Cauã Borges Faria (834437)

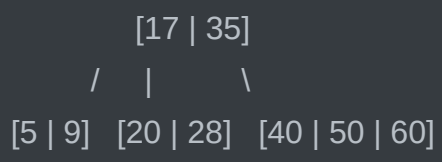
Questão 1

Uma **Árvore-B** é uma estrutura de dados em árvore **balanceada e de busca** especialmente projetada para funcionar bem em sistemas de memória secundária, como discos. Diferente das árvores binárias, cada nó da Árvore-B pode armazenar várias chaves e ter vários filhos, o que reduz a altura da árvore e, consequentemente, o número de acessos a disco necessários para operações de busca, inserção e remoção.

Ela é caracterizada por:

- Ser sempre balanceada.
- Permitir vários filhos por nó (definidos pela ordem `m`).
- Manter todas as folhas no mesmo nível.
- Operações de busca e atualização com complexidade **O(log$_m$ n)**.

Diagrama



Explicação:

- Cada nó pode armazenar de 1 a 3 chaves (ordem 4 → até 3 chaves e até 4 filhos).
- As chaves dentro de cada nó são armazenadas em ordem crescente.
- Os ponteiros (filhos) dividem os intervalos de valores:
  - Filhos à esquerda de 17 contêm valores menores que 17.
  - Filhos entre 17 e 35 contêm valores entre 17 e 35.
  - Filhos à direita de 35 contêm valores maiores que 35.

- Todas as folhas estão no mesmo nível.

🔍 **Busca:**

- Começa na raiz, verifica em qual intervalo a chave está e segue pelo ponteiro correspondente.
- Continua até chegar no nó folha ou encontrar a chave.

➕ **Inserção:**

- Busca a posição correta.
- Insere no nó se houver espaço.
- Se o nó estiver cheio, divide-o e promove a chave do meio para o nó pai.
- Se a raiz for dividida, uma nova raiz é criada.

➖ **Remoção:**

- Localiza a chave.
- Remove e ajusta os nós, emprestando ou fundindo chaves se necessário para manter a propriedade mínima.

Questão 2

A diferença principal entre construir um índice em memória secundária com uma árvore B e com uma ABBB (árvore binária de busca balanceada) está na eficiência de acesso ao disco. A árvore B é otimizada para sistemas de memória secundária, pois cada nó ocupa o tamanho de um bloco de disco e armazena várias chaves, reduzindo a quantidade de acessos a disco e diminuindo a altura da árvore.

Já a ABBB, sendo binária, possui altura maior e acessa um nó por vez, o que resulta em muitos acessos a disco — algo caro e ineficiente nesse ambiente. Por isso, árvores B são preferidas em índices de bancos de dados e sistemas de arquivos, enquanto ABBB são mais adequadas para buscas rápidas em memória primária.

Questão 3

ordem de uma árvore B (normalmente representada por m) é o número máximo de filhos que cada nó pode ter. Em outras palavras, define a capacidade dos nós da árvore.

Em uma árvore B de ordem m:

- Cada nó pode ter no máximo m filhos.
- Exceto a raiz, cada nó deve ter pelo menos ⌈m/2⌉ filhos.
- Cada nó (exceto a raiz) deve conter entre ⌈m/2⌉ - 1 e m - 1 chaves.
- A raiz pode ter de 2 até m filhos ou nenhum se for a única folha.

Questão 4

Um nó folha em uma árvore B é um nó que não possui filhos. Ele representa o nível mais baixo da árvore e armazena as chaves finais ou referências aos registros de dados.

Na estrutura da árvore B, todos os nós folha estão sempre no mesmo nível, garantindo que a árvore permaneça balanceada e que o caminho da raiz até qualquer folha tenha o mesmo comprimento, o que mantém a eficiência nas operações de busca, inserção e remoção.

Questão 5

A complexidade de uma árvore B em termos de número de acessos a disco no pior caso é $O(\log_m n)$, onde:

- n é o número de chaves armazenadas na árvore.
- m é a ordem da árvore (o número máximo de filhos por nó).

Questão 6

```
ALGORITMO BUSCA-ARVORE-B(Nó, Chave)
    i ← 1

    ENQUANTO i ≤ número de chaves em Nó E Chave > Nó.chave[i] FAÇA
        i ← i + 1
    FIM-ENQUANTO

    SE i ≤ número de chaves em Nó E Chave = Nó.chave[i] ENTÃO
        RETORNE (Nó, i)  // chave encontrada no Nó na posição i
    SENÃO SE Nó é folha ENTÃO
        RETORNE NULL     // chave não está na árvore
```

```
        SENÃO
            LEIA o filho filho[i] do Nó (do disco, se necessário)
            RETORNE BUSCA-ARVORE-B(filho[i], Chave)
        FIM-SE
    FIM-ALGORITMO
```

Questão 7

Número máximo de descendentes de uma página:
512

Número máximo de chaves armazenadas na página:
511 (sempre `m - 1`)

Número mínimo de descendentes de uma página (que não seja folha nem raiz):
⌈512 / 2⌉ = 256

Número de chaves em uma página com 200 descendentes:
Uma página com `k` descendentes possui **k - 1 chaves**
200 - 1 = 199 chaves

Profundidade/altura máxima da árvore (valor inteiro arredondado), com 1.000.000 de chaves:
Formula:
h ≤ log_base ((n + 1) / 2)
onde base = ceil(m / 2)

Substituindo:

```
    n = 1.000.000

    m = 512

    base = 256
```

h ≤ log_256 (500.000)
log_256 (500.000) = log_2 (500.000) / log_2 (256)
log_2 (500.000) ≈ 18,93
log_2 (256) = 8

18,93 / 8 = 2,366

Arredondando para cima:

→ **3**