Lista de exercícios – Estruturas de Dados / Organização de Dados I
Prof. Vinícius Gusmão

1) Se 100 elementos forem inseridos, um por um, em uma lista linear implementada com alocação sequencial com tamanho inicial igual a 4, qual o total de operações de escrita em memória...

   a) ...se for utilizada uma política de expansão de array que cria um novo array com tamanho *duas unidades* maior do que o tamanho do array anterior em caso de overflow?
   b) ...se for utilizada uma política de expansão de array que cria um novo array com tamanho *duas vezes* maior do que o tamanho do array anterior em caso de overflow?

2) Simule a execução de uma busca binária pelo elemento 82 no array abaixo, indicando as posições do array que foram lidas, a cada passo. Qual o total de leituras no array?

   2 5 7 10 25 27 28 33 50 52 60 61 80 88 99

3) Explique com suas palavras como se dá a remoção de um elemento de uma lista duplamente encadeada com *n* elementos, sabendo-se que esses elementos são todos números inteiros distintos...

   a) ...no caso de o parâmetro da função de remoção ser o valor do elemento a ser removido (isto é, um inteiro);
   b) ...no caso de o parâmetro da função de remoção ser um ponteiro para o nó da lista que se pretende remover.

   Quais as complexidades das remoções nos dois casos acima?

4) Que elementos estarão numa pilha, e em que ordem, após a seguinte sequência de operações sobre uma pilha inicialmente vazia?

   adicionar 6
   adicionar 7
   adicionar 8
   adicionar 9
   remover elemento
   adicionar 13
   adicionar 15
   remover elemento

remover elemento
remover elemento

5) E se a estrutura utilizada no exercício anterior fosse uma fila?

6) Sejam as chaves e frequências (relativas) de acesso abaixo.

Chaves:	27	34	88	5	14
Frequências:	2	3	4	3	1

   a) Desenhe uma árvore binária de busca de altura máxima e calcule o custo médio de acesso a uma chave presente na árvore.
   b) Desenhe uma árvore binária de busca de altura mínima e calcule o custo médio de acesso a uma chave presente na árvore.

7) Queremos organizar uma festa em uma empresa cujos empregados se organizam hierarquicamente na forma de uma árvore binária. A raiz dessa árvore é o presidente da empresa, que é o empregado identificado pelo número 1. Suponha que cada empregado possua um "coeficiente de diversão", que é um número indicando o quanto ele contribuirá para a diversão geral caso ele seja convidado para a festa. No entanto, ninguém na festa se divertirá caso um empregado qualquer e seu chefe imediato sejam ambos convidados para a festa. O coeficiente de diversão da festa como um todo é a soma dos coeficientes de diversão de cada convidado. Seu objetivo é descobrir qual o maior valor possível para o coeficiente de diversão de uma festa naquela empresa.

   a) Escreva em pseudo-código, ou na linguagem que você quiser, um programa para encontrar o coeficiente de diversão máximo de uma festa naquela empresa. *Dica: é bem natural escrevê-lo utilizando chamadas recursivas, de forma que o problema original seja quebrado em sub-problemas menores. Considere, por exemplo, que "festa_com_raiz(x)" (respectivamente, "festa_sem_raiz(x)") é uma função que calcula o coeficiente de diversão máximo de uma festa considerando-se apenas os empregados que pertecencem à sub-árvore de raiz x e supondo-se que x seja convidado (respectivamente, não seja convidado) para a festa. Utilizando essas funções, o coeficiente de diversão máximo para a sub-árvore de raiz x pode ser dado por uma função festa(x) que retorna o máximo entre festa_com_raiz(x) e festa_sem_raiz(x).*

   b) Pense um pouco: um mesmo subproblema poderá precisar ser resolvido mais de uma vez, ao longo da execução do seu programa?
   c) Adapte o seu programa de forma a usar memoização e acabar com o problema de se resolver a mesma coisa várias vezes. Qual a complexidade de seu programa agora, em função do número *n* de empregados da empresa?