

Respostas da prova de Estrutura de Dados - 2021.01

- Aluno: Diego Vasconcelos Schardosim de Matos
- Dre: 120098723

Questão 1:

Algoritmo feito em C++, adicionei a Classe Node para melhor entendimento.

```
class Node {
public:
    int data = 0;
    Node* next = nullptr;

    Node(int data): data(data) {}
    ~Node() {
        delete next;
    }
};

void insert(int data, Node** currentNode) {
    if(*currentNode == nullptr) {
        *currentNode = new Node(data);
        return;
    }

    if(data <= (*currentNode)->data) {
        Node* tempNode = *currentNode;
        Node* newNode = new Node(data);

        *currentNode = newNode;
        newNode->next = tempNode;
    } else {
        insert(data, &(*currentNode)->next);
    }
}
```

Questão 2:

2.1) Algoritmo feito em C++. Como o próprio Nó enxerga as subárvores *esquerda* e *direita*, adicionei a lógica para calcular a soma das chaves como método do Nó.

```
class Node {
public:
    Node* esquerda = nullptr;
    Node* direita = nullptr;
    int chave = 0;
```

```

    int soma = 0;

    Node(int chave): chave(chave), soma(chave) {}
    ~Node() {
        delete esquerda;
        delete direita;
    }

    void calculateSoma() {
        if(esquerda != nullptr && direita == nullptr)
            soma = esquerda->soma + chave;
        else if(esquerda == nullptr && direita != nullptr)
            soma = direita->soma + chave;
        else
            soma = esquerda->soma + direita->soma + chave;
    }
};

```

2.2) Algoritmo feito em C++. Dei preferencia à recursão, pois, após inserir o Nó, ao sair da recursão, é só chamar o função *calculateSoma* (A class Node é a mesma da questão anterior).

```

void Insert(int data, Node** currentNode) {
    if(*currentNode == nullptr) {
        *currentNode = new Node(data);
        return;
    }

    if(data <= (*currentNode)->chave) {
        Insert(data, &(*currentNode)->esquerda);
    } else {
        Insert(data, &(*currentNode)->direita);
    }

    (*currentNode)->calculateSoma();
}

```

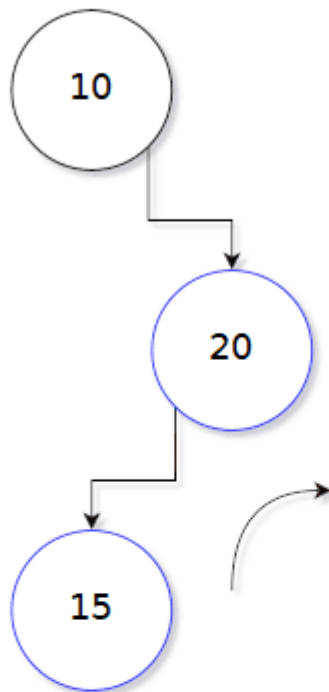
Questão 3:

PS: Por algum motivo o meu editor de imagens *comeu* algumas palavras. No lugar das reticências estaria o "Antes" e o "Depois". O diagrama completo pode ser visto [aqui](#).

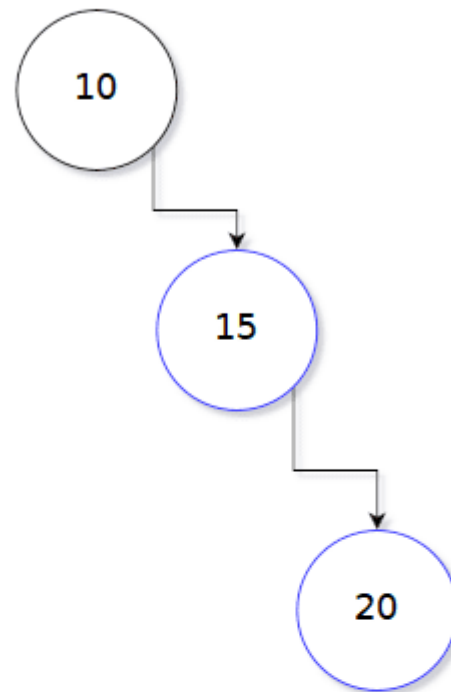
3.1) Foram inseridos, em sequência, os números: 10, 20 e 15. Ao inserir o 15, a árvore ficou desbalanceada, sendo necessário fazer uma **rotação para a direita** e depois **para a esquerda**.

Inserção de 10, 20, 15...

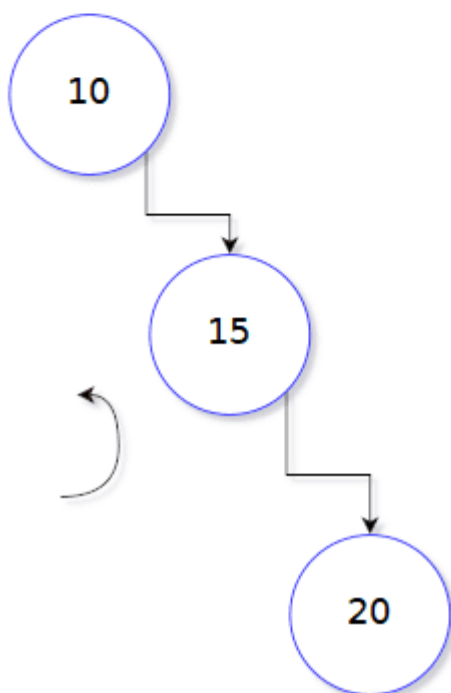
Rotação Para a Direita...



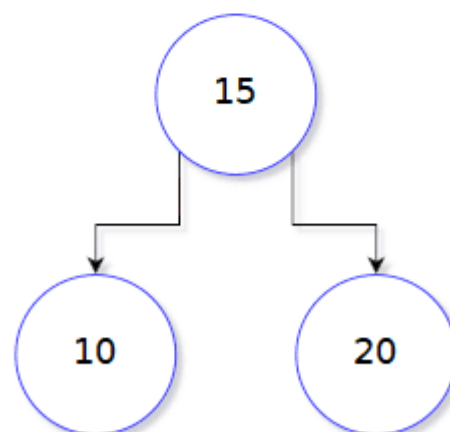
Rotação Para a Direita...



Rotação Para Esquerda...



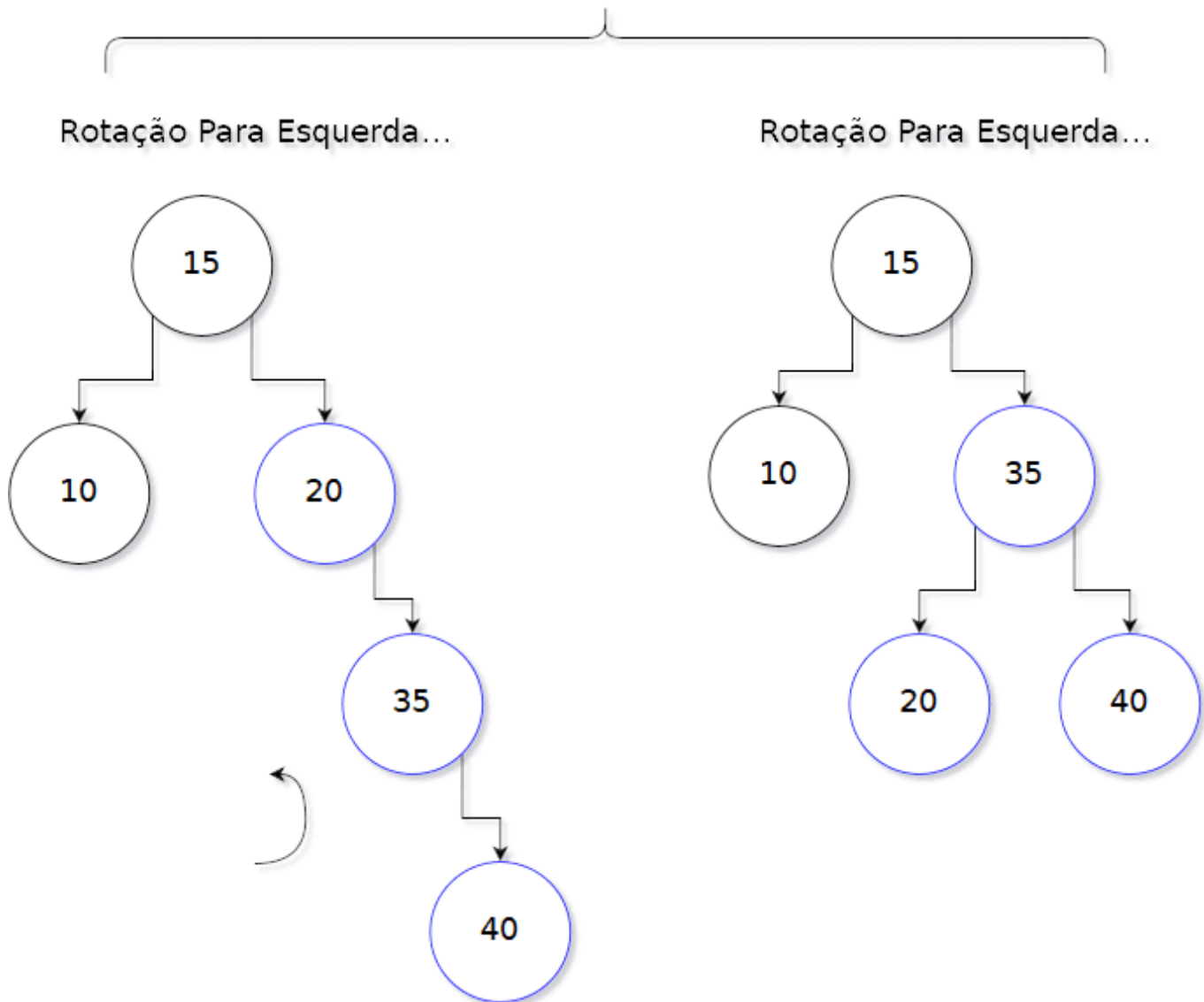
Rotação Para Esquerda...



Os Nós marcados em azul é quem está fazendo parte da rotação

Em seguida, inseri o 35 e 40, novamente, a árvore ficou desbalanceada, sendo necessário **rotacionar para esquerda**.

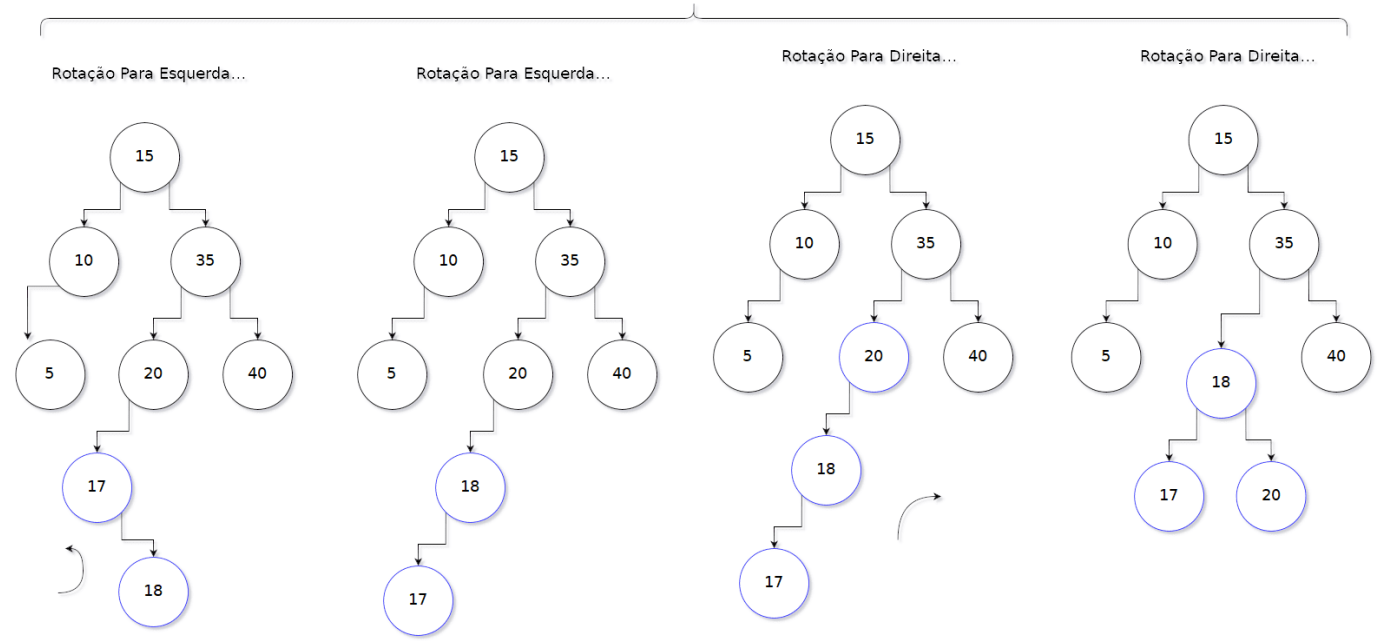
Inserção de 35, 40...



Os Nós marcados em azul é quem está fazendo parte da rotação

3.2) Depois de inserir, na árvore anterior, os números 5, 17 e 18, novamente a árvore ficou desbalanceada. Sendo necessário uma **rotação para a esquerda** e depois **para a direita**.

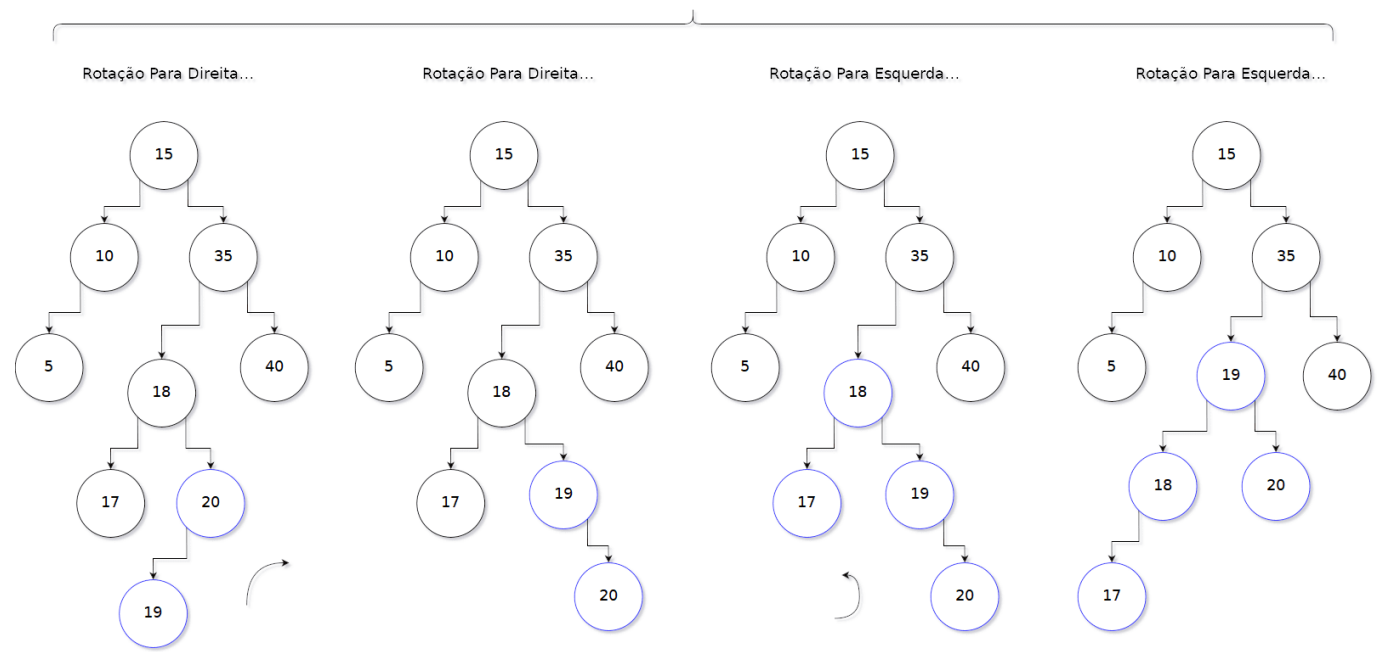
Inserção de 5, 17, e 1...

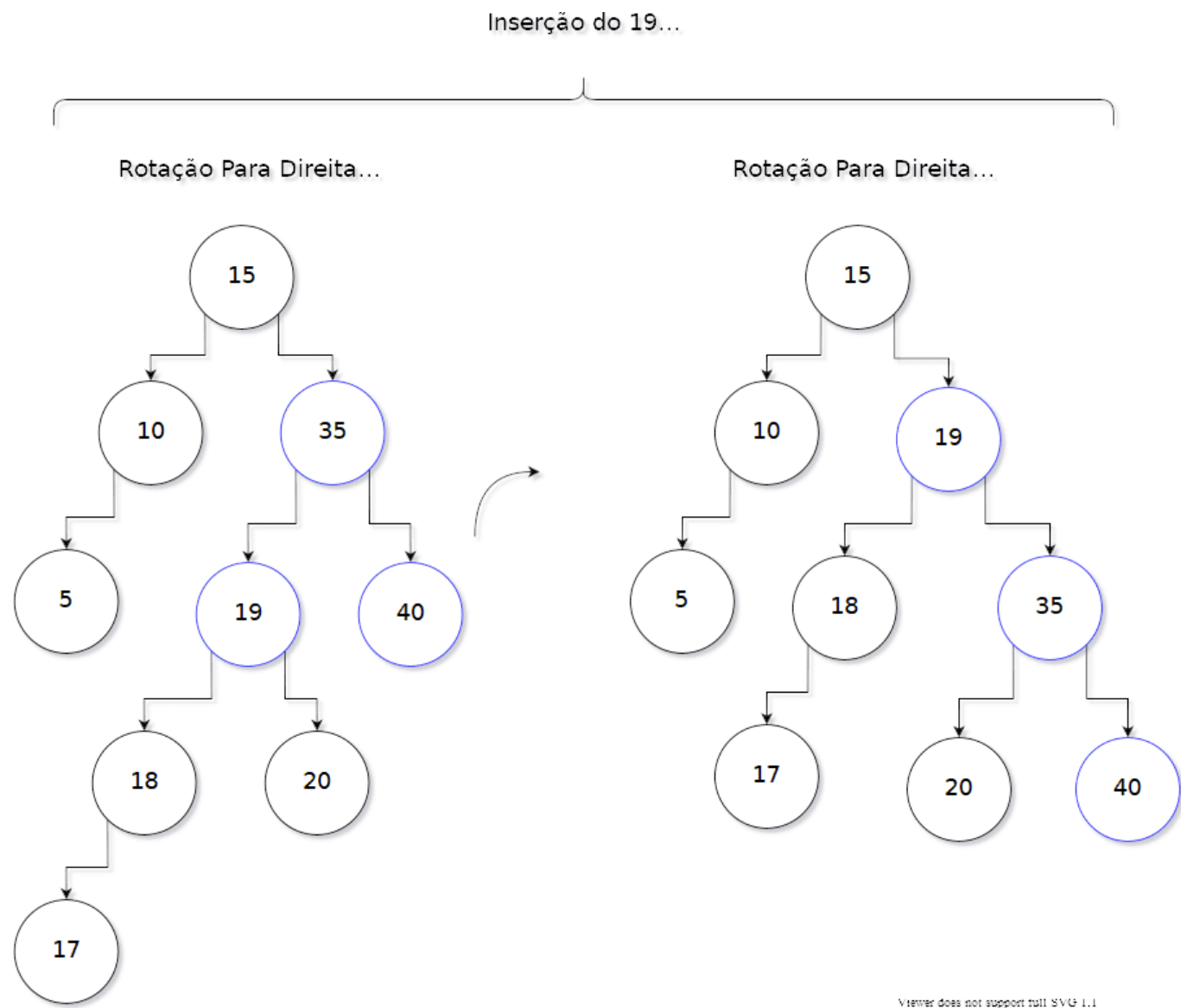


Os Nós marcados em azul é quem está fazendo parte da rotação

Em seguida, inseri o último número, 19. Esse foram necessárias 3 rotações: Para a Direita, Para a esquerda, e depois para a Direita

Inserção do 19...



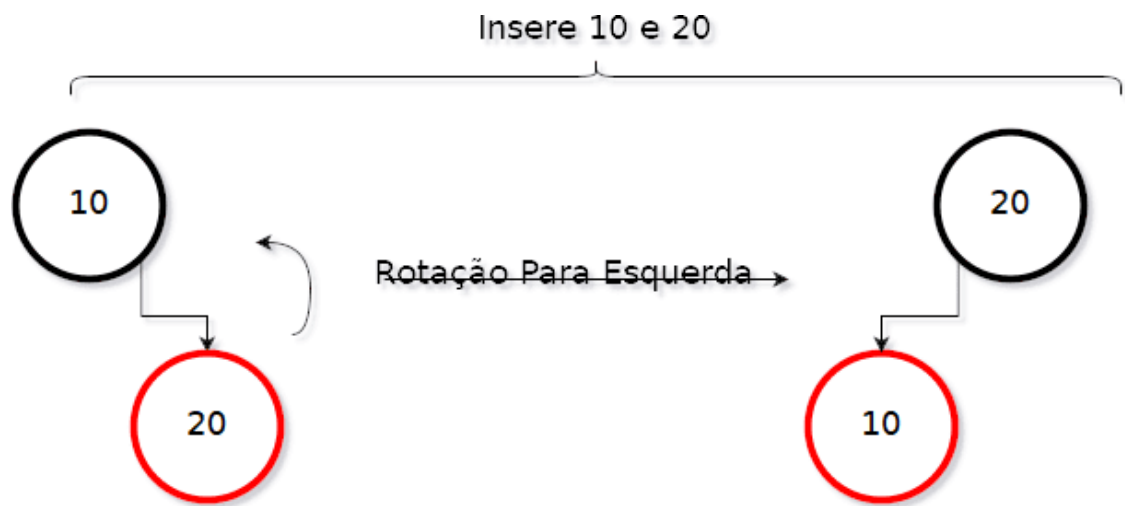


Os Nós marcados em azul é quem está fazendo parte da rotação

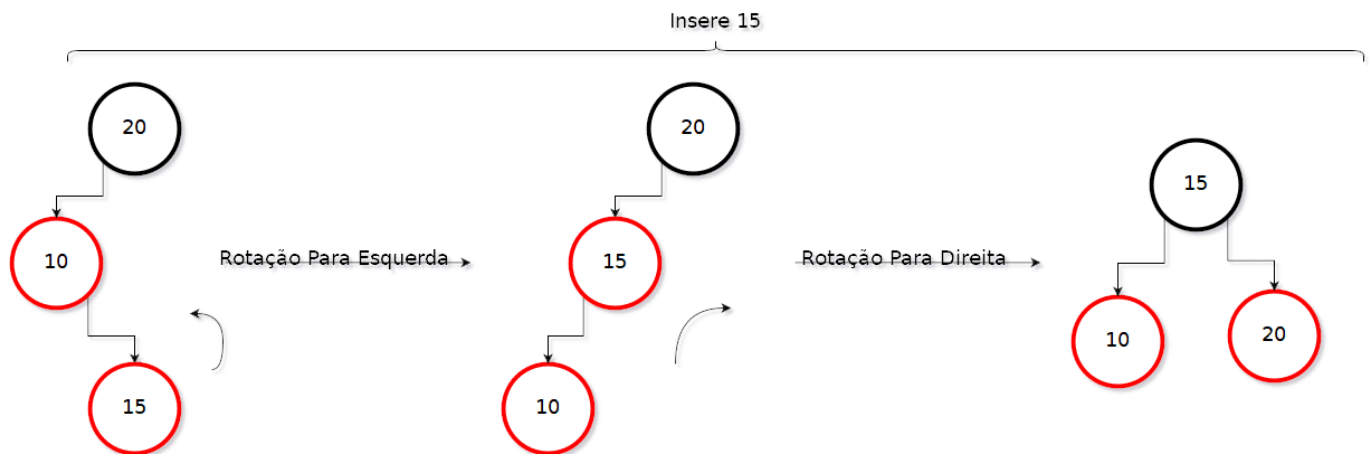
Questão 4:

PS: Novamente o meu editor de imagens causou problemas... Apesar de parecer que o texto esta hachurado/tachado, ele na realidade está em cima das setas. O Diagrama completo pode ser visto [aqui](#).

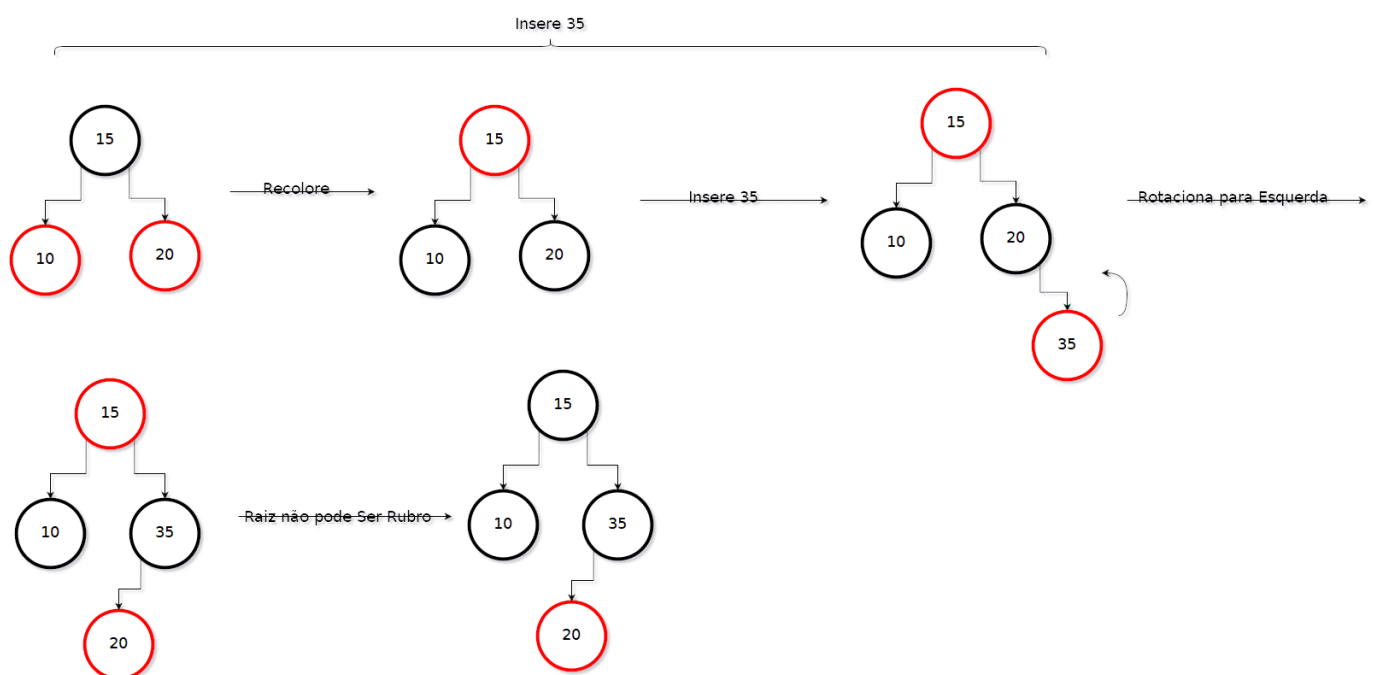
4.1) Depois de inserir o 20, houve a necessidade de rotacionar para a **esquerda**:



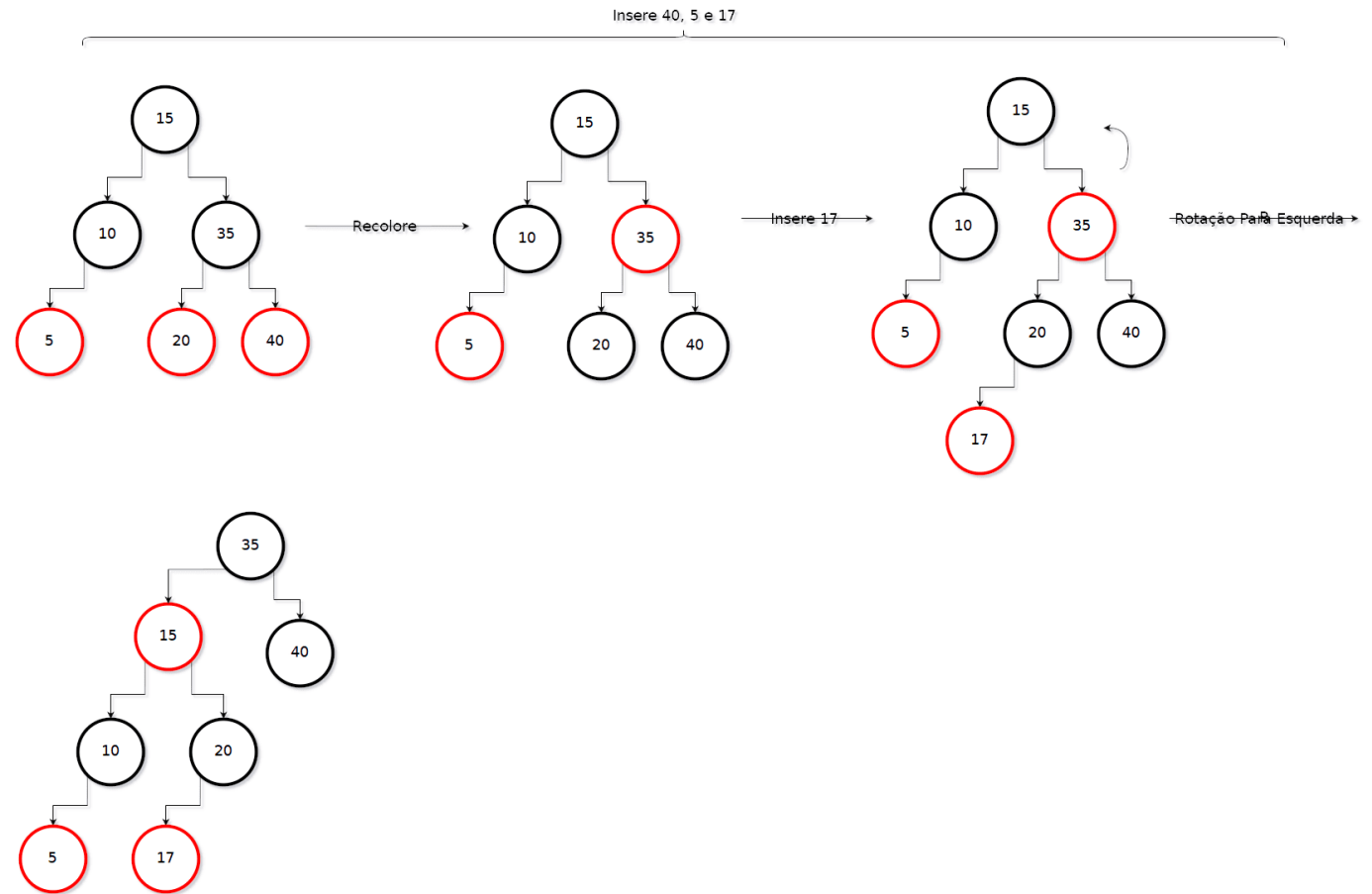
Ao inserir o 15:



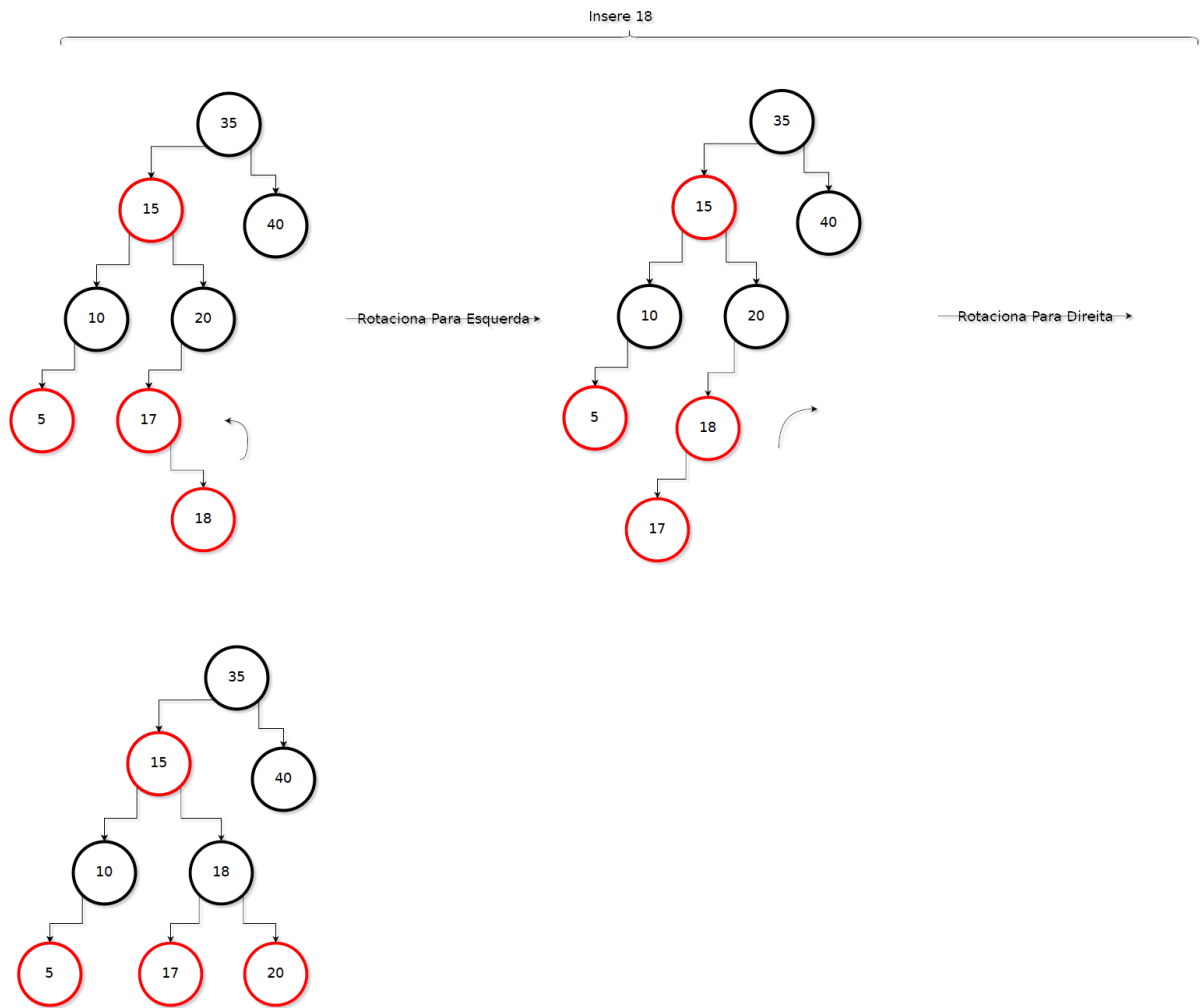
Então o 35:



4.2) Depois de inserir o 40 e o 5 na árvore anterior, houve outra rotação ao inserir o 17:



Mais uma ao inserir o 18:

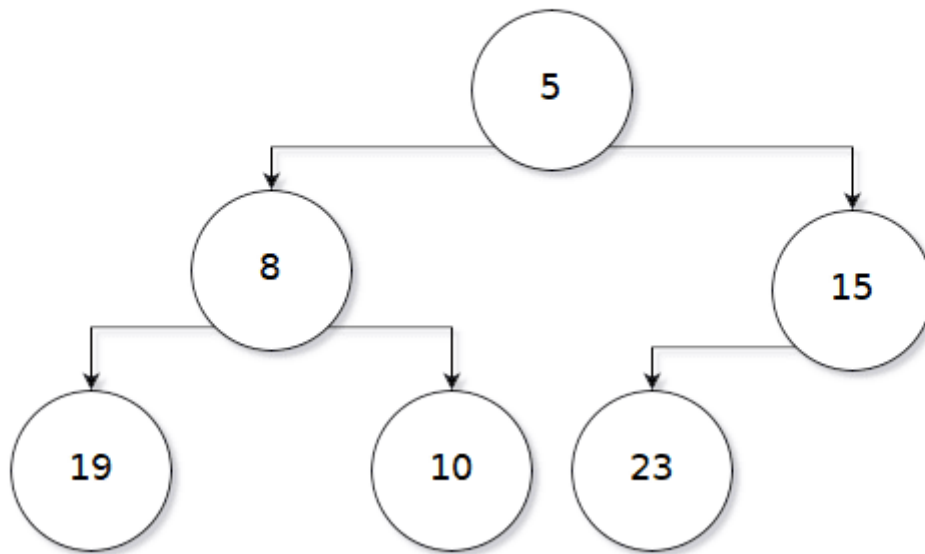


The diagram illustrates the insertion of node 19 into a B+ tree and the necessary rotations to maintain the tree's balance. The process is shown in five stages:

- Initial Tree:** A B+ tree with root 35. The left child is 15, and the right child is 40. Node 15 has children 10 and 18. Node 10 has children 5 and 17. Node 18 has child 20. Nodes 5, 17, and 20 are red.
- Recolorir:** The tree is the same as the initial tree, but nodes 15 and 18 are also red.
- Inserir 19:** Node 19 is added as the right child of node 20. Node 19 is red.
- Rotaciona Para Esquerda:** A left rotation is performed around node 18. Node 18 becomes the parent of nodes 15 and 20. Node 15 has children 10 and 17. Node 20 has children 19 and 5. Nodes 15, 18, 19, and 5 are red.
- Rotação Para Direita:** A right rotation is performed around node 18. Node 18 becomes the root. Node 15 is the left child of 18, and node 35 is the right child of 18. Node 15 has children 10 and 17. Node 35 has children 20 and 40. Node 10 has child 5, and node 20 has child 19. Nodes 15, 19, and 35 are red.

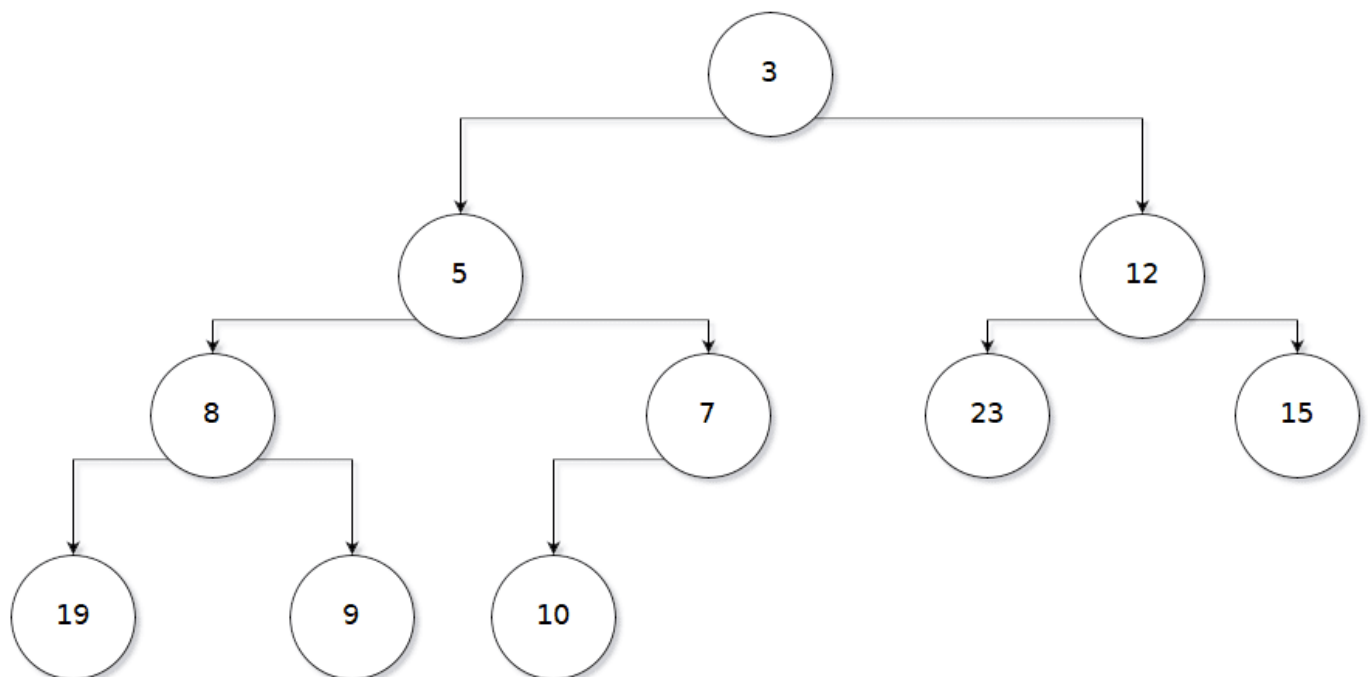
5.1) Representação em árvore da heap binária:

5.1- Desenho em árvore da Heap Binária



5.2) **Heap Resultante como árvore** após Inserção das chaves 12, 7, 9, 3 na Heap anterior:

5.2- Inserção das chaves 12,7,9 e...



5.3) Remoção da **Chave Mínima** da Heap Original (5.1)

5.3- Remoção da chave mínima

