Universidade Federal de Pernambuco
Centro de Informática
Bacharelado em Sistemas de Informação

IF969 Algoritmos e Estruturas de Dados — 2012.2
Prof. Paulo Fonseca

SEGUNDA LISTA DE EXERCÍCIOS
19 de Fevereiro de 2013

- Esta lista contém **04** questões escritas e **02** questões práticas de programação.
- As respostas às questões escritas devem ser entregues ao professor em papel escrito à mão (à tinta e legível) até o final da aula do dia **04/Mar/13**.
- As questões práticas devem ser submetidas através do sistema de correção automática em `www.cin.ufpe.br/~if969`, seção "Listas" até o dia **03/Mar/13 às 17h** (hora do servidor)
- A lista é **individual**.
- É permitido discutir as questões com os colegas e monitores mas as respostas devem ser escritas individualmente com suas próprias palavras.
- É permitido estudar códigos de terceiros mas *não é* permitido copiar trechos substanciais e apresentá-los como trabalho seu.
- A detecção de cópia implicará na atribuição de nota 0 (zero) à lista.
- O professor poderá solicitar aleatoriamente a explicação oral de suas respostas até a divulgação das notas. Essa explicação deverá ser feita sem assistência. Esteja certo de ter compreendido suas respostas.

QUESTÃO 1 (1,5 pts) Os algoritmos A e B gastam exatamente $T_A = c_A n \lg n$ e $T_B = c_B n^2$ microsegundos, respectivamente, para um problema de tamanho n. Pergunta-se:

a) Qual dos dois algoritmos é o mais eficiente do ponto de vista assintótico?
b) Sabendo-se que os algoritmos A gasta 10μs para processar $n = 1024$ itens e B gasta apenas 1μs para processar a mesma entrada, qual o melhor dos dois algoritmos para processar uma entrada de tamanho $n = 220$?

QUESTÃO 2 (1,5 pts) Considere o algoritmo de ordenação *Quicksort* com a função de partição a seguir.

Função *Partition*
Entrada $V = (v_1, \ldots, v_n)$: vetor a particionar;
l, r: limites do intervalo a particionar
Saída Particiona o trecho $v_l, \ldots, v_r$ e retorna a posição final do pivô

```
1  p ←⌊(l + r)/2⌋
2  Permuta v[l] ↔ v[p]
3  i, j ←l, r
4  enquanto i < j faça
5      enquanto i ≤ n ∧ v[i] ≤ v[l] faça
6          i ← i + 1
7      fim faça
8      enquanto v[j] > v[l] faça
9          j ← j − 1
10     fim faça
11     se i < j então
12         Permuta v[i] ↔ v[j]
13     fim se
14 fim faça
15 Permuta v[l] ↔ v[j]
16 devolva j
fim
```

a) Considerando um cenário no qual a operação mais custosa corresponde a permutar elementos (linhas 2, 12,15), exiba um exemplo do pior caso quando $n = 8$ (por simplicidade, considere uma ordenação de valores representados por $A < B < C < \cdots < X < Y$).
b) Ilustre a execução do algoritmo exibindo o vetor após cada execução da função *partition*.

QUESTÃO 3 (1,5 pts) Seja n um nó de uma árvore binária e $D(n)$ o número de *descendentes* de n (i.e. filhos, filhos dos filhos, filhos dos filhos dos filhos, etc.)

a) Escreva uma definição *recursiva* de $D(n)$.
b) Escreva um algoritmo em pseudo-código para imprimir os valores de $D(n)$ *em pós-ordem* para cada n de uma árvore binária fornecida como entrada.

QUESTÃO 4 (1,5 pts)

Escreva em pseudo-código um algoritmo *isBST* que recebe com entrada um apontador para a raiz de uma árvore binária *root* e retorna o booleano **True**, se a árvore dada é uma árvore de busca binária (BST), ou **False** caso contrário.

QUESTÃO 5 (2 pts)

Questão prática de implemetação. Ver Lista 2 em `www.cin.ufpe.br/~if969`.

QUESTÃO 6 (2 pts)

Questão prática de implemetação. Ver Lista 2 em `www.cin.ufpe.br/~if969`.