Trabalho Prático 2 em Algoritmos 1

Gustavo Freitas Cunha

Matrícula: 2020054498

Departamento de Ciência da Computação - Instituto de Ciência Exatas -
Universidade Federal de Minas Gerais
Belo Horizonte - MG - Brasil

gustavocunha@dcc.ufmg.br

1. Modelagem

Para resolução do problema proposto, a malha rodoviária é aqui modelada como um grafo G = (V, E). Cada cidade é um nó deste grafo e cada rodovia que sai de uma cidade u ∈ V até uma cidade v ∈ V é uma aresta direcionada (u, v) ∈ E de peso w, onde, conforme definido no enunciado, w é a capacidade da rodovia. Neste caso, estamos considerando, a cada aresta, um único sentido da rodovia. Note que uma rodovia de "mão-dupla" será representada por duas arestas, com sentidos diferentes e pesos que também podem (ou não) ser diferentes.

O problema proposto consiste em encontrar um caminho entre uma origem e um destino de forma que seu **gargalo** (a aresta de menor peso no caminho) seja máximo, já que a quantidade de suprimentos a ser transportada da cidade de origem para a cidade de destino será limitada a tal gargalo, que deverá ser máximo, pela definição do problema. A solução proposta encontra todos os caminhos possíveis entre a origem e o destino dados, utilizando uma busca em profundidade (DFS) adaptada[1]. A cada caminho encontrado pela DFS entre as duas cidades (nós) de entrada, faz-se o cálculo de seu gargalo, verificando a aresta de menor peso naquele caminho e o valor do gargalo é armazenado em um deque[2]. Por fim, através de uma busca linear, encontra-se o valor do maior gargalo armazenado neste deque, que é retornado.

Note que uma cidade, ou nó, é representada por um único inteiro de 1 a N (**onde N é o número de cidades, uma variável de entrada do problema**), subtraído de 1. A cidade 2 é representada pelo inteiro 1 e a cidade 1 é representada pelo inteiro 0, por exemplo. Essa subtração é necessária pois, conforme será detalhado nas próximas seções, lidaremos com uma matriz de adjacências, cuja primeira posição tem índice 0. A adaptação do índice das cidades é feita logo no início, na função main, onde as consultas são processadas, já enviando para a função encontraCaminhoMaximo (mais detalhes na seção 3) os **índices matriciais** (índice da cidade subtraído de 1) das cidades de origem e destino.

[1] Mais detalhes sobre os algoritmos utilizados na seção 3.
[2] Mais detalhes sobre estruturas de dados utilizadas na seção 2.

2. Estruturas de Dados

A representação computacional do grafo G, definido na seção anterior, é uma matriz de adjacências, denominada malha, N por N. Nessa matriz, teremos uma célula malha[i][j] = k se há uma aresta (i, j) ∈ E saindo de i e chegando em j, onde $0 \leq i, j < N$, com peso $k \geq 0$. Caso k seja igual a zero, tal aresta (ou rodovia) não existe.

Outra estrutura de dados utilizada foi o deque, biblioteca padrão de C++, é semelhante à uma lista encadeada e permite inserções e remoções no início e no fim. Esta estrutura foi utilizada na solução com duas finalidades:

1 - Armazenar caminhos: sequência ordenada de índices matriciais de cidades que compõem um caminho.

2 - Armazenar gargalos: sequência, na ordem em que são calculados, de gargalos dos caminhos encontrados pela DFS entre as cidades de origem e destino.

3. Algoritmos

- Busca em Profundidade (DFS) Adaptada

Após o processamento da malha rodoviária de entrada e a inicialização da matriz de adjacências que representará o grafo segundo o qual a malha foi modelada, é chamada a função encontraCaminhoMaximo, para cada consulta. Esta função, por sua vez, chama a função DFS, que nada mais é do que uma busca em profundidade que, em vez de encontrar um só caminho da origem até o destino recebidos, encontra TODOS os caminhos possíveis e calcula seus respectivos gargalos armazenando-os no vetor gargalos, recebido como referência. Segue seu pseudocódigo:

```
DFS(grafo, noOrigem, noDestino){
   C := sequência de nós que definem um caminho //inicialmente vazia
   G := lista dos gargalos dos caminhos encontrados
   if(origem = destino){
          Adiciona a C o gargalo de G
   }
   else{
          for each(nó n não explorado em grafo vizinho de noOrigem){
                DFS(grafo, n, noDestino)
          }
   }
   Marca o último nó de C como não explorado
```

```
    Remove o último nó de C
}
```

Após esta função ter sua execução finalizada, e retornar para encontraCaminhoMaximo, é encontrado e impresso na saída padrão o maior valor do deque que armazena os gargalos dos caminhos encontrados.

4. Análise de Complexidade

A solução implementada possui complexidade assintótica $O(QN^2)$, sendo N o número de cidades, M o número de rodovias e Q o número de consultas.

A função DFS claramente apresenta complexidade $O(N^2)$, já que encontra todos os possíveis caminhos entre origem e destino. A partir de cada vértice v de origem em um determinado estágio da recursão, existem até N vértices que podem ser visitados a partir dele. Já que temos N vértices, teremos complexidade assintótica polinomial de grau 2 em N.

A função encontraCaminhoMaximo é dominada assintoticamente por sua chamada à DFS - $O(N^2)$.

A função main possui, antes da chamada a encontraCaminhoMaximo, domínio assintótico de $O(N^2)$, obtido na inicialização da matriz de adjacências, onde todos as células são inicializadas com zero, representando um grafo (ou uma malha de rodovias, no caso), sem arestas (ou sem rodovias, no caso). Em seguida, encontraCaminhoMaximo é chamado Q vezes para execução das Q consultas, e, logo, obtemos $Q * O(N^2) = O(QN^2)$.

Note que o número M de rodovias não aparece na complexidade assintótica final porque ele é dominado à todo tempo pelas outras variáveis, apesar de também haver instruções cuja complexidade de tempo dependa de M, como o processamento inicial das rodovias, quando são inseridas na matriz de adjacências, na função main.

5. Instruções para Compilação e Execução

O arquivo único, main.cpp, de código desta solução está no diretório raiz TP. Para compilação e execução, siga os passos a seguir:

Primeiro, **com o terminal no diretório TP**, execute o seguinte comando, para compilação do código fonte e geração do arquivo em código de máquina, de nome tp02:

```
g++ -std=c++17 -lstdc++fs -Wall main.cpp -o tp02
```

Na sequência, com o terminal no mesmo diretório, entre com o comando a seguir, para execução:

```
./tp02 < <nome do arquivo de entrada>.txt
```

É necessário que o arquivo de entrada também esteja no diretório TP, caso contrário, é preciso indicar seu caminho no terminal.

Conforme solicitado, os resultados serão impressos na saída padrão (stdout).

Por motivos de possível incompatibilidade na arquitetura de computadores, o arquivo executável gerado em meu computador não foi entregue, sendo gerado por você ao compilar o código.