UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Universidade Tecnológica Federal do Paraná
Departamento Acadêmico de Computação

Segment Trees

Rafael R Soratto[1], Michel Gomes [1], João M. Lopes [1]
Orientador: Prof. Dr. André Kawamoto[1]

[1] Universidade Tecnológica Federal do Paraná (UTFPR) - Brasil

22 de maio de 2023

Sumário

Este trabalho foi realizado para a disciplina de Estrutura de Dados (PPGCC02) do programa de mestrado PPGCC de Campo Mourão - Paraná - Brasil. O professor André Kawamoto solicitou que cada equipe fizesse uma descrição completa sobre diferentes estruturas de dados relevantes para programação competitiva.

Segment Tree:

- As estruturas de dados são essenciais para desenvolver algoritmos eficientes [1];
- Uma dessas estruturas é a **árvore de segmentos**, que é muito útil para **consultas** eficientes e atualizações **flexíveis** em intervalos de elementos.
- Vamos explorar os conceitos básicos das segment trees, implementações e aplicações.

[1] Gupta; Canuto

RMQ, SOMA, Multiplicação, Divisão, Maior Elemento

- São exemplos de consultas em um intervalo;
- Se você fizer da forma convencional terá que comparar todos elementos com todos (custo N^2)
- Se utilizar a estrutura de segmentos a consulta tem custo $\log N$

Por exemplo, dado um vetor

$$a = [1, 3, -2, 8, -7]$$

que pode ser representado pela Figura é possível saber com precisão e velocidade a informação de soma total dos elementos de determinado segmento, ou então o maior e o menor número daquele sub-vetor.

Figura: Visualização do vetor $a = [1, 3, -2, 8, -7]$

e a operação de soma do segmento principal.

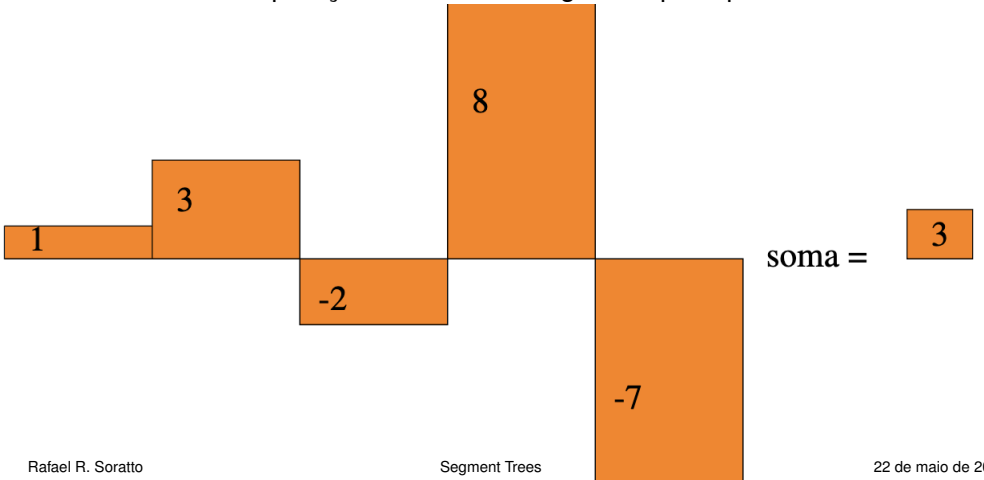
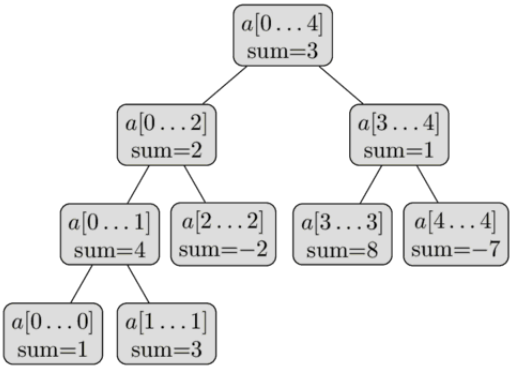


Figura: Árvore de Segmentos do vetor a utilizando a operação de **soma**.



Vantagens:

A árvore de segmentos é uma estrutura de dados que armazena informações sobre intervalos de um vetor em uma árvore. Suas principais vantagens são:

1. **Busca eficiente** de intervalos;
2. **Flexibilidade** nas modificações do vetor.

A definição formal da árvore de segmentos é a seguinte:
Dado um vetor $a[0 \ldots n-1]$, a árvore de segmentos permite realizar diversas operações simples em tempo $O(\log n)$, tais como:

- Calcular a soma dos elementos entre os índices l e r ($\sum_{i=l}^{r} a[i]$);
- Atualizar os valores dos elementos no vetor ($a[i] = x$).

Durante cada interação para construir a árvore calcula-se e armazena-se uma operação (por exemplo: soma, mínimo e máximo) para cada nível da árvore.

- Árvore de segmentos é uma **árvore binária completa** e **balanceada**;
  - todos os níveis da árvore estão completamente preenchidos, exceto possivelmente o último nível.
- Cada nó interno da árvore representa um **intervalo contíguo de elementos do vetor** e possui um **valor de agregação** desses elementos;
  - e cada nó folha representa um único elemento do vetor.
  - Nas folhas, o valor de agregação é o próprio elemento.
- Consultas em intervalos pode ser realizadas em **tempo** $O(\log n)$
- Requer apenas uma **quantidade linear de memória**:
  - A árvore de segmentos padrão requer $4n$ nós para trabalhar em um vetor de tamanho n

Por que precisa de uma quantidade linear de memória?

A representação da estrutura requer um **número linear de vértices**.

- um vetor de tamanho $a[0 \dots n-1]$ é separado em duas metades $a[0 \dots n/2-1]$ e $a[n/2 \dots n-1]$ e calculamos e armazenamos a soma de cada metade.
- Cada uma dessas duas metades, por sua vez, é dividida ao meio recursivamente até que todos os segmentos atinjam o tamanho 1.

- A maior parte das implementações não são construídas explicitamente. No pior caso, o **número de vértices** é estimado pela soma:

$$1 + 2 + 4 + \cdots + 2^{\lceil \log_2 n \rceil} \leq 2^{\lceil \log_2 n \rceil + 1} \leq 4n$$

.

- Sempre que n não for uma potência de dois, nem todos os níveis da Árvore de Segmentos serão totalmente preenchidos. Isto interfere diretamente na implementação.

Para construir cada segmento da árvore é utilizada uma função $F(x) = y$ onde:

1. $F(x)$ é a operação a ser realizada e consequentemente o valor que será armazenado em cada segmento: a soma dos elementos do segmento, o menor/maior número do segmento, entre outros tipos. No exemplo, armazena-se a soma dos valores do intervalo $f(x)$ realizada em um intervalo $[0, n]$.
2. E y é o resultado dessa função, por exemplo, a soma dos elementos do segmento ou o menor número.

- Para construção da árvore é necessário começar pelas folhas e aplicando a função de merge $F(x)$ para armazenar o valor de todos segmentos até a raíz.
- O procedimento de construção, se chamado em um vértice não folha, faz o seguinte:
  1. construir recursivamente os valores dos dois vértices filhos;
  2. mesclar os valores calculados desses filhos.

O tempo de construção da árvore é de $O(n)$

Assumindo que o tempo da operação de merge é constante (chamada N vezes). Inicia-se a construção no vértice raiz, portanto, ele pode armazenar o resultado das operações em toda a árvore de segmentos.

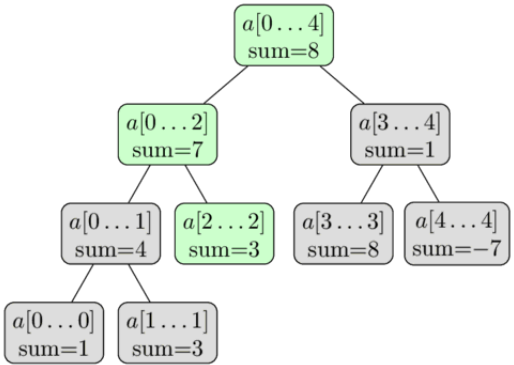
A complexidade da Consulta é $O(\log n)$

- Recebe dois inteiros l e r para calcular a soma dos elementos $a[l \dots r]$ em tempo $O(\log n)$. **A complexidade da consulta** é $O(\log n)$ por causa dos **níveis da árvore**.
- Para cada nível, visitamos não mais do que quatro nós.
- Altura da árvore é $O(\log n)$, obtém-se o tempo de execução desejado.
- Ou seja, visitamos no máximo $4 \log n$ vértices no total, e isso é igual a um tempo de execução de $O(\log n)$.

O **número de vértices** que precisão ser **atualizados** é $O(\log n)$.

- Para alterar um elemento e reconstruir a árvore basta considerar que cada nível de uma Árvore de Segmentos forma uma partição do vetor;
- Portanto, um elemento $a[i]$ contribui apenas para um segmento de cada nível.

Figura: Atualização do elemento $a[2] = 3$



Fonte: CP-Algorithms, *Segment Tree*.

Figura: Função Recursiva Principal

```
void buildSegmentTreeUtil(SegmentTree st, int node, int start, int end, int op) {
    if (start == end) {
        st.tree[node] = st.arr[start];
    } else {
        int mid = getMid(start, end);
        buildSegmentTreeUtil(st, 2 * node + 1, start, mid, op);
        buildSegmentTreeUtil(st, 2 * node + 2, mid + 1, end, op);
        st.tree[node] = do_operation(st.tree[2 * node + 1], st.tree[2 * node + 2], op)

    }
}
```

Fonte: Autoria Própria (2023).

- Implementamos uma versão bem completa da Segment Tree com 5 consultas diferentes (mínimo, máximo, soma, multiplicação e divisão);
- O Algoritmo é recursivo conforme mostrado anteriormente;
- Colocamos um *Clock* para monitorar o tempo de execução da estrutura;

Figura: Exemplo da operação soma e do vetor resultante

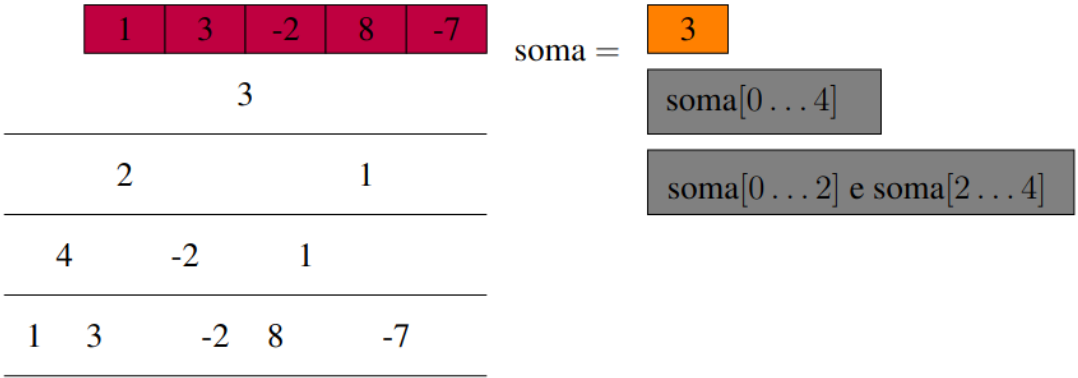


Figura: Exemplo da operação soma e do vetor resultante

```
* Default input *

Vetor de entrada:
1       2       3       4       5

Operacao: Soma

Soma do intervalo [0, 1]: 3
Soma do intervalo [0, 2]: 6
Soma do intervalo [3, 4]: 9
Soma do intervalo [0, 4]: 15
Arvore final:
15      6       9       3       3       4       5       1       2

Tempo de execucao: 0.000036 segundos
```

Geometria Computacional e Processamento de Imagens

- Eficaz na resolução de diversos problemas geométricos, tais como a remoção eficiente de ruídos em áreas com baixa textura, bem como o realce das bordas dos objetos [2].
- também esta associada ao mapeamento denso de disparidades, sendo útil na determinação de profundidades de cena [3], como ilustrado na Figura 7 e na Figura 8;

[2] Mei et al.; Six e Wood; Azali, Hamzah e Noh
[3] Scharstein, Szeliski e Zabih

Exemplo Aplicação

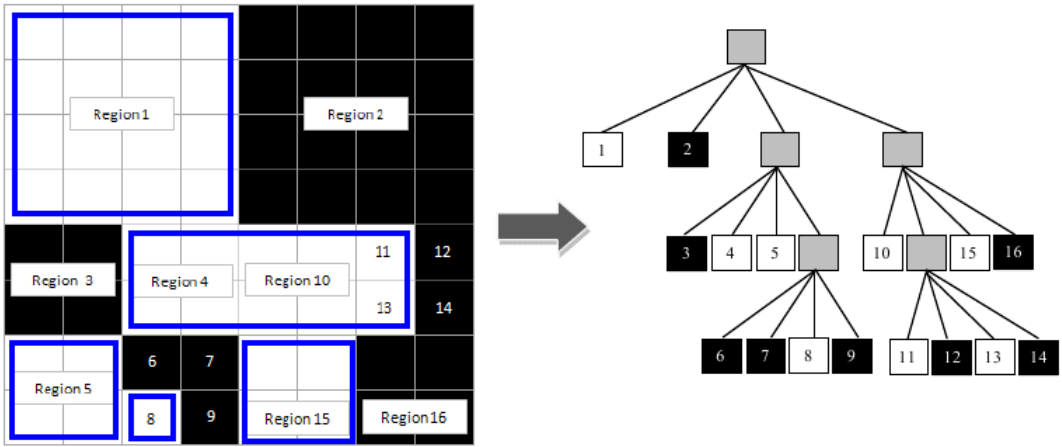


Figura: Uma quadtree, um tipo de Segment Tree, usada para mapeamento de imagens.

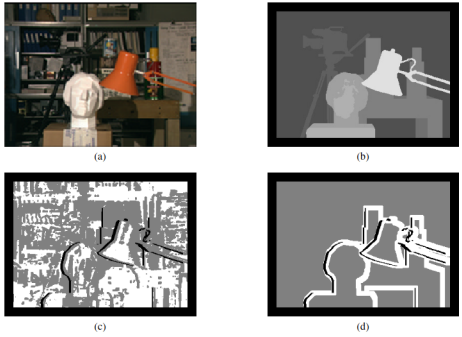


Figura: Exemplo de uma imagem segmentada: (a) imagem original, (b) disparidades verdadeiras, (c) regiões sem textura (branco) e regiões ocultas (preto), (d) regiões com descontinuidade de profundidade (branco) e regiões ocultas (preto).

- Roteamento de pacotes e endereçamento IP.
- No artigo mencionado[4], essa estrutura é utilizada para armazenar os campos de endereço de destino dos pacotes de rede;
- A Segment Tree é construída com base nas informações dos prefixos de endereços IP, onde cada prefixo é representado por um segmento com dois pontos finais.
- Os nós internos da árvore correspondem a intervalos que são uniões dos intervalos elementares associados às folhas.
- Essa abordagem garante eficiência tanto na classificação quanto no armazenamento dos filtros aplicados a cada intervalo de endereço IP.

[4] Su, “High-speed packet classification using segment tree”

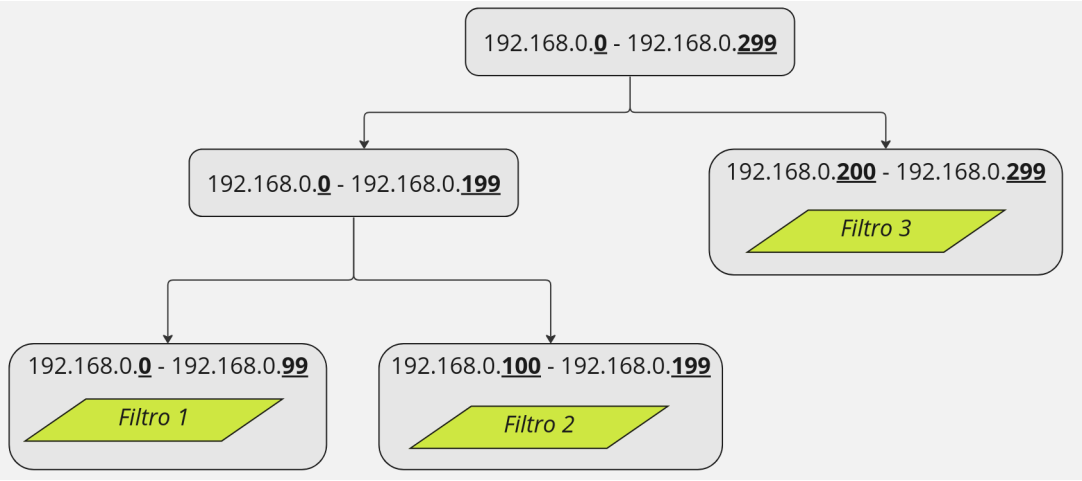


Figura: Exemplo de aplicação de filtro por segmentos de Ips utilizando Segment Tree, nesse caso por sufixo

Azali, M.N.Z., R.A. Hamzah e Z.M. Noh. “Disparity map algorithm using census transform and hierarchical segment-tree from stereo image”. Em: *Engineering Technology International Conference (ETIC 2022)*. Vol. 2022. 2022, pp. 244–249. DOI: 10.1049/icp.2022.2620.

Canuto, Fabricio. “Estruturas de Dados: Árvore de Segmentos”. Em: *Blog Código Fluente* (2021).

CP-Algorithms. *Segment Tree*. Accessed 2023. URL: https://cp-algorithms.com/data_structures/segment_tree.html#structure-of-the-segment-tree.

Gupta, Arpit. “Segment Trees: A Data Structure for Range Queries”. Em: *Medium* (2020).

Mei, Xing et al. “Segment-Tree Based Cost Aggregation for Stereo Matching”. Em: *2013 IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 313–320. DOI: 10.1109/CVPR.2013.47.

Scharstein, D., R. Szeliski e R. Zabih. “A taxonomy and evaluation of dense two-frame stereo correspondence algorithms”. Em: *Proceedings IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV 2001)*. 2001, pp. 131–140. DOI: 10.1109/SMBV.2001.988771.

Six e Wood. “Counting and Reporting Intersections of d-Ranges”. Em: *IEEE Transactions on Computers* C-31.3 (1982), pp. 181–187. DOI: 10.1109/TC.1982.1675973.

Su, Ching-Fong. “High-speed packet classification using segment tree”. Em: *Globecom ’00 - IEEE. Global Telecommunications Conference. Conference Record (Cat. No.00CH37137)*. Vol. 1. 2000, 582–586 vol.1. DOI: 10.1109/GLOCOM.2000.892083.

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Universidade Tecnológica Federal do Paraná
Departamento Acadêmico de Computação

Segment Trees

Rafael R Soratto[1], Michel Gomes [1], João M. Lopes [1]
Orientador: Prof. Dr. André Kawamoto[1]

[1] Universidade Tecnológica Federal do Paraná (UTFPR) - Brasil

22 de maio de 2023