Universidade Federal do Maranhão
Disciplina: Estrutura de Dados II
Discentes: André Filipe S. Barreto, João Víctor C. Gonçalves

Trabalho Prático II

São Luís-MA
2022

1) Ambiente de trabalho

Para a realização do trabalho e a execução dos testes, foi utilizado o Sistema Operacional Windows 11, 64 bits, 8GB de RAM (e o disco rígido HD), e a IDE utilizada para rodar os programas em Java foi o Intellij IDEA, uma vez que há um bom suporte para debugs e é um ambiente de execução muito prático.
Na criação das estruturas utilizadas (árvores e tabela hash), os termos inseridos dentro dessas, seja os termos provindo dos documentos ou as entradas do usuário, eram de tamanho 10, isto é, os menores eram desconsiderados e os maiores eram cortados com o método de substring.

2) Testes

Para testar as estruturas, três arquivos foram analisados, com as respectivas quantidades de produtos.:

victoriassecret.csv - 695 produtos
amz.csv - 83 produtos
hanky.csv - 623 produtos

Nos arquivos testados, a quantidade de termos presentes no índice invertido, quando implementado em qualquer uma das três estruturas, são:

victoriassecret.csv - 36
amz.csv - 91
hanky.csv - 45

Isso significa que quando os arquivos são lidos, independentemente das árvores ou da tabela hash, o índice invertido terá a mesma quantidade de termos, o que mudará será o tempo de execução e a memória gasta, como pode ser observado nas tabelas abaixo. É importante ressaltar que a quantidade de termos presentes no índice invertido não depende da estrutura utilizada, mas do arquivo a ser lido.

AVL Tree

	amz.csv	victoriassecret.csv	hanky.csv
Memória gasta antes da criação da estrutura	**3 MB**	**3 MB**	**3 MB**
Memória gasta após a criação da estrutura	**5MB**	**22 MB**	**10 MB**
Tempo de execução na criação da estrutura	**136ms**	**269ms**	**186ms**

RB Tree

	amz.csv	victoriassecret.csv	hanky.csv
Memória gasta antes da criação da estrutura	**3 MB**	**3 MB**	**3 MB**
Memória gasta após a criação da estrutura	**5MB**	**22 MB**	**10 MB**
Tempo de execução na criação da estrutura	**34ms**	**198ms**	**110ms**

Hash Encadeado

	amz.csv	victoriassecret.csv	hanky.csv
Memória gasta antes da criação da estrutura	**5 MB**	**3 MB**	**3 MB**
Memória gasta após a criação da estrutura	**5 MB**	**22 MB**	**10 MB**
Tempo de execução na criação da estrutura	**94 ms**	**197ms**	**117 ms**

Para realizar os testes, foi criada uma variável para guardar a memória gasta pelo programa antes da criação da estrutura, e foi observada, através de outra variável criada, a memória gasta após a criação da estrutura.
Para cálculo do tempo de execução, foi utilizado uma função nativa da linguagem JAVA, o *System.currentTimeMillis()*.
As imagens abaixo mostram as variáveis utilizadas e as respectivas saídas.

Memória utilizada pelo programa antes da criação e contador do tempo de execução iniciado (Hash encadeado- arquivo : victoriassrecrets.csv)

```
System.out.println("Memória usada antes da criação da estrutura: " + (runtime.totalMemory()-runtime
.freeMemory())/(1024*1024) + " MB");
long start = System.currentTimeMillis();
Dicionario<String, HashMap<Integer, Par>> dicionario;
```

Memória utilizada pelo programa depois da criação e contador do tempo de execução finalizado

Saída do programa

```
Memória usada antes da criação da estrutura: 3 MB
Memória usada depois da criação da estrutura: 22 MB
Tempo de execução: 190ms
```

Teste 1 :

Arquivo : amz.csv

Estrutura : RBTree

```
Memória usada antes da criação da estrutura: 3 MB
Memória usada depois da criação da estrutura: 5 MB
Tempo de execução: 35ms
```

```
Informe o(s) termo(s) desejados: xflamengoooo crisscross spaceeeeeeeeeee
================================================================================================================
                                        Produtos relevantes:
================================================================================================================
     O limiar é de: 2.209420303898299

       ==>  [81]  Wacoal Women's Amazing Assets Contour Bra (2.209420303898299)
       ==>  [56]  Calvin Klein Women's Modern Cotton Lightly Lined Bralette (2.209420303898299)
```

Nessa busca, há alguns pontos a serem considerados: O único termo existente nos arquivos é “crisscross” , ou seja, os termos “flamengoooo” e “spaceeeeeeeeeee” foram desconsiderados da busca. Outro ponto a se observar é o limiar, ou seja, a relevância dos produtos mostrados. São apenas mostrados os produtos em que os termos procurados possuem relevância são

acima do limiar definido . O limiar é calculado a partir da média das relevâncias dos termos válidos que foram encontrados no índice invertido (O limiar mínimo é a média encontrada). É importante ressaltar que produtos com relevância abaixo do limiar não são retornados/impressos na busca. Após mostrar os produtos relevantes, é retornado também o índice invertido. A listagem dos produtos relevantes está sendo realizada de modo que o produto é mostrada primeiro.

```java
public static double definirLimiar(HashMap<Integer, Produto> produtos){
    // calcula o limiar:      limiar = media das relevancias
    Integer size = produtos.size();
    double limiar = 0;

    for (Produto produto : produtos.values()){
        limiar += produto.getRelevancia();
    }

    limiar = (limiar / (double)size);
    return limiar;
}
```

Cálculo do limiar

É observável que a listagem dos termos do índice invertido está ordenada por ordem alfabética dos termos e cada termo vem acompanhado da lista de seus pares, que estão ordenadas pelo id do produto.

```
Quantidade de termos: 36

adjustable: <1, 15> <1, 23> <1, 26> <1, 34> <2, 38> <1, 61>
advantages: <1, 63>
appearance: <1, 67>
balconette: <1, 10>
breathable: <2, 6> <1, 25> <1, 44> <1, 60> <2, 73> <1, 76> <1, 78>
collection: <1, 2> <1, 7> <1, 72>
comfortabl: <1, 13> <1, 15> <1, 19> <1, 25> <1, 34> <1, 44> <2, 73>
contrastin: <1, 1> <1, 26> <1, 70>
convenienc: <1, 63>
convertibl: <1, 28>
coordinati: <1, 2> <1, 72>
crisscross: <1, 56> <1, 81>
customized: <1, 54>
disappeari: <1, 77>
effortless: <1, 32>
elasticize: <1, 45>
embroidere: <1, 70>
```

3) Conclusão

Através dos testes, foi possível observar que todas as estruturas apresentaram desempenhos similares com relação ao uso de memória, contudo, com diferenças no tempo de execução, sendo a estrutura RBTree a mais eficiente para os casos testados. É importante comentar que os arquivos presentes no dataset possuíam muitos caracteres que atrapalharam na criação do índice invertido da maneira correta, assim, foi necessário um tratamento prévio com várias regex para eliminar essas inconformidades.