Trabalho Prático MPEI

DETI

Universidade de Aveiro

Luís Couto, António Ferreira
(89078) luiscouto10@ua.pt (89082) ajferreira@ua.pt

8/12/2018

Conteúdo

Capítulo 1

Como correr os programas?

1.1 Antes de os correr

De modo a conseguir utilizar cada programa, o utilizador deve primeiramente criar um projeto e em seguida, dentro desse mesmo projeto, seguir as seguintes instruções:

1. Criar uma package chamada "Modulos"e, dentro dessa package, inserir as seguintes classes:
   - ContadorEstoc.java
   - CountingBloomFilter.java
   - MinHash.java
2. Criar uma package chamada "TestesModulos"e, dentro dessa package, inserir as seguintes classes:
   - PrimeiroTesteContadorEstoc.java
   - SegundoTesteContadorEstoc.java
   - PrimeiroTesteCountingBloomFilter.java
   - SegundoTesteCountingBloomFilter.java
   - TerceiroTesteCountingBloomFilter.java
   - QuartoTesteCountingBloomFilter.java
   - TesteMinHash.java
3. Criar uma package chamada "DemonstracaoConjunta"e, dentro dessa package, inserir as seguintes classes:
   - Main.java
4. Guardar os ficheiros *.txt* e *.data* no respetivo diretório de modo a poderem ser usados pelos testes e demonstração conjunta.

1.2 Correr os testes dos módulos

1.2.1 PrimeiroTesteContadorEstoc.java

Para correr este teste, o utilizador apenas precisa de dar *run* ao programa e introduzir uma probabilidade.

1.2.2 SegundoTesteContadorEstoc.java

O utilizador tem de dar *run* ao programa e introduzir uma probabilidade.

1.2.3 PrimeiroTesteCountingBloomFilter.java

Para correr este teste, o utilizador apenas precisa de dar *run* ao programa. Se desejar alterar o contéudo dos membros do Bloom Filter ou o conteúdo do que vai ser testado, pode mudar esses conteúdos nas linhas de código assinaladas com um comentário.

1.2.4 SegundoTesteCountingBloomFilter.java

Para correr este teste, o utilizador precisa de dar *run* ao programa e introduzir o número de hash functions que deseja usar.

1.2.5 TerceiroTesteCountingBloomFilter.java

É necessário apenas que o utilizador dê *run* ao programa e introduza o número de hash functions que deseja usar.

1.2.6 QuartoTesteCountingBloomFilter.java

O utilizador tem de dar *run* ao programa e introduzir o número de hash functions a usar, assim como a palavra que deseja contar quantas vezes aparece no ficheiro.

1.2.7 TesteMinHash.java

Para utilizar este programa o utilizador apenas precisa de dar *run*.

1.3 Correr demonstração conjunta

1.3.1 Main.java

Para utilizar este programa o utilizador apenas de dar *run*. Depois tem de introduzir uma probabilidade e de seguida o número de hash functions. O programa vai apresentar alguns resultados e depois o utilizador terá de introduzir um limiar que, na opinião dele, se for ultrapassado, um documento é considerado plágio de outro.

1.4 Quais os ficheiros a usar nos testes?

Todos os testes que requerem leitura de ficheiros ja vêm com eles escolhidos. No entanto, se o utilizador desejar, pode escolher outros ficheiros para análise, para isso precisa de escrever o nome e extensão do ficheiros que deseja ler nas respetivas linhas de código de cada teste, sendo que estas se encontram assinaladas com um comentário. Tem de se certificar também que esses ficheiros se encontram num diretório adequado que permita a sua leitura.

Capítulo 2

Funcionamento e resultado dos testes

2.0.1 PrimeiroTesteContadorEstoc.java

Este programa serve para testar o contador estocástico de maneira individual. A probabilidade inserida pelo utilizador é a probabilidade de o contador fazer uma contagem.
No final, este programa diz que se o contador contou ou não.

2.0.2 SegundoTesteContadorEstoc.java

Este programa serve para contar as palavras de um ficheiro usando o contador estocástico. O utilizador introduz a probabilidade do contador fazer uma contagem e, mediante essa probabilidade, o contador estocástico lê o ficheiro e faz a contagem das palavras.
No final, este programa diz o número total de palavras do ficheiro, o número de palavras contadas pelo contador estocástico, a percentagem de palavras que foram contadas e a diferença entre a probabilidade introduzida pelo utilizador e a calculada após a contagem.

2.0.3 PrimeiroTesteCountingBloomFilter.java

Neste programa, temos um pequeno Bloom filter preenchido manualmente, e vamos testar a presença de certos países nesse Bloom filter.
No final, o programa diz quais países não estão no Bloom Filter e quais países é que estão, assim como quantas vezes é que cada pais aparece no Bloom Filter.

2.0.4 SegundoTesteCountingBloomFilter.java

Neste programa, temos um certo número de strings geradas aliatóriamente e inseridas num Bloom Filter. Depois temos outro número de strings, também

aleatórias, que vamos testar e ver se pertencem ao Bloom Filter. No final, este programa diz se cada elemento pertence ou não ao Bloom Filter e calcula o número de falsos positivos.

2.0.5 TerceiroTesteCountingBloomFilter.java

Este programa vai comparar dois ficheiros. As palavras do primeiro ficheiro vão ser inseridas num Bloom Filter e, à medida que o segundo ficheiro é lido, o programa verifica se cada palavra desse ficheiro pertence ao Bloom Filter, sendo que imprime todas as palavras que não pertencem.
Resumindo, este programa imprime todas as palavras que pertencem a um ficheiro mas não ao outro, devolvendo também quantas vezes isso acontece (devolve o número total de palavras que pertencem a um ficheiro mas não ao outro).

2.0.6 QuartoTesteCountingBloomFilter.java

Este programa lê um ficheiro e introduz todas as suas palavras num Bloom Filter. Depois, o utilizador insere uma palavra e o programa conta quantas vezes essa palavra aparece no Bloom Filter.
No final, o programa diz ao utilizador quantas vezes a palavra introduzida aparece no ficheiro lido.

2.0.7 TesteMinHash.java

Este programa lê quatro ficheiros e no final diz qual a similaridade entre eles.

Capítulo 3

Demonstração conjunta

A nossa demonstração conjunta consiste em averigar a existência de plágio entre parte de um ***ficheiro1*** e um ***ficheiro2*** completo. Ao escolher a probabilidade, o utilizador vai essencialmente escolher quanto de ***ficheiro1*** é que vai ser comparado à totalidade do ***ficheiro2***, pois o contador, que conta consoante a probabilidade inserida, vai contar quantas palavras existem no ***ficheiro1*** e essas palavras vão ser inseridas num Bloom Filter à medida que são contadas. Depois vai ser verificada a existência de todas as palavras do ***ficheiro2*** no Bloom Filter. O programa tem vários outputs, para ajudar o utilizador a perceber o que está a acontecer, no entanto o resultado que interessa é o ultimo print, dado após o utilizador introduzir um valor para o limiar. Como a similaridade entre ambos os ficheiros é calculada antes, o programa compara-a depois ao limiar introduzido pelo utilizador. De notar que o número de hash functions introduzida pelo utilizador vai influenciar a similaridade.
Se a similaridade for maior ou igual ao limiar, o resultado vai ser que existe plágio.
Se a similaridade for menor que o limiar, o resultado vai ser que não existe plágio.

Uma das limitações deste programa é que considera palavras diferentes as mesmas palavras mas com pontuação ligada. Por exemplo, ***ola.*** e ***ola*** são consideradas palavras diferentes, pois uma delas tem um ponto final, no entanto deviam contar como palavras iguais. Uma maneira de isto não acontecer seria alterar o código de modo a ignorar pontuação.