MPEI - Métodos Probabilísticos para Engenharia Informática (2018/2019) **PL 05**

Palavras-chave: aplicações de métodos probabilísticos, simulação, *hash table* , *hash function*, *chaining hash table*

O objectivo deste exercício é avaliar o funcionamento de uma estrutura de dados importante, a *Chaining Hash Table*, e um dos conceitos que a suportam, *Hash Functions* (funções de dispersão).

Sabe-se que "o desempenho da tabela de dispersão depende da capacidade da função de hash para distribuir uniformemente as chaves pelos índices do array" e que "uma análise estatística da distribuição das chaves pode ser considerada". Neste exercício, usando geração de chaves de forma aleatória iremos efectuar a análise da distribuição das chaves.

Começaremos por criar os componentes essenciais para efectuar uma primeira simulação bastante simplificada. De seguida tornaremos um pouco mais realista.

- Primeira versão:

  1. Crie uma função que gere uma chave (string) com comprimento aleatório entre 3 e 20 assumindo uma distribuição uniforme (discreta) e em que as letras (apenas minúsculas e maiúsculas) são equiprováveis.
  2. Implemente uma função de hash simples.Sugestão: Use uma das seguintes funções de dispersão:
     `string2hash()`[1],
     `hashstring()`[2],
     `DJB31MA()`[3],
     `hashcode()`[4],
     Nota: as 3 últimas funções são apresentadas noutras linguagens, sendo um bom exercício adaptá-las para o Matlab/octave e tentar perceber as relações entre elas. Tenha em atenção a necessidade de implementar estas funções em aritmética inteira.
  3. Simule a inserção de 1000 das strings criadas pela função que desenvolveu numa *Chaining Hash Table* (que como se deve recordar usa internamente um array e listas ligadas). Como para a nossa simulação não necessitamos guardar as chaves e valores a elas associados, apenas necessitamos do array. Adopte para tamanho do array um valor que implique um factor de carga elevado (ex: 0.8) .
     Guarde informação sobre o número de chaves que foram mapeadas numa determinada posição do array até esse momento.
     Durante a simulação, visualize, em gráficos separados:
     - o número de strings que foram mapeadas pela hash function para cada posição;
     - o histograma desses números.
  4. Usando a informação sobre o número de chaves que foram mapeadas para cada posição do array após a inserção de todas as chaves, estime e represente graficamente a função de distribuição para a variável aleatória X definida como o número de chaves mapeadas para uma posição. Qual seria o valor médio do comprimento das listas ligadas neste caso ? Qual a variância ?

---

[1] `https://www.mathworks.com/matlabcentral/fileexchange/27940-string2hash`
[2] Funçãao usada em Programação II que poderá ser adaptada ao matlab/octave.
[3] Função baseada no algorítmo de Daniel J. Bernstein, ver sumário de MPEI de 2014 (Prof. Paulo Jorge Ferreira) ou slides TP.
[4] Implementação do método hashcode, em JAVA, para objetos do da classe String

- Segunda versão:

    1. Crie uma nova versão da função de geração das chaves assumindo que o tamanho das chaves segue uma distribuição normal (com média 10 e variância 5) e as letras seguem a distribuição das letras em Português.

       Nota: Utilize o script disponibilizado no material de apoio à UC para efectuar a estimativa dessa distribuição.
    2. Repita a simulação com 1000 chaves e os cálculos da última alínea da versão 1.

- Terceira versão (opcional):

    1. Altere a função de hash na segunda versão e repita a simulação e cálculos.

- Questões finais:

  A função de hash conseguiu o objectivo de espalhar as chaves/strings pelo array ? Temos de facto uma distribuição uniforme das chaves pelos índices do array ?