Trabalho Prático 2
Algoritmos 1

Mateus Faria Zaparoli Monteiro
RM: 2023028560

UFMG
Dezembro 2024

1 Introdução

O problema computcional a ser resolvido por meio deste trabalho prático envolvia o problema do fluxo máximo em um grafo e algumas consequências de encontrar tal fluxo.

Apresentou-se o problema utilizando o ambiente de uma fábrica, na qual e tinham diversos geradores elétricos e varios equipamentos que se utilizam dessa energia para funcionar, chamados de consumidores.

Assim, o trabalho consistia em encontrar o fluxo máximo da rede elética dessa fábrica, além de precisarmos encontrar a demanda não atendida, ou seja a quantidade de energia para que os consumidores operem de maneira correta, a energia desperdiçada na rede, energia que sai dos geradores e não chega totalmente nos consumidores.

Por fim, foi necessário responder quais conexões da rede, "fios elétricos" haviam sido saturados, ou seja, por eles passava um fluxo igual a sua capacidade.

2 Modelagem

Para modelar o problema e executar a solução foi utilizado C++ e conceitos de programação orientada a objetos. A rede elétrica foi modelada a partir de um grafo direcionado, criado utilizando uma classe, tal grafo foi construído utilizando duas matrizes de adjacência, a primeira representando as arestas entre i e j do grafo original, i e j significam que a aresta sai do vértice i e vai para o vértice da coluna j, a capacidade da aresta é o valor do inteiro na linha do elemento ij da matriz, já a segunda matriz representa as arestas reversas na construção do grafo residual.

Foi criada também uma classe auxiliar chamada "Ponto" que foi criada para representar os geradores e consumidores da rede, sendo utilizada como base para a criação do grafo, contendo funções como getDemanda() e getTipo() para facilitar a resolução do problema.

3 Solução

Para solucionar a questão principal do problema, que era a de qual o fluxo máximo na rede, e daí derivar todas as outras respostas foi utilizado o algoritmo, visto em sala, de Ford-Fulkerson, ele consiste de assumir fluxo igual a zero em todas as arestas, e caso uma aresta possua menos fluxo que sua capacidade permitimos que mais fluxo seja passado por ela, caso ela já possua uma quantidade positiva de fluxo passando por ela, seja esse fluxo menor igual sua capacidade, deixamos que tal fluxo volte e a quantidade de fluxo passando por tal aresta diminua.

Para que essas possibilidades sejam modelados cria-se um grafo residual no qual a capacidade das arestas originais passa a ser a capacidade original subtraida do fluxo que passa por aquela aresta, e a capacidade de uma aresta não existente no grago original, chamada de aresta reversa, é definida como o fluxo positivo da aresta original.

A partir disso, vai se aumentando gradualmente o fluxo por cada aresta do grafo até que não haja nenhum caminho possível no grafo que aumente o fluxo, é atingido assim o fluxo máximo.

Nesse algoritmo, precisamos diversas vezes achar um caminho simples da fonte até o destino, "source" e "sink", respectivamente, e, para isso utilizamos um algoritmo de busca em largura, BFS, observando se o vértice já foi visitado e se sua capacidade é positiva, fazemos isso partindo da fonte e indo até o destino vértice por vértice, caso haja um caminho em que todas as capacidades sejam positivas esse é um caminho de aumento. Achado esse caminho, procuramos a aresta de menor capacidade nele e alteramos o valor de cada aresta nesse caminho subtraindo essa capacidade de forma a saturar a aresta de menor capacidade caso essa aresta tenha a mesma direção da aresta do grafo original, e caso ela seja reversa, ou seja, tenha direção oposta aumentamos a capacidade dela o que corresponde a aumentar o fluxo dela no algoritmo, com isso procuramos outro caminho repetindo esse processo até não haver mais caminhos simples da fonte até o destino.

Quando não houver mais caminho simples da fonte ao destino o fluxo do grafo será o fluxo máximo e teremos resolvido a questão central do trabalho prático.

É essêncial ressaltar que o algoritmo de Ford-Fulkerson é desenhado para o caso de uma fonte e um destino final, no caso do trabalho temos diversas fontes, os geradores, e diversos destinos, os consumidores, para adaptar esse algoritmo ao problema foi criado, então um gerador universal que tem uma aresta para cada gerador com capacidade muito alta, e foi criada, também uma fonte universal e todos os consumidores se ligam a ela com uma aresta de capacidade igual a demanda daquele consumidor.

4 Análise de Complexidade

Para modelar o problema foram criadas duas matrizes de inteiros para cada grafo, um original e um residual e uma matriz extra para guardar as capacidades originais das arestas, para fazer a impressão das arestas críticas as quais foram saturadas utilizando o algoritmo de fluxo máximo, como as matrizes foram criadas um número arbitrário de vértices sua complexidade espacial é $O(n^2)$ sendo n o numero de vértices arbitrários.

Na BFS, criou-se uma queue de inteiros de tamanho variável que foi atualizada a cada execução da BFS , no pior caso essa queue pode abrigar todos os vértices do grafo fazendo com que sua complexidade espacial seja $O(n)$ onde n é o número de vertices do grafo.

Para o algoritmo de Ford-Fulkerson, por fim foram criados três vetores de tamanho igual ao numéro arbitrário de vértices, que foram utilizados para guardar se o vértice já havia sido visitado, se ele possuia antecessores, e se a aresta em questão era uma aresta reversa ou original, a complexidade espacial desse algoritmo foi, então de $O(n)$, com n igual ao numero de vértices.

Em relação à complexidade temporal temos que a BFS tem uma complexidade assintótica de $O(m + n)$ sendo m o número de vértices e n o número de arestas pois a BFS tem que processar cada vértice e aresta do grafo, a cada execução dela. O algoritmo de fluxo máximo foi implementando utilizando-se de um while para calcular o fluxo máximo, no pior dos casos o fluxo aumenta de 1 em 1 até chegar no fluxo máximo tendo esse loop tem custo máximo de $O(k)$, sendo k o valor do fluxo máximo, por si só. A cada iteração desse while chamamos a BFS citada acima uma vez também, dentro ainda desse while temos dois laços for, o primeiro atualiza o caminho do fluxo, e o segundo é responsável por atualizar o grafo residual, esses dois laços tem custo proporcional ao caminho de aumento, chamado de caminho aumentante, que no pior caso tem tamanho igual ao numero de vértices, n, do grafo, ou seja, complexidade temporal de $O(n)$. Desse modo a cada iteração do while temos o custo da BFS $O(m + n)$ e o custo dos laços $O(n)$ tendo um custo de $O(m + n)$ a cada while, e como no pior dos casos executamos ele k vezes, temos que o custo final desse algoritmo é de $O(k * (m + n))$.

Por fim, para entregar todas as resposta executamos algumas operações aritméticas de custo constante $O(1)$ e um laço while para calcular qual a aresta saturada de maior capacidade e imprimí-las em ordem decrescente, assim no pior caso passamos pelo vetor de arestas saturadas n vezes e para cada vez retiramos uma aresta, fazendo assim com que o custo temporal de dar o resultado de arestas críticas em ordem decrescente seja de $O(n^2)$.

5 Considerações Finais

O trabalho foi interessante no sentido de consolidar mais uma vez o método de criação e representação de problemas em grafos, além de necessitar da utilização de BFS, tópico abordado também no trabalho anterior. Outro fator interessante

foi o fato de exercitar a adaptação do problema para se encaixar no algoritmo de fluxo mínimo de ford-fulkerson. A parte mais fácil do problema foram as criações das classes Grafo e Ponto, e a leitura da entrada padrão, coisas que já fizemos diversas vezes ao longo do curso. As partes mais difíceis foram escrever o algoritmo de ford-fulkerson a partir dos slides vistos em aula e das descrições encontradas, além disso modelar o problema do grafo e das arestas e trabalhar com as arestas reversas e atualizar os fluxos e capacidades também exigiu muito durante a execução desse trabalho.

Por fim, é válido ressaltar que o desenvolvimento desse TP contribuiu para a prática da matéria de algoritmos uma vez que a matéria em si é em grande parte teórica.

6 Referências

1. https://en.wikipedia.org/wiki/Breadth-first_search
2. https://pt.wikipedia.org/wiki/Algoritmo_de_Ford-Fulkerson
3. Slides livro Algorithmic Design e de exemplo de Ford-Fulkerson
4. Aula do monitor a cerca do Trabalho Prático
5. Notas das aulas de algoritmos