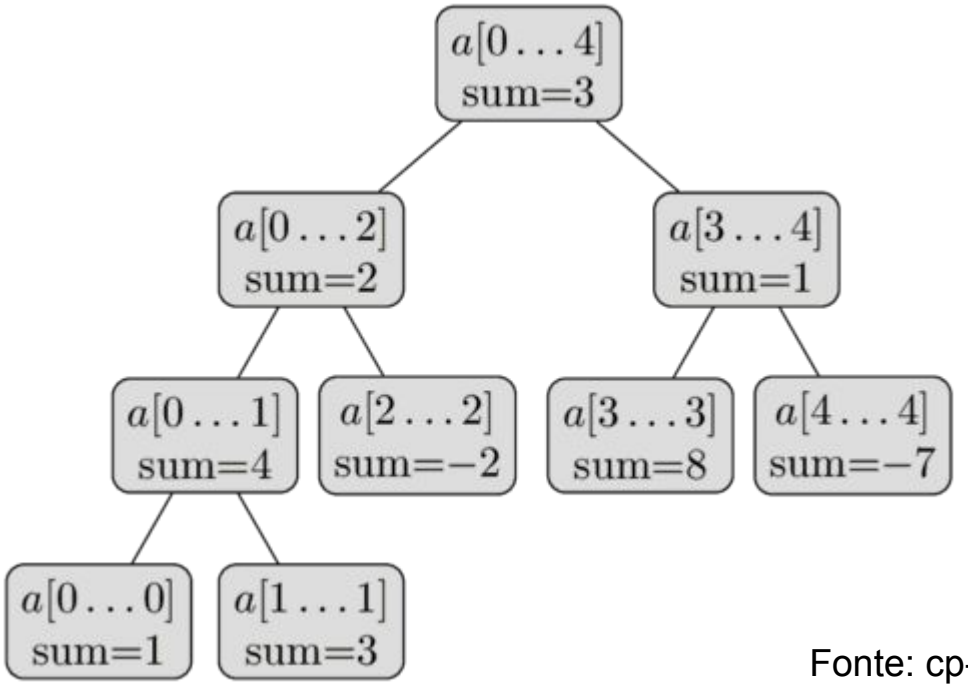
- Como armazenar a seg-tree?
  - Normalmente armazenadas de forma similar a heaps (em vetores)
  - Elementos ficam nas folhas. Cada nodo armazena apenas soma (ou similar) → intervalo é implícito.
  - Segtree terá, no máximo, 4n vértices → armazenamos sempre 4n vértices por simplicidade (algumas implementações podem armazenar menos)



Fonte: cp-algorithms

Segment Tree

- Ver implementação em segTreeSum.cpp (adaptado de https://cp-algorithms.com/data_structures/segment_tree.html#structure-of-the-segment-tree)

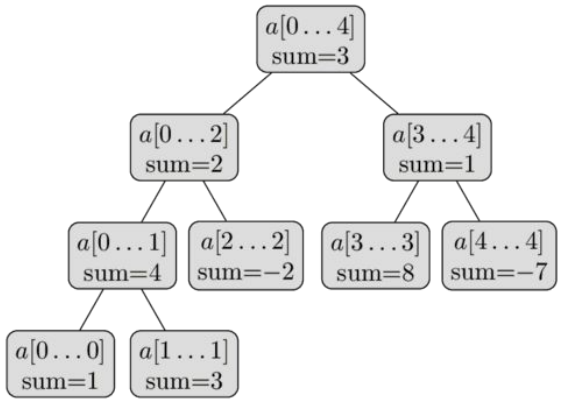


Fonte: cp-algorithms

Segment Tree

```
class SegTree {
public:
    SegTree(int n) {
        mxPos = n-1;
        t.resize(4*n,0);
    }
```

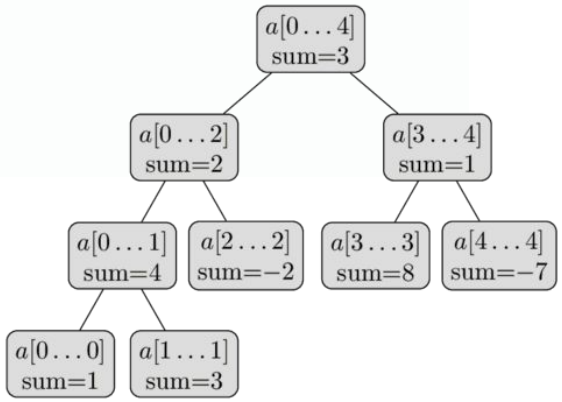
```
private:
    vector<int> t;
    int mxPos;


};
```



Fonte: cp-algorithms

Segment Tree

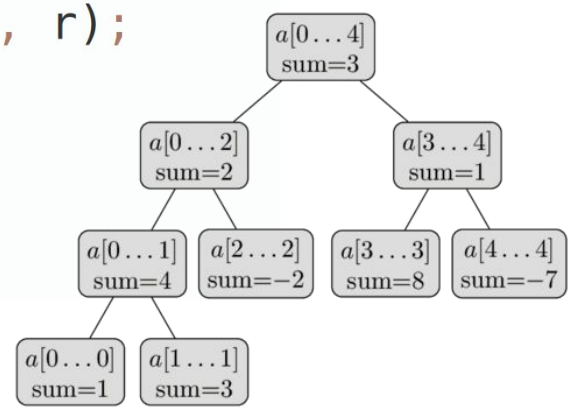
```
//constroi a segTree com os elementos de a
//inicialmente, tl=0, tr=n-1
void build(vector<int> &a, int tl, int tr, int v) {
    if (tl == tr) { //folha
        t[v] = a[tl];
    } else {
        int tm = (tl + tr) / 2; //meio do segmento
        //constroi nodos da esquerda ([tl,tm])
        build(a, tl, tm, v*2);
        //constroi nodos da direita ([tm+1,tr])
        build(a, tm+1, tr, v*2+1);
        //junta o resultado dos segmentos da esquerda e direita
        //em geral, aqui é lugar mais "adaptado"
        t[v] = t[v*2] + t[v*2+1];
    }
}
void build(vector<int> &a) {
    build(a,0,mxPos,1);
}
```



Fonte: cp-algorithms

Segment Tree

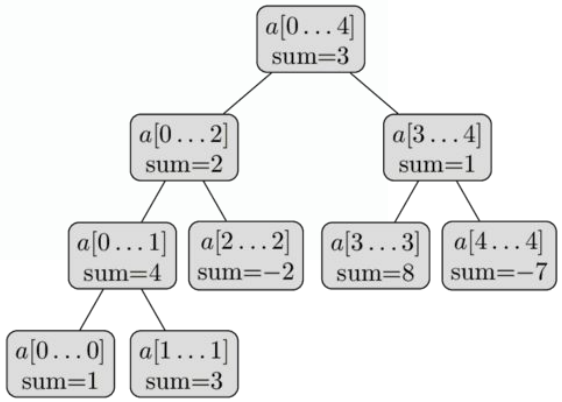
```
//consulta para encontrar a soma do intervalo [l,r]
//procura no vértice v, representando os intervalos
//[tl,tr] (na árvore)
int sum(int v, int tl, int tr, int l, int r) {
    if (l > r)
        return 0; //adaptar, caso outro tipo de consulta
    if (l == tl && r == tr) {
        return t[v];
    }
    int tm = (tl + tr) / 2;

    //chama SEMPRE para os dois lados
    //mesmo se desnecessário --> ok (primeiro if)
    //adaptar, caso outro tipo de consulta
    return sum(v*2, tl, tm, l, min(r, tm))
           + sum(v*2+1, tm+1, tr, max(l, tm+1), r);
}
int sum(int l, int r) {
    return sum(1,0,mxPos,l,r);
}
```



Fonte: cp-algorithms

Segment Tree

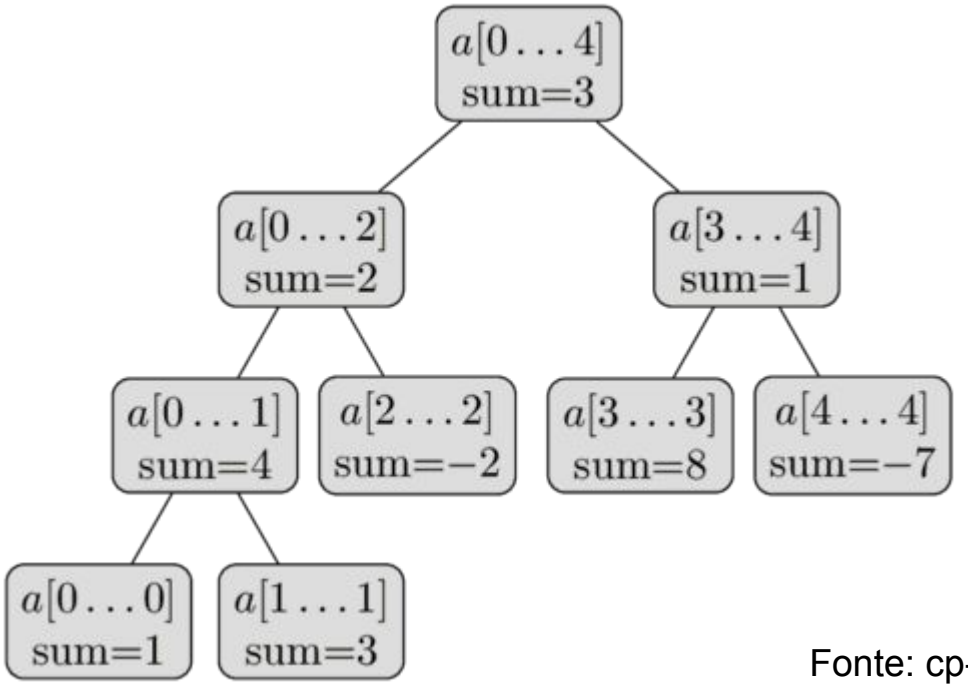
```
void update(int v, int tl, int tr, int pos, int new_val) {
    if (tl == tr) {
        t[v] = new_val;
    } else {
        int tm = (tl + tr) / 2;
        if (pos <= tm) //nodo está na esquerda?
            update(v*2, tl, tm, pos, new_val);
        else //nodo está na direita?
            update(v*2+1, tm+1, tr, pos, new_val);

        //atualiza raiz atual
        //adaptar, caso outro tipo de consulta
        t[v] = t[v*2] + t[v*2+1];
    }
}
void update(int pos, int new_val) {
    update(1,0,mxPos,pos,new_val);
}
```



Fonte: cp-algorithms

Segment Tree

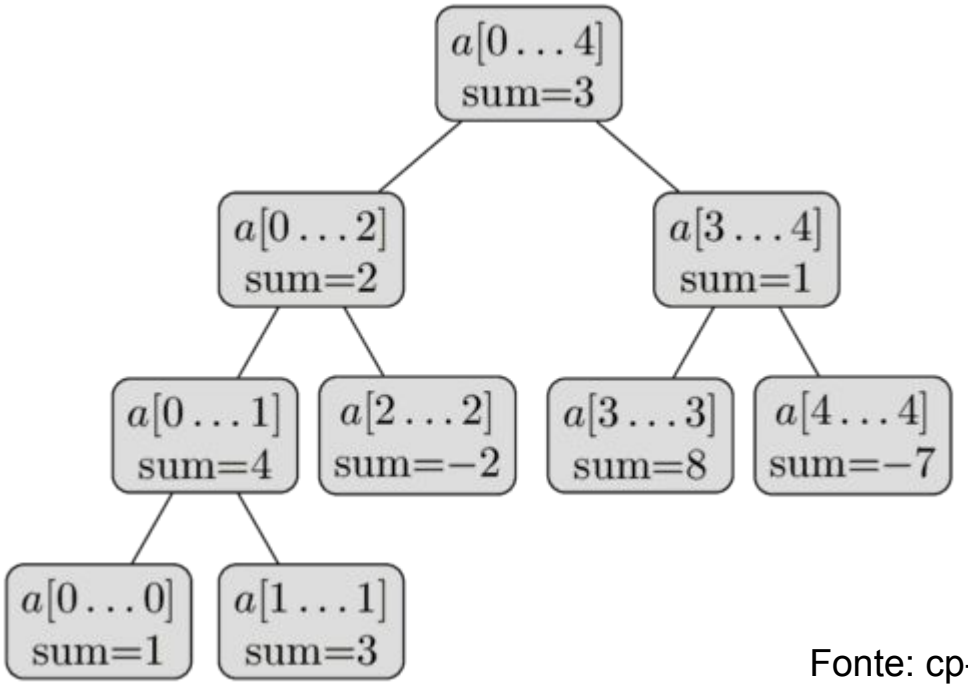
- Dado um array de inteiros, como encontrar a o maior elemento e a frequência dele?
  - O que adaptar na seg-tree?
  - Veja solução em: segTreeMaxAndFreq.cpp



Fonte: cp-algorithms

Segment Tree

- Outro problema: calcular máximo divisor comum (MDC,GCD) dos elementos em um intervalo ou mínimo múltiplo comum.
  - Ideia similar ao que vimos antes!
  - MDC([0,3]) = 5, MDC([4,6]) = 10 → MDC([0,6]) = …
  - MMC([0,3]) = 3, MMC([4,6]) = 9 → MMC([0,6]) = …
  - Adaptar função de combinação!



Fonte: cp-algorithms

Segment Tree

- Outro problema: contar quantos 0s em um intervalo e (usando outra função) encontrar o k-ésimo 0.
  - Contar quantos 0s: fácil (cada nodo armazena quantos 0s no intervalo)
  - Achar k-ésimo: usar informação acima

```
int find_kth(int v, int tl, int tr, int k) {
    if (k > t[v]) //quero o 3o 0,mas há só 2...
        return -1;
    if (tl == tr)
        return tl;
    int tm = (tl + tr) / 2;
    if (t[v*2] >= k) //quero o 3o 0 e na esquerda há 5 0s
        return find_kth(v*2, tl, tm, k);
    else
        return find_kth(v*2+1, tm+1, tr, k - t[v*2]);
}
```

por que isso?

Segment Tree

- Dado um valor K, achar o menor índice i do vetor onde a soma dos elementos entre [0,i] é >= K
  - Podemos resolver com busca binária em array de prefix sum
  - Em seg tree: dados podem ser atualizados dinamicamente
  - Solução com seg tree:
    - Guardar soma na árvore
    - Descer, indo para a direita ou esquerda comparando soma e K
    - Exemplo: se K=10 e soma do intervalo esquerdo = 5 → solução está no intervalo direito.

Segment Tree

- Dado um valor X, achar o primeiro elemento >= X em um array.
- Ideias? exemplo: X=3, V=[1,4,6,1,2,0] → 1

Segment Tree

- Dado um valor X, achar o primeiro elemento >= X em um array.
- Ideias? exemplo: X=3, V=[1,4,6,1,2,0] → 1
  - E se o problema fosse: tem um número >= 3 no intervalo [0,4] ?

Segment Tree

- Dado um valor X, achar o primeiro elemento >= X em um array.
- Ideias? exemplo: X=3, V=[1,4,6,1,2,0] → 1
  - E se o problema fosse: tem um número >= 3 no intervalo [0,i] ?
  - Solução: fazer busca binária para achar o primeiro índice i tal que temMaiorIgual(X,i) retorna true. Precisamos de uma seg tree que suporte a consulta “max”
  - $O(\log^2 n)$ (por que?)
  - Há outras soluções “andando na árvore”

Segment Tree

- Dado um intervalo, encontrar o subintervalo com maior soma (seg tree pode ter valores negativos)

Segment Tree

- Dado um intervalo, encontrar o subintervalo com maior soma (seg tree pode ter valores negativos)
  - Armazenar em cada nodo: (ex: [4,3,-9,7,1,-9,3,2])
    - Soma do intervalo todo (ex: 2)
    - Soma do maior prefixo do intervalo (ex: 7)
    - Soma do maior sufixo do intervalo (ex:5)
    - Maior soma no interior do intervalo (ex: 8)

Segment Tree

- Ver detalhes no exemplo do cp-programming (https://cp-algorithms.com/data_structures/segment_tree.html#finding-sub segments-with-the-maximal-sum). Nessa implementação criaram funções adicionais para separar melhor as operações e facilitar adaptação:

```
struct data {
    int sum, pref, suff, ans;
};

data combine(data l, data r) {
    data res;
    res.sum = l.sum + r.sum;
    res.pref = max(l.pref, l.sum + r.pref);
    res.suff = max(r.suff, r.sum + l.suff);
    res.ans = max(max(l.ans, r.ans), l.suff + r.pref);
    return res;
}
```

```
data make_data(int val) {
    data res;
    res.sum = val;
    res.pref = res.suff = res.ans = max(0, val);
    return res;
}
void build(int a[], int v, int tl, int tr) {
    if (tl == tr) {
        t[v] = make_data(a[tl]);
    } else {
        int tm = (tl + tr) / 2;
        build(a, v*2, tl, tm);
        build(a, v*2+1, tm+1, tr);
        t[v] = combine(t[v*2], t[v*2+1]);
    }
}
void update(int v, int tl, int tr, int pos, int new_val) {
    if (tl == tr) {
        t[v] = make_data(new_val);
    } else {
        int tm = (tl + tr) / 2;
        if (pos <= tm)
            update(v*2, tl, tm, pos, new_val);
        else
            update(v*2+1, tm+1, tr, pos, new_val);
        t[v] = combine(t[v*2], t[v*2+1]);
    }
}
```

Se negativo, maior prefixo/sufixo/resposta é intervalo vazio…

Segment Tree

```
//dado um intervalo [l,r]
//--> encontra o maior prefixo, sufixo, soma total e
//    maior soma dentro dele (principal resultado)
data query(int v, int tl, int tr, int l, int r) {
    if (l > r)
        return make_data(0); //intervalo vazio
    if (l == tl && r == tr) //um só elemento
        return t[v];
    int tm = (tl + tr) / 2;
    return combine(query(v*2, tl, tm, l, min(r, tm)),
                   query(v*2+1, tm+1, tr, max(l, tm+1), r));
}
```

Segment Tree - Guardando o intervalo todo nos nodos

- Em algumas aplicações, podemos querer guardar TODOS os elementos de um intervalo em cada nodo (em vez do min, max, soma, etc).
  - Exemplo: guardar sublistas ordenadas, maps, sets do intervalo, bit tree, outra seg tree!!!, etc.
- Memória: O(n log n)

Segment Tree - Guardando o intervalo todo nos nodos

- Responder a consultas do tipo: achar menor elemento >= X no intervalo [l,r]
  - Cada nodo contém o intervalo todo ordenado (construção em O(n log n))
  - Conhecida como merge sort tree

```
vector<int> t[4*MAXN];

void build(int a[], int v, int tl, int tr) {
    if (tl == tr) {
        t[v] = vector<int>(1, a[tl]);
    } else {
        int tm = (tl + tr) / 2;
        build(a, v*2, tl, tm);
        build(a, v*2+1, tm+1, tr);
        merge(t[v*2].begin(), t[v*2].end(), t[v*2+1].begin(), t[v*2+1].end(),
              back_inserter(t[v]));
    }
}
```

Segment Tree - Guardando o intervalo todo nos nodos

- Se intervalo onde estou buscando = intervalo da raiz atual
  - Faça busca binária para achar o 1o >= x
- Senão:
  - Acha o menor >=x nas subárvores esquerda e direita e pega o mínimo entre eles.
- O ($\log^2(n)$)

```
int query(int v, int tl, int tr, int l, int r, int x) {
    if (l > r)
        return INF;
    if (l == tl && r == tr) {
        vector<int>::iterator pos = lower_bound(t[v].begin(), t[v].end(), x);
        if (pos != t[v].end())
            return *pos;
        return INF;
    }
    int tm = (tl + tr) / 2;
    return min(query(v*2, tl, tm, l, min(r, tm), x),
               query(v*2+1, tm+1, tr, max(l, tm+1), r, x));
}
```

Segment Tree - Guardando o intervalo todo nos nodos

- Responder a consultas do tipo: achar menor elemento >= X no intervalo [l,r] E permitir modificações do tipo set(i,x)

Segment Tree - Guardando o intervalo todo nos nodos

- Responder a consultas do tipo: achar menor elemento >= X no intervalo [l,r] E permitir modificações do tipo set(i,x)
  - Armazenar lista ordenada em um multiset (suporta lower_bound e permite atualização (remova **UMA** ocorrência e insira))

```
void update(int v, int tl, int tr, int pos, int new_val) {
    t[v].erase(t[v].find(a[pos]));
    t[v].insert(new_val);
    if (tl != tr) {
        int tm = (tl + tr) / 2;
        if (pos <= tm)
            update(v*2, tl, tm, pos, new_val);
        else
            update(v*2+1, tm+1, tr, pos, new_val);
    } else {
        a[pos] = new_val;
    }
}
```

Segment Tree - Lazy propagation - somar em segmento

- Atualização: adicionar X ao segmento [l,r]
- Consulta: qual o valor do elemento na posição i?
- Solução: armazenar nos vértices o valor a ser adicionado no intervalo correspondente.
  - Exemplo: se for adicionar 2 ao intervalo todo → somar 2 à raiz.
- Veja os exemplos de código a seguir (do cp-programming)
- Complexidade da atualização: O(log n)

```
void build(int a[], int v, int tl, int tr) {
    if (tl == tr) {
        t[v] = a[tl];
    } else {
        int tm = (tl + tr) / 2;
        build(a, v*2, tl, tm);
        build(a, v*2+1, tm+1, tr);
        t[v] = 0;
    }
}

void update(int v, int tl, int tr, int l, int r, int add) {
    if (l > r)
        return;
    if (l == tl && r == tr) {
        t[v] += add;
    } else {
        int tm = (tl + tr) / 2;
        update(v*2, tl, tm, l, min(r, tm), add);
        update(v*2+1, tm+1, tr, max(l, tm+1), r, add);
    }
}
```

valor a ser somado ao intervalo

Essa parte da árvore é exatamente o intervalo onde quero somar?

Se não for, divida intervalo + recursão.

Segment Tree - Lazy propagation - somar em segmento

```
int get(int v, int tl, int tr, int pos) {
    if (tl == tr)
        return t[v];
    int tm = (tl + tr) / 2;
    if (pos <= tm)
        return t[v] + get(v*2, tl, tm, pos);
    else
        return t[v] + get(v*2+1, tm+1, tr, pos);
}
```

Exemplo da primeira chamada: temos que somar o valor que está na raiz (t[v]), representando o intervalo inteiro, e ir à parte da árvore referente ao elemento.

Segment Tree - somar em segmento sem lazy propagation!

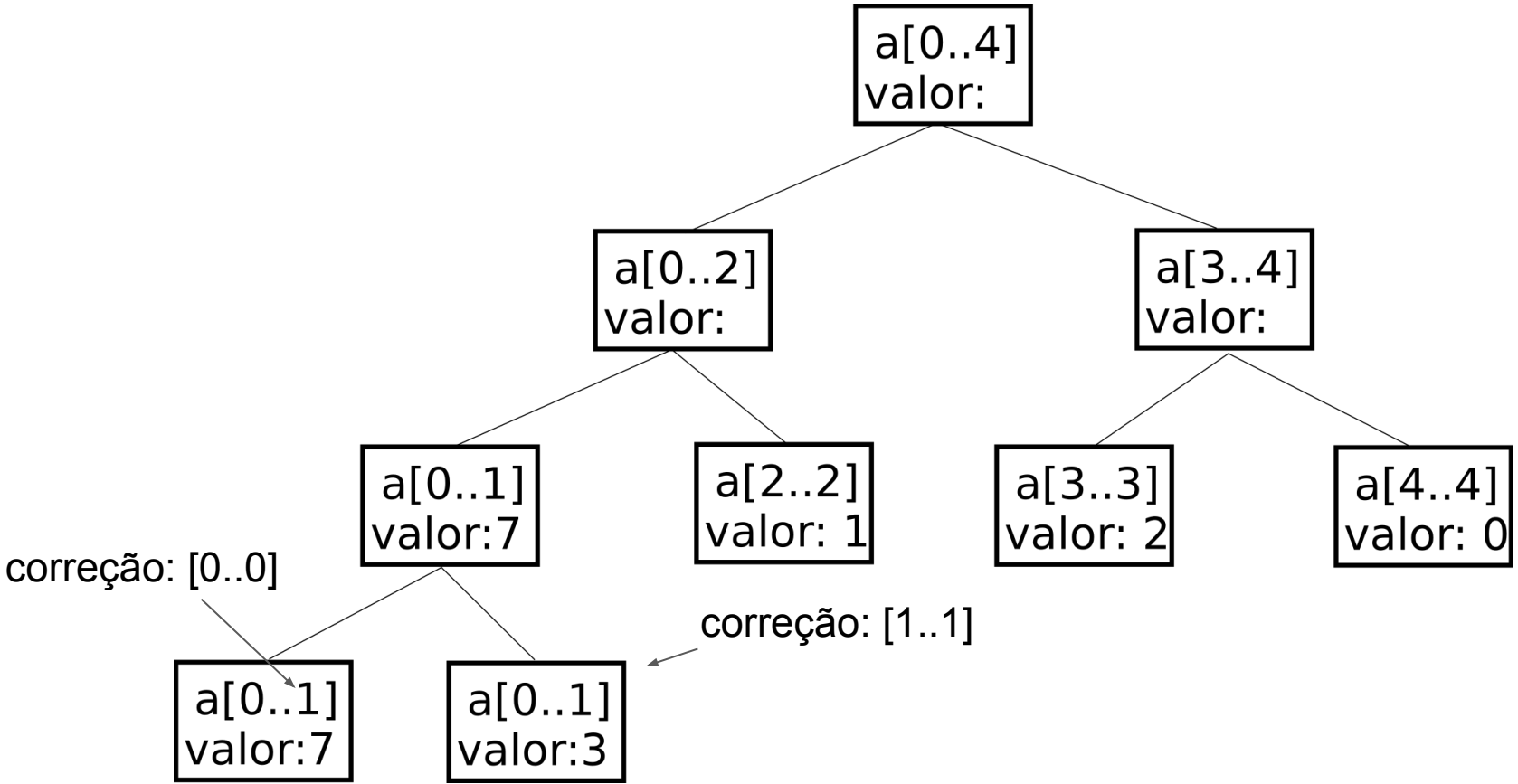
- Atualização: adicionar X ao segmento [l,r]
- Consulta: qual o valor do elemento na posição i?
- Outra solução:
  - Usar segtree que permite update atribuindo valor a posição e consulta descobrindo soma de intervalo.
  - Adicionar X ao intervalo [l,r] → fazer t[l] += X e t[r+1] += -X
  - Descobrir o valor da posição i → retornar sum([0,i])
- Exemplo:
  - Adicionar: 10 a [2,4], 20 a [3,5]
  - Qual o valor na posição 3? 5? 7? 1?

Segment Tree - Lazy propagation - atribuir a segmento

- Atualização: atribuir X a todos valores no segmento [l,r]
- Consulta: qual o valor do elemento na posição i?
- Solução (O(log n)): de novo, fazer propagação “preguiçosa”
  - Um nodo será marcado se o valor nele corresponder ao segmento inteiro.
  - Update: ao atribuir valor a segmento, marcamos o(s) nodo(s) correspondentes e atribuímos o valor a eles.
  - Nodos de baixo: não são atualizados (lazy), mas isso não é problema (pois as buscas pararão nos de cima).
  - Novo update: nodo de cima desmarcado, valor do nodo de cima e novo valor propagados para os intervalos.

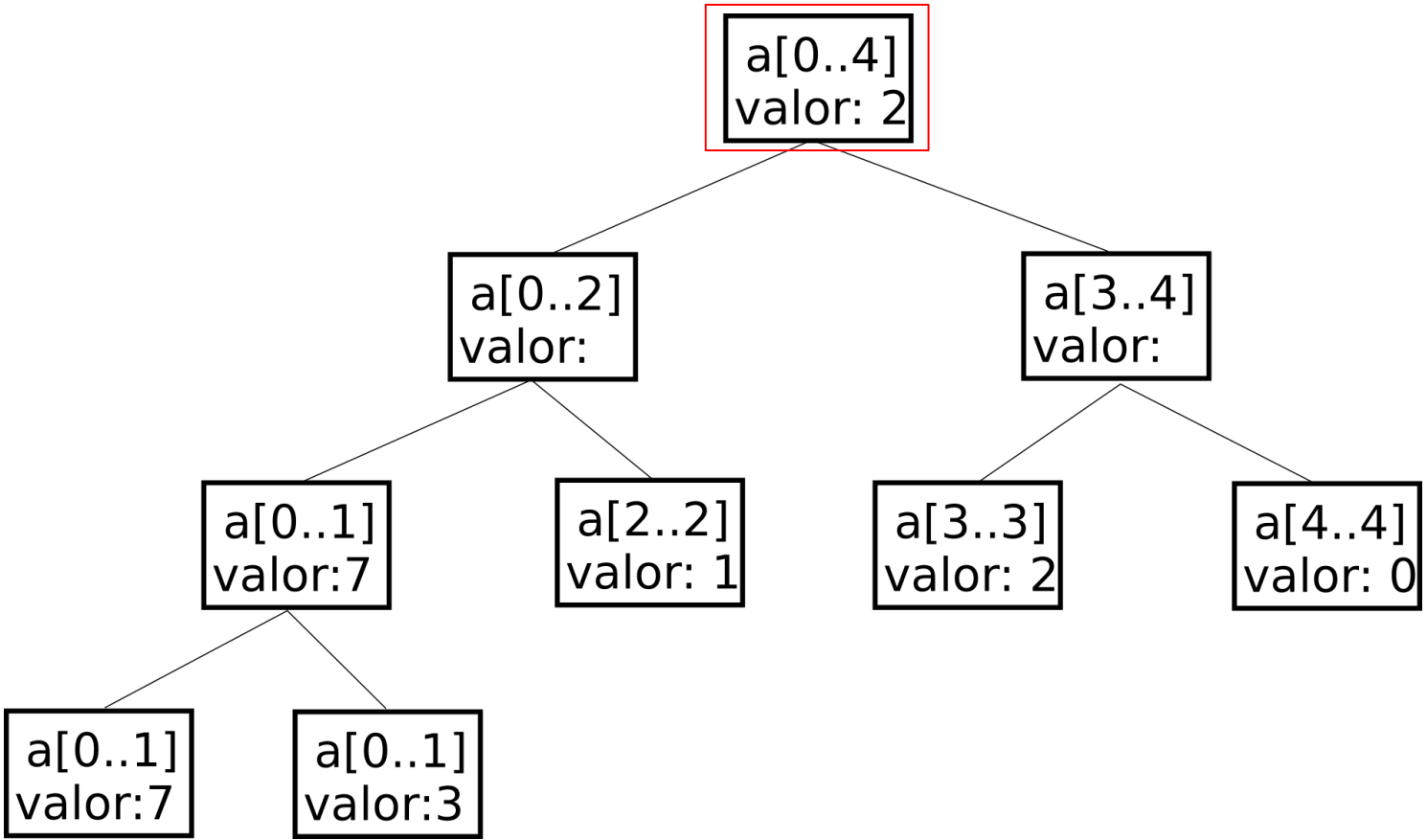
Segment Tree - Lazy propagation - atribuir a segmento

- Exemplo: [7,3,1,2,0] . Apenas as folhas estão “marcadas”
- Como atribuir 2 a [0...4] ?



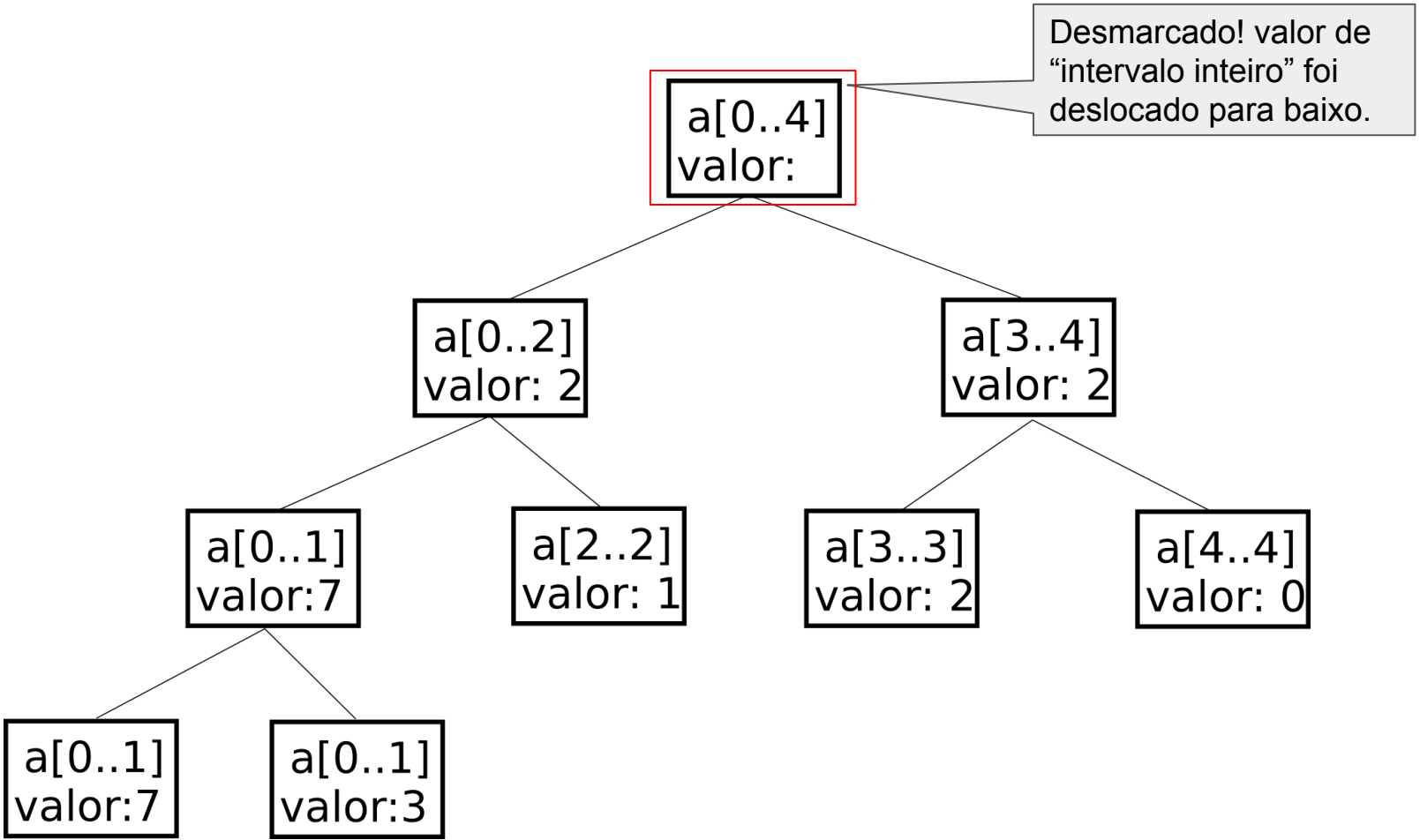
Segment Tree - Lazy propagation - atribuir a segmento

- Exemplo: [7,3,1,2,0] . Apenas as folhas estão “marcadas”
- Como atribuir 2 a [0...4] ? Como são as buscas?
- Atribuindo 7 a [0..1] . Como fica?



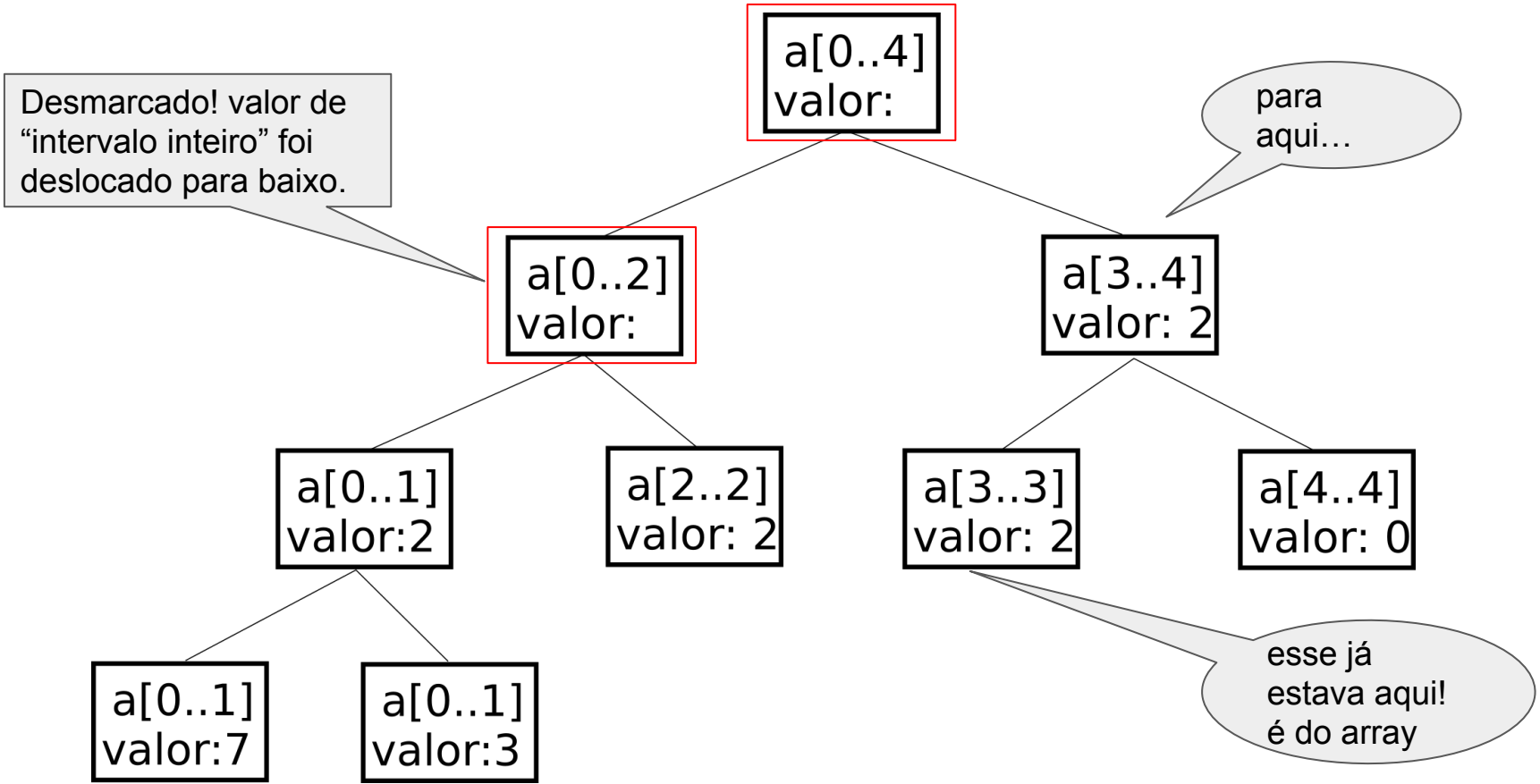
Segment Tree - Lazy propagation - atribuir a segmento

- Exemplo: [7,3,1,2,0] . Apenas as folhas estão “marcadas”
- Como atribuir 2 a [0...4] ? Como são as buscas?
- Atribuindo 7 a [0..1] . Como fica?
- O 2 se propaga pelos filhos dos nodos ao longo do caminho até [0..1]



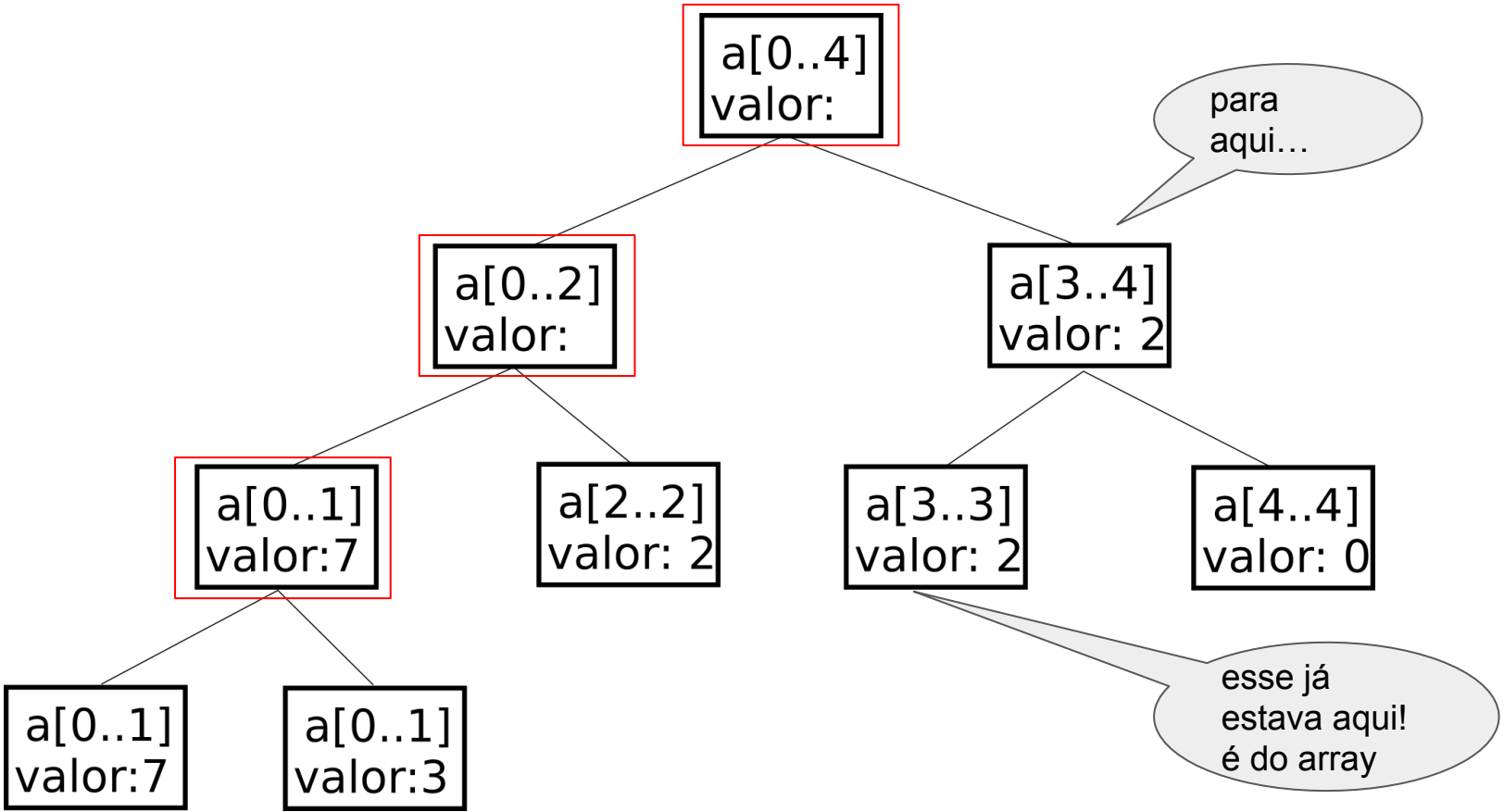
Segment Tree - Lazy propagation - atribuir a segmento

- Exemplo: [7,3,1,2,0] . Apenas as folhas estão “marcadas”
- Como atribuir 2 a [0...4] ? Como são as buscas?
- Atribuindo 7 a [0..1] . Como fica?
- O 2 se propaga pelos filhos dos nodos ao longo do caminho até [0..1]



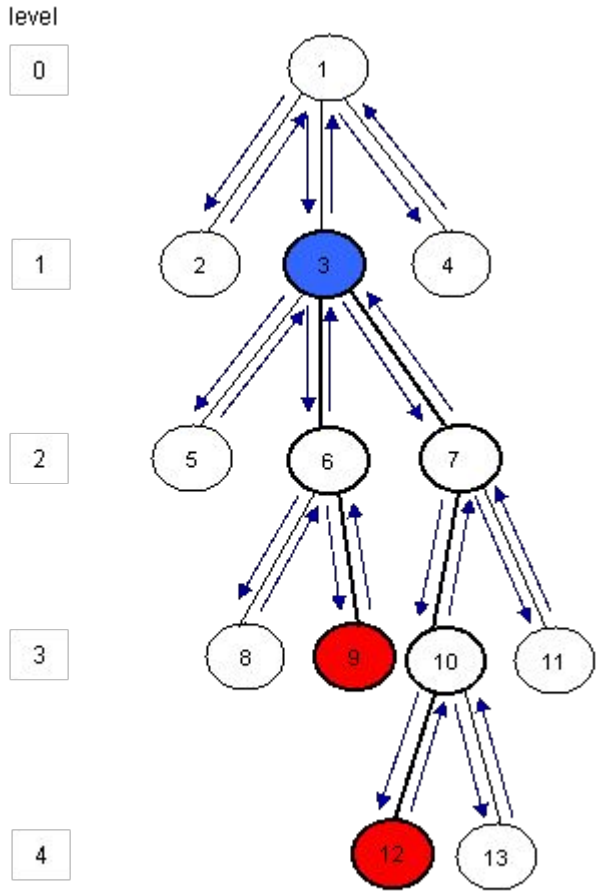
Segment Tree - Lazy propagation - atribuir a segmento

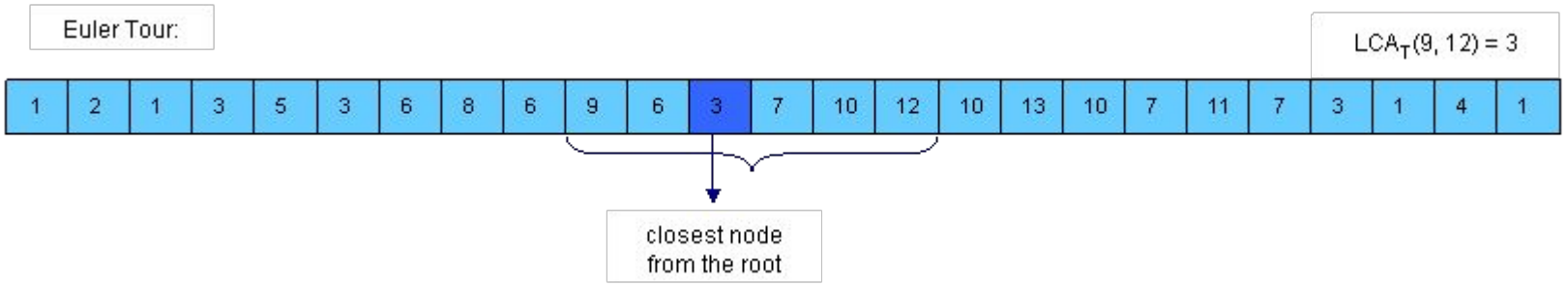
- Exemplo: [7,3,1,2,0] . Apenas as folhas estão “marcadas”
- Como atribuir 2 a [0...4] ? Como são as buscas?
- Atribuindo 7 a [0..1] . Como fica?
- O 2 se propaga pelos filhos dos nodos ao longo do caminho até [0..1]



Segment Tree - Outra aplicação: LCA

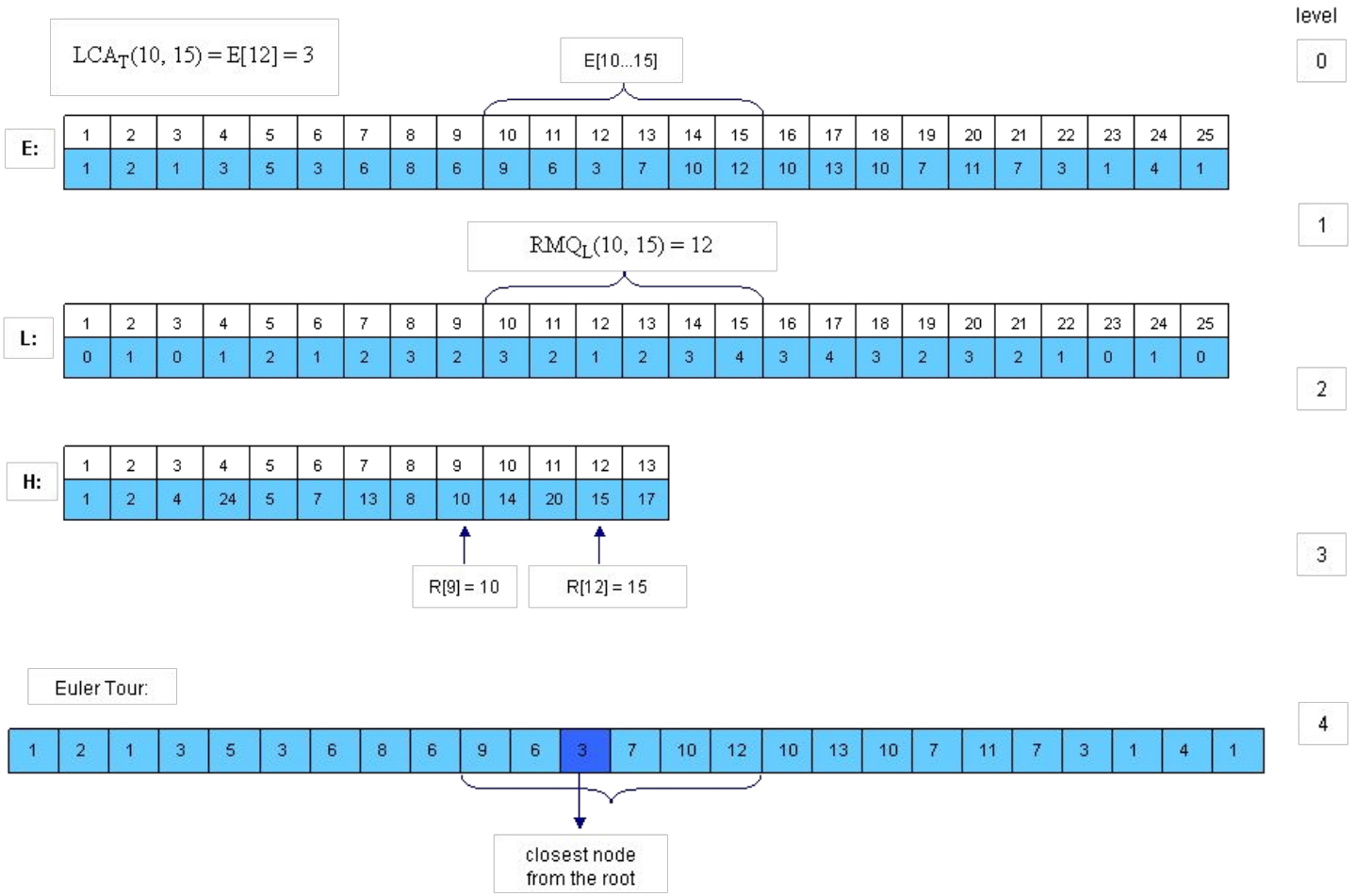
- Segtrees podem ser utilizadas para calcular LCA
- Fonte de consulta (e fonte das imagens destes slides):
  - https://www.topcoder.com/thrive/articles/Range%20Minimum%20Query%20and%20Lowest%20Common%20Ancestor
- Gere um passeio de Euler da árvore
- LCA: nó no menor nível que estiver entre duas ocorrências dos vértices da consulta no tour.
- Precisamos:
  - E[1, 2*N-1] – nodos visitados no Euler tour.
  - L[1, 2*N-1] – nível dos nodos acima
  - H[1, N] – H[i] primeira ocorrência de i em E (qualquer ocorrência serve)

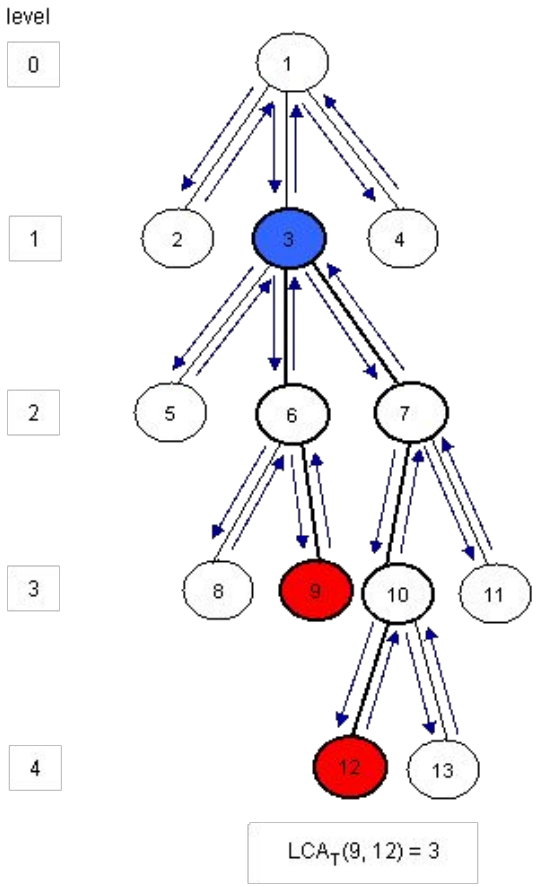




Segment Tree - Outra aplicação: LCA

- LCA(u, v) = E[RMQ(H[u], H[v])] (RMQ precisa retornar o índice)
- LCA(9,12):
- R[9] = 10 , R[12] = 15 → 9 e 12 aparecem entre posições 10 e 15 de E.
- Precisamos saber qual é o mínimo (nodo de menor nível) de L nos intervalos de 9 a 12 → é 1 (o nodo é o 3, que é o da posição 12 de E)





Segment Tree - comentários finais

- **Estudar** o máximo de variações de seg tree, **resolver** problemas, levar as várias implementações prontas (para evitar erros).
- Tem que praticar – não adianta apenas ler (pode cair variação diferente…)
- Muitas outras variações:
  - Seg tree 2D, 3D ….
  - Seg tree aplicada a calcular caminhos em árvores (suportando atualização de custos) – usa DFS para criar um array representando um grafo. Exemplo: https://www.youtube.com/watch?v=OSIi4Az_Pbc
  - XOR de intervalo
  - Seg tree persistente (guarda estados antigos da árvore)
  - ……
- Comece pelo cp-algorithms: https://cp-algorithms.com/data_structures/segment_tree.html

Segment Tree - exemplos de problemas

- Enemy is weak CodeForces - 61E
- Dado um vetor de números distintos, quantos índices i,j,k (i<j<k) possuem v[i] > v[j] > v[k] ?
- Exemplo: 10 8 3 1 → 4
- Solução trivial: O(n^3)
- Ideias?

Segment Tree - exemplos de problemas

- Enemy is weak CodeForces - 61E
- Dado um vetor de números distintos, quantos índices i,j,k (i<j<k) possuem v[i] > v[j] > v[k] ?
- Exemplo: 10 8 3 1 → 4
- Solução trivial: O(n^3)
- Ideias?
- Dado um índice j, o que precisamos saber para calcular quantas triplas i,j,k existem? Exemplo: 10 **8** 3 1 e j = 1

Segment Tree - exemplos de problemas

- Enemy is weak CodeForces - 61E
- Dado um vetor de números distintos, quantos índices i,j,k (i<j<k) possuem v[i] > v[j] > v[k] ?
- Exemplo: 10 8 3 1 → 4
- Solução trivial: O(n^3)
- Ideias?
- Dado um índice j, o que precisamos saber para calcular quantas triplas i,j,k existem? Exemplo: 10 **8** 3 1 e j = 1
  - Precisamos saber quantos elementos antes de j estao no intervalo [v[j]+1,INF]
  - E precisamos saber quantos elementos depois de j estao no intervalo [-INF, v[j]-1]
  - Criar seg tree que permite somar 1 (frequencia) ou -1 a um intervalo e consultar o valor em uma posição.
  - Duas seg-trees: uma para a esquerda e uma para a direita
    - 10 → soma 1 ao intervalo [0,10]
    - 8 → soma 1 ao intervalo [0,8] . Para saber quantos >= 8 → query(8)

Segment Tree - exemplos de problemas

- Enemy is weak CodeForces - 61E
- Dado um vetor de números distintos, quantos índices i,j,k (i<j<k) possuem v[i] > v[j] > v[k] ?
- Exemplo: 10 8 3 1 → 4
- Solução trivial: O(n^3)
- Ideias?
- Dado um índice j, o que precisamos saber para calcular quantas triplas i,j,k existem? Exemplo: 10 **8** 3 1 e j = 1
  - Precisamos saber quantos elementos antes de j estao no intervalo [v[j]+1,INF]
  - E precisamos saber quantos elementos depois de j estao no intervalo [-INF, v[j]-1]
  - Criar seg tree que permite somar 1 (frequencia) ou -1 a um intervalo e consultar o valor em uma posição.
  - Duas seg-trees: uma para a esquerda e uma para a direita
    - 10 → soma 1 ao intervalo [0,10]
    - 8 → soma 1 ao intervalo [0,8] . Para saber quantos >= 8 → query(8)
  - Desafio: podemos ter até 1M elementos, mas cada um pode ir até 1B !!! Precisaríamos de uma seg-tree com 1B índices! Ideias?

Segment Tree - exemplos de problemas

- Enemy is weak CodeForces - 61E
- Dado um vetor de números **distintos**, quantos índices i,j,k (i<j<k) possuem v[i] > v[j] > v[k] ?
- Exemplo: 10 8 3 1 → 4
- Solução trivial: O(n^3)
- Ideias?
- Dado um índice j, o que precisamos saber para calcular quantas triplas i,j,k existem? Exemplo: 10 **8** 3 1 e j = 1
  - Precisamos saber quantos elementos antes de j estao no intervalo [v[j]+1,INF]
  - E precisamos saber quantos elementos depois de j estao no intervalo [-INF, v[j]-1]
  - Criar seg tree que permite somar 1 (frequencia) ou -1 a um intervalo e consultar o valor em uma posição.
  - Duas seg-trees: uma para a esquerda e uma para a direita
    - 10 → soma 1 ao intervalo [0,10]
    - 8 → soma 1 ao intervalo [0,8] . Para saber quantos >= 8 → query(8)
  - Desafio: podemos ter até 1M elementos, mas cada um pode ir até 1B !!! Precisaríamos de uma seg-tree com 1B índices! Ideias?

Segment Tree - exemplos de problemas

- Ant colony CodeForces - 474F
- Entrada: array de números (ex: 1 3 2 4 2)
- Várias consultas em intervalos. Queremos saber quantos valores em um intervalo dividem todos os outros (na verdade, queremos R-L+1-isso).
- Exemplo: no intervalo [1,5] (**1 3 2 4 2**), temos 1 valor (1) que divide todos os outros.
- Exemplo: no intervalo [2,5] (1 **3 2 4 2**), temos 0 valor que divide todos os outros.
- Exemplo: no intervalo [3,5] (1 3 **2 4 2**), temos 2 valores (2 e 2) que dividem todos os outros.
- Como calcular isso? Alguma observação simples que possa ajudar?

Segment Tree - exemplos de problemas

- Ant colony CodeForces - 474F
- Entrada: array de números (ex: 1 3 2 4 2)
- Várias consultas em intervalos. Queremos saber quantos valores em um intervalo dividem todos os outros (na verdade, queremos R-L+1-isso).
- Exemplo: no intervalo [1,5] (**1 3 2 4 2**), temos 1 valor (1) que divide todos os outros.
- Exemplo: no intervalo [2,5] (1 **3 2 4 2**), temos 0 valor que divide todos os outros.
- Exemplo: no intervalo [3,5] (1 3 **2 4 2**), temos 2 valores (2 e 2) que dividem todos os outros.
- Como calcular isso?
- Alguma observação simples que possa ajudar (sempre faça brainstorm de observações simples! igual em PD)?
    - O valor que divide todos (se existir) será o menor deles!
    - Ele vai dividir todos os outros quando ….

Segment Tree - exemplos de problemas

- Ant colony CodeForces - 474F
- Entrada: array de números (ex: 1 3 2 4 2)
- Várias consultas em intervalos. Queremos saber quantos valores em um intervalo dividem todos os outros (na verdade, queremos R-L+1-isso).
- Exemplo: no intervalo [1,5] (**1 3 2 4 2**), temos 1 valor (1) que divide todos os outros.
- Exemplo: no intervalo [2,5] (1 **3 2 4 2**), temos 0 valor que divide todos os outros.
- Exemplo: no intervalo [3,5] (1 3 **2 4 2**), temos 2 valores (2 e 2) que dividem todos os outros.
- Como calcular isso?
- Alguma observação simples que possa ajudar (sempre faça brainstorm de observações simples! igual em PD)?
  - O valor que divide todos (se existir) será o menor deles!
  - Ele vai dividir todos os outros quando …. ele for igual ao GCD do intervalo!
  - Se for igual ao GCD → resposta será count(min)
  - Ou seja, preciamos de seg-tree que suporte:
    - GCD de intervalo
    - min de intervalo
    - countMin de intervalo

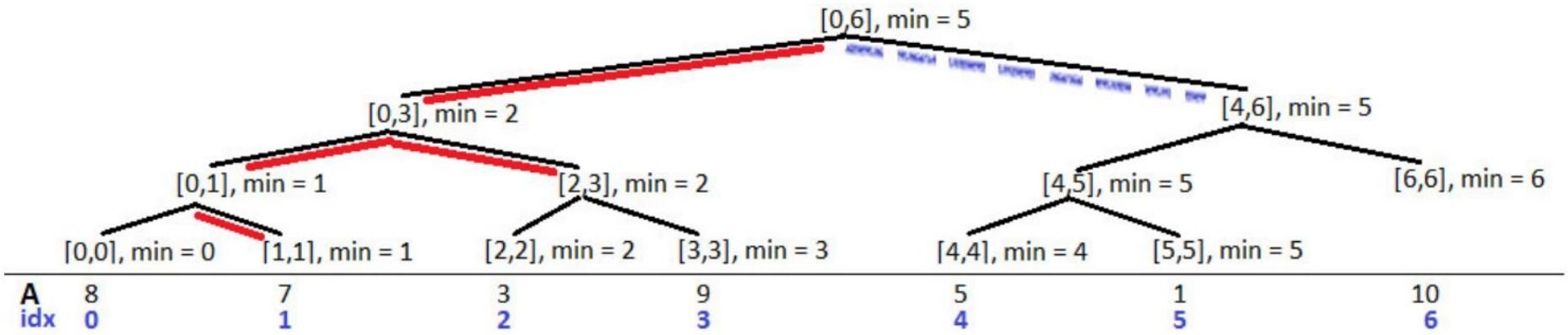
Segment Tree (slides extras baseados no CP2)

- Utilizada normalmente para RMQ (Range Minimum Queries)
- Permite consultas **dinâmicas**
- Exemplos:
  - Achar menor valor entre posições 2 e 5 de um array
  - Soma dos valores entre as posições 3 e 10 (fácil com prefix-sum, mas seg-tree → dinâmico)

Array A	Values =	8	7	3	9	5	1	10
	Indices =	0	1	2	3	4	5	6

Segment Tree

- Usa árvore binária (normalmente armazenada em um array)
- Raiz: resultado do intervalo todo
- RMQ(i,i) = i
- RMQ(i,j) = → percorrer árvore recursivamente, juntando resultados



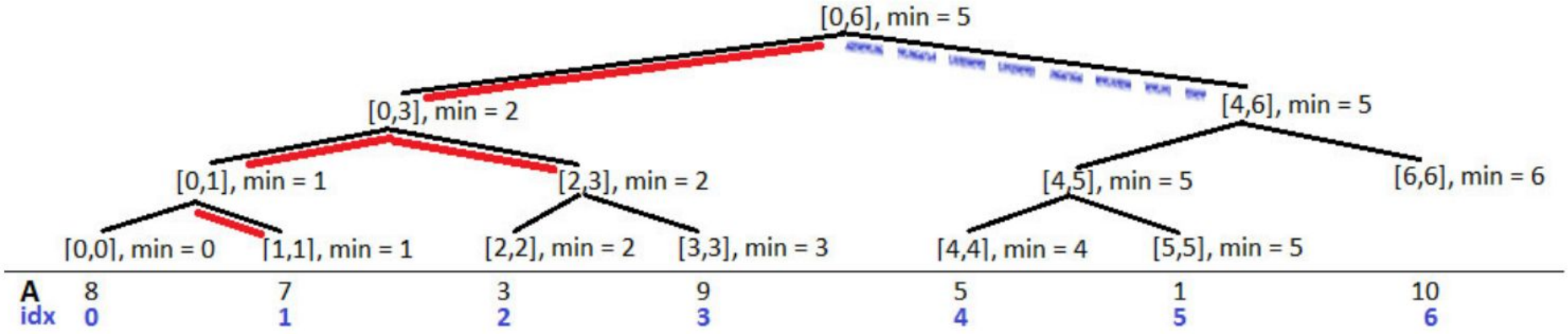

```
Array | Values  =  8 | 7 | 3 | 9 | 5 | 1 | 10
  A   | Indices =  0 | 1 | 2 | 3 | 4 | 5 | 6
```

Fonte: Competitive Programming 2

Segment Tree

menor entre posições [1,3], considerando busca no intervalo [0,6]

- Exemplo: RMQ([1,3], [0,6]) → busca no trecho vermelho
  - RMQ([1,3], [0,6]) → retorna RMQ([1,3], [0,3]) (resposta está à esquerda)
  - RMQ([1,3], [0,3]) → retorna melhor entre RMQ([1,3], [0,1]) e RMQ([1,3], [2,3])
    - RMQ([1,3], [2,3]) → retorna 2 ([2,3] está dentreo de [1,3])
    - RMQ([1,3], [0,1]) → retorna RMQ([1,3], [1,1]) = 1
  - Entre posição 1 (valor 7) e 2 (valor 3), o menor é 2 (valor 3)



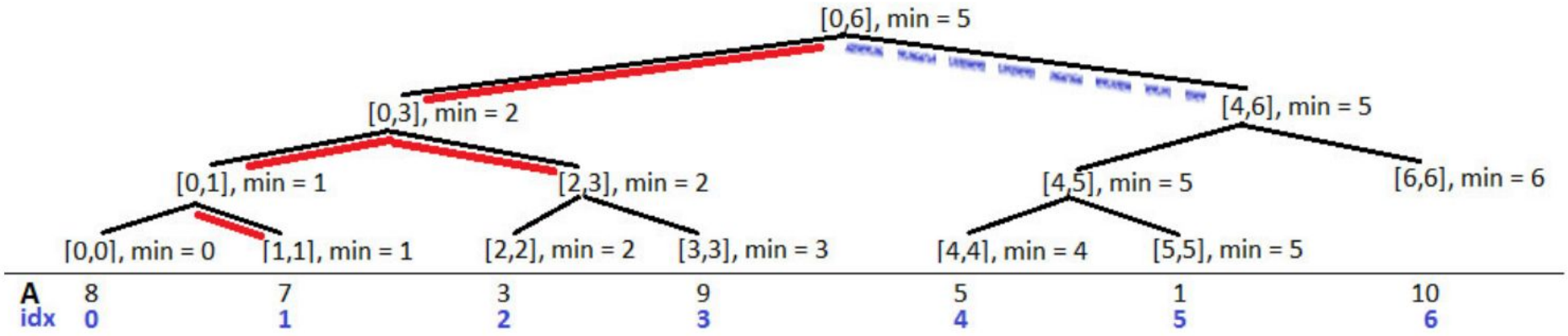

```
Array | Values  =  8 | 7 | 3 | 9 | 5 | 1 | 10
  A   | Indices =  0 | 1 | 2 | 3 | 4 | 5 | 6
```

Fonte: Competitive Programming 2

Segment Tree

menor entre posições [4,6], considerando busca no intervalo [0,6]

- Exemplo: RMQ([4,6], [0,6]) → busca no trecho azul
  - RMQ([4,6], [0,6]) → retorna RMQ([4,6], [4,6]) (direita)
  - RMQ([4,6], [4,6]) = 5
  - Note que processamos apenas o mínimo necessário da árvore.

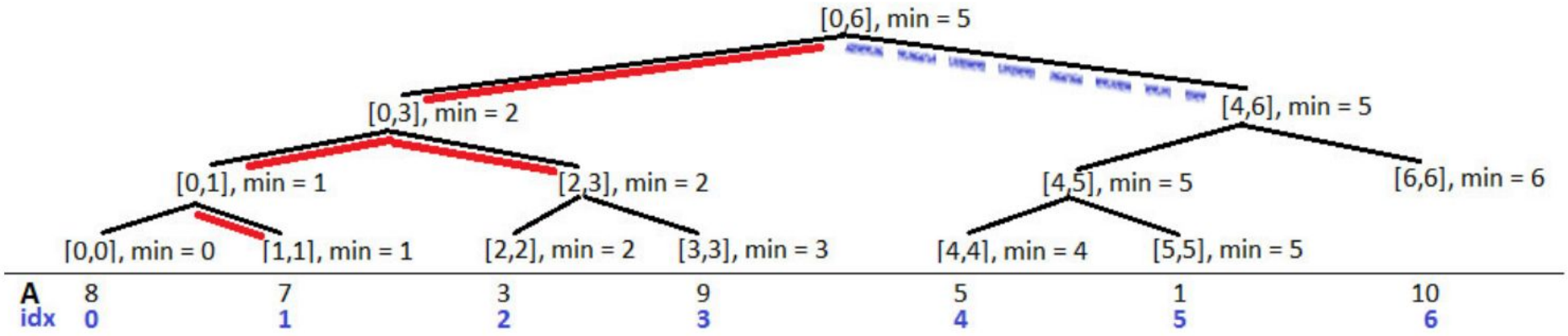


Array	Values =	8	7	3	9	5	1	10
A	Indices =	0	1	2	3	4	5	6

Fonte: Competitive Programming 2

Segment Tree

- Pior caso: 2 caminhos da raiz até folhas = O(2 log n) = O(log n)
- Exemplo de pior caso:

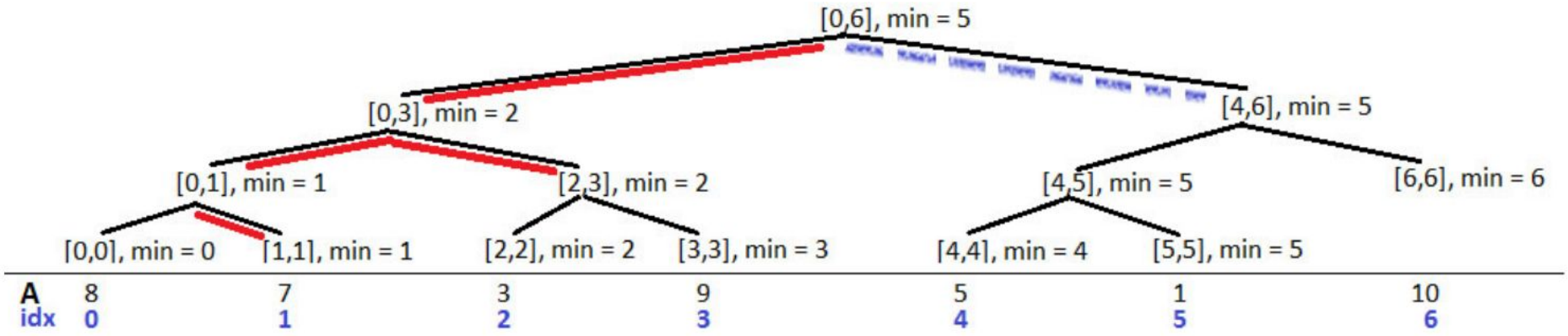


Array	Values =	8	7	3	9	5	1	10
A	Indices =	0	1	2	3	4	5	6

Fonte: Competitive Programming 2

Segment Tree

- Pior caso: 2 caminhos da raiz até folhas = O(2 log n) = O(log n)
- Exemplo de pior caso: RMQ([1,4],[0,6]), RMQ([3,4],[0,6])

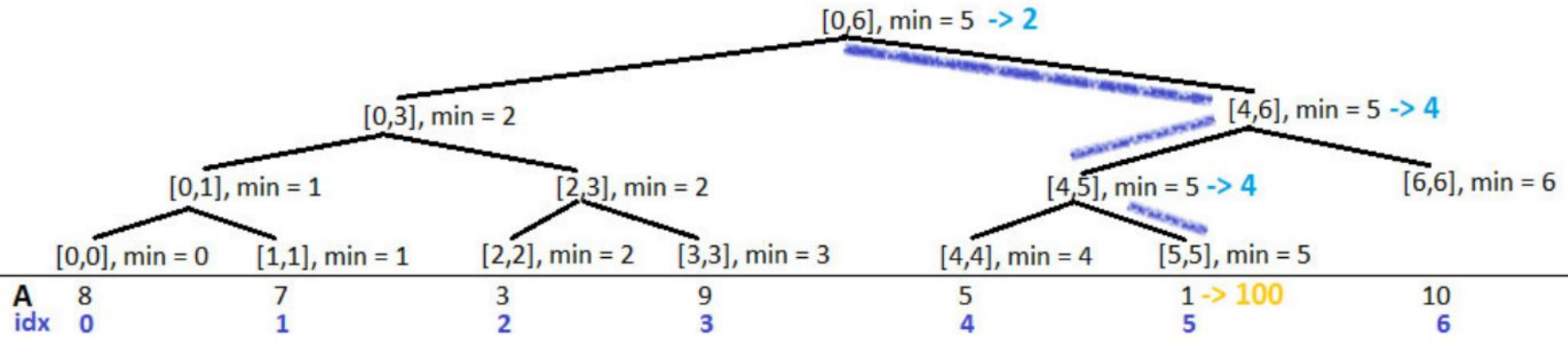



```
Array | Values  =  8 | 7 | 3 | 9 | 5 | 1 | 10
  A   | Indices =  0 | 1 | 2 | 3 | 4 | 5 | 6
```

Fonte: Competitive Programming 2

Segment Tree

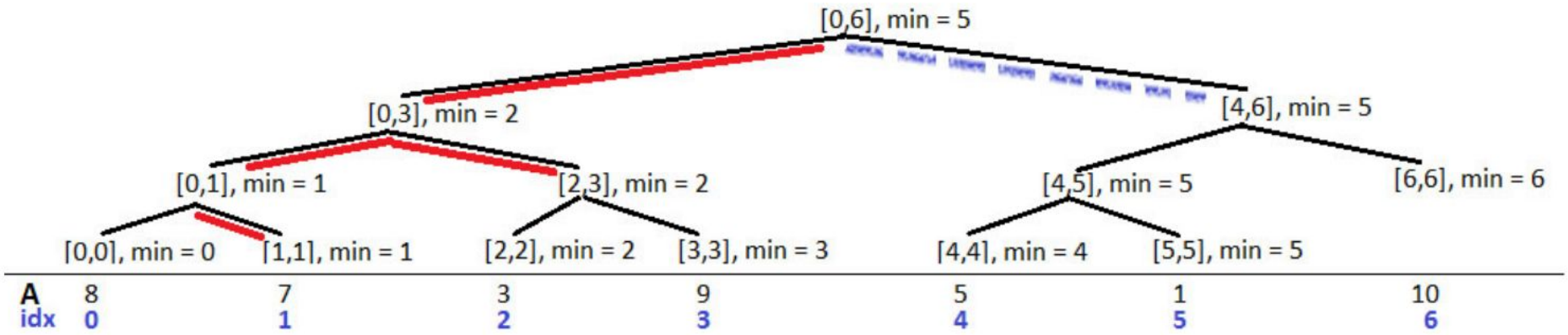
- Pior caso: 2 caminhos da raiz até folhas = O(2 log n) = O(log n)
- Exemplo de pior caso: RMQ([1,4],[0,6]), RMQ([3,4],[0,6])
- Principal utilidade: valores do array mudam frequentemente
  - Fácil atualizar seg-tree
  - Apenas segmentos no caminho até o elemento precisam ser atualizados.



Fonte: Competitive Programming 2

Segment Tree - código

- Cria seg-tree e guarda no vetor t
- Tamanho: número de elementos em árvore binária com N folhas
  - Exemplo: 4 folhas → 7 elementos (cálculo estranho d
- Ver código daqui: https://www.geeksforgeeks.org/segment-tree-set-2-range-maximum-query-node-update/?ref=rp (nesse exemplo, retorna máximo)
- Notem que ele retorna o valor máximo (pode ser adaptado facilmente para retornar a posição contendo o valor máximo).
- Raiz armazenada na posição 0 do array (da árvore)
- Filho esquerdo na posição 1, direito 2, …
  - [5,2,5,1,2,5,6,0,1,2,3,4,5,lixo, lixo] → similar a heaps…



Fonte: Competitive Programming 2