Complexity Analysis

US08

Função readBooFile

Leitura de linhas do arquivo:

- Complexidade: O(L), onde L é o número de linhas no arquivo.

Mapeamento de Operações e Materiais:

- Cada linha pode criar ou acessar nós no mapa nodeMap usando computeIfAbsent, que tem complexidade O(1)

Suboperações e Materiais:

- A função parseSubOperationsAndMaterials analisa suboperações e materiais presentes na linha.
- Se houver k suboperações ou materiais, a análise é O(k).

Complexidade total:

- O(L·k)

Método da Classe Tree

Método FillMaterialBST

Este método percorre todos os nós da árvore e insere materiais na árvore binária de busca.

Percorrer a árvore:

- Complexidade: O(N), onde N é o número total de nós na árvore.

Inserção em MaterialBST:

- A inserção tem complexidade O(logM), onde M é o número de materiais.

Complexidade total:

- O(N) + O(M·logM)

  M- número de materiais

Método FillNameMap

Este método percorre todos os nós da árvore e armazena-os num mapa de nomes.

Componentes principais:

Percorrer a árvore:

- Cada nó é visitado uma vez: O(N).

Inserção no nameMap:

- O(1)

Complexidade total:

- O(N)

Complexidade Geral do Sistema

Construção da árvore :

- O(L·k)

Preenchimento de MaterialBST:

- O(N)+O(M·logM)

Preenchimento de nameMap:

- O(N)

Complexidade Total- O(L·k)+O(N)+O(M·logM)

US10

A função insert insere um material na BST com base na quantidade.

Percorre a árvore comparando a quantidade do material com a quantidade do nó atual:

- Se menor, vai para a esquerda.
- Se maior, vai para a direita.
- Se igual, adiciona o material à lista de materiais do nó atual.

A profundidade é O(logN).

Listagem em ordem crescente ou decrescente

A função getMaterialsInOrder usa dois métodos auxiliares:

- inOrder: Percorre a árvore em ordem crescente.
- reverseInOrder: Percorre a árvore em ordem decrescente.

O custo de visitar todos os nós da árvore é proporcional ao número total de nós N.

Complexidade de Listagem:

- O(N)

Complexidade Total

Inserção de N materiais na árvore

O(N·log(N))