Unidade Curricular de Estrutura de Dados e Algoritmos I da Licenciatura em Engenharia Informática

UNIVERSIDADE DE ÉVORA

Jogo do Boggle

Discentes:

Bárbara Loureiro, nº48469

Diogo Miranda, nº 55222

Catarina Andrade, nº 55406

junho 2023

Docente:

Lígia Ferreira

Introdução

Boggle é um popular jogo de palavras, no qual os jogadores tentam encontrar palavras numa grelha de letras. Tradicionalmente, Boggle é jogado com um tabuleiro físico que contém 16 dados com letras, mas no código em análise, o tabuleiro é simulado por um programa de computador.

O objetivo deste projeto é implementar uma versão do jogo Boggle, permitindo que o tabuleiro seja definido através de um ficheiro e usando um dicionário de palavras também carregado de um ficheiro, para validar as palavras encontradas no tabuleiro.

Neste relatório, faremos uma análise detalhada da implementação do código, focando nas estruturas de dados usadas, no funcionamento do algoritmo e em como os requisitos foram atendidos. Também abordaremos possíveis melhorias e otimizações que poderiam ser aplicadas ao código.

Estruturas de Dados Utilizadas

O programa utiliza principalmente uma tabela de dispersão (hash table) para armazenar as palavras do dicionário e uma matriz 4x4 para representar o tabuleiro do Boggle. A tabela de dispersão é composta por entradas contendo palavras e um ponteiro para a próxima entrada (implementação de encadeamento para resolver colisões).

Tabela de Dispersão

A tabela de dispersão é definida através da estrutura `HashTable`, que contém um array de ponteiros para `Entry` e um contador de palavras inseridas. A estrutura `Entry` representa uma entrada na tabela de dispersão, que contém a palavra e um ponteiro para a próxima entrada.

A tabela de dispersão é utilizada para armazenar as palavras do dicionário e oferece um tempo de pesquisa eficiente. A função de dispersão utilizada (`hash`) é baseada em operações de bitshift e adição.

Matriz do Tabuleiro Boggle

O tabuleiro do Boggle é representado como uma matriz 4x4 de caracteres, onde cada elemento representa uma letra no tabuleiro.

Funcionamento do Programa

Carregamento de Dados

1. Carregamento do Dicionário: O dicionário é carregado do ficheiro especificado (corncob_caps_2023.txt) através da função loadDictionary. Esta função lê cada palavra do ficheiro e a insere na tabela de dispersão.

2. Carregamento do Tabuleiro Boggle: O tabuleiro é carregado do ficheiro especificado (boggle0.txt) pela função loadBoggle. Esta função lê cada caractere e preenche a matriz 4x4 que representa o tabuleiro.

Busca de Palavras no Tabuleiro

1. A função findWords é responsável por iniciar a busca de palavras no tabuleiro. Esta itera sobre cada célula do tabuleiro e chama a função recursiva findWordsUtil.

2. findWordsUtil é uma função recursiva que explora todas as direções possíveis a partir de uma célula específica para formar palavras. Utiliza uma matriz visited para manter o controle das células já visitadas.

3. Se uma sequência de caracteres forma uma palavra válida (presente no dicionário e com mais de 2 letras), a palavra e as suas coordenadas são impressas no output.

Resultado

Finalmente, o programa imprime o número total de palavras encontradas no tabuleiro.

Output

```
lidas 58109 palavras do dicionário
Encontrados 58062 prefixos distintos
******************
*   B O G G L E  *
******************
*     0  1  2  3 *
* 0   S  E  L  D  *
* 1   O  U  M  O  *
* 2   O  M  E  T  *
* 3   I  N  K  Y  *
******************
  SELDOM S:(0,0)->E:(0,1)->L:(0,2)->D:(0,3)->O:(1,3)->M:(1,2)
   SEOUL S:(0,0)->E:(0,1)->O:(1,0)->U:(1,1)->L:(0,2)
   SEMEN S:(0,0)->E:(0,1)->M:(1,2)->E:(2,2)->N:(3,1)
    SOUL S:(0,0)->O:(1,0)->U:(1,1)->L:(0,2)
    SOON S:(0,0)->O:(1,0)->O:(2,0)->N:(3,1)
    SOME S:(0,0)->O:(1,0)->M:(2,1)->E:(2,2)
     SUE S:(0,0)->U:(1,1)->E:(0,1)
     SUM S:(0,0)->U:(1,1)->M:(1,2)
    SUMO S:(0,0)->U:(1,1)->M:(1,2)->O:(1,3)
  SUMMON S:(0,0)->U:(1,1)->M:(1,2)->M:(2,1)->O:(2,0)->N:(3,1)
     SUM S:(0,0)->U:(1,1)->M:(2,1)
    SUMO S:(0,0)->U:(1,1)->M:(2,1)->O:(1,0)
    SUMO S:(0,0)->U:(1,1)->M:(2,1)->O:(2,0)
     SUE S:(0,0)->U:(1,1)->E:(2,2)
    SUET S:(0,0)->U:(1,1)->E:(2,2)->T:(2,3)
     ELM E:(0,1)->L:(0,2)->M:(1,2)
     EMU E:(0,1)->M:(1,2)->U:(1,1)
    EMUS E:(0,1)->M:(1,2)->U:(1,1)->S:(0,0)
   LUMEN L:(0,2)->U:(1,1)->M:(1,2)->E:(2,2)->N:(3,1)
   LUMEN L:(0,2)->U:(1,1)->M:(2,1)->E:(2,2)->N:(3,1)
     LOT L:(0,2)->O:(1,3)->T:(2,3)
    DOLE D:(0,3)->O:(1,3)->L:(0,2)->E:(0,1)
   DOLES D:(0,3)->O:(1,3)->L:(0,2)->E:(0,1)->S:(0,0)
  DOLMEN D:(0,3)->O:(1,3)->L:(0,2)->M:(1,2)->E:(2,2)->N:(3,1)
    DOME D:(0,3)->O:(1,3)->M:(1,2)->E:(0,1)
   DOMES D:(0,3)->O:(1,3)->M:(1,2)->E:(0,1)->S:(0,0)
    DOME D:(0,3)->O:(1,3)->M:(1,2)->E:(2,2)
     DOE D:(0,3)->O:(1,3)->E:(2,2)
     DOT D:(0,3)->O:(1,3)->T:(2,3)
    DOTE D:(0,3)->O:(1,3)->T:(2,3)->E:(2,2)
    OMEN O:(1,0)->M:(2,1)->E:(2,2)->N:(3,1)
 OMINOUS O:(1,0)->M:(2,1)->I:(3,0)->N:(3,1)->O:(2,0)->U:(1,1)->S:(0,0)
     USE U:(1,1)->S:(0,0)->E:(0,1)
    MELD M:(1,2)->E:(0,1)->L:(0,2)->D:(0,3)
    MUSE M:(1,2)->U:(1,1)->S:(0,0)->E:(0,1)
    MULE M:(1,2)->U:(1,1)->L:(0,2)->E:(0,1)
   MULES M:(1,2)->U:(1,1)->L:(0,2)->E:(0,1)->S:(0,0)
     MUM M:(1,2)->U:(1,1)->M:(2,1)
    MOLE M:(1,2)->O:(1,3)->L:(0,2)->E:(0,1)
   MOLES M:(1,2)->O:(1,3)->L:(0,2)->E:(0,1)->S:(0,0)
    MOLD M:(1,2)->O:(1,3)->L:(0,2)->D:(0,3)
     MOD M:(1,2)->O:(1,3)->D:(0,3)
```

```
    MEMO M:(1,2)->E:(2,2)->M:(2,1)->O:(2,0)
     MET M:(1,2)->E:(2,2)->T:(2,3)
     MEN M:(1,2)->E:(2,2)->N:(3,1)
     OLE O:(1,3)->L:(0,2)->E:(0,1)
     OLD O:(1,3)->L:(0,2)->D:(0,3)
     OLM O:(1,3)->L:(0,2)->M:(1,2)
    OMEN O:(1,3)->M:(1,2)->E:(2,2)->N:(3,1)
    OMEN O:(2,0)->M:(2,1)->E:(2,2)->N:(3,1)
    OINK O:(2,0)->I:(3,0)->N:(3,1)->K:(3,2)
     ONE O:(2,0)->N:(3,1)->E:(2,2)
   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)
    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)
    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)
     EMU E:(2,2)->M:(1,2)->U:(1,1)
    EMUS E:(2,2)->M:(1,2)->U:(1,1)->S:(0,0)
     EMU E:(2,2)->M:(2,1)->U:(1,1)
    EMUS E:(2,2)->M:(2,1)->U:(1,1)->S:(0,0)
    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)
     INK I:(3,0)->N:(3,1)->K:(3,2)
    INKY I:(3,0)->N:(3,1)->K:(3,2)->Y:(3,3)
   NOOSE N:(3,1)->O:(2,0)->O:(1,0)->S:(0,0)->E:(0,1)
     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)
     YET Y:(3,3)->E:(2,2)->T:(2,3)
     YEN Y:(3,3)->E:(2,2)->N:(3,1)
Foram encontradas 95 soluções
```

Conclusão

O código implementa com sucesso o jogo Boggle utilizando estruturas de dados fundamentais, como tabelas de dispersão e matrizes. A tabela de dispersão permite pesquisas eficientes, o que é crucial para verificar se uma sequência de caracteres está presente no dicionário. A abordagem de busca é baseada em backtracking, explorando recursivamente todas as possíveis combinações de caracteres no tabuleiro.

Embora funcional, o código pode ser otimizado e melhorado em diversos aspetos, como melhorar a implementação da tabela de prefixos.