ÁRVORE BINÁRIA DE BUSCA ÓTIMA (OBST)

AUGUSTO CÉSAR FARIAS CARVALHO E VINÍCIUS ALENCAR TEMÓTEO

PAA (2025.2)

MESTRADO - CIÊNCIA DA COMPUTAÇÃO - PROCC
UFS

O que é uma Árvore?

Uma árvore é uma estrutura de dados hierárquica composta por nós. Cada nó contém um valor e pode ter zero ou mais filhos.

- Raiz: O nó inicial da árvore.
- Nó: Cada elemento individual da árvore.
- Folhas: Nós sem filhos.
- Profundidade de um nó: O número de arestas que separa o nó da raiz.
- Altura da árvore: A profundidade do nó mais profundo.

O que é uma Árvore Binária de Busca?

Uma Árvore Binária de Busca (ABB) é uma árvore binária em que:

- Subárvore à esquerda: Contém apenas valores menores que o valor do nó.
- Subárvore à direita: Contém apenas valores maiores que o valor do nó.

Propriedade:

- Isso permite que a busca por um valor na árvore seja realizada de forma eficiente, seguindo um caminho de comparação de nós até encontrar o valor desejado.

O que é uma Árvore Binária de Busca Ótima (OBST)

Uma Árvore Binária de Busca Ótima (OBST) é uma árvore binária de busca que é organizada de forma a minimizar o custo total de busca, levando em consideração a frequência de acesso de cada chave.

- Objetivo: Minimizar o custo de busca, colocando as chaves mais acessadas mais próximas da raiz, o que reduz o número de comparações feitas.

- Custo de busca: O custo é a profundidade de cada nó, multiplicada pela frequência de acesso daquele nó.

Como Funciona o Cálculo de Custo?

- O custo de busca de uma árvore binária de busca é dado pela soma da profundidade de cada nó multiplicada pela frequência de acesso. Chaves com maior frequência devem estar mais próximas da raiz para minimizar o custo total.

  Custo total = ∑(Profundidade do nó x Frequência de acesso)

Como Construir uma OBST?

A construção de uma Árvore Binária de Busca Ótima (OBST) é feita de forma a minimizar o custo de busca. Para isso, utilizamos programação dinâmica para calcular a árvore com o menor custo possível.

- Passos principais:

- Tabela de custos: Criamos uma tabela onde calculamos o custo mínimo de uma subárvore para cada intervalo de chaves.
- Fórmula de recursão: Calculamos o custo mínimo considerando todas as possíveis raízes intermediárias para cada subárvore.
- Construção da árvore: A partir da tabela de custos, reconstruímos a árvore ótima.

Programação Dinâmica para OBST

A programação dinâmica é utilizada para resolver o problema de forma eficiente. O objetivo é **evitar recalcular soluções para subproblemas**, armazenando os resultados intermediários.

Passos do algoritmo:

- Inicialize uma tabela de custos C[i][j] que representa o custo mínimo da árvore que contém as chaves ki ate kj
- Calcule os custos de cada subárvore usando a **fórmula recursiva**.
- Para cada intervalo de chaves, escolha a raiz que minimiza o custo total da subárvore.

Considere as seguintes chaves e suas frequências de acesso:

O objetivo é construir uma árvore binária de busca ótima que minimize o custo total de busca.

- Passo 1: Organize as chaves em ordem crescente e calcule o custo das subárvores para diferentes combinações de raízes.
- Passo 2: Use a programação dinâmica para calcular o custo mínimo para diferentes subárvores e combinações de raízes.
- Passo 3: Reconstrua a árvore ótima a partir dos resultados da tabela de custos.

Resultado: Mostre a árvore binária de busca ótima com a raiz que minimiza o custo.

Complexidade do Algoritmo

O algoritmo de programação dinâmica para construir a **Árvore Binária de Busca Ótima** tem uma complexidade de:

- **Tempo**: $O(n^3)$, onde n é o número de chaves.
- **Espaço**: $O(n^2)$, devido ao armazenamento da tabela de custos e das decisões intermediárias.

Isso ocorre porque o algoritmo calcula o custo de todas as subárvores possíveis, o que leva um tempo cúbico.

Conclusão

- Árvore Binária de Busca Ótima (OBST): Uma árvore otimizada para minimizar o custo de busca, levando em consideração a frequência de acesso de cada chave.
- Algoritmo de Programação Dinâmica: Usado para calcular a estrutura ótima da árvore e minimizar o custo total de busca.
- Aplicações: A OBST é útil em sistemas como bancos de dados e compiladores, onde o tempo de busca é crucial.