Tópico 9: Árvores 2-3 e Árvores Rubro-Negras

Prof. Dr. Juliano Henrique Foleis

Estude com atenção os vídeos e as leituras sugeridas abaixo. Os exercícios servem para ajudar na fixação do conteúdo e foram escolhidos para complementar o material básico apresentado nos vídeos e nas leituras. Quando o exercício pede que crie ou modifique algum algoritmo, sugiro que implemente-o em linguagem C para ver funcionando na prática.

Vídeos

Árvores 2-3

Árvores Rubro-Negras

Leitura Sugerida

FEOFILOFF, Paulo. Estruturas de Dados. *Árvores 2-3* (Link)

FEOFILOFF, Paulo. Estruturas de Dados. *BSTs rubro-negras* (Link)

Exercícios dos materiais de leitura sugerida

Exercícios 1.1, 2.1, 2.2, 2.3, 3.1, 3.2 da página do Prof. Feofiloff (Árvores 2-3): (Link)

Exercícios 1.1, 1.2, 1.3, 1.4, 1.5, 1.7, 3.1, 3.2, 3.3, 3.4 da página do Prof. Feofiloff (Árvores Rubro-Negras): (Link)

Exercícios

1. Implemente as operações a seguir em uma árvore 2-3 em C++:

- Inserção
- Busca
- Altura

As chaves e os valores correspondentes podem ser números inteiros.

DICA: Crie 2 classes para implementar os nós da árvore, uma para os nós simples e outra para os nós duplos. Façam que ambas implementem a classe abstrata *No23*, que conterá as operações (métodos) comuns a ambos tipos de nós.

DICA 2: A implementação da árvore 2-3 em C é bem trabalhosa. **Faça se tiver tempo!**

2. Baixe a implementação criada e sala de aula no link. Implemente os métodos a seguir na classe `ARN`:

a. Implemente o método `ARN::tamanho()` que devolve o número de nós na árvore. Você pode criar um método privado auxiliar, conforme fizemos para implementar a inserção.

b. Existem duas medidas de altura para uma árvore rubro-negra. A primeira é a medida tradicional de altura, que é o número máximo de arestas percorridas a partir da raíz para atingir algum nó folha da árvore. Implemente o método `ARN::altura()` tanto de forma ansiosa quanto de forma preguiçosa.

c. A outra medida de altura de uma árvore rubro-negra é a altura negra. Neste caso, é o número de arestas negras percorridas a partir da raíz para atingir qualquer nó folha da árvore. Implemente o método `ARN::alturaNegra()` tanto de forma ansiosa quanto de forma preguiçosa.

d. O custo médio de uma busca bem-sucedida considerando que todos os nós são igualmente prováveis de serem buscados pode ser estimada pela expressão a seguir:

$$\left\lceil \frac{1}{N} \sum_{i=1}^{N} C[i] \right\rceil$$

tal que $C[i]$ é a profundidade do nó i na árvore, N é o número de nós da árvore e $\lceil\ \rceil$ é a função teto. Implemente a função `ARN::custoBuscaBemSucedida()`.

e. O custo médio de uma busca mal-sucedida considerando que todas as possíveis chaves tem a mesma probabilidade de serem buscadas pode ser estimada pela expressão a seguir:

$$\left\lceil \frac{1}{L} \sum_{i=1}^{L} (C[i] + 1) \right\rceil$$

tal que $C[i]$ é a profundidade do nó folha i na árvore, L é o número de nós folhas da árvore e $\lceil\ \rceil$ é a função teto. Implemente a função `ARN::custoBuscaMalSucedida()`.

f. Implemente o método `ARN::percentualNosVermelhos()` que retorne a percentagem de *links* rubros de uma ARN. Para N = { 10000, 100000, 1000000 } insira N chaves aleatórias em uma árvore inicialmente vazia e anote a percentagem de links rubros da árvore resultante. Para cada N, repita o experimento 30 vezes. Calcule a média e o desvio padrão e anote em uma tabela.

g. Implemente os métodos `ARN::minimo()`, `ARN::maximo()`, `ARN::antecessor()` e `ARN::sucessor()`.

BONS ESTUDOS!