ARVORES E GRAFOS

# Arvores binárias de busca

É uma estrutura de dados de árvore binária baseada em nós, que possui as seguintes propriedades:

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

As operações de pesquisa na árvore binária de busca são muito semelhantes. Digamos que queremos procurar um número, começamos na raiz e depois comparamos o valor a ser pesquisado com o valor da raiz, se for igual, terminamos a pesquisa, se for menor, sabemos que precisamos ir para a subárvore esquerda porque em uma árvore de busca binária todos os elementos na subárvore esquerda são menores e todos os elementos na subárvore direita são maiores.

Pesquisar um elemento na árvore binária de busca é basicamente esse percurso, a cada passo vamos para a esquerda ou para a direita e a cada passo descartamos uma das subárvores.

Forma

Descrição gerada automaticamente com confiança média

## Arvores balanceadas:

Ela é definida como árvore binária quando a diferença entre a altura da subárvore da esquerda e da subárvore da direita não é maior que m, onde m geralmente é igual a 1. A altura de uma árvore é o número de arestas no caminho mais longo entre o nó raiz e o nó folha.

## Arvores B e B+:

**Arvore B** - Ela armazena e mantém os dados em uma forma ordenada onde os filhos esquerdos da raiz são menores que a raiz, e os filhos direitos são maiores que a raiz em valor. Isso torna a busca eficiente e permite todas as operações em tempo logarítmico.

# NOÇÕES BÁSICAS DE GRAFOS:

Grafos são organizações não lineares compostas por um número finito de nós ou vértices e as arestas que os conectam.

Todo grafo consiste em um conjunto de pontos chamados vértices ou nós, que são conectados por linhas chamadas arestas, a partir daí as representações mais frequentes são a matriz de adjacências e a lista de adjacências.

A matriz de adjacência é uma representação sequencial usada para mostrar quais nós estão próximos uns dos outros, ou seja, ela verifica se existe alguma conexão entre nós em um grafo.

A lista de adjacência é uma representação vinculada, podendo manter uma lista de vizinhos para cada vértice no grafo nesta representação. Isso significa que cada vértice no grafo tem uma lista de seus vértices vizinhos.

Além das representações de grafos, também existem as operações que podem ser executadas nestes em estruturas de dados. São elas: a criação de grafos, inserção de vértice, exclusão de vértice, inserção de aresta e exclusão de aresta.

* **Inserir vértice** é quando adicionamos um vértice que, após a introdução de um ou mais vértices ou nós, o tamanho do grafo cresce em um, aumentando o tamanho da matriz em um nó, níveis de linha e coluna.
* A **exclusão de um vértice** refere-se à remoção de um nó ou vértice específico de um grafo que foi salvo. Se um nó removido aparecer no grafo, a matriz retornará esse nó. Se um nó excluído não aparecer no grafo, a matriz retornará o nó não disponível.
* **Inserir aresta** é quando a conexão de dois vértices fornecidos pode ser usada para adicionar uma aresta a um grafo.
* Em **excluir aresta** a conexão entre os vértices ou nós pode ser removida para excluir uma aresta. A

# PROBLEMAS E SOLUÇÕES ENVOLVENDO GRAFOS:

Um dos problemas relacionados aos grafos é o chamado problema do caminho mais curto. Tendo em mente um grafo ponderado, precisamos descobrir o caminho mais curto de um nó para outro, ou seja, aquele que otimiza uma função de custo.

Os problemas do caminho mais curto podem ser resolvidos com algoritmos que podem nos levar do nó A ao B com custo mínimo. Para isso, os algoritmos como:

* Breadth First Search
* Dijkstra
* Bellman Ford
* Floyd-Warshall

**Breadth First Search -** O algoritmo visita e marca com eficiência todos os nós principais em um grafo de maneira precisa. Ele seleciona um único nó (ponto inicial ou de origem) em um grafo e então visita todos os nós adjacentes ao nó selecionado. O BFS acessa esses nós um por um.

Depois que o algoritmo visita e marca o nó inicial, ele se move em direção aos nós não visitados mais próximos e os analisa. Uma vez visitados, todos os nós são marcados. Essas iterações continuam até que todos os nós do grafo tenham sido visitados e marcados com sucesso.

**Dijkstra** - Nos permite encontrar o caminho mais curto entre quaisquer dois vértices de um grafo.

**Bellman Ford** - Nos ajuda a encontrar o caminho mais curto de um vértice para todos os outros vértices de um grafo ponderado. É semelhante ao algoritmo de Dijkstra, mas pode trabalhar com grafos em que as arestas podem ter pesos negativos.

**Floyd-Warshall** - É um algoritmo para encontrar o caminho mais curto entre todos os pares de vértices em um grafo ponderado. Ele funciona tanto para os grafos ponderados direcionados quanto para os não direcionados.

**Depth First Search** - Algoritmo recursivo para pesquisar todos os vértices de um grafo ou estrutura de dados em árvore

**Conectividade** indica que existe um caminho do nó A para o B. A abordagem mais básica para resolver esse problema é com o **Breadth First Search** e **Depth First Search**.

Nos **ciclos negativos**, um grafo teria arestas negativas que podem reduzir ou atrapalhar todo o fluxo. Os algoritmos usados para resolver este problema são o Bellman Ford e o Floyd-Warshall.