Relatório - Exercício prático com Árvore B (remoção)

Casos de remoção observados

Ao executar a árvore inserindo os valores de 1 a N e removendo diversas chaves, observei:

1. Remoção em nó folha sem necessidade de ajustes
   ocorreu quando o nó ainda possuía mais de *t–1* chaves após a remoção.

2. Remoção em nó folha com empréstimo
   ocorreu quando um dos irmãos tinha ≥ *t* chaves.
   Logs exibiram chamadas para `borrowFromPrev()` e `borrowFromNext()`.

3. Remoção exigindo fusão (merge)
   ocorreu quando ambos os irmãos tinham *t–1* chaves.
   Logs mostraram `merge()` sendo acionado.

4. Remoção em nó interno usando predecessor
   acionada quando o filho esquerdo tinha ≥ *t* chaves.
   Confirmado pelo log: “usando PREDECESSOR”.

5. Remoção em nó interno usando sucessor
   acionada quando o filho direito tinha ≥ *t* chaves.
   Confirmado pelo log: “usando SUCESSOR”.

O algoritmo funcionou como esperado?

Sim, todos os casos clássicos da remoção em Árvore B ocorreram conforme a literatura.
Os logs ajudaram a verificar cada passo do processo e confirmaram a consistência do algoritmo.

Houve merge? Houve empréstimos?

Sim, houve ambos.

- Empréstimos ocorreram especialmente em remoções próximas das folhas.

- Fusões ocorreram quando nenhum irmão tinha chaves suficientes.

O que foi mais fácil/difícil?

Mais fácil:

- Remover chaves em nós folha.
- Observar visualmente a mudança de estrutura após cada remoção.

Mais difícil:

- Acompanhar mentalmente as fusões e redistribuições sem os logs.
- Entender por que certos casos preferem predecessor ou sucessor.

Em que situações a remoção simplifica o nó e quando ela complica?

Simplifica quando:

- O nó tem chaves sobrando (≥ t).
- A remoção ocorre em folha sem quebrar invariantes.

Complica quando:

- O nó possui apenas t–1 chaves.
- Precisamos recorrer a:
    - empréstimos,
    - fusões,
    - substituição por predecessor/sucessor.

Nesses casos, o algoritmo precisa garantir as propriedades da Árvore B a cada descida.