Universidade Federal do Ceará
Campus de Quixadá

Estrutura de Dados Avançada

Relatório do Trabalho Final sobre Árvores Balanceadas

Alunas: Bárbara Stéphanie Neves e Joyce Nayne Araújo
Professor: Fábio Carlos Sousa Dias

Julho
2017

Universidade Federal do Ceará
Campus de Quixadá

Estrutura de Dados Avançada

Relatório do Trabalho Final sobre Árvores Balanceadas

Relatório da disciplina de Estrutura de Dados Avançada (EDA) que consiste de experimentos computacionais para comparar o desempenho de três tipos de árvore.

Alunas: Bárbara Stéphanie Neves e Joyce Nayne Araújo

Matrículas: 388713 e 383868

Professor: Fábio Carlos Sousa Dias

Curso: Ciência da Computação

Julho
2017

Conteúdo

1 Apresentação

Este relatório consiste de experimentos computacionais para comparar o desempenho das seguintes estruturas de dados: árvore AVL, Splay e Rubro-Negra. Estas são árvores balanceadas que possuem as operações principais de inserção, busca e remoção. Os experimentos consistem em três tipos de instâncias que, para cada instância, é escolhido qual atributo será a chave de busca na árvore.

O par dado para este trabalho – par: (instância, chave) –, foi de Empresa e Nome. Cada instância possui os dados de Empresa que são: nome, CNPJ, inscrição estadual, data de abertura, site, e-mail, CEP, rua, número, bairro, cidade, estado, telefone e celular. Assim, usaremos ***nome*** como a chave de busca. Dessa forma, implementamos a classe da instância e utilizamos uma regra que realiza a comparação, para possibilitar a construção das árvores balanceadas.

Para os testes de desempenho, foi calculado o custo computacional – dado em milissegundos (*ms*) – das três operações: inserir, buscar e remover. Posteriormente, misturamos as três operações. Para o teste da operação de inserção, foram inseridos todos os registros de cada instância. E, para a busca e para a remoção, foi selecionado, aleatoriamente, 30% dos registros contidos na instância para realizar a busca do registro na árvore.

Para os testes em que são usados as três operações conjuntamente, a cada 20% dos registros inseridos, foram selecionados 30% dos que já foram inseridos para realizar a busca, e 10% para realizar a remoção. Adiante encontram-se as definições das árvores, a máquina utilizada para rodar os testes, e a conclusão dos resultados.

2 Configurações da Máquina

O computador usado para fazer a comparação computacional possui 3,8 GiB de memória, um processador Intel® Core^{TM} i3-4005U CPU @ 1.70GHz x 4, o sistema operacional é o Ubuntu 14.04 LTS, e o tipo de sistema é de 64-bit.

3 Resultados Computacionais

A seguir, são apresentadas tabelas contendo os tempos computacionais de implementações que utilizam as três árvores balanceadas vistas durante a disciplina: árvore AVL, árvore Splay e árvore Rubro-Negra.

Todas elas simulam uma aplicação que foi explicada em **1. Apresentação**. As tabelas abaixo especificam o tipo de arquivo em que as estruturas trabalharam, o tipo de operação que realizaram – se foi de inserção (I), busca (B) e remoção (R) –, e o tempo total que cada estrutura levou para rodar a instância pela chave (***nome***), tempo este dado em milissegundos.

3.1 Árvore AVL

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

Os testes de desempenho para esta árvore serão realizados calculando o custo computacional das três operações que ela possui da seguinte forma:

1. Para o teste da operação de inserção, devem ser inseridos todos os registros de cada instância.

2. Para o teste da busca, devem ser selecionados, aleatoriamente, 30% dos registros.

3. Para o teste da remoção, devem ser selecionados também, aleatoriamente, 30% dos registros.

Além dos três testes acima, também foi feito o teste que usa as três operações conjuntamente: a cada 20% dos registros inseridos, selecione 30% dos que já foram inseridos para realizar a busca, e 10% para a remoção.

Tabela 1: Custo computacional de cada operação de acordo com os três primeiros testes

Operações	I (100%)	B (30%)	R (30%)	**TOTAL**
Tempo	45 *ms*	7 *ms*	4 *ms*	56 milissegundos

Tabela 2: Resultados do último teste

Operações	Inserção, Busca e Remoção
Tempo total	22 milissegundos

3.2 Árvore Splay

A árvore Splay é mais simples que a AVL pois não força o equilíbrio e não mantém a informação de altura. É uma árvore auto-ajustável, as alterações tendem ao equilíbrio, e é utilizada para aplicações específicas, onde se realizam uma sequência de operações em um universo ordenado.

Em todas operações a árvore faz *Splay*. Essa função consiste em trazer um elemento *n* para a raiz da árvore utilizando sucessivas rotações e tantas quanto necessárias. Os objetivos são minimizar o número de acessos para achar a chave requerida e otimizar a eficiência das operações, através da frequência com que cada nó é acessado, mantendo estes nós na parte superior da árvore.

Os testes de desempenho também serão feitos da mesma forma para a Árvore Splay:

1. Para o teste da operação de inserção, devem ser inseridos todos os registros de cada instância.

2. Para o teste da busca, devem ser selecionados, aleatoriamente, 30% dos registros.

3. Para o teste da remoção, devem ser selecionados também, aleatoriamente, 30% dos registros.

Tabela 3: Custo computacional de cada operação de acordo com os três primeiros testes

Operações	I (100%)	B (30%)	R (30%)	**TOTAL**
Tempo	59 *ms*	5 *ms*	6 *ms*	70 milissegundos

Tabela 4: Resultados do último teste

Operações	Inserção, Busca e Remoção
Tempo total	30 milissegundos

3.3 Árvore Rubro-Negra

Uma árvore Rubro-Negra possui um conjunto de operações (inserção, remoção e busca). Porém, essa árvore é mais eficiente às árvores comuns devido ao fato da Rubro-Negra estar sempre balanceada. Este balanceamento se dá justamente pela característica que dá o nome a árvore, que vem de um bit extra em cada nó que determina se esta é *vermelha* ou *preta* dentro do conjunto de regras que rege a árvore. De maneira simplificada, uma árvore Rubro-Negra é uma árvore de busca binária que insere e remove de forma inteligente, para assegurar que a árvore permaneça aproximadamente balanceada.

Como foi feito nas duas árvores acima, os testes de desempenho da Árvore Rubro-Negra seguem o padrão:

Tabela 5: Custo computacional de cada operação de acordo com os três primeiros testes

Operações	I (100%)	B (30%)	R (30%)	**TOTAL**
Tempo	29 *ms*	10 *ms*	8 *ms*	47 milissegundos

4 Conclusão

Com base na implementação das estruturas tratadas neste relatório, obtivemos os seguintes resultados.

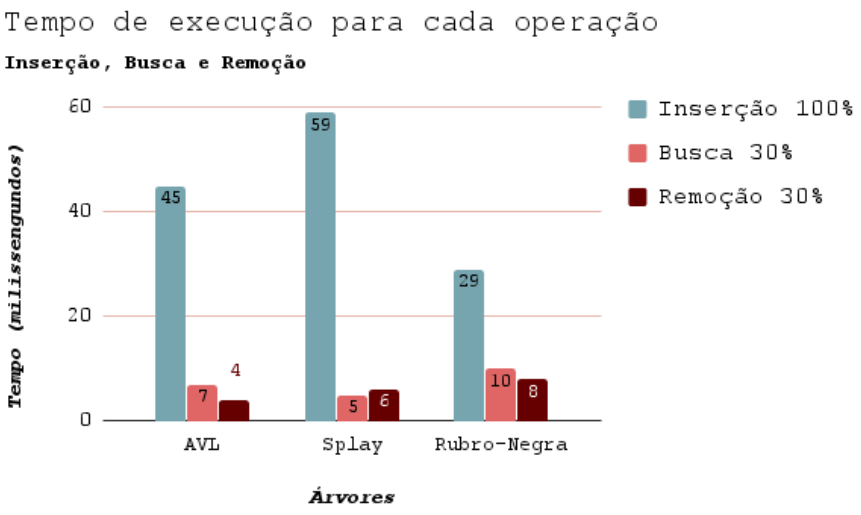

Figura 1: Gráfico do custo computacional de cada operação

O gráfico de barras acima compara o tempo das árvores de acordo com os três primeiros testes. Podemos fazer as seguintes conclusões:

1. A árvore AVL, em comparação com suas outras operações, foi mais rápida na remoção.
2. A árvore Splay foi mais rápida na busca. Com uma diferença de 1 *ms* para a operação de remoção.
3. A árvore Rubro-Negra foi mais rápida na remoção.
4. Comparando o tempo de inserção de cada árvore, a Rubro-Negra possui uma quantidade bem inferior das demais.
5. Por ordem, na inserção, temos: (1) Rubro-Negra, (2) AVL e (3) Splay.
6. Comparando o tempo de busca de cada árvore, a Splay possui uma quantidade bem inferior das demais.

7. Por ordem, na busca, temos: (1) Splay, (2) AVL e (3) Rubro-Negra.

8. Agora, comparando o tempo de remoção de cada árvore, por ordem, temos: (1) AVL, (2) Splay e (3) Rubro-Negra.

De acordo com os dados acima, a árvore AVL acaba tendo uma pequena vantagem em comparação com as demais. Apesar da Rubro-Negra ser bem mais rápida na inserção, e da AVL ser bem parecida com a Splay, o desempenho dela com a aplicação desse trabalho e dos três primeiros testes, foi o melhor.

Tivemos um pequeno problema no último teste, pois não conseguimos resultados na aplicação dele na árvore Rubro-Negra. Portanto, analisaremos o desempenho de somente duas árvores:

Tabela 6: Tabela das operações conjuntas

Árvores Bal.	AVL	Splay	**TEMPO**
TEMPO	22 *ms*	30 *ms*	52 milissegundos

Com base na tabela acima, novamente a AVL se torna mais eficiente. Uma explicação convincente para esse resultado deve-se ao fato de que a árvore AVL, além de executar as operações de inserção, busca e remoção em tempo O(log_n) (no qual n é o número de elementos da árvore), ela é bem rápida que a árvore Rubro-Negra para aplicações que fazem uma quantidade excessiva de buscas. Porém, como mostrado no gráfico acima, a Rubro-Negra é um pouco mais lenta para inserção e remoção. Isso se deve ao fato de as árvores AVL serem mais rigidamente balanceadas.