Algoritmos e Estruturas de Dados III
Documentação Trabalho Prático 1 - Entregando Lanches

Aluno: Semar Augusto da Cunha Mello Martins
Matrícula: 2016006905

Introdução:

Gilmar conseguiu um emprego na área de logística de uma empresa de entrega de lanches por meio de bicicletas. Após alguns anos de trabalho, houve uma mudança no tamanho das ciclofaixas, fazendo com que algumas ciclofaixas se tornassem difícil de se trafegar e colocando algumas ciclofaixas com apenas uma direção sendo permitida. Dessa forma, a dona da empresa, Rada, decidiu mapear as ciclofaixas da cidade e analisou o máximo de ciclistas que poderiam estar trafegando em uma ciclofaixa, visando a segurança dos mesmos.

O papel de Gilmar e do trabalho prático apresentado é, dado o mapa de ciclofaixas, a localização de todos os clientes e franquias e considerando um número de ciclistas arbitrariamente grande, determine o maior número de ciclistas que devem sair de cada franquia por hora para que aconteça o maior número de entregas possíveis. Ou seja, o trabalho prático 1 é um problema de fluxo máximo.

Metodologia:

Para resolver o problema de Gilmar, foi implementado o algoritmo de Edmonds-Karp, que é uma melhoria do algoritmo de Ford-Fulkerson devido ao uso de uma busca em largura para achar o menor caminho mínimo da rede residual. Para a implementação de Edmonds-Karp foi usado uma matriz de adjacência pois como para cada vertice são guardados somente dois inteiros, um grafo de uma empresa de entrega de lanches não deve ser grande o suficiente para não caber na memória primaria de um computador atual. Devido a uma confusão feita durante a implementação, o grafo que armazenava as capacidades dos fluxos se tornou o grafo residual. Para que o algoritmo de Edmonds-Karp funcione propriamente, foram adicionados dois vértices ao grafo, um que serve como a fonte (vértice que possui arestas saindo dele e ligando a todas as franquias) e outro como o sorvedouro (vértice que recebe arestas de todos os clientes).

O algoritmo de Edmond-Karp consiste na implementação de um grafo de uma rede residual (um grafo que consiste de 0's, caso não haja nenhuma aresta entre u e v; capacidade máxima - fluxo atual, caso haja uma aresta entre u, v; e uma aresta de retorno que consiste em -(fluxo atual). Essa aresta de retorno é armazenada na posição mat[v][u] ao invés de mat[u][v], ela serve para resolver casos de sobrecarga em uma aresta específica.). Nesse grafo, roda-se uma busca em largura e, para cada um dos caminhos aumentadores (caminhos aumentadores são caminhos no grafo de rede residual que saem da fonte e chegam no sorvedouro, sem repetição de vértices e passando sempre por arestas com peso diferente de zero, ou seja, que não atingiu seu potencial máximo ainda (pois as arestas são definidas por (capacidade máxima - fluxo atual). que não foram visitadas ainda que leve da fonte ao sorvedouro, aumenta-se o fluxo nesse caminho no menor número possível, para não sobrecarregar nenhuma aresta. Dessa maneira, sabemos que quando não houverem mais caminhos aumentadores, teremos chegado no fluxo máximo permitido pela rede de fluxo usada.

Análise de Complexidade:

main.c - O(E) - pois somente faz a leitura do arquivo, o qual é majoritariamente definido pelo numero de arestas que possui.

queue.c -

- **createEmptyQueue -** O(1)
- **emptyQueueTest -** O(1)
- **pushQueue -** O(1)
- **popQueue -** O(1)
- **printQueue -** O(tamanho da fila)

graphMatrix.c -

- **createGraph -** O(1)
- **insertEdge** - O(1)

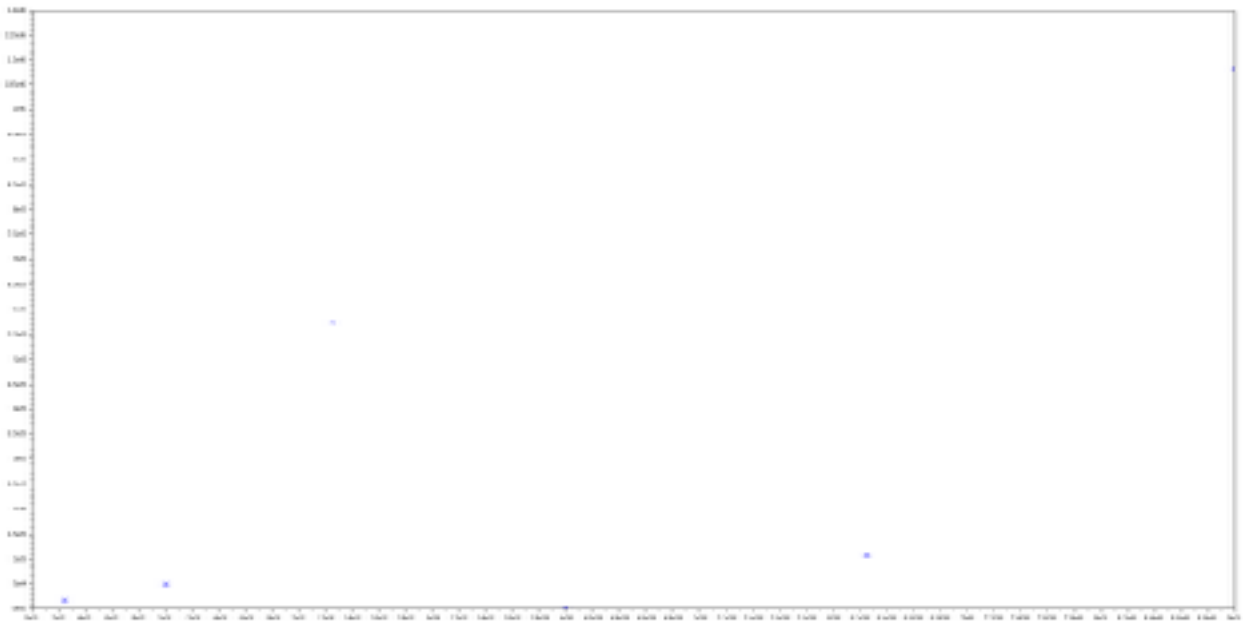
testEdgesExistence - O(1)
testIsolated - O(V)
firstAdjVertex - O(V)
nextAdjList - O(V)
removeEdge - O(1)
destroyGraph - O(V)
printGraph - O(V^2)

maximumFlow.c -
VisitBFS - O(V + E)
EdmondKarp - O(V(E^2))

TOTAL - O(V * Eˆ2)
Gasto de memoria: O(V^2) - Pois somente o grafo de residuo e fluxo são determinantes.

Experimentos:
Para testar o programa foram usados os casos toy e casos postados por alunos e monitores no Moodle.

Análise de Resultados:
Todos os testes foram executados em um macOS Sierra, com um processador Intel I7 de 2,2GHz e uma memória RAM de 16gb DDR3.
Apesar do algoritmo de edmondKarp ser O(V*Eˆ2), esse é um limite superior. Os resultados adquiridos durante os experimentos foram bastante diferentes do limite dado pelo algoritmo. O tempo de execução real depende da média de vértices adjacentes que cada vertice tem e também de se os vértices ligados ao sorvedouro são muitos e se eles são ou não preenchidos rapidamente pelo fluxo máximo permitidos por eles.
(dê zoom para enxergar devidamente o gráfico, foi plotado usando somente o número de arestas e o tempo gasto, já que o número de arestas é mais determinante, no limite superior, do que o número de vértices)



Conclusão:
O problema de Gilmar pode ser implementado de maneira simples, sem um gasto exagerado nem de processamento nem de memória.