Relatório - Algoritmos de caminho mínimo em grafos

Samuel Patrick Lopes de Oliveira - 23.1.8115
Túlio Vilela Lopes - 23.2.8029

1. Arquivos do projeto

- O código utilizado para a implementação está disponível no gitHub: https://github.com/Tulio8998/Programacao-de-Algoritimos-em-Grafos.git

dijkstra.py: encontra o caminho mais curto de um único ponto de partida para todos os outros nós em um grafo com arestas de peso não-negativo.

floyd-warshall.py: encontra os caminhos mais curtos entre todos os pares de nós em um grafo, incluindo aqueles com arestas de peso negativo (mas sem ciclos de peso negativo).

bellmanford.py: encontra o caminho mais curto de um único ponto de partida para todos os outros nós, mesmo em grafos com arestas de peso negativo.

listaAdjacencias.py: cada nó (vértice) no grafo tem sua própria lista. Essa lista contém todos os outros nós com os quais ele tem uma conexão (aresta).

matrizAdjacencias.py: é uma forma de representar um grafo usando uma matriz bidimensional (um "quadro" de linhas e colunas).

busca.py: define a estrutura para três algoritmos de busca em grafos: Busca em Profundidade e Busca em Largura. Ele importa módulos para lidar com representações de grafos (lista e matriz de adjacências), por exemplo.

2. Como executar o código?

A execução pode ser feita via terminal. É necessário que no caso do windows o usuário entre no terminal do windows(CMD), e vá até a pasta src e execute a main, onde o arquivo é executado.

Exemplo:*cd*C:\Users\patri\Downloads\Programacao-de-Algoritimos-em-Grafos-master\Programacao-de-Algoritimos-em-Grafos-master\src

Após isso execute a seguinte linha de código**:**

python3 main.py <arquivo> <origem> <destino>

Exemplo: python3 .\main.py .\data\rg300_768.txt 20 100

3. Métricas obtidas através dos testes

Grafos	Dijkstra			Bellman-Ford			Floyd-Warshall		
	T.Médio	M.média(MB)	Custo médio	T.Médio	M.média(MB)	Custo médio	T.Médio	M.média(MB)	Custo médio
toy	0.00	0.000449	2.66666	2.976	0.183	9	0.000409	0.001595	3
faceboook_combined	4.2724	0.225709	3.5	0.2976547	0.06820	4.2.	TEMPO_LIMITE		
rg300_768	0.0070	0.014498	286.2	0.01161	0.01318	234.4.	59.11163	3.566419	256.9
rg300_4730	0.0124	0.009779	12.7	0.0077	0.00667	14.6	16.43242	3.254822	14.5
rome99c	0.6184	0.276086	12848.4	1.465	0.22637	13.9	TEMPO_LIMITE		
USA-road-dt.DC	5.6505	0.803928	14208.9	13.99	0.63399	13.7	TEMPO_LIMITE		

4. Análise dos algoritmos

Dijkstra: Geralmente é o mais rápido e eficiente para grafos grandes e esparsos, como demonstrado nos grafos USA-road-dt.DC e rome99c, onde foi o único a completar a execução. Foi o algoritmo com maior eficiência.

Bellman-Ford: É uma boa alternativa, em grafos com arestas de pesos negativos. Apresentou um bom desempenho em termos de tempo e memória em grafos de menor escala e em facebook_combined chegou a sueperar Dijkstra em alguns casos. Agora, se tratando de grafos grafos muito grandes é pouco recomendado devido a sua complexidade que pode aumentar.

Floyd-Warshall: Mesmo que seja útil para encontrar todos os pares de caminhos mínimos em um grafo, sua alta complexidade de tempo o torna impraticável para grafos de grande. Ele apresentou TEMPOO_LIMITE em grafos como facebook_combined, rome99c e USA-road-dt.DC.

5. Conclusão

A escolha do algoritmo ideal depende da estrutura e do tamanho do grafo. Para grafos grandes e esparsos, **Dijkstra** se destaca como a melhor opção. Para grafos de menor porte, **Bellman-Ford** pode ser uma alternativa viável e, em alguns casos, até mais eficiente. **Floyd-Warshall** é a opção menos recomendada para grafos grandes devido à sua complexidade de tempo.