Estruturas de Dados II (DEIN0083) 2017.2
Curso de Ciência da Computação
Reposição - Prova 1

Prof. João Dallyson Sousa de Almeida — Data: 15/01/2018

Aluno: Robherson Victor de J. Costa — **Matrícula:** 2 0 1 3 0 1 4 2 7 7

Regras durante a prova:

- É vetada: a consulta a material de apoio, conversa com colega e a utilização de dispositivos eletrônicos. A não observância de algum dos itens acima acarretará a anulação da prova.

I. (2.0pt) Para cada função f(n) abaixo, dê um limite superior assintótico usando a notação "Big-O". Você deve dar o limite mais próximo possível (Não tente usar 100^n como resposta em todas as opções).

a) $f(n) = 40n^5 + 2n^3 + 8n^2$
b) $f(n) = 3n^3 - 3n^3 + 2n$
c) $f(n) = (logn)(n^2 + n^3)$
d) $f(n) = 20 * 4^n + 0.3245n$

II. (2.0pt) Utilize o teorema Mestre para analisar assintoticamente as recorrências a seguir:

a) $T(n) = 0.5T(n/2) + 1/n$	b) $T(n) = 16T(n/4) + n$
c) $T(n) = 7T(n/3) + n^2$	d) $T(n) = 3T(n/3) + n/2$

III. (2.0pt) Ordene as letras da string "QUESTION", contando a quantidade de comparações realizadas e apresentando o vetor após cada iteração, utilizando os seguintes algoritmos de ordenação:
a) Inserção: liste o vetor para cada elemento incluído na ordenação parcial até o momento.
b) Shellsort. Use 1,3,5,11 como a sequência de valores para h. Liste o vetor para cada novo valor de h, enquanto $h > 1$. Quando h=1, liste o vetor para cada elemento inserido na ordem parcial.
c) QuickSort, usando o elemento da direita da partição como pivô. Liste o vetor para cada nova partição completada com dois ou mais elementos.
d) Quantas comparações foram realizadas pelo algoritmos da letras a, b e c?

IV. (1.0pt) É possível modificar o algoritmo do QuickSort para tê-lo executando com tempo de melhor caso? Justifique sua resposta.

V. (1.0pt) Determine o tempo de execução do algoritmo abaixo. Justifique sua resposta.

```
int cont = 0;
for (int i = n; i > 0; i = i / 4){        n/4
    for (int j = 0; j < i; j++){          n/4
        cont += 1;
    }
}
```

TEMPO = $\frac{2n}{16}$

VI. (2.0pt) Utilize o algoritmo de ordenação HeapSort para ordenar o vetor [5, 7, 1, 8, 2, 3, 9] em ordem decrescente. Apresente, passo a passo (árvore intermediária) a estrutura da Heap após a construção. Apresente a solução da ordenação mostrando passo a passo (ilustrando a árvore e o vetor em cada iteração).