UNIVERSIDADE DE AVEIRO
DEPARTAMENTO DE ELECTRÓNICA, TELECOMUNICAÇÕES E INFORMÁTICA

MPEI - Métodos Probabilísticos para Engenharia Informática (2018/2019) **PL 06**

Palavras-chave: Estruturas de dados probabilísticas, Filtros de Bloom (*Bloom Filters*), Funções de dispersão.

Este trabalho prático tem por objectivo criar e testar um módulo que suporte a criação de *Bloom filters* (ex: [2]). Para tal, execute os seguintes passos:

1) Crie, em Matlab, um conjunto de funções que implementem as funcionalidades de um *Bloom Filter* básico. As funções devem ter os parâmetros necessários para que seja possível criar *Bloom filters* de diferentes dimensões e usando números diferentes de funções de dispersão (k).

   Sugestão 1: Criar pelo menos 3 funções [1, sec. 3.2]: uma para inicializar a estrutura de dados; outra para inserir um elemento (ou elementos) no filtro; uma terceira para verificar se um elemento pertence ao conjunto.

   Sugestão 2: Deve procurar, seleccionar e implementar uma função de dispersão que tenha bom desempenho [1]

2) Teste as funções que desenvolveu na criação de um pequeno *Bloom Filter* para guardar uma lista de países. Insira alguns nomes de países no filtro e teste a pertença desses e de alguns países adicionais que não pertençam a essa lista inicial.

3) Para um teste mais exaustivo:

   (a) Gere m=1000 strings aleatórias com 40 caracteres (considere como caracteres possíveis o conjunto de caracteres minúsculos, maiúsculos e algarismos) e preencha um *Bloom Filter*, de tamanho n=8000. Este *Bloom Filter* deve ter $k = 3$ funções de dispersão e as strings geradas devem obdecer às diferentes probabilidades de ocorrência das letras em português.

   (b) Gere um segundo conjunto de 10000 strings aleatórias com 40 caracteres e teste a pertença das mesmas ao *Bloom Filter* que preencheu antes. Quantas destas strings foram consideradas como pertencendo ao conjunto com que o filtro foi preenchido? Estava à espera deste resultado?

4) Repita o teste da questão anterior para um número diferente de funções de dispersão ($k = 1, \ldots, 15$), obtendo o número de falsos positivos para cada k. Represente num gráfico os valores obtidos, em função de k e sobreponha nesse gráfico os valores teóricos (Assuma a independência de hash functions e que cada uma seleciona cada posição do bloom filter com igual probabilidade). Nota: Assume-se que as 10000 strings de teste são todas diferentes das 1000 inseridas no *Bloom filter*. No entanto pode haver strings iguais.

5) [opcional] Seleccione dois textos com um número grande de palavras de `gutemberg.org` e utilize um Bloom Filter para determinar as palavras do segundo que não existem no primeiro.

   Existe possibilidade de alguma das palavras que identificou como não existentes no primeiro texto fazerem parte desse texto?

6) Adapte as suas funções para implementar um *Count Filter*. Aplique estas novas funções para conseguir mostrar para uma qualquer palavra de um livro o número de vezes que ocorre no livro. Esta contagem apenas deve ser mostrada para palavras que pertençam ao livro.

   Sugere-se a utilização de um livro do projecto Gutemberg.

[1] Nota Importante: É obrigatório manter a informação original sobre autores e afins em todas as funções que utilizar e que não sejam da vossa autoria. Adaptações de código existente apenas podem ser feitas se as condições de utilização definidas pelos autores o permitirem, mantendo sempre informação sobre o autor original e adicionando informação sobre quem fez a alteração/evolução. Neste trabalho sugere-se criação de código original para todas as funções com a excepção das funções de dispersão.

Referências

[1] James Blustein and Amal El-Maazawi. Bloom filters - a tutorial, analysis, and survey. Tecnhical Report CS-2002-10, Dalhousie University, Dec 2002.

[2] Jure Leskovec, Anand Rajaraman, and Jeff Ullman. *Mining of Massive Datasets*, chapter Mining Data Streams. Cambridge University Press, 2014.