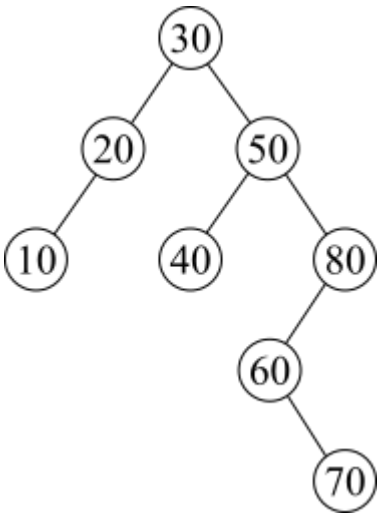

Enunciado. Implemente 6 (seis) novos métodos de Árvore Binária de Busca (não balanceada) descritos a seguir. Acrescente qualquer método auxiliar de nó ou de árvore que considerar necessário.

(1) determinação da altura da árvore:

```
int height();
```

Exemplo:

para o desenho acima, a saída deve ser:

```
4
```

(2) contagem do número de folhas:

```
int leaves();
```

Exemplo:

para o desenho acima, as folhas são 10, 40 e 70 e a saída deve ser:

```
3
```

(3) criação de uma lista com o menor (mínimo) e o maior (máximo) valor da árvore:

```
ArrayList<T> limits();
```

Exemplo:

para o desenho acima, a lista de saída deve ser:

```
[10, 80]
```

(4) criação de uma duplicação, em memória, da árvore:

```
BinaryTree<T> clone();
```

Dica: faça a inserção, a partir da sequência do percurso em 'pré ordem'.

(5) remoção de nós pelo número de seus filhos:

```
void filter(int n_child);
```

Exemplo:

para o desenho acima e `n_child = 1`, os nós a serem removidos são 20, 60 e 80.

Dica: faça uma lista com percurso em 'pós ordem' dos nós com `n_child` filhos, removendo-os nesta sequência.

(6) criação de um nova árvore que tenha todos os valores e a menor altura possível, ou seja, balanceada com base apenas no estabelecimento de uma nova ordem de inserção:

```
BinaryTree<T> balance();
```

Dica: dado o percurso 'em ordem':

- inserir a mediana: valor em [índice_central = (índice_inicial + índice_final) **div** 2]
- executar recursivamente para os valores entre o [índice_inicial] e [índice_central - 1]
- executar recursivamente para os valores entre o [índice_central + 1] e [índice_final]