Trabalho Prático 1

Aritmofobia

Matheus Guimarães Couto de Melo Afonso
Matrícula: 2021039450

Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte – MG – Brasil

matheusgcma@ufmg.br

1. Introdução

O trabalho prático 1 consiste em um problema no qual, dado um conjunto de cidades C (com n cidades) e um conjunto de estradas E (com m estradas) entre elas, deve-se encontrar o menor caminho entre as cidades 1 e n. Contudo, devem ser respeitadas duas restrições:

1. O caminho deve passar apenas por estradas cujo comprimento seja par;
2. O caminho deve passar por um número par de estradas.

2. Modelagem

O problema foi modelado utilizando um grafo ponderado e não-direcionado, no qual as cidades correspondem aos vértices e as estradas correspondem às arestas.

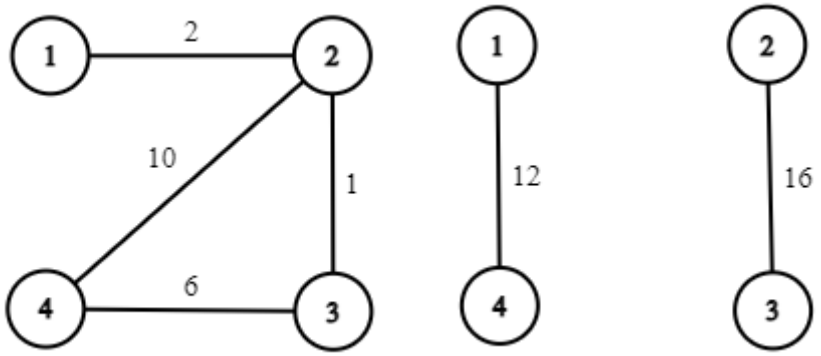
Para respeitar a primeira restrição, basta ignorar as entradas de arestas cujos comprimentos sejam ímpares e construir o grafo utilizando apenas as arestas pares. Contudo, a segunda restrição é um pouco mais trabalhosa de ser resolvida.

Inicialmente, a solução pensada foi encontrar todos os caminhos possíveis entre a origem e o destino e ver qual deles era o menor. Porém, a complexidade dessa estratégia é exponencial ($O(m * 2^n)$), o que a torna inviável em situações onde a entrada é muito grande.

Visto o problema na primeira implementação, foi necessário buscar uma outra alternativa em que o custo fosse menor. A solução encontrada foi realizar uma transformação no grafo de entrada, de forma que para qualquer par de arestas (u_1, v_1) de peso w_1 e (u_2, v_2) de peso w_2 no grafo original é criada uma aresta (u_1, v_2) de peso $w_1 + w_2$ no grafo transformado. Isso garante o cumprimento da segunda

restrição, uma vez que toda aresta do segundo grafo representa um par de arestas do grafo original.

Os grafos abaixo exemplificam as transformações feitas, baseado na entrada do caso de teste 1 fornecido pela professora.



Após a transformação, é executado o algoritmo de Dijkstra no grafo transformado, com origem no vértice 1, que encontra o menor caminho até todos os vértices do grafo partindo do vértice de origem. O Dijkstra constrói um vetor em que cada posição k representa o menor caminho entre 1 e k. Sendo assim, o menor caminho entre as cidades desejadas é dado pela posição $k - 1$ desse vetor.

3. Análise de Complexidade

O grafo é armazenado em uma lista de adjacências, em oposição a uma matriz. A complexidade de espaço de uma lista de adjacências é $O(n \log m)$.

No que diz respeito à complexidade de tempo, ela será analisada por cada etapa do código:

1. Entrada das arestas: $O(m)$.
2. Transformação do grafo: $O(n \log^2 m)$. A transformação itera pelos n vértices do grafo. A partir da lista de adjacências de cada vértice, passa pelas arestas de cada um $\log m$ vezes, e em cada aresta (u, v) encontrada na lista do nó u são executadas mais $\log m$ iterações pelas arestas do nó v.
3. Dijkstra: $O(n \log p)$. O algoritmo de Dijkstra tem complexidade $O(|V| \log |E|)$. Após a transformação, o número de vértices n não se altera, mas o número de arestas m está sujeito a aumentar ou diminuir, portanto a complexidade é dada em função do novo número de arestas p.

Portanto, a complexidade total é dada por $O(n \log^2 m)$, uma vez que o processo de transformar o grafo é o mais custoso.