PL 4

Algoritmos Probabilísticos

4.1 Funções de dispersão

O objectivo destes exercícios 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).

1. Crie uma função para gerar chaves constituídas por caracteres. O comprimento (i.e., o número de caracteres) de cada chave deve ser escolhido aleatoriamente (distribuição uniforme) entre i_{min} e i_{max}. A função deve ter como parâmetros de entrada o número N de chaves a gerar, os valores de i_{min} e i_{max}, um vector com os caracteres a usar nas chaves e um vector com as probabilidades de cada um dos caracteres a utilizar. Se a função for chamada sem o último parâmetro, deve considerar os caracteres equiprováveis (ver a documentação da função `nargin`).

   A função deve devolver um "cell array"com o conjunto de chaves geradas garantindo que as chaves são todas diferentes.

   (a) Gere um conjunto de $N = 10^5$ chaves usando todas as letras maiúsculas e minúsculas ('A' a 'Z' e 'a' a 'z') com igual probabilidade e em que $i_{min} = 6$ e $i_{max} = 20$.

   (b) Gere um conjunto de $N = 10^5$ chaves usando todas as letras minúsculas ('a' a 'z') com as probabilidades contidas no ficheiro `prob_pt.txt` e que correspondem às frequências das letras em Português (`https://pt.wikipedia.org/wiki/Frequ%C3%AAncia_de_letras`). Considere novamente $i_{min} = 6$ e $i_{max} = 20$.

2. Considere a função Matlab `string2hash()` [1] que implementa duas funções de dispersão diferentes. Considere ainda 2 funções Matlab fornecidas que são adaptações para Matlab das funções de dispersão `hashstring()` [2] e `DJB31MA()` [3].

   Utilizando separadamente cada uma destas quatro funções de dispersão, simule a inserção das chaves criadas no exercício 1a) em 3 *Chaining Hash Tables*, uma de tamanho 5×10^5, outra de tamanho 10^6 e a terceira de tamanho 2×10^6. Para cada uma das simulações (4 funções de dispersão $\times$ 3 tamanhos):

   (a) Guarde um vetor com os *hashcodes* obtidos.

   (b) Registe o número de atribuições a cada uma das posições de cada *Hash Table*.

   (c) Calcule o número de colisões (em cada *Hash Table* e para cada função de dispersão).

   (d) O tempo de execução da simulação.

3. Utilizando a informação obtida no exercício anterior, compare o desempenho das quatro funções de dispersão para cada tamanho diferente da *Hash Table*, relativamente a:

[1] `https://www.mathworks.com/matlabcentral/fileexchange/27940-string2hash`

[2] Função usada em Programação II que poderá ser adaptada ao matlab.

[3] Função baseada no algorítmo de Daniel J. Bernstein, ver sumário de MPEI de 2014 (Prof. Paulo Jorge Ferreira) ou slides TP.

(a) Uniformidade, de duas formas diferentes:

i. visualize os histogramas dos *hascodes* com 100 intervalos e verifique se os valores nos diferentes intervalos são similares;

ii. calcule os momentos de ordem 2, 5 e 10 das variáveis aleatórias correspondentes aos valores dos *hashcodes* divididos pelo comprimento da *Hash Table* (i.e, variável aleatória toma valores entre 0 e 1) e compare com os valores teóricos da distribuição uniforme. [4]

(b) Número de colisões e número máximo de atribuições numa mesma posição da *Hash Table*.

(c) Tempos de execução.

4. Repita os exercícios 2 e 3 usando agora as chaves criadas no exercício 1b). As conclusões destes resultados são semelhantes às do exercício 3?

A ser completado ...

[4] Na distribuição uniforme entre 0 e 1, o valor do momento de ordem n é igual a $\frac{1}{n+1}$.