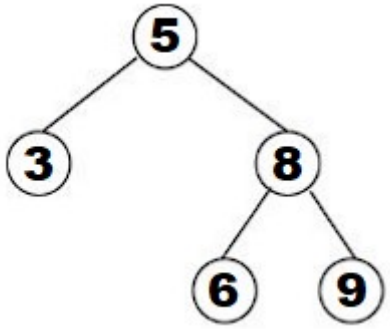
UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
INSTITUTO METRÓPOLE DIGITAL
ESTRUTURAS DE DADOS BÁSICAS II

TRABALHO – UNIDADE 2

Frequentemente, uma estrutura de dados suporta quase todas as operações que precisamos. Quando não é caso, podemos aumentar a estrutura pela adição de informações e/ou operações. Neste projeto, vocês devem implementar uma árvore binária de busca (ABB) aumentada que guarde valores **int** e que suporte, além das operações convencionais de busca, inserção e remoção, as operações elencadas a seguir:

1. **int enesimoElemento (int n)**: retorna o n-ésimo elemento (contando a partir de 1) do percurso em ordem (ordem simétrica) da ABB.
2. **int posicao (int x):** retorna a posição ocupada pelo elemento x em um percurso em ordem simétrica na ABB (contando a partir de 1).
3. **int mediana ()**: retorna o elemento que contém a mediana da ABB. Se a ABB possuir um número par de elementos, retorne o menor dentre os dois elementos medianos.
4. **double média (int x)**: retorna a média aritmética dos nós da árvore que x é a raiz.
5. **boolean ehCheia ()**: retorna verdadeiro se a ABB for uma árvore binária cheia e falso, caso contrário.
6. **boolean ehCompleta ()**: retorna verdadeiro se a ABB for uma árvore binária completa.
7. **String pre_ordem ()**: retorna uma String que contém a sequência de visitação (percorrimento) da ABB em pré-ordem.
8. **void imprimeArvore (int s)**: se "s" igual a 1, o método imprime a árvore no formato 1, "s" igual a 2, imprime no formato 2.

Considere como exemplo a árvore a seguir e sua impressão nos dois formatos.



Formato 1:

```
5 ------------------------------
        3----------------------
        8----------------------
                6-------------
                9-------------
```

Formato 2:

(5 (3) (8 (6) (9)))

Algumas operações descritas poderia ser facilmente implementada utilizando um percurso em ordem simétrica e, talvez, armazenando resultados em um vetor. Entretanto, esse procedimento **é ineficiente**. Ao invés disso, vamos **melhorar o desempenho dessas operações** aumentando os nós da ABB, isto é, armazenando informações extras em cada nó da árvore que simplificarão as operações. **Uma informação pode ser a quantidade de nós nas subárvores à direita e à esquerda**. **Outras informações necessárias, se for o caso, devem ser identificadas por vocês.** Seu algoritmo deve receber dois arquivos como parâmetros. O primeiro, contém uma descrição da ABB que será utilizada e é denominado arquivo de entrada da ABB. O arquivo de entrada da ABB contém uma sequência de valores inteiros separados por um espaço, os valores a serem armazenados na árvore. O segundo arquivo, denominado arquivo de comandos, contém uma sequência de operações (uma operação por linha) a serem realizadas pelo seu algoritmo. O arquivo de comando poderá utilizar as operações a seguir:

*Formato:
ENESIMO N
POSICAO N
MEDIANA
CHEIA
COMPLETA
IMPRIMA S
REMOVA N

*Exemplo:
IMPRIMA 2
MEDIANA
ENESIMO 10

Observações importantes:

1. Valores duplicados não serão permitidos na árvore. Tentativas de inserção de um valor duplicado devem ser identificadas e devidamente ignoradas.
2. Tentativas de remoção de um elemento que não existe na árvore também devem ser previstas e devidamente ignoradas.
3. A maneira mais prática de atender aos itens 1 e 2 é realizar uma chamada prévia à função BUSCAR. Entretanto, isso **não será permitido**. Vocês devem implementar a inserção e a remoção de forma independente do algoritmo de busca, para que o seu algoritmo seja o mais eficiente possível.
4. A propósito, **não será permitido o uso de estruturas de dados prontas de uma biblioteca** qualquer. Vocês terão que implementar a ABB. Quase todas as operações descritas são implementadas mais facilmente de maneira recursiva. Entretanto, vocês são livres para adotar a abordagem recursiva ou não.
5. Não serão aceitos trabalhos que não compilam ou não executam.
6. A linguagem de programação é livre (preferencialmente Java, C, C++). Vocês devem submeter um README com instruções para compilação e execução do seu programa, incluindo instalação de software necessário para execução em máquina com Ubuntu.
7. A submissão dos trabalhos deve conter:
    - código fonte
    - README
    - Breve relatório contendo os nomes dos integrantes do grupo, descrevendo sucintamente sua abordagem de solução e **contendo análise de complexidade assintótica** dos métodos implementados.

8. A implementação valerá 3,0 pontos para a segunda unidade.
9. Trabalhos nos quais forem identificados plágio, não receberão pontuação e a nota da prova da segunda unidade será, no máximo, 7,0.
10. O trabalho será realizado em grupos de, no máximo, 3 pessoas.