Tópicos Avançados em Estrutura de Dados

Atividade 15

Daniel Ughini Xavier	18.00022-3
Vítor Martin Simoni	18.00050-9
Leonardo Cury Haddad	18.00442-3

02/08/2020

Resumo

Para balancear uma árvore, é preciso utilizar algoritmos específicos para tal fim. Esse artigo visa revisar alguns dos proncipais métodos de balanceamento. Assim, serão cobertos as técnicas: AVL, rubro-negra e B-Tree.

Palavras-chave árvore, AVL, Rubo-negra, B-Tree, balanceamento.

Introdução

Árvores de busca são estruturas de dados derivadas da teoria dos grafos. Assim, toda árvore é um grafo planar e bipartido, porém, nem todo grafo é uma árvore. Como o nome já sugere, árvore de busca é uma técnica de organização de dados, onde é possível distribuir uma série de informações de forma que a manipulação dos dados seja eficiente. Com isso, é possível realizar buscas de valores, remover ou inserir informação.

Contudo, quanto maior a árvore cresce, mais demorada são suas operações, o que abre caminho para desenvolvimento de técnicas que visam reorganizar esses dados de forma que o tempo computacional seja o menor possível. O que torna uma árvore eficiente é ela estar equilibrada, ou seja, o lado esquerdo e direito possuem a mesma profundidade. Isso pois, os casos de busca se tornam equilibrados e o pior caso possível (ambas as folhas estão em lados opostos da árvore e à mesma profundidade) é amenizado, pois ambos os caminhos (esquerdo e direito) possuem quase a mesma profundidade. Apesar de existir diversas técnicas de organização de árvores, esse artigo se propõe a discutir apenas três: AVL, rubro-negro e B-Tree.

Desenvolvimento

Uma árvore binária (não vazia) é uma estrutura de dados caracterizada por possuir um elemento distinto (raíz), com duas ligações distintas para cada lado (sub-árvores). Dessa forma, é possível concatenar quantas árvores necessário para formar uma estrutura infinitamente complexa. Vale notar que, como a árvore é composta por sub-árvores, vários algoritmos usam lógica de recursividade para navegar pelos dados.

Para determinarmos o tempo de execução de algum algoritmo, o principal fator a se considerar é a profundidade dos nós em questão. Isso significa que, ao balancear a profundidade dos dois lados de uma árvore, você consegue equilibrar o tempo de execução do programa. Isso pois, o tempo de execução se torna mais homogêneo e o pior caso possível possui uma profundidade semelhante ao caso médio.

Árvores AVL

A AVL é uma árvore altamente balanceada, isto é, nas inserções e exclusões, procura-se executar uma rotina de balanceamento tal que as alturas das sub-árvores esquerda e sub-árvores direita tenham alturas bem próximas. Definição: Uma árvore AVL é uma árvore na qual as alturas das subárvores esquerda e direita de cada nó diferem no máximo por uma unidade.

Rotações em AVL

Na inserção utiliza-se um processo de balanceamento que pode ser de 4 tipos específicos:

RR	caso Right-Right	(rotação a esquerda)
LL	caso Left-Left	(rotação a direita)
LR	caso Left-Right	(rotação esquerda-direita)
RL	caso Right-Left	(rotação direita-esquerda)

Fator de Balanceamento

Coeficiente que serve como referência para verificar se uma árvore AVL está ou não balanceada O fator é calculado nó a nó e leva em consideração a diferença das alturas das sub-árvores da direita e da esquerda.

Árvores rubro-negra

Uma árvore rubro-negra é uma árvore binária de busca em que cada nó é constituído dos seguintes campos: cor (1 bit): pode ser vermelho ou preto. key (e.g. inteiro): indica o valor de uma chave. left, right: ponteiros que apontam para a subárvore esquerda e direita. pai: ponteiro que aponta para o nó pai.

Propriedades da árvore rubro-negra

Uma árvore rubro-negra é uma árvore binária de busca, com algumas propriedades adicionais. Quando um nó não possui um filho (esquerdo ou direito) então vamos supor que ao invés de apontar para nil, ele aponta para um nó fictício, que será uma folha da árvore. Assim, todos os nós internos contêm chaves e todas as folhas são nós fictícios. As propriedades da árvore rubro-negra são: 1 Todo nó da árvore ou é vermelho ou é preto. 2 A raiz é preta. 3 Toda folha (nil) é preta. 4 Se um nó é vermelho, então ambos os filhos são pretos. 5 Para todo nó, todos os caminhos do nó até as folhas descendentes contêm o mesmo número de nós pretos.

Consideração prática Considere uma árvore rubro-negra e o ponteiro T apontando para a raiz da árvore, os nós internos de uma árvore rubro-negra representam as chaves, as folhas são apenas nós fictícios colocados no lugar dos ponteiros nil. Assim, dedicaremos nossa atenção aos nós internos. Por economia de espaço, ao invés de representar todas as folhas, podemos fazer todos os ponteiros nil apontarem para uma mesma folha, chamada nó sentinela, que será indicado por nil[T].

Árvores B-Tree

Árvores B são estruturas usada para implementar TSs (tabelas de símbolos) muito grandes. Uma árvore B pode ser vista como um índice para uma coleção de pequenas TSs: o índice diz em qual das pequenas TSs está a chave que você procura. Pode-se dizer que uma árvore B é uma TS de TSs. Definição (Bayer e McCreight, 1972): Para qualquer inteiro positivo par M, uma arvore B (B-tree) de ordem M é uma árvore com as seguintes propriedades:

- cada nó contém no máximo M1 chaves,
- a raiz contém no mínimo 2 chaves e cada um dos demais nós contém no mínimo M/2 chaves,
- cada nó que não seja uma folha tem um filho para cada uma de suas chaves,
- todos os caminhos da raiz até uma folha têm o mesmo comprimento (ou seja, a árvore é perfeitamente balanceada).

Uma árvore B de ordem 4 é essencialmente uma árvore 2-3 (embora existam algumas difereças que destacaremos adiante).

Aplicações :

Árvores B são a estrutura subjacente a muitos sistemas de arquivos e bancos de dados. Por exemplo:

- o sistema de arquivos NTFS do Windows,
- o sistema de arquivos HFS do Mac,
- os sistemas de arquivos ReiserFS, XFS, Ext3FS, JFS do Linux, e
- os bancos de dados ORACLE, DB2, INGRES, SQL e PostgreSQL.

Desempenho O consumo de tempo de cada operação na memória rápida é muito menor que o tempo necessário para trazer uma página da memória lenta para a rápida ou levar uma página da memória rápida para a lenta. Assim, faz sentido ignorar o custo das operações na memória rápida e contar apenas o número de sondagens.

Considerações finais

Após desenvolver os conceitos da árovre AVL, rubro-negra e B-Tree, é possível destacar que, cada uma possui um cenário ideal de aplicação. Nenhuma solução é perfeita para todos os casos possíveis e, é preciso analisar qual tipo de árvore é mais adequada para a situação.

Primeiramente, a árvore AVL, por focar exclusivamente em equilibras os dois lados da árvore, ela é uma boa opção para quando é preciso estabilidade no manuseio da inforrmação. Logo, ela é mais indicada para operações de inserção, busca e remoção de dados. Isso pois, o tempo para realizar tais funções é $O(logn)$. Com isso, para grande volume de buscas ela é, inclusive mais rápida que a árvore rubro-negra. Porém, para inserções e remoções, a rubro

leva vantagem. Como a árvore rubro-negra é melhor para inserções e remoções, ela é mais indicada para operações críticas, em tempo real. Já a B-Tree é mais indicada para bancos de dados.

Referências Bibliográficas

· T. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein. Introduction to Algorithms, 2ª edição, The MIT Press, 2005.

· R. Sedgewick, Algorithms in C (parts 1-4), 3rd. edition, Addison-Wesley/Longman, 1998.

· https://www.ime.usp.br/ yw/publications/books/TeoriaDosGrafos.pdf