UNIVERSIDADE
DE ÉVORA

Trabalho – Boggle

Estruturas de Dados e Algoritmos I

Realizado por:

Mauro Serpa 54692
Tiago Pinto 54718
João Lóios 55469
Curso: Engenharia Informática

Introdução

O nosso objetivo com este trabalho é utilizar Estruturas de Dados da linguagem C, para resolver um Boggle.
Um Boggle é um jogo, constituído por uma grelha de 4x4 letras, onde o pretendido é encontrar as palavras escondidas na grelha, com as restrições de que não podemos repetir posições.
As Estruturas de Dados utilizadas para a implementação do nosso trabalho, foram stacks, listas e tabelas de hash.
Também foram utilizados alguns ficheiros, fornecidos pela professora:
● “boggle.txt”, um boggle, isto é uma matriz de caracteres;
● “corncob_caps_2023.txt”, uma lista de todas as palavras no inglês, num ficheiro de texto, uma palavra por linha.

Implementação

Inicialmente começamos por construir as tabelas de hash, para as palavras e prefixos, para isso, tivemos que fazer um loop que percorresse todas as palavras do dicionário, e depois a soma dos caracteres da palavra/prefixo ASCII. Para resolver colisões, fizemos o acesso quadrático. O tamanho da tabela das palavras, é o dobro número de palavras, com a intenção do load factor ser menor que 0,5, e para o tamanho da tabela dos prefixos, é o triplo do tamanho da tabela das palavras, com intenção que cada palavra tem 3 prefixos novos.

De seguida, implementamos o código para ler a matriz, de um dos ficheiros do boggle. Inicialmente, pedimos ao utilizador para escolher um dos três boggles, e depois o programa irá passar a matriz deste, para a variável “jogo”, através da função fgetc.

Na parte para resolver o boggle, primeiro criamos uma stack e uma função chamada “CreateDirections”, com tamanho 100 (pois, considerando que todas letras têm os caminhos disponíveis dá 84, somando os marcadores das letras, que são 15), com o objetivo de fazer push para a stack, de todas as direções possíveis para construir um prefixo válido, para qualquer posição do boggle, e por isso é que na função criámos uma variável para cada ponto cardeal.

Depois também criamos uma lista, que serve para guardar as soluções do boggle, com as coordenadas de cada letra de cada palavra encontrada.

Após a stack e a lista declaradas, implementamos um loop, que apenas irá terminar quando a stack estiver vazia, que percorre o boggle inteiro, e para verificar se a palavra existe ou não, guardamos as palavras na variável “palavra” e os prefixos na variável “resposta”, e depois para transformar um int num char, tivemos de somar ´0´ às variáveis inteiras das linhas e colunas.

Para o programa saber quando uma letra já não tem caminhos possíveis para seguir, fizemos push para a stack da string “–”.

Depois da função das direções ser executada, a stack faz pop para a variável check, que irá servir para traduzir o que está na stack para valores numéricos nas linhas e colunas da matriz.

Caso uma letra já tenha sido usada, então o programa passa para a letra seguinte.

Quando é encontrada uma palavra válida (para isso temos de ir ver à tabela de hash anteriormente implementada), o programa irá fazer printf da palavra achada e das coordenadas das letras que a constituem, e depois faz insert para a lista.

Conclusão

Com este trabalho, podemos concluir, que conseguimos aplicar muitos dos conhecimentos aprendidos ao longo deste semestre, em Estruturas de Dados e Algoritmos I.

Incialmente, tivemos alguma dificuldade ao realizar o código, na construção das tabelas de hash para as palavras, visto que o programa só lia a primeira palavra e quando a avança para as restantes modificava as mesmas. Mas, apesar desse problema ao inicio, após conseguirmos resolvê-lo, conseguimos continuar a construção do código.

Por fim, apesar de desafiador, gostamos de realizar este trabalho, e pretendemos melhorar os nossos conhecimentos e adquirir mais, futuramente.