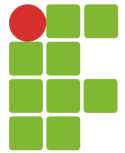
Lazy propagation

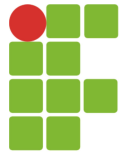
Tópicos Especiais em Algoritmos

Prof. Daniel Saad Nogueira Nunes

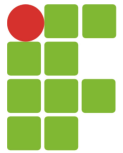
IFB – Instituto Federal de Brasília,
Campus Taguatinga



Sumário

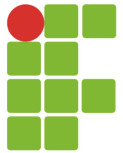


Sumário



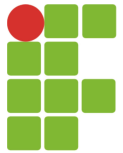
Árvores de segmentos

- Vimos as **árvores de segmento**, estruturas extremamente flexíveis e capaz de responder diversas consultas sobre intervalos de uma sequência.
- Tempo $\langle \Theta(n), \Theta(\lg n), \Theta(\lg n) \rangle$.
- Construção em tempo $\Theta(n)$.
- Consulta em tempo $\Theta(n)$.
- Atualização em tempo $\Theta(n)$.



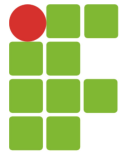
Árvores de segmentos

- Como visto, as árvores de segmentos aceitam a atualização de um valor $V[i]$ da sequência de entrada e isso se reflete em $\Theta(\lg n)$ nós da árvore, justificando o tempo de atualização obtido.
- E se quiséssemos atualizar um intervalo inteiro $V[i, j]$ da entrada?
- Fazer $j - i + 1$ operações de atualização individuais é muito custoso: tempo $\Theta(\lg n \cdot (j - i + 1))$.
- Proporcional ao tamanho do intervalos multiplicado por $\Theta(\lg n)$.
- Temos como fazer melhor?

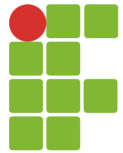


Lazy propagation

- Com a técnica de **lazy propagation** conseguimos fazer atualização sobre intervalos em tempo $\Theta(\lg n)$ utilizando memória adicional.
- O vetor adicional armazena a atualização dos valores sobre o intervalo.
- Os filhos de um nó só são atualizados quando realmente necessário. Ou seja, a propagação da atualização de um nó pai para os nós filhos é atrasada ao máximo.



Sumário

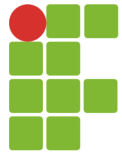


Lazy propagation

Problema

Dado uma sequência de inteiros $V[0, n-1]$, considere o problema de:

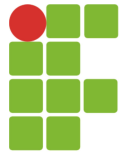
- Somar os elementos $V[i, j]$.
- Adicionar um valor de k para os elementos $V[i, j]$.



Lazy propagation

Ideia da solução

- A ideia é utilizar um vetor adicional lz, para cada nó da árvor de segmentos.
- Para um nó u da árvore de segmentos, $\mathrm{lz}[u]$ indica se há atualizações pendentes e o valor dessas atualizações.
- Quando necessário, $\mathrm{lz}[u]$ é propagado para os filhos de u, v, e w. Logo, atualizamos $\mathrm{lz}[v]$ e $\mathrm{lz}[w]$.

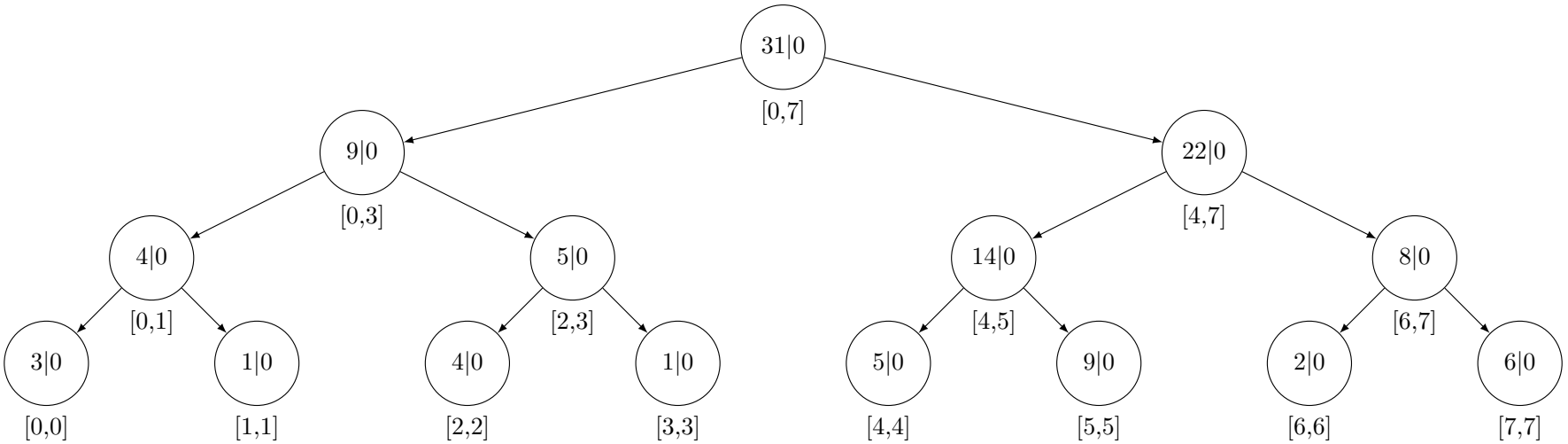


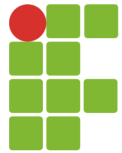
Lazy propagation

Exemplo

- Considere a árvore de segmentos para o vetor de entrada $V[0, n-1] = (3, 1, 4, 1, 5, 9, 2, 8)$.
- Cada nó u que representa um intervalo $[l, r]$, conterá a soma de $V[l, r]$.
- Inicialmente, $\mathrm{lz}[u] = 0$ para todos os nós, indicando que não há propagações pendentes.

Lazy propagation

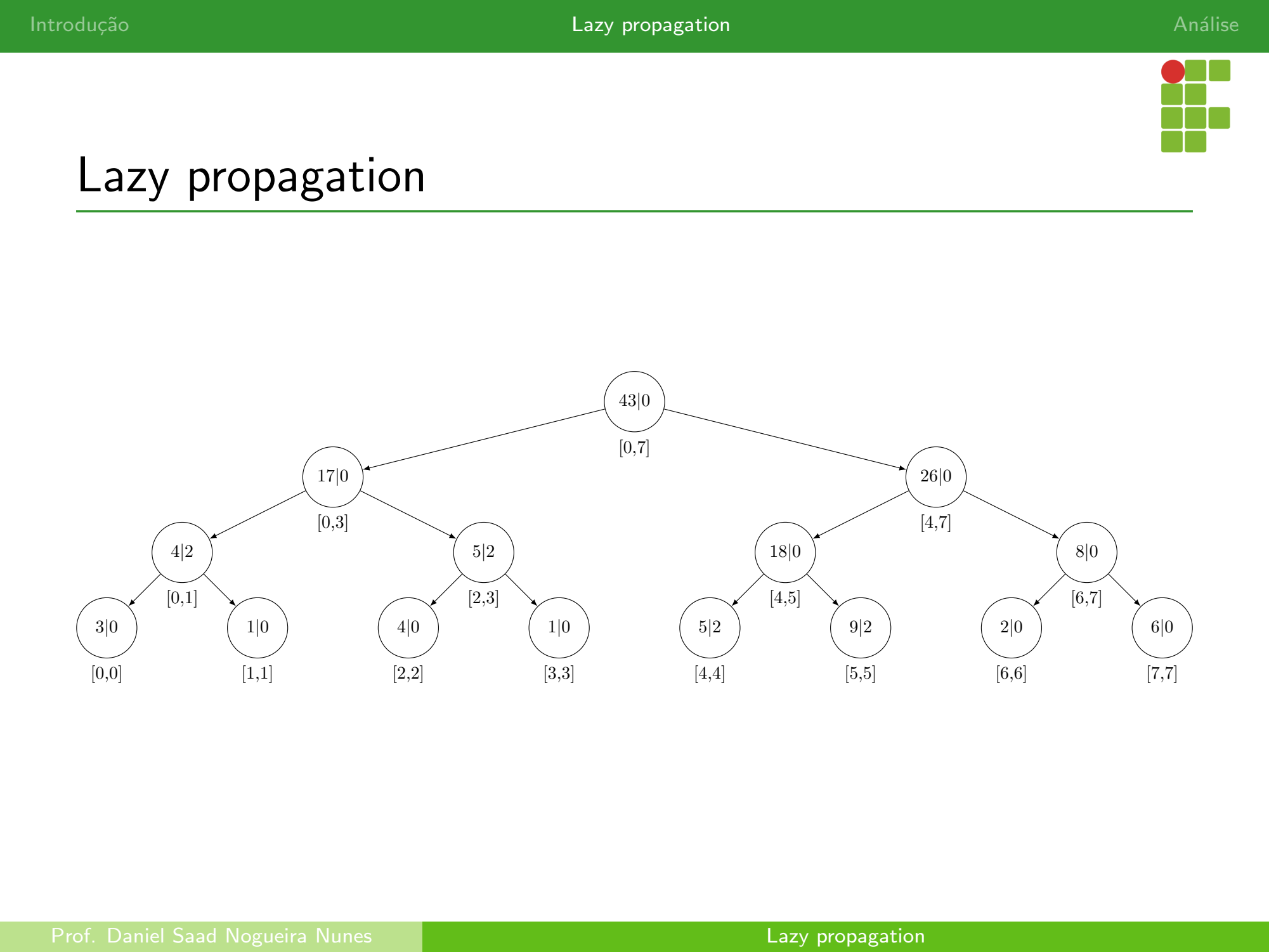




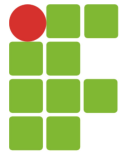
Lazy propagation

Exemplo

- Suponha que queremos adicionar o valor 2 a todos os elementos no intervalo $[0, 5]$.
- Adicionaremos 8 ao nó $[0, 3]$ e 4 ao nó $[4, 5]$.
- Além disso, marcaremos os nós $[0, 1]$ $[2, 3]$ e $[4, 4]$ e $[4, 5]$ com a propagação pendente, isto é, com valor 2.
- Claro que os ancestrais dos nós modificados também são atualizados.



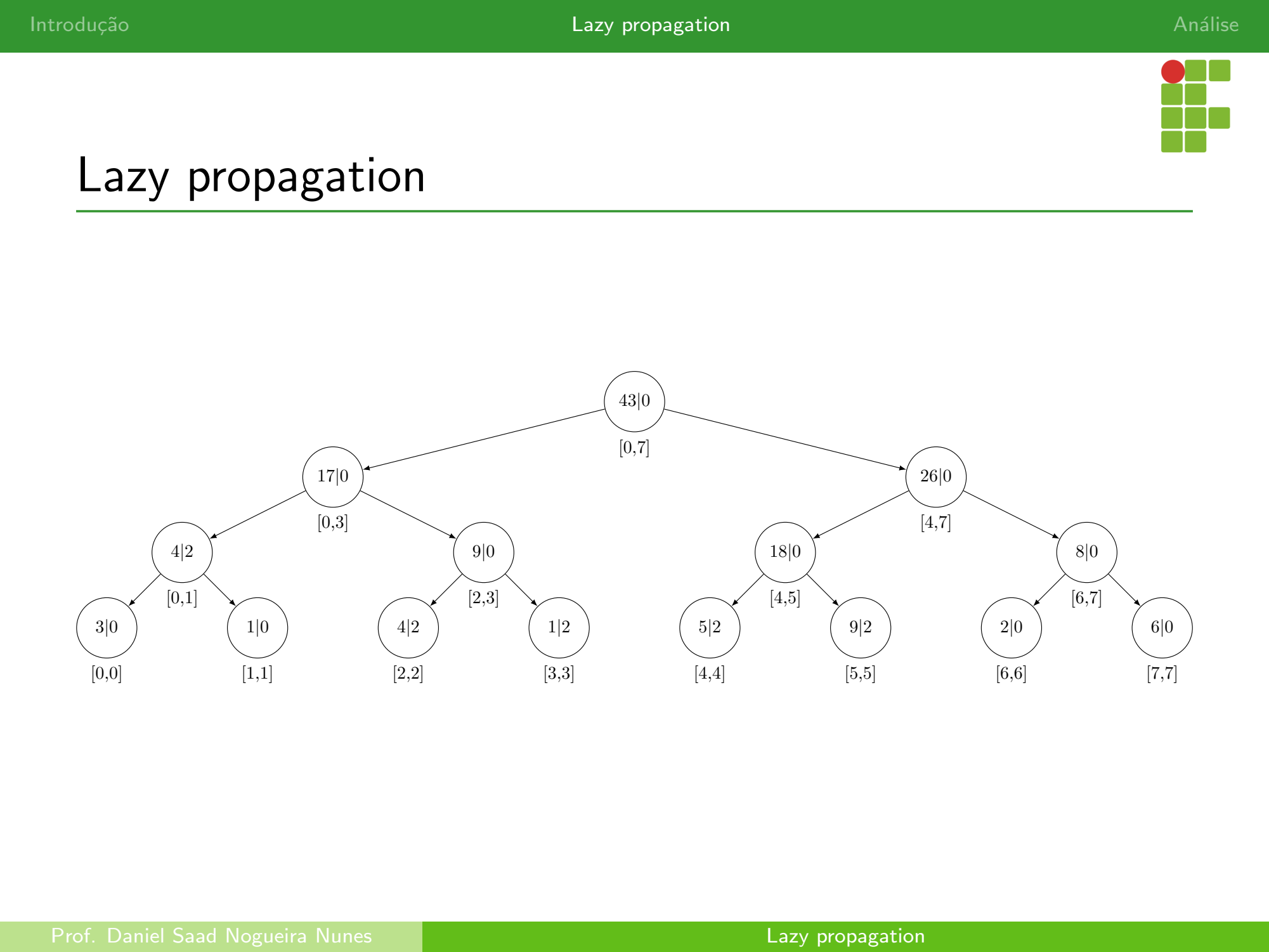
Lazy propagation



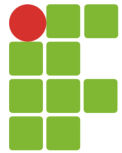
Lazy propagation

Exemplo

- Agora suponha que queremos saber qual o valor da soma sob o intervalo $[2, 5]$.
- Note que o nó $[4, 5]$ está preenchido com 18 e seu valor lz é 0, logo podemos pegar o valor diretamente do nó, sem a necessidade de propagar qualquer atualização para os seus filhos.
- Contudo o nó $[2, 3]$ possui um valor a ser propagado a seus filhos. Primeiro atualizamos o nó $[2, 3]$ e, em seguida, propagamos o valor de lz do nó aos seus filhos. Finalmente, marcamos o nó $[2, 3]$ sem qualquer pendência de propagação.
- Como o nó $[2, 3]$ tem tamanho 2, e o valor de atualização armazenado em lz é 2, então precisamos adicionar 4 a este nó.



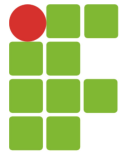
Lazy propagation



Lazy propagation

Exemplo

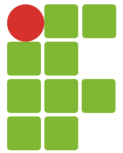
- Assim, o valor da soma sobre o intervalo $[2, 5]$ é 27.



Lazy propagation

Estratégia para propagação de um nó:

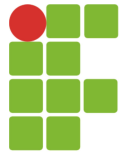
1. Se o nó não possui valor a ser propagado, o valor pode ser retirado diretamente dele.
2. Caso contrário, atualize o nó e propague a atualização para os seus dois filhos (caso possua).



Lazy propagation

Algorithm 1: PROPAGATE(T, lz, u, l, r)

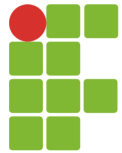
1 **if**($\text{lz}[u] = 0$) **return**
2 $T[u] \leftarrow T[u] + (l - r + 1) \cdot \text{lz}[u]$
3 **if**($l < r$)
4 $\quad \text{lz}[\text{LEFT}(u)] \leftarrow lz[\text{LEFT}(u)] + \text{lz}[u]$
5 $\quad \text{lz}[\text{RIGHT}(u)] \leftarrow \text{lz}[\text{RIGHT}(u)] + \text{lz}[u]$
6 $\text{lz}[u] \leftarrow 0$



Lazy propagation

Algorithm 2: $\textsc{sum}(T, \text{lz}, u, l, r, i, j)$

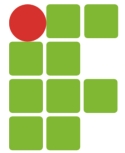
```
1 PROPAGATE(T, lz, u, l, r)
2 if( i > r ∨ j < l )
3     return 0
4 if( l ≥ i ∧ r ≤ j )
5     return T[u]
6 mid ← ⌊(l + r)/2⌋
7 ans1 ← SUM(T, lz, LEFT(u), l, mid, i, j)
8 ans2 ← SUM(T, lz, RIGHT(u), mid + 1, r, i, j)
9 return ans1 + ans2
```



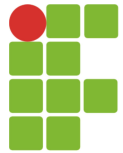
Lazy propagation

Algorithm 3: UPDATE$(T, \text{lz}, u, l, r, i, j, k)$

```
1 PROPAGATE(T, lz, u, l, r)
2 if( i > r ∨ j < l )
3     return
4 if( l ≥ i ∧ r ≤ j )
5     lz[u] ← k
6     PROPAGATE(T, lz, u, l, r)
7 mid ← ⌊(l + r)/2⌋
8 UPDATE(T, lz, LEFT(u), l, mid, i, j, k)
9 UPDATE(T, lz, RIGHT(u), mid + 1, r, i, j, k)
```



Sumário



Análise

- Com a técnica de propagação preguiçosa, conseguimos manter a atualização em tempo $\Theta(\lg n)$ para o problema da soma sobre intervalos, visto que o procedimento PROPAGATE leva tempo constante.
- Podemos conservar esse tempo para diversos outros problemas envolvendo árvores de segmentos.