Trabalho Prático - WeatherTools

Universidade de Aveiro

João Oliveira, Vasco Sousa

VERSAO FINAL

Trabalho Prático - WeatherTools

DETI

Universidade de Aveiro

João Oliveira, Vasco Sousa
(93282) joaoroliveira@ua.pt, (93049) jvcs@ua.pt

Dezembro de 2019

Capítulo 1

Introdução

O projeto que se segue apresentado visa simular um sistema que permita compararar diversas estações metereológicas. Este permite que o utilizador consiga escolher diversos parâmetros como mês, ano para recolher dados das diversas estações, e de seguida filtrar os dados ou então comparar a sua similaridade. Este projecto encontra-se integrado na disciplina de Métodos Probabilísticos de Engenharia Informática.

Capítulo 2

Manual de Instruções

O projeto é bastante compreensivel e *user friendly*. Para iniciar a aplicação é necessário abrir o arquivo WeatherTools.jar, que deve estar na pasta principal do projeto. Ao executar o programa são apresentadas várias opções como: *comboboxes* que permitem selecionar o mês e o ano, é fornecida a possibilidade de aumentar o zoom do mapa e de seguida clicar num dos pontos vermelhos (estação) e obter todos os seus dados para o mês e ano selecionados. Existe a possibilidade de utilizar dois métodos de filtro *"Average temperature"* e *"Average Rainfall"*, ativando estes filtros pode-se observar quais são as estações com determinadas caracteristicas. Existe também um botão que vai verificar a similiaridade de todas as estações num determinado ano (definido pelo utilizador em cima), tendo em conta o indice de similaridade mínimo definido pelo utilizador.

Capítulo 3

Classes

3.1 Classes/Packages

Breve resumo do que faz cada Classes/Packages utilizadas no trabalho:

3.1.1 Class BloomFilter

Cada estação possui um e nele todos os elementos da estação (Data, Precipitaçã, Temperatura) são passados para uma string que passa por uma função de **hash**, com o codigo gerado (int), acedemos a essa posição no bloomfilter que é colocada a um (1).

3.1.2 Class TestBloomFilter

Classe de teste à eficácia da classe BloomFilter e posterior avaliação com valores conhecidos.

3.1.3 Class MinHash

Classe responsável por realizar todas as operações relacionadas com o MinHash.

3.1.4 Class TestMinHash

Classe de teste à eficácia da classe MinHash e comparação com análise de resultados.

3.1.5 Class Station

Classe que representa uma estação. Esta contém tambem uma *LinkedList* de *RegisteredWeather*.

3.1.6 Class RegisteredWeather

O ResgisteredWeather contem toda a informação sobre um registo de tempo com uma determinada data.

3.1.7 Package Metrics

Package que contém classes que servem para transformar medidas presentes no dataset, como por exemplo graus Celsius, em objetos Java.

3.1.8 Class Dataset

Classe responsável por ler os ficheiros de dados e transforma-los nas classes a cima descritas.

3.1.9 package UI

Utilização de Java Swing para uma a criação de uma interface gráfica para melhorar a experiência do utilizador desta aplicação.

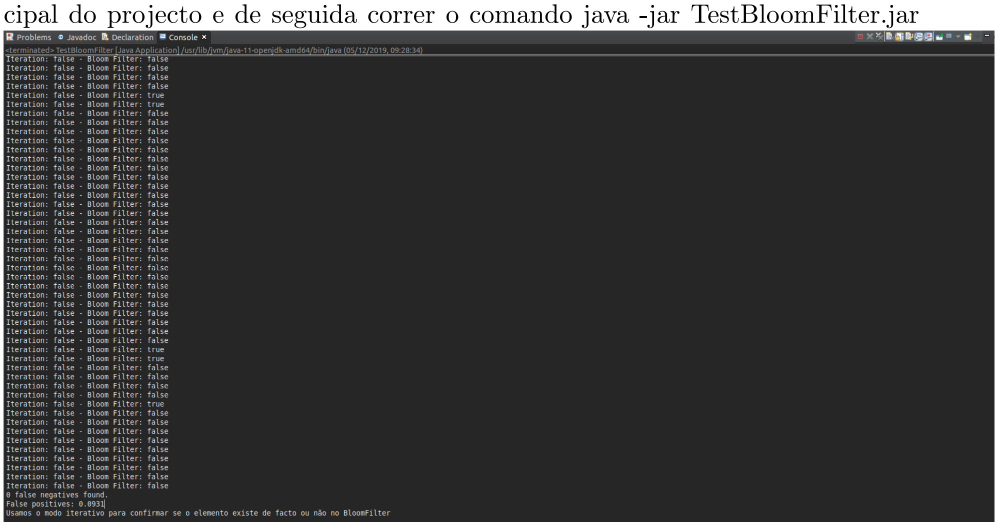
Capítulo 4

Testes

4.1 BloomFilter

4.1.1 Teste BloomFilter

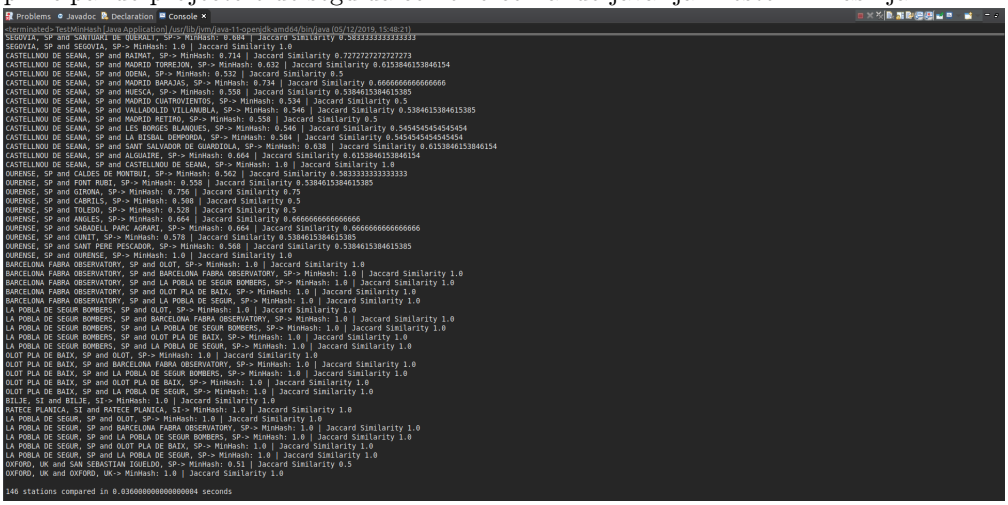
É criado uma lista e um bloomfilter. Após isto, são geradas strings aleatórias que contém data, precipitação e temperatura. Essas mesmas são adicionadas tanto à lista quanto ao bloomfilter e depois é verificado se um determinado elemento também ele aleatório pertence aos dois, tendo sempre a certeza se ele pertence ou não à lista visto que é feita uma pesquisa iterativa. De seguida são calculados os Falsos Negativos (0 como seria de esperar) e os falsos positivos. Para correr este teste deve-se abrir o terminal na pasta principal do projecto e de seguida correr o comando java -jar TestBloomFilter.jar



4.2 MinHash

4.2.1 Teste MinHash

Neste teste é importado o DataSet, gerando a tabela MinHash do dataset importado e após isto compara o grau de similiaridade de todos os dados entre si e apresenta uma lista com o grau de similiariade de cada um. Também foi calculado o indice de Jaccard pela formula teórica para determinar a precisão do nosso algoritimo MinHash. Para correr este teste deve-se abrir o terminal na pasta principal do projecto e de seguida correr o comando java -jar TesteMinHash.jar



Capítulo 5

Conclusões

Em suma os objetivos do trabalho foram todos concretizados nomeadamente, a aplicação do *BloomFilter* e do *MinHash*, este ultimo é mais eficiente para realizar comparações do que o algorítmo iterativo. É reconhecida a limitação do Dataset, pois este apresenta relativamente pouca informação para realizar uma simulação mais real, este facto fez com que tivesse sido necessário arredondar todas as temperaturas às unidades, só assim foi possível ter algumas estações com registos de temperaturas idênticos. Foram cumpridos os objetivos principais estabelecidos pelos docentes da unidade curricular.

Capítulo 6

Contribuições dos autores

Em termos de contribuição dos autores o trabalho foi realizado maioritariamente sempre em conjunto por isso desta forma pode-se dizer que a contribuição será de 50% para os dois elementos do grupo. É também importante realçar a importância da fundação Apache Software pois é a responsável pelo desenvolvimento do package Apache Commons Codec, que permitiu o uso do módulo MurmurHash3 que serve para fazer hash.