AEDS III - Trabalho Prático 1
Entregando Lanches

Ingrid Elisabeth Spangler

22 de Maio de 2017

1 Introdução

Uma franquia de lanchonetes precisa saber como entregar seus lanches mais eficientemente para seus clientes, dado um grafo com os endereços das franquias, das casas dos clientes e dados sobre o trânsito das ciclovias da cidade que conectam ambos. Há restrições sobre a quantidade de entregadores ciclistas que podem passar por hora nestas ciclovias, então a franquia precisa de um algoritmo que maximiza a quantidade de entregas possíveis dada as restrições, ou seja, o fluxo máximo de ciclistas que podem trafegar em um mesmo horário entre as lojas e os clientes sem que haja engarrafamento.

1.1 Grafos

Grafos são uma estrutura de dados que contém um conjunto de vértices e arestas, que se relacionam, podendo estes armazenar informações como peso, fluxo, direção, dentre outras. O problema utiliza um grafo direcionado com valores nas arestas referentes ao fluxo.

2 Solução do Problema

O algoritmo utilizado é o método de Ford-Fulkerson, publicado em 1956 por L. R. Ford, Jr. e D. R. Fulkerson para computar fluxo máximo em um grafo de fluxo. Ford-Fulkerson envia fluxo pelas arestas enquanto houver um caminho aumentador com capacidade positiva entre a origem e o destino, descontando o fluxo residual do grafo a cada iteração.

2.1 Busca em Largura

Para encontrar o caminho aumentador, o método usa o algoritmo de busca em grafos pesquisa em largura (Breadth-First Search ou BFS), cuja lógica consiste em visitar um vértice de cada vez e enfileirar seus vizinhos até que o destino seja alcançado. O caminho é salvo em um vetor.

2.2 Adaptação

Como Edmond-Karp suporta apenas uma “origem” e um “destino” para computar o valor, e há mais de uma franquia e mais de um cliente nos grafos do problema, foi necessário criar dois vértices auxiliares, um representando a “distribuidora matriz”, que possui todas as franquias como vértices adjacentes, com fluxo infinito de transporte, e o vértice final, para onde todos os clientes apontam, também com fluxo infinito. Estes cumprem o papel de fonte e destino no algoritmo, sem necessidades de maiores mudanças na heurística original.

Análise de Complexidade

A estrutura de dados utilizada foi um vetor de vértices, cada vértice contém um vetor de adjacência que armazena apenas dados do tipo inteiro. Portanto, o espaço máximo utilizado pelo programa será linearmente proporcional ao número de vértices e arestas. O tempo de execução do programa depende principalmente da co

Função	Tempo	Espaço
main()	$\mathcal{O}(E+F+C)$	$\mathcal{O}(1)$
G_init()	$\mathcal{O}(V)$	$\mathcal{O}(2V)$
G_insere()	$\mathcal{O}(1)$	$\mathcal{O}(A)$
G_free()	$\mathcal{O}(V)$	$\mathcal{O}(1)$
G_BFS()	$\mathcal{O}(V+A)$	$\mathcal{O}(V)$
G_Ford_Fulkerson()	$\mathcal{O}(VA^3)$	$\mathcal{O}(V)$
F_init()	$\mathcal{O}(1)$	$\mathcal{O}(1)$
F_push()	$\mathcal{O}(V)$	$\mathcal{O}(1)$
F_pop()	$\mathcal{O}(1)$	$\mathcal{O}(1)$
F_vazia()	$\mathcal{O}(1)$	$\mathcal{O}(1)$
F_free()	$\mathcal{O}(V)$	$\mathcal{O}(1)$
Cliente()	$\mathcal{O}(A^2)$	$\mathcal{O}(1)$

Confere()	$\mathcal{O}(A^2)$	$\mathcal{O}(1)$
Franquia()	$\mathcal{O}(1)$	$\mathcal{O}(1)$
Source_init()	$\mathcal{O}(F)$	$\mathcal{O}(F)$
S_insere()	$\mathcal{O}(1)$	$\mathcal{O}(1)$
min()	$\mathcal{O}(1)$	$\mathcal{O}(1)$

Avaliação Experimental

As respostas do programa foram corretas para todos os testes toys e casos extras testados.
O programa foi testado em um notebook com as seguintes especificações:

- Sistema: Manjaro Linux
- Kernel: 4.4.41-1-Manjaro
- Processador: core i7-3537U, 2.0GHz
- Memória: 8GB

A bateria de testes consistiu de 6 testes extras com altíssima densidade.

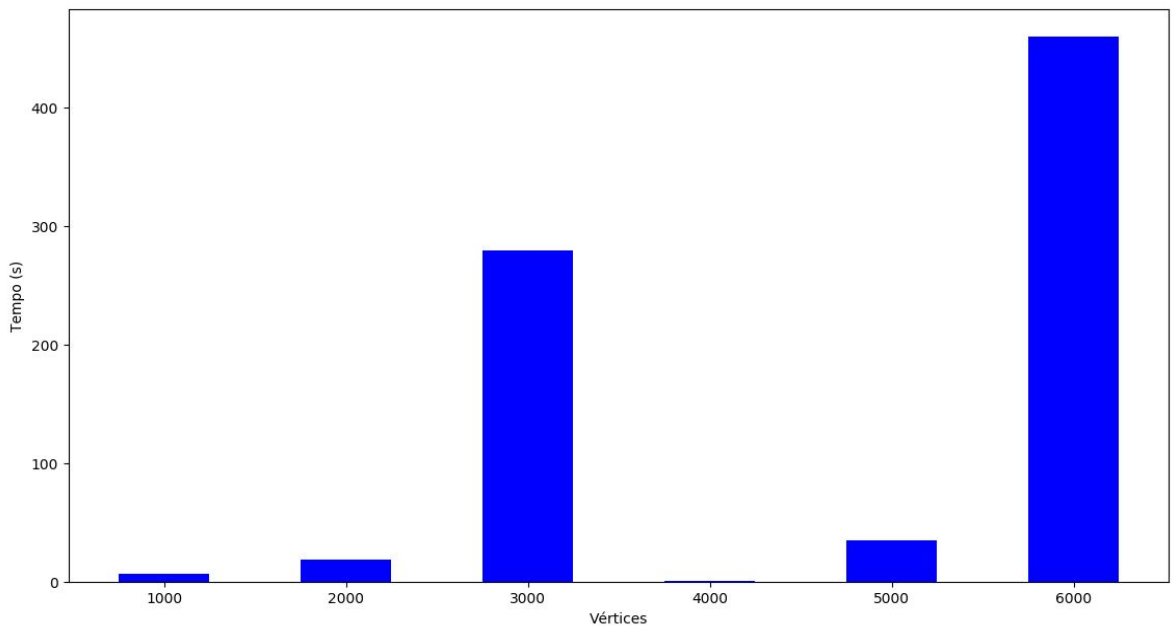
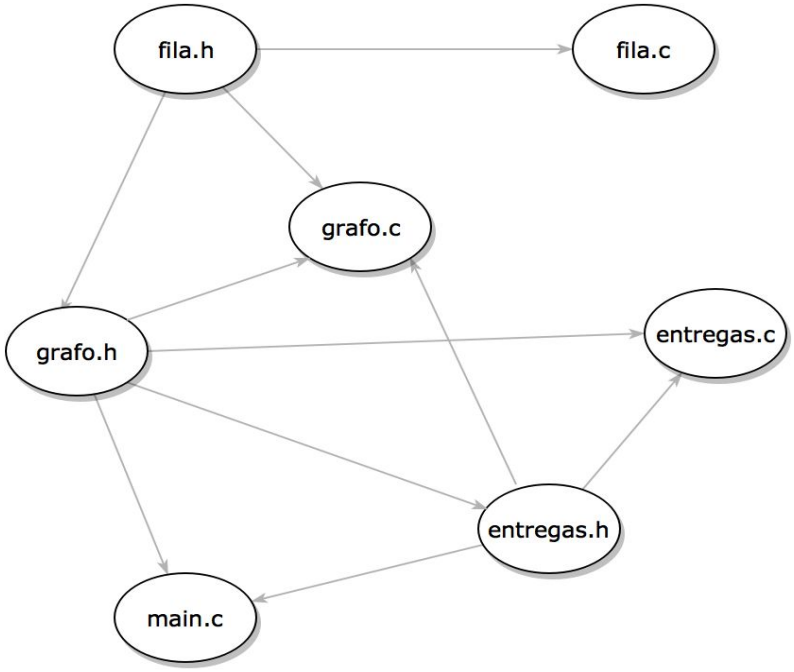


Figura: Tempo em segundos por vértice para cada caso de teste extremo

Diagrama de dependências



Conclusão

O trabalho não foi simples de ser implementado, porém serviu como uma boa introdução a projetos mais complexos e foi um grande aprendizado quanto à correta utilização de ponteiros e alocação dinâmica nestes.