Algoritmos e Estruturas de Dados

O TAD DIGRAPH

Repositório GitHub

108583 Luís Sousa

luisbfsousa@ua.pt

6 fevereiro 2024

Contents

1 Introdução

Neste trabalho será desenvolvido um programa que trabalha com Grafos Orientados. O programa conseguirá criar ficheiros, quer a partir de funções, quer a partir de um ficheiro de texto.

Será também desenvolvido um algoritmo Depth-First Searches de modo a verificar se um digrafo é fortemente conexo.

Estruturas de Dados

Os Digrafos são representados de acordo com a seguinte estrutura:

- `uint numVertices` - número de vértices no grafo
- `uint numEdges` - número de arestas no grafo
- `uint maxEdges` - número máximo de arestas permitidas
- `uint *inDeg` - grau de entrada do vértice V
- `uint *outDeg` - grau de saída do vértice V
- `uint *adjPos` - posição dos vértices adjacentes a V em adjVert
- `uint *adjVert` - array de vértices adjacentes (arestas)

2 Análise Formal

O objetivo deste algoritmo é determinar se um grafo orientado é fortemente conexo por meio de *Depth-First Searches*, explorando sistematicamente o digrafo.

A complexidade temporal do algoritmo é $O(n^2)$ devido à presença de dois loops "for" dentro um do outro. O loop "for" principal da função itera n vezes, por sua vez o loop "for" dentro da função de busca em profundidade volta a iterar n vezes. O resultado final é um total de $n * n = n^2$ operações, o que caracteriza uma complexidade temporal quadrática.

O algoritmo DFS é eficiente para grafos pequenos a médios, devido à sua complexidade porém para grafos grandes o tempo de execução pode não ser concluído em tempo útil.

A busca em profundidade é um algoritmo ideal para verificar se um digrafo é fortemente conexo mas como já foi dito anteriormente o algoritmo pode não ser eficiente para grafos maiores.

3 Análise Experimental

Para a verificar se os grafos orientados são fortemente conexos e o tempo de execação foram utilizados os grafos disponibilizados no 2º projeto da disciplina e mais alguns gerados por um ficheiro python criado para fornecer digrafos fortemente conexos num ficheiro texto com um número de arestas e vértices à escolha.

Alguns digrafos utilizados

- `DAG_1.txt` - 7 vértices e 9 arestas
- `DG_2.txt` - 15 vértices e 26 arestas
- `SWtinyDAG.txt` - 13 vértices e 15 arestas
- `conexo4.txt` - 69 vértices e 69 arestas
- `conexo5.txt` - 1000 vértices e 2000 arestas

	DAG_1	DG_2	SwtinyDAG	conexo4	conexo5
time	0.000011	0.00001	0.00001	0.000198	0.043749
calltime	0.000003	0.000003	0.000003	0.000054	0.011824
DigraphIsStronglyConnected	0	0	0	1	1

Como podemos observar pela tabela de resultados, o algoritmo utilizado irá verificar se o grafo orientado é fortemente conexo e indicar o tempo de execução da função de modo a verificar se o algoritmo desenvolvido está otimizado.

4 Conclusão

Em conclusão, o trabalho desenvolvido permitiu-me aprofundar o meu domínio na linguagem C, bem como aprofundar os meus conhecimentos sobre a estrutura de dados *Graph*.

Permitiu-me também desenvolver a capacidade de análise de otimização de código tal como uma adequada implementação de Depth-First Searches de modo a concluir se um digrafo é fortemente conexo.