universidade de aveiro  deti departamento de electrónica, telecomunicações e informática

Relatório do Segundo Projeto

O TAD GRAPH

Algoritmos e Estruturas de Dados

Equipa docente:
Professor Joaquim Madeira
Professor Professor
Professor João Manuel Rodrigues
Professor Mário Antunes
Professor Pedro Lavrador

Trabalho Realizado Por:
Luís Tojal nº119636
Diogo Duarte nº120482

1 Introdução

No ambito da unidade curricular de **Algoritimos e estruturas de dados**,realizamos um projeto cujo objetivo é desenvolver algoritmos sobre grafos sem pesos associados aos arcos, e efetuar a análise da eficiência computacional de algumas das estratégias desenvolvidas.

1.1 Grafo

Um grafo é uma representação abstrata de um conjunto de objetos e das relações existentes entre eles.É constituido por um conjunto de vértices e arestas.

2 Implementação

A implementação deste projeto, começou pela a analise dos ficheiros dos professores. Apartir dessa base inicial implementamos as funçãos que se encontravam incompletas.

- **TAD GRAPH**
- **Módulo BELLMAN-FORD**
- **Módulo TRANSITIVE-CLOSURE**
- **Módulo ALL-PAIRS-SHORTEST-DISTANCES**
- **Módulo ECCENTRICITY-MEASURES**

2.1 TAD GRAPH

2.2 Módulo BELLMAN-FORD

O algoritmo de **Bellman-Ford** é um algoritmo para encontrar o caminho de custo mínimo em um grafo orientado ou não orientado a partir de um vértice inicial, chamado de **source**. Este algoritmo possui uma complexidade superior ao algoritmo de Dijkstra, mas tem a vantagem de funcionar corretamente mesmo quando existem arestas com pesos negativos, desde que não formem ciclos negativos. Atravêz da análise do codigo proveniente no link fornecido pelo stor conseguimos desenvolver o algoritmo. No nosso código, começamos por inicializar os valores de **distância**, **antecessores** e **marcados**. A **distância** representa o custo do caminho entre o vértice inicial (source) e cada outro vértice, sendo atualizada apenas com a menor distância encontrada. Os **antecessores** são inicializados com o valor -1, indicando que nenhum caminho foi estabelecido inicialmente. Este atributo indica o vértice que permite alcançar um dado vértice com o menor custo. Por fim, o atributo **marcado** inicilizado 0, indica se é possível alcançar um determinado vértice a partir do vértice inicial.

Análise de Complexidade

Worst Case

Usando estes atributos previamente estabelecidos, iremos realizar o algoritmo para encontrar o caminho de custo mínimo. No pior caso possível, isto é, quando os vértices estão todos conectados entre si,grafo completo,neste caso o primeiro loop apenas irá executar 2 vezes pois após realizar a primeira iteração todos os vertices tem um custo de um 1 para o vertice inicial sendo impossivel formar um caminho mais curto.O numero de arestas será v-1 pois estes estão conectados a todos os vertices executo a eles próprios A complexidade do nosso código é dada por:

$$\underbrace{\sum_{i=1}^{V} 1}_{\text{Inicialize Values}} + \sum_{i=1}^{2}\sum_{i=1}^{V}\sum_{i=1}^{E} 1 = \sum_{i=1}^{V} 1 + \sum_{i=1}^{2}\sum_{i=1}^{V} E$$

$$= \sum_{i=1}^{V} 1 + \sum_{i=1}^{2}\sum_{i=1}^{V}(V-1) = V + \sum_{i=1}^{2} V*(V-1) = V + 2V^2 - 2V = 2V^2 - V$$

Nº de Vértices	Nº de Iterações	Atualizações de Distância	Time (s)	CalcTime (s)
2	4	1	0.000033	0.000033
3	15	2	0.000004	0.000004
4	28	3	0.000004	0.000004
5	45	4	0.001423	0.001423
6	66	5	0.000004	0.000004
7	91	6	0.000004	0.000004
8	120	7	0.000004	0.000004
9	153	8	0.000005	0.000005

Table 1: Métricas utilizadas nos testes com os valores fornecidos.

$$\mathcal{O}(V^2)$$

Best Case

O melhor caso acontece quando o grafo inicial não contém nenhuma aresta. Neste caso, os valores iniciais serão inicializados, e ao verificar que o grafo não contém arestas, o algoritmo será imediatamente retornado. A complexidade neste caso é:

$$\underbrace{\sum_{i=1}^{v} 1}_{\text{Inicialize Values}} = V$$

$$\Omega(V)$$

Tabela de Resultados

Nº de Vértices	Nº de Iterações	Atualizações de Distância	Time (s)	CalcTime (s)
202	202	0	0.000184	0.000184
302	302	0	0.000003	0.000003
402	402	0	0.000034	0.000034
502	502	0	0.000075	0.000075
602	602	0	0.000005	0.000005
702	702	0	0.000011	0.000011
802	802	0	0.000009	0.000009
902	902	0	0.000013	0.000013

Table 2: Métricas para o Melhor Caso com Atualizações de Distância Fixadas.

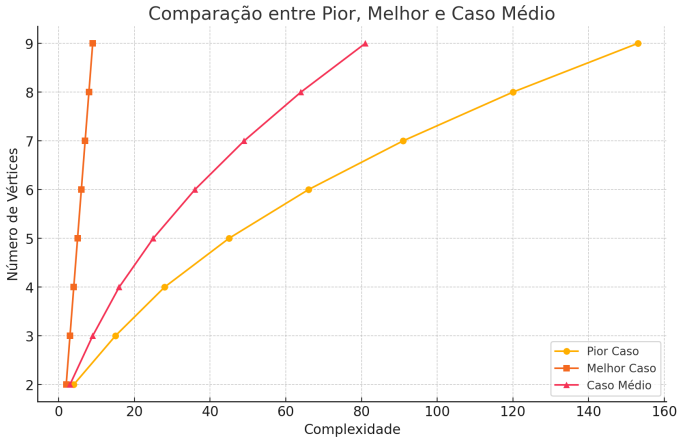

Figura 1: Gráfico comparando a complexidade e o número de vértices nos cenários de pior, melhor e caso médio.

2.3 Módulo TRANSITIVE-CLOSURE

O **Transitive-closure** tem os mesmo vertices que o grafo original.Contudo este cria uma aresta orientada para todos os verices alcançáveis a partir de um determinado vertice.Para a implementação deste modulo é nos pedido para usar o algoritmo de **BELLMAN-FORD** para determinar os vertices alcançáveis.

Análise de Complexidade

Worst Case

O pior casos deste módulo da implementação esta dependente da pior complexidade do algoritmo do **BELLMAN-FORD**. Como o pior caso do **BELLMAN-FORD** é um grafo completo para este tambem será tendo uma complexidade de

$$= \sum_{i=1}^{V} (2V^2 - V) = 2V^3 - V^2$$

Nº de Vértices	Nº de Iterações	Atualizações de Distância	Time (s)	CalcTime (s)
2	8	2	0.000022	0.000022
3	45	6	0.000003	0.000003
4	112	12	0.000003	0.000003
5	225	20	0.000003	0.000003
6	396	30	0.000003	0.000003
7	637	42	0.000003	0.000003
8	960	56	0.000003	0.000003
9	1377	72	0.000019	0.000019

Table 3: Métricas utilizadas nos testes para um novo caso.

$$\mathcal{O}(V^3)$$

Best Case

O melhor caso acontece quando o grafo inicial não contém nenhuma aresta. Neste caso ao verificar que o grafo não contém arestas, o algoritmo será imediatamente retornado. A complexidade neste caso é:

$$1$$

$$\Omega(V)$$

Nº de Vértices	Nº de Iterações	Atualizações de Distância	Time (s)	CalcTime (s)
202	1	0	0.000046	0.000046
302	1	0	0.000087	0.000087
402	1	0	0.000365	0.000365
502	1	0	0.000353	0.000353
602	1	0	0.000577	0.000577
702	1	0	0.000799	0.000799
802	1	0	0.001197	0.001197
902	1	0	0.001850	0.001850

Table 4: Métricas utilizadas nos testes no melhor caso.

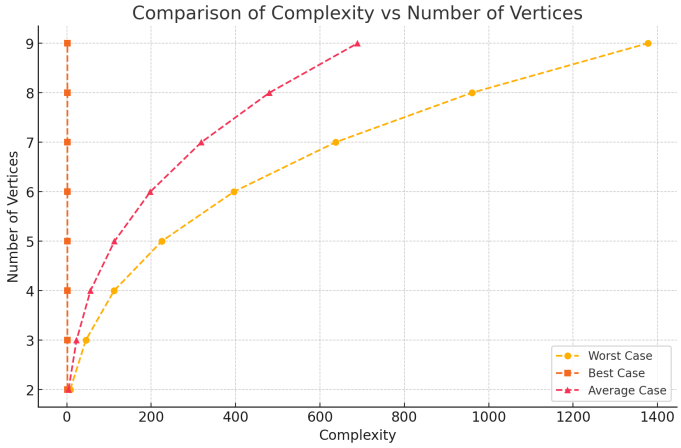

Figura 2: Gráfico comparando a complexidade e o número de vértices nos cenários de pior, melhor e caso médio.

2.4 Módulo ALL-PAIRS-SHORTEST-DISTANCES

A função `GraphAllPairsShortestDistancesExecute` calcula as distâncias mais curtas entre todos os pares de vértices em um grafo, utilizando o algoritmo de Bellman-Ford. Para cada vértice do grafo, o algoritmo de Bellman-Ford é executado para determinar as menores distâncias até todos os outros vértices. Os resultados são armazenados em uma matriz de distâncias.

Exemplo de saída:

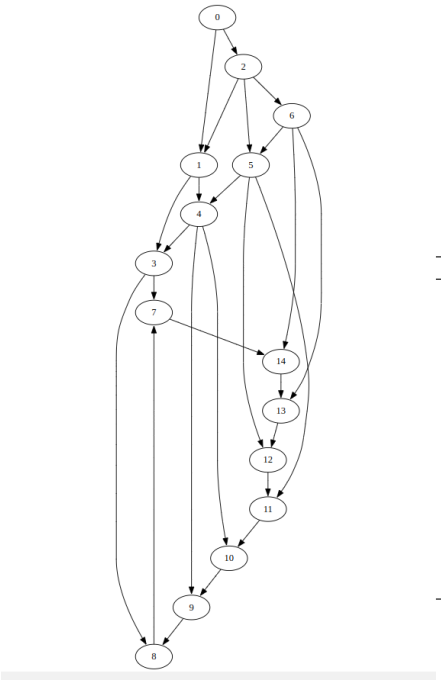

Figura 3: Exemplo de grafo utilizado para cálculos de excentricidade.

Table 5: Matriz de distâncias do grafo (15 vértices)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	1	1	2	2	2	2	3	3	3	3	3	3	3	3
1	INF	0	INF	1	1	INF	INF	2	2	2	2	6	5	4	3
2	INF	1	0	2	2	1	1	3	3	3	3	2	2	2	2
3	INF	INF	INF	0	INF	INF	INF	1	1	7	6	5	4	3	2
4	INF	INF	INF	1	0	INF	INF	2	2	1	1	6	5	4	3
5	INF	INF	INF	2	1	0	INF	3	3	2	2	1	1	5	4
6	INF	INF	INF	3	2	1	0	4	4	3	3	2	2	1	1
7	INF	INF	INF	INF	INF	INF	INF	0	7	6	5	4	3	2	1
8	INF	INF	INF	INF	INF	INF	INF	1	0	7	6	5	4	3	2
9	INF	INF	INF	INF	INF	INF	INF	2	1	0	7	6	5	4	3
10	INF	INF	INF	INF	INF	INF	INF	3	2	1	0	7	6	5	4
11	INF	INF	INF	INF	INF	INF	INF	4	3	2	1	0	7	6	5
12	INF	INF	INF	INF	INF	INF	INF	5	4	3	2	1	0	7	6
13	INF	INF	INF	INF	INF	INF	INF	6	5	4	3	2	1	0	7
14	INF	INF	INF	INF	INF	INF	INF	7	6	5	4	3	2	1	0

2.5 Módulo ECCENTRICITY-MEASURES

Este módulo é responsável por calcular e exibir as medidas de excentricidade de um grafo, incluindo o raio e o diâmetro, além de identificar os vértices centrais. As principais métricas calculadas são as seguintes:

- **Excentricidade de um vértice**: Representa a maior distância entre o vértice em questão e qualquer outro vértice acessível no grafo. Para vértices desconectados, a excentricidade é indicada como `INF`.
- **Raio do grafo**: É o valor mínimo de excentricidade entre todos os vértices do grafo.
- **Diâmetro do grafo**: Representa a maior excentricidade entre todos os vértices do grafo.
- **Vértices centrais**: São aqueles vértices cuja excentricidade é igual ao raio do grafo.

A função `GraphEccentricityMeasuresCompute` é responsável pelo cálculo dessas métricas, utilizando as distâncias mais curtas entre todos os pares de vértices. Durante a exibição dos resultados, os valores de `-1`, que indicam distâncias indefinidas, são substituídos por `INF`.

Exemplo de saída:

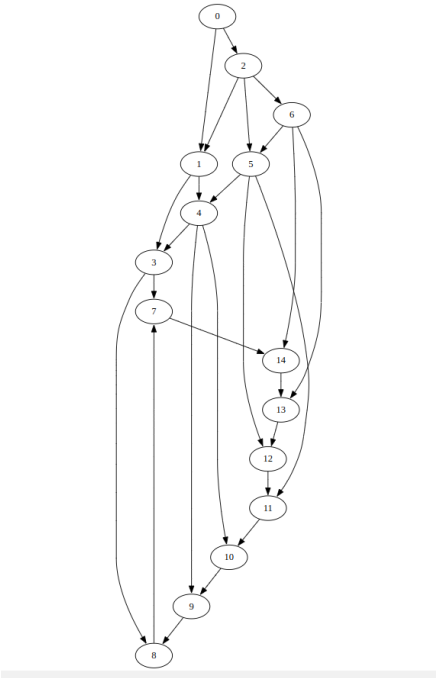

Figura 4: Exemplo de grafo utilizado para cálculos de excentricidade.

```
Graph radius: 3

Graph diameter: 7

Vertex eccentricities:
Vertex 0: 3
Vertex 1: 6
Vertex 2: 3
Vertex 3: 7
Vertex 4: 6
Vertex 5: 5
Vertex 6: 4
Vertex 7: 7
Vertex 8: 7
Vertex 9: 7
Vertex 10: 7
Vertex 11: 7
Vertex 12: 7
Vertex 13: 7
Vertex 14: 7

Central vertices:
Vertex 0
```

3 Conclusão

No âmbito deste trabalho concluimos o desenvolvimento das funções no ficheiro de implementação e garantimos o seu correto funcionamento. Aanalisamos o grau de complexidade para o pior e melhor caso dos módulos **TRANSITIVE-CLOSURE** e **BELLMAN-FORD**. Com a implementação deste trabalho, adquirimos conhecimentos importantes sobre diversos algoritmos e estratégias aplicáveis à utilização de grafos. Conseguimos aprofundar os nossos conhecimentos na linguagem **C** e desenvolver uma compreensão sólida sobre grafos, criando uma excelente base para uma futura análise mais profunda.Ao longo do projeto, fomos ainda aprendendo sobre novos algoritmos e maneiras de os tornar mais eficientes.

4 Referências

Para a realização deste projeto, foram consultados os seguintes materiais e recursos:

- Slides teóricos e guiões práticos da disciplina, disponibilizados pelos docentes.
- *Wikipedia - Bellman-Ford Algorithm*: Usado como referência para o desenvolvimento do algoritmo de Bellman-Ford.
- *Stack Overflow*: Recurso utilizado para esclarecer dúvidas e solucionar problemas encontrados no código.