Universidade de Évora

Estruturas de Dados e Algoritmos 1

Diogo Rafael nº 37859 —— Miguel Azevedo nº 36975

9 de Janeiro

Professora Lígia Ferreira

Introdução

O trabalho consiste num algoritmo de resolução de jogos Boggle, com o auxilio do dicionário da língua inglesa, para que possamos encontrar todas as palavras possíveis na matriz.

Para realizar este trabalho foram utilizadas as implementações realizadas ao longo do semestre, tais como as de **Hash Tables** com elementos **Element** e a de **Linked Lists** com elementos **SingleNode** .

Para a resolução do problema, começámos por criar uma Tabela de Hash com palavras e outra com prefixos. De seguida foi criada a matriz Boggle, esta consiste numa matriz 4x4 de elementos **Position** a partir de ficheiros dados, com uma moldura de elementos Null em seu redor. A moldura foi criada para facilitar a pesquisa na matriz. Prosseguimos a ver todas as palavras possiveis a partir de [X][Y] letra através de um método recursivo na classe **Boggle**.

Implementações de Estruturas

Hash Table

Esta implementação foi feita nas aulas, contém todos os métodos pedidos no enunciado da Prática #9. Para tratar das posições(**procPos(T s)**) utilizámos o método **hashcode()** do Java e em caso de colisão é feito tratamento linear de colisões.

Linked List

Também feita nas aulas de acordo com o enunciado da Prática #5 embora que contenha mais dois métodos extra **copy(LinkedList<T>newlist)** e **outPutCorreto(LinkedList<Position>ll)**. Têm como objetivo respectivamente copiar os contéudos de uma ***LinkedList*** para outra ***LinkedList*** vazia e retornar uma String com o objetivo de ***printar*** a solução no formato pedido no enunciado do trabalho.

Utilizámos *LinkedLists* pois considerámos a melhor opção para tratar e ***printar*** as respostas obtidas e também como fazer a procura.

Classe Boggle

Esta classe contém a matriz com o jogo, o construtor trata-se da chamada do método **lerBoggle(String filename)**. Este irá ler o ficheiro dado no input e vai proceder à criação de uma matriz Boggle com NULLS à sua volta.

Contém ainda dois métodos **private void solve(Hash palavra, Hash prefixo, LinkedList ll, LinkedList[] sol, int llsize, int i, int j)** (recursivo) e **solve(Hash palavra, Hash prefixo)** que chama o primeiro.

O primeiro método consiste numa pesquisa recursiva na matriz com várias verificações (ser **Null** ou já ter sido visitado, ser palavra mas não prefixo, ser prefixo mas não palavra, ser prefixo e palavra ou não ser nem um nem outro), deixando para trás marcas de visitado na **Position** da matriz em que é chamado para evitar que fique preso na recursão. A palavra irá sendo formada através de uma **LinkedList** e caso a palavra formada pertença ao dicionário irá ser adicionada a um **array** soluções.

O segundo método **LinkedList[] sol** irá retornar todas as soluções do Boggle.

Classe Main

Esta classe, em termos de métodos, apenas contém o método **dicionario(HashTable prefixos, HashTable palavras, String filename)** em que irá criar todos os prefixos e palavras do dicionario dado num ficheiro filename e adicionar às respectivas **Tabelas de Hash**.

De seguida, foi criado um Boggle dado um ficheiro e onde foi chamado o **método solve** com todas as soluções do mesmo.

Conclusão

Considerámos este trabalho uma boa forma de por em prática a utilização de Estruturas de Dados que foram desenvolvidas ao longo do semestre. Apesar de alguns problemas que fomos tido ao longo do trabalho, estes ajudaram ainda mais a melhorar e perceber tanto as implementações dadas como também o seu funcionamento em Java.

Em geral, através deste trabalho conseguimos obter conhecimentos e utilizar os que adquirimos nas aulas de forma a chegar ao produto final pretendido.