INSTITUTO DE CIÊNCIA E TECNOLOGIA
Bacharelado em Ciência e Tecnologia
Disciplina: Algoritmos e Estruturas de Dados II
Profa. Lilian Berton

Relatório

Trabalho II: **Recuperação de textos usando Árvore B**

Renata Sendreti Broder, 112347
nov./2017

INTRODUÇÃO

A proposta do trabalho referente ao segundo módulo da Unidade Curricular foi a de se criar um repositório de aproximadamente 30 textos curtos, salvos em arquivos .txt, que deveriam, de alguma maneira, ser armazenados na estrutura de uma árvore B de acordo com o código do texto. Para tanto, foi necessário implementar a estrutura de uma árvore B: seus métodos de inserção, remoção, impressão e busca. Ademais, foi proposto que se implementasse uma *query*, de modo que, ao usuário é pedido que digite uma palavra a ser buscada nos textos armazenados na árvore. Se a palavra não é uma das *stopwords* - uma lista de palavras que podem ser consideradas irrelevantes no processamento do texto - e é encontrada em algum deles em frequência significativa, deve-se abrir o texto na tela para o usuário. Para isto, foi necessário implementar métodos para calcular a frequência das palavras em cada texto inserido - a frequência considerada significativa foi definida como maior ou igual a 3.

Árvore B

A árvore B foi projetada principalmente para o acesso em memória secundária, minimizando o acesso a ela. Desta maneira, a sua página - nome da estrutura que consiste no nó da árvore - não possui apenas um item, mas uma quantidade pré definida de itens, estabelecida por um número t, assim, a quantidade máxima de itens de uma página é (2*t)-1 e a quantidade mínima é t-1. Uma característica importante da árvore B é que a altura de todas as páginas folhas é sempre a mesma, portanto ela está sempre balanceada.

DESENVOLVIMENTO

A resolução do trabalho foi feita em Java e, para tanto, três classes foram implementadas. O motivo da escolha desta linguagem foi o aprendizado de uma nova, pois foi pretendido integrar os conhecimentos apreendidos nas UCs Algoritmos e Estrutura de Dados II e Programação Orientada a Objetos, sendo este o primeiro projeto por mim desenvolvido em Java.

Classe ‘MeuItem’

O objeto declarado como ‘MeuItem’ tem o propósito de armazenar os dados referentes a um item de texto, tendo seu código, o título, o caminho e o vetor de frequência de palavras como atributos.

```
public class MeuItem{

    private int codTexto;
    private String tituloTexto;
    private String caminhoTexto;
    private HashMap<String, Integer> vetorFreq;

...
}
```

É este MeuItem que será a estrutura considerada para cada item a ser inserido ou removido da árvore. A necessidade de cada atributo se explicita a seguir:

- Código do texto: é este código que é considerado para a inserção na árvore, de modo que a lógica desta consiste em verificar se o valor deste é maior ou menor do que os textos já inseridos na árvore, para encontrar o lugar no qual o novo texto deve ser inserido. É também possível realizar buscas a partir deste código, o algoritmo para esta tem a mesma lógica da inserção;
- Título do texto: este é armazenado para que seja mostrado na impressão da árvore, sendo possível saber - para além do código, que não diz nada sobre o texto - mais sobre o texto;
- Caminho do Texto: o resultado da query, se retornar algum texto, deve abrir na tela para o usuário, por este motivo, é necessário guardar o *path*;
- Vetor de frequência: utilizando o hash map é possível acessar uma posição específica com pouco custo. Como a query é baseada em uma palavra, que é a *key* do hash map, é possível resolver rapidamente se ela deve ou não ser considerada significativa para os textos.

Execução e métodos

Os 32 textos foram escolhidos arbitrariamente a partir de assuntos de interesse próprio nos sites CartaCapital; revista piauí - perfis, reportagens, ensaios e humor; Nexo Jornal; e Omelete.

Após salvos, os textos foram armazenados em um diretório específico para que pudessem ser acessados pelo programa. A classe, que possui o método principal, é executada e oferece a opção de os itens de texto serem armazenados na árvore pela ordem natural ou por uma ordem definida pelo usuário. Os textos vão sendo lidos pelo programa - é neste momento que se eliminam as *stopwords* e os caracteres especiais e se armazenam o código, o título, o caminho do texto e o vetor de frequência de palavras correspondente - e, em seguida, inseridos na árvore.

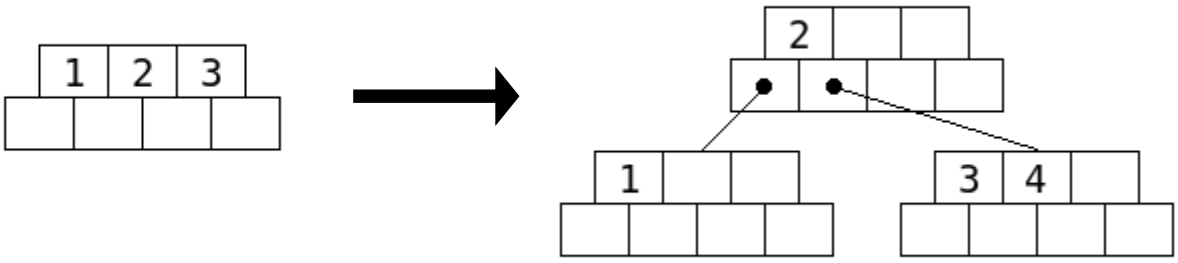
```
ÁRVORE B

Digite a quantidade [1, 32] de textos que serão inseridos na árvore:
7
Como você deseja inserir os textos na árvore?
    Ordem natural (de 1 até 7) (1)
    Manualmente (2)
1
Inserindo em ordem...
```

A inserção pode ser feita na ordem natural, de 1 a n - sendo n a quantidade de textos que o usuário deseja inserir, para este trabalho, limitado a 32 -, ou manualmente, em uma ordem definida pelo usuário. Em ambos os casos, a inserção é feita da mesma maneira, pois a Árvore B tem regras pré definidas. Deste modo, a inserção de um item se inicia com a busca pelo lugar que ele deve ocupar na árvore, pois os itens em uma página se encontram em ordem crescente, e as páginas filhas, sejam elas compartilhadas ou não por um item, contém itens que são necessariamente maiores do que os das páginas à sua esquerda e menores do que o das páginas à sua direita.

Ademais, as quantidades mínima e máxima de itens em uma página são pré-definidas. Neste caso, t = 2, portanto a quantidade máxima de itens é 3 e a mínima, 1, e quando uma página atinge a quantidade máxima, é necessário particioná-la na metade, de maneira que o item do meio vai para a página em um nível inferior - quanto mais inferior, mais próximo da raiz - e a árvore aumenta de tamanho em nível. Neste processo,todos os filhos de cada todos os filhos de cada é necessário prestar atenção à quantidade mínima de itens de uma página.

A imagem abaixo exemplifica a inserção do 4:

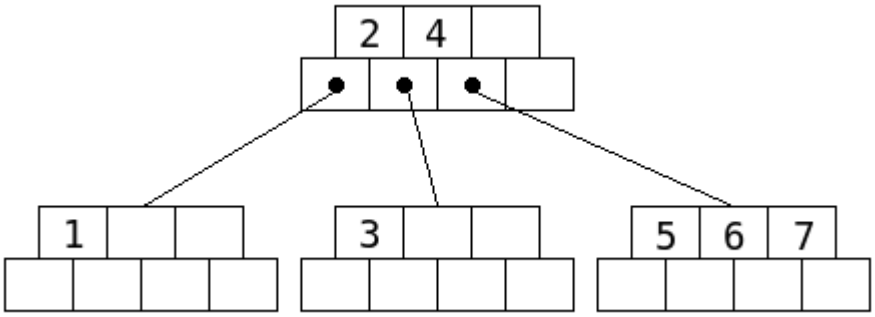


Como a página estava cheia, o item 2 foi para o nível inferior e o 4, por ser maior do que o 2, foi para a página à sua direita, ocupando a posição também à direita do 3.

As opções oferecidas no menu exibido ao usuário são as que se pode ver na imagem abaixo. Para que o programa pare de executar, basta digitar qualquer outro número. O menu apenas se encontra disponível se a árvore contém pelo menos um item, pois todas as funções disponíveis são implementadas unicamente na árvore.

```
O que você deseja fazer agora?
    Imprimir a árvore (1)
    Realizar uma query (2)
    Remover um texto pelo código (3)
    Buscar e abrir um texto pelo código (4)
```

Para a opção (1), a árvore é impressa de maneira que seja impresso, inicialmente, a página raiz, que está no nível 0 e, em seguida, todos os seus filhos. Para melhor entendimento, abaixo encontra-se um esquema da árvore com os 7 textos inseridos de acordo com a ordem dos números naturais. A estrutura da página, formada por 7 espaços, possui um vetor de tamanho máximo 3 acima, e, abaixo, os apontadores para as páginas filhas. Observe que na página pai existem 2 itens - ou seja, o 2 e o 4 - e esta página possui a quantidade de página filhas igual a quantidade de itens da página +1.



Esquema da Árvore B com 7 itens inseridos de acordo com a ordem dos números naturais

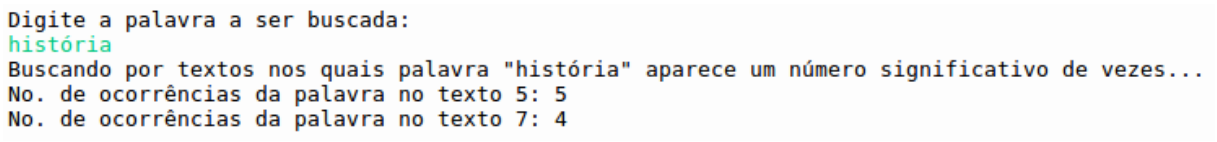
Na impressão por nível, inicialmente são impressos aqueles que estão no nível 0, que é a raiz. Depois, para cada item da página, são impressos os itens da página que se encontra mais à esquerda - observe, na imagem abaixo, que na linha após “Esq (2)”, é impresso apenas o item 1, que está na página filha mais à esquerda do item 2. Como dois itens - 2 e 4 - compartilham uma página filha, esta é considerada filha da esquerda do item à direita, ou seja, do 4. Como a quantidade de filhos de uma página é igual à quantidade de itens +1, apenas o item mais à direita possui um filho da direita. Neste caso, a página filha da direita do item 4 é aquela que contém os itens 5, 6 e 7. Se a árvore possuísse mais níveis, ela faria o mesmo procedimento para cada página, iniciando sempre pelo que está mais à esquerda - seja a página ou o item. Na captura de tela exibida abaixo, foram removidos os títulos do texto para melhor visualização e entendimento.

```
ÁrvoreB:
        Nivel 0: | CodTexto: 2 | CodTexto: 4|
  Esq (2):
        Nivel 1: | CodTexto: 1|
  Esq (4):
        Nivel 1: | CodTexto: 3|
  Dir (4):
        Nivel 1: | CodTexto: 5 | CodTexto: 6 | CodTexto: 7|
```

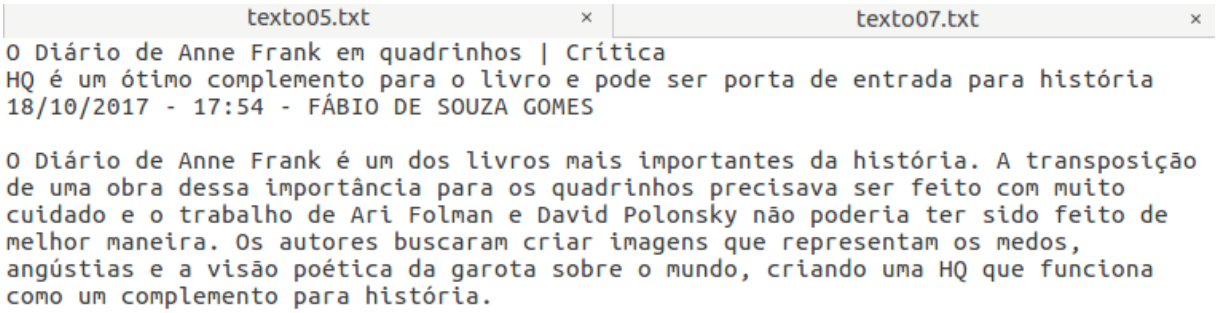
Exemplo da Árvore B impressa por nível

Na opção (2), o usuário digita uma palavra que será buscada no vetor de frequência - implementado através de um *hash map* - de todos os itens presentes na árvore. Se a palavra for encontrada em uma frequência maior do que duas palavras em um determinado texto, este é aberto na tela para o usuário, e o programa retorna

o número de ocorrências juntamente ao código do texto. É possível observar a realização de uma query pela imagem abaixo:



Os textos 5 e 7, respectivamente com 5 e 4 ocorrências da palavra buscada “história” é aberto para o usuário:



A opção (3) permite ao usuário remover um texto da árvore. Os métodos de remoção se iniciam com a busca pelo item, pois ele só pode ser removido se estiver na árvore, caso contrário, o programa retorna uma mensagem. Ao encontrar o item, se ele estiver na folha, ele é removido e os itens à sua direita são deslocados uma posição para a esquerda, no entanto deve-se prestar atenção à quantidade mínima, neste caso igual a 1, estabelecida pelo t. Se a página da qual o item removido agora tiver menos do que 1 item, deve-se olhar a quantidade das páginas adjacentes, olhando a página irmã que possui mais itens e pegando um item emprestado desta; no entanto, se esta não possuir mais do que o mínimo+1, ela deve ser fundida juntando com os itens da página pai; ao ocorrer o estouro - ou seja, a quantidade máxima ser atingida -, o item do meio ocupa o lugar na página pai e os à direita dele ocupam o lugar na página da qual o item foi removido.

Se o item estiver em uma página interna, ele deve ser trocado de lugar com a sua chave sucessora imediata, que poderá ocupar seu lugar sem distorcer a lógica da árvore, virando um item de página folha e sendo tratado como descrito no parágrafo acima.

Para a opção (4) foi implementada uma busca semelhante à busca da remoção. O usuário digita o código do texto a se buscar na árvore e, se ele for encontrado, o texto é aberto na tela, caso contrário, retorna-se uma mensagem informando que o item não está na árvore.

CONSIDERAÇÕES FINAIS

Com a realização deste trabalho, foi não apenas possível verificar a alta complexidade da implementação da Árvore B, mas também aprender, de maneira mais próxima e prática do que apenas teórica, como o algoritmo funciona. Ademais, foi necessário pensar em problemas reais enfrentados na manipulação de arquivos e no processamento de texto, ainda que de forma menos complexa do que como é feita “no mundo real” - isto é, fora da academia.

Portanto, o processo de implementação do trabalho acabou por ser realmente enriquecedor para o aprendizado de conteúdo que está na ementa da UC e também fora dela. Deve-se também ressaltar que, por a proposta ser criativa, ela acabou por ser desafiadora, fator que motivou o desenvolvimento do trabalho.

REFERÊNCIAS

ZIVIANI, Nivio. **Projetos de algoritmos com implementações em Java e C++:** Implementações. Disponível em <http://www2.dcc.ufmg.br/livros/algoritmos-java/implementacoes-06.php>. Acesso em: 18 out. 2017.

AGUIAR CIFERRI, Cristina Dutra de. **Árvore B, B* e B+:** Slides. Disponível em <http://wiki.icmc.usp.br/images/8/8e/SCC578920131-B.pdf>. Acesso em: 06 nov. 2017.