Árvores de segmentos

Matheus de Mello Santos Oliveira **Orientador: Carlos Eduardo Ferreira**

Departamento de Ciência da Computação, Instituto de Matemática e Estatística, Universidade de São Paulo

A estrutura

A utilização da estrutura de dados conhecida como árvores de segmentos é bastante frequente em competições de programação. Este trabalho apresenta a teoria básica da estrutura e suas implementações assim como suas aplicações, extensões e ênfase na resolução de problemas de programação competitiva. Para melhor esclarecer, demostraremos que o tipo de problema que gostaríamos de resolver é: dado um conjunto armazenado em uma estrutura de acesso aleatório aos seus índices (como um vetor) e uma operação associativa, desejamos responder eficientemente o resultado desta operação aplicada aos elementos de um intervalo desse conjunto múltiplas vezes.

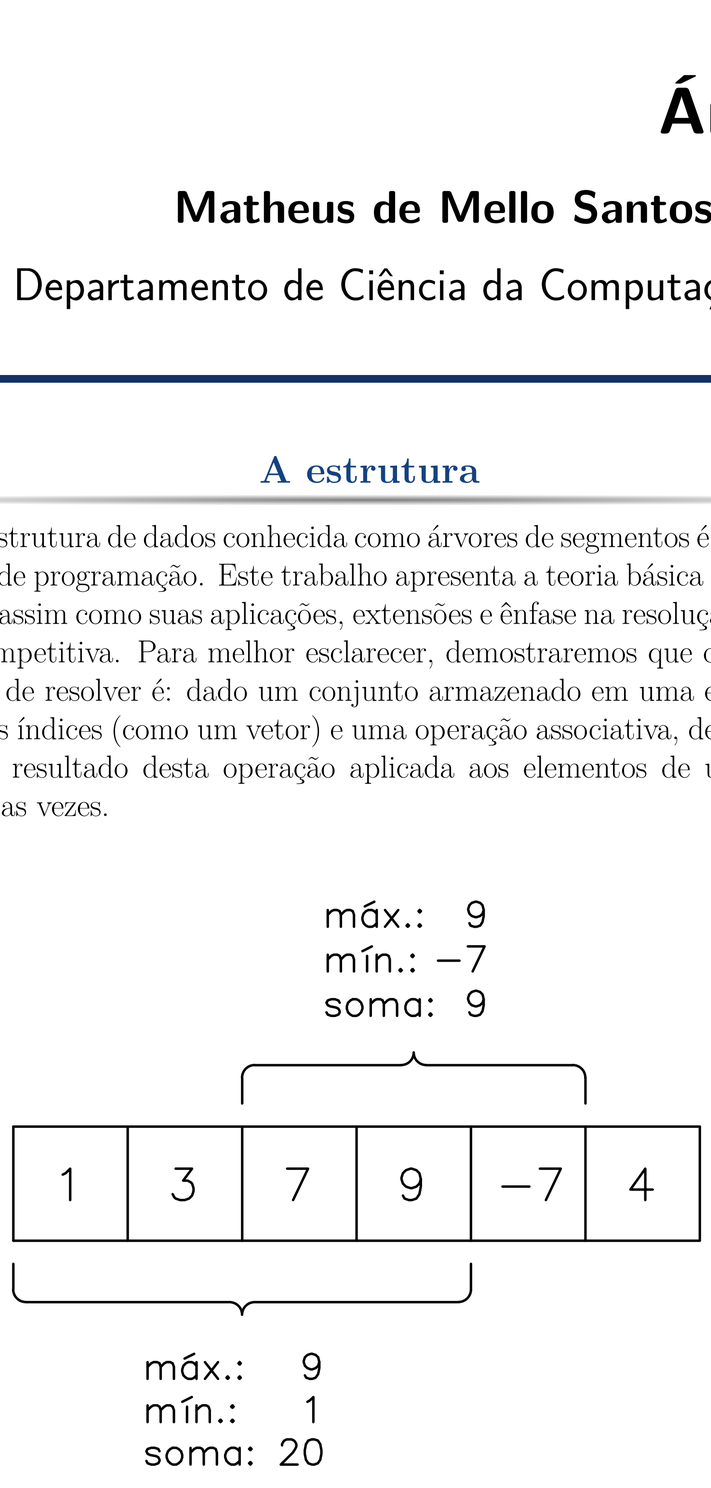

Figura 1: Exemplo de estrutura, com operações aplicadas em intervalos.

Características

Para qualquer árvore de segmento as seguintes características seguem verdadeiras:

- É uma árvore binária, ou seja, cada nó possui dois filhos;
- Cada nó representa um intervalo do vetor fundamental e este nó pode conter informações sobre mais do que uma função aplicada em intervalos do vetor fundamental;
- Cada folha da árvore está associada a um intervalo que contém apenas um elemento;
- Subindo na árvore, cada nó pai representa a união dos resultados das funções aplicadas nos intervalos (disjuntos) representados por seus dois filhos;
- O nó raiz representa o vetor inteiro;

Para ilustrar segue uma figura de uma possível árvore de segmento relativa a um vetor fundamental de tamanho 5.

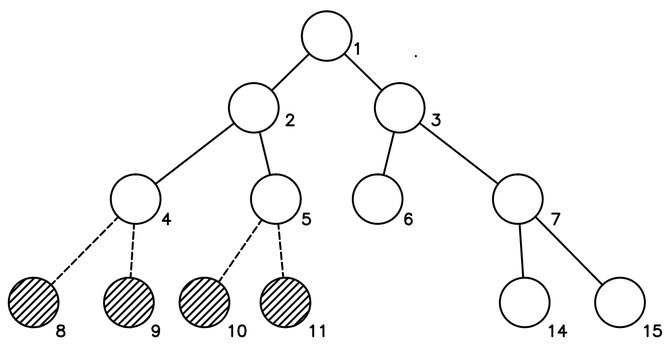

Figura 2: Árvore de segmento relativa a um vetor fundamental de tamanho 5.

Divisão e conquista

A ideia da solução utilizando o paradigma de divisão e conquista conserva-se no seguinte raciocínio:

- Se o intervalo que estamos analisando possui apenas um elemento então este elemento é trivialmente a resposta para esse intervalo;
- Caso contrário, dividimos o intervalo em dois subintervalos de aproximadamente mesmo tamanho, e aplicamos a operação associativa desejada nos dois subintervalos;

Ou seja:

$$a_i \text{ se } x = y$$
$$\text{ou}$$
$$operação(f(x, \lfloor \tfrac{x+y}{2} \rfloor)), f(\lfloor \tfrac{x+y}{2} \rfloor + 1, y) \text{ caso contrario}$$

Aplicações desta estrutura não se limitam a vetores de números inteiros e, frequentemente, são utilizados para resolver problemas em áreas como geometria computacional onde um ponto pode guardar muitas informações além de sua posição no plano.

Exemplo ilustrativo

Em uma cidade distante, o Secretário de Educação, preocupado com alguns índices requereu que fosse feito um levantamento sobre o indivíduo mais novo entre diversos intervalos da sociedade. Deveremos considerar ainda que haverá alterações na lista de idade dos habitantes do município considerando-se a saída de velhos moradores e a chegada de novos integrantes. Assim, tendo a lista de idade de todos os habitantes do município, nossa estrutura precisa retornar a informação do indivíduo mais novo, quando perguntado inúmeras vezes sobre diversos intervalos.
Vamos supor a seguinte lista de idades:

$$(5, 3, 11, 7, 1)$$

A ideia do algorítimo é iniciar pelo nó que representará a árvore em sua totalidade e chamar recursivamente para as metades de, aproximadamente, mesmo tamanho, sendo o caso base quando só há um sub elemento na sub árvore para qual a função está sendo chamada.

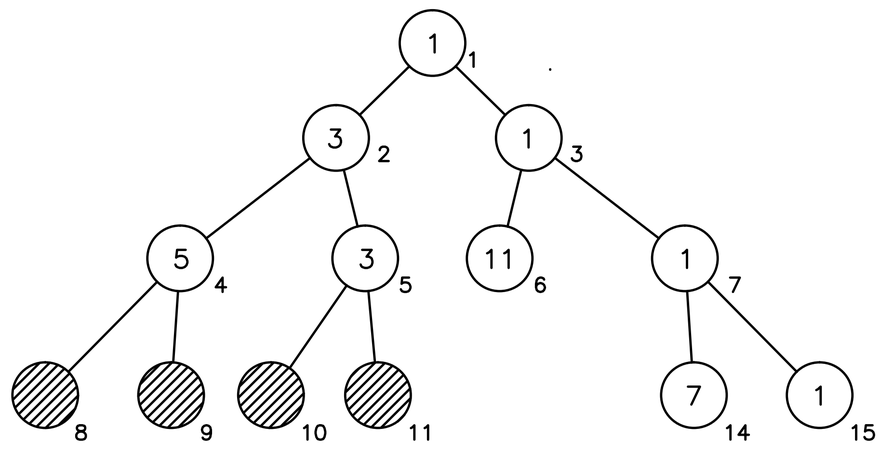

Figura 3: Representação dessa árvore.

Suponhamos que gostaríamos de atualizar a 4ª casa da lista em questão, de 1 para 17:
Lista atualizada

$$(5, 3, 11, 7, 17)$$

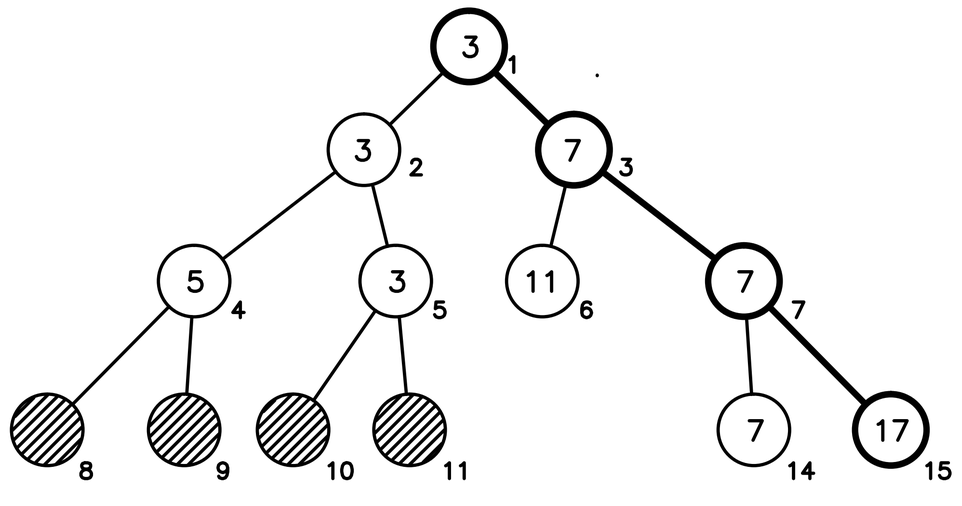

Figura 4: Árvore atualizada.

Propagação preguiçosa

Uma das formas de adaptação da árvore de segmentos, de forma a otimizar a utilização da ferramenta e maximizar sua aplicação é a Lazy Propagation ou propagação preguiçosa. Como o próprio nome sugere, a propagação preguiçosa é a forma de propagar a atualização de um determinado intervalo na árvore apenas quando for necessária sua utilização sem que seja necessário caminhar por todos os nós da árvore fazendo com que o gasto de tempo do algorítimo seja menor. A base da propagação preguiçosa se dá através da marcação do nó na árvore que deve ser alterado, uma vez que ambas as árvores desfrutam do mesmo índice e, quando existe a necessidade de consultar os dados de um segmento, verifica-se na árvore auxiliar se há marcação em algum nó; havendo a marcação, faz-se atualização dos nós filhos até a profundidade necessária para a consulta, zerando-se o nó da árvore auxiliar e marcando a atualização apenas no nó que esteve pendente de atualização.

Mais informações

`https://linux.ime.usp.br/~matheusmso/mac0499`