Aula prática 10

Esta aula tem como objetivo estudar tabelas de dispersão (*hash tables*), e analisar a performance de algumas operações com as mesmas.

1 Pretende-se analisar o comportamento de tabelas de dispersão em comparação com listas ligadas. Para tal deverá fazer uso da biblioteca fornecida em **tabdispersao.c/.h**. Comece por analisar os ficheiros da biblioteca antes de procurar resolver as alíneas seguintes.

A tabela de dispersão implementada baseia-se no endereçamento fechado (encadeamento). Adicionalmente, cada elemento é composto pelo par chave/valor. Note que apenas a chave define a posição do elemento.

1.1 Crie uma função que leia o ficheiro de texto **englishwords.txt** e vá inserindo todas as linhas do ficheiro de dados numa lista ligada. De seguida, crie uma função semelhante, mas que faça o mesmo para tabelas de dispersão (utilize a função de dispersão hash_djbm) com chave e valor iguais à palavra. Que conclusões tira quanto à performance de ambas as estruturas de dados?

Nota: Para cronometrar ambas as funções pode usar a biblioteca time (#include <time.h>) para obter a hora atual. Use o exemplo seguinte como referência:

```
clock_t inicio, fim;
double tempo;

inicio = clock();

/* tarefa a verificar */

fim = clock();
tempo = (double)(fim - inicio) / CLOCKS_PER_SEC;
printf("tempo em segundos: %lf\n", tempo);
```

1.2 Deverá agora implementar uma função que procure um elemento identificado pela chave na tabela de dispersão e devolva o valor associado.

const char* *tabela_valor*(**tabela_dispersao** *td, **const char** *chave)

Corra um teste semelhante ao da alínea anterior em que procure *n* elementos gerados aleatoriamente da lista e cronometre o tempo que essa operação demorou. Que conclusões tira dos resultados obtidos?

1.3 Implemente a função de hash_krm. Esta função calcula a *hash* com base na seguinte fórmula:

$$hash(i) = hash(i\text{-}1) + chave[i], \text{ sendo } hash(0) = 7$$

A função deve receber uma *string* para a qual se pretende calcular a *hash* e o tamanho da tabela de dispersão. Nota: chave[i] refere-se ao carácter na posição i da chave.

Experimente correr novamente os testes do primeiro exercício usando a função de hash_krm em vez da hash_djbm. Que diferenças verifica na inserção/pesquisa de elementos na tabela?

2 Pretende-se implementar um programa que verifique o *login* dos utilizadores, utilizando para esse efeito uma tabela de dispersão para armazenar os pares *login / password*. Utilize a implementação da tabela de dispersão disponibilizada na alínea anterior.

O programa deverá inicialmente ler e guardar os dados de *login* contidos no ficheiro **passwords.txt**. Posteriormente, deverá perguntar ao utilizar os dados e imprimir a validação dos *logins*, isto é, se cada par *login / password* está de acordo com os registos contidos do ficheiro.

Exemplo

```
Login: maria
Password: contrary
Authentication succeeded
Login: tiago
Password: contrary
Authentication failed
Login: ricardo
Password: sheeplost
Authentication failed
```

3 Suponha que pretende alterar a função de *hash* de uma tabela de dispersão. Que consequências tem essa alteração numa tabela já preenchida com alguns elementos? Implemente a seguinte função:

int *tabela_alterahash*(**tabela_dispersao** *td, **hash_func** *hfunc)

O primeiro argumento da função é a tabela de dispersão e o segundo é um apontador para a nova função de *hash*. A função deve retornar 0 se for bem sucedida e -1 caso contrário.