Relatório do trabalho de Estrutura de Dados e Algoritmos 1

João Santos nº 51966 | Diogo Matos nº 54466 | Pedro Gomes nº 54554

21 de junho de 2023

Introdução

Neste trabalho pretendia-se desenvolver um programa utilizando Estruturas de Dados lecionadas em linguagem C para obter todas as soluções possíveis, ou seja, todas as palavras possíveis de se formar num tabuleiro de Boggle (este jogo é um género de "sopa de letras"). Dado um ficheiro com o boggle (tabuleiro) o programa deve resolvê-lo recorrendo a um outro ficheiro, sendo este um dicionário com as palavras em inglês.

Para a resolução deste trabalho foi utilizada a implementação das tabelas de hash lineares lecionadas nas aulas, alterando apenas a função Hash() para aceitar strings.

Instruções de execução

Para executar o código, você deve inserir o seguinte comando no terminal: "*gcc hash-linear.c main.c -o teste && ./teste*".

Exemplo de Output (boggle0.txt)

```
MOUSE  M:(2,1)->O:(1,0)->U:(1,1)->S:(0,0)->E:(0,1)
MOULD  M:(2,1)->O:(1,0)->U:(1,1)->L:(0,2)->D:(0,3)
MOO  M:(2,1)->O:(1,0)->O:(2,0)
MOON  M:(2,1)->O:(1,0)->O:(2,0)->N:(3,1)
MU  M:(2,1)->U:(1,1)
MUSE  M:(2,1)->U:(1,1)->S:(0,0)->E:(0,1)
MULE  M:(2,1)->U:(1,1)->L:(0,2)->E:(0,1)
MULES  M:(2,1)->U:(1,1)->L:(0,2)->E:(0,1)->S:(0,0)
MUM  M:(2,1)->U:(1,1)->M:(1,2)
MOO  M:(2,1)->O:(2,0)->O:(1,0)
MOOS  M:(2,1)->O:(2,0)->O:(1,0)->S:(0,0)
MOOSE  M:(2,1)->O:(2,0)->O:(1,0)->S:(0,0)->E:(0,1)
MOUSE  M:(2,1)->O:(2,0)->U:(1,1)->S:(0,0)->E:(0,1)
MOULD  M:(2,1)->O:(2,0)->U:(1,1)->L:(0,2)->D:(0,3)
MONEY  M:(2,1)->O:(2,0)->N:(3,1)->E:(2,2)->Y:(3,3)
MONK  M:(2,1)->O:(2,0)->N:(3,1)->K:(3,2)
MONKEY  M:(2,1)->O:(2,0)->N:(3,1)->K:(3,2)->E:(2,2)->Y:(3,3)
ME  M:(2,1)->E:(2,2)
MEMO  M:(2,1)->E:(2,2)->M:(1,2)->O:(1,3)
MET  M:(2,1)->E:(2,2)->T:(2,3)
MEN  M:(2,1)->E:(2,2)->N:(3,1)
MINE  M:(2,1)->I:(3,0)->N:(3,1)->E:(2,2)
MINK  M:(2,1)->I:(3,0)->N:(3,1)->K:(3,2)
MINKE  M:(2,1)->I:(3,0)->N:(3,1)->K:(3,2)->E:(2,2)
EM  E:(2,2)->M:(1,2)
EMU  E:(2,2)->M:(1,2)->U:(1,1)
EMUS  E:(2,2)->M:(1,2)->U:(1,1)->S:(0,0)
EM  E:(2,2)->M:(2,1)
EMU  E:(2,2)->M:(2,1)->U:(1,1)
EMUS  E:(2,2)->M:(2,1)->U:(1,1)->S:(0,0)
TO  T:(2,3)->O:(1,3)
TOLD  T:(2,3)->O:(1,3)->L:(0,2)->D:(0,3)
TOME  T:(2,3)->O:(1,3)->M:(1,2)->E:(0,1)
TOMES  T:(2,3)->O:(1,3)->M:(1,2)->E:(0,1)->S:(0,0)
TOME  T:(2,3)->O:(1,3)->M:(1,2)->E:(2,2)
TOE  T:(2,3)->O:(1,3)->E:(2,2)
TEN  T:(2,3)->E:(2,2)->N:(3,1)
TYKE  T:(2,3)->Y:(3,3)->K:(3,2)->E:(2,2)
ION  I:(3,0)->O:(2,0)->N:(3,1)
IM  I:(3,0)->M:(2,1)
IN  I:(3,0)->N:(3,1)
INK  I:(3,0)->N:(3,1)->K:(3,2)
INKY  I:(3,0)->N:(3,1)->K:(3,2)->Y:(3,3)
NO  N:(3,1)->O:(2,0)
NOOSE  N:(3,1)->O:(2,0)->O:(1,0)->S:(0,0)->E:(0,1)
NE  N:(3,1)->E:(2,2)
NET  N:(3,1)->E:(2,2)->T:(2,3)
KEN  K:(3,2)->E:(2,2)->N:(3,1)
KEY  K:(3,2)->E:(2,2)->Y:(3,3)
YE  Y:(3,3)->E:(2,2)
YET  Y:(3,3)->E:(2,2)->T:(2,3)
YEN  Y:(3,3)->E:(2,2)->N:(3,1)

Número total de palavras encontradas: 116
```

Figura 1: Output do boggle0.txt.

loadBoggleBoard()

Descrição:

A função **loadBoggleBoard** é responsável por carregar o tabuleiro do jogo Boggle a partir de um ficheiro (boggle0.txt, boggle1.txt, boggle2.txt).

Dentro da função, o ficheiro é aberto no modo de leitura onde é usado a função *fopen*. Se ocorrer uma falha ao abrir o ficheiro, uma mensagem de erro é exibida e o programa é encerrado.

Em seguida, há um loop que percorre as linhas e as colunas do tabuleiro do Boggle. A cada iteração, um caractere do ficheiro é lido e guardado na posição correspondente da matriz. Além disso, a posição visitada é inicializada como *false*, que indica a posição da matriz como não visitada.

Após percorrer todo o tabuleiro, o ficheiro é fechado através da função *fclose*.

loadDictionary()

Descrição:

A função **loadDictionary** abre o "corncob_caps_2023.txt" no modo de leitura.

Em seguida, lê cada palavra do ficheiro e a armazena temporariamente em uma variável chamada *word*. Cada palavra lida é copiada para uma variável de leitura temporária chamada *read_word* e inserida em uma tabela *hash* de palavras completas *(wordTable)*.

Além disso, depois de armazenar uma palavra, a função gera e insere os prefixos de cada palavra na tabela hash de prefixos (*prefixTable*), isto que ocorre com um loop.

Após a leitura de todas as palavras, e de inseridos todos os prefixos de cada palavra, o ficheiro é fechado.

findWords

Descrição:

A função **findWords** é responsável por encontrar as palavras no tabuleiro do Boggle.

A função recebe o tabuleiro (*board*), as tabelas hash de palavras completas (*wordTable*) e de prefixos (*prefixTable*), além das coordenadas da posição atual (*row e col*). A função também recebe uma estrutura de resultado de palavras (*wordResult*) e um contador de palavras encontradas (*count_word*).

A função verifica se a posição atual está dentro dos limites do tabuleiro e se já foi visitada. Se não atender a essas condições, a função retorna. Caso contrário, a posição é marcada como visitada e a letra correspondente é adicionada à palavra em construção.

Em seguida, é verificado se a palavra em construção existe nas tabelas hash onde é utilizada a função **Find**. Se a palavra for encontrada em ambas as tabelas de hash, é incrementado o contador de palavras encontradas e a palavra é impressa seguida do seu caminho, ou seja, é impresso cada letra da palavra com a sua posição.

Se a palavra em construção for um prefixo válido, a função é chamada recursivamente para as posições vizinhas, onde explora todas as direções adjacentes.

Ao finalizar o processamento de uma posição, a marcação de visitado é desfeita e o tamanho da palavra é decrementado.

solveBoggle()

Descrição:

A função **solveBoggle** carrega o tabuleiro (boggle0.txt, boggle1.txt, boggle2.txt).

Inicia as tabelas hash para armazenar palavras e prefixos do dicionário, carrega o dicionário para as tabelas hash e inicia uma estrutura para armazenar as palavras resultantes.

Em seguida, ela percorre cada posição do tabuleiro e verifica as palavras existentes a partir de cada posição, onde conta o número total de palavras encontradas.

Por fim, a função imprime o número total de palavras encontradas e liberta a memória das tabelas hash.

main()

Descrição:

A função principal (**main**) especifica o nome dos ficheiros de entrada (estas que têm de ser trocadas consoante a localização dos ficheiros) contendo o tabuleiro Boggle e o dicionário em inglês. Em seguida, chama a função **solveBoggle** para resolver o Boggle onde usa os ficheiros fornecidos.

Conclusão:

Com este trabalho evoluímos o nosso conhecimento e capacidade em implementar com Estruturas de Dados em C. Para além disso, houve certas funções que não sabíamos que existiam, e que podémos aprender (e.g. "strcmp", "strncpy").