AED1 - Lista 6
Árvores binárias, árvores binárias de busca, heaps

Seguem alguns exercícios relacionados com árvores binárias, árvores binárias de busca e heaps.

1 - [14.1.3] - Sejam X e Z dois nós de uma árvore binária. Mostre que existe no máximo um caminho com origem X e término Z.

2 - [14.2.3] - Imprima as folhas de uma árvore binária em ordem e-r-d.

3 - [14.2.5] - Modifique o algoritmo iterativo com pilha que realiza percurso e-r-d em uma árvore binária, para que: (i) ele faça a varredura r-e-d de uma árvore binária, (ii) ele faça a varredura e-d-r de uma árvore binária.

4 - [14.1.5] - [Expressões Aritméticas] Árvores binárias podem ser usadas, de maneira muito natural, para representar expressões aritméticas (como ((a+b)*c-d)/(e-f)+g, por exemplo). Discuta os detalhes desta representação.

5 - [14.2.7] - [Expressões Aritméticas] Considere a árvore binária usada para representar expressões aritméticas do exercício anterior. Discuta a relação entre os percursos e-r-d e e-d-r desta árvore e as notações infixa e posfixa.

6 - [14.3.1] - Desenhe uma árvore binária com 17 nós que tenha a menor altura possível.

7 - [14.3.2] - Escreva uma função iterativa que calcule a altura de uma árvore binária.

8 - [14.4.1] - Escreva uma função que preencha corretamente todos os campos pai de uma árvore binária.

9 - [14.4.4] - A profundidade de um nó em uma árvore binária é a distância entre o nó e a raiz da árvore. Escreva uma função que imprima o conteúdo de cada nó de uma árvore binária precedido de um recuo em relação à margem esquerda do papel. Esse recuo deve ser proporcional à profundidade do nó. Exemplo:

```
        555                555
      /      \                333
   333      888                 111
  /     \                       444
111   444                     888
```

10 - [15.1.1] - Escreva uma função que decida se uma dada árvore binária é ou não é de busca.

11 - [15.2.2] - Modifique o algoritmo visto em aula para calcular o predecessor de um nó em uma árvore binária de busca, para que ele não precise usar o campo pai.

12 - [15.2.3] - Escreva uma função que transforme um vetor crescente em uma árvore de busca balanceada.

13 - [15.2.4] - Escreva uma função que transforme uma árvore de busca em um vetor crescente.

14 - [15.4.3] - Modifique o algoritmo visto em aula para remover um nó de uma árvore binária de busca, para que ele não precise usar o campo pai.

15 - [10.1.1] - Mostre que todo vetor decrescente é um max-heap. Mostre que a recíproca não é verdadeira.

16 - [10.1.3] - Escreva uma função que decida se um vetor v[0 .. m - 1] é ou não um max-heap.

17 - [14.4.5] - Em que condições uma árvore binária pode ser considerada um heap? Escreva uma função que transforme um max-heap em uma árvore binária quase completa. Escreva uma versão da função desceHeap para um max-heap representado por uma árvore binária.

18 - [10.1.6] - Suponha que v[0 .. 2^k - 2] é um max-heap. Mostre que mais da metade dos elementos do vetor está na última “camada” do max-heap, ou seja, em v[2^(k - 1) - 1 .. 2^k - 2].

19 - [10.5.1] - [Ordem Decrescente] Escreva uma versão do algoritmo heapSort que rearranja um vetor v[0 .. n - 1] de modo que ele fique em ordem decrescente.

20 - [10.5.2] - [Ordenação de Strings] Escreva uma versão do algoritmo heapSort que coloque um vetor de strings em ordem lexicográfica (“ordem alfabética”). Veja https://www.ime.usp.br/~pf/algoritmos/aulas/string.html.

Para revisar conceitos sobre árvore binárias, árvores binárias de busca e heaps, além de encontrar mais exercícios, acesse:

- https://www.ime.usp.br/~pf/algoritmos/aulas/bint.html
- https://www.ime.usp.br/~pf/algoritmos/aulas/binst.html
- https://www.ime.usp.br/~pf/algoritmos/aulas/hpsrt.html