Prof. Igor Machado Coelho
LISTA 2 – Árvores

Nome: Eloyse Fernanda da Silva Costa

1. Considere uma árvore binária completa composta pelos seguintes elementos (representação sequencial): 10,20,15,12,8,5,7, 1 e 2.

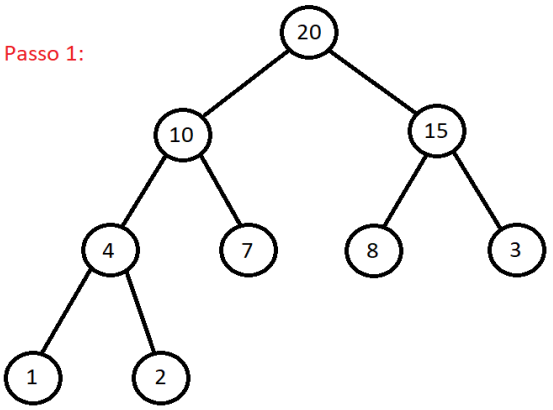
(a) Apresente o percurso de pré-ordem na árvore
Resposta: 10, 8, 5, 1, 2, 7, 20, 15, 12

(b) Apresente o percurso em-ordem na árvore
Resposta: 1, 2, 5, 7, 8, 10, 12, 15, 20

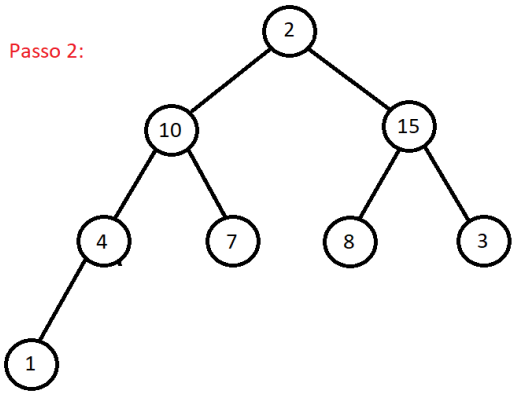
(c) Apresente o percurso de pós-ordem na árvore
Resposta: 2, 1, 7, 5, 8, 12, 15, 20, 10

2. Considere uma estrutura MAX-heap representada pelo seguinte vetor de níveis: 20, 10, 15, 4, 7, 8, 3, 1, 2
(a) efetue a remoção do elemento de maior prioridade: desenhe a árvore vetor passo-a-passo

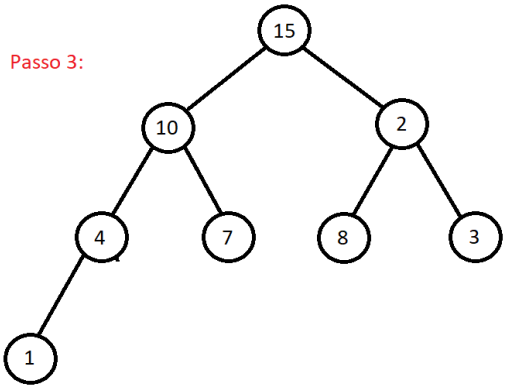
Passo 1:



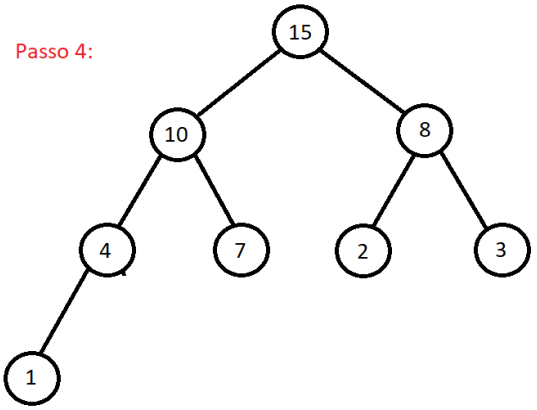
Passo 2:

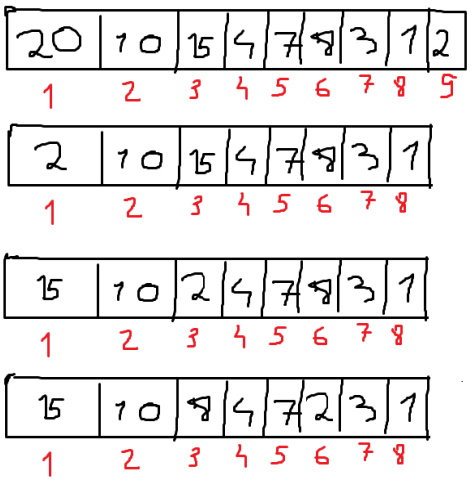


Passo 3:

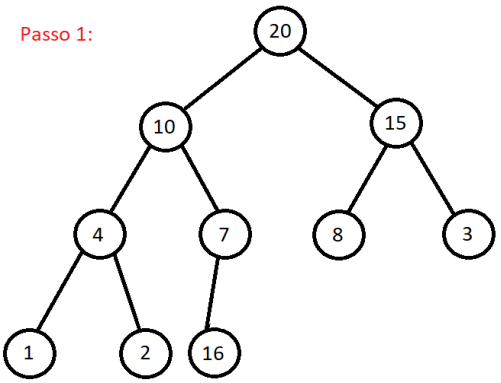


Passo 4:

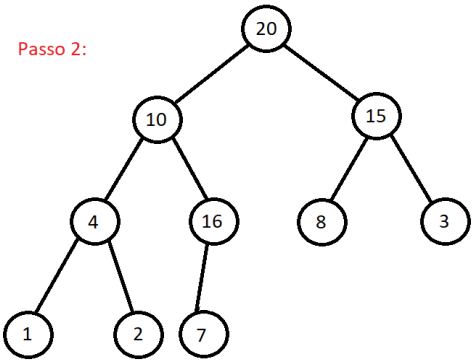




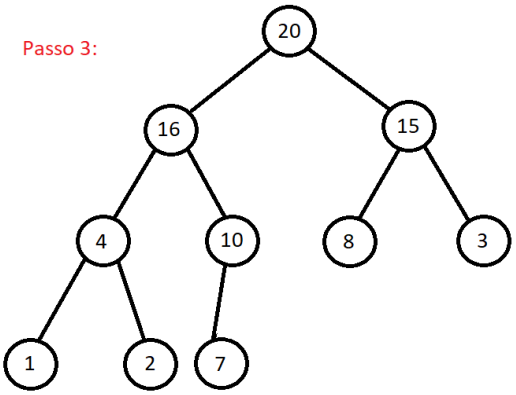
(a) efetue a inserção do elemento 16 (sem considerar a remoção anterior): desenhe a arvore e vetor passo-a-passo

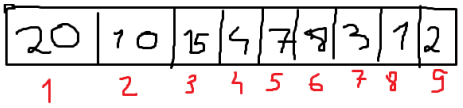


Passo 2:

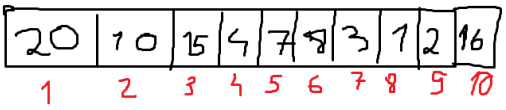


Passo 3:

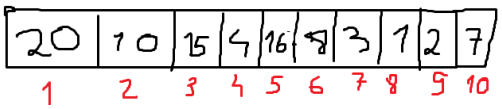




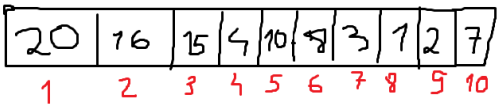
1º etapa



2º etapa



3º etapa



4º etapa

3. Considere a seguinte estrutura para uma árvore binária:

```
typedef struct node{
   int info;
   struct node *left, *right;
}Node;

//Essa foi a estrutura que usei em c
```

(a) Escreva um algoritmo para computar a soma das folhas

```
int somaDaArvore(Node* root){
   if(root == NULL)
       return 0;
   return (root->info + somaDaArvore(root->left) + somaDaArvore(root->right));
}
```

(b) Escreva um algoritmo para efetuar um percurso de pós-ordem

```
void posOrdem(Node* root){
        if(root == NULL)
            return;
        posOrdem(root->left);
        posOrdem(root->right);
        printf("%d ", root->info);
}
```

(c) Escreva um algoritmo para efetuar um percurso de em-ordem

```
void emOrdem(Node* root){
   if(root == NULL)
       return;
   emOrdem(root->left);
   printf("%d ", root->info);
   emOrdem(root->right);
}
```

(d) Escreva um algoritmo para efetuar um percurso de pré-ordem

Resposta:

```
void preOrdem(Node* root){
   if(root == NULL)
       return;
   printf("%d ", root->info);
   preOrdem(root->left);
   preOrdem(root->right);
}
```

(e) Escreva um algoritmo para computar a altura de um dado nó

```
Node* buscarNo(Node* root, int value){
   if(root == NULL)
       return NULL;
   else{
       if(root->info == value)
           return root;
       else{
           if(value < root->info)
               return buscarNo(root->left, value);
           else
               return buscarNo(root->right, value);
       }
   }
}

int altura(Node* root){
   if(root == NULL || root->left == NULL && root->right == NULL)
       return 0;
   else{
       int altura_esquerda = altura(root->left) + 1;
       int altura_direita = altura(root->right) + 1;

       if(altura_esquerda > altura_direita)
           return altura_esquerda;
       else
           return altura_direita;
   }
```

```
}
int alturaArv(Node *root, int value){
    Node *node = buscarNo(root, value);
    if(node){
        return altura(node);
    }else
        return -1;
}
```

(f) Escreva um algoritmo para computar o fator de balanceamento de um dado nó

```
int balanceamento(Node *root){
    if(root){
        return altura(root->right) - altura(root->left);
    }else{
        return 0;
    }

}
```

(g) Escreva um algoritmo para percorrer a árvore em níveis

```
int altura(Node* root){
    if(!root)
        return 0;
    else{
        int altura_esquerda = altura(root->left);
        int altura_direita = altura(root->right);
        if(altura_esquerda >= altura_direita)
            return altura_esquerda+1;
        else
            return altura_direita+1;
    }
}


void percorreNiveis(Node* root, int nivel){
    if(root == NULL)
        return;
```

```
    if(nivel == 0)
        printf("%d -> ", root->info);
    else{
        percorreNiveis(root->left, nivel - 1);
        percorreNiveis(root->right, nivel - 1);
    }
}

void escreveEmNivel(Node* root){
    int altura1 = altura(root);
    for(int i = 0; i < altura1; i++) {
        printf("Nivel %d: ", i);
        percorreNiveis(root, i);
        printf("\n");
    }
}
```

(h) Escreva um algoritmo para computar o produto dos nós

```
int produto(Node* root){
    if(root == NULL)
        return 1;
    return (root->info * produto(root->left) * produto(root->right));
}
```