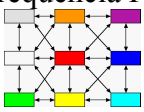
Estruturas de Dados
2014/15
Frequência F2



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 direita.

II-2 (2 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)

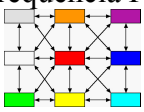
Considere a seguinte estrutura de dados, utilizada para guardar informação sobre um grafo:

```
typedef struct NODO{
        int VD;             //vértice destino
        int custo;          //custo da aresta
        struct NODO *nseg;
}Nodo;
```

II-1 (2 valores) Implemente uma função que remova os lacetes do grafo. Considere que tem à sua disposição a função “Nodo* removeElemento(Nodo *L, Nodo *e)”, que remove da lista “L” o elemento apontado por “e”.

Protótipo: void removeLacetes(Nodo **G, int tv);

II-2) (2 valores) Implemente uma função que receba um grafo e informação sobre dois vértices (origem e destino). A função deverá devolver “1” se for possível fazer o caminho entre os vértices origem e destino no



máximo em dois passos, isto é, através de (no máximo) 2 arestas.

Protótipo: int caminhoPossivel2Arestas(Nodo **G, int tv, int vo, int vd);

II-3) (3 valores) Implemente uma função que devolva um vector com os vértices do grafo que são simultaneamente adjacentes a todos os vértices especificados por parâmetro da função.

Protótipo: int* adjacentesTodos(Nodo **G, int tv, int *teste, int totTeste, int *tadj);

Exemplo: adjacentesTodos(G, tv, [1,2,3], 3, &tadj) deverá devolver os vértices do grafo que forem simultaneamente adjacentes aos vértices "1", "2" e "3" ("tadj" deverá devolver o total de vértices que satisfazem a condição).

II-4) (3 valores) Suponha que o grafo definido é utilizado para registar os pagamentos entre sócios de um clube. Neste caso, uma aresta "vo→vd" indica que o sócio "vo" pagou ao sócio "vd" um valor especificado pelo respectivo custo da aresta. Implemente uma função que liste o ID do sócio (vértice) que mais enriqueceu, isto é, onde a diferença entre o que recebeu e o que pagou for máxima.

Protótipo: void listaSocioRico(Nodo ** G, int tv);