PUC Minas
A B A K Ó S
Instituto de Ciências Exatas e Informática

Trabalho Pratico de Grafos 01

7 de dezembro de 2022

Caio Massote
Felipe Leal
Rodrigo Paiva

Resumo

Os algoritmo de grafos são muito diversos e são usados em várias áreas para auxiliar na resolução de muitos problemas. Dois caminhos são disjuntos em arestas quando partem de um mesmo vértice e tem o mesmo destino, porém passando por arestas diferentes em todas as etapas.

A proposta deste trabalho trata-se de criar e documentar um algoritmo capaz de encontrar o maior número de caminhos disjuntos em arestas possível dado um par de vértices, e listar o número de caminhos encontrados e esses caminhos na ordem em que as arestas são utilizadas.

Portanto, para constatar a funcionalidade do algoritmo, este também será submetido a uma série de teste para assegurar o funcionamento adequado do algoritmo para diversos tipos de grafo.

Palavras-chave: Grafos, Caminhos Disjuntos, Ciência da Computação

1 INTRODUÇÃO

Este documento apresenta como foi desenvolvido o trabalho prático 02 da disciplina de grafos. A partir disso o documento foi dividido em seções, nas quais cada uma delas aborda um determinado tema, com o intuito de esclarecer o objetivo do trabalho, qual o algoritmo que foi desenvolvido e como ele foi implementado, além de especificar a sua complexidade.

A implementação do código referente a esse trabalho foi feita totalmente utilizando a linguagem Java, as únicas bibliotecas utilizadas foram, LinkedList e Queue.

2 CONCEITOS BÁSICOS

Antes de iniciarmos a abordagem referente aos algoritmos, é de suma importância explicar alguns conceitos que foram utilizados para a execução métodos além de facilitar a compreensão dos objetivos do projeto.

1. **Rede de Fluxos:** ma rede de fluxo é um grafo direcionado e ponderado G = (V, E) em que cada aresta se associa a cada aresta e (pertence) E um valor de capacidade u(e) > 0.

   Existem dois vértices especiais em uma rede de fluxo:

   (a) **Vértice s:** “source”(ou fonte) que representa a origem fluxo;

   (b) **Vértice t:** “terminal”(ou sumidouro) que representa o destino do fluxo.

   Os demais nós da rede são denominados nós internos. Além disso, assume-se que não há arestas entrando em s nem saindo de t, que todo vértice possui pelo menos uma aresta incidente a ele e que as capacidades são inteiras.

2. **Rede Residual:** Dado um fluxo f em uma rede G = (V, E), a rede residual G’(f) é um grafo direcionado ponderado que:

   (a) Possui os mesmos vértices que G, isto é, V(G’) = V(G);

   (b) Para toda aresta pertencente (v, w) E tal que f(e) < u(e), G’(f) contém a aresta direta (v, w) com capacidade (residual) igual a ur(e) = u(e) - f(e);

   (c) Para toda aresta pertencente (v, w) E tal que f(e) > 0, G’(f) contém a aresta reversa (w, v) com capacidade (residual) igual a ur(e) = f(e).

   Um caminho na rede residual saindo da fonte s até o sumidouro t é chamado de caminho aumentante (ou caminho de aumento de fluxo).

3. **Fluxo Máximo:** Um fluxo é dito máximo quando todo caminho de s a t possui pelo menos uma aresta saturada.

4. **Caminho:** Um caminho é um uma sequência de vértices conectados por arestas que não se repetem.

5. **Caminhos Disjuntos:** Um par de caminhos em um grafo é disjunto se os caminhos não têm arcos em comum entre si, ou seja, se nenhum arco do grafo é usado por ambos os caminhos.

3 MÉTODO UTILIZADOS

Para desenvolver o algoritmo solicitado durante o percurso da matéria Teoria de Grafos e Computabilidade, utilizamos uma forma adaptada do método Ford-Fulkerson para realizar a tarefa de encontrar os caminhos disjuntos.

3.1 Método Ford-Fulkerson

O método Ford-Fulkerson ou algoritmo Ford-Fulkerson é um algoritmo guloso que calcula o fluxo máximo em uma rede de fluxo. A ideia por trás do algoritmo é a seguinte: desde que haja um caminho da origem (nó inicial) até o sorvedouro (nó final), com capacidade disponível em todas as arestas do caminho, enviamos fluxo por um dos caminhos. Então encontramos outro caminho, e assim por diante. Um caminho com capacidade disponível é chamado de caminho de aumento .

A partir disso, o algoritmo possui o seguinte funcionamento:

1. Inicialize o fluxo em todas as bordas para 0.

2. Embora haja um caminho de aumento entre a origem e o coletor, adicione esse caminho ao fluxo.

3. Atualize o gráfico residual.

Vale a pena ressaltar, o caminho inverso, se necessário, porque se não os considerarmos, talvez nunca encontremos um fluxo máximo.

3.1.1 Detalhes sobre a implementação

O método Ford-Fulkerson leva em consideração a capacidade residual de cada aresta e, no nosso algoritmo, implementamos reduzindo a capacidade das mesmas em uma unidade, já que a ideia era achar a quantidade de caminhos disjuntos em arestas em um grafo direcionado. Sendo assim, o fluxo máximo é igual ao número máximo de caminhos disjuntos no grafo entre a fonte e o sumidouro. Para implementar esse algoritmo realizamos um método de busca em

largura, que recebe o respectivo grafo que é passado em uma matriz de adjacência, a fonte, e o sumidouro. A sua principal função é montar na memória a estrutura do grafo que foi passado como parâmetro, definindo os parentes de cada vértice, além de retornar um valor booleano se existe ou não um caminho entre a origem e o destino.

A principal função implementada chamada "achaCaminhosDisjuntos", inicialmente cria uma rede residual exatamente igual ao grafo original e, assim, o algoritmo entra em uma repetição chamando a busca em largura, ou seja, a condição de entrada e de continuar nessa estrutura de repetição é exatamente o que o nosso método de busca em largura retorna, se existe ou não um caminho entre a fonte e o sumidouro. Após entrar na estrutura de repetição o algoritmo faz o caminho reverso (sumidouro-fonte) e checa novamente se há um caminho. Desse modo, no grafo residual, ele inverte todas as arestas que já foram utilizadas, incrementa um valor ao número máximo de caminhos disjuntos, concatena em uma string o caminho que foi feito e passa esse novo grafo residual como parâmetro para o nosso algoritmo de busca em largura, repetindo o processo até não haver mais caminhos disjuntos.

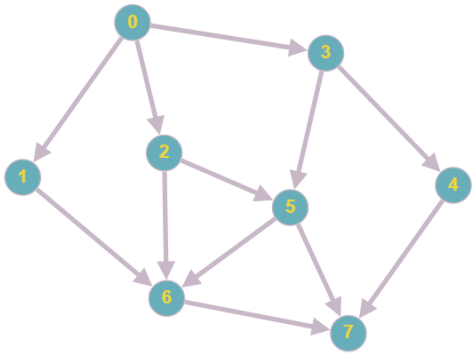
Portanto, após sair da repetição é chamado um método para imprimir todos os caminhos possíveis encontrados e quantos são.

4 TESTES

Nessa sessão vamos apresentar os resultados obtidos ao testar o nosso algoritmo explicado no documento ate então, mostrar exemplos e expor como geramos os grafos de teste.

4.1 Primeiros testes

Como primeiro teste de funcionamento do algoritmo utilizamos o seguinte grafo:



Grafo gerado para teste.

O algoritmo nos deu a seguinte saída para origem 0 e destino 7:

```
Caminhos:
0-1 1-6 6-7
0-2 2-5 5-7
0-3 3-4 4-7
Maximo de Caminhos Disjuntos em Arestas no Grafo: 3
```

Resultados

4.2 Testes

Para realizar testes no nosso algoritmo com grafos de topologias e tamanhos diferentes utilizamos os seguintes tipos:

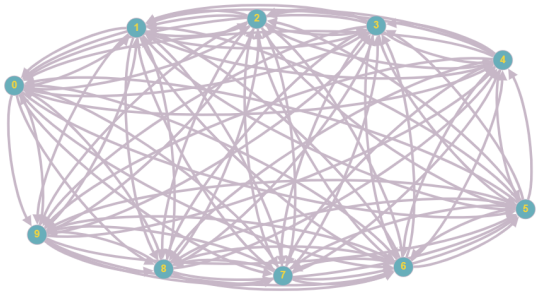
1. Grafo Completo
2. Grafo Regular
3. Grafo Cíclico

Realizamos os testes com 10, 100, 500 e 1000 arestas. Implementamos um algoritmo para gerar esses tipos de grafos aleatoriamente.

4.2.1 Grafos Completos

Grafo Completo – um grafo onde todos os seus vértices tem o grau máximo. Ou seja, existe aresta presente entre todos os pares de vértices.

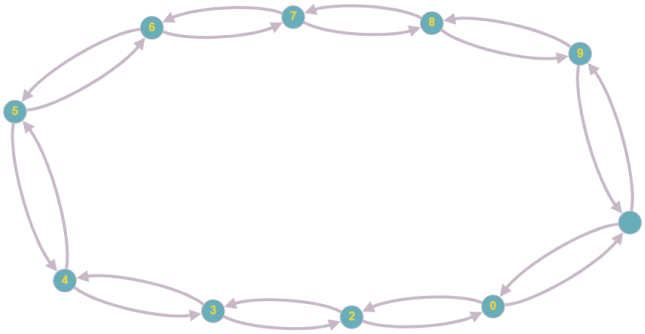
Para exemplificar o tipo de grafo que utilizamos para testar, representamos o menor grafo utilizado, sendo ele:



4.2.2 Grafos Regulares

Grafo Regular – um grafo é regular, quando todos os seus vértices tem o mesmo grau. Ou seja, ele é r-regular.

Para esses testes utilizamos, como o menor grafo, o seguinte:

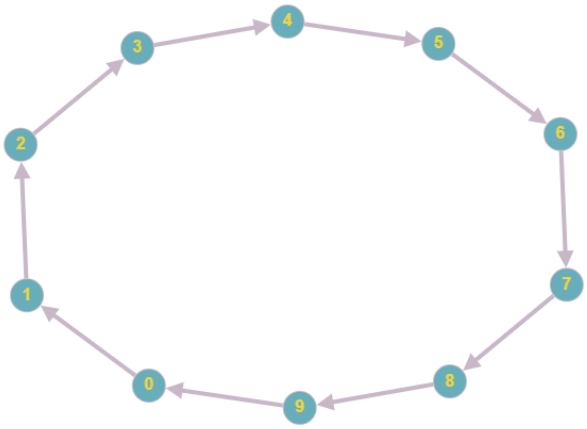


Grafo 2-regular de 10 vértices

4.2.3 Grafos Cíclicos

Grafo Cíclico - um grafo ciclico ou grafo circular é um grafo que consiste de um único ciclo, ou em outras palavras, um número de vértices conectados em uma rede fechada.

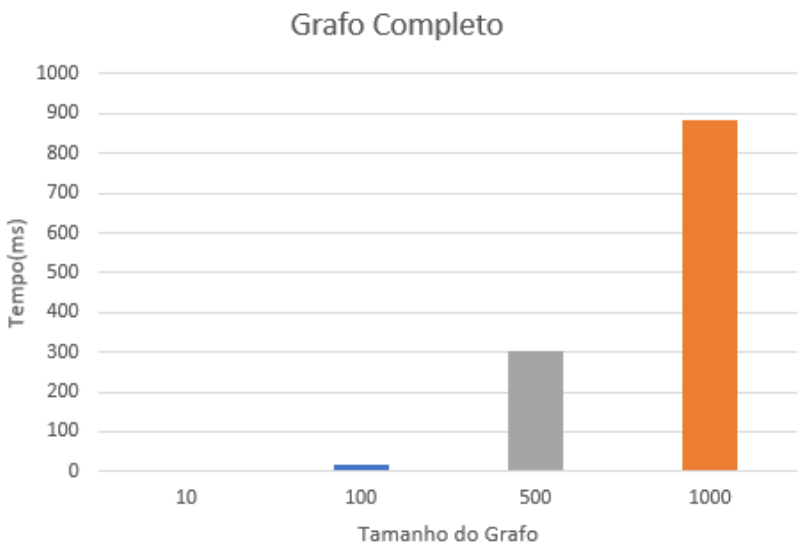
Para esses testes utilizamos, como o menor grafo, o seguinte:



Grafo cíclico de 10 vértices

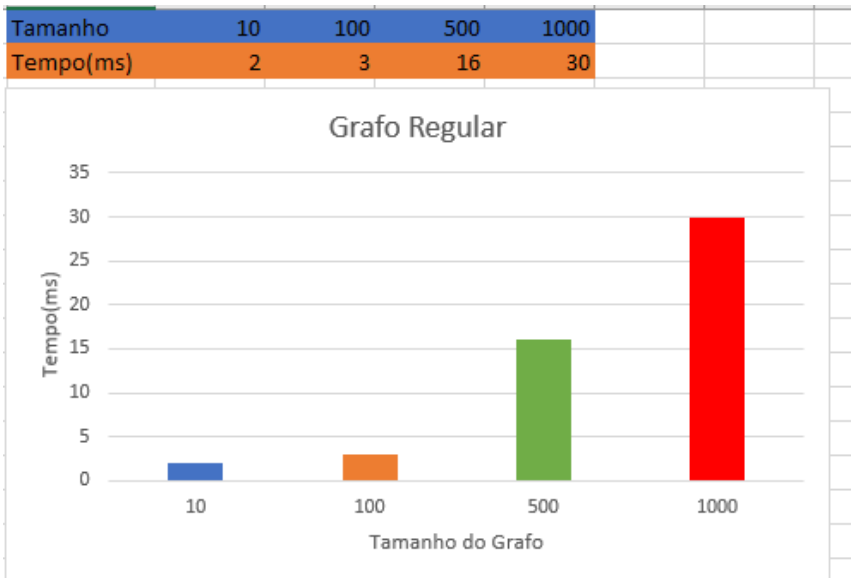
5 RESULTADOS E GRÁFICOS

Tamanho	10	100	500	1000
Tempo(ms)	2	18	305	882



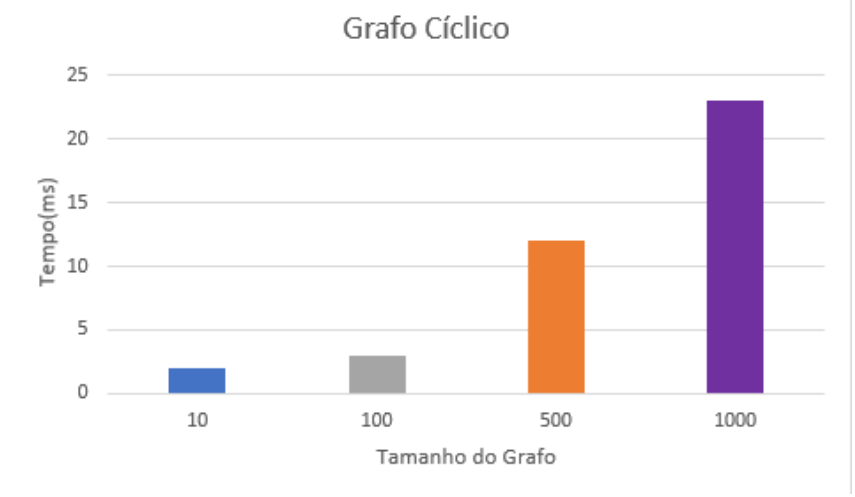
Resultados Grafo Completo

Tamanho	10	100	500	1000
Tempo(ms)	2	3	16	30



Resultados Grafo Regular

Tamanho	10	100	500	1000
Tempo(ms)	2	3	12	23



Resultados Grafo Cíclico

REFERÊNCIAS