UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA (DAINF)
CURSO ENGENHARIA DE COMPUTAÇÃO

ANNA CAROLINE DE OLIVEIRA SOUSA

O PROBLEMA DO MENOR CAMINHO - ALGORITMO DE DIJKSTRA

TRABALHO PARA DISCIPLINA ESTRUTURA DE DADOS 2

Curitiba, 2021

Resumo – O presente artigo possui como objetivo abordar “O problema do menor caminho (algoritmo de Dijkstra)” bem como a sua resolução, descrição das escolhas seguidas e seus principais desafios encontrados durante a execução do mesmo.

Palavras-chave: grafo, menor caminho, algoritmo, peso.

Descrição do problema

Os grafos são estruturas de dados presentes na ciência da computação, e os algoritmos para trabalhar com eles são fundamentais e de grande aplicação em diversas áreas. Uma de suas principais aplicações é o cálculo do caminho mínimo, o qual é muito utilizado em GPS e afins. O cálculo do caminho mínimo consiste em calcular o caminho mais curto entre os vértices de um grafo, sendo que cada aresta tem um comprimento ou “peso” associado.

A representação de um grafo pode seguir duas maneiras padrões distintas, sendo elas: como uma coleção de listas de adjacências ou como uma matriz de adjacências. Aliás, quando é necessário implementar um algoritmo para o menor caminho é recomendável seguir o conceito de matriz, pois é possível saber mais rápido se há ou não uma aresta conectando dois vértices dados, ou outra aplicação é quando o grafo é denso. Entretanto, é necessário ressaltar que o custo computacional para a matriz é maior quando comparado com a lista de adjacências.

Um algoritmo específico para este problema é o algoritmo de Dijkstra, o qual supõe que todos os “pesos” de arestas no grafo de entrada são não negativos, sendo que o grafo pode ser ou não dirigido. Além disso, o algoritmo considera um conjunto de menores caminhos, o mesmo é iniciado em um determinado vértice e em cada passo é buscado nas adjacências dos vértices aquele que dispõe de menor distância relativa, o processo é repetido até que todos os vértices alcançados sejam analisados. Outrossim, a complexidade apresentada pelo algoritmo no pior caso é da ordem de $O(|E| + |V| \log|V|)$ – sendo V e E respectivamente o número de vértices e arestas, entretanto é necessário ressaltar que quando comparado a outros algoritmos do mesmo princípio o tempo de execução é inferior, contudo, quando bem implementado.

Ademais, o Dijkstra é considerado um algoritmo guloso. Os algoritmos gulosos estão relacionados a problemas de otimização, onde funcionam através de uma sequência de passos, dispondo de um conjunto de opções. Cada escolha é realizada com base no seguinte princípio: aquela que parece ser a melhor opção no momento, na esperança que a escolha condicione uma ótima solução para a situação global do problema. Não obstante, é necessário ressaltar que nem sempre é possível produzir soluções ótimas, mas é uma excelente solução para diversos problemas.

Como o tópico escolhido aborda os conceitos vistos na disciplina

O conhecimento de linguagens de programação é de fundamental importância para os programadores, entretanto, apenas isto não capacita suficientemente e de maneira adequada. Tendo isto em vista, é imprescindível dispor de conhecimentos de modo a organizar estruturalmente os dados bem como tornar mais ágil a busca, ou seja, a estruturação dos dados é essencial. Logo, é possível afirmar que para desenvolver e projetar o software que possui como pressuposto encontrar o menor caminho, a presente matéria é indispensável.

Assim como visto no decorrer da disciplina, os grafos são estruturas abstratas que podem modelar problemas do mundo real. Ademais, é possível verificar que um algoritmo de um determinado problema de grafos se aplica e pode resolver diversos problemas do cotidiano. Aliás, o problema do menor caminho é de fato aplicado para diversos problemas reais, como, por exemplo: transporte, GPS, sites de mapeamento web, entre outros.

Deste modo, usando como base a teoria dos grafos é possível encontrar ramificações da mesma e ampliar o seu uso. Pois, o princípio seguido diz a respeito das relações que existem entre os objetos – vértices – de um determinado conjunto, a depender de sua aplicação, podem e devem haver distinções, como: orientação de arestas, vértices serem ou não ligados a si mesmos, disporem de "pesos", entre outras características. Logo, é factível afirmar que o algoritmo apresentado é uma ramificação da teoria básica de grafos, a qual condiciona o suporte necessário à estruturação do algoritmo e suas características básicas.

Além disso, é possível realizar um comparativo e afirmar que o algoritmo apresentado possui semelhança com outros dois: o algoritmo de busca em largura e o algoritmo de Prim. Primeiramente, é semelhante ao de busca em largura – o qual foi estudado no decorrer da disciplina – devido ao fato de que o conjunto de vértices de menor peso correspondem ao conjunto de vértices pretos em uma busca em largura. Enquanto, no algoritmo de Prim – calcula árvores espalhadas mínimas – devido ambos disporem de fila de prioridade mínima – quando selecionado este caso para o Dijkstra – para encontrar o vértice mais leve.

Solução encontrada

Quando pensamos computacionalmente é fácil perceber que o problema do menor caminho é solucionado com o auxílio de uma matriz – onde seguindo suas características padrão – dispõe de linhas e colunas, conforme a quantidade de arestas. Sendo assim, a principal estratégia encontrada para resolução do problema apresentado foi o seu fracionamento em partes menores, com objetivo de ter um maior controle do funcionamento do programa bem como sua garantia de qualidade. Portanto, o primeiro ponto desenvolvido foi a constituição de uma matriz de tamanho apropriado.

Além disso, é necessário ressaltar que o grafo utilizado não é dirigido, esta escolha foi realizada com base na percepção de que o usuário tem a possibilidade de partir de qualquer ponto do grafo, e não apenas em pontos com rotas pré-determinadas. Assim, é fácil perceber que as arestas (u,v) e (v,u) são iguais e com isso é possível afirmar que a matriz A – a qual representa o grafo – é equivalente a sua matriz transposta ($A = A_T$). Deste modo, para a constituição da matriz do grafo, são recebidos apenas uma entrada e a mesma é refletida nas coordenadas da transposta. Com isso, é possível afirmar que o principal objetivo é deixar um código mais compacto e com o mesmo funcionamento, dispondo do auxílio das funções *insereVertice(Grafo g, int v, int w, int peso)* e *removeVertice(Grafo g, int v, int w)*.

O conceito de matriz de adjacência utilizado auxilia a identificar se há ou não uma ligação entre os vértices apresentados – caso haja, será representado pelo número referente ao peso e caso contrário pelo zero em sua respectiva posição nas arestas correspondentes na matriz. Logo, isto auxilia a encontrar os caminhos válidos entre um vértice e outro, e sua diagonal principal é composta por zeros devido não ser necessário ou até mesmo uma rota válida, para o problema apresentado, sair de um vértice tendo como destino ele mesmo.

A execução do programa é baseada em uma estimativa de caminhos mais curtos. Com o auxílio da função *menorCusto(Grafo g, int origem, int destino)* é possível encontrar a menor aferição de "pesos" entre as arestas que ligam os vértices – o qual, neste caso representa a distância entre cada ponto apresentado no mapa disponíveis no arquivo em pdf. Enquanto, que com base na função *menorCaminho(Grafo g, int s, int t, int *caminho)* é encontrado os caminhos que podem ser percorridos.

Entretanto, é necessário salientar que quando inicializado um grafo o seu predecessor é inicialmente atribuído com o valor de -1. Aliás, a estimativa do caminho mais curto é inicialmente dada por infinito – porém, no programa apresentado este valor é aferido com o auxílio da biblioteca <limits.h> utilizando o INT_MAX como valor inicial – excetuando o vértice inicial a ele mesmo, onde neste caso é igual a zero. Sendo que esta estimativa no decorrer do programa acaba tornando-se a distância mais curta de caminho encontrada durante sua execução.

O princípio do algoritmo consiste em realizar estimativas de caminho, sendo que enquanto houver caminhos a serem analisados o denominado como menor inicialmente poderá ser trocado, caso se encontre algum menor. Este processo de trocar o caminho, por outra menor estimativa, é denominado relaxamento das arestas vizinhas. Outrossim, o caminho apresentado como resultado, consiste no menor dentre algumas possibilidades, conforme é a proposta do algoritmo.

Descrição dos testes

A constituição de um programa não se limita apenas a escrever algumas linhas de código, é necessário assegurar que o mesmo dispõe de qualidade e que funciona para situações distintas. Com o intuito de promover um programa apto, alguns testes foram realizados durante e após o seu desenvolvimento.

Durante a constituição da base do programa, o mesmo foi testado para um grafo com pesos simples e não direcionados. O teste foi realizado com uma versão mais simplificada do que o grafo apresentado como "zona sul", com o intuito de ter um maior controle do seu funcionamento e garantir que o resultado fosse conforme o esperado. Com base nisto, o mesmo foi expandido para outros grafos com maiores dimensões, de modo a alcançar o objetivo inicialmente proposto – um mapa onde selecionado o ponto de partida e destino apresenta o melhor trajeto.

Após realizar o teste exposto acima, outros grafos foram implementados, bem como um menu de seleção. Os novos grafos adicionados correspondem aos mapas: zona norte, zona sul e estados – conforme apresentado no pdf de mapas – com pesos e arestas diferentes. Aliás, com base nisto, é possível verificar a eficiência do grafo e do algoritmo proposto, pois o cálculo é contabilizado para cada situação, conforme a escolha do usuário.

Por fim, é possível selecionar posições distintas de início e fim do caminho desejado e na maioria dos caminhos foram obtidos bons resultados. Ademais, algumas otimizações foram realizadas com o intuito de minimizar o custo computacional bem como aumentar a eficiência do código.

Referência utilizadas no desenvolvimento

[A] CORMEN, LEISERSON, RIVEST e STEIN. Algoritmos: Teoria e Prática. Editora Elsevier, 6º tiragem.

[B] CELES, CERQUEIRA e RANGEL. Introdução a Estrutura de Dados. Editora Campus, 11º edição.

[C] FEOFILOFF, Paulo. IME USP, Curitiba – Pr, Brasil,
Acessado em 01/08/2021, às 15:40h.
https://www.ime.usp.br/~pf/algoritmos_para_grafos/

[D] ROMAN, Norton. Estrutura de dados - Aula 28 - Grafos - Algoritmo de Dijkstra, Youtube, 20/04/2017. Disponível em: <https://www.youtube.com/watch?v=ovkITlgyJ2s&t=999s>. Acessado em 04/04/2021, às 10:20h.