Trabalho Prático 1: Entregando lanches

Algoritmos e Estruturas de Dados III – 2017/1
Entrega: 21/05/2017

1 Introdução

Gilmar perdeu seu emprego devido a uma brecha que seu empregador encontrou na justiça trabalhista, que o demitiu sem hesitar pra cortar seus gastos. Ele ficou muito chateado com o ocorrido e saiu em busca de emprego.

Por sorte, sua amiga Rada, que gosta muito de bicicletas, abriu uma firma de entrega de lanches operados por ciclistas, uma vez que esta cidade tem boas ciclofaixas. Rada sabe da experiência de Gilmar e ofereceu a ele um emprego no setor de logística, Gilmar será responsável por coordenar a entrega de lanches em diversos pontos da cidade.

A empresa funciona da seguinte maneira. Para cada franquia da firma de Rada temos ciclistas saindo para fazer entregas. Cada ciclista pode trafegar por qualquer sequência de caminhos ligado por ciclofaixas desde que ele sempre chegue em qualquer um dos clientes de Rada localizados na cidade.

Passaram-se vários anos e tudo corria muito bem para Rada e Gilmar, até que um dia Dora, a empreiteira, assumiu a prefeitura da cidade. Dora calculou que seria muito benéfico para a cidade se algumas vias fossem alargadas para que suportasse maior quantidade de veículos automotores, uma vez que esta estaria sofrendo com engarrafamentos. Fazer isso, no entanto, atrapalharia o negócio de Rada, pois suas bicicletas encontrariam dificuldades para trafegar em algumas vias, além disso cada via agora possui ciclofaixa indo em apenas uma única direção.

A fim de zelar pela segurança de seus funcionários entregadores de lanches, Rada mapeou cada interseção e ciclofaixa da cidade e analisou qual seria o limite de ciclistas que deveriam trafegar em cada ciclofaixa por hora.

A cidade então foi mapeada em interseções que se ligam por ciclofaixas, em cada ciclofaixa ligando duas interseções deve haver uma quantidade máxima de ciclistas trafegando por hora por questões de segurança.

Sabe-se que a cidade possui várias franquias da firma de Rada, ela possui um número arbitrariamente grande de ciclistas em cada franquia. Além disso, a cidade possui vários pontos onde os clientes de Rada estão localizados.

Misteriosamente tanto os clientes como as franquias estão localizadas nas interseções mapeadas por Rada.

Agora é tarefa de Gilmar definir como vai ser a logística desse negócio todo. Ele possui um mapa de interseções e ciclofaixas (que só seguem uma única direção) ligando-as. Uma interseção pode ou não conter uma franquia de Rada como também pode ou não conter um cliente, mas nunca conterá ambos. Em cada ciclofaixa ligando duas interseções temos um limite de segurança para o número de ciclistas trafegando por hora.

O trabalho de Gilmar é entregar para Rada qual é a quantidade máxima de ciclistas que pode sair das franquias por hora de modo que a segurança dos ciclistas seja levada em conta. Ajude Gilmar propondo uma solução para este problema baseada em grafos e implemente-a.

2 Entrada e saída

Entrada A entrada começa com uma linha contendo quatro números inteiros V, E, F e C indicando respectivamente a quantidade de interseções, a quantidade de ciclofaixas, o número de interseções onde há uma franquia e o número de interseções onde há clientes.

Cada uma das E linhas seguintes contém 3 números inteiros u,v e m $0 \leq u, v < V$, $m > 0$, que indica que há uma ciclofaixa na direção de u para v e que nessa ciclofaixa só podem trafegar m ciclistas por hora.

Cada uma das F linhas seguintes contém 1 número inteiro f $0 \leq f < V$ que indica que há uma franquia naquela interseção indicada pelo índice f.

Cada uma das C linhas seguintes contém 1 número inteiro c $0 \leq c < V$ que indica que há clientes naquela interseção indicada pelo índice c.

Para cada entrada é garantido que $V \geq 2, E \geq 1, F \geq 1, C \geq 1$

Exemplos de entrada

```
6 8 2 2
0 2 8
0 3 4
1 2 8
1 3 8
2 3 6
2 4 8
3 4 3
3 5 11
0
1
4
5
```

Saída Imprima em uma linha o número máximo de ciclistas que podem sair das franquias por hora.
Exemplo de saída

22

3 O que deve ser entregue

Deverá ser submetido um arquivo **.zip** contendo somente uma pasta chamada `tp1` e dentro desta deverá ter: (i) Documentação **em formato PDF** e (ii) Implementação.

Documentação Poderá ter no máximo 10 páginas e deverá seguir tanto os critérios de avaliação discutidos na Seção 4.1, bem como as diretrizes sobre a elaboração de documentações disponibilizadas no *moodle*. Além disso, a documentação deverá conter análise experimental validando as complexidades de tempo e espaço.

Implementação Código fonte do seu TP (*.c* e *.h*), com solução baseada em **grafos**.

Makefile Inclua um *makefile* na submissão que permita compilar o trabalho. É obrigatório o uso das *flags*: **-Wall -Wextra -Werror -std=c99 -pedantic** na compilação.

4 Avaliação

Eis uma lista **não exaustiva** dos critérios de avaliação que serão utilizados.

4.1 Documentação

Introdução Inclua uma breve explicação do problema que está sendo resolvido no seu trabalho e um resumo da sua solução.

Solução do Problema Você deve descrever a solução do problema de maneira clara e precisa, detalhando e justificando os algoritmos e estruturas de dados utilizados. Para tal, artifícios como pseudo-códigos, exemplos ou diagramas podem ser úteis. Note que documentar uma solução não é o mesmo que documentar seu código. **Não** é necessário incluir trechos de

código em sua documentação nem mostrar detalhes de sua implementação, exceto quando estes influenciem o seu algoritmo principal, o que se torna interessante.

Análise de Complexidade Inclua uma análise de complexidade de tempo e espaço dos principais algoritmos e estrutura de dados utilizados. Cada complexidade apresentada deverá ser devidamente **justificada** para que seja aceita.

Avaliação Experimental Sua documentação deve incluir os resultados de experimentos que avaliem o tempo de execução de seu código em função de características da entrada. Cabe a você gerar entradas para esses experimentos. Por exemplo: em um trabalho de grafos seria interessante variar o número de vértices mantendo a proporção de arestas a mesma, variar o número de arestas mantendo o número de vértices também seria válido (Obviamente utilizando valores consideravelmente grandes mas não grandes a ponto de ser um problema inviável de tratar). Para tal, um gráfico mostrando o tempo de execução em função do tamanho da entrada pode ser interessante. Você também deve interpretar os resultados obtidos. Comente sobre cada gráfico ou tabela que você apresentar mostrando o que é possível concluir a partir dele.

4.2 Implementação

Linguagem & Ambiente O seu programa deverá ser implementado na linguagem **C** e poderá fazer uso de funções da biblioteca padrão da linguagem. Trabalhos que utilizem qualquer outra linguagem de programação e/ou que façam uso de outras bibliotecas que não a padrão serão zerados. Além disso, certifique-se que seu código compile e funcione corretamente nas máquinas **Linux** dos laboratórios do DCC.

Casos de teste A sua implementação passará por um processo de correção automatizado, portanto, o formato da saída do seu programa deve ser idêntico aquele descrito nessa especificação. Saídas com qualquer divergência serão consideradas erradas, mesmo que as divergências sejam *whitespaces*. e.g. espaços, *tabs*, quebras de linha, etc. Para auxilia-lo na depuração do seu código, será fornecido um pequeno, **não-exaustivo**, conjunto de entradas e suas respectivas saídas. É seu dever certificar-se que seu código funciona corretamente para qualquer entrada válida.

Alocação Dinâmica Algoritmos e estruturas de dados deverão fazer uso de memória alocada dinamicamente (`malloc()` ou `calloc()`). Certifique-se que seu programa utiliza essas regiões de memória corretamente, pois os monitores penalizarão implementações que realizam *out-of-bounds access* e que tenham vazamento de memória (não desalocar memoria dinâmica).
A alocação dinâmica deverá fazer uso das funções malloc() ou calloc() da biblioteca padrão C, bem como liberar tudo o que for alocado utilizando free(), para gerenciar o uso da memória. **DICA:** Utilize `valgrind` antes de submeter o seu TP.

Qualidade do código Seu código também será avaliado no quesito de legibilidade, dando atenção, porém não limitando-se, aos seguintes items: (i) **INDENTAÇÃO**; (ii) nomes de variável e função descritivos e claros; (iii) Modularização adequada; (iv) Comentários dentro de funções, explicando o que certos trechos mais complicados fazem; (v) Comentários fora de funções, explicando, em alto-nível, o que as funções mais importantes fazem; (vi) funções concisas que desempenham somente uma tarefa; (vii) **Proibido uso de variáveis globais**.

Atrasos Trabalhos poderão ser entregues após o prazo estabelecido, porém sujeitos a uma penalização regida pela seguinte fórmula:

$$\Delta_p = \frac{2^{d-1}}{0.32}\,\%$$

Por exemplo, se a nota dada pelo corretor for 70 e você entregou o TP com 4 dias corridos de atraso, sua penalização será de $\Delta_p = 25\%$ e, portanto, a sua nota final será: $N_f = 70 \cdot (1 - \Delta_p) = 52.2$. Note que a penalização é exponencial e 6 dias de atraso resultam em uma penalização de 100%.

5 Consideração Final

Assim como em todos os trabalhos dessa disciplina é estritamente proibida a copia parcial ou integra de códigos, seja da internet ou de colegas. Utilizaremos o algoritmo *MOSS* para detecção de plágio em trabalhos, seja honesto. Você não aprende nada copiando código de terceiros nem pedindo a outra pessoa que faça o trabalho por você. Se a cópia for detectada, sua nota será zerada e os professores serão informados para que as devidas providências sejam tomadas.

HAVE FUN!!!