Prova de Estrutura de Dados
Aluna: Livia Barbosa Fonseca

1-)

```c
//Estrutura da lista - possui dois campos: um número e um ponteiro para
a próxima estrutura
typedef struct no{
    int valor;
    struct no *prox;
} no;

//Função que insere o uma chave x em uma lista encadeada
void inserirNaOrdem(no **p, int x){
    //Criando três estruturas auxiliares
    no *atual, *novo, *anterior;

    //Alocando memória para o novo nó da estrutura e passando a chave x
    //para ela
    novo = (no *) malloc(sizeof(no));
    novo->valor = x;

    //A estrutura auxiliar atual recebe a lista de estruturas e a
    //anterior fica vazia
    atual = *p;
    anterior = NULL;

    //Se a lista recebida for vazia
    if(atual == NULL){
        novo->prox = NULL;
        *p = novo;
    //Se a lista não for vazia
    } else{
    //Percorrendo a lista para verificar onde encaixar x de forma
    //crescente
        while(atual != NULL && atual->valor < x){
            anterior = atual;
            atual = atual->prox;
        }
    //Quando a chave x é maior do que o valor analisado da lista saímos
    //do looping anterior
        novo->prox = atual;

       //Inserindo o no novo
```

```
        if(anterior == NULL){
            *p = novo;
        }else{
            anterior->prox = novo;
        }
    }
}
```

2-)

2.1-

```
int somaNo (No *ptraiz){
    //Se a árvore estiver vazia, a soma é zero -- retorna 0
    if(ptraiz == NULL){
      return 0;
    }
    //Calculando recursivamente a soma
    ptraiz->soma = (ptraiz->chave + somaNo(ptraiz->esq) +
somaNo(ptraiz->dir));
    //Retornando a soma
    return ptraiz->soma;
  }
```

2.2-

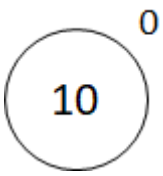
```
 void inserir(No *ptraiz, int x){
    //Se a árvore estiver vazia
    if(ptraiz == NULL){
     //Alocando memória para o novo no da estrutura
      No* novo =  (No*) malloc (sizeof (No));
      //Criando a árvore
      novo->chave = x;
      novo->esq = NULL;
      novo->dir = NULL;
      novo->soma = x;
      //Retornando o no criado
      return novo;

    //Se a árvore não estiver vazia
    }else{
      //Inserindo a chave dada e somando de forma recursiva
      if(x < ptraiz->chave){
        ptraiz->soma += x;
        ptraiz->esq = inserir(ptraiz->esq,x);
      }
```

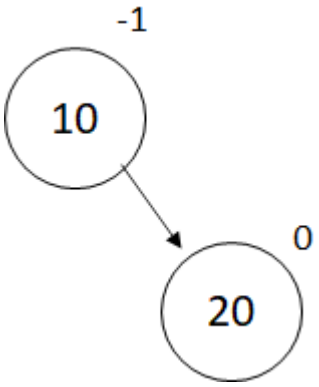
```
      if(x > ptraiz-> chave){
        ptraiz->soma += x;
        ptraiz->dir = inserir(ptraiz->dir, x);
      }
      //Retornando a árvore
      return ptraiz;
    }
  }
```

3-)
3.1- Números: 10,20,15,35,40

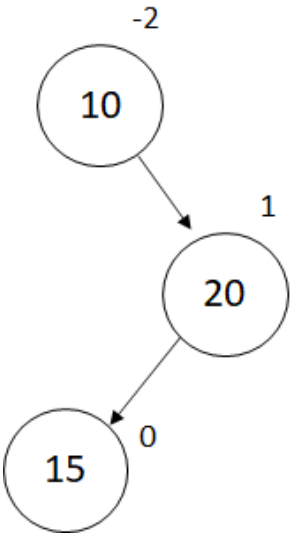
Inserindo o 10:



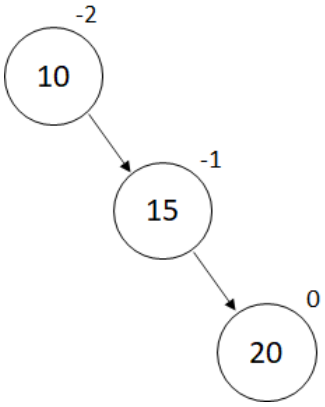
Inserindo o 20:

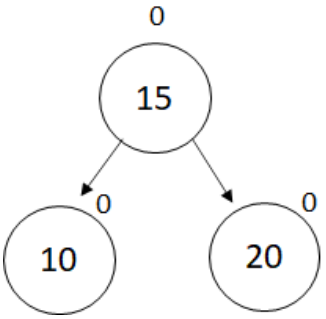


Inserindo o 15:



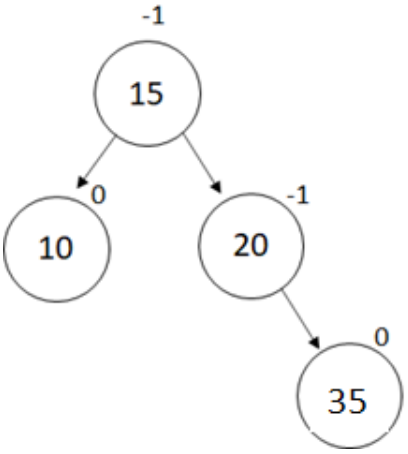
Balanceamento igual a -2, precisamos fazer uma rotação.

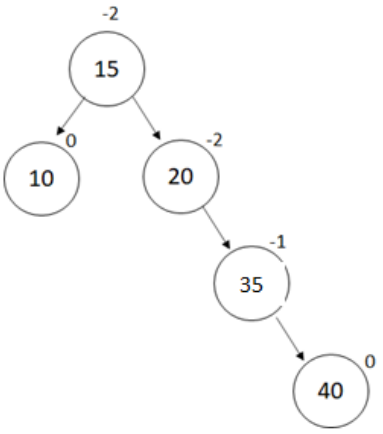




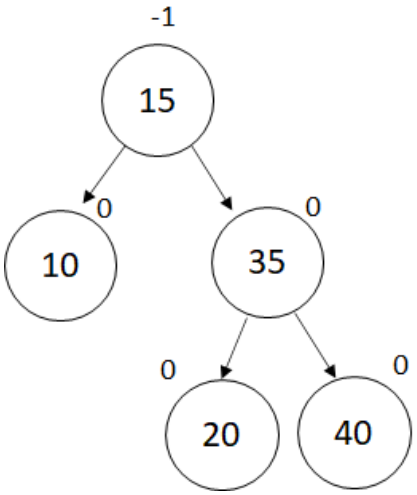
Inserindo o 35:

Inserindo o 40:



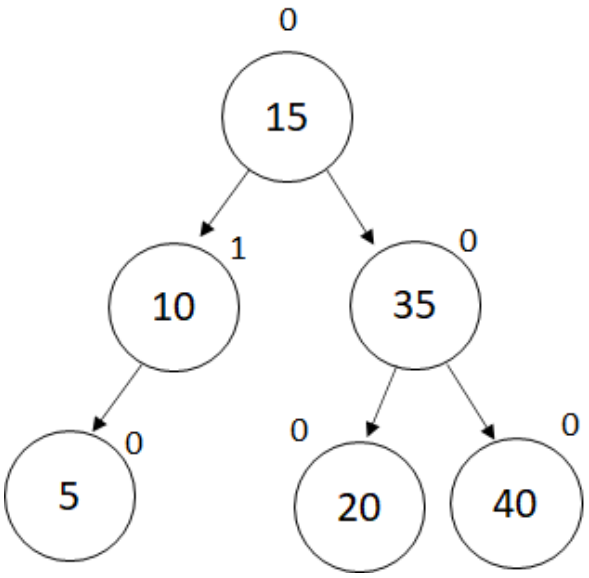


Balanceamento igual a -2, temos que fazer uma rotação.

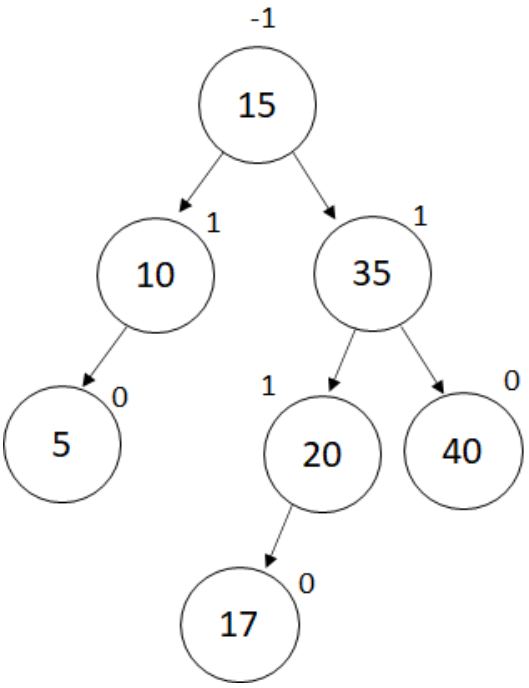


3.2-) Inserir números 5,17,18,19

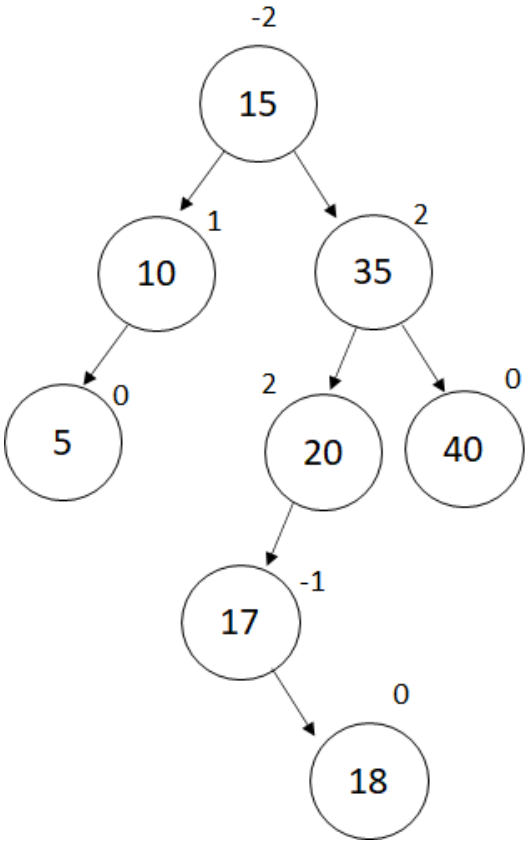
Inserindo o 5:



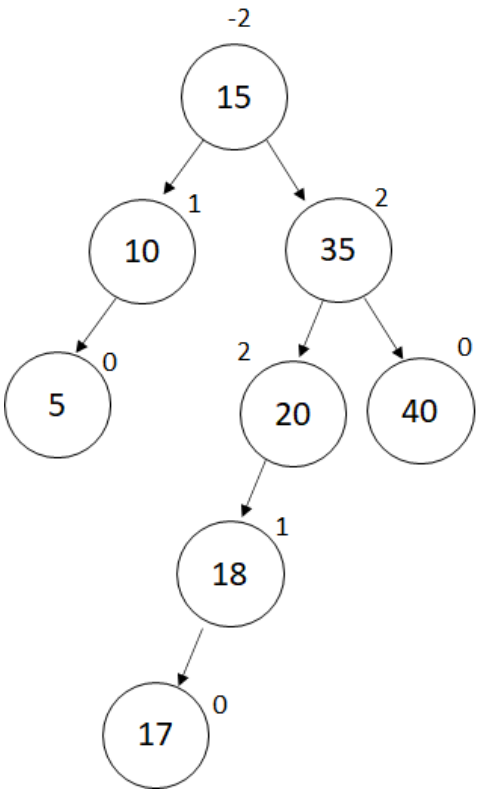
Inserindo o 17:

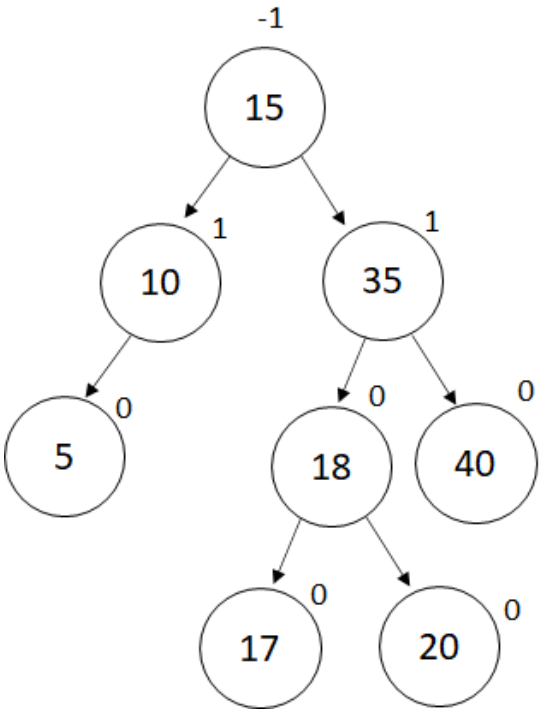


Inserindo o 18:

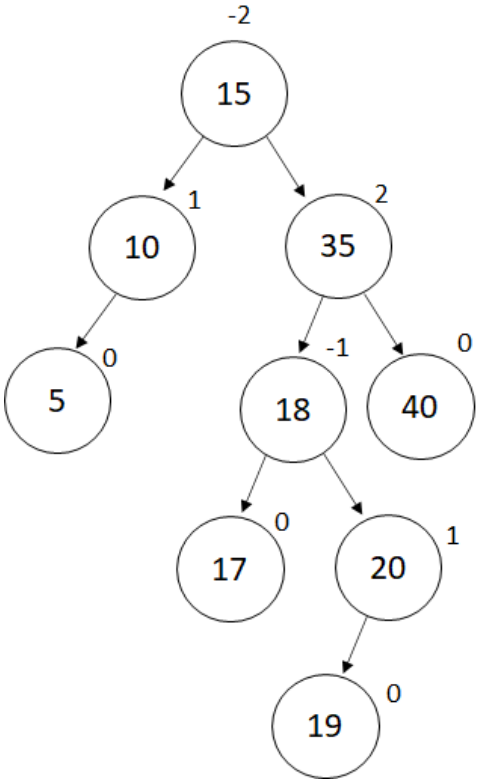


Balanceamento igual a -2 e a 2, temos que fazer uma rotação.

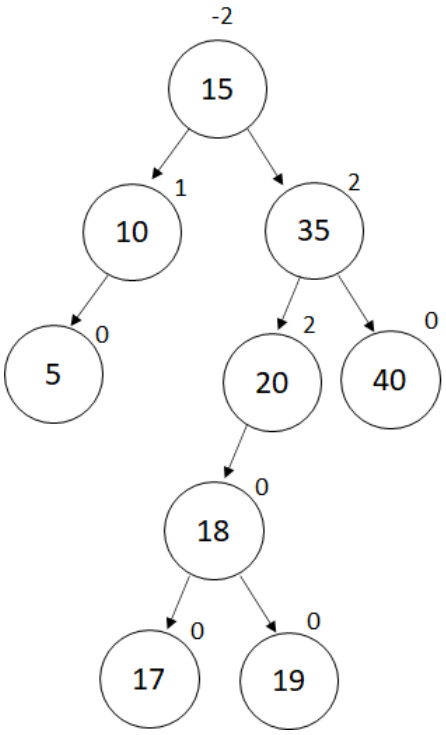


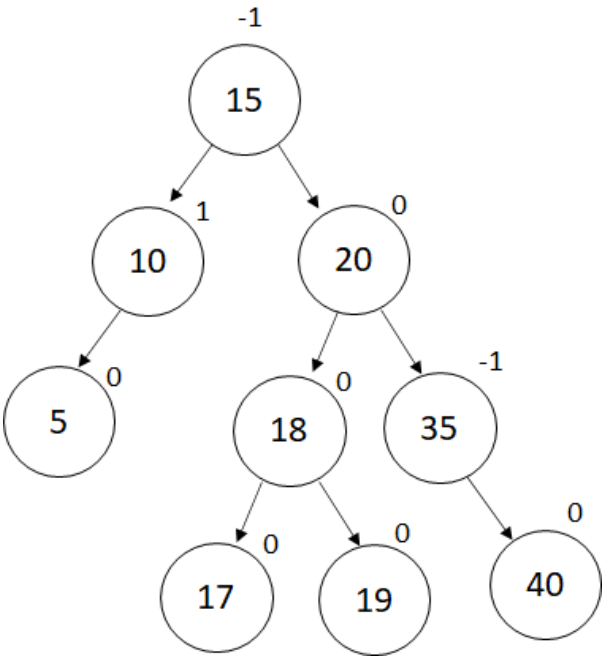


Inserindo 19:



Balanceamento igual a -2 e a 2, temos que fazer uma rotação.

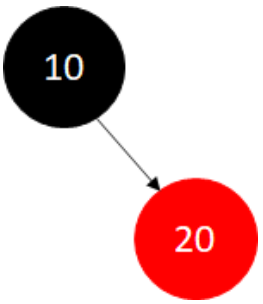




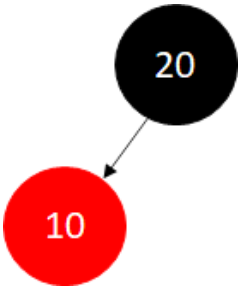
4-)
4.1-) 10,20,15,35,40

Inserindo 10:

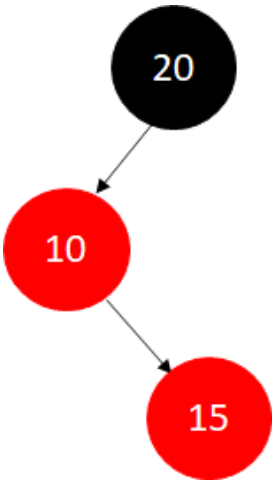
Inserindo 20:



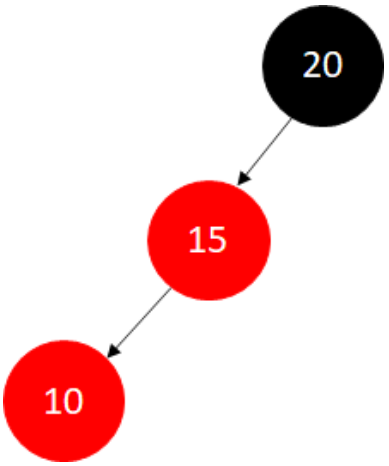
Ajustando:

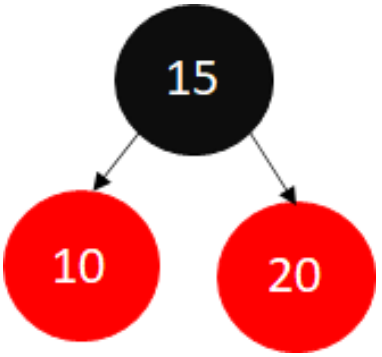


Inserindo 15:

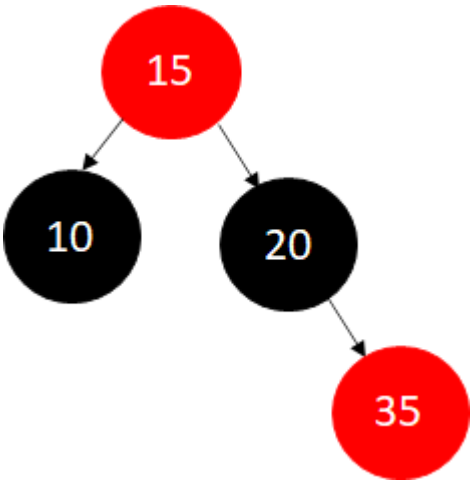


Ajustando:

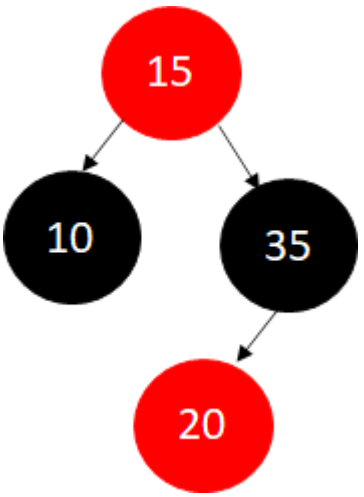




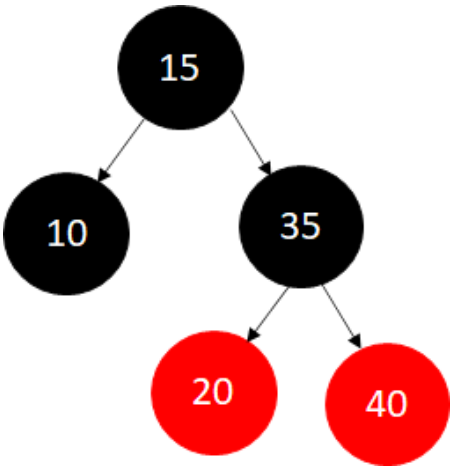
Inserindo 35:



Ajustando

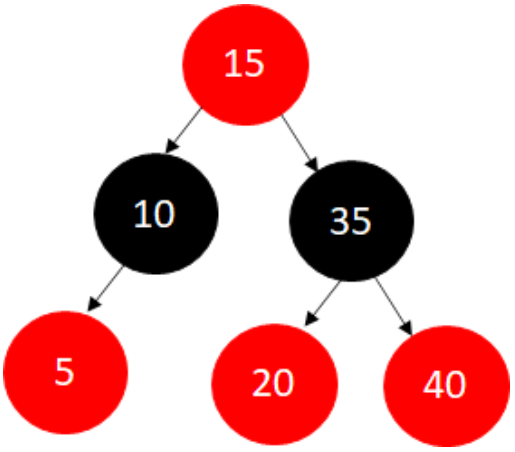


Inserindo 40:

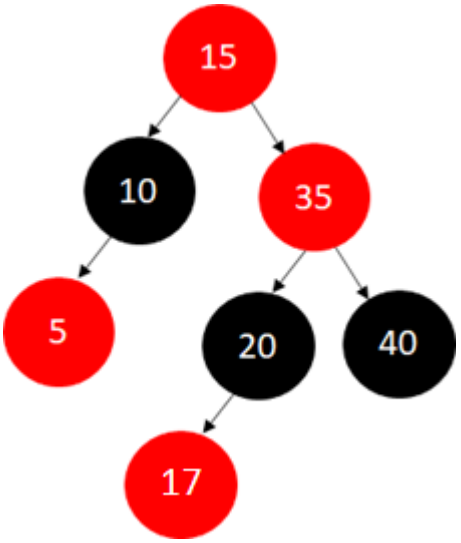


4.2-)Inserir os números 5,17,18,19

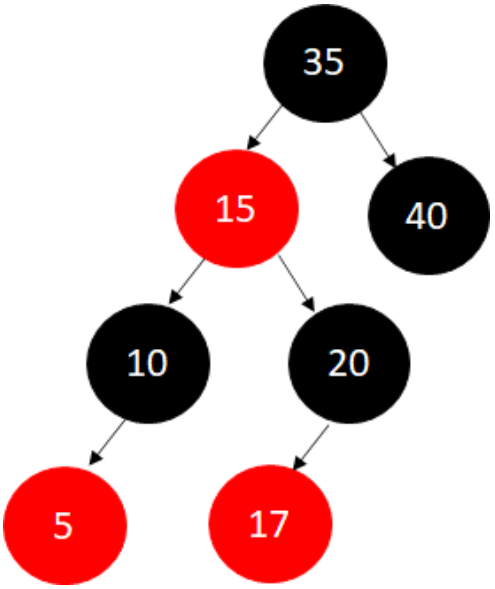
Inserindo o 5:



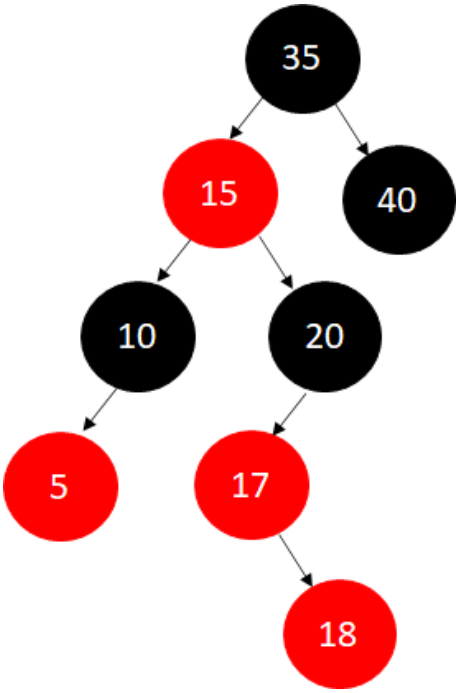
Inserindo o 17:



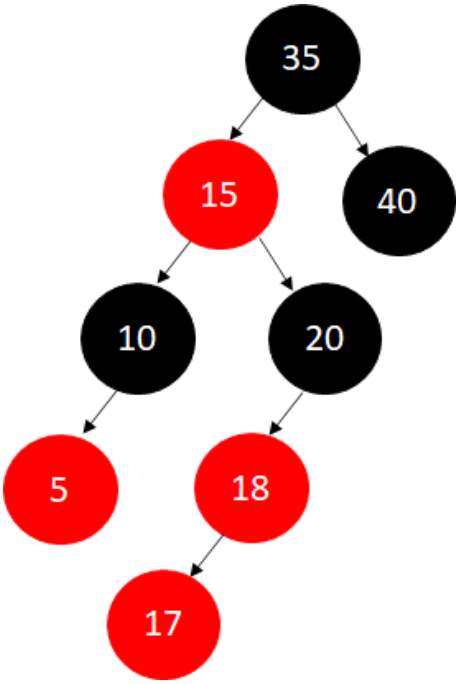
Ajustando:

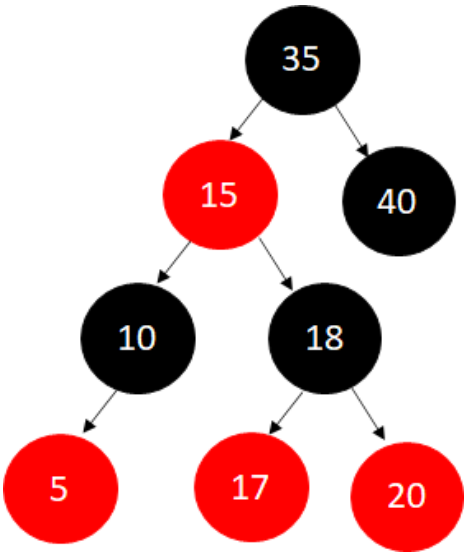


Inserindo o 18:

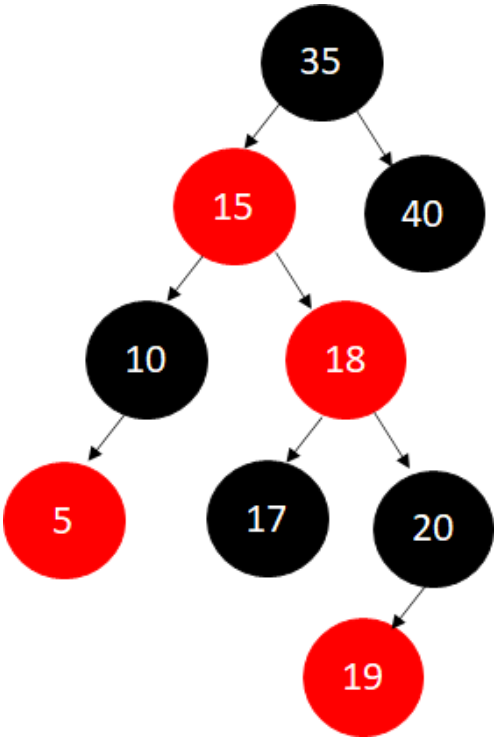


Ajustando:

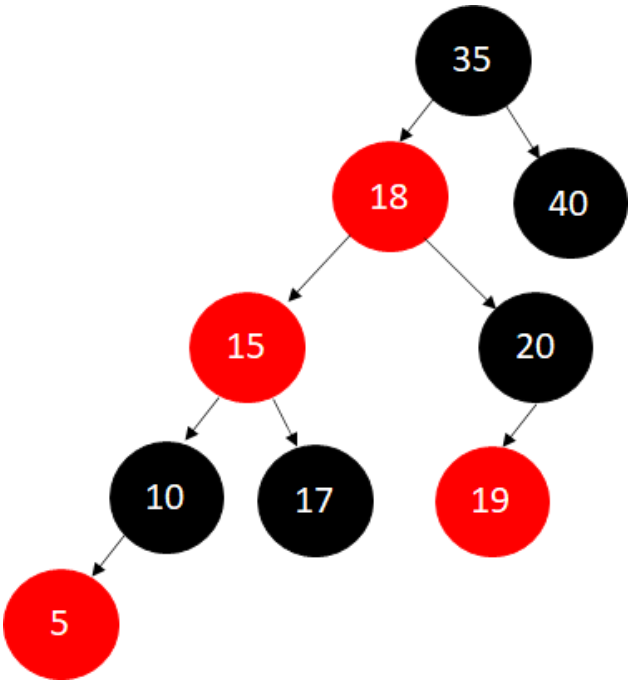


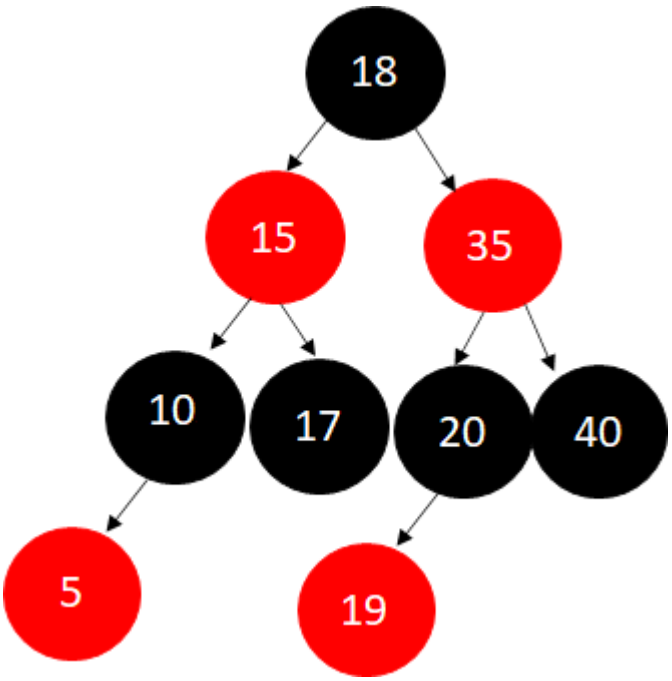


Inserindo o 19:



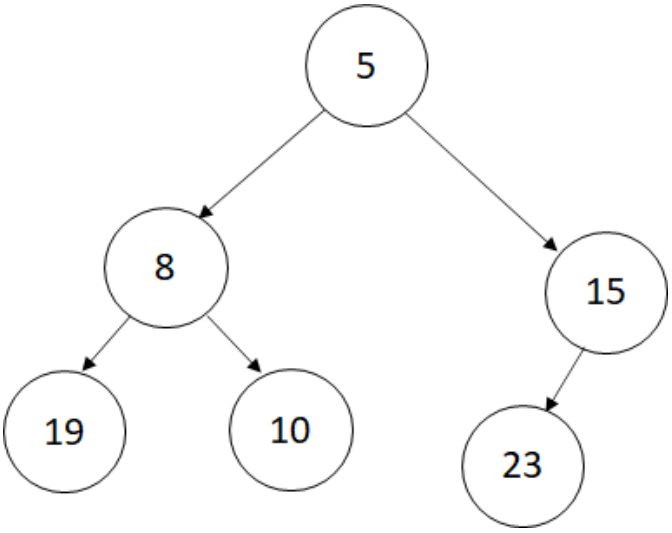
Ajustando:





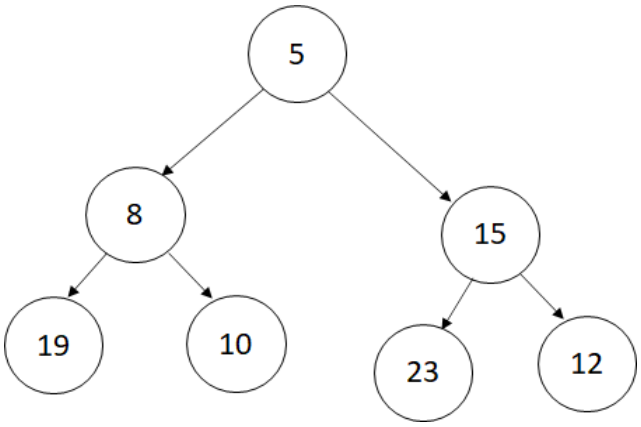
5-)

5.1-) 5,8,15,19,10,23

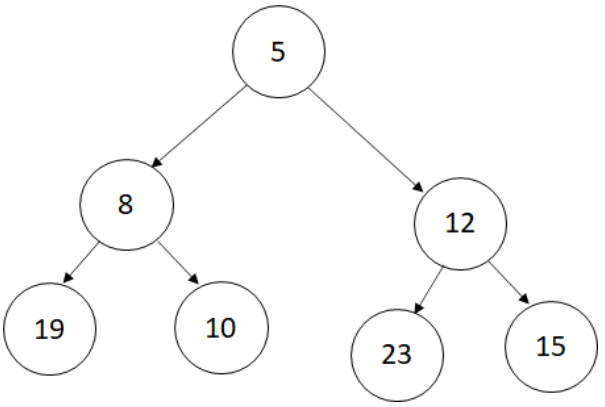


5.2-)Inserir 12,7,9,3

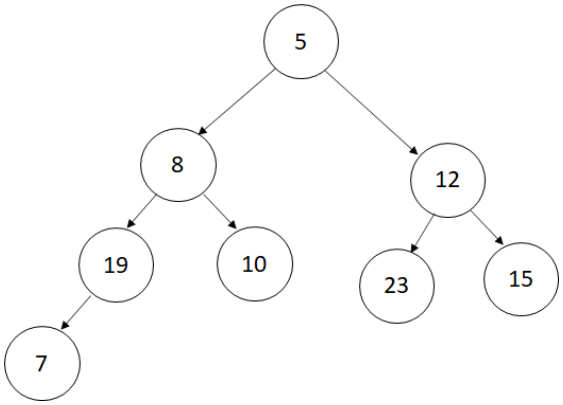
Inserindo 12:



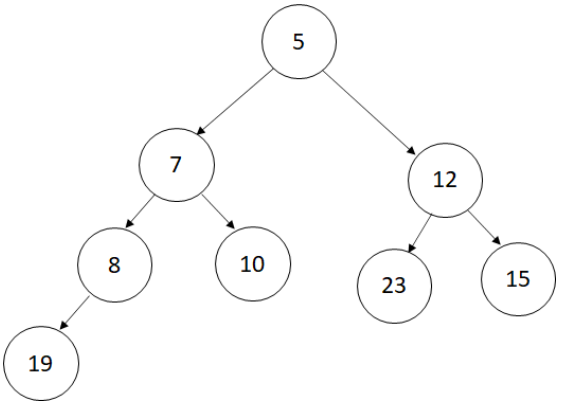
Ajustando:



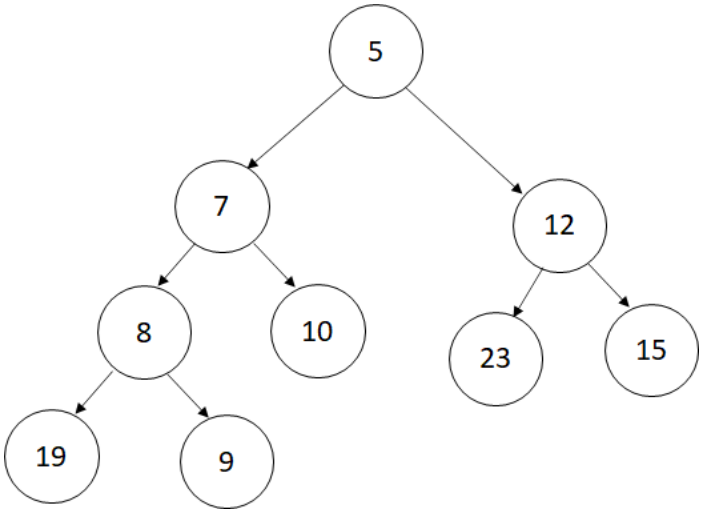
Inserindo 7:



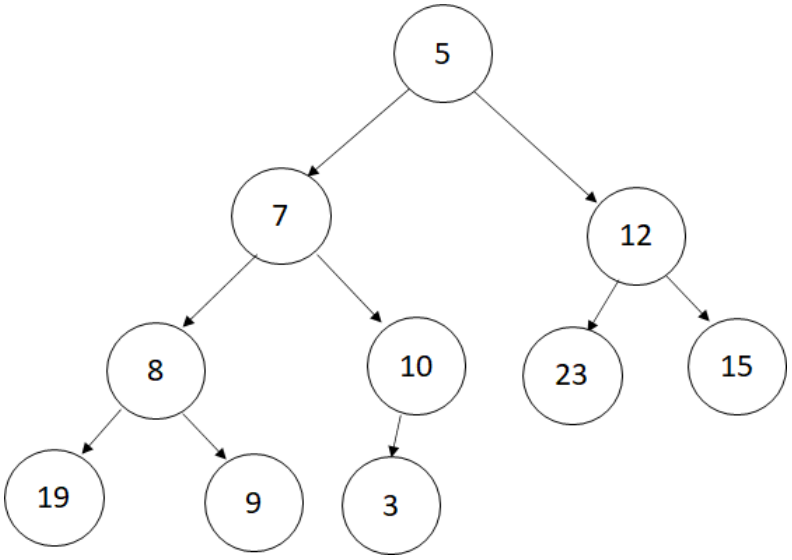
Ajustando:



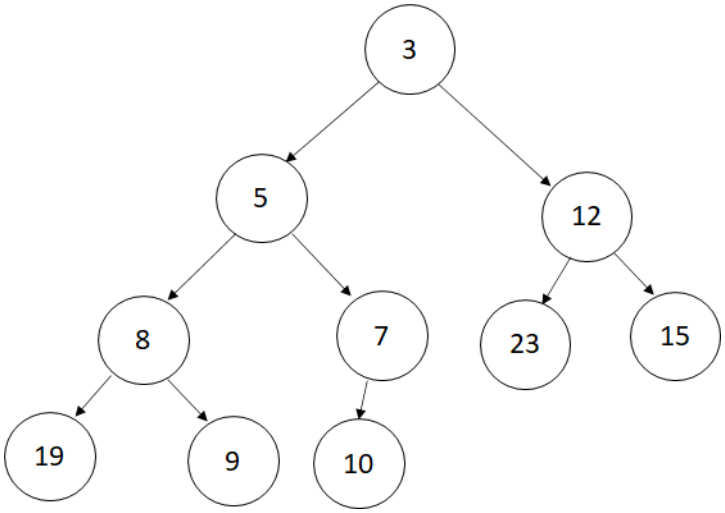
Inserindo o 9:



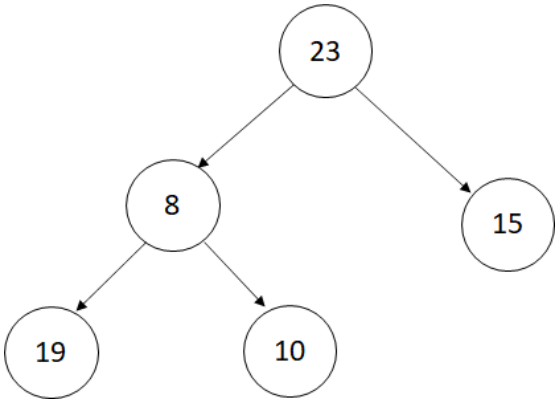
Inserindo o 3:

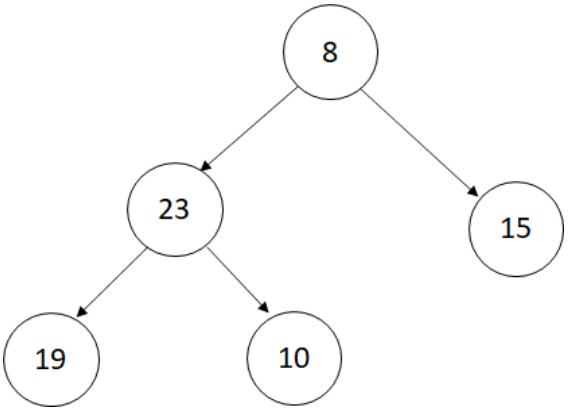


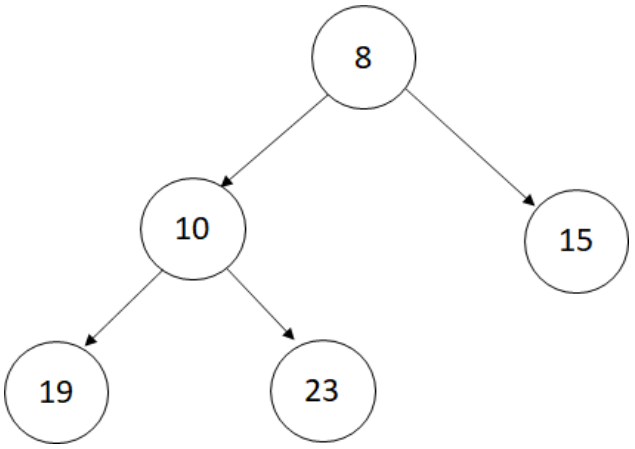
Ajustando:



5.3- Remova a chave mínima:
O número 5 foi removido







5	8	15	19	10	23			

8	15	19	10	23				