Rio de Janeiro, 5 de Outubro de 2009.
PROVA 1 DE PROJETO E ANÁLISE DE ALGORITMOS
PROFESSOR: EDUARDO SANY LABER
DURAÇÃO: 3 HORAS

1. (2.0pt) Seja $A = (a_1, a_2, \ldots, a_n)$ uma sequência de números distintos. Uma inversão em A é um par (a_i, a_j) tal que $i < j$ e $a_i > a_j$. O número de inversões de uma sequência captura o grau de ordenação de uma sequência.

Escreva um pseudo-código para um algoritmo que recebe uma sequência A e determina o número de inversões de A. Analise o algoritmo encontrando uma função $f(n)$ tal que $T(n) = \theta(f(n))$, aonde $T(n)$ é a complexidade de pior caso do algoritmo proposto para uma entrada de tamanho n.

2. (2.0) Seja $G = (V, E)$ um grafo conexo e não direcionado que esta representado por uma lista de adjacências. Um par de vértices $x, y \in V$ é *importante* se a remoção simultânea de x e y do grafo desconecta o grafo, ou seja, se $G - \{x, y\}$ é desconexo. Explique como seria um procedimento para contar o número de pares importantes de um grafo. Qual a complexidade do algoritmo proposto?

3. (2.0) Considere o pseudo-código abaixo

Main
    i $\leftarrow$ 0
    Enquanto $i^3 < n$
        i++
        Proc(i)
    Fim Enquanto
End Main

Proc(i)
    Se i=1 Return
    Senão
        $j \leftarrow 0$
        Proc(i/2)
    Fim Se
Fim Proc

a) Encontre uma função $f(i)$ tal que $T(i) \in O(f(i))$, aonde $T(i)$ é a complexidade de pior caso do Procedimento **Proc**. Exiba a função $f(i)$ mais justa que você conseguir.

b) Encontre uma função $f(n)$ tal que $T(n) \in O(f(n))$, aonde $T(n)$ é a complexidade de pior caso do Procedimento **Main**. Exiba a função $f(n)$ mais justa que você conseguir. Note que **Main** chama **Proc**.

4. (2.0) Seja $D = (V, E)$ um grafo direcionado. O vértice $u \in V$ é *perigoso* se existe um vértice $v \neq u$ em D tal que existe caminho de u para v mas não existe caminho de v para u. Descreva como seria um algoritmo para testar se um **dado** vértice u é perigoso ou não e análise sua complexidade. Obs: Algoritmos eficientes ganham mais pontos.

5. (3.0). Seja A uma lista de $n \geq 100$ inteiros não necessariamente distintos.

a) Descreva um algoritmo para determinar se existe um elemento de A que aparece uma única vez em A. O algoritmo retorna **sim** se tal inteiro existir e **não** caso contrário. Determine a complexidade do algoritmo proposto. Obs: Algoritmos eficientes ganham mais pontos.

b) Seria possível obter um algoritmo mais eficiente se soubéssemos que os inteiros estão estão no conjunto $\{1, \ldots, 2n\}$? Como?

c) Seria possível obter um algoritmo ainda mais eficiente se soubéssemos que os inteiros estão estão no conjunto $\{1, \ldots, n/2\}$? Como?