# MC202 EF - 2. sem. 2008 - Lista de Exercícios

Considere as seguintes definições utilizadas para construir árvores binárias:

```
typedef struct no *apno;
typedef struct no {  char * info; apno esq, dir, pai; } no;
```

1 - Escreva uma função para determinar a altura de uma árvore binária (uma árvore vazia tem altura zero, uma árvore com um único nó tem altura 1, e para uma árvore não vazia, sua altura é sempre a maior altura das suas sub-árvores mais um).

2 - Escreva uma função para determinar se duas árvores são isomorfas (duas árvores são isomorfas se (a) as duas são vazias ou (b) as duas têm um único nó e o conteúdo deles é igual ou (c) o conteúdo das raízes das duas árvores é igual, as sub-árvores esquerdas são isomorfas e as sub-árvores direitas são isomorfas.

3 - Uma árvore T satisfaz ao critério de balanceamento AVL se para toda sub-árvore S de T, a diferença entre as alturas das sub-árvores esquerda e direita de S é no máximo 1 (em valor absoluto). Escreva a função para verificar se uma árvore T satisfaz a esse critério:

```
int satisfazAVL(apno T)
```

4 - Escreva a função clone() que retorna a cópia de uma árvore binária passada como parâmetro. Sua função deve obedecer ao seguinte protótipo:

```
apno clone(apno tree)
```

5 - Supondo disponível a operação *insere(X,t),* para inserir um valor *X* numa árvore binária de busca cuja raiz é *t* e um vetor V com N elementos ordenado em ordem crescente

a- descreva uma forma de criar uma árvore de busca balanceada a partir do mesmo.
b- estime o tempo dessa operação.

6 - Escreva uma função, de acordo com o protótipo abaixo, que retorna os valores mínimo e máximo contidos numa árvore binária de busca não vazia. Faça uma estimativa de tempo e uso de memória para a sua função.

```
void min_max(int *min, int *max, apno p);
```

7 - É dado um vetor com ***N*** elementos onde todos estão ordenados em ordem crescente exceto por ***k*** elementos que estão fora de ordem. Qual o método de ordenação que você utilizaria para ordenar esse vetor da maneira mais rápida possível considerando as seguintes possibilidades:

a - k é bem menor que $log_2$ N
b - k é aproximadamente igual a $log_2$ N

8 - É dada uma lista contendo ***N*** nomes formados por caracteres alfabéticos, cada um de tamanho menor ou igual a um certo ***k***. Supondo que esses nomes podem ser mantidos em memória, qual o método mais eficiente para ordená-los.

9 - Considere um vetor *A* com *n* elementos onde $A_1 < A_2 < .. < A_k > A_{k+1} > A_{k+2} > ... > A_n$, ou seja, está ordenado em ordem crescente até o k-ésimo elemento, e a partir desse elemento está ordenado em ordem decrescente

a - Escreva uma função para determinar o valor de k em tempo O(log n).

b - Dado o vetor da questão anterior, proponha um algoritmo para ordenar os elementos em tempo O(n).

10 - Considere o conjunto de valores ( 12, 15, 9, 11, 5, 8). Qual seria a permutação desses elementos correspondente ao pior caso para cada um dos seguintes algoritmos de ordenação:
*a* - bubble sort *b* - heap sort *c* - quick sort

11 - É dada uma estrutura do tipo 'lista de listas' onde cada nó é definido como:

```
typedef node *link;
typedef struct node {
  char *info;      /* informação associada ao nó  */
  link next;       /* próximo nó da lista         */
  link childList;  /* lista de 'filhos'           */
} node;
```

Escreva uma função para liberar toda a memória ocupada por uma lista desse tipo, tendo como parâmetro o apontator para o início da mesma.

12 - Descreva como cada um dos algoritmos de ordenação (bubblesort, heapsort, quicksort) se comporta (em tempo e memória) em cada um dos seguintes casos:
a - uma entrada qualquer
b - a entrada já está ordenada
c - a entrada está ordenada em ordem inversa à desejada
d - a entrada já está ordenada, a menos de um conjunto muito pequeno de elementos que está fora de ordem. Justifique cada item da sua resposta.

13- Dada uma árvore rubro-negra com altura h indique (a) o número mínimo de nós dessa árvore e (b) o número máximo de nós para essa árvore.

14 - Escreva uma função para verificar se uma árvore atende aos requisitos de ser uma árvore rubro-negra. Suponha que a árvore é formada por nós do seguinte tipo:

```
typedef struct node* link;
struct node{Item item; link left, right, father; int colour; };
```

15 - Considere um vetor de tamanho n formado por elementos do tipo node, da questão anterior. Esse vetor continha originalmente uma árvore binária, mas uma função mal escrita destruiu os apontadores para as sub-árvores esquerda e direita(**`left`** e **`right`**), deixando intacto o apontador para o pai (**`father`**). Você deve escrever uma função que reconstrói a árvore original, supondo que no vetor, o índice do filho esquerdo é sempre menor que o índice do filho direito e que se um nó tem apenas um filho, esse filho é o filho direito A sua função deve retornar o apontador para a raiz da árvore.

16- Um programador inventou uma função que é capaz de associar um string com tamanho menor ou igual a um certo k, a um número inteiro único. Discuta a possibilidade de se usar essa função como função de hashing.

17 - É dado um vetor de inteiros com tamanho N que está ordenado em ordem crescente até o k-ésimo elemento e em ordem decrescente a partir desse elemento. Escreva uma função que ordene esse vetor em tempo linear.

18 - Considere o trecho de código abaixo onde é declarada a função **`insere()`** que insere o valor k numa lista de valores ordenada em ordem crescente, preservando a ordenação:

```
...
#define TRUE 1
#define FALSE 0

typedef struct no* apno;

struct no{ int info; apno prox; }

apno novo_no(int k, apno pr){
  apno p;
  p=(apno)malloc(sizeof(struct no));
  p->info=k; p->prox=pr;
  return p;
}

int insere(apno *aplista, int k){
  if(*aplista != NULL) if((*aplista)->info < k)
                          return insere(&(*aplista)->prox),k);
  *aplista=novo_no(k,*aplista);
  return TRUE;
}
...
```

Reescreva uma função não recursiva equivalente a **`insere()`**.

19 - Considere um programa que imprime os n primeiros inteiros de uma série através do comando abaixo (k é o elemento da série sendo impresso):

```
...
printf("%d, ",k);
...
```

a - Dê uma estimativa do número de caracteres impressos (logarítimica, linear, 'n log n' ou quadrática) considerando que a série é (a1) uma progressão aritmética, (a2) progressão geométrica (justifique suas respostas).
b - Proponha um tipo de série em que o número de caracteres impressos cresce exponencialmente em relação a n (justifique sua resposta).