Analisando e Entendendo a Remoção em Árvores B

---

1. Objetivos

- Compreender a operação de **remoção** em Árvore B.
- Exercitar:
    - casos de empréstimo à esquerda e à direita,
    - casos de fusão,
    - remoção de predecessor e sucessor.
- Fazer **pequenas modificações** no código existente.
- Experimentar remoções e observar o impacto na estrutura.

---

2. Atividade Proposta

Parte A – Entendimento prático da remoção

Utilizando o código para BTree: BTree.java.

O estudante deverá:

1. Executar o programa.
2. Inserir automaticamente todos os valores de 1 a N (ex.: N=63).
3. Realizar uma sequência de remoções e observar visualmente (via `print()`) como a árvore se reorganiza.

A sequência sugerida de remoções é:

1. Remover uma chave em nó folha que **não exige ajustes**
2. Remover uma chave em nó folha que exige **empréstimo**
3. Remover uma chave em nó folha que exige **fusão (merge)**

4. Remover uma chave em nó interno, explorando:
    - Remoção via **predecessor**
    - Remoção via **sucessor**

Seria interessante **registrar imagens ou cópias da saída** antes e depois de cada operação.

Parte B – Instrumentar o código com logs explicativos

Deverão ser inseridas mensagens de debug que indiquem:

- Quando ocorre um **empréstimo da esquerda**
- Quando ocorre um **empréstimo da direita**
- Quando ocorre uma **fusão**
- Quando a remoção usa **predecessor**
- Quando a remoção usa **sucessor**

Exemplo:

```
System.out.println("-> borrowFromPrev(): emprestando chave do irmão esquerdo.");
System.out.println("-> merge(): realizando fusão dos nós " + idx + " e " + (idx+1));
```

Isso permite uma **visualização** do algoritmo sendo executado.

3. Entregáveis

✔ Código modificado com logs

✔ Pequeno relatório respondendo:

- Que casos de remoção foram observados?
- O algoritmo funcionou conforme esperado?
- Houve merge? Houve empréstimos?
- O que foi mais fácil/difícil?
- Em que situações a remoção simplifica o nó e quando ela complica?