MINISTÉRIO DA EDUCAÇÃO
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO PIAUÍ
CURSO : Analise e desenvolvimento de Sistemas
DISCIPLINA : Estrutura de Dados 2

Nome:_____

PROVA NO 2

dia 19.09.2024

1. São VERDADEIRAS as seguintes afirmativas: (1.0 pt)

<A> O processo de busca em um árvore é muito mais rápido do que o processo de busca em listas ligadas. No entanto, quando a árvore encontra-se ASSIMETRICA a eficiência do processo de busca pode ser comprometida.

<B> Uma árvore AVL é aquela na qual as alturas das subárvores esquerda e direita de cada nó diferem no máximo por um.

<C> Para calcular a quantidade total de nós em um árvore cheia, é possível elaborar o cálculo da quantidade total de nós a partir da altura da árvore.

<D> Uma árvore é balanceada se a diferença na altura de ambas as subárvores de qualquer nó na árvore é maior do que 1 (um).

<E> O algoritmo de Morris é aplicado para balancear uma árvore binária, se baseia no fato de que o percurso in-order é muito simples para árvores degeneradas, nas quais nenhum nó tem filhos à esquerda.

a. (x) <A>, <B>, <C>
b. ( ) <A>,<B>, <E>
c. ( ) <A>, <B>
d. (x) <A>, <B>, <C>
e. ( ) NDA

2. Sobre o código abaixo, é correto afirmar que: (2.0 pts)

```
template<class T>
void balancear(T vetor[], int first, int last){
    if (first<=last) {
        int middle = (first + last)/2;
        insert(vetor[middle]);
        balancear(vetor,first,middle-1);
        balancear(vetor,middle+1,last);
    }
}
```

a. ( ) No algoritmo acima a estrutura "vetor" não está ordenada, o algoritmo “balancear” usa a estratégia de ordenar o “vetor” e somente, após isto, realiza o balanceamento da árvore.

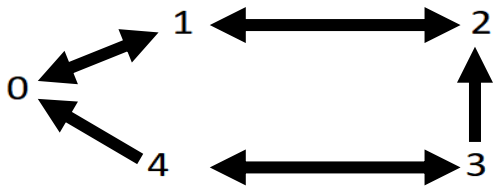
b. ( ) A estratégia de balanceamento do algoritmo corresponde ao algoritmo AVL.

c. ( ) A estratégia de balanceamento do algoritmo corresponde ao algoritmo DSW.

d. (x) O algoritmo pode ser impróprio quando a árvore tem que ser usada enquanto os dados nela a ser incluídos ainda estão chegando.

3. Marque a alternativa FALSA: (2.0 pts)

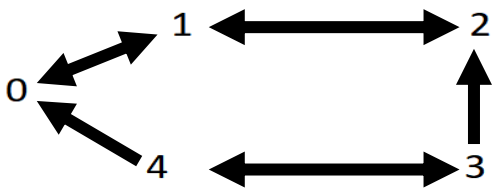
<A> O grafo abaixo é simétrico: f



<B> Um grafo simétrico e sem laços é chamado “grafo não-orientado”. v

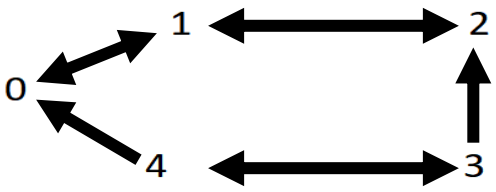
<C> Um grafo pode ser representado através das seguintes estruturas de dados: matriz de adjacências e listas de adjacências. V

<D> O grafo abaixo é um grafo orientado: v



<E> A matriz de adjacências utiliza um vetor de duas dimensões V, tal que ADJ[v,w]=1, se existe uma aresta v->w em no grafo G. V

<F> O grafo abaixo: F



Pode ser representado através de uma lista de adjacências da seguinte maneira:

0 -> 1

1 -> 0 -> 2

2 -> 1 ->3

3 -> 2 -> 4

4-> 0 -> 3

a.( ) <A>, <D> c.( ) <A>,<D>,<F>

b.(x) <A>, <F> d.( ) <B>,<C>,<F> e.( ) NDA

4. Sobre árvores AVL, considere as seguintes operações de rotação para balancear a árvore AVL: (2.0 pts)

I. Rotação simples à direita (RR).
II. Rotação simples à esquerda (RL).
III. Rotação dupla à direita (DRR).
IV. Rotação dupla à esquerda (DRL).

Dado o seguinte trecho de pseudocódigo para uma inserção em uma árvore AVL:

```
função inserir_avl(T, chave)
        se T é vazia
           criar novo nó com chave
      senão se chave< T.chave
              T.esquerda = inserir_avl(T.esquerda, chave)
              se |altura(T.esquerda) - altura(T.direita)| > 1
                 realizar operação de rotação necessária
      senão se chave> T.chave
               T.direita = inserir_avl(T.direita, chave)
```

se |altura(T.esquerda)- altura(T.direita)| > 1
realizar operação de rotação necessária

Qual das seguintes opções descreve corretamente quando a rotação simples à direita (RR) deve ser aplicada durante a inserção?

a.( ) A rotação simples à direita (RR) não é usada durante a inserção em árvores AVL.

b.(x) Quando a chave é inserida na subárvore esquerda do filho esquerdo do nó desbalanceado.

c.( ) Quando a chave é inserida na subárvore direita do filho esquerdo do nó desbalanceado.

d.( ) Quando a chave é inserida na subárvore esquerda do filho direito do nó desbalanceado.

e.( ) Quando a chave é inserida na subárvore direita do filho direito do nó desbalanceado.

5. Árvores B são largamente utilizadas na construção de índices em implementações de bancos de dados. Considere as seguintes afirmativas sobre esse tipo de organização: (2.0 pts)

I. Há apenas um nó raiz.

II. O algoritmo de remoção de uma chave NÃO preserva o balanceamento da árvore, o que é feito periodicamente nos bancos de dados por meio de um processo de limpeza dos índices.

III. O algoritmo de inserção preserva o balanceamento da árvore, criando novos nós e alterando a estrutura da árvore quando necessário.

IV. Numa tabela de banco de dados onde a chave de indexação é composta por mais de uma coluna, a ordem dessas colunas no comando de criação do índice é irrelevante.

Assinale se:

a.( ) todas as afirmativas estão corretas;

b.( ) somente as afirmativas I, II e IV estão corretas;

c.( ) somente as afirmativas II e III estão corretas;

d.(X) somente as afirmativas I e III estão corretas;

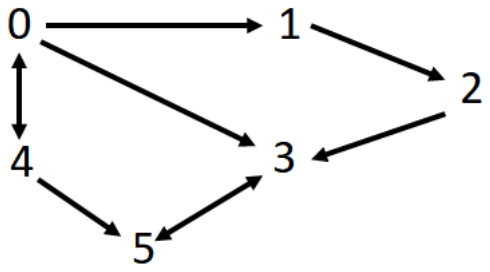
e. ( ) nenhuma das alternativas está correta.

6.Sobre o código abaixo: (2.0 PTS)

```
void y(lista l){
    if (l=NULL) return;
   y(l->prox);
   print(l->item);

}
void x(int v,int w, lista Adj[]){
                    fila f = fila(100); // tamanho depende do grafo!!
                    enfileira(no(v,NULL),f);
                    while (!vaziaf(f)){
                              lista c = desenfileira(f);
                              if (c->item==w){
                                        y(c);
                                }
                       else {
                         for (lista s=adj[c->item];s;s=s->prox)
                                   if (!pert(s->item,c))
                                      enfileira(n(no->item,clone(c)),f);
                              destroi(&c);
                              }
            }
          destroi(&f);
 }
```

Considerando que a função “pert” procura a ocorrência de um item (“s->item”) em uma lista (“c”).
Considerando que “adj” armazena a lista de adjacência do grafo abaixo, aplicando a função “x” com os seguintes parâmetros: “x(0,5,adj)” o resultado obtido será:



a.(x) Obtém as 3 (três) listas abaixo, **respectivamente:**
[5,3,0] e exibe seu inverso [0,3,5]
[5,4,0] e exibe seu inverso [0,4,5]
[5,3,2,1,0] e exibe seu inverso [0,1,2,3,5]

b. ( ) Obtém a lista [5,3,0] e exibe seu inverso [0,3,5] .

c. ( ) Obtém a lista [5,3,0] e exibe [5,3,0].

d.( ) NDA