Universidade Federal do Paraná – UFPR
Setor de Ciências Exatas / Departamento de Informática – DInf
Algoritmos e Estruturas de Dados II – CI1056 / 2º semestre de 2019
Professores: André Grégio e Marcos Castilho.

NOME: Vinicius Tihara V. Date, GRR: 20190367 Data: 06/09/2019

- O teste é individual e sem consulta, exceto o contido em sua mente;
- A interpretação das questões faz parte da avaliação;
- Esta folha de prova contém questões no verso.

1 | 2 | 3 | 4 | 5

1ª Prova

1. (**55 pontos** – 10/5/20/5/5/10) Sobre ordenação, responda as questões abaixo considerando o algoritmo mostrado a seguir e o vetor `V = [5, 4, 3, 2, 1]`.

```
ALG(vetor v; inteiro n)  // o vetor eh de inteiros e n eh o numero de elementos
{
    PARA i := 1 até n-1 FAÇA
    {
        jmin := i
        PARA j := i+1 até n FAÇA
            SE v[j] < v[jmin] ENTÃO
                jmin := j
        aux := v[i]
        v[i] := v[jmin]
        v[jmin] := aux
    }
}
```

n-1 + n-2 + n-3 + n ... 1

$\frac{(n-1+1).n-1}{2} = \frac{(n-1)^2}{2}$?

   i Qual é a função de custo de `ALG`?

   ii Qual é o nome pelo qual `ALG` é conhecido e citado na literatura? Selection

   iii Implemente o algoritmo do *insertion sort* em PSEUDO-CÓDIGO com o seguinte protótipo: `InsertionSort(vetor a, inteiro n)`, sendo a um vetor de inteiros e n o tamanho dele.

   iv Para ambos, `ALG` e *insertion sort*: desenhe o vetor parcialmente ordenado ao final de uma rodada completa do laço mais interno (variável de controle j).

   v Faça uma tabela com as funções de custo considerando o número de comparações entre os elementos do vetor para o melhor e pior caso de ambos os algoritmos `ALG` e *insertion sort*.

   vi Os métodos da bolha e da inserção têm um caso de entrada especial para o qual eles apresentam comportamento mais eficiente do que quadrático. Há alguma maneira de obter o mesmo para `ALG`? Que modificações poderiam ser feitas no código de `ALG` para se obter este objetivo? Qual seria o novo custo para este caso após as modificações? Explique.

2. (**10 pontos** – 5/5) Dado o algoritmo a seguir, responda as questões:

```
ALG(tad_matriz A, B, var tad_matriz C)
{
    SE A.col = B.lin ENTÃO
    {
        PARA i := 1 até A.lin FAÇA
            PARA j := 1 até B.col FAÇA
            {
                C.m[i,j] := 0
                PARA k := 1 até B.col FAÇA
                    C.m[i,j] := C.m[i,j] + A.m[i,k] * B.m[k,j]
            }
    }
    SENÃO imprimir ("erro!")
}
```

i Descreva suncintamente como o algoritmo funciona (isto é, o que faz com as entradas) e qual o resultado final.

ii Qual a função de custo desse algoritmo (em relação às multiplicações realizadas)? Detalhe como você chegou nesse valor.

3. (**35 pontos** – 14/6/10/5) Considere o vetor `V = [1, 2, 3, 4]` para responder as questões a seguir.

i Implemente em PSEUDO-CÓDIGO uma função iterativa para a **busca binária** que retorne o índice do vetor que contém o elemento procurado, caso ele seja encontrado, ou "0" quando o elemento não estiver no vetor. O protótipo é: `buscaBinIt(vetor v, inteiro x,` onde v é o vetor de entrada, x é o elemento a ser procurado, p é o índice do primeiro elemento do vetor e u é o índice do último elemento do vetor.

ii Faça o teste de mesa para as seguintes chamadas e mostre o custo com relação ao número de comparações entre os elementos do vetor, para cada caso:

- `imprimir (buscaBinIt(v, 2, 1, 4))`
- `imprimir (buscaBinIt(v, 5, 1, 4))`
- `imprimir (buscaBinIt(v, 4, 1, 4))`

iii Implemente em PSEUDO-CÓDIGO um algoritmo baseado na busca binária que encontre *todas* as ocorrências de x em um vetor de inteiros qualquer com menor custo possível. Você pode usar o protótipo `buscaBinIt` acima definido.

iv Qual é o custo deste algoritmo no pior caso? Explique.