Trabalho Prático I - Algoritmos I

Augusto Guerra de Lima
2022101086

Departamento de Ciência da Computação; Universidade Federal de Minas Gerais
Belo Horizonte, Minas Gerais; Primavera de 2024
augustoguerra@dcc.ufmg.br

1 Introdução

Este trabalho aborda um problema algorítmico envolvendo centros urbanos conectados por estradas. O problema é subdividido em três subproblemas principais: a determinação de uma capital entre os centros urbanos, a definição de batalhões secundários (quando possível) e o estabelecimento de patrulhamentos pelas estradas (quando aplicável). Como será demonstrado ao longo do texto, é possível modelar os centros urbanos e suas conexões como um grafo direcionado, o que permite o uso de algoritmos clássicos em teoria dos grafos para alcançar as soluções desejadas.

Na segunda seção, será apresentada a modelagem utilizada para o problema, enquanto a terceira seção abordará a solução com foco nos algoritmos aplicados para computar as respostas. A quarta seção incluirá uma análise da complexidade assintótica do programa desenvolvido, e, por fim, o texto será concluído com as considerações finais e as referências utilizadas.

2 Modelagem

Em uma primeira análise, observa-se que o problema pode ser modelado como um **grafo direcionado não ponderado** $G := (V, E)$, em que o conjunto de vértices V representa os **centros urbanos**, e o conjunto de arestas direcionadas E corresponde às **estradas** que conectam esses centros.

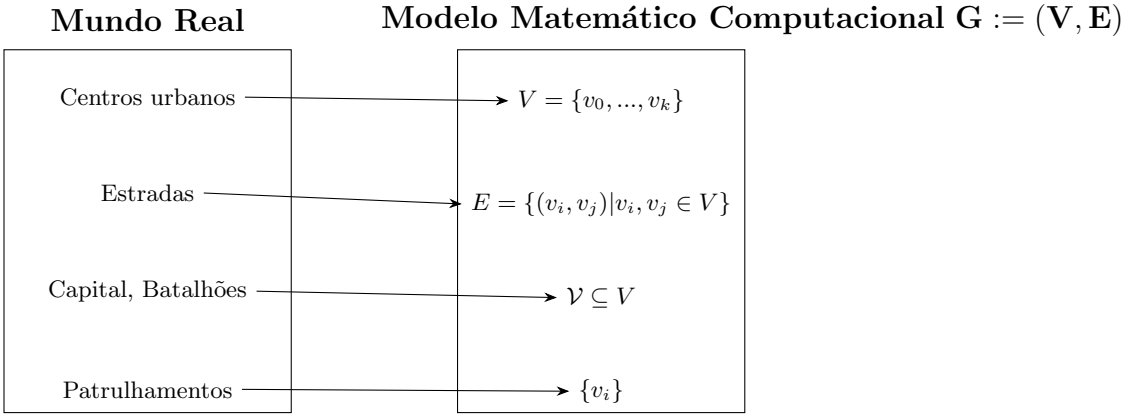
Contudo, conforme mencionado na primeira seção, é necessário identificar uma **capital** e eventuais **batalhões secundários**, ambos definidos como centros urbanos — ou seja, vértices — que recebem tais designações. Para isso, faz-se necessário os definir.

Definição 2.1: A **capital** é o vértice em um grafo direcionado que possui alcance a todos os demais vértices, minimizando a soma das distâncias necessárias para alcançá-los.

Definição 2.2: Cada componente fortemente conexa do grafo direcionado que não inclui a capital possui *exatamente um* **batalhão**, definido como um vértice dentro da componente que possui uma aresta ponte incidente, menor distância da capital até ele e menor soma de distâncias entre os outros vértices de sua componente fortmente conexa.

Outro conceito relevante, parte do terceiro subproblema algorítmico é uma **patrulha**.

Definição 2.3: Uma **patrulha** é um circuito, não necessariamente simples, dentro de uma componente fortemente conexa que se inicia a partir de um **batalhão secundário** e cobre exatamente todas as arestas da componente.



3 Solução

3.1 Capital

Para determinar o vértice capital, foi utilizado o algoritmo *Breadth-First Search* (BFS). Este algoritmo foi escolhido devido à sua capacidade de computar as menores distâncias entre o vértice fonte (`s`) e os demais vértices em um grafo não ponderado.

A execução da função `bfs`, que mapeia um vértice fonte para a soma das distâncias até os demais vértices, é realizada para cada vértice $v \in V$. O vértice capital é então determinado como aquele que minimiza tal soma. Vértices que não conseguem alcançar todos os outros vértices têm suas distâncias mapeadas para infinito (`INF`).

Foi garantido que existe ao menos um vértice que satisfaz as condições necessárias para ser considerado o vértice capital.

Algorithm 1 Descrição do algoritmo para determinar a capital

Domínio: Grafo direcionado não ponderado.
Imagem: Capital do grafo direcionado
`min` $\leftarrow \infty$
for cada $v \in V$ **do**
    Compute a soma $\sum_{v_i \in V} \delta_{v_i}$ das distancias para todos os outros vértices a partir de v com BFS
    if $\sum_{v_i \in V} \delta_{v_i} <$ `min` **then**
        `capital` $\leftarrow v$
        `min` $\leftarrow \sum_{v_i \in V} \delta_{v_i}$
    end if
end for

3.2 Batalhões secundários

Para determinar os batalhões secundários, foi utilizado o algoritmo de *Kosaraju*, uma aplicação de *Depth-First Search* (DFS), o qual permite identificar as componentes fortemente conexas em um grafo direcionado. Conforme mencionado anteriormente, cada uma das componentes fortemente conexas que não contém a capital possui *exatamente um* batalhão secundário.

O primeiro passo consistiu na execução do método `kosaraju`, o qual delimitou as componentes fortemente conexas do grafo direcionado.

Após a determinação das componentes fortemente conexas, ou *strongly connected components* (SCC), para cada uma delas foi atribuído *exatamente um* batalhão secundário, de acordo com dois critérios, a saber:

(1) Menor distância do vértice capital até o vértice;

(2) Minimização da soma das distâncias entre todos os demais vértices da componente fortemente conexa (SCC).

A descrição do algoritmo em pseudocódigo é apresentada a seguir:

Algorithm 2 Descrição do algoritmo para determinar batalhões secundários

Domínio: Grafo direcionado não ponderado.
Imagem: Batalhões secundários determinados para cada SCC.
Execute o método `kosaraju` para identificar as SCCs.
for cada SCC **do**
    Tome o vértice de menor distancia a partir da capital.
    if Vértices empataram **then**
        Compute a soma para os outros vértices da SCC com uma BFS.
        Tome o que minimiza a soma dentre os outros vértices da SCC.
    end if
end for

É importante ressaltar que as distâncias a partir da capital já haviam sido previamente computadas durante o processo de escolha da capital. A minimização das somas das distâncias é um critério utilizado apenas quando ocorre empate entre vértices *candidatos* no primeiro critério. A função responsável pelo procedimento de desempate é a `bfs_criteria`, a qual calcula a soma das distâncias de um vértice candidato a batalhão secundário para os demais vértices de sua respectiva componente fortemente conexa.

3.3 Patrulhamentos

O problema de patrulhamento consiste em determinar, para cada componente fortemente conexa, um circuito que se inicia a partir do vértice batalhão, percorre todas as estradas - arestas - e retorna ao batalhão. Este circuito, contudo, não é necessariamente *euleriano*, ou seja, algumas arestas podem ser revisitadas, caso preciso.

Para a resolução desse problema, o algoritmo requer uma etapa de pré-computação, realizada com o auxílio de duas buscas em largura (BFSs). A primeira BFS é executada no grafo $G := (V, E)$ e identifica, para cada vértice da componente fortemente conexa do batalhão, o seu vértice pai na árvore gerada, bem como a aresta pela qual ele é alcançado. A segunda BFS é executada no grafo transposto $G^T := (V, E^T)$ e realiza um processo análogo: para cada vértice, determina seus vértices filhos e as respectivas arestas pelas quais eles o alcançam.

Com isso, é suficiente manter uma lista das arestas da componente fortemente conexa que necessitam ser visitadas, atribuindo a cada aresta um índice e identificando os vértices (u, v) que a compõem. Dessa forma, para cada aresta, constrói-se o caminho utilizando o vetor de pais e, para retornar ao batalhão, utiliza-se a estrutura dos filhos. As arestas visitadas durante o processo são marcadas, garantindo assim que todas as arestas da componente fortemente conexa sejam percorridas.

A seguir, apresenta-se o pseudocódigo:

Algorithm 3 Descrição do algoritmo para descrever os patrulhamentos

```
1: Domínio: Componente fortemente conexa que contém o batalhão B.
2: Imagem: Patrulhamento na SCC a partir de B.
3: Execute uma BFS em SCC ⊂ G a partir de B e armazene os pais e arestas dos vértices.
4: Execute uma BFS em SCC^T ⊂ G^T a partir de B e armazene os filhos e arestas dos vértices.
5: Construa uma lista de arestas da SCC a serem visitadas.
6: for cada e aresta na lista de arestas do
7:     if e não foi visitada then
8:         construa o caminho até e marcando as arestas do caminho como visitadas, usando a lista de pais.
9:         construa o caminho até B marcando as arestas do caminho como visitadas, usando a lista de filhos.
10:        armazene os vértices dos caminhos.
11:    end if
12: end for
13: imprima os vértices do circuito.
```

4 Análise de complexidade

4.1 Capital

Proposição 4.1: A complexidade assintótica do *tempo de execução* para determinar o vértice capital pertence a $\mathbf{O}(|V| \cdot (|V| + |E|))$.

Prova: Para determinar a capital para todos os vértices $v \in V$, é realizada uma chamada ao algoritmo `bfs`. Em cada execução do algoritmo *Breadth-First Search* (BFS), cada vértice é visitado tantas vezes quanto o seu grau de entrada. Pelo Teorema do Aperto de Mãos, sabemos que $\sum_{v \in V} d(v) \in \mathbf{O}(|E|)$. A inicialização dos vértices, assim como o cálculo da soma das distâncias, possui complexidade pertencente ao conjunto $\Theta(|V|)$. Dessa forma, a complexidade total do algoritmo é dada por $\mathbf{O}(|V| + |E|)$.

Como a chamada é realizada para cada um dos vértices $v \in V$, a complexidade total pertence ao conjunto $\mathbf{O}(|V| \cdot (|V| + |E|))$. Para K_n ou um grafo denso, a complexidade assintótica torna-se um polinômio de grau 3, embora com constantes superiores às do algoritmo de *Floyd-Warshall*. ■

4.2 Batalhões secundários

Proposição 4.2: A complexidade assintótica do *tempo de execução* para designar todos os batalhões secundários pertence a $\mathbf{O}(c \cdot (|V| + |E|))$ onde $c < |V|$ é a quantidade de vértices candidatos em uma componente fortemente conexa caso exista empate no primeiro critério.

Prova: A primeira etapa do método envolve uma chamada ao algoritmo de *Kosaraju*, que realiza duas execuções do algoritmo *Depth-First Search* com complexidade $\mathbf{O}(|V| + |E|)$. Para a coloração do grafo e, consequentemente, para a separação em componentes fortemente conexas, temos complexidade constante para essas atribuições. A

construção do grafo transposto de G, denotado por $G^T := (V, E^T)$, possui complexidade $\Theta(|E|)$; assim, a complexidade de tempo de execução do algoritmo de *Kosaraju* pertence ao conjunto $\mathbf{O}(|V| + |E|)$.

Após a divisão do grafo, é necessário designar um batalhão secundário para cada componente fortemente conexa. Como as distâncias da capital aos vértices já foram pré-computadas, essa etapa possui complexidade $\mathbf{O}(|V|)$, caso não existam empates no primeiro critério. No entanto, em caso de empates, seja c a quantidade de vértices candidatos (geralmente $c \ll |V|$), o pior caso ocorre quando c está em uma componente fortemente conexa de cardinalidade muito próxima à do grafo original G. Nesse cenário, é necessário realizar uma chamada ao algoritmo `bfs_criteria` (BFS) para cada candidato, resultando em uma complexidade de $\mathbf{O}(c \cdot (|V| + |E|))$ no caso de empates. Caso contrário, a complexidade do algoritmo de *Kosaraju* domina assintoticamente. ■

4.3 Patrulhamentos

Proposição 4.3: A complexidade assintótica do *tempo de execução* para descrever os patrulhamentos aplicáveis pertence ao conjunto $\mathbf{O}(|V|^3)$.

Prova: Este é um limite assintótico superior. A execução das BFSs possui complexidade em $\mathbf{O}(|V| + |E|)$. Considerando que é preciso iterar sobre todas as arestas, $\mathbf{O}(|E|)$, em um grafo denso, $\mathbf{O}(|V|^2)$. Suponha que, para cada aresta, seja descrito um caminho $v_0, v_1, \ldots, v_k$, onde $v_i \in V$ para $i = 0, 1, \ldots, |V| - 1$. Dessa forma, a complexidade pertence a $\mathbf{O}(|V| \cdot |V|^2)$, ou seja, $\mathbf{O}(|V|^3)$. ■

4.4 Análise de complexidade espacial

Proposição 4.4: A complexidade *espacial* do programa desenvolvido considerando todos os algoritmos implementados pertence ao conjunto $\mathbf{O}(|V| + |E|)$.

Prova: A classe `graph_c` utiliza uma lista de adjacência padrão, cuja complexidade espacial esta em $\mathbf{O}(|\mathbf{V}| + |\mathbf{E}|)$. Além disso, incorpora outras estruturas, como vetores para armazenar a coloração dos vértices, com complexidade $\mathbf{O}(|V|)$, e uma lista de vértices para cada componente fortemente conexa, com complexidade $\mathbf{O}(|E|)$, além de índices para as arestas. Os demais algoritmos baseiam-se em métodos de travessia, como BFS e DFS, assim como no algoritmo de *Kosaraju*; todos esses algoritmos possuem estruturas de controle com complexidade $\mathbf{O}(|V| + |E|)$. A complexidade de espaço da classe pertence ao conjunto $\mathbf{O}(|V| + |E|)$ portanto. ■

5 Considerações finais

Neste trabalho, foi solucionado o problema de definir, entre os centros urbanos, qual será designado como capital e quais serão classificados como batalhões secundários. Além disso, abordou-se o problema de organizar as patrulhas em uma área específica de um batalhão. A modelagem matemática e computacional do problema foi realizada por meio de grafos, objetos matemáticos de suma relevância.

O problema foi dividido em três subproblemas. O primeiro, de natureza mais simples, serviu como base inicial; o segundo apresentou maior complexidade em sua implementação, com decisões e critérios de decidibilidade mais elaborados. Já o último, notavelmente mais desafiador, abre uma discussão sobre uma classe de problemas em grafos relacionada à identificação de circuitos e caminhos, bem como à minimização desses percursos. Essa é uma área de grande riqueza teórica, que envolve discussões sobre classes de complexidade algorítmica.

Por fim, os algoritmos clássicos de grafos para travessia e determinação de componentes fortemente conexas mostraram-se robustos e apresentaram uma característica notável: sua grande *adaptabilidade*. Esse aspecto é amplamente presente nos algoritmos para grafos, pois, uma vez definido o algoritmo canônico, ele pode ser personalizado, modificado e reinterpretado de acordo com as necessidades específicas do problema, de maneira bastante flexível. Essa adaptabilidade foi exercitada durante o desenvolvimento deste trabalho.

6 Referências

[1] BALAKRISHNAN, V.K. *"Indroductory Discrete Mathematics"*. 2nd ed. Garden City, New York: Dover Publications, 1991.

[2] CORMEN, Thomas H. et al. *"Introduction to Algorithms"*. 3rd ed. Cambridge, Massachusetts: MIT Press, 2009.