Universidade Estadual de Maringá
Departamento de Informática
Disciplina 1207 – Algoritmos e Estruturas de Dados I – Turma 31 - 2007
Profa. Josiane

TRABALHO 4º BIMESTRE

Instruções gerais:

1) Os exercícios abaixo devem ser implementados em linguagem Pascal;
2) Somente os arquivos .pas deverão ser entregues. Coloquem nomes que representem o exercício nos arquivos. Exemplo: exe01.pas ou goldbach.pas
3) Serão avaliados:
    a. A corretude do programa em relação ao que foi pedido no exercício;
    b. A colocação em prática dos conceitos que foram discutidos em sala de aula de forma correta;
    c. A **eficiência** do algoritmo implementado;
    d. O capricho da implementação e a endentação;
    e. A estruturação do programa em módulos e a nomeação das variáveis, funções e procedimentos;
    f. A forma de interação com o usuário.
4) Todos os exercícios devem ser compactados juntos (zipados) e enviados por e-mail para josianempf@gmail.com, com o subject "Trab4AED" (sem espaços), até as 23:59hs do dia 21/11/2007. Coloque seu nome e R.A. como nome do arquivo campactado. Exemplo: maria42536.zip (favor utilizar esta forma de compactação);
5) ****Nos dias 26, 28 e 30/11/2007, estes trabalhos devem ser apresentados à professora, individualmente, por seus respectivos autores.****
6) **Não** serão avaliados os trabalhos:
    a. Que chegarem fora do prazo;
    b. Que não compilarem;
    c. Que não foram compactados em um só arquivo;
    d. Que não tiverem identificação (nome e R.A.);
    e. Que não seguirem todas estas instruções;
    f. Que possuirem código semelhante.
    g. Que não forem apresentados por seus autores em uma das datas determinadas acima.
7) Não se esqueça que o trabalho vale 20% da nota do bimestre, e que é uma preparação para a prova do bimestre.

Exercícios do trabalho: Todos os programas devem ser feitos utilizando funções e procedimentos de forma adequada. Comente qual é o objetivo de cada um desses módulos.

1) Complemente a versão recursiva para as operações de inserção e remoção em árvores binárias de busca vistas em sala de aula, para que esta se torne uma implementação de uma árvore AVL. Para isso, implemente os procedimentos/funções para balanceamento que serão introduzidas nas operações de inserção e remoção de árvores binárias de busca:

(a) Operação que calcula a altura de um nó *p* com base na altura de seus filhos, guardando esta informação na estrutura de dados do nó.
(b) Operação que calcula o Fator de Balanceamento de um nó (FB) com base na altura de seus filhos. (Os itens (a) e (b) pode ser implementados de forma conjunta)
(c) Operação que rotaciona um nó *p* à sua esquerda.
(d) Operação que rotaciona um nó *p* à sua direita.
(e) Operação que rebalanceia uma subárvore com raiz em *p*.
(f) Operação que recalcula a altura de todos os nós de uma subárvore com raiz em *p*. (Que deve ser chamada após o rebalanceamento)]

2) Defina a estrutura de dados e implemente as seguintes operações para uma tabela *hash* com M = 50 posições e chaves possíveis de 0 a 499, que implementa ***hash* duplo**:
   (a) Inserção de uma chave *k*, mostrando sempre a seqüência de sondagem que foi pesquisada para a inserção.
   (b) Remoção de uma chave *k*, mostrando sempre a seqüência de sondagem que foi pesquisada para a remoção.
   (c) Procura por uma chave *k*, mostrando sempre a seqüência de sondagem que foi pesquisada para a remoção e devolvendo o índice da chave *k* na tabela.

3) Um programador precisa implementar um algoritmo de ordenação para fazer a ordenação dos cadastros dos clientes de uma empresa. A base de dados de clientes está em disco, é muito grande e está toda desordenada. Isto implica que o algoritmo de ordenação implementado deve ser bastante rápido e além disso deve fazer o menor número de trocas possíveis (por causa do acesso a disco).
   O programador deseja comparar alguns métodos de ordenação, baseado em suas implementações em memória primária, para tomar a decisão de qual implementar. Para o programador, o importante é estudar o algoritmo em relação ao número de trocas de elementos e o número de comparações feitas em cada um dos algoritmos de ordenação.
   Implemente um programa no qual o programador possa executar 4 métodos de ordenação: seleção, inserção, *quicksort* e *heapsort*. O programa deve gerar aleatoriamente o conjunto de dados para ordenar (no máximo 10 números inteiros), e dar como resposta o conjunto ordenado e o número de comparações e trocas feitas para cada dos algoritmos de ordenação, para que o programador possa decidir qual algoritmo implementar.
   O método *heapsort* deve utilizar uma UNIT que você deve implementar com as rotinas vistas em sala de aula para manipularem uma árvore *heap*.