Métodos de Programação II

LESI / LMCC

21 de Junho 2003

Questão 1 [*Análise de Algoritmos*] Considere a seguinte definição:

> *Seja $\mathcal{A}$ um conjunto de pares ordenados (x, y) em que $x < y$, com x, y números inteiros positivos.*
> *(x, y) diz-se um par* mínimo *se não existe qualquer outro par $(x', y') \in \mathcal{A}$ tal que $x \leq x'$ e $y' \leq y$.*

Considere agora o tipo de dados seguinte com que se representa os ditos pares ordenados:

```
typedef struct { int p; int s; } pair;
```

A função seguinte pretende resolver o problema. O seu resultado é o vector `v`, que depois da execução do algoritmo conterá, na posição `i`, o valor 1 sse `A[i]` for um par mínimo (no caso contrário deverá conter 0). Considera-se que o vector `A` se encontra à partida ordenado por ordem crescente da primeira componente dos pares.

```
void minpairs (pair A[], char v[])
{
  int i,j;
  for (i=1; i<=N; i++)  v[i] = 1;
  for (i=N; i>=1; i--)
    for (j=i-1; j>=1; j--)
      if ((A[i].s <= A[j].s)) v[j] = 0;
}
```

1. Efectue a análise assimptótica do tempo de execução no pior e no melhor casos do algoritmo. Justifique convenientemente a sua resposta.

2. O ciclo interior obedece ao seguinte invariante:

   > *No fim de cada iteração,* `v`$[k] = 0$ *sse* `A`$[k]$`.p` $\leq$ `A[i].p` *e* `A[i].s` $\leq$ `A`$[k]$`.s`*, para qualquer k tal que $j \leq k \leq i - 1$.*

   Estude as suas propriedades de inicialização, preservação, e terminação.

3. O algoritmo pode ser melhorado (no que respeita ao seu comportamento temporal) mantendo a mesma estrutura de dois ciclos. Para isso complete as expressões P e Q no algoritmo abaixo. Diga o que se alterou no comportamento no pior e no melhor casos.

   ```
   void minpairs (pair A[], char v[])
   {
     int i,j;
     for (i=1; i<=N; i++)  v[i] = 1;
     for (i=N; i>=1; i--)
       if (___P___)
         for (j=i-1; j>=1; j--)
           if (___Q___ && (A[i].s <= A[j].s)) v[j] = 0;
   }
   ```

Nome:_____ **Número:**_____ **Curso:**_____

Métodos de Programação II

LESI / LMCC

21 de Junho 2003

Questão 2 [*Análise de Algoritmos*] O algoritmo de ordenação Tree Sort utiliza o seguinte procedimento para ordenar uma lista de n elementos:

- constrói uma árvore binária de pesquisa, inserindo os elementos da lista um a um.
- percorre a árvore binária de pesquisa por forma a obter uma versão ordenada da lista.

As seguintes declarações pressupõem um funcionamento deste tipo:

```
typedef struct treenode TreeNode, *Tree;

struct treenode {
  int num;
  struct treenode *esq;
  struct treenode *dir;
}

Tree insertTree(Tree t, int num);

void treeSort(int A[], int n);
```

Responda às seguintes questões, tendo em atenção que elas podem ser respondidas por qualquer ordem:

1. Forneça uma implementação da função `insertTree` e escreva uma equação de recorrência que exprima o seu tempo de execução no pior caso.
2. Implemente a função `treeSort`, justificando a forma como irá percorrer a árvore binária de pesquisa. Utilize a função `insertTree` para construir a árvore binária de pesquisa.
3. Analize o tempo de execução de `treeSort` no pior caso, e exprima esse comportamento utilizando a notação Θ.
4. Que notação ($\mathcal{O}$, Ω, ou Θ) é mais adequada para exprimir directamente o comportamento do algoritmo no pior caso? Justifique.
5. Repita a análise do tempo de execução de `treeSort` para o melhor caso.

Nome:_____ **Número:**_____ **Curso:**_____

Métodos de Programação II

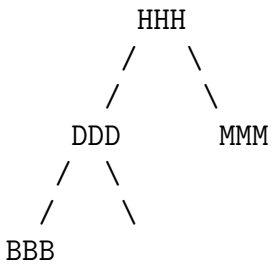
LESI / LMCC

21 de Junho 2003

Questão 3 [*Árvores Binárias e AVL*]

1. Descreva por palavras suas os princípios subjacentes à utilização de árvores AVL.

2. Considere a seguinte árvore AVL. Mostre o resultado de se inserir sucessivamente nesta árvore os nós com chave AAA e CCC, preservando sempre o invariante destas árvores.



3. Escreva em **C** uma função que receba uma árvore (AVL) e um inteiro d, e efectue uma rotação simples da árvore á direita (se $d = 1$) ou à esquerda (se $d = 0$). Utilize o tipo de dados:

```
struct _node_ {
  Info i;
  struct _node_ * e;
  struct _node_ *d;
};

typedef struct _node_ *bintree;
```

4. Diga por palavras suas como implementaria um algoritmo iterativo (i.e. sem utilização de recursividade) para efectuar uma travessia PREORDER de uma árvore binária.

Nome:_____ **Número:**_____ **Curso:**_____

Métodos de Programação II

LESI / LMCC

21 de Junho 2003

4

Questão 4 [*Grafos*] Considere o grafo representado pela seguinte lista de adjacências:

$$\begin{array}{ll} A: & B(4), F(2) \\ B: & A(1), C(3), D(4) \\ C: & A(6), B(3), D(7) \\ D: & A(6), E(2) \\ E: & D(5) \\ F: & D(2), E(3) \end{array}$$

(por exemplo, existem arcos de A para B com peso 4 e de A para F com peso 2)

1. Desenhe o grafo assim representado. Classifique-o de acordo com os vários critérios que conhece.

2. Quantos caminhos mais curtos existem do nó C para E? Qual destes seria determinado pelo algoritmo de Dijkstra (sobre a representação fornecida)? Justifique.

3. Assumindo as seguintes definições de tipos de dados (representação mista de grafos):

```
typedef struct _edge_ {
  int dest;
  int weight;
  struct _edge_ *next;
} edge;

typedef edge* Graph[MAX];
```

   Escreva uma função que determine se existem dois arcos com o mesmo peso num grafo (deverá devolver 1 ou 0).

4. Efectue a análise assimptótica do tempo de execução da função que escreveu. Como se poderia tornar esta função mais eficiente (em relação ao tempo de execução), sendo conhecido à partida o valor máximo dos pesos?

Nome:_____ **Número:**_____ **Curso:**_____