AV2 - Estrutura de Dados - 25.1 (Manhã)

Nome: _____

Parte 1: Questões Objetivas

1) Denomina-se _____ de um nodo de uma árvore o número de subárvores subordinadas diretamente a ele.

a) altura

b) profundidade

c) caminho

d) nível

e) grau (X)

2) Na definição de árvore, não há relação entre número de subárvores e grau.

a) Certo (X)

b) Errado

3) Estrutura de nós com atributos esquerda e direita refere-se a:

a) Árvores de busca binárias

b) Pilhas

c) Filas

d) Listas ligadas

e) Árvores binárias (X)

4) Qual árvore binária pode ser classificada como ABB? *(imagem)*

5) Árvore B possui estrutura:

a) cada nó tem máx d-1 filhos

b) minimiza tempo de acesso (X)

c) folhas em 3 níveis

d) cada nó tem máx 2d-1 filhos

6) Definição incorreta sobre árvores:

a) conjunto finito de vértices

b) grau = número de subárvores

c) nó sem subárvores é folha

d) Altura de v é número de nós da raiz até v (X)

e) nível de v é número de nós do caminho raiz até v

7) Repetição da questão anterior — incorreta:

d) Altura de v é número de nós da raiz até v (X)

8) Pós-ordem resultou em: 41 44 33 47 55 52 36 30 — qual árvore? *(imagem)*

9) Inserção AVL 5,10,12,8,7,11,13 — pré-ordem:

c) 5,7,8,10,11,12,13 (X)

10) Percurso in-order da árvore mostrada: *(imagem)*

11) Número mínimo de rotações para balancear ABB: *(imagem)*

12) CONSULTA(x): percorre direita → retorna:

c) valor máximo (X)

13) Busca em amplitude da árvore: *(imagem)*

14) Árvore B após inserir 11: *(imagem)*

15) Diferença de alturas = 2 e FE = 1 → rotação:

b) rotação à direita (X)

16) KD-Tree:

c) alterna comparações entre x e y por nível (X)

17) Treap combina:

b) ordem BST + prioridade heap (X)

18) Altura da Patricia:

b) 3 (X)

19) Hash h(k)=k mod 5:

b) I e II apenas (X)

20) Árvore B após inserir 5: *(imagem)*

21) Quantos nós folha? *(imagem)*

22) Inserção em min-heap:

c) insere como folha e sobe (X)

23) Árvore splay:

c) move nó acessado para raiz (X)

24) V/F:

Resposta: c) V - V - F (X)

25) Árvore de Merkle:

c) raiz verifica integridade (X)

26) B, B*, B+ e 2-3-4:

c) I, II e III corretas (X)

27) Árvores BSP:

d) todas corretas (X)

Parte 2: Questões Discursivas e Práticas

QUESTÃO 2)

Merkle: incluir A,B,C,D,E,F,G; remover C

Treap: incluir (A,7),(B,4),(C,8),(D,55),(E,94),(F,82),(G,46),(A,16); remover B e nó 4

Splay: inserir 10,30,20,5,40,25,85; remover 25

BSP: sala 2D, objetos A-F nas coordenadas

Heap: inserir D,E,R,J,C,K,A,B — mostrar heap

PATRICIA: incluir rom, roupa, rose, rotor, rock, amor, anca, amo; remover amor

2-3-4: inserir 10–90; remover 10

QUESTÃO 3)

Função somaFolhas:

```
struct noArv {
   int info;
   struct noArv* esq;
   struct noArv* dir;
};

int somaFolhas(NoArv* raiz) {
   if (!raiz) return 0;
   if (!raiz->esq && !raiz->dir)
      return raiz->info;
   return somaFolhas(raiz->esq) + somaFolhas(raiz->dir);
}
```

QUESTÃO 4)

Função NivelDaChave:

```
int NivelDaChave(NoB* raiz, int chave) {
   if (!raiz) return -1;
```

```
    for (int i = 0; i < raiz->qtOcupados; i++) {
        if (raiz->vChaves[i] == chave)
            return 0;
        if (chave < raiz->vChaves[i]) {
            int n = NivelDaChave(raiz->vLinks[i], chave);
            return n >= 0 ? n + 1 : -1;
        }
    }
    int n = NivelDaChave(raiz->vLinks[raiz->qtOcupados], chave);
    return n >= 0 ? n + 1 : -1;
}
```

QUESTÃO 5)

AVL: inserir 30,20,10,40,50,25,5,55,28; remover 40 e 50

QUESTÃO 6) 1,5 pontos Utilize o espaço adequado do caderno de resposta para mostrar a B-Tree de ordem 3, resultante após a inserção de cada um dos números 10, 20, 5, 6, 12, 1, 8, 15 (nesta ordem). Escreva uma função chamada percursoLargura que receba a raiz de uma árvore binária de busca (ABB) e exiba os valores dos nós em ordem de nível (também conhecido como percurso em largura). A função deve utilizar uma fila auxiliar para controlar os nós a serem visitados. Considere que o TAD fila existe: typedef struct Fila {

```
No* dados[100];
 int ini, fim;
 } Fila;
void inicializaFila(Fila* f) {
 f->ini = f->fim = 0;
}
int filaVazia(Fila* f) {
 return f->ini == f->fim;
}
void enfileira(Fila* f, No* no) {
f->dados[f->fim++] = no;
}
No* desenfileira(Fila* f) {
 return f->dados[f->ini++]; }
```

Exemplo de chamada: percursoLargura(raiz);

Implemente uma função chamada ehAVL que receba a raiz de uma árvore binária e retorne 1 se ela for uma árvore AVL válida (isto é, balanceada com fator de equilíbrio entre -1 e 1 em todos os nós), ou 0 caso contrário.A função deve realizar verificações recursivas para checar se a propriedade de balanceamento é mantida em toda a árvore.

```
int altura(No* raiz) {
if (!raiz) return 0;
 int he = altura(raiz->esq);
int hd = altura(raiz->dir);
return (he > hd ? he : hd) + 1;
}
```

QUESTÃO 7)

Função ContaChavesBTree:

```
int ContaChavesBTree(BTreeNode* raiz) {
   if (!raiz) return 0;
   int total = raiz->n;
   if (!raiz->folhas)
      for (int i = 0; i <= raiz->n; i++)
         total += ContaChavesBTree(raiz->filhos[i]);
   return total;
}
```