MC202 — Estruturas de Dados
Lista de Exercícios 2
Árvores Rubro-Negras

Primeiro semestre de 2017 - Turmas B e C
Professor: Emilio Francesquini
francesquini@ic.unicamp.br

Legenda:
⋆ – Fácil
⋆⋆ – Médio
⋆ ⋆ ⋆ – Difícil

1. ⋆ Qual a altura mínima de uma árvore binária de busca rubro-negra que contém N nós?

2. ⋆ [Feofiloff] Dê um exemplo de uma árvore binária de busca cujas folhas têm todas a mesma profundidade, mas nem todo caminho da raiz até um link null tem o mesmo número de links negros.

3. ⋆ [Feofiloff] Seja x um nó de uma árvore binária de busca rubro-negra. Mostre que todos os caminho que levam de um nó x até um NULL têm o mesmo número de links pretos.

4. ⋆⋆ [Feofiloff] Qual a menor e maior altura de preto que uma árvore binária de busca rubro-negra com N nós pode ter?

5. ⋆ [CLRS] Qual é o maior número possível de nós internos em uma árvore rubro-negra que possuem altura de preto igual a k? Qual é o menor número possível?

6. ⋆ Desenhe uma árvore rubro-negra que contém dois caminhos da raiz até as folhas tal que um caminho seja o dobro do tamanho do outro.

7. ⋆⋆ [Feofiloff] Suponha que x é um nó de uma árvore binária de busca rubro-negra. Suponha que x.dir == NULL mas x.esq != NULL. Prove que x.esq.cor == Vermelho e x.esq não tem filhos (ou seja, x.esq.esq == NULL e x.esq.right == NULL).

8. ⋆ [Feofiloff] Escreva um código que calcula a altura (total) e a altura de preto de uma árvore binária de busca rubro-negra.

9. ⋆⋆ [Feofiloff] Qual o número mínimo e o número máximo de links rubros numa árvore binária de busca rubro negra com N nós? Qual o número máximo?

10. ⋆ ⋆ ⋆ [CLRS] Teorema fundamental das rotações. Mostre que qualquer árvore binária de busca pode ser transformada em qualquer outra sobre o mesmo conjunto de chaves por uma sequência apropriada de rotações. (Dica: mostre primeiramente que são necessárias $n-1$ rotações para transformar qualquer árvore binária de busca em uma "lista ligada à direita")

11. ⋆⋆⋆ [SW 3.3.22] Encontre uma sequência de chaves para inserir em uma árvore binária de busca comum e uma árvore binária de busca rubro-negra de modo que a altura da árvore binária de busca comum seja menor que a altura (total) da árvore binária de busca rubro-negra, ou mostre que uma tal sequência não existe.

12. ⋆⋆ [CLRS] Vamos definir uma nova árvore binária de busca rubro-negra-relaxada como uma árvore binária de busca que satisfaz as Regras 1, 3, 4 e 5. Em outras palavras, permitiremos que a raíz assuma qualquer cor, vermelha ou preta. Considere que uma árvore rubro-negra-relaxada tenha a raíz vermelha. Bastaria colorirmos a raíz de preto para transformar essa árvore em uma árvore rubro-negra?

13. ⋆ [SW 3.3.10] Desenhe a sequência de árvores binárias de busca rubro-negras que resulta da inserção das chaves E A S Y Q U T I O N , nesta ordem, em uma árvore inicialmente vazia. (Desenhe apenas a árvore binária de busca no final de cada inserção; não desenhe as árvores binárias de busca temporárias que aparecem durante a operação.)

14. ⋆ [SW 3.3.11] Desenhe a sequência de árvores binárias de busca rubro-negras que resulta da inserção das chaves Y L P M X H C R A E S , nesta ordem, em uma árvore inicialmente vazia.

15. ⋆ [SW 3.3.17] Gere duas árvores binárias de busca rubro-negras aleatórias com 16 nós cada. Desenhe-as (à mão ou usando um programa). Compare-as com as árvores binárias de busca comuns (não balanceadas) construídas com as mesmas chaves.

16. ⋆⋆ [CLRS] Considere uma árvore binária de busca rubro-negra criada pela inserção de n nós usando a função que vimos em aula. Mostre que se $n > 1$, então a árvore possui ao menos um nó vermelho.

17. ⋆⋆ [CLRS] Imagine que a estrutura criada para armazenar os nós da nossa árvore rubro-negra não possua um campo para armazenar o pai de cada nó. Modifique a função `NoArvRN *insereArvRN(NoArvRN *raiz, NoArvRN *novo)` vista em aula para se adaptar a esta restrição.

18. ⋆ [CLRS] Mostre que depois de executar a função `consertaRemocaoArvRN` (vista em aula) a raíz da árvore DEVE ser preta.

19. ⋆ [CLRS] Mostre que se na chamada da função `removeArvRN` (vista em aula) ambos `x` e `x->pai` são vermelhos, então a propriedade 4 pode ser restaurada por uma chamada à função `consertaRemocaoArvRN`.

20. ⋆ [SW 3.3.37] As árvores lembram de sua história. Mostre que árvores binárias de busca rubro-negras lembram de sua história. Por exemplo, se você inserir uma chave que é menor que todas as chaves da árvore e logo em seguida remover o mínimo da árvore, você pode obter uma árvore diferente da original.