Trabalho Prático 1
Algoritmos I

Vitor Augusto Reis Azevedo
2023002294

Universidade Federal de Minas Gerais
Belo Horizonte - MG - Brasil

vitoraugreis@dcc.ufmg.br

---

1. Introdução

O problema está ambientado em um império sob ameaça de invasão, e o imperador decidiu implementar novas medidas de segurança para garantir uma resposta rápida. Para isso, uma equipe de cartógrafos foi contratada para reunir informações essenciais para o conselho de segurança. Eles definiram que um estado seria composto por uma capital, centros urbanos e estradas de mão única. No entanto, ao realizar esse trabalho manualmente, perceberam que era muito trabalhoso e estabeleceram três tarefas a serem resolvidas computacionalmente:

1. Definir a capital: A capital deve ser um centro urbano de onde as tropas podem alcançar todos os outros centros de maneira eficiente.

2. Identificar centros que precisam de batalhões secundários: Caso as tropas não consigam retornar à capital devido a estradas de mão única, é necessário construir batalhões secundários para apoio.

3. Planejar rotas de patrulhamento: Em situações de aumento de tensões, é preciso definir quantas patrulhas são possíveis e suas rotas, começando e terminando em cada batalhão.

2. Modelagem

O problema descrito foi modelado como um grafo direcionado não ponderado. Cada vértice do grafo representa um centro urbano do estado, enquanto cada aresta direcionada simboliza uma estrada de sentido único entre dois centros. Não há necessidade de atribuir pesos às arestas, pois todas as estradas são consideradas equivalentes. Para a solução, leva-se em conta apenas o número de estradas percorridas, e não a distância entre os centros.

A resolução do problema foi dividida nas seguintes etapas, utilizando algoritmos específicos:

1. Para determinar a capital, foram realizadas buscas em largura (BFS) a partir dos vértices do grafo, verificando qual vértice permite alcançar todos os demais centros urbanos de maneira eficiente.

2. Para definir quais centros urbanos necessitam de batalhões secundários, foi aplicado o algoritmo de Kosaraju para encontrar as componentes fortemente conectadas do grafo.
3. Na definição das rotas de patrulhamento, foram utilizados o algoritmo húngaro para resolver problemas de emparelhamento, o algoritmo de Hierholzer para gerar a rota a partir de um grafo euleriano, e buscas em largura (BFS).

3. Solução

No programa principal, há a leitura de todas as estradas e centros urbanos que constituem o estado. Cada uma das estradas é inserida no tipo abstrato de dados (TAD) Grafo, criado para representar o estado. Este TAD não possui a capacidade de armazenar os nomes dos centros urbanos, assim cada centro é representado por um índice e na impressão das informações, no programa principal, os nomes dos centros são recuperados a partir destes índices. Após criação do grafo, com a adição de todos os centros (vértices) e estradas (arestas), podemos dar início ao algoritmo para resolver os problemas.

3.1. Definição da Capital

Para definir a capital, utilizamos a Busca em Largura (BFS). A BFS é um método eficiente para explorar grafos, partindo de um vértice inicial e visitando todos os seus vizinhos antes de avançar para os vizinhos desses vizinhos, garantindo que as distâncias calculadas sejam sempre as menores possíveis em grafos não ponderados. Nesse contexto, a BFS tem dois objetivos principais: calcular as distâncias do vértice inicial até todos os outros centros urbanos, armazenando essas distâncias em um vetor, e somar essas distâncias para obter o valor total, que será utilizado na decisão sobre a capital.

Outro elemento crucial no algoritmo é um vetor que indica quais vértices foram visitados durante a execução. Inicialmente, a BFS é usada para calcular as distâncias individuais entre o vértice candidato a capital e todos os outros. Porém, consideramos esse vértice como uma candidata válida somente se, e apenas se, ele permitir alcançar todos os demais centros urbanos - condição verificada pelo vetor de vértices visitados.

Após confirmar essa condição, soma-se as distâncias da candidata a capital a todos os outros centros. Se essa soma for menor do que a soma obtida para qualquer outra candidata avaliada anteriormente, o vértice atual passa a ser a capital provisória.

Esse processo é repetido para todos os vértices do grafo. Ao final da execução, o centro urbano que minimiza a soma das distâncias para todos os demais será definido como a capital do estado.

3.2. Definição dos Batalhões Secundários

Para a definição dos batalhões secundários, devemos encontrar as componentes fortemente conectadas do grafo. Uma componente fortemente conectada é um subgrafo no qual todos os vértices são acessíveis uns aos outros, direta ou indiretamente. Ou seja, levando isso para o problema inicial, vamos identificar todos os conjuntos de centros urbanos nos quais é possível realizar a ida e a volta entre eles. Após encontrar esses

conjuntos, vamos definir qual centro urbano deve ter um batalhão secundário, com base no centro pertencente ao conjunto com a menor distância até a capital.

Para isso, utilizamos o algoritmo de Kosaraju. Este algoritmo realiza duas passagens de Busca em Profundidade (DFS) para identificar essas componentes. A DFS é um algoritmo de exploração de grafos que começa a partir de um vértice e explora o máximo possível ao longo de um caminho antes de retroceder para explorar outros caminhos. Isso é feito de forma recursiva ou utilizando uma pilha, visitando cada vértice apenas uma vez.

A primeira passagem da DFS ocorre no grafo original e tem como objetivo determinar a ordem de término dos vértices. Esta ordem será fundamental na segunda passagem, pois, no grafo transposto, ela garante que as buscas subsequentes explorem primeiro os vértices mais influentes de cada componente.

Após isso, o grafo é transposto, ou seja, todas as arestas têm seu sentido invertido. Isso é necessário porque, no grafo original, as conexões entre vértices são unidirecionais, e a transposição permite explorar essas conexões no sentido oposto para identificar os grupos de vértices mutuamente conectados.

Por fim, o algoritmo realiza a segunda passagem da DFS, desta vez no grafo transposto, seguindo a ordem dos vértices armazenados na pilha. A partir do vértice no topo da pilha, a DFS explora os vértices conectados na transposição, identificando cada componente fortemente conectada (CFC). Cada vez que uma nova DFS é iniciada, ela representa uma nova CFC, e os vértices alcançados durante essa busca são agrupados na mesma componente. Cada componente é armazenada em um vetor, que será utilizado na etapa final do processo.

Ao final do algoritmo de Kosaraju, todas as CFCs são definidas, e é realizada a verificação de qual vértice de cada componente possui a menor distância até a capital (essas distâncias estão armazenadas em um vetor, portanto, não é necessário outro algoritmo para isso). Esse vértice é então definido como um batalhão secundário.

3.3. Definição das Rotas de Patrulhamento

Para a definição das rotas, vamos considerar cada uma das componentes fortemente conectadas (CFC) encontradas no problema anterior. Se a CFC possui apenas um vértice, não há rota a ser feita; caso contrário, o algoritmo descrito a seguir é utilizado.

Para cada CFC, inicialmente armazenamos todos os vértices da componente em um conjunto e identificamos qual é o batalhão da CFC. Em seguida, geramos um subgrafo (lista de adjacência) que consiste apenas nos vértices e arestas pertencentes à CFC. Todo o processo será realizado sobre esse subgrafo, não mais sobre o grafo original.

Com o subgrafo gerado, uma função verifica o balanceamento de todos os vértices. O balanceamento é obtido pela operação:

$$balanço = n° \, arestas \, que \, saem \, do \, vértice - número \, de \, arestas \, que \, entram \, no \, vértice$$

Ou seja, um vértice está balanceado se, e somente se, seu balanço for igual a 0. O balanço de cada vértice é armazenado em um vetor. Esses balanços são importantes para verificar uma característica necessária para a criação da rota: o subgrafo deve ser Euleriano. Para que isso aconteça, todos os vértices devem ser balanceados. Caso exista um desbalanceamento, haverá métodos para corrigir isso.

Após calcular o balanço de cada vértice, verificamos se o subgrafo é Euleriano. Se for, não há necessidade de alterações e podemos avançar diretamente para a execução do algoritmo de Hierholzer.

O algoritmo de Hierholzer encontra um caminho ou circuito Euleriano em um grafo utilizando uma abordagem de busca em profundidade. Ele começa no batalhão da CFC, que inicialmente não possui arestas visitadas, e segue pelas arestas até não haver mais arestas disponíveis. Quando chega a um vértice sem arestas não percorridas, o circuito é fechado. Se houver vértices com arestas não exploradas, o algoritmo reinicia a busca e conecta os ciclos formados até completar o caminho ou circuito Euleriano.

Caso o grafo não seja Euleriano, precisamos balancear os vértices desbalanceados. Para isso, separaremos os vértices com balanço negativo dos com balanço positivo, criando novos caminhos no subgrafo entre vértices negativos e positivos. Esses novos caminhos duplicam algumas arestas entre os vértices analisados, mas mantêm o balanço dos vértices inalterado. Apenas os dois vértices analisados terão seus balanços ajustados: o vértice com balanço negativo terá seu balanço aumentado em um e o com balanço positivo terá seu balanço diminuído em um.

Para garantir a menor rota possível, utilizamos o algoritmo húngaro para emparelhar vértices negativos com positivos de forma a minimizar o número de estradas no caminho final. Para isso, executamos uma Busca em Largura (BFS) de todos os vértices com balanço negativo para todos os vértices com balanço positivo, construímos uma matriz com base no número de estradas entre os vértices e o algoritmo húngaro realiza o emparelhamento perfeito.

Após isso, adicionamos as arestas resultantes dos novos caminhos ao subgrafo, balanceando o grafo. Com o grafo balanceado, executamos o algoritmo de Hierholzer.
Ao final, teremos todas as rotas possíveis para o estado.

4. Análise de Complexidade

Considere n o número de vértices do grafo e m o número de arestas do grafo:

3.1. Definição da Capital

3.1.1. Complexidade de Tempo: Cada execução da BFS tem custo $O(n + m)$. Como a busca em largura é executada para todos os vértices do grafo temo que a complexidade final da definição da capital é $O(n(n + m))$.

3.1.2. Complexidade de Espaço: O espaço adicional para armazenar a fila, o vetor de vértices visitados e o vetor de distâncias tem complexidade $O(n)$, logo a complexidade de espaço final na definição da capital é $O(n)$.

3.2. Definição dos Batalhões Secundários

3.2.1. Complexidade de Tempo: O algoritmo de Kosaraju tem complexidade $O(n + m)$ e na seleção de todos os batalhões temos a complexidade de $O(n)$, logo a complexidade final na definição dos batalhões secundários é $O((n + m) + n) = O(n + m)$.

3.2.2. Complexidade de Espaço: Na transposição do grafo, necessitamos de espaço adicional para armazenar o grafo com complexidade $O(n + m)$ e na DFS temos a

necessidade do espaço para a pilha de ordem $O(n)$, ou seja, a complexidade final de espaço é $O((n + m) + n) = O(n + m)$.

3.3. Definição das Rotas de Patrulhamento

3.3.1. Complexidade de Tempo: A verificação do balanceamento de cada vértice tem complexidade $O(n)$, a transformação do grafo não euleriano em um grafo euleriano a partir do algoritmo húngaro tem complexidade (n^3) e o algoritmo de Hierholzer tem complexidade $O(m)$, logo a complexidade para a definição das rotas é $O(n^3 + n + m) = O(n^3)$.

3.3.2. Complexidade de Espaço: Para a geração do subgrafo da componente conexa é necessário um espaço da ordem $O(n + m)$ e para a matriz utilizada no algoritmo húngato é necessário um espaço de $O(n^2)$, assim a complexidade de espaço é $O(n + m + n^2) = O(n^2)$.

5. Considerações Finais

O trabalho uniu a teoria aprendida em sala de aula com a prática e foi uma grande oportunidade de aprendizado para mim. Particularmente, tive maior facilidade na resolução dos dois primeiros problemas propostos, devido à simplicidade dos algoritmos e ao contato prévio em sala de aula. No terceiro problema, houve uma dificuldade considerável, especialmente com o algoritmo húngaro, mas essa dificuldade contribuiu significativamente para o meu aprendizado.

6. Referências

Jussara Marques de Almeida. **Slides disponibilizados na disciplina de Algoritmos I**. Departamento de Ciência da Computação (UFMG), 2024, Belo Horizonte.

KLEINBERG, Jon. **Algorithm Design**. 3ª ed. revista e ampliada. 2011, Editora Cengage.

Site **GeeksForGeeks - Hierholzer's Algorithm for directed graph**. Disponível em https://www.geeksforgeeks.org/hierholzers-algorithm-directed-graph.

UNIVESP. **Pesquisa Operacional - Aula 18 - Modelo de Designação: Método Húngaro**. Youtube, 20 de abril de 2017. 19m43s. Disponível em: <https://youtu.be/Kue4UUxstoU?si=vlMBXoV2BoHc6boL>. Acesso em 15 de novembro de 2024.