UNIVERSIDADE DE ÉVORA

Escola de Ciências e Tecnologias

Engenharia Informática

Estruturas de Dados e Algoritmos

2019/2020

- Jogo do Boggle -

Docentes: Lígia Maria Rodrigues da Silva Ferreira

Discentes: Sarah Simon Luz – 38116

Ana Ferro – 39872

Descrição do Trabalho

Neste trabalho pretende-se criar um programa que seja capaz de produzir soluções para um jogo de Boggle. No Boggle há uma grelha de 4x4 letras, sendo o objetivo do jogo encontrar o maior número de palavras possível e quanto mais letras tiverem as palavras encontradas maior a pontuação. As palavras são formadas começando por uma qualquer letra da grelha e seguidamente uma qualquer letra adjacente (posições adjacentes são N, S, W, E, NE, SW e NW). A única restrição é que não é possível repetir a mesma letra (posição), na mesma palavra.

Classe Boogle

- static void **inserirDic**(HashTable<String> dicionario)

  Lê do ficheiro "allWords.txt" todas as palavras e insere-as começando com a primeira letra até formar a palavra completa, associando uma flag que indica que de facto é uma palavra do dicionário.

- static char[][] **readBoggle**(String input)

  Lê um ficheiro de texto, com uma única linha e 16 caracteres e posiciona-os numa matriz 4x4 de chars, retorna a matriz.

- static boolean **possible**(char[][] boggle, int line, int column, Position word)

  Verifica se é possível seguir para a posição adjacente (int line, int column) sem ultrapassar os limites do tabuleiro e se essa posição adjacente já foi usada previamente.

- static LinkedList<String> **searchWord**(Position word, char[][] boggle, HashTable<String> dicionario, LinkedList<String> words)

  Mecanismo principal. É uma função recursiva que no caso base é verificado se a palavra formada pertence ao dicionário.

  Se o caso base falhar, verifica se a flag associada à palavra é verdadeira para poder adicionar à linked list de retorno. Irá iterar o array duplo de inteiros, chamado adjacentes, somando às coordenadas do último elemento da word para obter todos os "vizinhos". Em cada elemento chama a função possible para verificar se é um caminho viável. Caso seja, esse elemento é adicionado à word e a função searchWord é chamada. Quando voltar é removido o vizinho adicionado, para poder verificar os seguintes.

  No final é retornado a linked list com os caminhos das palavras encontradas.

Classe Position

- public **Position**()
  Declara a linked list letters.

- public void **add**(char letter, int line, int column)
  Adiciona à linked list letters uma array list com a letra (código ascii), a linha e a coluna.

- public void **remove**()
  Remove a última letra da palavra formada por todas as letras contidas na linked list e remove o último elemento da mesma.

- public String **toString**()
  Vai iterar a linked list e para cada elemento vai convertê-lo para o formato de output desejado.

- public char **toChar**(int 1)
  Converte para char e retorna-o.

- public String **getWord**()
  Retorna uma string formada pelas letras já existentes nessa instância Position.

- public ArrayList **format**(char letter, int line, int column)
  Formata os dados na maneira desejada (converte o char para código ascii, linha, coluna) e adiciona na array list. Retorna a array list.

- public boolean **contains**(char letter, int line, int column)
  Verifica se essa posição/letra está contida dentro da linked list.

- public ArrayList **getLast**()
  Retorna o último elemento da linked list.