Trabalho:
Jogo do Boggle

Objetivos

Utilização da linguagem Java e dos tipos abstratos de dados lecionados e implementados pelos alunos, para resolver um problema do tipo ”Sopa de Letras”. O aluno deve providenciar juntamente com os ficheiros pedidos a implementação de todas as estruturas de dados que usar na resolução do trabalho.

Descrição

O programa é capaz de produzir soluções para um jogo de Boggle. Este jogo aparece em alguns jornais americanos e rivaliza com as habituais palavras cruzadas ou as mais antigas sopas de letras. No Boggle há uma grelha de 4x4 letras, sendo o objetivo do jogo encontrar o maior nu´mero 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. A única restrição e que não é possível repetir a mesma letra(posição), na mesma palavra.

Trabalho realizado:
João Pedro Bilo Cavaco
l42470@alunos.uevora.pt

Implementação

Listas: Escolhi usar a estrutura ArrayList que implementei durante as aulas práticas. Embora existam algumas vantagens na utilização de uma LinkedList relativamente à complexidade temporal, após ter testado o programa com ambas as estruturas cheguei à conclusão que o tempo de execução é praticamente igual usando qualquer uma das estruturas.

Hashtable: De acordo com o enunciado devemos implementar uma HashTable com um método de hashing fechado de acesso à nossa escolha. Decidi usar parte de uma HashTable que implementei durante as aulas práticas que utiliza o método de acesso Double Hashing.

Position: Classe que agrupa no mesmo objeto o conceito de linha e coluna.
Possui 2 atributos(linha e coluna), o respetivo construtor, getters e uma função **equals()** para poder comparar duas instâncias da classe Position diferentes.

Node: Classe que representa uma "posição completa" do Boggle. Possui um atributo Position que indica a sua posição no tabuleiro do Boggle, um atributo letter que define a letra que está contida nessa posição e uma lista de Posições vizinhas. Possui um Construtor e os respectivos getters

Board: Classe que representa o tabuleiro do jogo Boggle e que é formado por uma lista de Nodes. Possui apenas um atributo (a lista de Nodes), um getter (para a lista de Nodes) e um Construtor

Class Boggle

Atributos:

- **Board:** tabuleiro do jogo boggle
- **HashTable:** tabela com todas as palavras presentes no ficheiro *all_words.txt*

Métodos:

- **createUI()**: método que utiliza os componentes disponibilizados no pacote *java.swing* para criar uma pequena janela em que é disponibilizado um *JTextField* onde o utilizador pode introduzir uma String que levará a criação do Boggle Board e um *JTextArea* onde serão listados todos os resultados encontrados pelo método *solve().*
- **Boggle():** construtor secundário que invoca o construtor primário e o método *solve()* e mostra a lista de resultados na *JTextArea*.
- **Boggle(*String*):** construtor que invoca o método *importDictionary()* e que inicializa e configura o atributo *board* através do método *createBoard()*.
- **createBoard(*String*):** método que utiliza a String passada como argumento e inicializa o atributo board com as Nodes que foram criadas por cada *Character* da *String*. O tamanho do Board criado irá ser a raiz quadrada do comprimento da String passada como argumento(Por exemplo se a String passada for **seldoumoometinky***(16 caracteres)* então o board é de 4x4).
- **createNode(Character,Position,Integer):** método que retorna uma nova Node composta por o Character, Position e Lista de vizinhos criada pelo método **getNeighbours()** passados como argumentos**.**
- **getNeighbours(Position,Integer):** método que retorna uma lista de Nodes que contém todos os vizinhos da Node na Position passado como argumento, se esta existir e estiver dentro do tabuleiro.
- **isValidPosition(Position,Integer):** método que retorna true se a posição passada como argumento se encontra dentro de um tabuleiro de dimensão passada pelo argumento Integer .
- **ImportDictionary():** método que lê linha a linha o ficheiro all_words.txt e põe todas as ocorrências na HashTable tabela.

- **PositionToNode(Position, Board):** método que retorna uma Node, se esta existir no Board, baseada na sua Position
- **removeDuplicates(List):** método que recebe um MyArrayList e remove todas as entradas duplicadas.
- **solve():** método que retorna uma String que contém o resultado final do Jogo do Boggle atual. Este método começa por criar uma Lista vazia que irá guardar todas as palavras encontradas no board e que estejam contidas na tabela, depois itera por todas as Nodes existentes no board e em cada iteração utiliza uma função recursiva **backtracking()** para obter todas as palavras presentes no dicionário que possam ser obtidas a partir da Node em iteração.
- **backtracking(String,Node,Board,List):** método recursivo auxiliar ao método **solve()** que começa por verificar se a String passada como argumento está contida no dicionario(se estiver é adicionada à lista de resultados passada como argumento) depois, cria uma lista e adiciona-lhe todos as Nodes vizinhas da Node passada como argumento. Depois vai iterar sobre a lista de vizinhas e se uma vizinha já tiver sido visitada irá ser ignorada(pois no jogo do Boggle não se podem obter palavras que passem 2 vezes pela mesma letra), se não tiver sido visitada então esta é adicionada à lista de posições visitadas e o método chama-se a ele mesmo passando como argumentos a concatenação da palavra que foi passada como argumento e a letra da vizinha que está a ser iterada, a Node vizinha que está a ser iterada, o tabuleiro de jogo, a lista com as Positions já visitadas e a Lista de resultados.