Estruturas de Dados e Algoritmos I

Trabalho 2 - Boogle

Gabriel Charrua 32457

2015/2016

Introdução

No âmbito da disciplina de Estruturas de Dados e Algoritmos I, foi-nos proposto a criação de um jogo - Boogle com o auxilio dos tipos abstratos de dados e a utilização da linguagem Java.

Descrição

Pretende-se que o seu programa seja 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 sopa de letras. No Boggle há uma grelha de 4x4 letras, sendo o objectivo do jogo encontrar
o maior número de palavras possível e quanto mais letras tiverem as palavras encontradas maior
a pontuação. As palavras são formadas usando uma qualquer letra da grelha e escolhendo para
formar as palavras qualquer letra adjacente. A única restrição é que não é possível repetir a mesma
letra(posição), numa mesma palavra.

Classes

Elemento tabela:

Classe que contém 2 construtores de modo a criar um objecto que contém um elemento do tipo E e um boleano;

HashTable:

Classe abstrata que contém o métodos:

estaAtivo(PosiçãoAtual) - Verifica se a posição atual está ativa através do boleano presente em cada objeto dentro do Array de objetos do tipo ElementoTabela

Ocupados() - Retorna quantos elementos do array estão ocupados/ativos.

hash(s) - Calcula o índice onde será colocada a String s que é passada através do parâmetro.

tamanho() - retorna o tamanho do array de objetos do tipo ElementoTabela.

proximaPosição(s) - Método abstrato que é implementado pela classe HashQuadratica visto que foi este o tipo de hashing fechado escolhido.

criarTabela(dim) - cria a tabela com o tamanho passado pelo parâmetro (dim) ou pelo tamanho predefinido (200) se não tiver nenhum parâmetro.

ePrimo(n) e proximoPrimo(n) - Algoritmos não criados por mim, utilizados para calcular o próximo número primo depois de n.

tornarVazia() - Faz 'reset' a todos os elementos da tabela tornando-os em 'null'

procurar(x) - Procura o elemento x na tabela e retorna true se a tabela o contiver ou false caso contrário.

insere(x) - calcula o indice onde x será inserido
se x já se encontra na tabela não faz nada
se x não se encontra na tabela insere-o na tabela como ativo (True)
se 80% da tabela se encontrar ocupada faz o rehash da tabela

rehash() - cria uma nova tabela com o dobro do tamanho da antiga
- copia os elementos da antiga tabela para a nova

toString() - utiliza buffers para fazer o 'append' dos elementos da tabela

print() - faz então o print da tabela com os seus elementos

HashQuadratica: Classe que por hereditariedade recebe os métodos da classe HashTable e implementa o método (anteriormente abstrato) proximaPosição(s). Contém ainda os construtores da hashtable com os parâmetros n ou nulo que a farão de tamanho n ou 200 respetivamente.

próximaPosição(s) - Calcula o valor de hash(s) e se não encontrar um elemento igual ou um indice vazio na tabela volta a calcular um novo indice sendo que o hashing é quadrático.

Boogle: Classe onde se encontra o Main(), começando por se criar um objeto Boogle sendo este uma hashQuadratica de Strings.
De seguida utiliza-se um buffer para ler as palavras do ficheiro AllWords e À medida que são lidas, são introduzidas na hash table (Boogle.insere(line);)

lerBoogle(); - Método que lê um ficheiro do tipo:

A B C D
E F G H
I J K L
L M N O

e adiciona todos os caracteres numa única String, removendo os espaços e os newLines retornando a String.

permutation(str) ,
permutation(prefix, str) - Métodos não criados mas editados por mim que calculam todas as permuntações com as letras do Boogle e caso exista alguma presen na hashtable (Boogle) faz o print da mesma.

solve() - Aplica o método permutation à string retornada por lerBoogle.

Position: Classe não utlizada que tinha como objectivo criar um objeto com os atributos linha, coluna e caracter para mais tarde fazer o print do caracter com a devida posição do mesmo no Boogle

NOTA: Como o número de permutações com 16 letras = 290594304000 o programa demora muito a correr.

Conclusão

Com a realização deste trabalho consegui por em prática os conhecimentos obtidos sobre hashtables, embora com algumas dificuldades.

Sites consultados para a obtenção dos algoritmos:
http://stackoverflow.com/
http://introcs.cs.princeton.edu/java/23recursion/Permutations.java.html