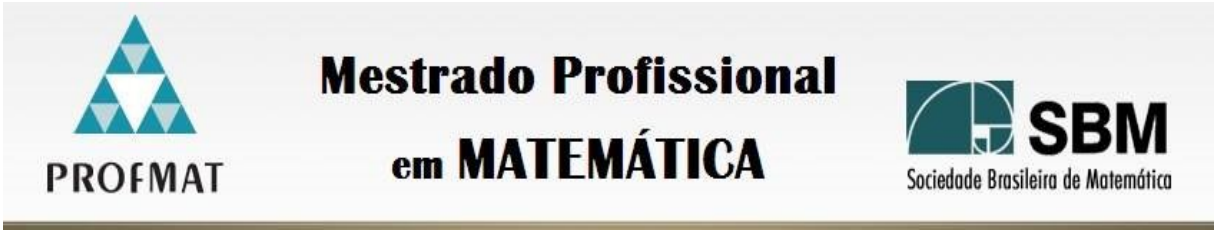


INSTITUTO DE MATEMÁTICA PURA E APLICADA

Trabalho de Conclusão de Curso

Árvores: Algoritmos e Aplicações

LAURO DE AVELLAR E ALMEIDA

Orientador: Paulo Cézar Carvalho

Rio de Janeiro
02/03/2018

Resumo:

Este trabalho tem como objetivo transmitir um pouco do conhecimento básico de como surgiu a Teoria dos Grafos, o que vem a ser uma Árvore, e em particular, o que é uma Árvore Geradora Mínima (AGM), os algoritmos de obtenção de AGMs, além de sua aplicação no auxílio do desenvolvimento do raciocínio estruturado no Ensino Fundamental II.

Além da apresentação dos dados acima citados, ele encerra um estudo de caso, com a descrição e análise do comportamento dos alunos ao utilizar-se de um dos algoritmos de obtenção de AGMs que foram mostrados.

Palavras Chaves: Grafos, Árvores, AGMs, Educação

Abstract:

This work aims to transmit a little of the basic knowledge of how the Theory of Graphs came to be, which becomes a Tree, and in particular, what is a Minimum Spanning Tree (MST), the algorithms of obtaining MSTs, in addition of its application in support of the development of structured reasoning in Elementary School II.

In addition to presenting the above data, it closes a case study, describing and analyzing student behavior using one of the algorithms for obtaining AGMs that were shown.

Keywords: Graphs, Trees, MSTs, Education

Agradecimentos

Agradeço a minha família, e principalmente, minha esposa Rozana, pela paciência demonstrada nos dois anos em que ficava fora o dia inteiro, etapa necessária para alcançar a titulação de "**Mestre**";

Agradeço ao Professor *PAULO CÉZAR CARVALHO*, pela oportunidade que me deu em minha reta final. Graças a ele, essa conquista tornou-se possível;

Agradeço aos professores *Antônio Carlos Branco* e *Moacyr Alvim Silva*, pela avaliação justa de meu trabalho final e, principalmente, das sugestões feitas para o prosseguimento futuro das atividades nele propostas.

Agradeço ao Professor *Carlos Gustavo Tamm de Araujo Moreira - GUGU*, pelo apoio que sempre demonstrou a nossa turma, sempre disposto a tirar nossas dúvidas, mesmo que não fosse de suas disciplinas;

Agradeço aos meus colegas de turma, pelo apoio mútuo trocado entre nós, sem o qual muitos de nós não teríamos chegado até essa etapa. Somos a última turma do programa PROFMAT no IMPA nesta etapa, e talvez em definitivo, e isso aumentou nosso vínculo. Em especial, um agradecimento post mortem ao nosso colega ADILSON, que se foi na fase final;

Agradeço ao meu pai OGUM e a minha mãe OYÁ, por terem me dado a determinação e a força necessárias para vencer as etapas, e nunca perder a esperança em conquistá-las, vencendo assim as demandas que o caminho me impôs;

À todos, Muito Obrigado!!

Capítulo 1

Teoria dos Grafos: O que é, de onde vem?

1.1 Introdução

Pode-se dizer que teoria dos grafos nasceu na cidade de Königsberg da antiga Prússia, hoje chamada Kaliningrado, na atual Rússia.

Foi um típico desafio lógico, já que consistia em fazer um passeio passando pelas sete pontes, porém uma vez sobre cada pontes que interligavam as partes da cidade que era cortadas por vertentes do rio Pregel, formando uma ilha na parte central.

O matemático suíço Leonhard Euler, em 1736, não só conseguiu elucidar o problema como acabou por criar uma teoria que se aplica a vários problemas deste tipo.

Ele usou um modelo simplificado das ligações entre as regiões e estabeleceu um teorema que diz em que condições são possíveis percorrer cada linha exatamente uma vez e voltar ao ponto inicial. E este foi o primeiro teorema da Teoria dos Grafos, e o modelo usado para resolver o problema o primeiro exemplo de grafo documentado.

Os grafos são usados na área de TI na criação de fluxogramas, redes de comunicação (como redes Lan e WLan) modelos de fluxo de dados, algoritmos de escalonamento, layout de circuitos, algoritmos de pesquisa e ordenação e modelos de máquinas de estado. Infelizmente, naquela época, era tudo muito teórico, com pouca visão prática da disciplina.

No Brasil, praticamente não havia literatura em português, que pudesse ser considerada como livro texto básico para o assunto.

O livro de Paulo Osvaldo Boaventura (cuja primeira edição é de 1973), por exemplo, ainda não havia se firmado como um dos guias da disciplina, o que explica a disciplina não ser cogitada até bem pouco tempo como sendo parte integrante do currículo de formação de professores de matemática.

Durante muitos anos, apenas alguns cursos de Matemática apresentavam a disciplina a seus graduandos de bacharelado, e assim mesmo, após o

período de disciplinas obrigatórias.

Apenas os itens mais famosos, como o problema das pontes de Königsberg, ou o problema das quatro cores, eram questões habitualmente citadas nos textos básicos, a maioria apostilas feitas pelos próprios professores da cadeira.

E isso, apesar do boom da computação no Brasil nos anos 1980 e 1990, algoritmos como o de Kruskall, Prim ou Dijkstra (todos da década de 1960), não faziam parte da teoria vista normalmente, uma vez que computadores ainda não era comuns (Informática, como um curso de nível superior, ainda estava se firmando. Era a época em que se devia entrar num curso de Matemática, para depois migrar para Informática).

Deste modo, a ideia de se aplicar conceitos para o ensino de tópicos de Matemática, ou para se estruturar o raciocínio dos alunos, não era vista como viável para esse tipo de trabalho

1.2 Notas Históricas

1.2.1 Introdução Histórica

Como todos os ramos da Matemática, a Teoria dos Grafos teve seu início por conta da curiosidade humana, na sua eterna busca na resolução de um problema.

A diferença é que, enquanto os outros ramos tiveram sua origem em problemas práticos, ligados ao trabalho diário da pessoa, a Teoria dos Grafos tem o seu início determinado pela resolução de problemas (um em particular), que mais parecem passatempos, além de não ter tido sua teoria unificada logo de início.

Diversos resultados esparsos foram obtidos, em diversas situações desconectas, e que somente séculos depois foram percebidos como sendo partes de um mesmo assunto.

Como vemos em **[Lloyd]**, foi Leonard Euler o primeiro a utilizar os conceitos básicos para resolver o problema das pontes de Königsberg.

Pela importância do método utiizado por Euler em mostrar que o problema não tinha solução, e ainda, demonstrar as condições necessárias para que tal tipo de problema pudesse ser resolvido, ele é considerado o Pai da Teoria dos Grafos.

Figura 1.1: Leonard Euler

Euler ainda escreveu alguns resultados da teoria, mas como ele próprio não acreditava que fosse se tornar uma coisa séria, suas anotações ficaram escondidas, no meio de sua enorme produção.

Esses resultados ficaram perdidos por décadas, só sendo redescobertos após outros estudiosos trabalharem na Teoria dos Grafos.

Temos ainda os resultados de Gustav Robert Kirchhoff, nascido na mesma Königsberg (1824 - 1877), cujo trabalho com circuitos elétricos utilizou os conceitos de grafos, e em particular, árvores.

Esses resultados serviram de incentivo para que outros pesquisadores e cientistas, das mais diversas áreas, utilizassem desses conceitos em seus trabalhos.

Isso aconteceu com o trabalho de Arthur Cayley (1821 - 1895), que utilizou os conceitos de árvores em seu trabalho como químico orgânico, especificamente, isômeros de hidrocarbonetos alifáticos saturados.

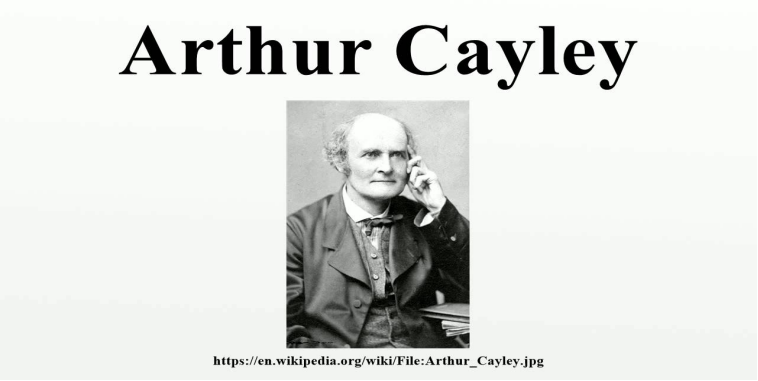


Figura 1.2: Arthur Cayley

A teoria dos grafos contou ainda com a importante contribuição de William Rowan Hamilton (1805 - 1865) - matemático, físico e astrônomo - que ao inventar um jogo simples, consistindo na busca de um percurso fechado envolvendo todos os vértices de um dodecaedro regular, de tal modo que cada um deles fosse visitado uma única vez, deu origem ao estudo de grafos Hamiltonianos, que têm por problema-base encontrar um caminho fechado, que passasse por todos os vértices do dodecaedro.

Já no século XX, o trabalho de William Thomas Tutte (1917 - 2002), criptologista, foi de grande importância no desenho de grafos. A introdução do uso de técnicas da álgebra linear (matrizes) para desenhar grafos facilitou o manuseio e armazenamento das informações em computadores.

Por conta do crescente poder dos computadores em manipulação de dados, alguns problemas (que seriam meros passatempos, pois eram impossíveis de serem trabalhados em grande escala) passaram a ser resolvidos, tais como traçado de rotas de entregas, fluxo de mercadorias, entre outros.

A partir de 1970, a teoria dos grafos teve um grande salto com o desenvolvimento acelerado dos computadores, os quais, após a 2a guerra mundial alcançaram seu apogeu, com a criação do primeiro software para microcom-

putador criado pelos estudantes William (Bill) Gates e Paul Allen, o qual era uma adaptção do BASIC (Beginners All-Purpose Symbolic Instruction Code, ou "Código de Instruções Simbólicas para todos os Propósitos dos Principiantes") para o ALTAIR (computador pessoal projetado em 1975).

Anos mais tarde, Gates e Allen fundaram a Microsoft, uma das mais bem sucedidas companhias de softwares para microcomputadores.

Foi então que surgiram publicações referentes a algoritmos de grafos, abrindo assim possibilidades para utilização aplicada da teoria dos grafos.

No Brasil a teoria dos grafos chegou, segundo **[Boaventura]** (p.3), no ano de 1968 com a apresentação de trabalhos sobre esta teoria no I Simpósio Brasileiro de Pesquisa Operacional.

Desde então, algumas universidades como UFRJ, UFF, USP, UNESP e UNICAMP, começaram a realizar trabalhos de pesquisa sobre teoria dos grafos, sendo que hoje em dia essas mesmas universidades possuem em seus quadros de docentes, pesquisadores em teoria dos grafos e aplicações.

1.2.2 As Pontes de Königsberg

Königsberg (atualmente Kaliningrad), no ano de 1776, possuia 7 pontes ligando as diversas partes da cidade, cortada pelo rio Pregel, e com partes da cidade situadas em duas ilhas deste rio, além das margens do rio.

Pelo fato da cidade se espalhar por essas regiões, a população começou a se perguntar se era possível estabelecer um trajeto que passasse por todas as áreas da cidade, de forma a passar apenas uma vez por cada ponte.

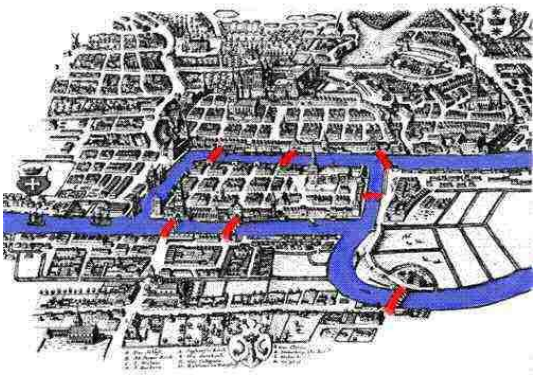


Figura 1.3: Mapa de Konigsberg, com fontes destacadas

Apesar dos diversos trajetos propostos, a população sempre se via obrigada a passar por uma das pontes mais de uma vez. Mas, apesar de não conseguirem encontrar um trajeto, não conseguiam também afirmar que tal trajeto seria impossível.

Leonhard Euler (1707-1783), em 1736 analisou o problema, e utilizando elementos de Teoria dos Grafos, demonstrou em um artigo que, tal trajeto só seria possível, se em cada região, o número de pontes que chegam a ela for par.

Embora os elementos e, nem mesmo a própria Teoria dos Grafos ainda não estivessem determinados, o raciocínio utilizado por ele é a base dos problemas de trajeto mínimo.

Observe que a ponte 7 não foi usada no trajeto, apesar de todas as regiões de Konigsberg já terem sido percorridas

Euler separou a cidade em regiões (vértices), trabalhou as pontes como sendo as ligações entre as regiões (arestas), e trabalhou em cima de combinatória para demonstrar a impossibilidade de tal trajeto, além de estabelecer as condições em que tal trajeto seria possível.

Este problema é considerado como sendo o pontapé inicial da Teoria dos Grafos, embora o estabelecimento da mesma como um ramo da Matemática

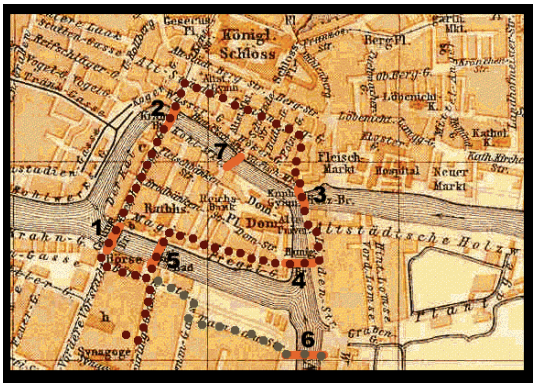


Figura 1.4: Um dos caminhos propostos por Euler

só ocorreu décadas depois, quando foi observado que uma série de resultados obtidos isoladamente, para problemas distintos, continham elementos em comum.

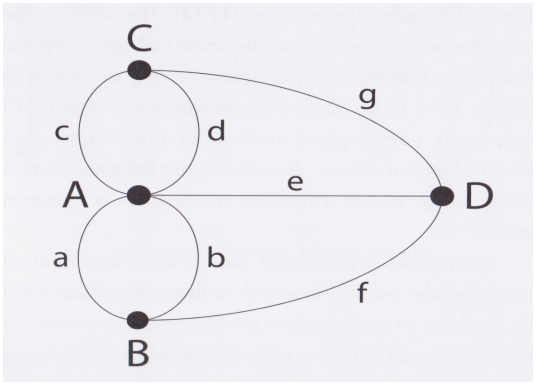


Figura 1.5: Grafo representando as regiões e as pontes da cidades

1.3 Teoria dos Grafos e a Educação Matemática

O desenvolvimento de teorias matemáticas que relacionam elementos de conjuntos discretos é bastante recente se comparado à história da "matemática contínua". Exemplo disso, a Teoria dos Grafos, formulada já no século XVIII e que ainda assim foi "redescoberta muitas vezes" (HARARY, 1973, apud [**Boaventura**], p. 2).

Nas últimas décadas, pesquisas da área de Educação e Educação Matemática (BRIA, COSENZA, CAMPOS, 2000; MUNIZ JUNIOR, 2007; MALTA, 2008; DEGGERONI, 2010) sugerem a abordagem da Teoria dos Grafos no Ensino Fundamental e Médio, como viés para discussão de problemas de matemática aplicada ao cotidiano escolar.

Muitos pesquisadores e educadores tem apresentado trabalhos e cursos, mostrando a viabilidade da inclusão de conceitos em nível de ensino médio, uma vez que a Teoria dos Grafos é uma excelente ferramenta para a modelagem de muitas situações.

De acordo com a situação proposta, podemos verificar seu nível de dificuldade, e a partir daí, observa-se que várias delas se referem a questões apresentadas em nível de Ensino Médio, e até nível de Ensino Fundamental II.

O nível dessas atividades pode ser classificado de acordo com a finalidade e a área de destino, de forma a separarmos o que pode ser feito em nível estudantil básico, médio ou em nível superior.

Quanto à finalidade, a classificação se faz entre atividades que usam modelos discretos e atividades de modelagem. Enquanto estas requerem interpretação, formulação e análise de um problema, aquelas já apresentam modelos prontos a serem analisados pelos alunos.

Quanto à área de destinação , as atividades dividem-se de acordo com a dificuldade na interpretação do enunciado ou na resolução do problema. Este trabalho também apresenta atividades mais complexas, pois é importante que os alunos do curso de matemática conheçam essa teoria desde um nível mais acessível à Educação Básica até em níveis mais aprofundados, como aplicações de Engenharia.

Observe que os problemas da Educação Básica, são problemas que normalmente se dão como desafio lógico aos alunos. Embora eles não tenham um embasamento teórico para explicar o porque de suas soluções, eles chegam

		Quanto à finalidade	
		Atividades que usam modelos discretos	Atividades de modelagem
Quanto à área de destinação	Educação Básica	*Desenhando na ponta do lápis*	*O Problema das Pontes de Konisberg*
	Ensino Superior	*Problemas de fluxo máximo em rede*	*O problema de destinação de funcionários*

Quadro 1: Classificação dos problemas apresentados, de acordo com a finalidade e com a área de destinação.

Figura 1.6: Quadro comparativo

à estas através de caminhos que são passíveis de receber essas explicações, baseadas na teoria.

No caso do "Desenhando na ponta do lápis" resume-se na construção de determinados desenhos sem que a ponta do lápis seja retirada do papel. A Teoria dos Grafos surge neste problema como uma ferramenta que permita ao aluno estudar a possibilidade de desenhar uma figura.

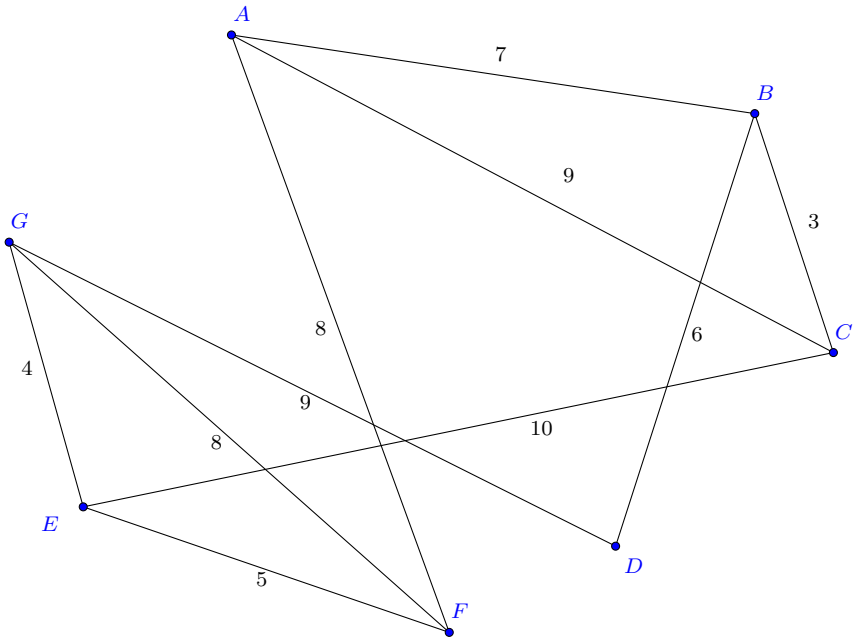
Como as figuras são, normalmente, apresentadas prontas, esta atividade caracteriza-se como uso de modelo discreto, já que o objetivo principal é fazer uma análise dos desenhos propostos.
O mesmo raciocínio foi usado no problema das pontes de Königsberg.

1.4 Grafos - De onde veio esse conceito ?

1.4.1 Definição

Um GRAFO é um ente matemático, formado por vértices e arestas, representando uma situação-problema, onde cada vértice representa um objeto e/ou pessoa; e as arestas representam as relações entre dois destes vértices.

Por conta disso, um grafo G é representado através de seu conjunto de vértice V; e de seu conjunto E de arestas, tendo assim a notação G(V,E), conforme define [**Jurk01**].



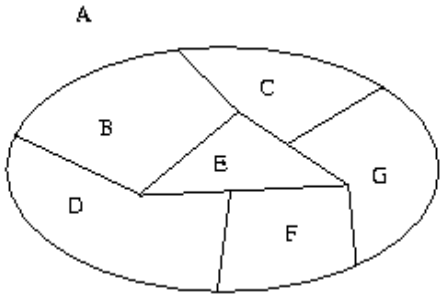


Figura 1.7: Mapa das regiões

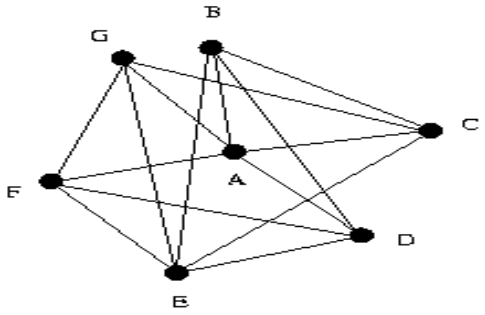


Figura 1.8: Grafo correspondente ao problema das regiões

A figura a seguir ilustra um conjunto de terras separadas por muros. Supondo que todas são cheias de água, como podemos esvaziar todas, furando um número mínimo de buracos nos muros?

Se cada terra e a área exterior formam o conjuntos de vértices, e se cada muro separando duas áreas é representado por uma aresta, obtemos o grafo ilustrado na figura.

A solução para esse problema passa pela aplicação do conceito de **ÁRVORE**, mas para definirmos o que vem a ser um grafo deste tipo, precisamos antes caracterizar o que vem a ser VIZINHANÇA e MATRIZ DE ADJACÊNCIA, o que é um GRAFO CONEXO, e o conceito de CICLO ou CIRCUITO em um grafo.

1.4.2 Vizinhança de um Vértice

Em teoria dos grafos, diz-se que um vértice é adjacente de um vértice **v**, em um Grafo **G**, quando este vértice que está ligado a **v** por uma aresta.

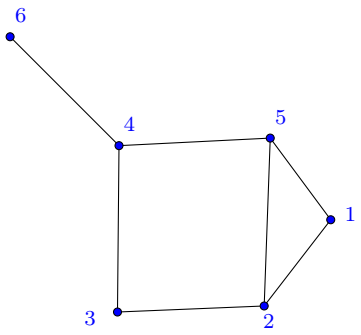
A vizinhança ou adjacência de um vértice **v** em um grafo **G** pode ser representado por um subgrafo de **G**, constituído por todos os vértices adjacentes a **v** e todas as arestas ligando esses vértices ao vértice **v**.

No caso de um vértice precisar ser ligado à ele mesmo, chamamos isso de *laço*. Essa situação não é do interesse do nosso estudo, e por isso não será abordada as suas utilizações.

Por exemplo, a imagem abaixo mostra um gráfico G(V,E) de 6 vértices e 7 arestas.

O vértice 5 é adjacente aos vértices 1, 2 e 4, mas não é adjacente aos vértices 3 e 6.

A vizinhança do vértice 5 é o grafo com três vértices, 1, 2 e 4, e uma aresta conectando os vértices 1 e 2.



Isso nos permite estabelecer uma relação entre os vértices, o que faz com que possa ser montada o que se chama de **MATRIZ DE ADJACÊNCIA** de um grafo.

Essa matriz nos permite montar uma imagem algébrica do grafo, permitindo assim um trabalho com o mesmo, sem precisarmos ter a imagem do grafo.

Com isso, o uso dessa abordagem também permitiu que os algoritmos computacionais de manipulação dos grafos fossem criados, como os algoritmos de Dijkstra, Prim e Kruskal que, apesar de terem uma fácil manipulação intuitiva no grafo, a implementação do mesmo num computador não é tão simples sem o uso de matrizes.

No caso do grafo G dado como exemplo anteriormente, se assumirmos que uma aresta existente é representada pelo valor "1"; e que uma aresta inexistente é representada pelo valor "0", teremos a seguinte matriz de adjacência:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Observe que nossa matriz de adjacência é uma matriz QUADRADA e SIMÉTRICA, com $n(V)$, por conta de cada vértice poder estar ou não ligado aos demais.

Desta forma, uma aresta ligando o vértice **i** ao vértice **j**, é representada pelo valor **(1)** no lugar do elemento $(a(i, j))$, onde **i** será o vértice de partida, e o **j** será vértice de chegada da aresta.

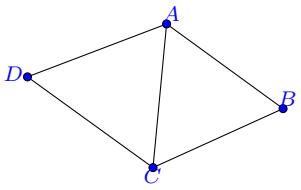
Verifique também que, a não ser que algum vértice tenha um laço, a diagonal principal desta matriz será composta, exclusivamente, por elementos iguais a **(0)**.

1.4.3 Conexidade

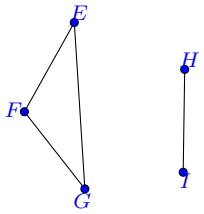
Segundo [**Jurk01**] (p.26-31), podemos descrever a conexidade como sendo a propriedade inerente aos vértices de um grafo em que, partindo de um vértice **O** qualquer, sempre é possível alcançar qualquer outro vértice **D** deste grafo, usando um grupo qualquer de arestas. Este caminho de arestas entre esses vértices é chamado de **CADEIA**.

Desta forma, qualquer grafo que apresente essa propriedade é dito como sendo **CONEXO**, enquanto que os que não possuem são chamados de **NÃO-CONEXOS** ou **DESCONEXOS**.

Assim, basta que se apresente um par de vértices que não possua qualquer cadeia interligando-os para se mostrar que o grafo é desconexo.



Grafo 1



Grafo 2

Podemos observar que, no GRAFO 1, sempre existe um caminho entre quaisquer dois vértices do grafo, seja qual for o critério de escolha destes.
Exemplo:

Entre os vértice B e D temos os caminhos: B-A-D; B-C-D; B-C-A-D e B-A-C-D.

Observe que não existe uma exigência de que o caminho seja único, ou o de menor distância. Basta exibir um caminho.

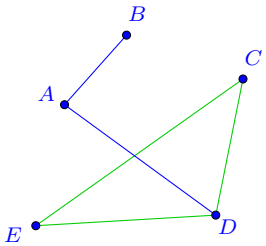
Já no GRAFO 2, não existe qualquer aresta ligando um vértice do conjunto E,F,G à um vértice do conjunto H,I.

Por exemplo, não é possível estabelecer um caminho entre os vértices E e I.

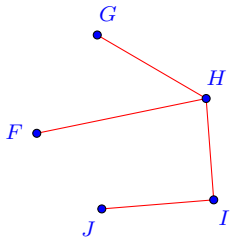
1.4.4 Ciclo ou Circuito

Um **CICLO** ou **CIRCUITO**, como vemos em [**Boaventura**], é qualquer caminho em que, partindo de um vértice qualquer pertencente à ele, pode-se retornar a este, após percorrer todo o caminho, sem ter de passar novamente pelos vértices percorridos.

Exemplos de Grafos, com e sem Ciclo, respectivamente



Grafo 1



Grafo 2

No exemplo dado acima temos, no GRAFO 1, um exemplo de grafo com um circuito entre os vértices C,D,E (destacado em verde).

Observe que, escolhido qualquer um dos três vértices, é possível retornar a ele, passando apenas uma vez pelos outros vértices que compõe o ciclo.

Já no GRAFO 2, não é possível partir de um ponto e retornar a ele, sem ter de passar novamente pelos vértices percorridos.

A ideia do ciclo é muito utilizada em situações que todos os vértices (ou arestas) do grafo precisam ser percorridos, retornando ao ponto de partida.

As definições de **Grafo Euleriano** e de **Grafo Hamiltoniano** são feitas utilizando-se dessas noções.

Um *Grafo Euleriano* é aquele que percorre todas as arestas do grafo, sem repetição destas.

O problema das pontes de Königsberg é demonstrado como sendo impossível, justamente por ser necessário passar por uma ponte (fazendo o papel de aresta, neste caso) mais de uma vez.

Observe que um grafo só pode ser chamado de EULERIANO (ou seja ter um caminho que passe por todas as suas arestas sem repetição) se todos os vértices tiverem um número par de arestas incidentes, ou no máximo dois vértices terem número ímpar de arestas (a razão do problema das pontes não ter solução).

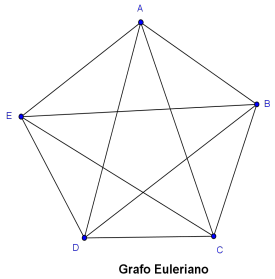


Figura 1.9: Grafo Euleriano

Já um grafo que possua um *Caminho Hamiltoniano* é aquele em que se percorre todos os seus vértices, sem que seja preciso passar por um vértice mais de uma vez.

Repare que, na figura abaixo, os dois exemplos de caminhos não utilizam todas as arestas do grafo (na realidade, são complementares), mas estes dois caminhos não são os únicos possíveis.

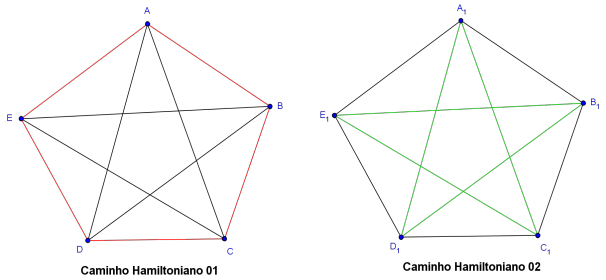


Figura 1.10: Caminhos Hamiltonianos

Além do caminho A-B-C-D-E-A, e do caminho A-C-E-B-D-A, apresentados nos grafos da figura, poderíamos ter o caminho A-C-D-E-B-A, que usa arestas dos dois caminhos anteriores, formando assim um terceiro caminho.

Essa ideia, somada ao conceito de árvore, será a origem dos algoritmos utilizados na construção das chamadas **ÁRVORES MÍNIMAS GERADORAS**.

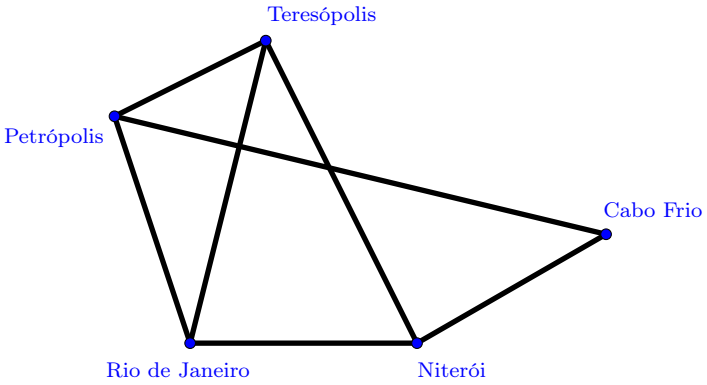
1.4.5 Conceitos Adicionais

Alguns conceitos estão apresentados nesta seção pois, apesar de não serem imprescindíveis para o trabalho desenvolvido, ajudam a expressar melhor o conteúdo. São eles:

1 - Grafo Rotulado

É o grafo em que os vértices recebem nomes (ou outra designação qualquer), diferentes das letras maiúsculas habituais.

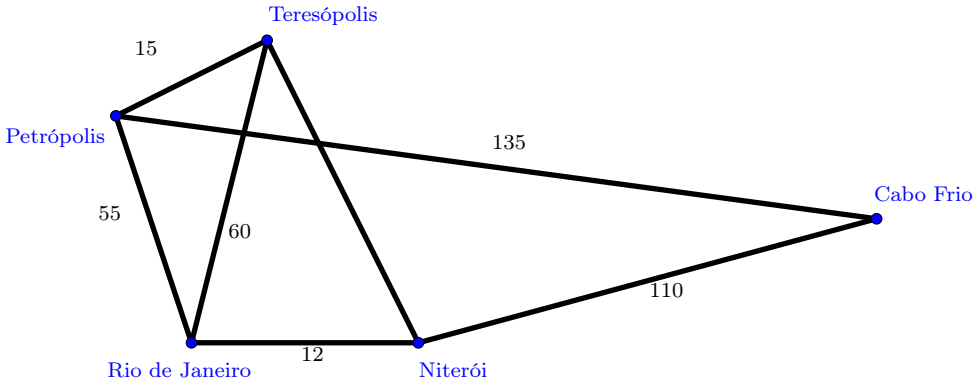
Exemplo de Grafo Rotulado



2 - Grafo Valorado

É o grafo em que suas arestas tem valores atribuídos a elas.
Em geral, este grafo trabalha em conjunto com o anterior.

Exemplo de Grafo Rotulado e Valorado



Embora o conceito **Árvore** não dependa do grafo ser valorado, a maior parte das situações onde a estrutura se aplica envolve valores, o que acaba forçando o uso de grafos valorados.

Capítulo 2

Árvores

2.1 Conceitos Básicos

Definição

Podemos definir uma ÁRVORE como sendo um grafo conexo, sem ciclos.

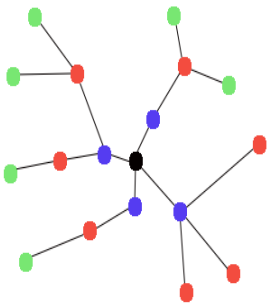


Figura 2.1: Exemplo de Árvore

Como nem todo grafo é uma árvore, podemos afirmar que um grafo será uma árvore se e somente se existir um único caminho entre dois vértices quaisquer deste grafo.

Jurkiewicz define, em **[Jurk02]** (p67-68), as seguintes correlações, que podem ser utilizadas como definições alternativas para o conceito de ÁRVORE, na forma de um teorema:

Teorema Seja T um grafo com n vértices.
As seguintes afirmações são equivalentes:
(i) T é uma árvore.
(ii) T não contém ciclos e tem (n - 1) arestas.
(iii) T é conexo e tem (n - 1) arestas.
(iv) T é conexo e toda aresta é uma ponte[1].
(v) Todo par de vértices de T é ligado por um único caminho.
(vi) T não contém ciclos, mas a adição de uma aresta produz um único ciclo.

A demonstração feita por Jurkiewicz segue o esquema de (se então), indo de um item para o seguinte, retornando ao item inicial no fim.
Apenas acrescentei imagens elucidativas à cada passagem da demonstração, tornando o entendimento mais fácil ao leitor.

Demonstração:
Seja um grafo T(V,E), com os vértices u,v $\in$ V(T), e uv a aresta que os liga, uv $\in$ E(T).

(i) $\Rightarrow$ (ii): Pela definição de árvore, T não contém ciclos.
Portanto, a retirada de uma aresta uv separa u de v e o grafo é separado em um par de árvores T_1 e T_2 com n_1 e n_2 vértices, respectivamente, tais que $n = n_1 + n_2$.
Por indução, o número de arestas de T_1 é $(n_1 - 1)$ e o número de arestas de T_2 é $(n_2 - 1)$.
Acrescentando a aresta uv, concluímos que o número de arestas de T é, portanto, $(n_1 - 1) + (n_2 - 1) + 1 = n - 1$.

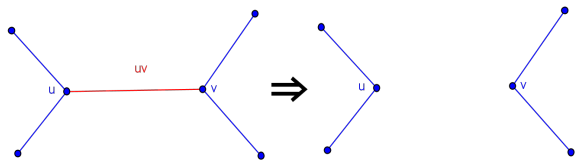


Figura 2.2: (i) ==> (ii)

[1]Ponte: Aresta que, se for retirada, desconecta o grafo. Também chamada de "Aresta de Corte"ou "Istmo".

(ii) ⇒ (iii): Se T fosse desconexo, cada componente seria uma árvore.

Por indução, o número de arestas em cada componente é inferior em uma unidade ao número de vértices e o número total de arestas seria inferior a $n - 1$.

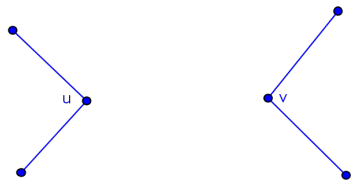


Figura 2.3: (ii) ==> (iii)

(iii) ⇒ (iv): A retirada de qualquer aresta separa o grafo, pois n - 2 arestas são insuficientes para conectar o grafo.

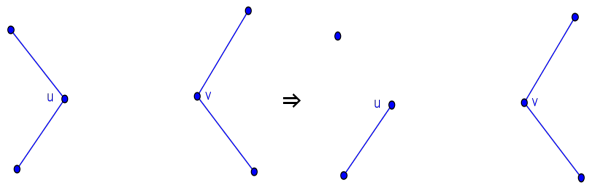


Figura 2.4: (iii) ==> (iv)

(iv) ⇒ (v): Se existisse mais de um caminho entre dois vértices, o grafo teria um ciclo e haveria uma aresta que não separaria o grafo.

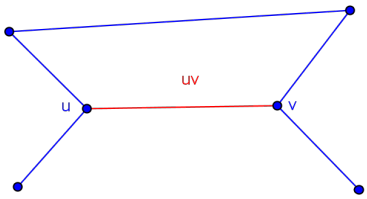


Figura 2.5: (iv) ==> (v)

(v) ⇒ (vi): Se T contivesse um ciclo, haveria um par de vértices ligado por mais de um caminho. Sejam u, v esses vértices.

A adição de uma aresta uv', inexistente nesta suposição, concatenada com o caminho (único) entre u e v, produz um ciclo.

Se este ciclo não fosse único, a retirada da aresta uv' deixaria dois caminhos distintos entre u e v.

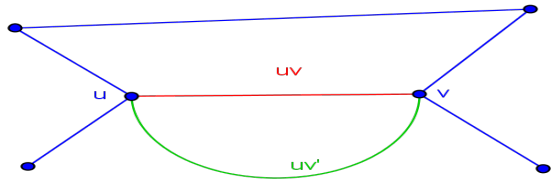


Figura 2.6: (v) ==> (vi)

(vi) ⇒ (i): Basta mostrar que T é conexo. Se T fosse desconexo, uma aresta ligando duas componentes não produziria um ciclo.

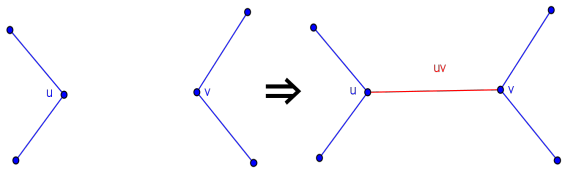


Figura 2.7: (vi) ==> (i)

No caso do grafo não ser conexo, temos a chamada **FLORESTA**, grafo desconexo formado por duas ou mais árvores.

O grafo da figura 2.3 é um bom exemplo de FLORESTA, onde cada um de seus componentes é uma árvore.

Todo grafo conexo possui, pelo menos, uma árvore de extensão associada a ele (formada por todos os seus vértice, e algumas de suas arestas).

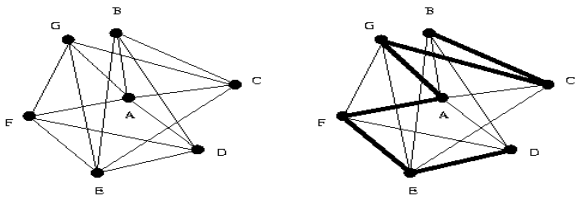


Figura 2.8: Exemplo de Grafo e sua Árvore associada

Essa figura é um ótimo exemplo do que é uma ÁRVORE GERADORA MÍNIMA, próximo assunto à ser abordado.

2.2 Árvore Geradora Mínima

Uma Árvore Geradora Mínima (de agora em diante, AGM - em inglês: Minimum Spanning Tree, ou MST), é uma árvore que contém todos os vértices do grafo original, de forma que a soma de todos os valores das arestas usadas para alcançar todos eles seja o menor possível.

Uma AGM não é necessariamente única para um grafo, embora em grafos pequenos esse fenômeno aconteça.

2.2.1 Algoritmos Gulosos

Um ALGORITMO GULOSO (em inglês: greedy algorithm) é um algoritmo que parte da técnica de conseguir uma solução em um local do grafo, tentando obter assim uma solução ótima para o grafo todo, juntando as soluções obtidas localmente., conforme vemos em [**AGuloso**].

Em geral, são algoritmos simples, e que podem fornecer a melhor solução. Existe o risco de não serem bem executados, se deixarem fechar um ciclo, o que geraria loops em sua implementação em programas de computador.

Um algoritmo guloso é usado, normalmente, em problemas de otimização. O objetivo é achar um conjunto de candidatos que possuam o melhor valor para uma situação objetiva, procedendo um algoritmo passo a passo.

1. Inicialmente, assume-se que a resposta é um conjunto vazio. A cada passo, o algoritmo seleciona um candidato a ser acrescentado no conjunto.

2. Ele tenta selecionar o melhor elemento do conjunto de candidatos. Se o acréscimo desse elemento no conjunto solução é viável, ele sera colocado no conjunto, senão ele sera rejeitado e nunca mais será considerado.

3. Se ele é viável, o algoritmo testa se esse conjunto é uma solução. Se é, o algoritmo pára e retorna essa solução. Senão, ele continua:

Um exemplo geral de *algoritmo guloso*:
função guloso(C: conjunto) : conjunto
{C é o conjunto de todos os candidatos}
S := {} (Conjunto que constitui a solução, inicialmente vazio)
Enquanto S não é uma solução e C não vazio
    x := Melhor elemento de C
    C := C - {x}
    Se x é viável então S := S U {x}
    Se S é uma solução então retornar S
        Senão retornar Não encontrei uma solução
fim enquanto
fim guloso

Esse tipo de algoritmo é dito guloso porque ele sempre tenta pegar o melhor pedaço, sem considerar as consequências no futuro.

Uma vez um candidato incluído no conjunto de solução, ele fica sempre. Isso significa que, para retornar uma solução ótima, a função de seleção do próximo candidato é crucial.

Eis um exemplo de algoritmo guloso para dar o troco de 90 centavos, visto em [**Arvore**].
função TROCO(C: conjunto de moedas) : conjunto de moedas
C é o conjunto de todos os candidatos
S := {}
Enquanto (Soma das moedas em S < 90) e (C não vazio)
    x := Moeda de maior valor em C
    C := C - {x}
    Se (Soma dos valores em S + x < 90) então S := S U {x}
    Se Soma dos valores em S = 90 então retornar S
        Senão retornar Não encontrei uma solução
fim Enquanto
fim TROCO

Note que o algoritmo pode retornar uma solução que não é ótima:
TROCO({50, 50, 25, 25, 10, 10, 10, 10, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1})
Retorna: {50,25,10,1,1,1,1,1}

Pior, pode não retornar uma solução, quando na verdade existe uma:
TROCO({50, 50, 25, 25, 10, 10, 10, 10, 1, 1, 1, 1})
Retorna: Não encontrei uma solução

Exemplos retirados do site (http://www.professeurs.polymtl.ca/michel.gagnon/Disciplinas/Bac/Grafos/Arvores/arvores.html, em 11/07/2017)

Pelo seu modo de trabalho, esses algoritmos são a base para o processo de geração de árvores mínimas. Por isso, problemas de caminho mínimo, de fluxo de rede, são resolvidos através destes algoritmos.

Os principais algoritmos gulosos para a geração de árvores mínimas são os de Primm e de Kruskal. São algoritmos muito semelhantes, com diferenças apenas na maneira de se implementar a escolha das arestas.

Esses algoritmos não garantem a unicidade do caminho encontrado, mas a solução encontrada é sempre ótima.

2.2.2 Algoritmo de Primm

O algoritmo de Prim é um algoritmo guloso, empregado para encontrar uma árvore geradora mínima num grafo conectado, valorado e não direcionado, conforme vemos em [**Prim**].

Isso significa que o algoritmo encontra um subgrafo do grafo original no qual a soma total das arestas é minimizada e todos os vértices estão interligados.

O algoritmo foi desenvolvido em 1930 pelo matemático Vojtěch Jarník e depois pelo cientista da computação Robert Clay Prim em 1957 e redescoberto por Edsger Dijkstra em 1959.

Outros algoritmos conhecidos para encontrar árvores geradoras mínimas são o algoritmo de Kruskal (estudaremos a seguir) e algoritmo de Boruvka (não vamos analisá-lo neste trabalho). No entanto estes algoritmos podem ser empregados em grafos desconexos, enquanto o algoritmo de Prim precisa de um grafo conexo.

O algoritmo de Prim encontra uma árvore geradora mínima para um grafo desde que ele seja valorado e não direcionado.

Por exemplo, se os vértices deste grafo representassem cidades e as arestas fossem estradas de terra que interligassem estas cidades, como poderíamos determinar quais estradas asfaltar gastando a menor quantidade de asfalto possível para interligar todas as cidades?

O algoritmo de Prim, neste caso, fornecerá uma resposta ótima para este problema, que não necessariamente é única.

Algoritmo genérico

Um algoritmo genérico para o algoritmo de Prim é dado da seguinte forma:

- Escolha um vértice S para iniciar o subgrafo
  enquanto houver vértices que não estão no subgrafo
  selecione uma aresta segura
  insira a aresta segura e seu vértice no subgrafo

- Esse algoritmo começa com a escolha da aresta de menor peso do grafo.

- A partir dela, sempre usando arestas ligadas à vértices que já estejam conectados à árvore em formação, escolhe-se a próxima aresta de menor valor possível, desde que está não feche um circuito.

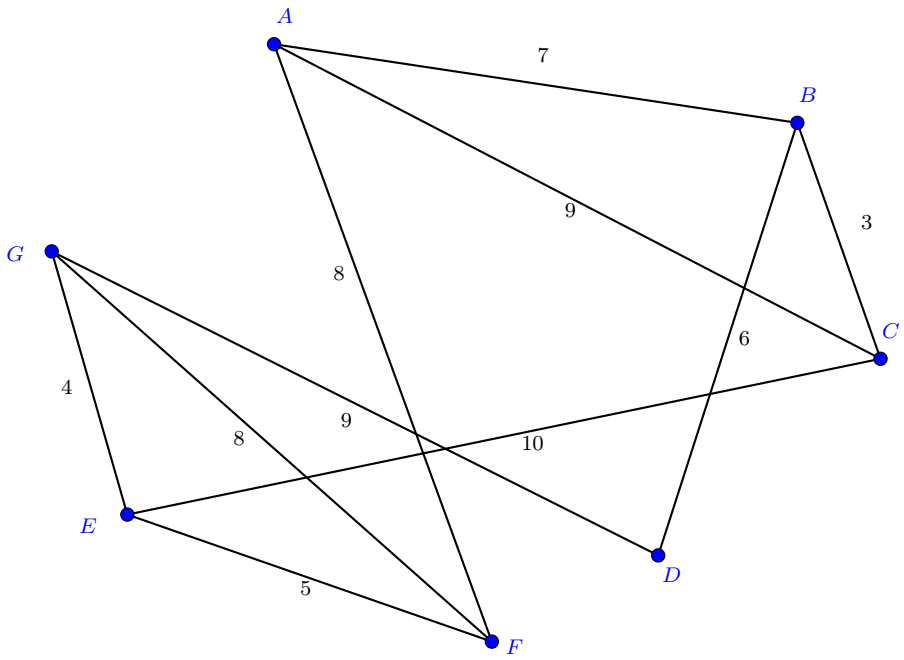
- O algoritmo encerra-se quando o último vértice for conectado à árvore.

Exemplo de Codificação do Algoritmo

função prim(G) G é grafo
Escolhe qualquer vértice do grafo como vértice inicial de partida
V(G) = Vértices de G
s := seleciona um elemento v $\in V(G)$
$Para(v \in V(G))$
$\pi[v] := nulo$
$Q := \{(0, s)\}$
$S :=\emptyset$
$Enquanto$(Q $\neq\emptyset$)
v := extrair-mín(Q)
S := S $\cup\{v\}$
$Para.cada..u..adjacente..v$
$se((u \neq S).e.((pesoDaAresta(\pi[u] \Rightarrow u) > pesoDaAresta(v \Rightarrow u)))$
$Q := Q\{(pesoDaAresta(\pi[u] \Rightarrow u), u)\}$
$Q := Q \cup \{(pesoDaAresta(v \Rightarrow u), u)\}$
$\pi[u] := v$
$retorna..\{(\pi[v], v)|v \in V(G)..e..\pi[v] \neq nulo$
$fimPrim$

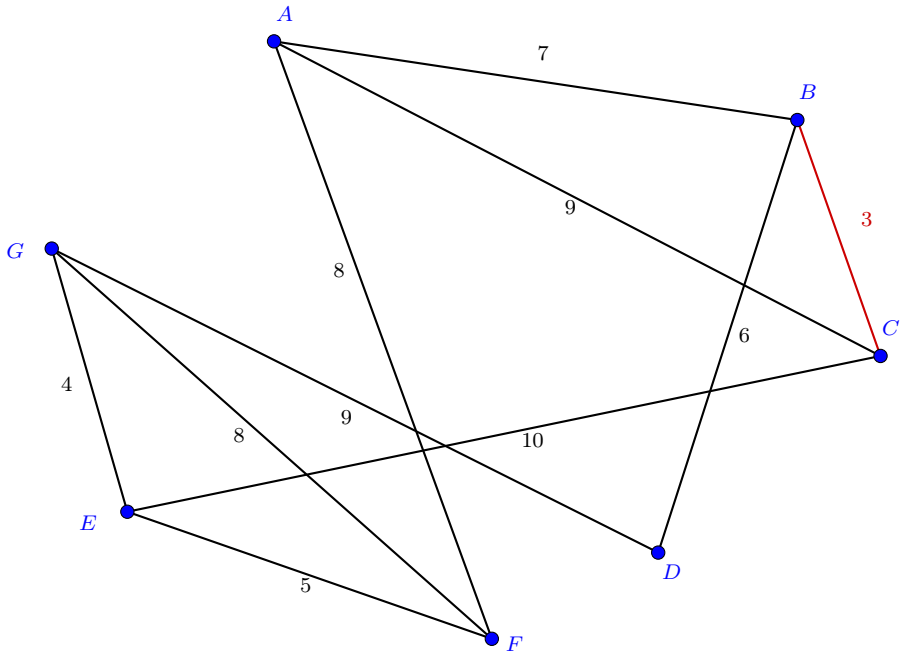
Vamos verificar a aplicação do algoritmo em um grafo-exemplo:

Modelo de um Grafo - Algoritmo de Prim



Como devemos escolher um vértice para começar, podemos escolher entre os vértices B ou C, uma vez que estes são as extremidades da aresta de menor valor do grafo G

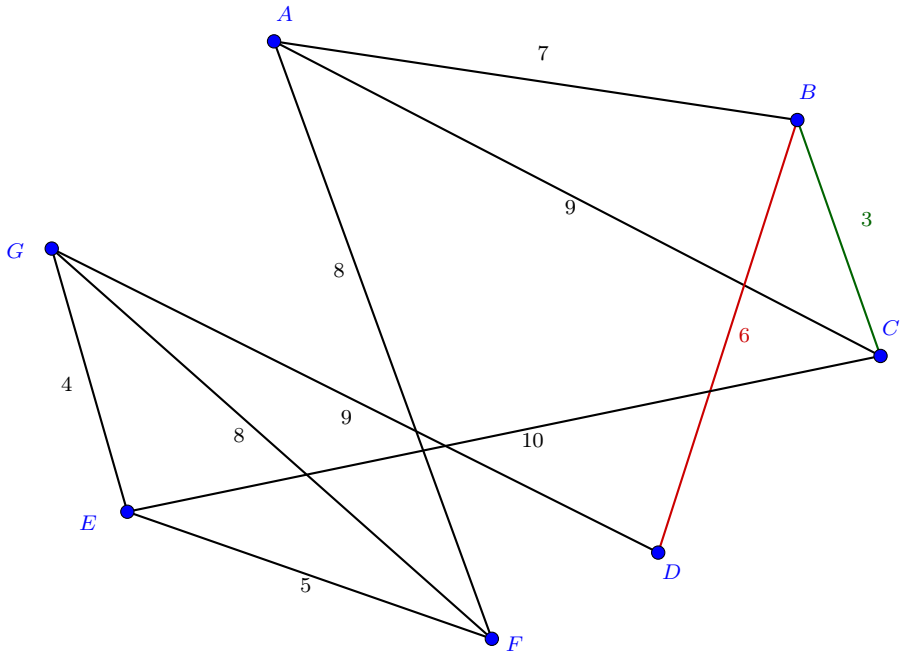
Algoritmo de Prim - 1º Passo



Como estamos com a aresta BC destacada, pelo algoritmo devemos pegar a aresta de menor valor que tenha extremidade em B ou em C.

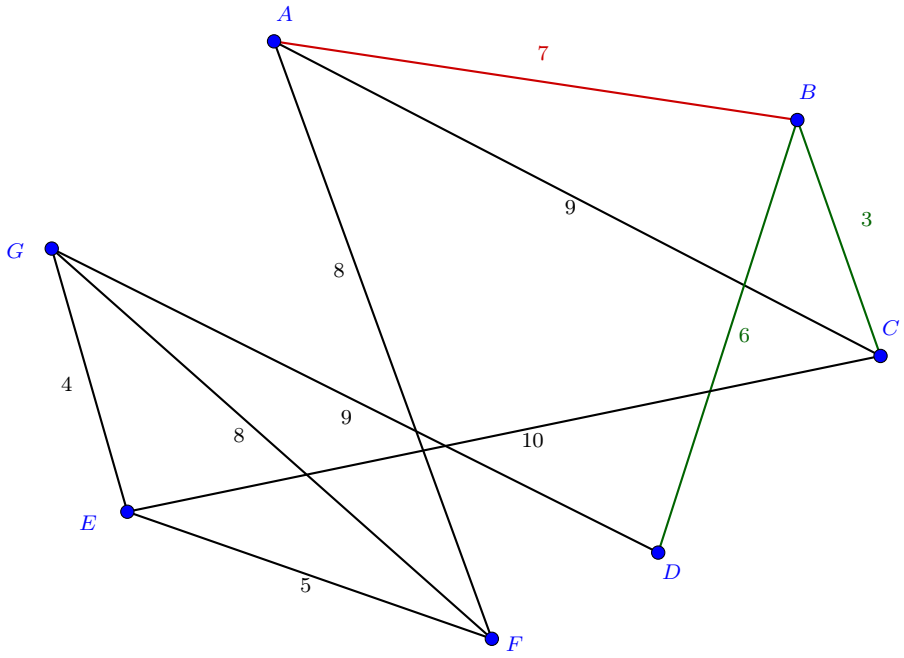
Essa necessidade faz com que o algoritmo acabe limitado a grafos conexos, uma vez que apenas arestas conectadas a vértices já utilizados poderiam ser escolhidas.

Algoritmo de Prim - 2º Passo



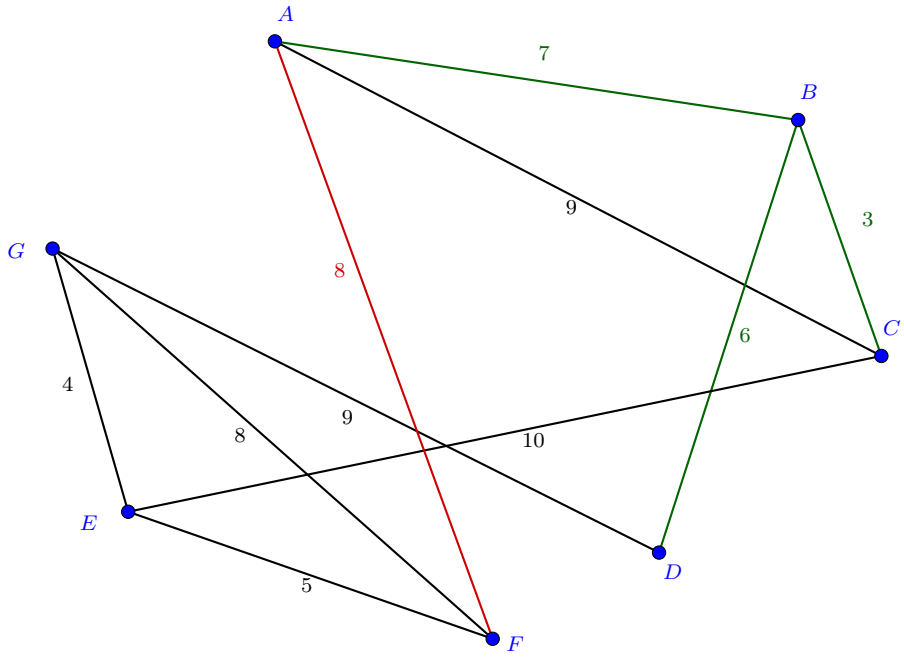
A partir deste ponto, seguimos o mesmo raciocínio do passo anterior, até que todos os vértices do grafo tenham sido conectados a nossa AGM.

Algoritmo de Prim - 3º Passo

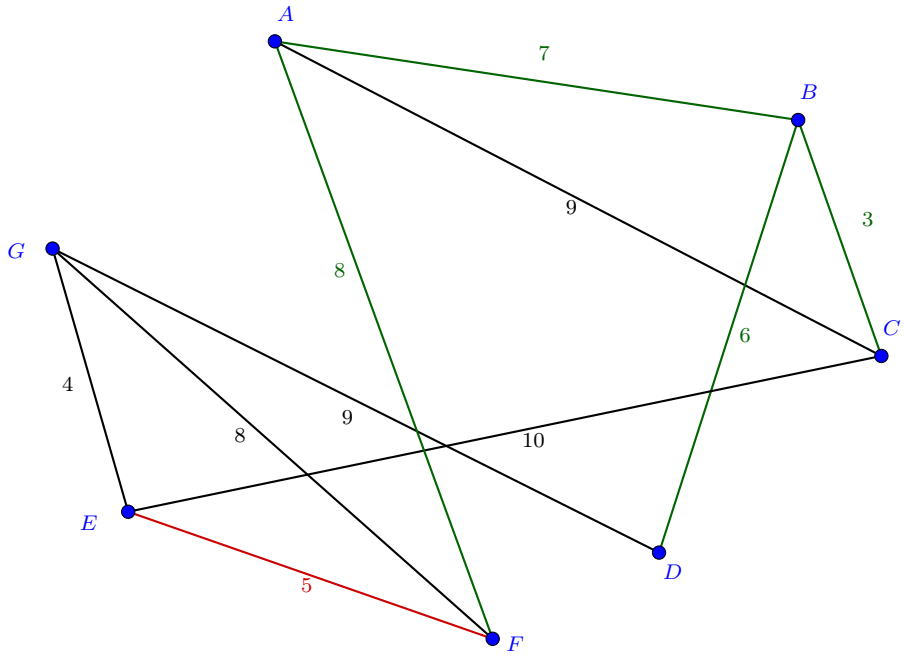


Observe que, apesar de o grafo ter arestas com valores menores, a limitação de utilizar apenas arestas ligadas a vértice já conectados à AGM impede o uso destas.

Algoritmo de Prim - 4º Passo

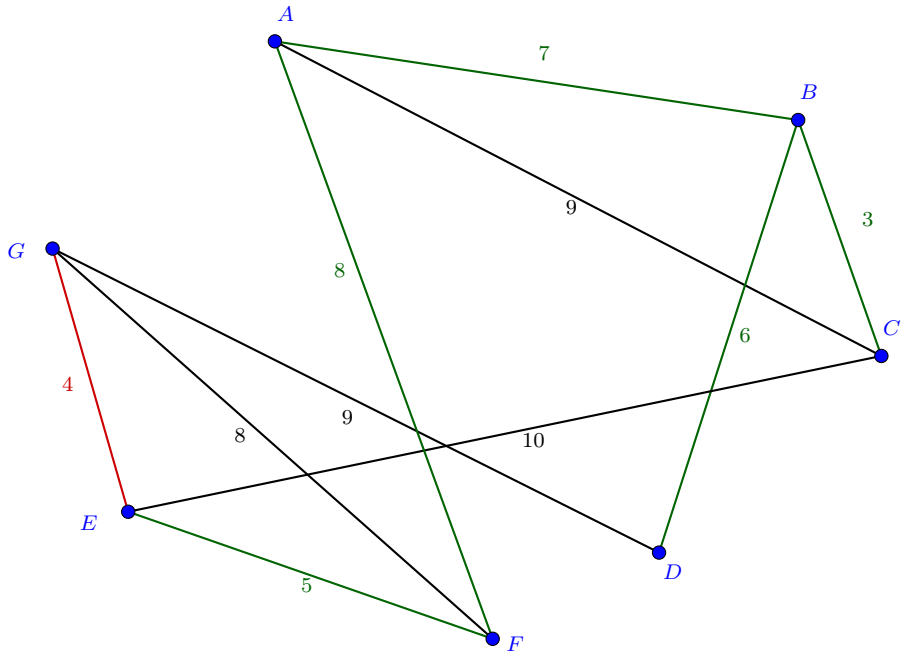


Algoritmo de Prim - 5º Passo



Neste ponto, falta apenas conectar o vértice G, para completar a AGM.

Algoritmo de Prim - 6º Passo



Neste ponto, temos nossa AGM construída. Não podemos garantir que essa será a árvore obtida, se escolhermos outro vértice para ponto de partida.

Após a análise do algoritmo de Kruskal com o mesmo grafo, vamos comparar as AGMs para verificar se as diferenças existem ou não.

2.2.3 Algoritmo de Kruskal

O algoritmo de Kruskal é um algoritmo em teoria dos grafos que busca uma árvore geradora mínima para um grafo conexo com pesos.

Isto significa que ele encontra um subconjunto das arestas que forma uma árvore que inclui todos os vértices, onde o peso total, dado pela soma dos pesos das arestas da árvore, é minimizado.

Se o grafo não for conexo, então ele encontra uma floresta geradora mínima (uma árvore geradora mínima para cada componente conexo do grafo).

Esse algoritmo funciona nos mesmos moldes que o algoritmo de Primm, com a diferença de que a próxima aresta a ser escolhida não precisa estar conectada à árvore que está se formando, vencendo inclusive a limitação do grafo ter de ser conexo, que o algoritmo de Prim exige.

A condição da aresta escolhida não fechar um circuito também faz parte deste algoritmo.

Seu funcionamento é mostrado a seguir:

1. crie uma floresta F (um conjunto de árvores), onde cada vértice no grafo é uma árvore separada.

2. crie um conjunto S contendo todas as arestas do grafo

3. enquanto S for não-vazio, faça:
   remova uma aresta com peso mínimo de S,
   se essa aresta conecta duas árvores diferentes,
   adicione-a à floresta, combinando duas árvores numa única árvore parcial,
   do contrário, descarte a aresta
   fim enquanto.

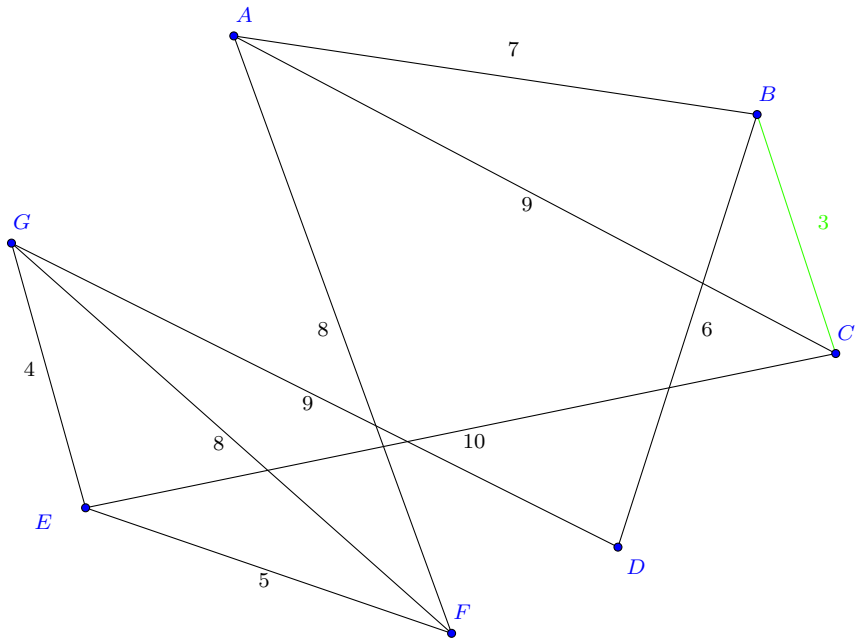
Ao fim do algoritmo, a floresta tem apenas um componente e forma uma árvore geradora mínima do grafo.

Com o uso de uma estrutura de dados eficiente, o algoritmo de Kruskal possui complexidade de tempo igual a $O(mlog(n))$, onde m representa o número de arestas e n o número de vértices, como visto em [**Kruskal**].

Devemos observar que uma estrutura de dados só é necessária para sua implementação computacional. o algoritmo de Primm possui a mesma complexidade de tempo e necessidade de uso de estrutura de dados para sua implementação computacional.

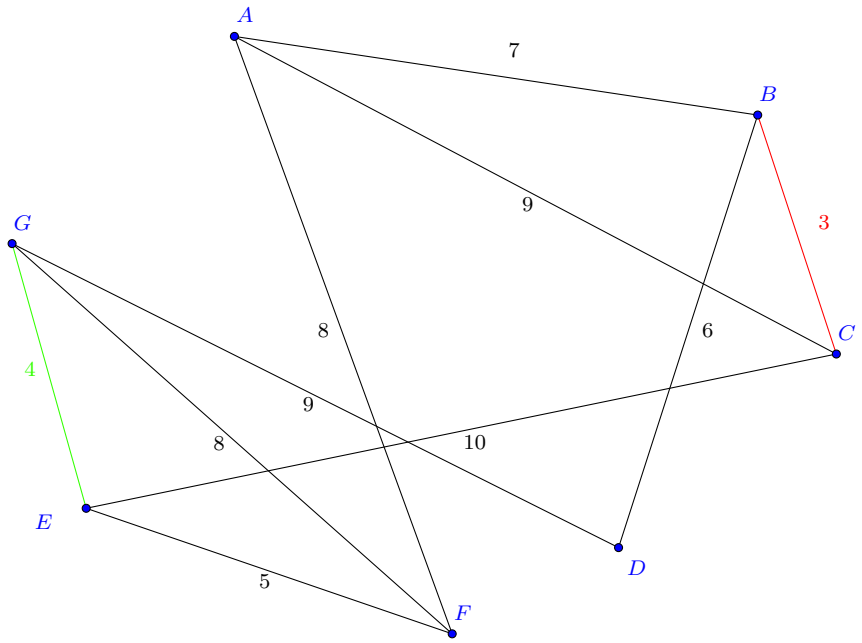
A seguir, temos a sequência de passos necessários para se encontrar à AGM relativa ao grafo utilizado no exemplo.

Modelo de um Grafo



Como a aresta BC é a que apresenta o menor valor fixado, começamos por escolhe-la para iniciar a construção da nossa árvore.

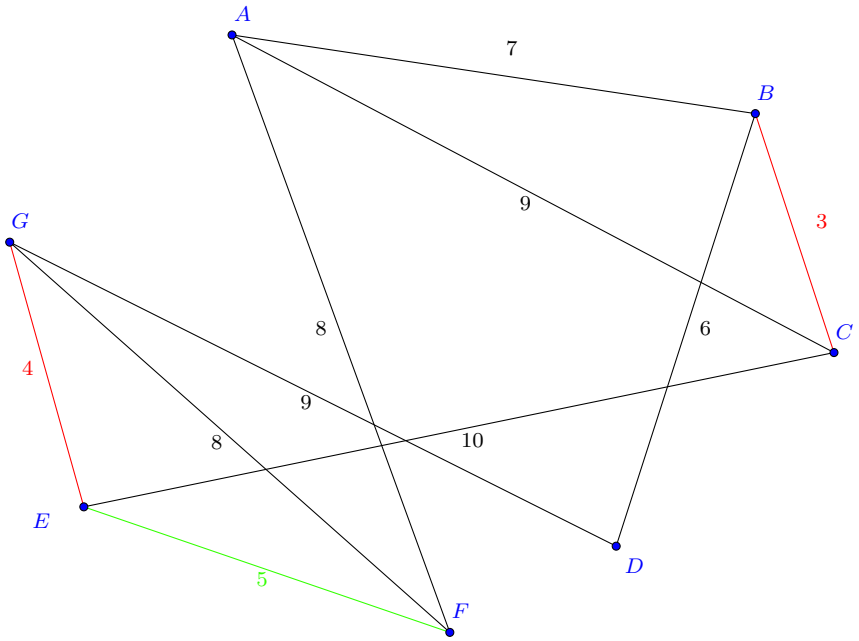
Árvore Gradora - 2º passo



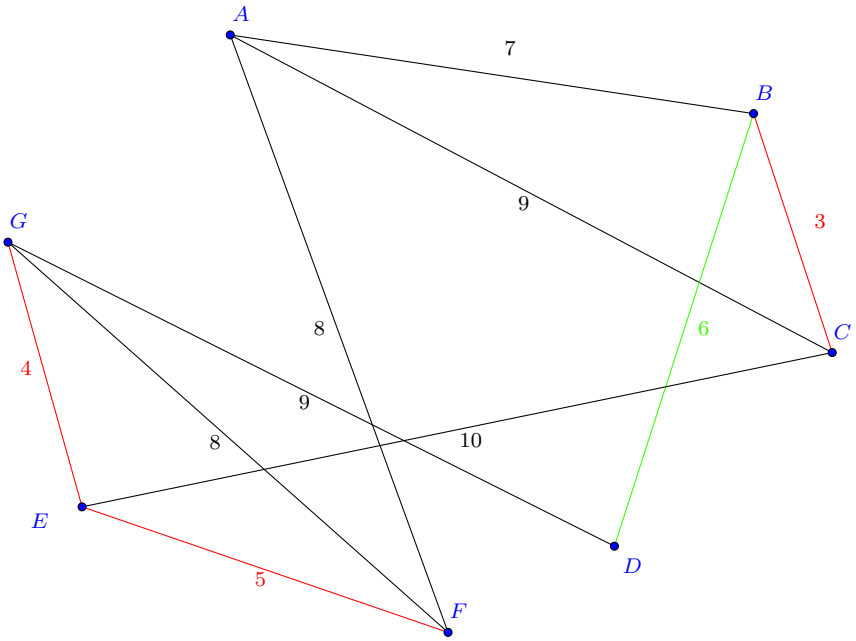
Observe que a próxima aresta escolhida foi a aresta EG, que não possui qualquer conexão com a aresta BC. A escolha foi baseada apenas no seu valor, e no fato de não fechar um ciclo na árvore construída até o momento.

Do quarto passo até o final, mostra-se a construção da árvore procurada, através do algoritmo.

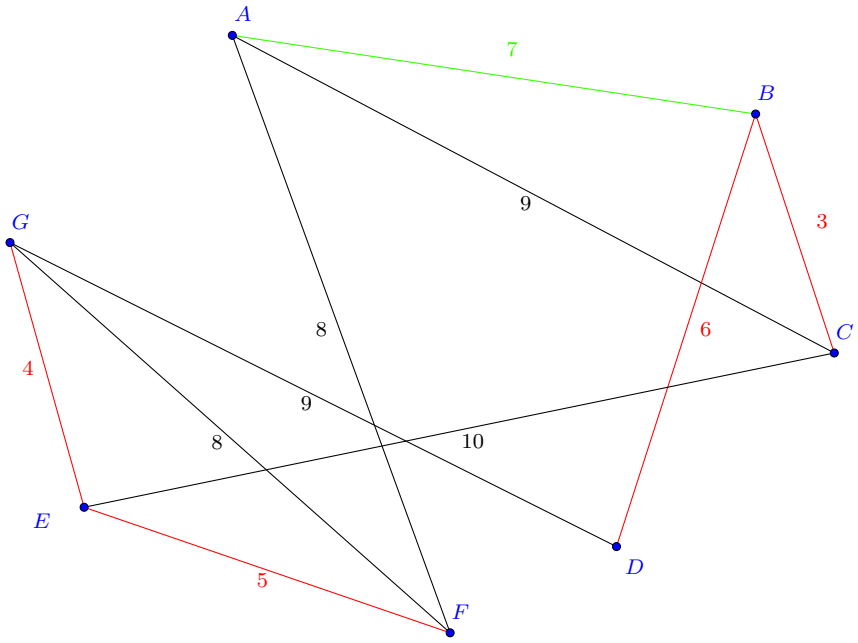
Árvore Geradora - 3º passo



Árvore Geradora - 4º passo

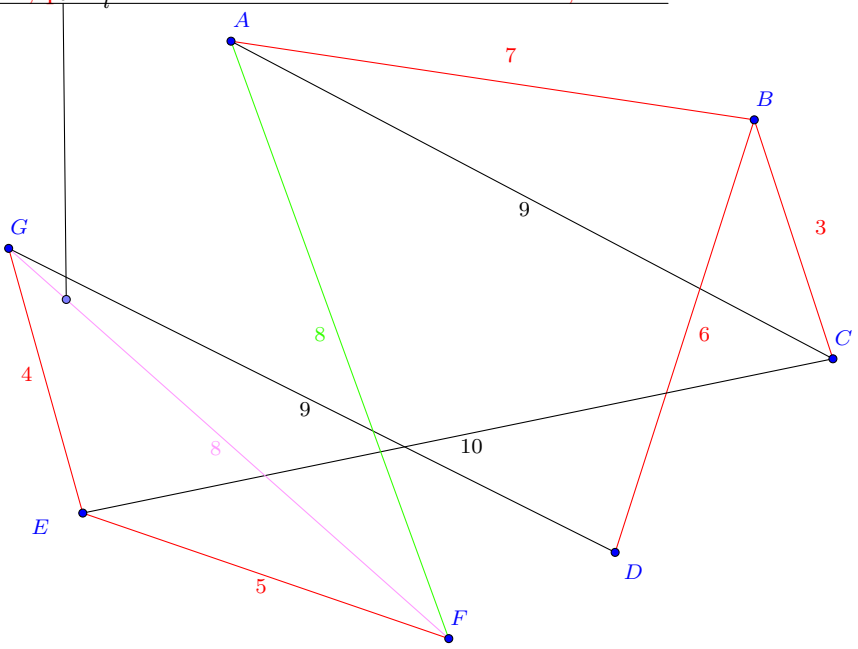


Árvore Geradora - 5º passo

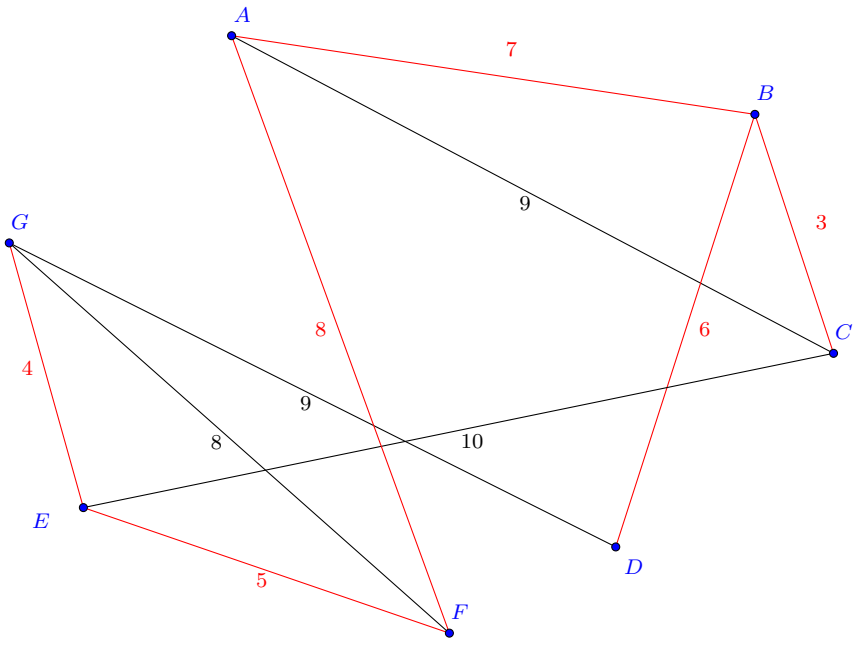


Árvore Geradora - 6º passo

- Não pode ser usada, pois fecha um ciclo entre os vértices E, F e G



Árvore Geradora Mínima - Resultado



Podemos observar a AGM deste grafo destacada em vermelho, e em preto estão as arestas do grafo original que não foram utilizadas na montagem da mesma.

2.2.4 Comparação das AGMs

Agora que já trabalhamos os dois algoritmos, podemos comparar seus resultados

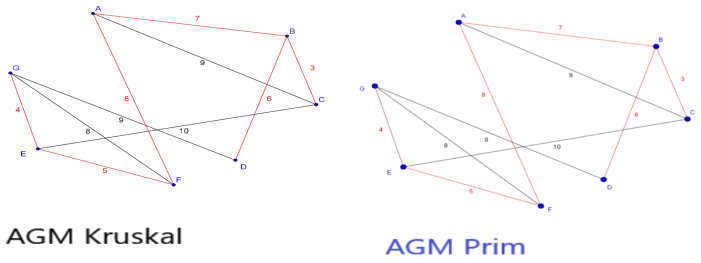


Figura 2.9: Comparação de AGMs

Podemos observar que as árvores obtidas pelos dois algoritmos são idênticas. Desse modo, apenas a facilidade de manipulação manual de Kruskal em relação a Primm (o fato da próxima aresta não ter de estar ligada a um dos vértices já atingidos) pesa a seu favor.

Em termos computacionais, a implementação é quase a mesma, mas esse tópico não faz parte do nosso objetivo, e por isso, não aprofudaremos o assunto.

Se tivermos de utilizar algum algoritmo para introduzir o assunto no ensino fundamental, Kruskal é o mais indicado por ser o mais intuitivo.

Capítulo 3

Análise de Caso

3.1 Apresentação

O caso analisado foi feito, utilizando-se uma proposição de uso do algoritmo de Kruskal, em uma turma de 8º ano da Escola Municipal Benevenuta Ribeiro, no Méier - Rio de Janeiro.

A tarefa consiste em duas etapas:

1ª Etapa

1ª - Apresentação de um mapa de rotas entre algumas cidades, e pede-se para que se encontre a rota com menor gasto para empresa, sem fechar um volta entre cidades; e que todas as cidades sejam atingidas.

A questão é colocada para os alunos como sendo um desfio lógico, em que após encontrar a rota pedida, os alunos descrevam como conseguiram chegar a tal resposta.

Após análise dos raciocínios obtidos, sugere-se aos alunos tentarem obter a mesma resposta, utilizando-se do raciocínio mais votado.

2ª - Apresentação da noção de grafos em geral; e do algoritmo de Kruskal, em particular, seguida da aplicação do algoritmo no mesmo grafo mostrado anteriormente na primeira etapa.

2ª Etapa

1ª - Realização de exercício utilizando o algoritmo de Kruskal, num grafo de maior complexidade.

2ª - Comparação dos resultados obtidos na 1ª e 2ª etapas

3.2 1ª Etapa

Foi apresentado aos alunos um grafo (sem que se use o termo), representando uma rede de estradas entre algumas cidades do estado do Rio de Janeiro.

A ideia é tentar construir uma rota que passe por todas as cidades do grafo. O exercício é apresentado como sendo um exercício de raciocínio lógico.

A turma participante estava com 24 alunos no momento.

Exercício:

"Dado as rotas entre as cidades do esquema abaixo, encontre uma rota que passe por TODAS as cidades deste, sem repetição. Assuma que todas as estradas são em mão-dupla. Destaque com um lápis de cor a rota escolhida"

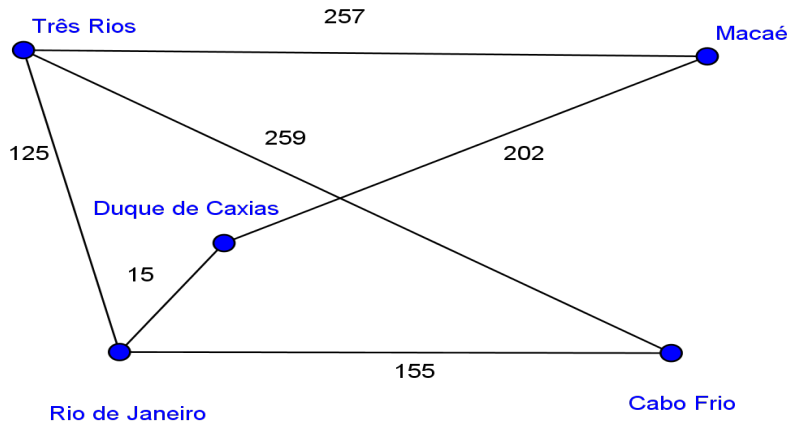


Figura 3.1: Rede de Estradas

Após um período de 10 minutos, foi perguntado sobre as possíveis soluções para o problema proposto. Surgiram 3 soluções, com as seguintes frequências:

$$\begin{bmatrix} Solução & Opção1 & Opção2 & Opção3 \\ Alunos & 10 & 9 & 5 \end{bmatrix}$$

A opção 1 foi a única que não apresentou um ciclo, embora não houvesse sido dito que não poderiam ser feitos ciclos.

Os alunos aparentam ver uma necessidade de que seja possível fazer este tipo de traçado, mesmo que este apenas acrescente mais uma estrada, e que todas as cidades poderiam ser atingidas da mesma forma, se esta fosse retirada.

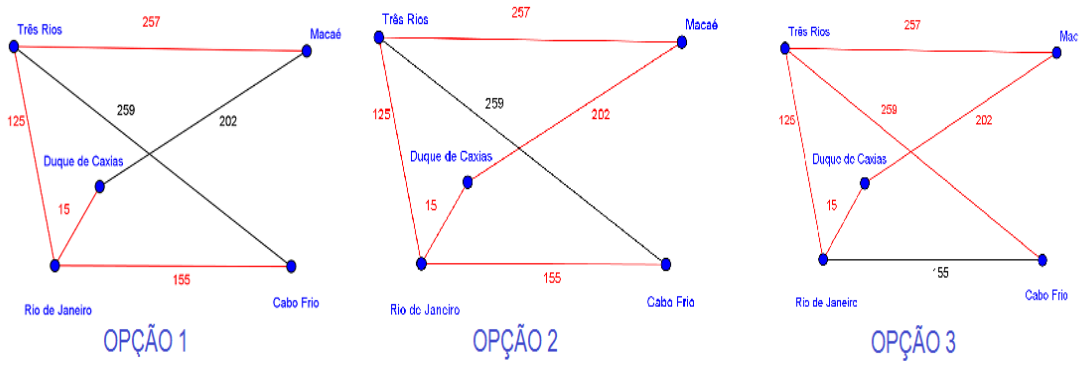


Figura 3.2: Soluções Sugeridas

Continuando com a atividade, foi pedido que os alunos encontrassem um trajeto, sem voltas (ciclos), de forma que o trajeto fosse o menor possível.

Exercício:
"Usando o esquema da atividade anterior, encontre agora uma rota nas mesma condições anteriores, mas sem que exista uma volta ou laço no trajeto."

Neste caso, apenas uma solução foi encontrada, embora os caminhos para chegarem à ela tenham sido diversificados.

Alguns alunos alegaram terem usado as rotas com menor valor, independentemente de as cidades estarem todas ligadas inicialmente.

Outros alunos disseram terem pego as rotas a medida que chegavam numa cidade do mapa, tentando construir a rota de forma direta.

Apenas dois alunos apresentaram uma solução diferente desta, mas não era a de menor distância, uma vez que o trajeto até Macaé era feito a partir de Três Rios, e não a partir de Duque de Caxias, aumentando assim a rota estabelecida.

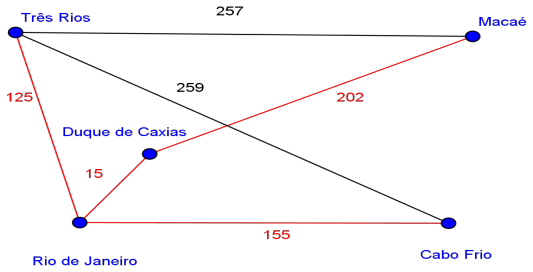


Figura 3.3: Solução Menor Rota

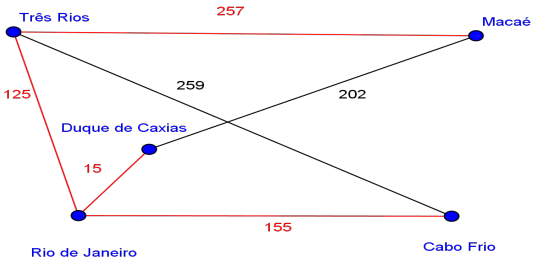


Figura 3.4: Solução Rota Discordante

3.3 2ª Etapa

Nesta etapa, foi proposto aos alunos encontrarem uma solução para o problema dado numa folha a parte, com uma explicação um pouco mais detalhada sobre o algoritmo de Kruskall.

A AGM solução é mais complexa do que a proposta no exercício anterior, mas também é encontrada sem maiores dificuldades.

Nesta etapa, os alunos apresentaram uma maior resistência ao trabalho, por conta do desenho da malha, mas após começarem a manipular a malha proposta, perceberam não ser mais difícil.

Após a aplicação do algoritmo, a maioria obteve a AGM abaixo. Apenas um dos grupos não conseguiu chegar ao mesmo resultado, em virtude de ser uma malha com maior número de componentes, e por conta da limitação do tempo de aula.

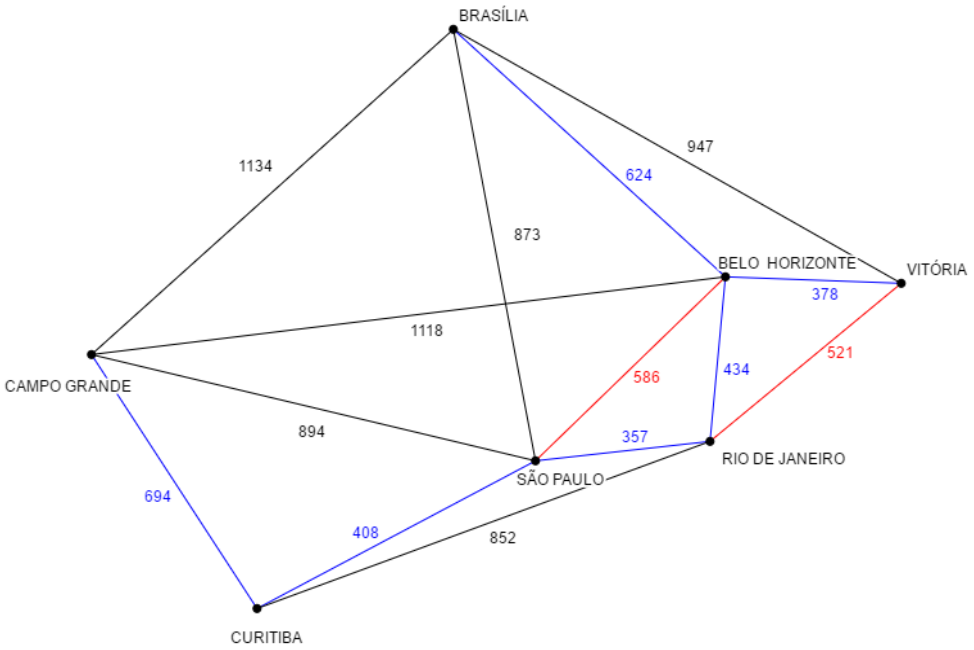


Figura 3.5: AGM Vôo entre as Capitais

Observe que a AGM (em azul) em si é bastante simples, mas como

existem várias arestas com peso baixo que não podem ser utilizadas, por formarem ciclos (duas estão destacadas em vermelho), isso fez com que alguns alunos sentissem uma maior dificuldade em encontrar rapidamente a AGM.

Após o encerramento do trabalho, recebi sugestões de utilizar variações do tema para as etapas iniciais, sem usar a palavra "rota", uma vez que esta traz a noção de locais pré-determinados para se passar.

Ao invés disso, foi sugerido que se usasse um exercício com termos do tipo "gasto mínimo para construção de uma estrada interligando cidades"ou "rede elétrica/telefônica". As ideias são análogas, e não tem a ideia de rota embutida.

Após as conclusões, segue uma cópia em anexo do trabalho proposto à turma.

Capítulo 4

Conclusões

Após observar os resultados obtidos nos trabalhos realizados pelos alunos da turma em que apliquei os exercícios, pude tirar algumas conclusões:

I Existe um bom campo de trabalho para a Teoria dos Grafos, tanto no Ensino Fundamental II como no Médio, apesar da insistência dos cursos de formação (Licenciatura) não colocarem o uso da Teoria dos Grafos como ferramenta para ensino de Matemática.

II No Ensino Fundamental II (8º e 9º anos), seu uso será na construção do processo de raciocínio, de elaboração de estratégias para a resolução de problemas, uma vez que um algoritmo nada mais é do que uma sequência de passos que nos levam a resposta procurada.

A descoberta desses passos virá numa etapa posterior do trabalho, uma vez que muitos deles não conseguem seguir um roteiro de forma eficiente.

O trabalho com algoritmos como esse pode ajudar nessa etapa.

III No 6º e 7º anos, as estruturas de AGMs podem ajudar como problemas de contextualização para situações de uso de adição de naturais (6º ano) e inteiros (7º ano); além dos conceitos de comparação de números naturais e inteiros

IV Ainda nos 6º e 7º anos, em virtude da má construção da estrutura em que as operações devem ser realizadas (muitos alunos mal sabem armar a conta), a ideia de seguir algoritmos pode ajudar a reconstrução deste tópico, uma vez que identificar os passos a serem seguidos para a realização de um cálculo é uma dificuldade visível em muitos alunos.

Exemplo: Muitos não sabem se devem começar a operar um cálculo de adição (ou subtração / multiplicação) da direita para a esquerda,

ou vice-versa.

V Também podemos usar essa estrutura de raciocínio algoritmizado, para fazer os alunos de 8º ano saberem (e compreenderem) as propriedades de diversas figuras planas, e saber diferenciá-las corretamente.

Um exemplo é quando se pergunta "Como definir um quadrado?". A resposta obtida, em 95 por cento dos casos é "Quatro lados iguais". Mas não entendem quando se deseja um losango e afirma-se que aquilo foi o que eles descreveram.

O uso deste modelo de raciocínio, em conjunto com um software de Geometria Dinâmica, como o GEOGEBRA, pode ajudar a melhorar essa situação, além de servir de base para a construção de estrutura semelhante ao chegarem na Geometria Euclidiana Espacial.

Deste modo, podemos finalizar verificando que são poucas (ou nenhuma) as razões, para que alguns tópicos de Teoria dos Grafos não sejam utilizados no ensino de Matemática nos níveis Fundamental e Médio e, por consequência, passem a fazer parte do currículo obrigatório da formação de professores, em nível de curso superior.

ANEXO

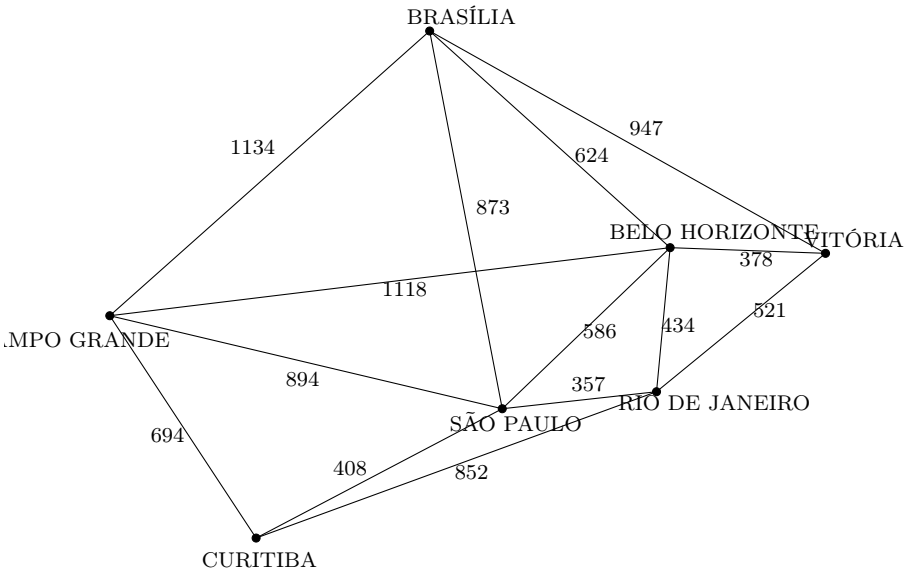
TEORIA DOS GRAFOS
ALGORITMO DE KRUSKALL

Trabalho com a malha aeroviária brasileira, com alunos do segundo segmento do ensino fundamental

Esse trabalho mostra a aplicabilidade do algoritmo de Kruskall em turmas do Ensino Fundamental - 2º segmento.
Uma vez que a explicação do algoritmo seja passada a eles, o processo em si torna-se simples, reduzindo o problema a um desafio de raciocínio lógico-matemático.
A ideia é encontrar o que chamamos de ÁRVORE GERADORA MÍNIMA, que é uma representação do roteiro de menor custo/distância/gasto de uma determinada situação.

Algoritmo de KRUSKALL
1 - Escolha a aresta de menor valor.
2 - Verifique se esta aresta não fecha uma volta entre as cidades já atendidas.
Se fechar um circuito, escolha nova arresta.
3 - Verifique se todas as cidades foram atendidas.
Se sim, o trabalho acabou. Se não, voltar ao passo 1.

Agora, aplique o algoritmo a malha aeroviária abaixo desenhada, e monte a AGM correspondente.
Use um lápis verde para destacar a rota traçada.
Após a conclusão da rota, some os valores correspondentes para dizer qual a distância percorrida nesta rota.
Na sua opinião (baseando-se na rota estabelecida acima), qual cidade seria a mais indicada para se colocar a sede de uma empresa que atenda todas essas cidades?



Bibliografia

Lloyd Norman L. Biggs; E. Keith Lloyd; Robin J. Wilson; Livro: Graph Theory, de 1976,, publicado por Clarendon Press, em Inglês.

Jurk01 Samuel Jurkiewicz; Paulo Oswaldo Boaventura Netto, Livro: Grafos: Introdução e Prática, de 2009, publicado pela editora Edgard Blücher, em Português, 1ª Reimpressão - 2011, com 156 páginas.

Boaventura Paulo Oswaldo Boaventura Netto, Livro: Teoria e Modelos de Grafos, de 1979, publicado pela editora Edgard Blücher, em Português, 1ª Edição, com 249 páginas.

Jurk02 Samuel Jurkiewicz, Livro: Grafos: Uma Introdução, de 2009, publicado por IMPA-SBM, em Português, 1ª Edição, com 119 páginas.

AGuloso Texto sobre ALGORITMO GULOSO, da Wikipédia, consultado em 30/04/2017, em Português, url = http://pt.wikipedia.org/wiki/Algoritmo guloso.

Prim Texto sobre ALGORITMO de PRIM, da Wikipédia, consultado em 28/05/2017, em Português, url = https://pt.wikipedia.org/wiki/Algoritmo de Prim.

Kruskal Texto sobre ALGORITMO de KRUSKAL, da Wikipédia, consultado em 28/05/2017, em Português, url = https://pt.wikipedia.org/wiki/Algoritmo de Kruskal.

Arvore Sem autor definido, Título: Árvores, consultado em 21/05/2017, em Português, na url = http://www.professeurs.polymtl.ca/michel.gagnon/Disciplinas/Bac/Grafos/Arvores/arvores.html

Sumário

Lista de Figuras