*Métodos Probabilísticos para Engenharia Informática Ano letivo 2018/2019*

Docente: Carlos Bastos

**Relatório do trabalho prático de MPEI**

Trabalho realizado por:

João Dias nº89236

Tomás Costa nº89016

**Teste dos Módulos**

Tal como pedido no enunciado, foram feitos 3 programas (módulos), um para o *contador estocástico*, outro para o *Counting Bloom Filter* e outro para descobrir itens similares usando *minhash*. Aqui se segue uma breve descrição de como correr os vários programas.

*Contador Estocástico*

Começa-se por correr o programa *TesteContador.java*, a probabilidade não é pedida, apenas está definida nas variáveis e é-nos dado como return values o número de contagens que fez e a respetiva percentagem.

Exemplo do output esperado:

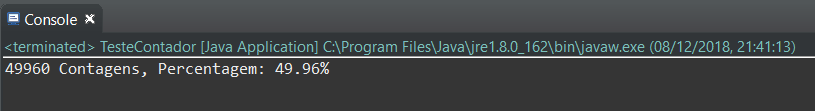


Figura - Output de TesteContador

*Counting Bloom Filter*

Começa-se por correr o programa *testebloom.java,* o programa começa por um criar um filtro, de seguida o ficheiro teste.txt é lido e todas as palavras do mesmo são inseridas no filtro, caso sejam inseridas com sucesso estas são impressas. Posteriormente a isto é-nos ainda possível ver o número de ocorrências de uma palavra, remover uma palavra do Bloom Filter e verificar a presença de uma palavra tudo isto através de um sistema interativo.

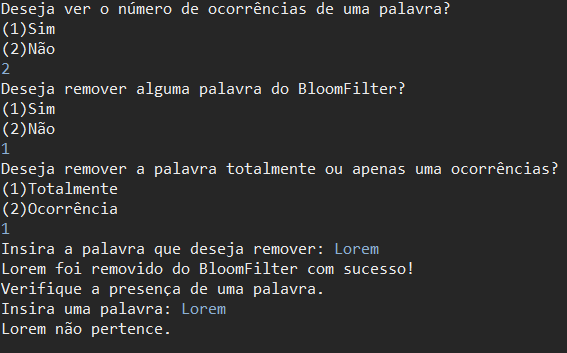
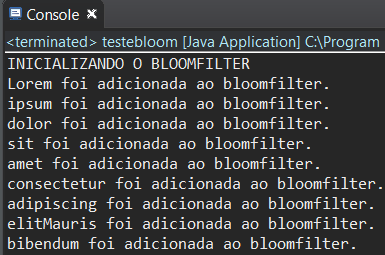


Figura 3- opções disponibilizadas

Figura 2 - output inicial

*Itens Similares*

Começa-se por correr o programa *SimilaresTeste.java,* este lê o file *u.data* que contém os filmes (usados na aula prática), depois calcula a matriz de semelhanças ( 1 – distância) e depois verifica users parecidos conforme o threshold definido.

Dá-nos o par: User1:User2:Semelhança

**Módulo Final**

Neste módulo, agregámos os módulos anteriormente referidos num só de modo a aplicar as diferentes funcionalidades de cada um para um só exercício. Decidimos então usar o conceito de lotaria e explorar tudo que pode ser feito com as variáveis que esta proporciona desde nome de vencedores a datas e números de lotaria associados às mesmas. Para ver este módulo em “ação” começa-se por correr o programa *ModuloFinal.java* e um menu vai ser apresentado.

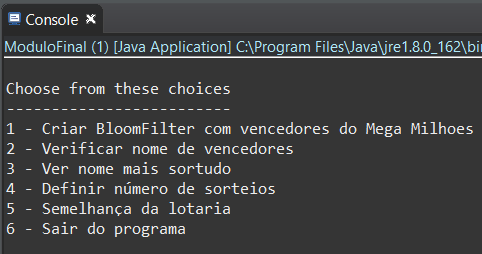


Figura 4 - Menu inicial

Para aceder às opções “2” e “3” temos de primeiro executar a “1” para inicializar o filtro, o mesmo se aplica à opção “5”, para esta ser acedida primeiro temos de executar o “4” visto que é este que define o número de elementos que vamos ler para a semelhança de “5”.

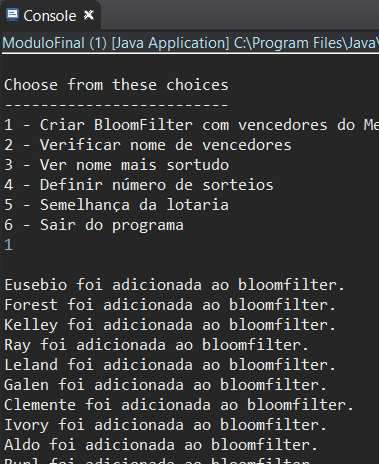
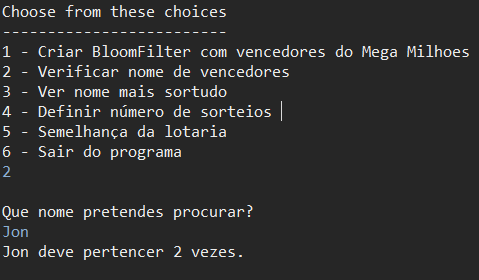
****

Figura 6 – Opção (2) – verifica se um nome pertence (e quantas vezes) ao conjunto de vencedores

Figura 5 – Opção (1) – criam um filtro com os nomes dos vencedores da lotaria

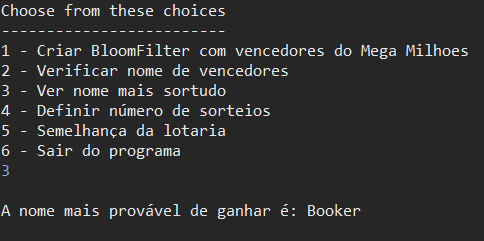
****

Figura 7 – Opção (3) – Vê qual o nome mais predominante

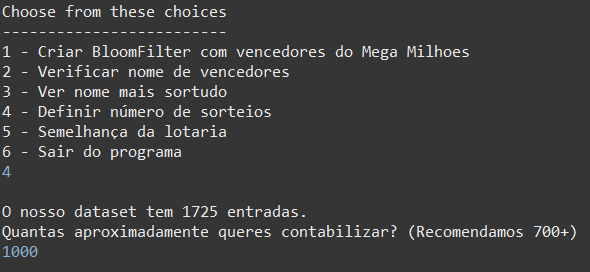
****

Figura 8 – Opção (4) – Define o número de entradas que o user pretende contabilizar

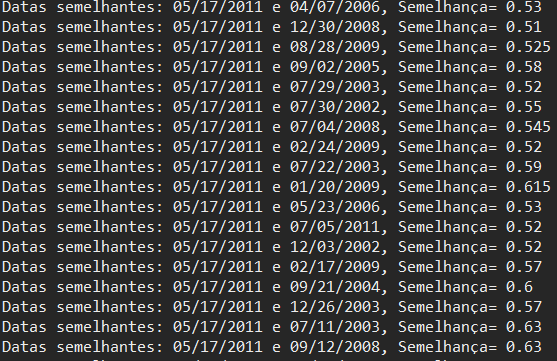
****

Figura 9 – Opção (5) - (parte do output) – Aqui podemos ver os pares similares e a sua semelhança

**Conclusão**

Este trabalho permitiu-nos mesmo ver e explorar ainda mais o poder que as hashfunctions e minhash verdadeiramente têm e aplicá-las num problema real.

Face aos resultados obtidos podemos ver que é muito raro sair a mesma combinação e que, apesar da grande utilidade que estas funções possuem, estas têm sempre as suas desvantagens. Sendo que uma delas, nos Bloom Filters, é a possibilidade de falsos positivos, daí a “incerteza”. Esta incerteza é verdadeiramente problemática em programas que necessitam de exatidão.