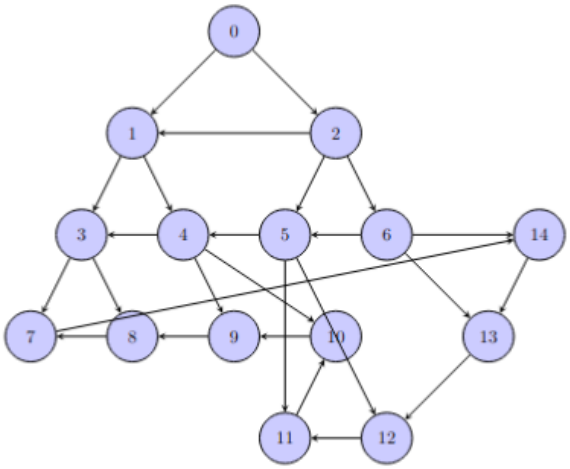
universidade de aveiro

Projeto 2

Algoritmos e Estruturas de Dados



Análise e Aplicações de Grafos em C

Daniel Martins e Eduardo Romano
115868 118736
Janeiro 2025

Contents

1 Introdução e Objetivos

Os grafos são estruturas fundamentais na ciência da computação, amplamente utilizadas para resolver problemas complexos em diversas áreas. Este projeto tem como objetivo estudar e implementar algoritmos aplicados a grafos, abordando conceitos teóricos e práticos.

Entre os algoritmos explorados estão o **Bellman-Ford**, para cálculo de caminhos de menor custo; o fecho transitivo (**Transitive Closure**), para analisar conexões indiretas entre vértices; e medidas de excentricidade (**Eccentricity Measurements**), que avaliam a importância relativa dos vértices.

Além disso, o projeto envolve o uso de estruturas de dados específicas, como listas ordenadas (**Sorted Lists**) e pilhas (**Stacks**), para implementar e otimizar os algoritmos. Foram fornecidos programas de teste para validar as soluções implementadas, garantindo a correção dos algoritmos.

O trabalho também inclui a análise da complexidade do algoritmo de **Bellman-Ford** e do algoritmo de construção do **Transitive Closure** aplicados a grafos orientados.

2 Módulo Bellman-Ford

O algoritmo de **Bellman-Ford** é utilizado para encontrar o caminho de custo mínimo entre um vértice de origem e os outros vértices em um grafo, sendo capaz de lidar com arestas de peso negativo. O algoritmo baseia-se no **princípio de relaxamento de arestas**, onde a estimativa da distância de um vértice v é atualizada se um caminho mais curto for encontrado passando por um vértice u.

O algoritmo é dividido em três fases principais:

1. **Fase de Inicialização**:
    - Alocação de memória para as estruturas de dados (*marked*, *distance*, *predecessor*) e inicialização dos valores.

- A complexidade é $O(V)$, onde V é o número de vértices.

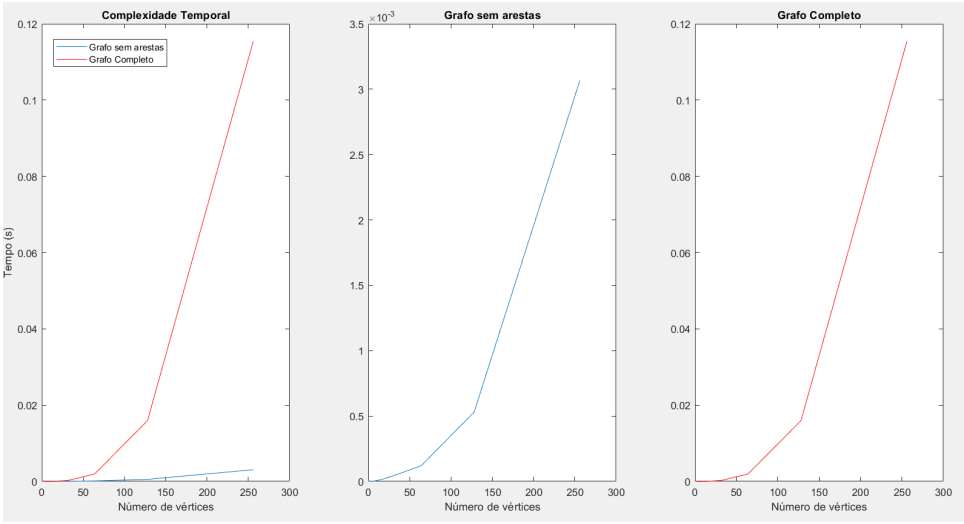
2. **Fase do Relaxamento das Arestas**:
   - O relaxamento é realizado em três loops:
     (a) O primeiro itera $V - 1$ vezes.
     (b) O segundo itera V vezes.
     (c) O terceiro itera por todos os vértices adjacentes a u, o que pode ocorrer E vezes.
   - A complexidade total desta fase é $O(V \cdot E)$, onde E é o número de arestas.
3. **Ajuste Final**:
   - O valor infinito (∞) é substituído por -1 para os vértices inalcançáveis.
   - A complexidade é $O(V)$.

Complexidade Geral: A complexidade total do algoritmo é $O(V \cdot E)$, onde V é o número de vértices e E o número de arestas.

Complexidade Temporal: Para medir a complexidade do algoritmo consideramos um melhor e pior caso, sendo estes um grafo sem arestas e um grafo completo, respetivamente.

Para a complexidade temporal do algoritmo, utilizamos a estrutura **Instrumentation**, disponibilizada pelo professor, sendo anotado o valor de **caltime** de cada função

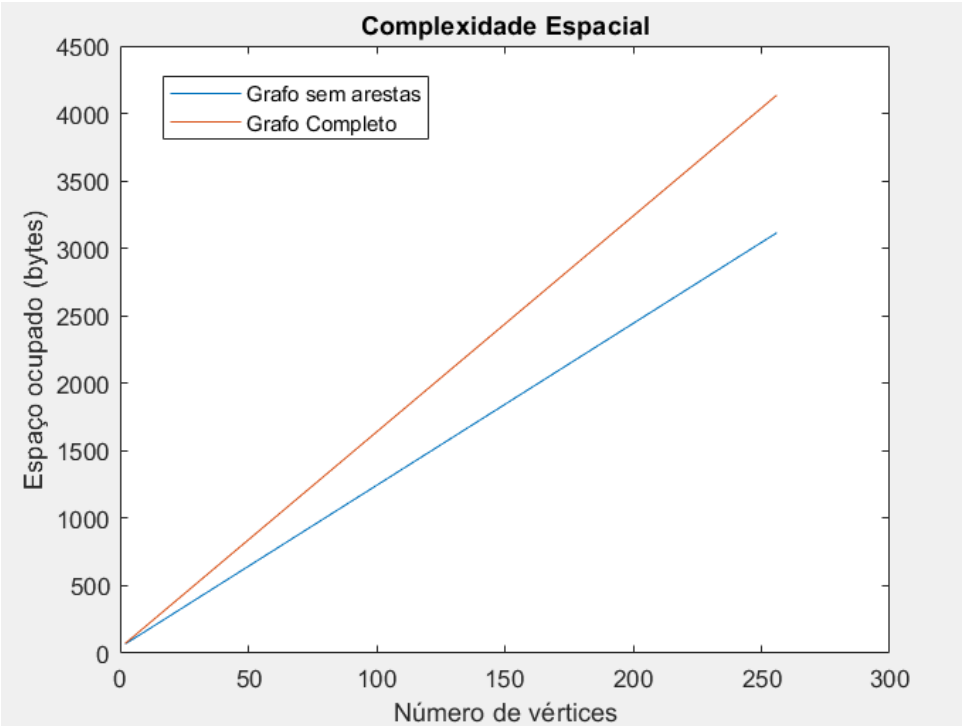
Podemos visualizar a complexidade temporal deste algoritmo através desta dispersão de dados feita em **MATLAB**:



Observa-se que o algoritmo difere em termos de complexidade consoante estejamos na presença do melhor e do pior caso.
O grafo completo, onde $E{=}V \cdot (V-1)/2$ origina uma complexidade do tipo $O(V \cdot V \cdot (V-1)/2)$, sendo esta uma complexidade cúbica, do tipo $O(V^3)$.
O grafo sem arestas, onde E é igual a zero, origina uma complexidade do tipo $O(V^2)$.
No caso geral, a complexidade tenderá a ser cúbica.

Complexidade Espacial: As métricas utilizadas para medir a complexidade espacial foram o espaço ocupado pela estrutura **_GraphBellmanFordAlg**, o espaço ocupado pelos *arrays* **distance**, **marked** e **predecessor** e o espaço temporário utilizado pelo *array* **adj_vertices**.

Foi feita uma dispersão de dados em **MATLAB** que traduz o comportamento do algoritmo.



Como o array **adj_vertices** é libertado a cada iteração, podemos observar que a complexidade espacial do algoritmo tende para um valor constante tanto no pior como no melhor caso, sendo a sua diferenção o valor constante de cada array **adj_vertices**.

3 Módulo Transitive Closure

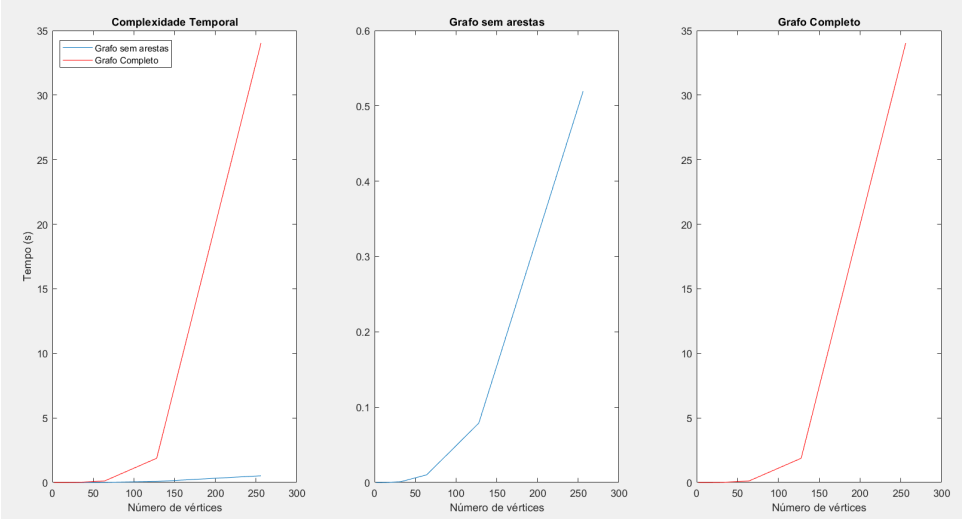
O fecho transitivo de um grafo $G = (V, E)$ mantém o mesmo conjunto de vértices V do grafo original, mas modifica o conjunto de arestas E. No fecho transitivo, existe uma aresta orientada entre os vértices u e v se, no grafo original, v for alcançável a partir de u, ou seja, se existir um caminho orientado entre u e v.

Para a solução desenvolvida , dividimos o código em 2 **for loops**:

1. **Outer Loop**: Neste ramo da função, iteramos por todos os vértices v do grafo e executamos o algoritmo de **Bellman-Ford** para cada vértice v, determinando o caminho mais curto do vértice v a todos os outros vértices do grafo.

2. **Inner Loop**: Neste ramo da função, iteramos por todos os vértices $bellman\_v$ que não são o atual vértice escolhido arbitráriamente e verificamos quais os vértices que foram alcançados através do algoritmo de **Bellman-Ford**. Caso tenham sido alcançados, o fecho transitivo irá ter uma aresta a conectar os vértices v e $bellman\_v$.

Similarmente ao algoritmo de **Bellman-Ford**, establecemos o melhor caso como um grafo sem arestas, um grafo nulo portanto, e como pior caso um grafo completo. Podemos observar que a complexidade do algoritmo no seu pior caso tende para $O(V^3 \cdot V)$, visto que irá correr o algoritmo de **Bellman-Ford** em todos os vértices, sendo esta uma complexidade quártica, e que no seu melhor caso tende para $O(V^3)$, uma complexidade cúbica , apesar de aparentar ser linear comparado com o pior caso. Podemos confirmar as nossas previsões através da seguinte dispersão de dados feita em **MATLAB**:



4 Conclusão

Neste trabalho, analisámos a complexidade de dois algoritmos aplicados a grafos orientados: o algoritmo de **Bellman-Ford** e o algoritmo do **fecho transitivo**. Para o **Bellman-Ford**, verificámos que no pior caso, com um grafo completo, a complexidade é cúbica ($O(V^3)$), enquanto no melhor caso, com um grafo sem arestas, a complexidade é quadrática ($O(V^2)$). No caso geral a sua complexidade será cúbica. Já para o **fecho transitivo**, no pior caso, com um grafo completo, a complexidade é quártica ($O(V^4)$), enquanto no melhor caso, com um grafo sem arestas, a complexidade é cúbica ($O(V^3)$). No caso geral a sua complexidade será quártica. A escolha entre os algoritmos deve considerar a estrutura do grafo e os requisitos de desempenho, sendo crucial avaliar a densidade e conectividade do grafo para determinar o impacto na execução de ambos os algoritmos. N

Daniel Martins e Eduardo Romano
115868 118736
Janeiro 2025