# Algoritmos para Automação e Sistemas
# Terceira Lista de Exercícios

## Pilhas, Filas e Listas

1) Descreva o resultado de cada operação na sequência ENQUEUE(Q,4), ENQUEUE(Q,1), ENQUEUE(Q,3), DEQUEUE(Q), ENQUEUE(Q,3) e DEQUEUE(Q) sobre uma fila Q inicialmente vazia armazenada no arranjo Q[1..6].

2) Reescreva ENQUEUE e DEQUEUE para detectar o estouro negativo e o estouro positivo de uma fila.

3) Descreva o resultado de cada operação na sequência PUSH(S,4), PUSH(S,1), PUSH(S,3), POP(S), PUSH(S,8) e POP(S) sobre uma pilha S inicialmente vazia armazenada no arranjo S[1..6].

4) Reescreva PUSH e POP para detectar o estouro negativo e o estouro positivo de uma pilha.

5) Explique como implementar duas pilhas em um único arranjo A[1..n] de tal modo que nenhuma das pilhas sofra um estouro positivo, a menos que o número total de elementos em ambas as pilhas juntas seja *n*. As operações PUSH e POP devem ser executadas no tempo O(1).

6) Implemente uma pilha usando uma lista simplesmente ligada L. As operações de PUSH e POP ainda devem demorar o tempo O(1).

7) Implemente uma fila através de uma lista simplesmente ligada L. As operações ENQUEUE e DEQUEUE ainda devem demorar o tempo O(1).

8) Implemente as operações de INSERT, DELETE e SEARCH em uma lista duplamente encandeada.

9) Implemente as operações de dicionário INSERT, DELETE e SEARCH, usando listas circulares simplesmente ligadas. Quais os tempos de execução dos seus procedimentos?

## Árvores Binárias

Forneça como entrada para os exercícios 12, 13, 14, 15 e 18 uma arquivo contendo árvore binária da seguinte forma (ou seja, cada nodo é separado por uma nova linha):

Nodo1; Filho da Esquerda; Filho da Direita
Nodo2; Filho da Esquerda; Filho da Direita
Nodo3; NIL; NIL

10) Uma árvore binária de pesquisa tem 10 nós. Os nós foram inseridos na seguinte ordem: F C E F G A B I H J. Desenhe a respectiva árvore e realize os seguintes percursos.

a) Faça o percurso em ordem da árvore.
b) Faça o percurso em pré-ordem da árvore.
c) Faça o percurso em pós-ordem da árvore.

11) Verifique se o código abaixo é equivalente ao da função e-r-d (esquerda-raiz-direita, ou seja, caminhamento em ordem):

```
while (1) {
  while (x != NULL) {
    p[t++] = x;
    x = x->esq;
  }
  if (t == 0) break;
  x = p[--t];
  printf ("%d\n", x->conteudo);
  x = x->dir;
}
```

12) Escreva uma função que calcule o número de nós de uma árvore binária.

13) Modifique o algoritmo de busca para que ao invés dele encontrar um nó, ele encontre o pai deste.

14) Escreva um algoritmo que, dada uma árvore binária, remove o nó contendo o menor valor.

15) A profundidade (= *depth*) de um nó x em uma árvore binária com raiz r é a distância entre x e r. Mais precisamente, a profundidade de x é o comprimento do (único) caminho que vai de r até x. Por exemplo, a profundidade de r é 0 e a profundidade de r->esq é 1. Escreva uma função que determine a profundidade de um nó em relação à raiz da árvore.

**Árvores Enraizadas com Ramificações Ilimitadas**

16) Escreva um procedimento de tempo *O(n)* que imprima todas as chaves de uma árvore enraizada arbitrária de n nós, onde a árvore é armazenada usando a representação de filho da esquerda, irmão da direita.

17) A representação de filho da esquerda e irmão da direita de uma árvore arbitrária utiliza três ponteiros em cada nó: filho da esquerda, irmão da direita e pai. A partir de qualquer nó, seu pai pode ser alcançado e identificado em tempo constante, e todos os seus filhos podem ser alcançados e identificados em tempo linear no número de filhos. Mostre como usar apenas dois ponteiros e um valor booleano em cada nó para que o pai de um nó ou todos os seus filhos possam ser alcançados e identificados em tempo linear no número de filhos.

**Árvores AVL**

18) Implemente as operações de inserção, remoção e busca em C/C++ para:

a) construir a seguinte árvore AVL: 3, 2, 1, 4, 5, 6 e 7
b) buscar e remover os nós 7 e 2 respectivamente
c) crie uma árvore binária com 10K nós (usando valores aleatórios) e compare o tempo de busca usando uma árvore binária sem balanceamento e com balanceamento.

**Data de entrega: 09 de junho de 2015 (terça-feira).**
**Após esta data será descontado 2 pontos por dia de atraso.**
**A lista de exercícios deve ser resolvida e entregue individualmente.**

02/06/2014