CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DO RN

GERÊNCIA DE TECNOLOGIA DA INFORMAÇÃO E EDUCACIONAL DE TELEMÁTICA

Exercícios de Estrutura de Dados I

1. Implemente uma árvore binária de pesquisa que armazene números inteiros em seus nós. Considere que as chaves são os próprios elementos (números inteiros).

2. Insira, em uma árvore binária de pesquisa, inicialmente vazia, itens com as seguintes chaves(inserir necessariamente nesta ordem):

   $$\{30, 40, 24, 58, 48, 26, 11, 13\}$$

   Desenhe a árvore após cada inserção.

3. Escreva um algoritmo para remover itens de uma árvore binária de pesquisa. Utilizando seu algoritmo, remova os seguintes itens, nesta ordem, da árvore resultante do exercício anterior: $\{26, 11, 24, 30, 40\}$. Desenhe a árvore após cada remoção.

4. Mostre que para inserções do mesmo conjunto de chaves em uma árvore binária de pesquisa, se a sequência de inserção for diferente, as árvore resultantes podem ser diferentes.

5. A utilização de árvores binárias para armazenar itens tem como objetivo otimizar as operações de `inserção`, `remoção` e `busca` do TAD Dicionário, obtendo $O(\log n)$ para estas operações. Istro é sempre verdade? justifique com exemplos.

6. Dada uma árvire binária de pesquisa ABP, escreva um algoritmo que mostre o maior e o menor elemento da árvore. Qual a complexidade do algoritmo?

7. Quando removemos um nó v de uma árvore binária de pesquisa, podemos substituir v por nó da sub-árvore da esquerda (se tiver). Determine uma escolha ótima para o nó que deve substituir v, considerando que está árvore é implementada através de um *array*.

8. Sobre a sequência a seguir:

   $$\{2, 5, 16, 4, 10, 23, 39, 18, 26, 15)\}$$

   (a) Desenhe a árvore de execução do algoritmo *merge-sort*

   (b) Desenhe a árvore de execução do algoritmo *quick-sort*, onde o pivô é sempre o último elemento da sequência

   (c) Desenhe a árvore de execução do algoritmo *quick-sort*, onde o pivô é sempre o elemento do meio da sequência

   (d) Escreva um texto comparando o tempo de execução dos três algoritmos.

9. Desenhe o conteúdo da árvore de execução do algoritmo *quick-sort* onde a ordenação é feita no próprio *array*. Para facilitar desenhe o conteúdo de l e r em cada chamado a *quick-sort*.

   O algoritmo já está implementado no livro texto nas páginas 432,433 e 434 – trechos de código 10.6 e 10.7

10. Implemente os algoritmos *merge-sort*, *quick-sort* determinístico (o pivô é sempre o último elemento),e *quick-sort* randômico. Aplique estes algoritmo aos casos de teste dos exercícios de ordenação anterior e preencha a tabela abaixo:

Arquivo	*Heap-sort*	*merge-sort*	*quick-sort* det.	*quick-sort* rand.
teste_1000_1.dat				
teste_1000_2.dat				
teste_1000_3.dat				
teste_10000_1.dat				
teste_10000_2.dat				
teste_10000_3.dat				
teste_100000_1.dat				
teste_100000_2.dat				
teste_100000_3.dat				

    Escreva um texto explicando, em linhas gerais, o tempo de execução de cada algoritmo.

11. Qual versão do *quick-sort* você indicaria para ordenar sequencias "quase" ordenadas, a versão determinística (com o pivô sendo o último) ou a versão randômica? por que?

12. Suponha uma seqência com n elementos, onde cada elemento possui uma cor: azul ou vermelho. Projeto um algoritmo "ordene" os elementos baseado na cor, onde os elementos azuis ficam a esquerda e os vermelhos ficam a direita. Este algoritmo deve executar em tempo liner.