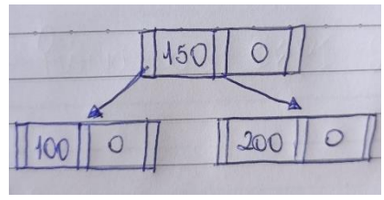
Universidade Federal do Piauí – UFPI
Campus Senador Helvídio de Barros – CSHNB
Curso de Sistemas de Informação Bloco: IV
Disciplina: Estruturas de Dados II
Professora: Juliana Oliveira de Carvalho
Acadêmico: Dayan Ramos Gomes Matrícula: 20209001285

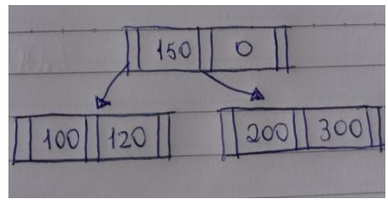
ATIVIDADE DE FIXAÇÂO 5

1. Descreva passo a passo usando texto, desenho e apresente as partes do código utilizadas para inserir elementos em uma árvore 2-3 para as seguintes sequências:

a) Sequência: 100, 150, 200, 300, 120

Para inserir o cem, o algoritmo entra na parte 1 do código e insere o 100, para inserir o 150 o algoritmo entra na parte 2 e na parte 3 do código, onde chama a função adiciona que irá adicionar o 150 na posição 2 da raiz, para inserir o 200 o algoritmo entra na parte 2 e na parte 4 do código, pois como não há espaço na raiz para o 200, esse nó dá raiz será quebrado, onde o 150 irá subir e o 200 colocado em um novo nó e como a raiz não tem pai, entra na parte 5 e o 200 é inserido. Assim, para inserir o 300, o algoritmo entra na parte 7 do código e chama a função recursiva para o nó do centro da raiz, assim o algoritmo irá na segunda chamada entrar na parte 2 e 3 do código e adicionar o 300 na segunda posição do nó em que está o 200. Para inserir o 120, o algoritmo entra na parte 7 do código e entra no primeiro *if* chamando a função recursiva passando o nó da esquerda da raiz, assim, na segunda chamada da função o algoritmo irá entrar na parte 3 e 3 do código e adicionar o 120 na segunda posição do nó onde está o 100.

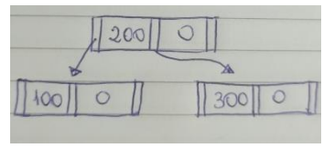


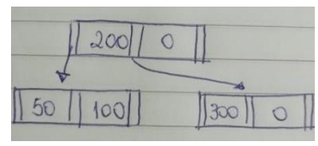


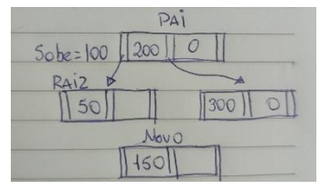

```
1 if (*Raiz == NULL)
    *Raiz = criaNo(valor, NULL, NULL, NULL);
  else
  {
2   if (ehFolha(*Raiz) == 1)
    {
3     if ((*Raiz)->NInfos == 1)
      {
        adiciona(Raiz, valor, maiorNo);
      }
4     else // quando não tem espaço
      {
        struct Arv23 *novo;
        novo = quebraNo(Raiz, valor, sobe, maiorNo);
5       if (Pai == NULL)
        {
          struct Arv23 *no;
          no = criaNo(*sobe, *Raiz, novo, NULL);
          *Raiz = no;
        }
6       else
          maiorNo = novo;
      }
    }
```

b) Sequência: 200, 100, 300, 50, 150, 120

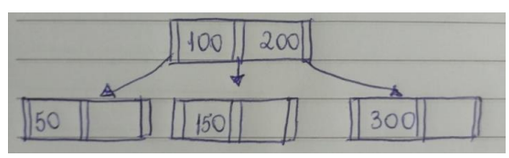
Para inserir o 200, como é o primeiro elemento, o algoritmo entra na parte 1 do código e cria o nó com o elemento 200, para inserir o 100 o algoritmo entra na parte 2 e 3 do código e chama a função *adiciona*, onde lá o 100 será adicionado e como é menor que o 200, vai ser trocado de posição. Para inserir o 300, o algoritmo entra na parte 2 e 4 do código, pois como não há espaço na raiz para o 300, esse nó dá raiz será quebrado, onde o 200 irá subir e o 300 colocado em um novo nó e como a raiz não tem pai, entra na parte 5 e o 200 é inserido. Para inserir o 50, o algoritmo entra na parte 7 do código e entra no primeiro if, fazendo uma chamada recursiva da função passando o nó da esquerda da raiz como a nova raiz, assim, nessa segunda chamada o algoritmo irá entrar na parte 2 e 3 do código e chama a função *adiciona*, onde o 50 vai ser inserido na primeira posição do nó e o 100 passado para segunda posição. Para inserir o 150, o algoritmo irá entrar na parte 7 do código e irá fazer uma chamada recursiva passando o nó da esquerda como a nova raiz e a raiz atual como o pai, assim, nessa segunda chamada da função, o algoritmo irá entrar na parte 4 do código, onde o nó será quebrado e o 100 irá subir e o 150 será colocado em um novo nó que será retornado na recursividade da função, assim, quando retornar, o algoritmo irá entrar na parte 8 e 9 do código, onde o 200 irá para segunda posição e o 100 inserido com o ele, o filho do centro, passa a ser o filho da direita e o novo nó com 150 será o filho do centro da raiz. Para inserir o 120, o algoritmo entra na parte 7 e chama a recursividade para o nó do centro, onde nessa chamada o algoritmo entra na parte 2 e 3, chama a função adiciona que bota o 150 para a segunda posição e insere o 120 na primeira posição deste nó.

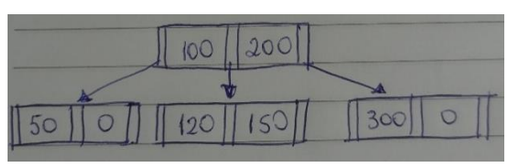






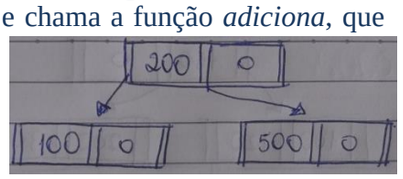

```
7 else{ // quando não é folha
    if (valor < (*Raiz)->Info1)
      maiorNo = insereArv23(*Raiz, &((*Raiz)->esq), valor, sobe);
    else if ((*Raiz)->NInfos == 1 || (valor < (*Raiz)->Info2))
      maiorNo = insereArv23(*Raiz, &((*Raiz)->cen), valor, sobe);
    else
      maiorNo = insereArv23(*Raiz, &((*Raiz)->dir), valor, sobe);

8   if (maiorNo != NULL){
9     if ((*Raiz)->NInfos == 1){
        adiciona(Raiz, *sobe, maiorNo);
        maiorNo = NULL;
      }
10    else{// quando não tem espaço
        int sobe1;
        struct Arv23 *novo;
        novo = quebraNo(Raiz, *sobe, &sobe1, maiorNo);
11      if (Pai == NULL){
          struct Arv23 *no;
          no = criaNo(sobe1, *Raiz, novo, NULL);
          *Raiz = no;
          maiorNo = NULL;}
12      else
          maiorNo = novo;
      }
    }
  }
return maiorNo;
```

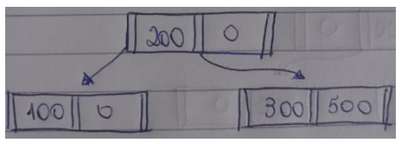


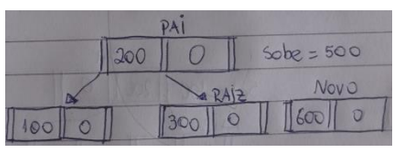


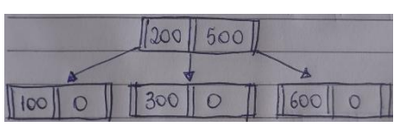
c) Sequência: 100, 500, 200, 300, 600, 450

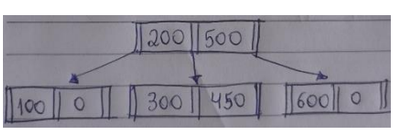
Para inserir o 100, como é o primeiro elemento, o algoritmo entra na parte 1 do código e cria o nó com o elemento 100, para inserir o 500 o algoritmo entra na parte 2 e 3 do código e chama a função *adiciona*, que insere o 500 na segunda posição do nó onde está o 100. Para inserir o 200, o algoritmo entra na parte 2 e 4 do código, pois como não há mais espaço na raiz para o 200, o nó será quebrado, onde o numero 200 irá subir para o nó de cima, o 500 colocado em um novo nó e como a raiz não tem pai será criado um novo nó com o 200, onde o filho da esquerda é o nó com o 100 e o filho do centro é o novo nó com o 500. Para inserir o 300, o algoritmo entra na parte 7 do código e faz uma chamada recursiva para o filho do centro, assim, nessa chamada o algoritmo entra na parte 2 e 3 do código, onde chama a função *adiciona* que vai inserir o 300 na primeira posição do nó e passar o 500 para a segunda posição. Para inserir o 600, o algoritmo entra na parte 7 do código e faz uma chamada recursiva para o filho do centro da raiz, assim, nessa chamada o algoritmo entra na parte 2 e 4 do código, pois como esse nó não tem espaço para o 600 ele será quebrado, onde o valor 500 irá subir para o nó de cima e o valor 600 colocado em um novo nó, como esse nó tem pai, entra na parte 6 e esse novo nó e retornado na recursividade, assim, após a chamada recursiva retornar o novo nó, o algoritmo entra na parte 8 e 9 do código e adiciona o 500 que é o valor que subiu na segunda posição do nó, junto ao 200, e o novo nó com o 600 passa a ser filho da direita da raiz. Para inserir o 450, o algoritmo entra na parte 7 do código e faz uma chamada recursiva para o filho do centro da raiz, assim, nessa chamada, o algoritmo entra na parte 2 e 3 do código e adiciona o 450 na segunda posição do nó, junto ao 300.











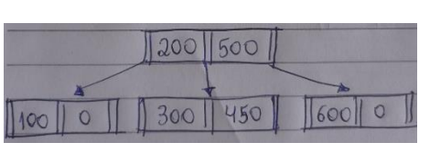
2. Considere a árvore da sequência c) do exercício 1 e responda:

a) Mostre uma possível sequência que gere a mesma árvore.

Sequência: 200, 100, 300, 500, 600, 450

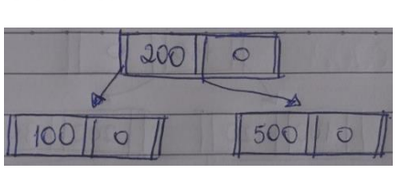
b) O que acontece se inserirmos o número 200.

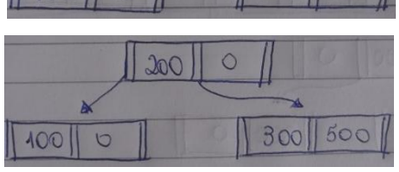
Como o código não é feito para inserir números repetidos, o algoritmo vai entrar na parte 7 do código e o vai entrar no *else* que irá fazer uma chamada recursiva para o filho da direita da raiz, e vai inserir o 200, junto ao 600, assim a arvore já não vai mais obedecer a regra de uma árvore 2 3, tendo assim um erro.



3. Explique com exemplos a função adicionaNo.

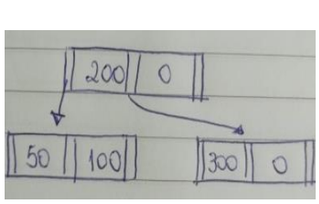
Em um caso que queremos adicionar o número 300 nesta árvore, quando a chamada recursiva estiver no nó onde está o 500, o algoritmo chama a função adiciona passando o valor 300 a ser adicionado, dentro da função adiciona, o algoritmo entra na parte 2 do código, pois o 300 é menor que o valor que esta na primeira posição, assim, o 500 é passado para posição 2, assim como o filho do centro é passado para o filho da direita, o 300 é adicionado na posição 2 e o filho do centro recebe o MaiorNo que neste caso é NULL.

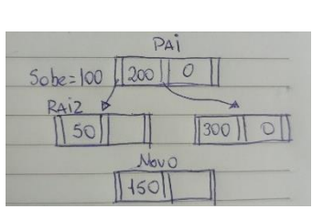


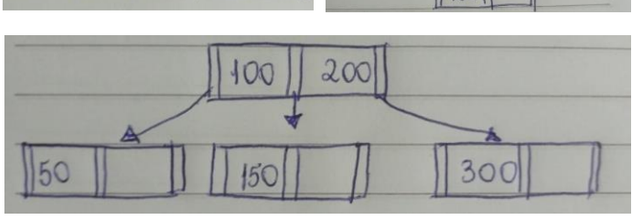



```
void adiciona(struct Arv23 **Raiz, int Valor, struct Arv23
*MaiorNo){
1 if (Valor > (*Raiz)→Info1)
  {
    (*Raiz)→Info2 = Valor;
    (*Raiz)→dir = MaiorNo;
  }
2 else{
    (*Raiz)→Info2 = (*Raiz)→Info1;
    (*Raiz)→Info1 = Valor;
    (*Raiz)→dir = (*Raiz)→cen;
    (*Raiz)→cen = MaiorNo;
  }
  (*Raiz)→NInfos = 2;
}
```

Em outro caso, após tentarmos inserir o 150 no filho da esquerda dessa árvore, o nó é quebrado e o 100 vai ser o valor que vai subir e ser inserido no nó raiz, assim, o algoritmo chama a função adiciona, como o 100 é menor que o 200 o algoritmo irá entrar na parte 2 dessa função e passar o 200 para a segunda posição e seu nó do centro que está com o 300, passa a ser o nó da direita da raiz, adiciona o valor 100 na primeira posição e o maior nó que na função quebraNo estava com o 150, passa a ser o filho do centro da raiz.



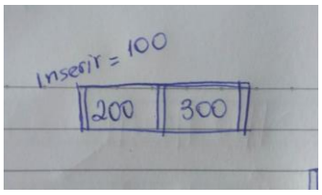


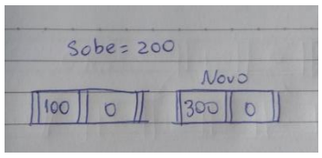


4. Explique com exemplos a função quebraNo

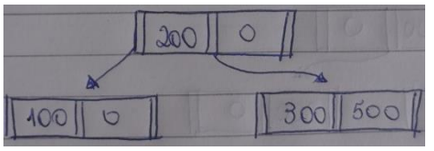
```
struct Arv23 *quebraNo(struct Arv23 **Raiz, int valor, int
*sobe, struct Arv23 *MaiorNo){
  struct Arv23 *Novo;

1 if (valor < (*Raiz)->Info1){
    *sobe = (*Raiz)->Info1;
    Novo = criaNo((*Raiz)->Info2, (*Raiz)->cen, (*Raiz)->dir,
    NULL);
    (*Raiz)->Info1 = valor;
    (*Raiz)->cen = MaiorNo;
  }
2 else if (valor < (*Raiz)->Info2){
    *sobe = valor;
    Novo = criaNo((*Raiz)->Info2, MaiorNo, (*Raiz)->dir, NULL);
  }
3 else{
    *sobe = (*Raiz)->Info2;
    Novo = criaNo(valor, (*Raiz)->dir, MaiorNo, NULL);
  }
  (*Raiz)->Info2 = 0;
  (*Raiz)->NInfos = 1;
  (*Raiz)->dir = NULL;

  return (Novo);
}
```

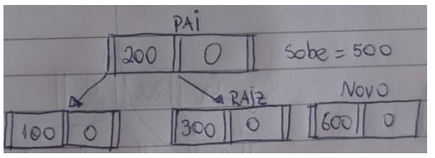
Em um caso que queremos adicionar o 100 nesta árvore, a função *quebraNo* será chamada e o algoritmo irá entrar na parte 1 do código, onde atribui a variável *sobe* o valor do meio, que no caso é o 200, cria um novo nó com o maior valor que é o 300, onde o filho do centro da raiz passa a ser o filho da esquerda desse novo nó e o filho da direita da raiz passa a ser o filho do centro desse novo nó, assim, o 100 vai ser adicionado na posição 1 da raiz e o filho do centro da raiz recebe *MaiorNo* que nesse caso é NULL.

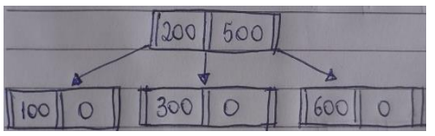




Em outro caso em que queremos adicionar o número 600 no filho do centro da raiz na árvore mostrada a direita, vemos que esse nó não tem espação, assim, a função *quebraNo* será chamada como valor 600 a ser adicionado, o algoritmo irá entrar na parte 3 do código, pois o 600 é maior que os dois valores do nó, assim, a variável *sobe* irá receber o valor da posição 2 que no caso é o 500 e um novo nó será criado com o maior valor que neste caso é o 600, onde o filho da direita da raiz passa a ser o filho da esquerda desse novo nó e o *MaiorNo*, que neste caso é NULL passa a ser o filho do centro desse novo nó.







Lembrando que no final do *quebraNo* o valor da posição 2 da raiz é 0, pois em toda quebra de nó essa posição fica vazia, e o valor de informações do nó também é mudado para 1, pois em todos os casos de quebra a raiz no fim fica com apenas o menor valor dos três valores utilizados