

**ALUNO: Dimas Curti De Almeida Junior**

**Lista 0 – (revisão árvore binária) – Complexidade de Algoritmos**

**Professora: Elvira Padua Lovatte**

**Cursos: Ciência da Computação e Engenharia de Computação** Data de entrega: 13/03

Assunto: Análise da estrutura árvore binária

1. Qual a altura máxima e mínima de uma árvore com 100 nós?

Log (100) = 6, altura mínima: 6

n-1, altura máxima: 99

1. Qual é o menor número de níveis que uma árvore binária com 67 nós podemos apresentar?

Log (67) = 6, nível: 6

1. Considerando o nível que contem a raiz de uma árvore binária como sendo o nível 0, determine o número mínimo e máximo de nós no nível 5.

n = 5, h = 5, 2^h 🡺 2^5 = 32 que será o máximo de nós no nível 5.

mínimo de nós no nível 5 será 1.

1. Uma árvore binária completa com n nós folhas contém quantos nós?

Nos: n\*2-1

1. Dada uma ABB inicialmente vazia, insira (E DESENHE) os seguintes elementos (nessa ordem): M, F, S, D, J, P, U, A, E, H, Q, T, W, K.
2. Dada uma ABB inicialmente vazia, insira (E DESENHE) os seguintes elementos (nessa ordem): A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z. O que se pode observar?

Podemos observar que ela se torna uma lista encadeada.

1. Escreva uma função recursiva que conte o número de nós de uma árvore binária. Entrada: estrutura árvore em que cada nó é um item com 3 campos Saída: quantidade de vértices.

public int TotCentral() {

int n = 0;

return TotCentral(this.raiz, n);

}

private int TotCentral(No arv, int n) {

if (arv != null) {

n = TotCentral(arv.getEsq(), n);

n++;

n = TotCentral(arv.getDir(), n);

}

return n;

}

1. Considerando uma árvore de busca com n nós, qual é a relação entre o número de comparações (entre a chave procurada e chaves em nós) e a altura da árvore?

Escreva um algoritmo que determine se uma árvore binária é completa ou não.

Entrada: estrutura árvore em que cada nó é um item com 3 campos

Saída: Sim ou Não

public boolean[] ComPre() {

boolean[] b = new boolean[1];

b[0] = true;

int[] n = new int[1];

n[0] = 0;

int[] f = new int[1];

f[0] = 0;

return ComPre(this.raiz, n, b, f);

}

private boolean[] ComPre(No arv, int[] n, boolean[] b, int[] f) {

if (arv != null) {

if (arv.getEsq() == null && arv.getDir() == null) {

if (f[0] == 0) {

f[0] = n[0];

} else {

if (f[0] == n[0] && b[0] == true) {

b[0] = true;

} else {

b[0] = false;

}

}

}

n[0] = n[0] + 1;

ComPre(arv.getEsq(), n, b, f);

ComPre(arv.getDir(), n, b, f);

n[0] = n[0] - 1;

}

return b;

}

1. Escreva um algoritmo que conte o número de folhas de uma árvore binária. Entrada: estrutura árvore em que cada nó é um item com 3 campos Saída: quantidade de vértices folhas.

public int[] FolhaPre (){

int []n= new int[1];

n[0]=0;

return FolhaPre (this.raiz, n);

}

private int[] FolhaPre (No arv, int[] n){

if (arv != null) {

if(arv.getEsq()==null&&arv.getDir()==null){

n[0]++;

}

FolhaPre (arv.getEsq(), n);

FolhaPre (arv.getDir(), n);

}

return n;

}

1. Escreva um algoritmo para calcular a altura de uma árvore binária.

Entrada: estrutura árvore em que cada nó é um item com 3 campos.

Saída: altura da árvore.

public int[] AltPre() {

int[] n = new int[1];

n[0] = 0;

int[] f = new int[1];

f[0] = 0;

return AltPre(this.raiz, n, f);

}

private int[] AltPre(No arv, int[] n, int[] f) {

if (arv != null) {

if (arv.getEsq() == null && arv.getDir() == null) {

if (f[0] < n[0]) {

f[0] = n[0];

}

}

n[0] = n[0] + 1;

AltPre(arv.getEsq(), n, f);

AltPre(arv.getDir(), n, f);

n[0] = n[0] - 1;

}

return f;

}