

# Projeto MPEI

Departamento de Eletrónica, Telecomunicações e  
Informática

Pedro Mateus, Pedro Almeida, Tomás Martins  
(88858) pedro.valente@ua.pt, (89205) pedro22@ua.pt, (89286) tomasfilipe7@ua.pt

11 de Dezembro de 2018

# Capítulo 1

## Explicação

### 1.1 Teste do BloomFilter

*Nota:* Neste projeto temos dois Bloom Filters funcionais, mas apenas é utilizado o `BloomFilter.java`. Isto ocorreu porque se juntou um membro ao nosso grupo após o começo do trabalho e este membro já tinha um Bloom Filter feito, decidimos mantê-lo pois mostra outra implementação possível deste módulo.

Executando o ficheiro `Teste_BF.java` irá ser apresentado um menu que irá ter o seguinte aspeto:

```
-----
      Teste BloomFilter
-----
1 - Adicionar valores
2 - Verificar se valor pertence
3 - Contar quantas vezes aparece o valor
4 - Remover valor
5 - Testar BloomFilter com strings aleatorias
6 - Sair do programa
Escolha:
```

Figura 1.1: Menu do teste do Bloom Filter

Seleccionando a:

- *Opção 1* o utilizador irá poder adicionar um valor através do terminal
- *Opção 2* irá ser pedido ao utilizador que insira um novo valor através do terminal e o programa irá verificar se este já existe no Bloom Filter
- *Opção 3* sendo este um Counting Bloom Filter, o utilizador ao seleccionar esta opção irá inserir um valor e irá de seguida ser apresentado o número de vezes que este existe no Bloom Filter
- *Opção 4* esta opção remove um valor escolhido pelo utilizador
- *Opção 5* se o utilizador seleccionar esta opção irão ser inseridas 10000 strings aleatórias no Bloom Filter e de seguida verifica se outras 1000000

strings pertencem ao Bloom Filter apresentando o número, assim como a percentagem de falsos positivos

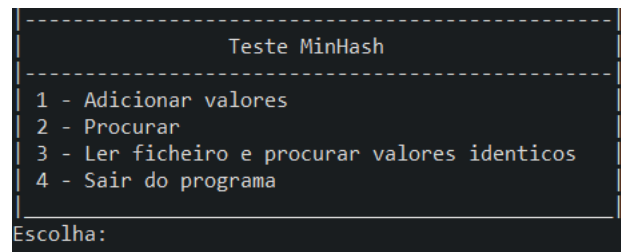
- *Opção 6* esta opção irá encerrar o programa

## 1.2 Teste do Contador Estocástico

Executando o ficheiro `TestCounter.java` irá contar 100 vezes até um milhão, incrementando com uma probabilidade de 0.5. Para cada ciclo ele soma o erro, calculando assim a soma total do erro. Com o erro total é calculado de seguida a respetiva média e a percentagem de erro.

## 1.3 Teste do MinHash

Executando o ficheiro `Test_MinHash.java` irá ser apresentado um menu que irá ter o seguinte aspeto:



```
-----
                        Teste MinHash
-----
1 - Adicionar valores
2 - Procurar
3 - Ler ficheiro e procurar valores identicos
4 - Sair do programa
-----
Escolha:
```

Figura 1.2: Menu do teste da MinHash

Selecionando a:

- *Opção 1* o utilizador poderá adicionar valores através do terminal
- *Opção 2* permite ao utilizador procurar por valores idênticos aos que inseriu previamente (*Opção 1*)
- *Opção 3* imprime o ficheiro `Voos.txt` e pergunta ao utilizador um valor para procurar por elementos idênticos
- *Opção 4* irá encerrar o programa

## 1.4 Main

Executando o ficheiro `ProjectMain.java` irá ser apresentado um menu que irá ter o seguinte aspeto:

```
-----
Main do Projeto
-----
1 - Ver ficheiro
2 - Ver telemoveis
3 - Ver o numero stock de cada loja
4 - Imprimir stock de uma loja
5 - Procurar em todas as lojas
6 - Procurar em uma loja
7 - Sair do programa
Escolha:
```

Figura 1.3: Menu da aplicação da Main

- *Opção 1* esta opção permite ao utilizador visualizar o ficheiro `phones.txt` que contém todas as lojas, telemóveis e preços
- *Opção 2* esta opção permite ao utilizador visualizar todos os telemóveis disponíveis
- *Opção 3* esta opção permite ao utilizador ver a quantidade de telemóveis disponíveis em cada loja
- *Opção 4* o utilizador pode visualizar o stock de uma loja à sua escolha.
- *Opção 5* o utilizador deverá inserir um telemóvel através do terminal e o programa irá procurar por esse telemóvel em todas as lojas
- *Opção 6* o utilizador deverá inserir uma loja através do terminal e o programa irá demonstrar todos os telemóveis dessa loja
- *Opção 7* esta opção irá encerrar o programa

## Capítulo 2

# Resultados

### 2.1 Bloom Filter

**Nota:** a imagem e o texto abaixo corresponde ao teste do BloomFilter com Strings aleatórias (Opção 5 do menu), visto ser possível testar de outras formas.

```
Adicionado com sucesso ao BloomFilter
Adicionado com sucesso ao BloomFilter
Adicionado com sucesso ao BloomFilter
Adicionado com sucesso ao BloomFilter
Verificando...
Terminado!

Falsos positivos em 1000000 strings aleatorias: 10982
Porcentagem de falsos positivos: 0,00110 %
```

Figura 2.1: Resultados do Bloom Filter

Como é possível verificar na figura, todas as 10000 Strings aleatórias foram adicionadas com sucesso ao `BloomFilter.java`. De seguida é verificado se as outras 100000 Strings aleatórias pertencem ao BloomFilter (é esperado que nenhuma pertença visto serem aleatórias). Caso se verifique que a String pertence ao BloomFilter, é assumido como um falso positivo. No final é apresentado o número total de falsos positivos e calculada a sua percentagem, verificando-se bastante reduzida (0,00110 %).

A HashFunction utilizada no BloomFilter foi a *Universal Hashing*. Este tipo de HashFunction já garante que as funções geradas são descorrelacionadas entre si.

## 2.2 Contador Estocástico

90	999056	944
91	1001348	1348
92	999856	144
93	999684	316
94	999234	766
95	1001330	1330
96	1000890	890
97	1000026	26
98	999920	80
99	1001796	1796
100	1001220	1220
-----		
Media de erro -> 718		
Percentagem de erro -> 0.071862%		

Figura 2.2: Resultados do Contador

Na coluna 2 é apresentado o valor do resultado do contador. Pode-se verificar que nem sempre é obtido o valor esperado de 1000000 e assim, na terceira coluna é apresentadar a margem de erro fase ao valor esperado. No final, é feita uma análise de todos os erros, sendo apresentada a média destes, assim como a sua percentagem.

## 2.3 MinHash

**Nota:** a imagem e o texto abaixo corresponde ao teste do módulo MinHash lendo a informação de um ficheiro (Opção 3 do menu), visto ser possível testar de outras formas, como por exemplo, adicionar valores um a um.

```
22:35
LH 1180
Frankfurt
22:40
FR 4599
Memmingen
22:40
TF 347
London, Gatwick
22:45
EZY1457
Geneva
22:55
FR 7475
Paris, Beauvais
23:20
FR 8347
London, Stansted

Procurar:
geneva
geneva --> semelhante a: Geneva | Dist. Jaccard: 0,400000
geneva --> semelhante a: Geneva | Dist. Jaccard: 0,400000
geneva --> semelhante a: Geneva | Dist. Jaccard: 0,400000
geneva --> semelhante a: Geneva | Dist. Jaccard: 0,400000
geneva --> semelhante a: Geneva | Dist. Jaccard: 0,400000
```

Figura 2.3: Resultados MinHash

No teste do módulo da Minhash é introduzido na MinHash o conteúdo de um ficheiro e de seguida é imprimido esse mesmo conteúdo. Depois é perguntado ao utilizador um valor para procurar por valores idênticos que possam estar no conteúdo do ficheiro.

Se forem encontrados valores idênticos, estes são imprimidos no terminal assim como a respetiva distância de Jaccard. Caso não sejam encontrados valores idênticos é apresentada a seguinte mensagem:

```
Paris, Beauvais
23:20
FR 8347
London, Stansted

Procurar:
voos
Nenhum elemento semelhante encontrado!
```

Figura 2.4: Sem resultados

## Capítulo 3

# Aplicação de uso conjunto

Esta aplicação tem como propósito ler um ficheiro de texto que contém lojas, telemóveis e correspondentes preços. O utilizador pode então procurar por um determinado telemóvel, em todas as lojas ou apenas numa loja à escolha, averiguar o stock de cada loja, ver todos os telemóveis existentes e ainda visualizar o próprio ficheiro de texto. É utilizado um BloomFilter para cada loja presente no ficheiro e durante a leitura é verificada a que loja pertence cada telemóvel e respetivo preço, sendo adicionado ao BloomFilter correspondente.

Ainda durante a leitura do ficheiro, é adicionado à minhash todos os telemóveis e respetivos preços. Isto vai permitir procurar elementos parecidos, neste caso, procurar um telemóvel em todas as lojas ou numa só loja.

O stock de cada loja (número de telemóveis) é calculado utilizando o contador estocástico, tal como no Bloom Filter é criado um contador para cada loja.

```
-----Stock-----  
Loja da Esquina : 40 telemoveis disponiveis  
PhoneHouse      : 36 telemoveis disponiveis  
Cheap Sales     : 40 telemoveis disponiveis  
Tele4u          : 42 telemoveis disponiveis  
Compra Aqui     : 34 telemoveis disponiveis
```

Figura 3.1: Stock de cada loja



A procura de um telemóvel é realizada da seguinte maneira:

- Definir o que procurar;
- Minhash procura por elementos semelhantes, retornando uma lista com todos os semelhantes encontrados. Caso não haja a lista fica vazia;
- Todos os elementos presente na lista retornada do ponto anterior, são verificados se pertencem ao BloomFilter de cada loja. Caso seja, imprime esse elemento;

```
Searching for "PHONE~":  
  
Loja da Esquina  
-----  
IPHONEX 300€  
POCOPHONE      449€  
IPHONE6SPLUS   722€  
IPHONE7S       704€  
IPHONE8 520€  
IPHONEXR       660€  
RAZER_PHONE    746€  
IPHONEXS       410€  
IPHONE7 542€  
IPHONE7PLUS    398€
```

Figura 3.2: Exemplo de Procura de Telemóvel