Métodos de Programação II

ESI / MCC

Ano Lectivo de 2004/2005 (2a Chamada)

Questão 1 Considere que se representam polinómios guardando apenas os seus coeficientes por ordem decrescente do grau. Assim, por exemplo, o polinómio $4 + 2\,x^3$ será guardado num vector cujas primeiras 3 posições têm os valores 2, 0, 0 e 4.
Considere as funções abaixo que calculam o valor de um polinómio num ponto.

```
double valor1 (double p[], int g, double x) {
  double r; int k;

  r = 0; k = 0;
  while (k <= g) {
     r = (r * x) + p [k];
     k = k+1;
  }
  return (r);
}

double valor2 (double p[], int g, double x) {
  double r; int k;

  r = 0; k=0;
  while (g >= 0) {
     r = r + p[k] * power (x,g) ;
     k = k+1;
     g = g-1;
  }
  return (r);
}
```

1. Escreva predicados apropriados à pré e pós-condições de ambas estas funções (de acordo com a descrição informal feita acima).

2. Para cada uma das funções, apresente os invariantes necessários à prova da correção das funções.

3. Admitindo que a função `power` usada em `valor2` executa em tempo logarítmico relativo ao valor do expoente, determine o tempo de execução de cada uma das soluções em função do grau do polinómio. Admita ainda que a multiplicação de *double*s se faz em tempo constante.

Questão 2 Relembre os algoritmos de inserção e pesquisa em árvores binárias de procura e em árvores de procura balanceadas. No que respeita ao tempo de execução de cada um destes (quatro) algoritmos (em função do número de elementos da árvore), identifique os melhor e pior casos, indicando limites superiores para cada um destes casos. Em particular para a inserção, pronuncie-se para o caso de fazermos uma única inserção ou várias consecutivas.

Questão 3 Relembre o algoritmo de travessia em profundidade de um grafo a partir de um dado vértice.

```
void DF (Grafo g, int numVert, int o,  int df[]) {
int i; Stack s; AdjList p;
int v[MaxVertice];

for(i=0; (i<numVert); v[i++] = 0);
v[o] = 1;
s = newStack(); push(s,o); i=0;
while (tamanho (s) > 0){
  o = pop (s); df[i++] = o;
  for (p=g[o]; p != NULL; p = p->next)
     if (v [p->destino] == 0){
        push (s, p->destino);
        v[p->destino] = 1;
     }
}
```

1. Ao executarmos esta travessia, calculam-se quais os vértices alcançáveis do nodo dado, i.e., quais os vértices para os quais há um caminho **desse** nodo. Descreva e codifique uma alternativa para calcular, dado um nodo, quais os vértices para os quais existe um caminho **para** esse vértice. Modifique, se entender por bem, a representação de grafos a usar.

2. Relembre, por exemplo da implementação do algoritmo de Prim para o cálculo de uma árvore geradora de custo mínimo, o uso de um vector para armazenar a dita árvore. Nesse vector guarda-se para cada vértice o índice correspondente ao seu "pai". Modifique o procedimento acima de forma a, em vez de retornar um vector com os índices, retornar um vector com a árvore correspondente à travessia efectuada.

Questão 4 Considere que para um determinado problema se desenvolveram duas soluções, cujos tempos de execução, são dados (em função da dimensão N do seu input) pelas seguintes recorrências.

$$T_1(N) = \begin{cases} c_1 & \text{se} \quad N \leq 1 \\ 2T_1(N/3) & \text{se} \quad N > 1 \end{cases}$$

$$T_2(N) = \begin{cases} c_2 & \text{se} \quad N \leq 1 \\ 3T_2(N/3) & \text{se} \quad N > 1 \end{cases}$$

1. Para cada uma delas *adivinhe* uma solução e mostre informalmente a sua validade.

2. Em qual (ou quais) das classes de complexidade estudadas (P, NP e NP-completo) poderá incluir o problema em causa? Justifique a sua resposta.