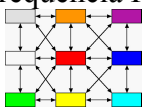
Estruturas de Dados
2016/17
Frequência F2



Universidade da Beira Interior
Departamento de Informática

(Grupo I - Árvores)

II-1 (2 valores) Esquematize a estrutura de uma árvore binária (não balanceada) que ficaria balanceada através de 2 operações consecutivas de rotação à esquerda.

II-2 (3 valores) Considere a seguinte estrutura de dados, utilizada para registar os clientes de uma empresa de telecomunicações:

```
typedef struct NODO_AB{
        int BI;                 //chave de ordenação na árvore
        char nome[80];
    int totalMinutos;  //total de minutos facturados
        struct NODO_AB *fe, *fd;
}NodoAB;
```

Implemente uma função que mostre (ordenadamente) o BI dos elementos com valor total de minutos facturados superior a um limite.

Protótipo: void mostraClientes(NodoAB *A, int totMinutos);

II-3 (3 valores) Implemente uma função que remova todas as folhas de uma árvore.

Protótipo: NodoAB* cortaFolhas(NodoAB *A);

II-4 (3 valores) Implemente uma função que compare 2 árvores binárias e confirme se estas têm igual estrutura, isto é, independentemente do conteúdo de cada nó, o total de nós e a sua organização em ambas as árvores é ou não igual (1=sim, 0=não).

Protótipo: int igualEstrutura(NodoAB *A1, NodoAB *A2);

(Grupo II - Grafos)

2) Considere a seguinte estrutura de dados relativa à implementação de um grafo, utilizando listas de adjacência. O grafo serve para registar as ligações de "Amigos" na rede social "*LivroDeCaras*", cuja característica mais distintiva é o facto das relações de amizade poderem não ser bidireccionais, isto é, "X" ser amigo de "Y", mas o contrário não se verificar. Assim, no respectivo grafo, uma aresta "X→Y" significa que "X" é amigo de "Y".

```
typedef struct AMIGO{
        int verticeAmigo;
        int custo;    //a utilizar na alínea c)
        struct AMIGO *nseg;
} Amigo;
```

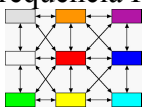
2a) (3 valores) Implemente uma função que imprima o ID das pessoas (vértices) que são amigas de todas as outras pessoas registadas.

Protótipo: void amigoTodos(Amigo **G, int tv);

2b) (3 valores) Implemente uma função que verifique se existem relações de amizade repetidas, isto é, casos em que "X→Y" mais que uma vez. A função deve retornar "1" em caso afirmativo ou "0" caso contrário.

Protótipo: int amigosRepetidos(Amigo **G, int tv);

2c) (3 valores) Implemente uma função que imprima o ID dos amigos de grau até "n" de um determinado elemento. Por exemplo, no caso de "X→Y" e "Y→Z", considera-se que "X" é amigo de "Y" em grau 1 e amigo de "Z" em grau 2.



Considere que tem à sua disposição a seguinte função:

int** Dijkstra(Amigo **G, int tv, int vo, int vd);

Esta função devolve a tabela que resulta da execução do algoritmo de Dijkstra, isto é, uma matriz com “tv” linhas e 3 colunas, em que a primeira coluna descreve os vértices explorados, a segunda o respectivo custo e a terceira o predecessor de cada vértice (“-1” indica “sem predecessor”).

Protótipo: void mostraAmigosGrauN(Amigo **G, int tv, int codigoPessoa, int grauN);