UNIVERSIDADE FEDERAL DE MINAS GERAIS
ALGORITMOS E ESTRUTURAS DE DADOS III
2017/1

TRABALHO PRÁTICO 1 – ENTREGANDO LANCHES

Deiziane Natani da Silva – TD1

1. Introdução

O trabalho consiste em praticar os conceitos sobre grafos estudados em sala. O objetivo é ajudar Gilmar, que é responsável pela logística em seu novo emprego. A empresa que Gilmar trabalha possui diversas franquias de entregas de lanches pela cidade. Essas entregas são feitas por vários ciclistas que transitam por ciclo faixas que vão em apenas um sentido. Cada ciclo faixa tem um limite de ciclistas que podem transitar por ela por hora. O trabalho de Gilmar é determinar qual a maior quantidade de ciclistas que podem trafegar por cada ciclo faixa por hora.

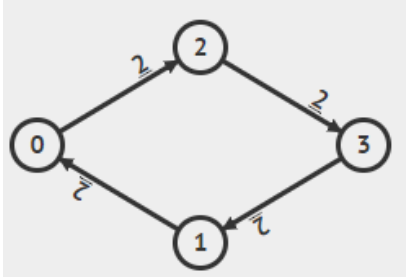
A cidade é representada por um mapa onde há interseções que possuem ou clientes, ou franquias, podendo também não ter nenhum dos dois. O problema é modelado através de um grafo ponderado e com base nele é determinado o maior fluxo.

2. Solução do Problema

Para resolver o problema e ajudar Gilmar, é criado um grafo ponderado onde cada vértice representa uma interseção, e cada aresta representa uma ciclofaixa, tendo como peso a quantidade de ciclistas permitidas por hora. Esse grafo é representado por uma matriz VxV onde V é o número de vértices do grafo. Cada linha e cada coluna representam um vértice de 0 a V-1. Caso o valor em uma posição da matriz seja 0, quer dizer que não há uma aresta que liga o vértice A ao vértice B. Caso o valor seja maior que 0, então esse valor representa o peso de uma aresta que liga os dois vértices. Exemplo:

	0	**1**	**2**	**3**
0	0	0	2	0
1	2	0	0	0
2	0	0	0	2
3	0	2	0	0

Grafo



Na matriz acima, temos 4 vértices de 0 a 4, onde as posições com valor 0 significam que não existem arestas. Onde há 2, significa que existem uma aresta com peso 2.

Vértices marcados como *source* são as franquias, e *sink* são os clientes.

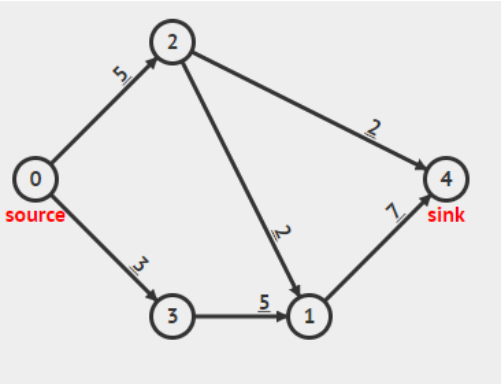
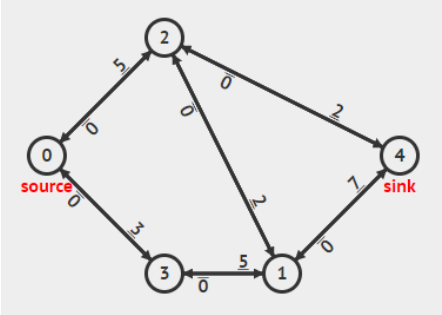


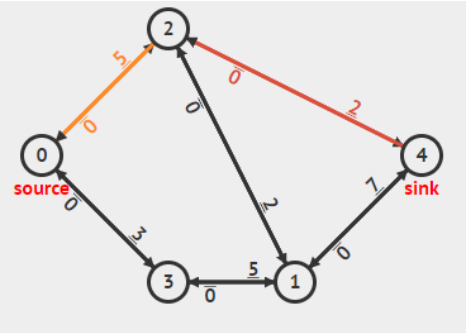
Figura 1 - Franquia no vértice 0, cliente no vértice 4

Após a criação do grafo, é utilizado o método de Ford-Fulkerson para encontrar o fluxo máximo de um vértice a outro. Nessa implementação, é usado o algoritmo de Edmonds-Karp, uma implementação do método de Ford-Fulkerson, que garante que o cálculo do fluxo sempre irá terminar. Utilizando o grafo da Figura 1, a execução do algoritmo ocorre da seguinte forma:

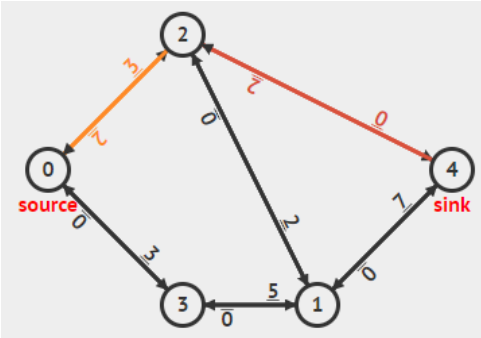
- É criado um grafo residual, onde todas as arestas no sentido contrário têm fluxo 0



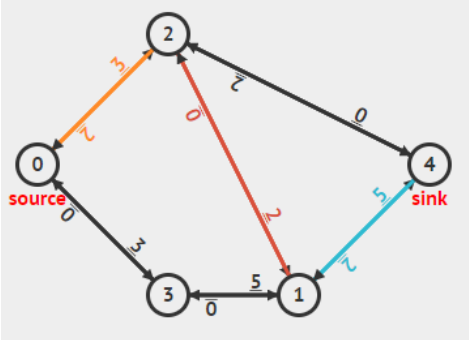
- Em seguida é efetuada uma busca em largura para achar um caminho de aumento. A busca em largura consiste em começar por um vértice e explorar todos os vértices adjacentes. Então, para cada um desses vértices mais próximos, exploramos os seus vértices vizinhos inexplorados e assim por diante, até que ele encontre o alvo da busca. No caso, o caminho mais curto a ser escolhido de 0 a 4, é primeiro por 2 e em seguida de 2 a 4. Note que o gargalo é 2 (aresta vermelha).



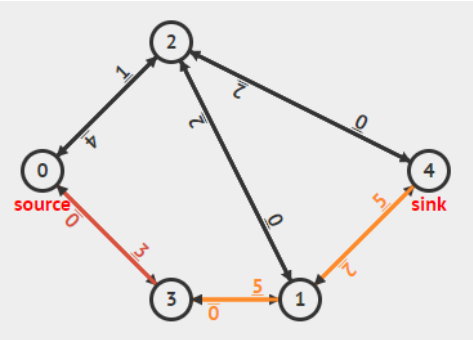
- O valor de gargalo encontrado é decrementado das arestas do caminho. Esse valor é armazenado no grafo residual. Note que a aresta vermelha agora possui peso 0, o que significa que nenhum ciclista pode mais transitar por aquela via.



- Em seguida, é verificado se há mais algum caminho de 0 a 4. Se encontrado o mesmo procedimento anterior é executado, até que não haja nenhum caminho restante.

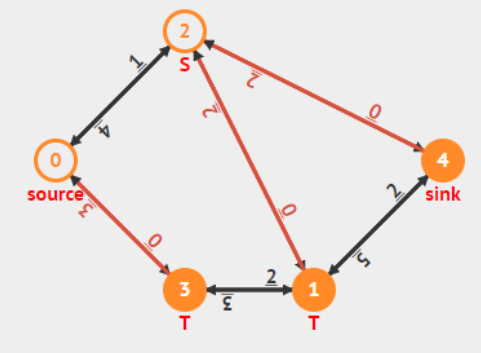


2- Novo caminho encontrado (Gargalo = 2)



3- Novo caminho encontrado (Gargalo = 3)

- Após todos os caminhos serem explorados temos o valor do fluxo máximo, que é a soma de todos os valores encontrados anteriormente, 7.



4*- Nenhum caminho possível encontrado*

Como podemos ter várias franquias e vários clientes, esses passos são repetidos novamente para cada par diferente de franquias e clientes, sendo o fluxo máximo final a soma de todos os resultados.

3. **Análise de Complexidade**

Busca em largura (BFS): Supondo que o grafo seja representado por listas de adjacência, a complexidade de tempo da busca em largura é O(V+A), sendo V o número de vértices e A o número de arestas. Como aqui o grafo foi representado por uma matriz de adjacência, o complexidade é $O(V^2)$, já que ele percorrerá a matriz de tamanho VxV.

Algoritmo Ford-Fulkerson (Utilizando Edmond-Karps): A ideia do algoritmo de Edmond-Karps é usar a busca em largura no método de Ford-Fulkerson, já que a busca em largura sempre pega o menor número de vértices. Quando usada a BFS, o pior caso é reduzido para $O(VA^2)$. Mas como a implementação utilizada usa uma matriz de adjacência, o BFS é $O(V^2)$ a complexidade de tempo do algoritmo fica $O(V^3A^2)$. Esse valor poderia ser reduzido com a utilização de uma lista de adjacências ao invés da matriz ou com a utilização do algoritmo de Dinic.

4. **Testes**

Foram executados todos os testes toy disponibilizados no moodle. O programa se mostrou eficaz em todos os casos, resultando no output correto.

Também foram feitos testes com entradas disponibilizadas por colegas no fórum. Essas entradas são grafos grandes, que obedecem a estrutura exigida pela especificação do problema.

O primeiro, big_1, com 1000 vértices e 249750 arestas, sendo 3 vértices franquias e 3 clientes. O resultado encontrado foi 21004, e seu tempo de execução foi 176s.

O segundo, big_2, com 2000 vértices e 999500 arestas, 3 franquias e 3 clientes. Teve como resultado 14720, e seu tempo de execução foi 251s.

O terceiro, big_3, com 3000 vértices e 2249250 arestas, 3 franquias e 3 clientes. O resultado foi 75149, com tempo de execução 1650s.

É possível perceber que praticamente dobrando o número de vértices e de arestas, o tempo de execução cresce de forma muito rápida, já que a complexidade de tempo do algoritmo é $O(V^3A^2)$, um polinômio de grau 3, porém o número de arestas é muito superior ao número de vértices em todos os casos.

5. **Referências**

O algoritmo de Ford-Fulkerson utilizado pode ser encontrado em: http://www.geeksforgeeks.org/ford-fulkerson-algorithm-for-maximum-flow-problem/

Alguns dos exemplos utilizados nos testes
https://drive.google.com/file/d/0B-hnEpxV4DH8X2Rvekhad19JMVk/edit