1.Introdução

Uma Árvore de Busca Binária (ou BST) é uma estrutura de dados usada em ciência da computação para organizar e armazenar dados de forma ordenada. Cada nó em uma Árvore de Busca Binária tem no máximo dois filhos, um filho esquerdo e um filho direito, com o filho esquerdo contendo valores menores que o nó pai e o filho direito contendo valores maiores que o nó pai. Essa estrutura hierárquica permite operações eficientes de busca, inserção e exclusão nos dados armazenados na árvore

2.Propriedades de uma BST

- A subárvore esquerda de um nó contém apenas nós com chaves menores que a chave do nó.
- A subárvore direita de um nó contém apenas nós com chaves maiores que a chave do nó.
- A subárvore esquerda e direita também devem ser uma árvore de busca binária.
- Não deve haver nós duplicados

3.Aplicações de uma BST

Um BST suporta operações como busca, inserção, exclusão, máximo, mínimo, piso, teto, maior, menor, etc. em tempo O(h) onde h é a altura do BST. Para manter a altura menor, BSTs de auto balanceamento (como AVL e Red Black Trees) são usados na prática. Esses BSTs de auto balanceamento mantêm a altura como O(Log n). Portanto, todas as operações mencionadas acima se tornam O(Log n). Junto com elas, o BST também permite a travessia de dados em ordem classificada em tempo O(n).

Uma Árvore de Busca Binária Auto Balanceada é usada para manter um fluxo de dados ordenado. Por exemplo, suponha que estamos recebendo pedidos on-line e queremos manter os dados ativos (na RAM) em ordem ordenada de preços. Por exemplo, desejamos saber o número de itens comprados a um custo abaixo de um determinado custo a qualquer momento. Ou desejamos saber o número de itens comprados a um custo maior do que o custo determinado.

Um BST pode ser usado para classificar um grande conjunto de dados. Ao inserir os elementos do conjunto de dados em um BST e, em seguida, executar uma travessia em ordem, os elementos serão retornados em ordem classificada. Quando comparado a algoritmos de classificação normais, a vantagem aqui é que podemos inserir/excluir itens posteriormente em tempo O(Log n).

4.Vantagens de uma BST

- **Busca eficiente:** complexidade de tempo O(log n) para busca com um BST auto balanceado

- **Estrutura ordenada:** elementos são armazenados em ordem classificada, facilitando encontrar o próximo ou o anterior elemento
- **Inserção e exclusão dinâmicas:** elementos podem ser adicionados ou removidos eficientemente
- **Estrutura balanceada:** BSTs balanceados mantém uma altura logarítmica, garantindo operações eficientes
- **Fila de prioridade duplamente finalizada:** em BSTs, podemos manter tanto o máximo quanto o mínimo eficientemente

5.Desvantagens de uma BST

- Não auto balanceado: BSTs desbalanceados podem levar a um desempenho ruim
- Complexidade de tempo do pior caso: No pior caso, BSTs podem ter uma complexidade de tempo linear para pesquisa e inserção
- Sobrecarga de memória: BSTs exigem memória adicional para armazenar ponteiros para nós filhos
- Não adequado para grandes conjuntos de dados: BSTs podem se tornar ineficientes para conjuntos de dados muito grandes
- Funcionalidade limitada: BSTs suportam apenas operações de pesquisa, inserção e exclusão