UNIVERSIDADE
FEDERAL RURAL
DE PERNAMBUCO

Universidade Federal Rural de Pernambuco
Departamento de Matemática

Mestrado Profissional em Matemática

ALGORITMO GULOSO

Camila Mendonça Morais

DISSERTAÇÃO DE MESTRADO

Recife
2014

Universidade Federal Rural de Pernambuco
Departamento de Matemática

Camila Mendonça Morais

ALGORITMO GULOSO

Trabalho apresentado ao Programa de Mestrado Profissional em Matemática do Departamento de Matemática da Universidade Federal Rural de Pernambuco como requisito parcial para obtenção do grau de Mestre em Matemática.

Orientador: *Prof. Dr. Thiago Dias*

Recife
2014

Dedico esta dissertação a minha família, namorado e a todos os amigos que me incentivaram e ajudaram de muitas formas, sofrendo e sorrindo junto comigo para que a sua concretização fosse possível.

AGRADECIMENTOS

Agradeço primeiramente a Deus pela força e coragem durante esta caminhada. Aos meus pais, Dolores e Carlos, e minha irmã, Carolina, que me apoiaram em todos os momentos. Agradeço aos meus demais familiares, especialmente aos meus avós, que com palavras de amor e apoio estiveram sempre presentes, acreditando em mim em todas as horas.

Ao meu namorado, Guilherme, que nunca me deixou desistir, sendo muito companheiro do começo ao fim, me ajudando, animando, e aconselhando em todos os momentos, com palavras de estímulo para que eu concretizasse meus objetivos, sendo sempre compreensivo e prestativo. Obrigada, meu amor.

A todos os meus amigos, que me incentivaram e compreenderam minha ausência, especialmente aos meus colegas de turma, Diogo, Walfrido e Fabiano, pelo apoio e por tornarem todos os dias de aula e estudo mais agradáveis.

Agradeço ao meu orientador, Thiago Dias, pela credibilidade e auxílio, pela enorme paciência, especialmente na execução do programa utilizado, e por apesar do pouco tempo disponível, estar sempre presente e solícito a me ajudar. A todos aqueles que de alguma forma contribuíram ou torceram pela concretização desta etapa da minha vida, obrigada.

A mente que se abre a uma nova ideia, jamais volta ao seu estado original.

—ALBERT EINSTEIN

RESUMO

O presente trabalho tem como objetivo principal estudar o Algoritmo Guloso, espécie de algoritmo de otimização, e algumas de suas aplicações, para posterior desenvolvimento de uma sequência didática a ser abordada com alunos do Ensino Médio. Neste estudo, a construção e lógica do algoritmo foram relacionadas a grafos e árvores, conceitos os quais foram previamente estudados e analisados como requisitos para a compreensão das propriedades e características do algoritmo. Primeiramente, fizemos uma síntese de como surgiu a Teoria dos Grafos; em seguida retratamos alguns conceitos sobre grafos em geral, como sua definição, propriedades, classificações e percursos. Na sequência, definimos árvores - um tipo especial de grafo - e estudamos alguns de seus principais teoremas fundamentais para a posterior compreensão do algoritmo, além de alguns métodos de codificação, como o código de Prüfer. Finalmente, definimos o Algoritmo Guloso, especialmente o algoritmo de Kruskal, utilizando uma situação prática para exemplificar sua aplicação. Após toda a fundamentação, desenvolvemos uma sequência didática para ser trabalhada em cinco aulas. Nesta sequência didática, atividades envolvendo grafos e árvores foram progressivamente realizadas, com questões contextualizadas como exercícios, para que na última aula da sequência o Algoritmo Guloso fosse definido e estudado, e os alunos capacitados a utilizá-lo na análise de um projeto, que seria utilizado como instrumento final de avaliação. Esta sequência didática tem como objetivo estimular o raciocínio lógico dos estudantes, além de introduzir estes conceitos em seu currículo escolar do Ensino Médio.

Palavras-chave: Algoritmo guloso; otimização; grafos; árvore gulosa; sequência didática.

ABSTRACT

This research aims to study the Greedy Algorithm, a type of optimization algorithm, and some of its applications, in order to develop a didactic sequence to be applied with secondary level students. In the study, the construction and logic of the algorithm were related to the graph and trees, concepts which were previously studied and analyzed as requisites to the comprehension of the properties and characteristics of the algorithm. Firstly, we synthetized the elaboration of the Theory of Graphs; then, we presented some concepts about graphs in general, such as its definition, properties, classifications and percusses. Next, we defined trees - a special type of graph - and studied some of its fundamentals theorems for the comprehension of the algorithm, as well as some methods of codification such as the Prüfer code. Finally, we defined the Greedy Algorithm, specially the Kruskal algorithm, using a practical setting in order to exemplify its application. After the theoretical fundaments, we develop a didactical sequence to be applied in five classes. In this didactical sequence, activities which involve graphs and trees were progressively applied, with contextualized questions such as exercises in such a way that in the last class of the sequence, the Greedy Algorithm could be defined and studied, and the students were able to use them to analyze a project, which would be used as a final instrument of evaluation. This didactical sequence aims to stimulate the student's logical reasoning, as well as to introduce these concepts in their school curriculum on secondary level.

Keywords: Greedy Algorithm; optimization; graph; greedy tree; didactical sequence.

SUMÁRIO

LISTA DE FIGURAS

INTRODUÇÃO

Existem muitas situações ou problemas matemáticos que parecem de simples resolução, mas que na prática há uma grande dificuldade para sistematizar as informações fornecidas e organizá-las de maneira que possa levar a algum método de resolvê-los.

Um exemplo é o seguinte problema: Um motorista de caminhão de lixo deve recolher todos os resíduos residenciais de um determinado bairro. É possível que ele colete esse lixo passando por cada esquina uma única vez e retornando ao mesmo lugar que partiu?

Esse é um típico caso em que a utilização de grafos esquematiza de forma clara e direta todas as possibilidades, levando à visualização de uma possível estratégia para solucionar o problema. Além disso, existem algoritmos que foram criados para indicar a melhor escolha que atenda a determinado objetivo. Algoritmos deste tipo podem ser denominados como de otimização, pois ao lidar com uma sequência de passos consegue indicar qual é a melhor opção entre as possibilidades. Um exemplo é o algoritmo guloso, objeto de estudo deste trabalho, que a cada passo escolhe a opção que parece ser a melhor naquele determinado momento com o intuito de satisfazer o objetivo proposto.

Pelo fato das escolhas feitas por este algoritmo dependerem apenas da situação presente, ou seja, da informação disponível naquele momento, em alguns casos ele não possibilita a escolha ótima, o que vai depender do objetivo de cada situação. Entretanto, ele é eficiente em diversos casos, e são considerados de certo modo simples de criar, já que uma vez que a opção foi escolhida não se pode voltar atrás, não tendo, portanto, a necessidade de rever as possibilidades. A denominação *guloso* vem do fato do algoritmo selecionar sempre a melhor opção do momento, escolher sempre o melhor "pedaço", sem pensar em consequências futuras. No decorrer deste trabalho veremos, de maneira geral, como a lógica do algoritmo guloso é criada e desenvolvida, estando associado à utilização de grafos e árvores, os quais serão definidos e analisados nos capítulos iniciais.

No capítulo 1 são retratados alguns conceitos básicos sobre grafos em geral, suas definições, propriedades e classificações, assim como sua origem, com uma breve síntese sobre a Teoria dos Grafos e seu processo de evolução. Com ênfase em alguns teoremas mais importantes, através de exemplos práticos ressaltaremos como a utilização da Teoria de grafos auxilia na resolução e simplificação de problemas.

No capítulo 2 falamos sobre possíveis percursos em um grafo, tais como caminhos, ciclos e passeios, além dos conceitos de conexidade, subgrafos e componentes conexas. Analisando os estudos de Euler relativos a grafos, discutiremos sobre passeios Eulerianos e suas propriedades.

O capítulo 3 define o conceito de árvores, tipo específico de grafo, analisando algumas de suas propriedades e teoremas considerados mais importantes como requisitos para o posterior entendimento do algoritmo guloso. Ainda neste capítulo serão vistas técnicas de armazenamento de árvores, onde o código de pai e o código de Prüfer serão definidos

e trabalhados.

Após o estudo destes conteúdos necessários para o entendimento do algoritmo guloso, ele será retratado no capítulo 4, onde, através de um exemplo prático, chegaremos a sua definição e posterior execução. Exporemos dois métodos para executar o algoritmo guloso: o método de Prim e o de Kruskal. Este último será estudado detalhadamente. Ainda neste capítulo, o problema do Caxeiro Viajante é citado, fazendo analogia a problemas cotidianos.

Finalizando este trabalho, uma sequência didática será proposta no último capítulo, com o objetivo de trabalhar com alunos do Ensino Médio o conteúdo desenvolvido. Será realizada em cinco aulas de 50 minutos cada, iniciando com o estudo de grafos e árvores, e concluindo com a aplicação do algoritmo guloso na análise de um projeto, que será utilizado como instrumento final de avaliação.

CAPÍTULO 1

GRAFOS

1.1 TEORIA DOS GRAFOS

Há muito tempo, os cidadãos de Königsberg (onde hoje fica Kalingrado, Rússia) propuseram um desafio. A cidade era dividida em quatro distritos por braços do rio Preguel, que eram conectados por 7 pontes (Figura 1.1). Seria possível realizar um trajeto em todos os distritos passando por todas as pontes uma única vez?

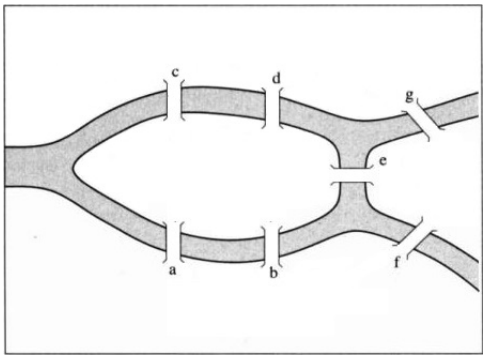


Figura 1.1 Pontes de Königsberg

Em 1736, o matemático Euler publicou um artigo que demonstrava que era impossível realizar tal passeio. Esse resultado é considerado como o primeiro teorema da Teoria dos Grafos, ainda que Euler não tenha utilizado essa terminologia. Contudo, na época esse resultado foi visto pela comunidade científica como uma espécie de solução a uma charada matemática, não sendo a ele dada a devida importância, o que contribuiu para que essa teoria ficasse estagnada por mais de cem anos.

Finalmente, em 1847, Gustav Robert Kirchhoff, cientista da cidade de Königsberg, utilizou modelos de grafos ao estudar circuitos elétricos, criando a Teoria das árvores. Isso abriu precedentes para que outros cientistas começassem a notar a provável aplicabilidade desta teoria, como o irlandês Willian Rowan Hamilton, que definiu como *grafo hamiltoniano* aquele que contém um percurso fechado passando uma única vez por cada vértice; ou o britânico Arthur Cayley, que utilizou a ideia de árvores para aplicações como a enumeração dos isômeros dos hidrocarbonetos alifáticos saturados, em química orgânica.

Após o desenvolvimento dos computadores, a partir de 1970, a teoria dos Grafos obteve um grande avanço, especialmente com o surgimento de softwares para microcomputadores, como o MS Project, criado pela Microsoft. Este avanço tecnológico desencandeou o surgimento de publicações referentes a algoritmos de grafos, o que possibilitou uma utilização aplicada da teoria.

Sendo atualmente uma das áreas mais importantes da Matemática discreta, a Teoria dos Grafos tem sido aplicada a muitas áreas (Informática, Investigação Operacional, Economia, Sociologia, Genética, etc.), pois um grafo constitui o modelo matemático ideal para o estudo das relações entre objetos discretos de qualquer tipo.

1.2 CONCEITOS BÁSICOS E DEFINIÇÕES

Muitas situações podem ser descritas e representadas convenientemente por um esquema ou diagrama que consiste em um conjunto de pontos e linhas que ligam aos pares alguns desses pontos. Por exemplo, a figura 1.2 abaixo esboça um grafo das pontes de Königsberg, onde os distritos estão representados por pontos e letras, e as pontes por linhas que ligam esses pontos.

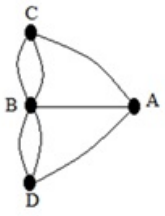


Figura 1.2 Grafo das pontes de Königsberg

Definição 1.1. *Um grafo G consiste num conjunto finito e não vazio $V(G)$ de elementos denominados vértices ou nós, que podem ou não estar ligados aos pares por arestas. O conjunto finito formado por estas arestas é denominado $A(G)$. Portanto, escreve-se $G = (V, A)$ para indicar que o grafo G tem o conjunto de vértices V e o conjunto de arestas A.*

A forma adotada para representar as arestas (reta, curva, etc.) não é relevante, importando apenas o par de nós que ela conecta. Uma aresta que liga os vértices x e y, por exemplo, é o conjunto $\{x, y\}$, que é subconjunto de V, ou o par ordenado (x, y) pertencente a V x V, ou ainda simplesmente a *aresta* xy.

Algumas possibilidades, como por exemplo se uma aresta pode conectar um nó a si mesma (*laço*), ou se mais de uma aresta pode conectar o mesmo par de nós (*arestas paralelas*) serão consideradas neste trabalho de acordo com os objetivos que se deseja alcançar. Vamos analisar o exemplo das pontes de Königsberg. Existe a situação em que mais de uma ponte liga o mesmo par de distritos, então é conveniente que se adote arestas paralelas. Mas não teria sentido uma ponte ligando um distrito a si mesmo, então neste caso não há necessidade de laços. Quando um grafo apresenta arestas paralelas ou laços, pode ser chamado de *multigrafo*. Já quando em um determinado grafo não há nenhum destes elementos, ele é denominado *grafo simples*.

Na figura seguinte temos um exemplo de multigrafo, onde as arestas a_1 e a_2 são paralelas, pois ambas ligam o mesmo par de vértices (1 e 2); e a aresta a_3 é um laço, pois conecta o vértice 2 a ele mesmo.

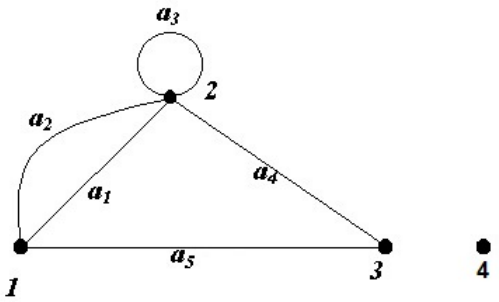


Figura 1.3 Exemplo de multigrafo

Um grafo é chamado de *dirigido, direcionado* ou *dígrafo* quando suas arestas são orientadas, como por exemplo, grafos utilizados para representar estradas de sentido único conectando cidades, ou quem sorteou quem em um amigo secreto. (Figura 1.4)

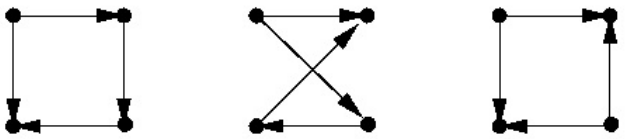


Figura 1.4 Exemplos de grafos dirigidos

De volta a figura 1.2, podemos observar que *três* das pontes passam pelo distrito A, *cinco* das pontes passam pelo distrito B, *três* pelo distrito C e *três* pelo distrito D. Denominamos, portanto, que os *graus dos vértices* que representam os distritos A, B, C e D são 3, 4, 3 e 3, respectivamente, pois é a quantidade de arestas que saem de cada vértice. Representamos estes graus como g(A) = 3, g(B) = 5, e assim por diante.

Definição 1.2. *O grau $g(v)$ de um vértice v, sem laços, é equivalente ao número de arestas que conectam v a outros vértices, ou seja, a quantidade de arestas que incidem em v.*

No caso em que v possui um laço, seu grau é aumentado em dois, pois é como se a aresta que forma o laço incidisse em v duas vezes.

Ao somar os graus de cada vértice, uma aresta é contabilizada por ambos os vértices os quais ela incide, ou seja, cada aresta é contabilizada duas vezes. Daí segue a seguinte proposição:

Proposição 1.1. *A soma dos graus de todos os vértices de um grafo é igual ao dobro de seu número de arestas.*

Podemos concluir, portanto, que a soma dos graus de todos os vértices de qualquer grafo é um número par, já que corresponde ao dobro do número de arestas. Se omitirmos desta soma todos os termos pares (os graus dos vértices cuja quantidade de arestas

incidentes é par), ainda obteremos um número par, que corresponde à soma dos termos ímpares. Mas para que a soma de termos ímpares resulte num número par, é necessário que a quantidade destes termos seja par. Ou seja, em um grafo há uma quantidade par de vértices cujo grau é ímpar.

Corolário 1.1. *Em um grafo, o número de vértices com grau ímpar é par.*

Do corolário acima, podemos tirar as seguintes conclusões: Se um grafo tem um número ímpar de vértices, então o número de vértices com grau par, é ímpar. Isso é facilmente dedutível, pois o total de vértices de um grafo é a soma dos vértices de grau par com os vértices de grau ímpar. Se o grafo tem um número ímpar de vértices ao total, e todo grafo possui um número par de vértices de grau ímpar, então o número de vértices com grau par é ímpar. Da mesma forma, se um grafo possui um número par de vértices, então o número de vértices com grau par, é par.

Proposição 1.2. *Todo grafo simples possui pelo menos dois vértices com o mesmo grau.*

Demonstração. Seja G um grafo com n vértices. Cada um desses vértices pode ter seu grau variando de 0 a $n-1$, ou seja, existem n possibilidades para o grau de cada vértice. Entretanto, se um dos vértices tem grau igual a $n-1$, então ele está conectado a todos os demais vértices, ou seja, não poderia existir um vértice de grau 0. Logo, em um grafo G de n vértices ou existe algum vértice de grau $n-1$ ou algum vértice de grau 0, mas não os dois ao mesmo tempo. Isso implica que os n vértices existentes em G teriam apenas $n-1$ opções para seu grau, logo, pelo princípio da casa dos pombos, pelo menos dois vértices teriam o mesmo grau. ■

Dizemos que dois vértices são *adjacentes* quando são conectados por uma mesma aresta (por exemplo os vértices 1 e 3 da figura 1.3), e vértices adjacentes ao mesmo nó x são chamados de seus *vizinhos* (o vértice 3 da figura 1.3 tem como vizinhos os vértices 1 e 2, por exemplo). Chamamos de nó ou vértice *isolado* aquele que não possui aresta incidindo sobre ele, ou seja, tem grau zero (como o vértice 4 da figura 1.3).

Os *grafos vazios*, que são os mais simples, não possuem arestas, ou seja, todos os seus vértices têm grau zero; já um grafo simples que possui todas as arestas possíveis é denominado *grafo completo* (Figura 1.5). Se um grafo G possui n vértices e é completo, então cada um de seus vértices tem grau $n-1$. Como a soma dos graus de todos os vértices é igual ao dobro do número de arestas (Proposição 1.1), então um grafo completo de n vértices possui $\frac{n(n-1)}{2}$ arestas.

Dizemos que um grafo é o *complemento* de outro se ambos possuírem o mesmo conjunto de nós, e as arestas de um não estiverem contidas no outro. Então um grafo vazio, por exemplo, é o *complemento* de um grafo completo e vice-versa.

Já um grafo G de n vértices, onde apenas um destes vértices possui grau igual a $n-1$ e todos os demais vértices possuem grau 1 é denominado *estrela*.

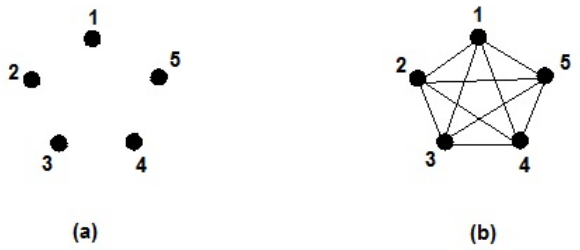


Figura 1.5 (a) Exemplo de grafo vazio; (b) Exemplo de Grafo completo

CAPÍTULO 2

PERCURSOS

2.1 PASSEIOS, CICLOS E CAMINHOS

Considerando um grafo G, o percurso que consiste em uma sequência de vértices consecutivos (conectados por alguma aresta) é denominado *passeio*. Contudo, se mais de uma aresta pode conectar o mesmo par de nós, que é o caso de G conter arestas parelelas, há a necessidade de especificação de qual aresta está sendo utilizada para formar o percurso.

Definição 2.1. *Um passeio P em um grafo G é uma sequência finita e não vazia do tipo $(v_0, a_1, v_1, a_2, ..., a_k, v_k)$, cujos termos são alternadamente vértices v_i e arestas a_i, onde para todo i, $1 \leq i \leq k$, os extremos das arestas a_i são os vértices v_{i-1} e v_i, sem que necessariamente todos os vértices sejam percorridos.*

O passeio mais curto possível é aquele de um único vértice v_0. O primeiro e o último vértice do percurso são denominados *extremidades* do percurso. Quando as extremidades não coincidem, define-se que o passeio é aberto. E, neste caso, se o passeio não contiver nenhum vértice repetido, ele recebe uma denominação especial.

Definição 2.2. *Um caminho C em um grafo G é um passeio ou sequência finita e não vazia do tipo $(v_0, a_1, v_1, a_2, ..., a_k, v_k)$, cujos termos são alternadamente vértices v_i e arestas a_i, os quais não se repetem no percurso.*

Exercício 2.1. *Seja G um grafo de n vértices, onde vértices consecutivos são conectados por uma única aresta, formando um caminho que passa por todos os vértices do grafo. Mostre que G possui $n - 1$ arestas.*

Já no caso em que o primeiro e último vértices do passeio são os mesmos, ou seja, $v_0 = v_k$, o o passeio é definido como *fechado*.

Definição 2.3. *Um ciclo C em um grafo G é uma sequência finita e não vazia do tipo $(v_0, a_1, v_1, a_2, ..., a_k, v_k)$, cujos termos são alternadamente vértices v_i e arestas a_i, onde o último vértice é igual ao primeiro, ou seja, $v_0 = v_k$, e v_0 e v_k são os primeiros vértices do passeio a coincidirem.*

Perceba que ao conectarmos as extremidades de um caminho, ou seja, conectando o último vértice ao primeiro, obtemos um ciclo. Obviamente, tanto em caminhos quanto em ciclos, os vértices do grafo podem estar em diferentes disposições, sendo possível que arestas se intersectem (Figura 2.1).

O *comprimento* de um caminho, passeio ou ciclo é o seu número de arestas. Um ciclo de comprimento k é usualmente chamado de k-ciclo.

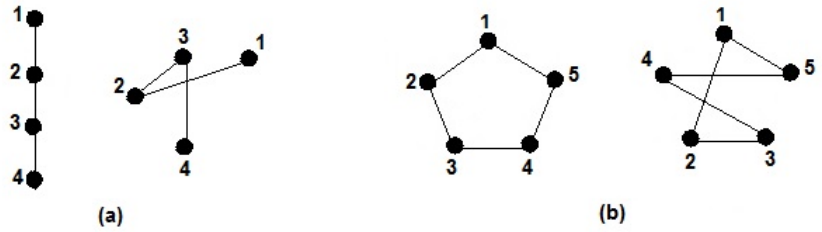


Figura 2.1 (a) Exemplos de caminhos; (b) Exemplos de ciclos

Exemplo 2.1. *Encontre todos os caminhos cujo complemento é um caminho.*

Resolução. Podemos começar o exercício fazendo um teste para alguns caminhos (Figura 2.2).

Número de vértices	Caminho	Complemento
1		
2		
3		
4		
5		

Figura 2.2 Tabela de caminhos e seus complementos

O caminho com 1 vértice, por exemplo, tem como complemento ele mesmo, ou seja, seu complemento é um caminho;

O caminho contendo 2 vértices, que possui 1 aresta, tem como complemento o grafo com 2 vértices e nenhuma aresta, que não é um caminho;

O caminho contendo 3 vértices, que possui 2 arestas, tem como complemento o grafo com 3 vértices e apenas 1 aresta, ou seja, não é um caminho;

O caminho contendo 4 vértices, que possui 3 arestas, tem como complemento o grafo com 4 vértices e 3 arestas, que é um caminho;

O caminho contendo 5 vértices, que possui 4 arestas, tem como complemento o grafo com 5 vértices e 6 arestas, que não é um caminho.

Continuando o raciocínio, intuitivamente chegamos à conclusão que apenas os caminhos com 1 e 4 vértices terão como complemento um caminho.

Contudo, para ter certeza que apenas esses dois caminhos possuem essa característica, vamos formalizar a resolução. Seja C um caminho com n vértices, logo C possui $n-1$ arestas. Seja $\bar{C}$ o complemento de C. Então a soma da quantidade de arestas de C e $\bar{C}$ é o máximo de arestas que um grafo poderia ter, que é dado por $\frac{n(n-1)}{2}$. Assim, denominando A_C e $A_{\bar{C}}$ a quantidade de arestas de C e $\bar{C}$, respectivamente, temos a seguinte equação:

$$A_C + A_{\bar{C}} = \frac{n(n-1)}{2}.$$

Sabemos que $A_C = n-1$. Para que o complemento $\bar{C}$ de C também seja um caminho, $\bar{C}$ deve possuir $n-1$ arestas, ou seja, $A_{\bar{C}} = n-1$. Substituindo estes dados na equação, temos:

$$n-1+n-1 = \frac{n(n-1)}{2} \Rightarrow 2n-2 = \frac{n(n-1)}{2} \Rightarrow 4n-4 = n^2-n \Rightarrow n^2-5n+4 = 0.$$

Resolvendo a equação, encontramos que $n = 1$ ou $n = 4$. Ou seja, os únicos caminhos cujo complemento também é um caminho são o de 1 vértice e o de 4 vértices. ■

Proposição 2.1. *Se G é um grafo cujo grau de todos os seus vértices é pelo menos dois, então G contém um ciclo.*

Demonstração. Se G possui laços ou arestas paralelas, o resultado é óbvio. Vamos supor, então, que G é um grafo simples de n vértices. Seja P um passeio em G começando pelo vértice v_1. Percorrendo P, vamos escolher v_2 como vértice adjacente a v_1; v_3 como vértice adjacente a v_2; e assim por diante, de modo que para cada vértice v_i, com $1 \leq i \leq n-1$, temos v_{i+1} como seu vértice adjacente (o que pode ser garantido pelo fato do grau de qualquer um dos vértices de G ser no mínimo 2). Assim, temos que o passeio P é dado por $v_1v_2v_3...$

Como G possui um número finito de vértices, em algum momento P passará por algum vértice que já havia participado do passeio anteriormente. Se v_k é o primeiro vértice a ser escolhido mais de uma vez, então as duas ocorrências de v_k e as partes do passeio entre elas formam um ciclo em G. ■

Um ciclo é definido como *hamiltoniano* se ele contém todos os vértices do grafo ao qual pertence, ou seja, é um passeio fechado que passa por todos os vértices uma única vez. Se um determinado grafo G contém um ciclo hamiltoniano, ele é denominado *grafo hamiltoniano*. Este nome se deve ao matemático irlandês Sir William Rowan Hamilton (1805-1865).

2.2 GRAFOS E COMPONENTES CONEXOS

Definição 2.4. *Um grafo H é chamado de subgrafo de um grafo G ($H \subseteq G$) se todos os vértices de H são também vértices de G, e todas as arestas de H são também arestas de G, ou seja, $V(H) \subseteq V(G)$ e $A(H) \subseteq A(G)$. Nesse caso, também pode-se dizer que G é supergrafo de H.*

Um subgrafo H de G pode ser obtido removendo-se algumas das arestas e vértices de G. Obviamente, ao remover-se um vértice, automaticamente são removidas todas as arestas que o conectam a outro vértice.

Quando $H \subseteq G$ e $H \neq G$, então H é um subgrafo próprio de G, e escreve-se $H \subset G$. Definimos H como grafo *gerador* de G, quando além de ser um subgrafo de G, dentre os vértices de H se encontram todos os vértices de G, ou seja, $H \subseteq G$ e $V(H) = V(G)$. O subgrafo H é dito *maximal* em relação a certa propriedade K se, além de possuir a propriedade, nenhum outro subgrafo J de G, tal que $H \subset J \subset G$, também a possua.

Definição 2.5. *Quando em um grafo G, quaisquer dois vértices u e v puderem ser conectados por um caminho C, com extremidades u e v, onde C é subgrafo de G, então G é denominado grafo conexo. Caso contrário, é desconexo.*

Não existe relevância, quanto à conexidade, se dois vértices são conectados por um caminho ou por um passeio, pois se eles podem ser conectados por um passeio, então podem ser conectados por um caminho.

Considere um passeio P em um grafo G com 8 vértices v_i, com $1 \leqslant i \leqslant 8$, por exemplo, onde P é representado pela sequência de vértices $v_1v_2v_3v_1v_2v_3v_4v_5v_6v_7v_4v_5v_7v_8$ (Figura 2.3 abaixo).

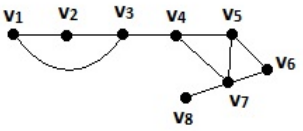


Figura 2.3 Passeio P

Como o vértice v_1 é o primeiro que se repete, vamos eliminar do passeio todos os vértices que estão entre ele e sua repetição, incluindo a própria repetição. Após esse procedimento, obtemos o passeio dado por $v_1v_2v_3v_4v_5v_6v_7v_4v_5v_7v_8$. Agora o próximo vértice a se repetir é o v_4. Aplicando o mesmo procedimento, eliminando os vértices que estão entre ele e sua repetição, obtemos o passeio $v_1v_2v_3v_4v_5v_7v_8$, que, como podemos perceber, é um caminho de v_1 a v_8. Este procedimento pode ser aplicado a qualquer passeio. Enunciando o resultado, temos:

Proposição 2.2. *Se dois vértices quaisquer são conectados por um passeio, então também podem ser conectados por um caminho.*

Se removermos uma aresta uv de um grafo conexo G, o grafo resultante G' pode ou não ser conexo. Um exemplo simples onde G' não seria conexo, é quando G é um caminho e tem uma de suas arestas removidas, resultando num grafo desconexo.

Em contrapartida, temos a seguinte proposição:

Proposição 2.3. *Se G é um grafo conexo que contém um ciclo C, ao removermos uma das arestas de C o grafo remanescente G' ainda será conexo.*

Demonstração. Considere dois nós u e v arbitrários pertencentes a G e suponha que retiramos do ciclo C uma aresta a. Se G é conexo, podemos garantir que ele contém um caminho P que conecta u a v. Se a aresta retirada a não pertencer a P, então os vértices u e v ainda estarão sendo conectados por P.

Supondo agora que a aresta a, que pertence ao ciclo C, também pertença ao caminho P, onde $a = xy$. Vamos considerar que percorrendo o caminho de u a v, o vértice x venha antes do y. Ao retirar a aresta a ou xy, ainda existirá um caminho de u a x e um caminho de y a v. Porém, também existirá um caminho de x a y, pois ambos os vértices pertencem ao ciclo C. Logo, se existe um caminho de u a x, um caminho de x a y e um de y a v, então existe um caminho ou um passeio de u a v para qualquer escolha de u e v. Caso o percurso seja um passeio, então podemos garantir que existe caminho de u a v (Proposição 2.2). Assim, o grafo resultante G' é conexo. ■

Diz-se que H é um *subgrafo conexo maximal* de G, se além de H ser um subgrafo conexo de G, não existir nenhum supergrafo próprio de H que seja subgrafo conexo de G.

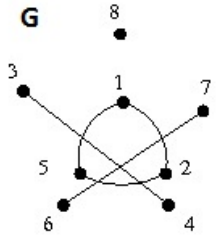


Figura 2.4 Exemplo de grafo desconexo

Na figura 2.4 acima, temos como exemplo o grafo G, desconexo, que é formado por quatro subgrafos conexos maximais. Portanto, nada impede que G possua mais de um subgrafo conexo maximal. Assim, se G é um grafo desconexo, então poderá ser subdividido em subgrafos conexos, pois até o subgrafo que consiste de um único vértice (e nenhuma aresta) é conexo. Um grafo formado por três vértices e nenhuma aresta, por exemplo, é composto por três subgrafos distintos, onde cada um dos vértices seria um desses subgrafos. Os subgrafos conexos maximais de um grafo G são denominados de *componentes conexas*.

Ainda na figura 2.4, G é formado por quatro componentes conexas distintas: os vértices 1, 2 e 5 e as arestas que os conectam; os vértices 3 e 4 e a aresta que os conecta; os vértices 6 e 7 e sua respectiva aresta incidente; e o vértice 8, como pode ser melhor visualizado na figura 2.5 a seguir.

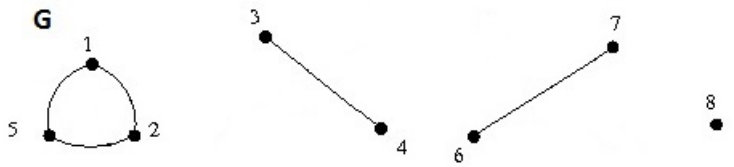


Figura 2.5 Exemplo de grafo desconexo com componentes conexas destacadas

Consequentemente, podemos concluir que todo vértice de um grafo G pertence a

alguma componente conexa, pois ainda que seu grau fosse *zero* ele mesmo seria uma componente conexa. Ademais, cada vértice ou aresta de G está contido em uma única componente conexa, pois do contrário, duas componentes conexas distintas teriam um vértice ou aresta em comum, e sua união resultaria em um único subgrafo conexo maximal, o que contraria a maximalidade das componentes conexas em relação à inclusão. Portanto, concluímos que componentes diferentes de G não possuem vértices nem arestas em comum.

Definição 2.6. *Um grafo H é uma componente conexa de um grafo G se H é subgrafo conexo maximal de G, ou seja, qualquer outro subgrafo de G que contenha H é desconexo. Duas componentes conexas distintas não possuem elementos em comum.*

Exemplo 2.2. *Em um país existem 9 aeroportos numerados de 1 até 9. Existe voo direto do aeroporto i para o aeroporto j se $3|ij$, com $ij = 10i + j$. Sabendo disso, existe voo (possivelmente com escalas) do aeroporto 1 para o aeroporto 9?*

Resolução. Do aeroporto 1, os possíveis voos diretos são para os aeroportos 2, 5 e 8, pois 3 divide 12, 15 e 18, e não divide ij quando $i = 1$ e $j = 3, 4, 6, 7 ou 9$, ou seja, não divide 13, 14, 16, 17 e 19.

Do aeroporto 2, os possíveis voos diretos são para os aeroportos 1, 4 e 7, pois 3 divide 21, 24 e 27.

Seguindo este raciocínio, para encontrar todos os voos diretos de i para j basta verificar todos os múltiplos de 3 de dois algarismos distintos que vão de 1 a 9. São eles: 12 e 21, 15 e 51, 18 e 81, 24 e 42, 27 e 72, 36 e 63, 39 e 93, 45 e 54, 48 e 84, 57 e 75, 69 e 96, 78 e 87.

Vamos representar o problema utilizando um grafo, onde os aeroportos serão os vértices, e quando existir voo direto entre dois aeroportos, eles serão conectados por uma aresta. Como o critério de divisibilidade por 3 é que a soma dos algarismos seja um múltiplo de 3, se $3|ij$ então $3|ji$, pois $i + j = j + i$, como pode ser observado. Assim, no grafo cada aresta representa possíveis voos diretos de ida e volta entre os vértices (aeroportos) que ela conecta. Então os voos do aeroporto 1 para o 2 e do aeroporto 2 para o 1, por exemplo, serão representados por uma única aresta que conectará os vértices 1 e 2. Desse modo, obtemos o seguinte grafo:

O grafo obtido, como pode ser visto, não é conexo, pois ele possui duas componentes conexas, o que pode ser melhor observado na seguinte figura:

Assim, para o aeroporto 9 só existem voos dos aeroportos 3 e 6, ou seja, não existe voo (nem com escalas) do aeroporto 1 para o aeroporto 9, pois os vértices 1 em 9 estão em diferentes componentes conexas. ■

Exemplo 2.3. *Prove que ao conectar dois vértices u e v pertencentes a um grafo G por uma aresta, cria-se um ciclo C se, e somente se, u e v estiverem na mesma componente conexa.*

Resolução. $\Rightarrow$ Vamos supor, por absurdo, que u e v estejam em duas componentes conexas distintas, ou seja, G é desconexo e u e v não estão ligados por caminho em G.

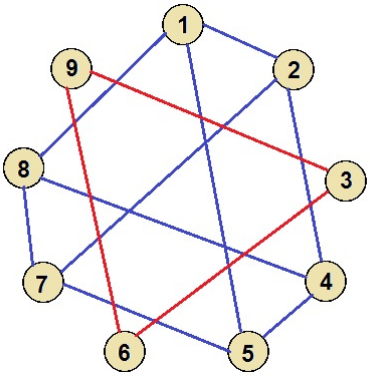


Figura 2.6 Problema do aeroporto

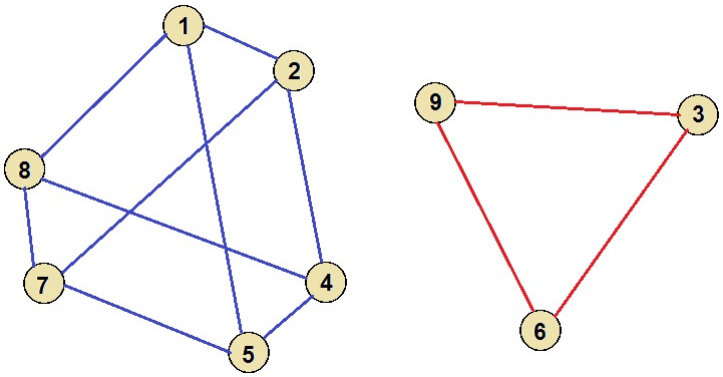


Figura 2.7 Problema do aeroporto (2)

Se adicionarmos a aresta uv, um ciclo C é formado no grafo resultante G'. Retirando a aresta uv pertencente a C, o grafo volta à condição inicial, sendo que u e v estariam ligados por um caminho (pela proposição 2.3), chegando a uma contradição. Logo, se ao conectar u e v em G um ciclo C é formado, então u e v estão na mesma componente conexa.

$\Leftarrow$ Agora, por hipótese, u e v estão em uma mesma componente conexa de G. Temos que provar que ao conectar u e v por uma aresta uv, um ciclo C será formado. Se u e v estão na mesma componente conexa, que é um subgrafo conexo, então consequentemente u e v podem ser conectados por um caminho M em G (Definição 2.5). Como existe um caminho M de u a v, ao adicionar a aresta uv existirão dois caminhos distintos de u a v, ou seja, um ciclo C será formado. ■

2.3 PASSEIOS EULERIANOS

Após o desafio das pontes do rio Preguel, proposto pelos cidadãos de Königsberg e solucionado por Euler, que deu origem à Teoria dos Grafos, foi definido como *passeio Euleriano*

aquele que passa por cada aresta do grafo exatamente uma vez, podendo ser aberto ou fechado, e *Grafo Euleriano* o grafo que contém um passeio Euleriano. Após analisar cuidadosamente diferentes tipos de grafo quanto à possível existência de passeios Eulerianos, foi enunciado o seguinte Teorema:

Teorema 2.1. *a)Um grafo conexo G possui um passeio Euleriano fechado se, e somente se, não possui nenhum vértice com grau ímpar.*

b) Um grafo conexo possui exatamente dois vértices com grau ímpar se, e somente se, ele tem um passeio Euleriano que começa em um desses vértices e termina no outro.

c) Se um grafo conexo possui mais de dois vértices com grau ímpar, então ele não tem passeio Euleriano.

Demonstração. (a) $\Rightarrow$ A demonstração da ida é bem simples. Supondo, por hipótese, que o grafo G possui um passeio euleriano fechado P, então cada vez que P passa por um vértice v , ele utiliza duas arestas incidentes: a que chega a P e a que sai de P, no sentido do passeio. Isso se deve ao fato do passeio ser fechado. Pela definição de passeio Euleriano cada aresta ocorre uma única vez, então o grau de cada vértice é par.

$\Leftarrow$ Vamos provar a volta por indução. Temos, por hipótese, que G só possui vértices de grau par. O caso em que G não possui arestas é trivial. Então, por hipótese de indução, vamos supor que o resultado é válido para G com menos de n arestas. Pela conexidade de G, podemos garantir que todos os seus vértices possuem grau maior ou igual a dois; e pela Proposição 2.1, garantimos que G contém um ciclo C.

Se C possui todas as arestas de G, a prova está concluída. Vamos supor, então, que $C \neq G$. Removendo de G todas as arestas de C, obtemos um grafo remanescente H, eventualmente desconexo, com menos arestas que G. Todos os vértices de H continuam possuindo grau par, pois o que foi removido de G foi um ciclo. Pela hipótese de indução, cada componente de H possui um passeio Euleriano fechado, e como G é conexo, cada componente de H possui pelo menos um vértice em comum com C. Agora, podemos traçar o passeio Euleriano fechado seguindo as arestas de C, até um vértice não isolado de H ser alcançado, e traçar um passeio Euleriano fechado na componente de H que contém tal vértice. Em seguida, continuamos a traçar o passeio pelas arestas de C até encontrar outro vértice não isolado que pertença a outra componente conexa de H, traçando outro passeio Euleriano fechado por ela, e assim por diante. O processo se repete sucessivamente até que o trajeto tenha passado por todas as componentes de H e retorne ao vértice inicial, finalizando o passeio Euleriano fechado.

(b) $\Rightarrow$ Seja $2n-1$ o grau de um vértice v_1 pertencente a um grafo G, com $n \in N$, e sejam a_i, $1 \leq i \leq 2n-1$, todas as arestas que passam por v_1. Vamos supor, primeiramente, um passeio P que começa em v_1. Neste caso, a aresta a_1 sai de v_1, enquanto a aresta a_2 chega a v_1, e assim por diante. Como o grau de v_1 é ímpar, então a aresta a_{2n-1} sairá de v_1, e o passeio acabará em outro vértice v_k. Seja b_1 uma aresta que chega a v_k, b_2 uma aresta que sai deste vértice, e assim por diante. Como o passeio termina em v_k, então a última aresta b_i que chega a ele tem índice ímpar, logo v_k tem grau ímpar. Ou seja, P começa e termina em vértices de grau ímpar.

Suponhamos, então, que P não começa em v_1. Então, em algum momento, a aresta a_1 vai chegar a v_1, enquanto o passeio vai seguir pela aresta a_2 saindo do vértice, e assim por

diante. Neste caso, sempre que o índice de a_i for ímpar, significa que a_i está chegando a v_1, no sentindo do passeio, da mesma forma que estará saindo de v_1 sempre que seu índice for par. Como o grau de v_1 é ímpar, significa que a última aresta a passar por v_1, aquela de índice $2n-1$, estará chegando a v_1, encerrando assim o passeio neste vértice. Como cada aresta está sendo usada uma única vez no passeio, então P é Euleriano. Portanto, se um grafo G possui algum vértice de grau ímpar, um passeio Euleriano em G ou começa ou termina neste vértice. Logo, se G possui exatamente dois vértices de grau ímpar, ele possui um passeio Euleriano que começa em um destes vértices e termina no outro.

$\Leftarrow$ Por hipótese, G possui um passeio Euleriano que começa no vértice v_1 e termina no vértice v_2, com $v_1, v_2 \in G$. Se adicionarmos em G uma aresta k, conectando v_1 a v_2, então o grafo resultante G' possuirá um passeio Euleriano fechado, e de acordo com a parte a deste Teorema, todos os vértices de G' têm grau par. Ao remover k de G' obtemos novamente o grafo G, e os graus dos vértices v_1 e v_2 diminuem em 1 unidade, ou seja, passam a ter grau ímpar, enquanto os graus dos demais vértices permanecem inalterados. Assim, v_1 e v_2 são os únicos vértices com grau ímpar em G.

(c) Como demonstrado na parte b deste Teorema, se um grafo conexo G possui um vértice de grau ímpar, então um passeio Euleriano em G começa ou termina neste vértice. Logo, se G possuir mais de dois vértices de grau ímpar, é impossível que contenha um passeio Euleriano P, pois P começaria em um deles e terminaria no outro. ■

De volta ao problema das pontes de Königsberg, seria impossível um passeio de modo que se atravessasse cada ponte uma única vez (passeio Euleriano), já que os distritos representavam vértices de graus 3, 5, 3, e 3, ou seja, mais de dois vértices de grau ímpar.

Exemplo 2.4. *Seja $G = (V; E)$ um grafo simples conexo não-euleriano. Queremos construir um grafo H que seja euleriano e que contenha G como subgrafo. Considere os seguintes possíveis processos de construção:*

(a) Acrescenta-se um novo vértice, ligando-o a cada vértice de G por uma aresta.

(b) Acrescenta-se um novo vértice, ligando-o a cada vértice de grau ímpar de G por uma aresta.

(c) Cria-se uma nova cópia G' do grafo G e acrescenta-se uma aresta ligando cada par de vértices correspondentes.

(d) Escolhe-se um vértice arbitrário de G e acrescentam-se arestas ligando este vértice a todo vértice de grau ímpar de G.

(e) Duplicam-se todas as arestas de G.

(f) Acrescentam-se arestas a G até se formar o grafo completo com $|V|$ vértices.

Quais dos processos acima sempre constroem corretamente o grafo H?

Resolução. Primeiramente, se H é Euleriano então contém um passeio Euleriano, e de acordo com o Teorema 2.1 isso implica que todos os vértices de H têm grau par (passeio Euleriano fechado) ou H possui no máximo dois vértices de grau ímpar (passeio Euleriano aberto). Da mesma forma, se G é não-Euleriano, então possui mais de dois vértices de grau ímpar, e de acordo com o Corolário 1.1, G possui uma quantidade par de vértices com grau ímpar. Vamos analisar cada um dos processos de construção de H propostos.

(a) Ao acrescentar um vértice v e conectá-lo aos vértices de G por arestas, os vértices que tinham grau par passam a ter grau ímpar, e os que tinham grau ímpar passam a ter grau par. Assim, se G possuía mais de dois vértices de grau par, H terá mais de dois vértices de grau ímpar, e neste caso não seria Euleriano. Então este processo nem sempre constrói o grafo H corretamente.

(b)Ao acrescentar um vértice v e conectá-lo aos vértices de G de grau ímpar por arestas, os vértices de grau par continuarão com o mesmo grau, e os vértices de grau ímpar passarão a ter grau par. Além disso, o grau de v será igual à quantidade de vértices de grau ímpar em G, que sabemos que é par. Assim, neste processo de construção todos os vértices de H terão grau par, o que garante que H é Euleriano.

(c) Criando uma cópia G' de G, ao ligar cada um de seus vértices ao vértice correspondente em G, todos os vértices pertencentes a G e a G' terão seu grau aumentado em 1, ou seja, o grau que era par agora será ímpar e vice-versa. Mas se G possui dois ou mais vértices de grau par, H terá quatro ou mais vértices de grau ímpar, e nesse caso não seria Euleriano, ou seja, este processo nem sempre constrói o grafo H corretamente.

(d) Ao escolher um vértice v arbitrário de G, existem duas possibilidades: v ter grau par e v ter grau ímpar. Vamos analisar primeiramente o caso em que o grau de v é par. Como ele será conectado a todo vértice de grau ímpar G, que possui uma quantidade par de vértices de grau ímpar, então o grau de v permanecerá par, pois a soma de dois números pares resulta em um número par. Já os demais vértices de G de grau par continuarão com o mesmo grau, já que nenhuma aresta será conectada a eles. E os vértices de G que possuíam grau ímpar terão em H mais uma aresta conectada a cada um deles, ou seja, seus graus passarão a ser par. Neste caso, todos os vértices de H teriam grau par.

Vamos considerar agora o caso em que o vértice v escolhido arbitrariamente tem grau ímpar. Como G tem um número par de vértices de grau ímpar, excluindo v resta uma quantidade ímpar destes vértices. Logo, como v será conectado aos demais vértices de grau ímpar em G, serão adicionadas a ele um número ímpar de arestas. Assim, seu grau passará a ser par, pois a soma de dois números ímpares resulta em um número par. Os demais vértices de grau ímpar em G terão mais uma aresta adicionada a eles em H, ou seja, passarão a ter grau par. Como os vértices de G de grau par não terão arestas adicionadas a eles, permanecerão com grau par em H. Assim, em ambos os casos H possui todos os vértices de grau par, ou seja, é Euleriano.

(e) Ao duplicar todas as arestas de G, cada um dos vértices de H terá o dobro de arestas incidentes que possuía em G, ou seja, o grau de cada um destes vértices será dobrado. Com isto, garantimos que todos os vértices de H terão grau par (pois todos os graus são múltiplos de 2), ou seja, H é Euleriano. Logo, este processo sempre constrói corretamente o grafo H.

(f) Ao transformar G de n vértices em um grafo completo H, o grau de cada um desses vértices será $n-1$. Então se n for par, ou seja, se G tiver um número par de vértices, todos os vértices de H terão grau ímpar. Neste caso, H não seria Euleriano para $n > 2$, ou seja, este processo nem sempre constrói o grafo H corretamente.

Portanto, dos processos acima, os que sempre constroem corretamente o grafo H são os descritos em (b), (d) e (e). ■

CAPÍTULO 3

ÁRVORES

3.1 DEFINIÇÃO

Um grafo G é denominado *árvore* se ele for conexo e não contiver qualquer ciclo como subgrafo. Por ser conexo, entendemos que quaisquer dois vértices de G podem ser conectados por um caminho, ou seja, G precisa de uma quantidade mínima de arestas que satisfaçam a essa condição. Ademais, por não conter ciclos, G não deve ter arestas excessivas, ou seja, deverá ter uma quantidade máxima de arestas para respeitar a essa condição. A figura 3.1 mostra algumas variedades de árvores.

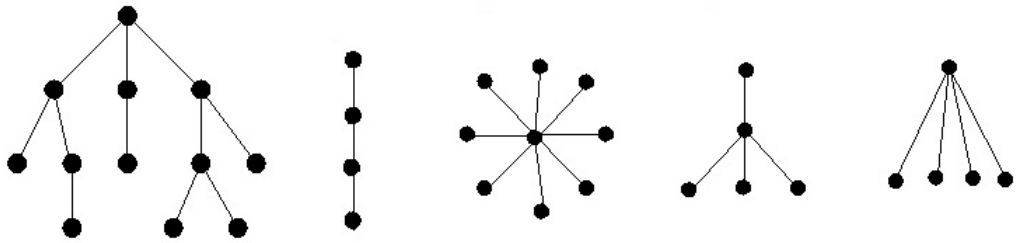


Figura 3.1 Exemplos de árvores

Teorema 3.1. *(a) Um grafo G é uma árvore se, e somente se, além de ser conexo, a remoção de qualquer uma de suas arestas o tornaria desconexo.*

(b) Um grafo G é uma árvore se, e somente se, não contiver nenhum ciclo, mas a adição de qualquer nova aresta implicaria na criação de um.

(c) Um grafo G é uma árvore se, e somente se, cada dois de seus vértices são conectados por um único caminho.

(d) Um grafo G é uma árvore, se e somente se, pode ser obtido através do Procedimento de Crescimento-de-Árvore, descrito a seguir:

1) Começa-se com um simples vértice;

2) Um novo vértice é criado e conectado através de uma nova aresta a qualquer vértice já existente, e esse processo é repetido qualquer número de vezes.

Demonstração. (a) $\Rightarrow$ Se o grafo G é uma árvore, então ele é conexo e a remoção de qualquer uma de suas arestas resulta em um grafo desconexo. Pela definição de árvore, segue diretamente que G é conexo. Então, basta provar que ao remover qualquer uma de suas arestas ele deixa de ser. Vamos supor, por absurdo, que removendo a aresta uv de G ele resulte em um grafo G' ainda conexo. Então, podemos garantir que existe em G' um caminho P que conecta u a v (pela definição de grafo conexo). Portanto, se

adicionarmos de volta a aresta uv a G', resultando novamente em G, o caminho P e a aresta uv formarão um ciclo em G, o que contradiz diretamente a definição de árvore (que não pode conter nenhum ciclo). Logo, se o grafo G é uma árvore, ou seja, é conexo, então a remoção de qualquer uma de suas arestas implica em um grafo desconexo.

$\Leftarrow$ Se G é um grafo conexo e a remoção de qualquer uma de suas arestas resulta em um grafo G' desconexo, então G é uma árvore. Para comprovar isto, basta mostrar que G não contém nenhum ciclo, pois por hipótese G é conexo. Vamos supor, por absurdo, que G contenha um ciclo C. Ao remover uma aresta uv de C, ainda continuará existindo um caminho que conecta u a v (Proposição 2.3), e, portanto, o grafo resultante G' continuará sendo conexo, o que contradiz a hipótese de que ao remover uma aresta de G o grafo resultante é desconexo. Portanto, se G é um grafo conexo e ao remover qualquer uma de suas arestas o grafo G' resultante for desconexo, então G é uma árvore.

(b)$\Rightarrow$ Se o grafo G é um árvore, então ele não contém nenhum ciclo, mas a adição de qualquer aresta implicaria na criação de um ciclo no grafo resultante. Pela definição de árvore, segue diretamente que G não possui nenhum ciclo. Portanto, basta provarmos que se adicionarmos uma aresta uv a G, surge um ciclo C no grafo resultante G'. Sejam u e v vértices de G. Como G é uma árvore, então, por conexidade, existe um caminho C $ux_1x_2...v$ conectando u a v, onde x_i também são vértices de G. Ao adicionar a G uma aresta uv, obtemos o ciclo $ux_1x_2...vu$ em G'.

$\Leftarrow$ Queremos provar que se um grafo G não contém nenhum ciclo e ao adicionarmos uma aresta a ele o grafo resultante contém um ciclo, então G é uma árvore. Note que basta provar que G é conexo. Considere dois vértices u e v arbitrários pertencentes a G. Se u e v estão ligados por uma aresta, então estão conectados por um caminho. Senão, acrescente a aresta uv. Deste modo, produziremos um ciclo C. Removendo a aresta uv de C, temos que uv estão ligados por um caminho em G (Proposição 2.3), logo G é conexo.

(c) $\Rightarrow$ Se G é uma árvore (hipótese), então G é conexo, logo quaisquer dois vértices u e v pertencentes a G podem ser conectados por um caminho C. Temos que provar que esse caminho é único. Vamos supor, por absurdo, que C não seja único, ou seja, existe outro caminho C' em G, diferente de C, que conecte u e v. Seguindo o caminho na direção de u a v, seja x o último vértice que C e C' possuem em comum antes de se separar, e seja y o vértice onde C e C' voltam a se encontrar.

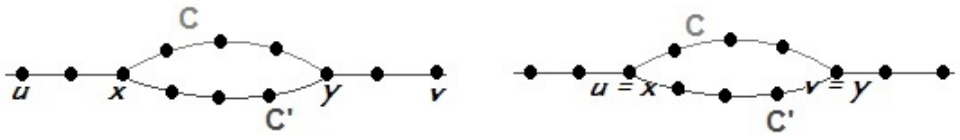


Figura 3.2 Caminhos C e C'

Perceba que x pode ou não coincidir com u, e y pode ou não coindidir com v (Figura 3.2). Como x e y são as extremidades onde C e C' se separam, então todos os vértices entre x e y ou pertecem a C ou pertencem a C'. Logo, existe um ciclo contendo x e y, o que contraria a hipótese de que G é uma árvore.

$\Leftarrow$ Por hipótese, existe um único caminho que conecta dois vértices u e v quaisquer

de G. Logo, G é conexo (pois existe caminho de u a v) e G não contém nenhum ciclo (pois este caminho é único). Logo, G é uma árvore.

(d) $\Rightarrow$ Vamos provar por indução sobre o número de vértices. Seja G uma árvore de um único vértice, então G foi construída pelo Procedimento de Crescimento-de-árvores, pois este é o primeiro passo do procedimento. Vamos supor, então, que G possui pelo menos dois vértices. De acordo com a Proposição 2.1, se os graus dos vértices de G fossem todos maiores ou iguais a 2, então G possuiria um ciclo. Mas, como G é um árvore, então não contém nenhum ciclo, o que implica que pelo menos um de seus vértices tem grau 1, o qual chamaremos de v. Ao remover v de G, juntamente com sua aresta incidente, obtemos um grafo G'. Certamente, G' é conexo, pois quaisquer vértices x e y pertencentes a G' podem ser conectados por um caminho de G, o qual não passa por v que tem grau 1. Como G' foi obtido por uma remoção de vértice e aresta de G, que não continha nenhum ciclo por ser árvore, então G' também não contém nenhum ciclo, logo G' é árvore.

Assim, pela hipótese de indução, toda árvore G' com menos vértices que a árvore G surge por construção através do Procedimento de Crescimento-de-árvore. Então, podemos afirmar que G surge de G' ao executar mais um passo do procedimento, ou seja, toda árvore G pode ser obtida através dele.

$\Leftarrow$ Seja um grafo G obtido pelo Procedimento de Crescimento-de-Árvore. Como ele começou a ser criado com um único vértice, então neste momento, era uma árvore. Se conseguirmos mostrar que durante as etapas do procedimento este grafo nunca deixou de ser uma árvore, é imediato que o grafo final G é uma árvore. Mais precisamente, demonstraremos que se G' é uma árvore e que se G foi obtido ao adicionar um novo vértice e conectá-lo a um vértice de G', então G é uma árvore.

Suponhamos que G foi obtido ao conectar, através de uma aresta a_1, um novo vértice y a um certo vértice x de G'. Como G' é árvore, então é conexo, assim qualquer vértice w de G' pode ser conectado ao vértice x por um caminho em G'. Podemos garantir que G também é conexo, pois como um vértice w qualquer de G' pode ser conectado a x por um caminho, e como x é ligado a y por a_1, então w pode ser conectado a y, e além disso quaisquer dois vértices de G que estejam em G' já poderiam ser conectados.

Então, para provar que G é uma árvore, basta mostrar que não possui nenhum ciclo. Ao conectar y a G' por a_1 e formar G, é evidente que o vértice y possui grau 1, pois sua única aresta incidente é a_1. Como G' é árvore, não existe nenhum ciclo em G', e consequentemente, G também não possui nenhum ciclo. ■

O Procedimento de Crescimento-de-árvore estabelece algumas propriedades de árvores. Uma delas é relacionada ao número de arestas que uma árvore possui, o qual depende exclusivamente do número de vértices da mesma.

Teorema 3.2. *Toda árvore que possui n vértices, possui $n-1$ arestas.*

Demonstração. Como toda árvore pode ser construída através do Procedimento de Crescimento-de-árvore, após o primeiro passo, ela possui 1 vértice e nenhuma aresta, ou seja, a quantidade de vértices excede em 1 a quantidade de arestas. Ao executar o segundo passo, sempre é adicionado à arvore *um* vértice e *uma* aresta. Como toda a árvore é construída através da repetição sucessiva do segundo passo do procedimento, ou seja,

adicionando sempre um vértice e uma aresta à árvore, a diferença entre a quantidade de vértices e arestas da árvore não se alterará em nenhum momento. Logo, a árvore sempre terá um vértice a mais que a quantidade de arestas. Assim, se G é uma árvore de n vértices, então G possui $n - 1$ arestas. ■

3.2 ÁRVORES ENRAIZADAS E ÁRVORES GERADORAS

Seja G um grafo conexo com n arestas a_i, onde $1 \leq i \leq n$. Quando uma de suas arestas a_i é removida, o gráfico remanescente pode ou não permanecer conexo. Caso ele fique desconexo, dizemos que a_i é uma *aresta-de-corte*. Consequentemente, podemos concluir que toda aresta de uma árvore é aresta-de-corte (Teorema 3.1).

Vamos agora considerar o caso em que G não é uma árvore, ou seja, entre suas arestas há pelo menos uma que não é aresta-de-corte. Se removermos cada uma dessas arestas a_i até que todas as arestas restantes sejam arestas-de-corte, o grafo resultante G', que é subgrafo de G, é uma árvore, e é denominado *árvore geradora* de G. Como o processo de remoção de arestas pode ser feito de várias maneiras, um grafo conexo pode ter várias árvores geradoras.

É possível, em uma árvore G qualquer, que um dos seus vértices seja escolhido e designado como *raiz*, a qual determina uma hierarquia aos demais vértices, dependendo da "distância" que eles se encontrem dela. A este tipo de árvore denominamos árvore *enraizada*, onde cada aresta é direcionada no sentido de se afastar da raiz.

Definição 3.1. *Dada uma árvore G qualquer, uma raiz de G é qualquer vértice de G escolhido e denominado, a partir da escolha, de raiz. A ávore G é então chamada de árvore enraizada.*

Seja G uma árvore enraizada com raiz r, e sejam u e v vértices quaisquer de G. Da parte (c) do Teorema 3.1, sabemos que existe um único caminho que conecta r a v. Se neste caminho o vértice u precede v imediatamente, então u é chamado *pai* de v. Os outros vizinhos de v são denominados seus *filhos*. Vértices com o mesmo pai são chamados de *irmãos*. A raiz r não tem pai.

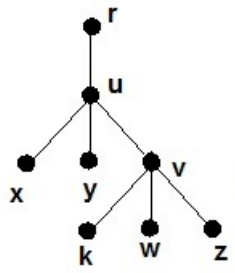


Figura 3.3 Árvore enraizada

Na figura 3.3 acima, temos um exemplo de uma árvore enraizada de raiz r, onde o vértice u é pai dos vértices x, y e v, que são irmãos, assim como os vértices k, w e

z também o são, pois são filhos do mesmo pai v. Assim, um vértice pode ter qualquer número de filhos, incluindo *zero*, como é o caso dos vértices x, y, k, w e z ainda na mesma figura. Neste caso, são chamados de $folhas$. Portanto, em uma árvore com mais de um vértice, suas folhas são aqueles de 1, diferentes da raiz. No exemplo da figura 3.3, a raiz r tem grau 1, mas não é folha.Quando a árvore consiste em um único vértice, então este é uma folha.

Proposição 3.1. *Em uma árvore G todo vértice diferente da raiz possui um único pai.*

Demonstração. Seja r a raiz de uma árvore G e seja v um de seus vértices (diferente de r). Da parte (c) do Teorema 3.1, sabemos que G contém um único caminho conectando r a v, e por definição o vértice que vem imediatamente antes de v neste caminho é denominado seu pai. Como o caminho é único, tal vértice também é único, tenho v, portanto, um único pai. ■

Proposição 3.2. *Em uma árvore enraizada, todo vértice é pai de seus filhos.*

Demonstração. Seja G uma árvore enraizada de raiz r e sejam v e u vértices de G, onde u é filho de v. Queremos provar que v é pai de u.

Seja P o caminho de v a r, neste sentido. Podemos concluir que u não é o primeiro vértice depois de v, pois se fosse, então u seria pai de v, e não seu filho. Além disso, u também não pode ser outro vértice de P, pois além da aresta uv, existiria um outro caminho em P de u a v, e em uma árvore dois vértices são conectados por um único caminho (parte c do Teorema 3.1). Ou seja, u não faz parte de P. Assim, ao adicionarmos o vértice u a P e a aresta uv, obtemos um caminho P' de u a r. Mas v será o primeiro vértice depois de u em P', logo v é pai de u. ■

Teorema 3.3. *Toda árvore G com no mínimo dois vértices, possui pelo menos dois vérties de grau 1.*

Demonstração. Pela proposição 2.1, podemos concluir que G possui pelo menos um vértice de grau 1, pois caso contrário conteria um ciclo ou um vértice de grau 0, e ambas as condições contrariam a definição de árvore. Considere que v é este vértice de guau 1. Vamos supor um passeio P em G, sem repetição de arestas, começando por v. Como G é uma árvore e não possui ciclos, então P não pode passar por um mesmo vértice mais de uma vez, logo P não pode terminar em v. Seja u o vértice onde o passeio termina. Como P passa por u uma única vez, então u possui uma única aresta incidente, logo u tem grau 1. Portanto, se G é uma árvore com pelo menos dois vértices, então possui no mínimo dois vértices de grau 1. ■

Exercício 3.1. *Prove que se uma árvore G possui um vértice de grau d, então possui pelo menos d vértices de grau 1.*

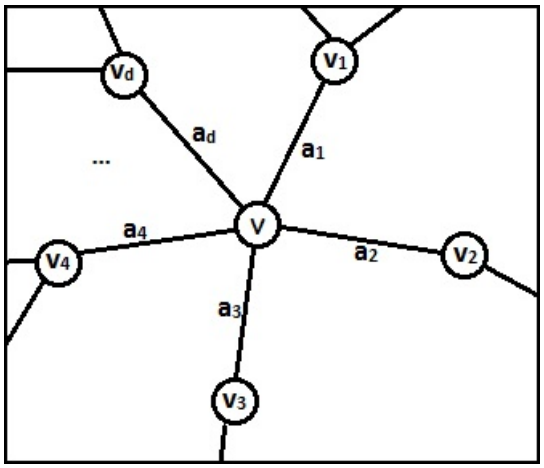


Figura 3.4 Recorte de G

Resolução. Seja v o vértice de G que tem grau d, e sejam v_1, v_2, ... e v_d os vértices conectados a v pelas arestas a_1, a_2, ... e a_d, respectivamente, como ilustrado na figura 3.4 abaixo.

Como G é arvore, ou seja, quaisquer dois pares de vértices v_i e v_j (com $i \neq j$ e $1 \leq i, j \leq d$) só podem ser conectados por um único caminho (parte c do Teorema 3.1), então se removermos de G o vértice v e as arestas a_1, a_2, ... e a_d, o grafo remanescente G' será formado por d componentes conexas, onde cada uma delas conterá um dos vértices v_i $(1 \leq i \leq d)$. Como cada componente conexa não contém nenhum ciclo (pois G era árvore), então podemos concluir que cada uma dessas componentes será também uma árvore.

De acordo com o Teorma 3.3, se cada uma destas componentes possuírem no mínimo dois vértices, então cada uma terá no mínimo dois vértices de grau 1, onde os vértices v_i podem ser folhas de G ou não. Caso contrário, a componente terá um único vértice, ou seja, será a própria folha v_i. Em todo caso, pelo menos uma das folhas é folha de G, então G possui no mínimo d vértices de grau 1, como queríamos demonstrar.

Uma outra solução deste problema é considerar o Procedimento de Crescimento-de-árvores. Supondo que G foi construída inicialmente pelo vértice v e todas as suas d arestas remanescentes, ligadas aos vértices $v_1, v_2, ...v_d$. Então nesta etapa de construção G possuía d vértices de grau 1. Ao acrescentar arestas e vértices a um vértice v_i, este deixa de ter grau 1, mas pelo menos mais uma folha é criada. Assim, a quantidade de vértices de grau 1 nunca diminuirá, sendo no mínimo d, ou seja, se uma árvore G possui um vértice de grau d, então possui pelo menos d vértices de grau 1. ■

Seja G uma árvore com n vértices. A depender da utilidade ou finalidade atribuída a G, ou que informações sobre ela você deseja registrar, é possível que haja a necessidade de especificar ou nomear seus vértices e arestas. Assim, cada um dos seus n vértices podem ser numerados de "0" a "n-1", e cada aresta pode ser referida listando suas extremidades, ou seja, os vértices que ela conecta. Este tipo de árvore, cujos vértices são rotulados de 0 a $n-1$, é denomidado *árvore rotulada* (Figura 3.5).

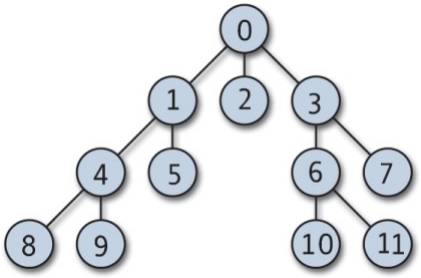


Figura 3.5 Árvore rotulada

Existem diversas maneiras de se armazenar uma árvore rotulada. Um modo bem útil e eficiente é através de uma codificação. Tomando G como uma árvore enraizada onde o vértice 0 é tido como sua raiz, podemos nos referir a cada aresta pelos dois vértices que ela conecta, citando primeiro o mais distante da raiz e depois a extremidade mais próxima da mesma. Organizando este código da aresta com um número abaixo do outro, temos que o número que representa o vértice da linha de baixo é o pai do vértice representado na linha acima.

A ordem em que as arestas aparecem pode ser determinada por vários critérios. Por exemplo, as arestas podem ser ordenadas pela ordem crescente de sua primeira extremidade. Tomando como exemplo a árvore da figura 3.5, onde o vértice 0 é sua raiz, temos a seguinte tabela:

1	2	3	4	5	6	7	8	9	10	11
0	0	0	1	1	3	3	4	4	6	6

Como pode ser observado, o vértice 0 não aparece na primeira linha, pois como é a raiz, não é filho de nenhum outro vértice. Ademais, como uma árvore de n vértices possui $n-1$ arestas (Teorema 3.2), este será o número de colunas da tabela. No caso da árvore tomada como exemplo, se ela possui 12 vértices, a tabela apresenta 11 colunas. Como a primeira linha será sempre numerada pelos vértices de 1 a $n-1$, em ordem crescente, ela pode ser dispensada. Assim, o código da árvore será composto pela segunda linha da tabela, e é chamado de *código de pai*.

Sendo considerado como um aprimoramento do código de pai, existe um determinado código onde o critério para estabelecer a ordem das arestas é mais refinado. Este é denominado *código de Prüfer*, e em vez de ser formado por uma sequência de $n-1$ rótulos, será formado por $n-2$. Já a ordem em que as arestas são listadas, considerando uma árvore G de $n-1$ vértices, é determinada da seguinte maneira:

1) Procura-se a folha (vértice de grau 1 que não seja pai de ninguém) de menor rótulo, e a aresta com essa extremidade é registrada.

2) Em seguida, este vértice e aresta são removidos, atualizando a árvore, e este procedimento é repetido até que a última aresta seja listada.

Vamos tomar como exemplo novamente a árvore da figura 3.5. Como pode ser observado, a folha de menor rótulo é o vértice 2. Ao removê-lo, junto com a aresta $2-0$, a

folha de menor rótulo será o vértice 5. Ao ser removido com a aresta $5-1$, a folha de menor rótulo será agora o vértice 7, e após sua remoção e a da aresta $7-3$ será o vértice 8. Removendo o vértice 8 e a aresta $8-4$, a folha de menor rótulo será o vértice 9, que será removido junto com a aresta $9-4$. Agora o vértice 4 é uma folha, e a de menor vértice, portanto a próxima a ser removida, junto com a aresta $4-1$. O vértice 1 se torna a folha de menor rótulo, sendo removido com a aresta $1-0$. Seguindo o procedimento, os próximos vértices e arestas a serem removidos serão o 10 com a aresta $10-6$, o 11 com a aresta $11-6$, o 6 com a aresta $6-3$, e o 3 com a aresta $3-0$, ficando o código do seguinte modo:

2	5	7	8	9	4	1	10	11	6	3
0	1	3	4	4	1	0	6	6	3	0

Este código é denominado *código estendido de Prüfer*. Como a raiz é o único vértice que não é removido, a última aresta a ser listada será sempre incidente a ela. Assim, sabemos que a última entrada a ser listada na segunda linha é o vértice 0. Portanto, sendo o código de Prüfer dado pela segunda linha, não precisamos da última entrada, já que será sempre a mesma. Logo, este código será formado por uma sequência de $n-2$ rótulos. Mas como advinhar a árvore a partir do código de Prüfer?

O raciocínio para responder a esta pergunta é bem simples. Vamos explicá-lo ultilizando o mesmo exemplo, ou seja, suponha que nos foi fornecido o seguinte código:

0	1	3	4	4	1	0	6	6	3

Primeiramente, se o código possui 10 entradas, podemos deduzir que a árvore procurada possui 12 vértices rotulados de 0 a 11. Para deduzir a primeira entrada da linha de cima, devemos considerar que esta foi a folha de menor rótulo a ser removida. Como os vértices que aparecem no código dado são pais de algum vértice, então podemos deduzir que não são folhas. Assim, a primeira entrada da linha de cima não pode ser nenhum dos vértices 0, 1, 3, 4 ou 6. Logo, podemos concluir que a primeira folha a ser removida foi o vértice 2, pois se ele não fosse folha seria pai de alguém e apareceria no código.

Ao concluir que 2 é a primeira entrada da linha de cima no código estendido, deduzimos que a segunda entrada será a folha de menor rótulo após o 2 ser removido. Os mesmos vértices ainda estão no código, ou seja, a segunda entrada ainda não pode ser nenhum dos vértices 0, 1, 3, 4 ou 6, sendo portanto, o vértice 5 o de menor rótulo. Pelo mesmo motivo anterior, concluímos que o 5 é a segunda entrada. Seguindo o mesmo raciocínio, vamos descobrir que a terceira, a quarta e a quinta entrada da primeira linha do código estendido são, respectivamente, os vértices 7, 8 e 9.

Agora, o vértice 4 não se encontra mais entre os vértices do código cujas entradas correspondentes da linha de cima ainda não foram descobertas. Assim, podemos concluir que após a remoção do 9, o vértice 4 se tornou uma folha, sendo portanto a próxima a ser removida, pois é a de menor rótulo. Sabendo que o 4 é a sexta entrada, notamos que o vértice 1 não é pai de nenhuma aresta ainda a ser descoberta, logo será a próxima folha a ser removida. De modo semelhante, obtemos as próximas entradas até chegar ao código estendido.

A partir deste raciocínio, podemos enunciar o seguinte Teorema:

Teorema 3.4. *A segunda linha de um código estendido de Prüfer determina a primeira.*

Demonstração. Pela proposição 3.1, sabemos que em uma árvore enraizada uma folha não é pai de nenhum vértice. Como a primeira linha do código contém, da esquerda para a direita, a folha de menor rótulo, então no momento em que o vértice aparece na primeira linha significa que não é mais pai de nenhum vértice que ainda não tenha sido citado.

Se a segunda linha contém os pais das respectivas folhas da linha acima, então os vértices que aparecem como entrada na primeira linha não podem estar a sua direita na linha de baixo. Como a folha que aparece na linha de cima é aquela de menor rótulo no momento, então na segunda linha do código basta observar qual é o vértice de menor rótulo que não foi computado (significando que não é pai de ninguém, portanto uma folha) e este será a primeira entrada da linha de cima. Além disso, se um vértice já apareceu como entrada na linha de cima, significa que é folha, e que foi 'retirado' da árvore para continuar o procedimento, logo não aparecerá como entrada novamente.

Resumindo o procedimento, concluimos que cada entrada na primeira linha do código estendido de Prüfer é o menor rótulo que não ocorre na primeira linha antes dele, nem na segunda linha abaixo ou depois dele. Assim, a segunda linha do código determina a primeira. ■

Teorema 3.5. *Em um conjunto de n vértices, podem ser construídas n^{n-2} árvores rotuladas distintas.*

Demonstração. Sabemos que o código de Prüfer é uma sequência de $n-2$ números, compreendidos entre 0 e $n-1$. Assim, se provarmos que toda sequência deste tipo é código de Prüfer de alguma árvore rotulada de n vértices, basta verificar quantas sequências podem ser formadas neste padrão e descobrimos quantas árvores rotuladas existem sobre n vértices.

Dada uma sequência de $n-2$ entradas com números naturais compreendidos de 0 a $n-1$, vamos acrescentar o 0 ao final da sequência e formar uma primeira linha escrevendo acima de cada entrada o menor rótulo que não ocorre na mesma linha antes dele e nem abaixo ou depois dele na linha de baixo. Assim, por definição, esta tabela é o código estendido de Prüfer de alguma árvore. Ou seja, a geração aleatória de qualquer sequência de $n-2$ rótulos dentre os n rótulos relativos a n vértices, implica em gerar o correspondente Código de Prüfer de uma árvore qualquer de n vértices.

Agora, temos apenas que contabilizar quantos códigos de Prüfer diferentes podem ser formados sobre n vértices. Para cada uma das $n-2$ entradas do código de Prüfer, há n possibilidades de escolha, pois pode ser qualquer um dos n vértices. No caso de todas as posições serem ocupadas pelo mesmo rótulo, por exemplo, a árvore formada seria uma estrela. Assim, o total de possibilidades para o código de Prüfer sobre n vértices é de n^{n-2}. Como cada um destes códigos determina uma árvore diferente, então sobre um conjunto de n vértices existem n^{n-2} árvores rotuladas distintas. ■

CAPÍTULO 4

ALGORITMO GULOSO

4.1 ÁRVORE GULOSA E MÉTODO OTIMISTA

Seja G um grafo de n vértices, onde cada um desses vértices representa um lote de terra que se encontra em determinado lago. O dono do lago, com o intuito de caminhar de um lote para outro, deseja construir pontes que os conectem dois a dois, representadas em G por arestas. Obviamente, não há a necessidade de construir uma ponte entre cada dois lotes distintos, pois se há conexão do lote 1 para o lote 2 e do lote 2 para o lote 3, por exemplo, há como ir do lote 1 ao 3 passando pelo lote 2. Entretanto, cada uma destas pontes possui um determinado custo, dependendo da distância entre os lotes e outros fatores. Supondo que o custo da construção de cada ponte possível tenha sido avaliado, como o dono do lago deve escolher quais pontes construir de modo que obtenha um custo mínimo?

Primeiramente, se ele deseja ter acesso a todos os lotes, pelo menos uma ponte deve chegar a cada um deles, ou seja, G deve ser conexo. Como o custo desta construção tem que ser mínimo, a menor quantidade possível de pontes deve ser construída, ou seja, não deve existir nenhuma aresta em G que possa ser removida sem que G deixe de ser conexo. Logo, podemos concluir que G é uma árvore (Teorema 3.1), ou seja, possui $n-1$ arestas ou que o lago possui $n-1$ pontes (Teorema 3.2).

Na figura 4.1 abaixo, temos um exemplo onde o lago possui 10 lotes de terra, então consequentemente foram construídas 9 pontes, representadas por arestas.

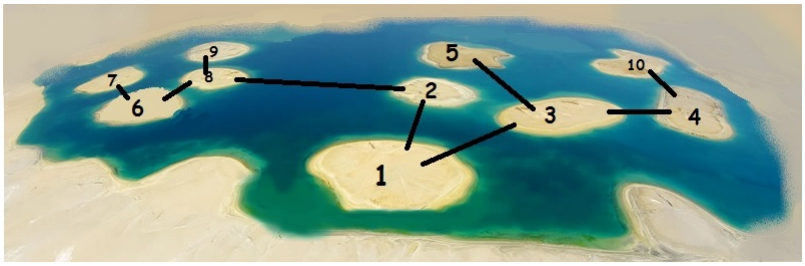


Figura 4.1 Exemplo de lago com 10 lotes de terra

Porém, como a construção de cada ponte possui um custo diferente, não se trata apenas de quantidade, e sim do valor atribuído a cada uma. Associando a cada aresta o valor do custo da ponte, o qual podemos chamar de peso, temos que o custo total dessa obra será a soma dos pesos de cada aresta. Portanto, para que o custo total seja mínimo, as pontes (ou arestas) devem ser escolhidas de modo que a soma de seus valores seja mínimo. Este método de seleção é chamado de *método otimista* ou *algoritmo guloso*.

Assim, a árvore G de menor custo, escolhida pelo método otimista para representar esses lotes e as respectivas pontes que o interligam, é definida como *árvore geradora de custo mínimo* ou *árvore gulosa*.

Existem várias maneiras de construir uma árvore gulosa, as quais são determinadas pelos critérios de construção estabelecidos, o que vai definir o tipo de algoritmo utilizado. Voltando ao problema das pontes, por exemplo, poderia ser definido como critério que ao construir a primeira ponte, todas as demais devem ser construídas a partir dela, ou seja, uma vez dentro do lago em um determinado lote, a única maneira de se deslocar para outros lotes é através da construção de pontes, sempre escolhendo a menos custosa. Esse tipo de algoritmo guloso é denominado *algoritmo de Prim*. Vamos traduzir este processo de construção da árvore G para o conceito de grafos.

Seja G a árvore de n vértices v_i $(1 \leq i \leq n)$ a ser contruída, onde a_j é o conjunto das suas possíveis arestas, onde cada uma possui um peso diferente, com $j \geq 1$. Vamos considerar G, inicialmente, como um grafo vazio. Seja a_1 a aresta de menor peso que liga v_1 a v_2, por exemplo. Então a_1 será a primeira aresta adicionada a G. Seja a_2 a próxima aresta de menor peso que conecta v_1 ou v_2 a um de seus vizinhos. Então a_2 será a próxima aresta adicionada a G. Seguindo esse raciocínio, a próxima aresta a ser adicionada será aquela de menor peso que incida em um dos vértices pertencentes à componente conexa em formação. Este processo deve ser repetido até que os n vértices de G estejam conectados, ou seja, até que G seja conexo. Obviamente, como G é uma árvore, o processo de crescimento de árvores deve ser respeitado nesta construção (seção 3.3).

Suponha agora um novo critério de construção para as pontes. Se no conjunto dos n lotes, após a construção da primeira ponte que conecta os lotes 1 e 2, haja a possibilidade de que a segunda ponte a ser construída conecte os lotes 4 e 7, por exemplo. Neste caso, durante o processo de construção das pontes, é permitido se deslocar pelo lago de uma outra maneira, como por exemplo de barco. Traduzindo para o conceito de grafos, isso significa que nos n vértices de G, após a adição da primeira aresta menos custosa a_1, a próxima aresta adicionada não precisa fazer parte da componente conexa que contém a_1, e assim por diante. Logo, uma vez que os critérios de crescimento de árvores sejam respeitados, qualquer aresta que o algoritmo guloso julgar como a mais conveniente pode ser adicionada na sequência, até que a árvore G seja construída. Este tipo de algoritmo guloso é denominado *algoritmo de Kruskal*.

Existem outros tipos de algoritmo guloso que obedecem a diferentes critérios, mas focaremos especificamente no *algoritmo de Kruskal*. Neste caso, considerando que o algoritmo construirá a árvore de custo mínimo G, que possui n vértices $(v_1v_2...v_n)$ e arestas com diferentes pesos, a primeira aresta a ser adicionada será aquela de menor peso. Em seguida, a próxima aresta a ser adicionada será novamente a de menor peso e que conecte quaisquer dois vértices ainda não conectados por nenhum caminho em G. Este processo será repetido até que as $n-1$ arestas sejam construídas. Lembre-se que como G é uma árvore de n vértices, então possui $n-1$ arestas (Teorema 3.2) e G será um grafo conexo que não possui nenhum ciclo (parte a do Teorema 3.1).

Definição 4.1. *O algoritmo de Kruskal executado em um grafo G de n vértices, onde*

$A = (a_1, a_2, ...)$ é o conjunto das possíveis arestas a conectarem os vértices de G, cada uma com um respectivo peso, funciona da seguinte maneira:

1) Seja a_i uma aresta de A de menor peso. Se a_i não criar ciclo em G, então é adicionada a G. Caso contrário, é descartada de A.

2) O processo é repetido até que A seja vazio.

A questão é, seguindo esse método de otimização, denominado *método otimista*, é possível garantir que a árvore gulosa G construída é realmente a de menor custo? Ou seja, qualquer árvore H diferente de G formada pelos vértices $v_1v_2...v_n$ possui no mínimo o mesmo custo de G? A resposta a essa pergunta estabelece a seguinte proposição:

Proposição 4.1. *Se G é uma árvore gulosa sobre n vértices construída pelo método otimista do algoritmo de Kruskal, então qualquer árvore G' diferente de G, construída sobre os mesmos n vértices, terá custo igual ou maior ao custo de G.*

Demonstração. Vamos considerar, no processo de construção de G, o passo em que foi adicionada a primeira aresta i, que não é aresta de G'. Se adicionarmos i a G', obteremos um ciclo C (parte b do Teorema 3.1). Como G é árvore, então não contém C inteiramente, ou seja, existe alguma aresta j pertencente ao ciclo C (logo pertencente a G') que não é aresta de G (Figura 4.2).

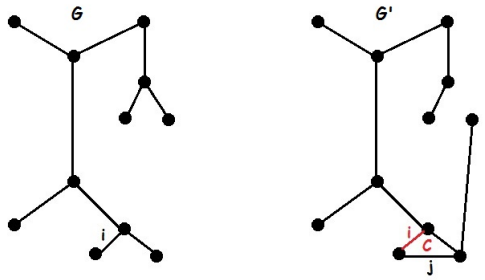


Figura 4.2 Construção de árvores 1

Se em G' adicionarmos i e removermos j, obteremos uma terceira árvore G'' (podemos garantir que G'' é árvore pelo argumento usado na proposição 2.3), como pode ser visto na figura 4.3 abaixo.

Queremos mostrar que o preço de G'' é menor ou igual ao preço de G'. Como a única diferença entre estas duas árvores são as arestas i (que está contida apenas em G'') e j (que está contida apenas em G'), é equivalente mostrarmos que i é no máximo tão cara quanto j. Vamos supor, por absurdo, que i é mais cara que j. Isso signigica que, por algum motivo, a aresta j estava impossibilitada de ser escolhida em G, já que o algoritmo guloso escolhe sempre a aresta mais barata. A única razão que poderia justificar essa escolha seria o caso de a escolha de j em G implicar na criação de algum ciclo C' formado pelas arestas de G já selecionadas. Porém, como estamos considerando, no processo de formação de G, que i é a primeira aresta que não está em G' e foi adicionada a G, então todas estas arestas previamente adicionadas a G (antes de i) fazem parte de G'. Como j também pertence a G', então podemos concluir que todas as arestas de C' pertencem a

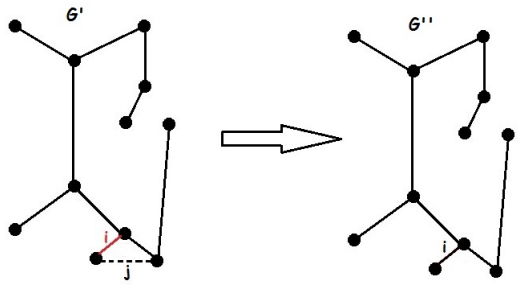


Figura 4.3 Construção de árvores 2

G', o que é absurdo, pois se G' é árvore não pode conter nenhum ciclo. Então, provamos por contradição que i não pode ser mais cara que j, ou seja, G'' não pode ser mais cara que G'.

Assim, podemos substituir G' por G'', já que temos certeza que ela não é mais cara. Ademais, pode ser facilmente observado que em relação a G', G'' coincide com G em mais arestas, pois G'' foi obtida de G' ao se adicionar uma aresta pertencente a G (i) e ao remover uma aresta que não pertencia a G (j). Logo, se G'' for diferente de G, podemos repetir o processo utilizando o mesmo argumento, e vamos chegar sempre a árvores que não são mais caras que G' e coincidem com G em mais arestas, o que implica que em algum momento a árvore obtida deste processo será a própria G. Assim, fica demonstrado que G não era mais cara que G', ou seja, G' tem custo igual ou maior a G. ■

Vamos considerar, agora, um outro método para construção destas pontes. Suponha que o dono do lago, em posse da informação de quais são todas as possíveis pontes a serem construídas e seus respectivos custos, decida que, em vez de ir erguendo as pontes mais baratas, ele vá eliminando o projeto das pontes mais caras e os marcando como impossíveis, até que todos os projetos que não forem essenciais para satisfazer à condição inicial (que cada lote possua pelo menos uma ponte que leve a ele e o conecte aos demais lotes) sejam eliminados. Assim, após eliminar tantas pontes caras quanto possível, ele realizaria a obra inteira.

Considerando estas possíveis pontes a serem erguidas como arestas em um grafo A, indicadas por seus respectivos pesos, e os n lotes representados por vértices $v_1v_2...v_n$, vamos descrever o algoritmo que selecionará as arestas que serão eliminadas de A até que o grafo se torne uma árvore H sobre os n vértices. Denominado por *algoritmo guloso pessimista*, este selecionará a aresta de maior peso, e caso sua remoção não torne o grafo remanescente desconexo, será eliminada. Mais uma vez, o algoritmo indicará a próxima aresta de maior peso, e se sua remoção não comprometer a conexidade do grafo, também será eliminada. O processo será repetido até que todas as arestas restantes sejam arestas-de-corte, ou seja, até que o grafo H remanescente seja uma árvore.

Entretanto, será que esse método de seleção também levaria ao grafo de menor custo possível, ou seja, levaria à árvore gulosa? Curiosamente, este custo também seria o mais barato, tanto quanto o custo da árvore gulosa obtida pelo método otimista. Isto será

demonstrado na proposição a seguir.

Proposição 4.2. *Se H é uma árvore sobre n vértices construída pelo algoritmo guloso pessimista, então seu custo será igual ao de qualquer árvore gulosa G construída pelo algoritmo de Kruskal sobre os mesmos n vértices.*

Demonstração. Seja A o grafo do qual foram eliminadas as arestas selecionadas pelo algoritmo pessimista até se chegar à árvore H. Considerando esse processo de eliminação, seja i a primeira aresta a ser removida de A que pertença a G. Se removermos i de G, o grafo resultante será desconexo (parte a do Teorema 3.1), possuindo duas componentes. Como H é árvore, e portanto conexa, podemos garantir que ela contém alguma aresta j, não pertencente a G, que conecta estes dois componentes.

Temos que provar que j é no máximo tão cara quanto i. Vamos supor, por absurdo, que i seja mais barata que j. Considerando o processo de construção de H, por algum motivo a aresta j não foi removida antes de i. A única razão que justificaria essa escolha é o fato de que se j fosse removida, o grafo resultante se tornaria desconexo. Se adicionarmos i novamente a H, um ciclo C será formado (parte b do Teorema 3.1). Como i faz parte da mesma componente de j, então C contém ambas as arestas i e j. Neste caso, qualquer aresta de C poderia ser removida sem comprometer a conexidade de H (proposição 2.3), o que contraria o fato da remoção de j implicar na desconexidade de H. Logo, podemos concluir que j é no máximo tão cara quanto i.

Neste caso, poderíamos substituir em G a aresta j pela aresta i sem aumentar seu custo, resultando em outra árvore gulosa G'. É fácil observar que G' possui mais arestas coincidentes a H que G, pois G' foi obtida de G a partir da remoção de uma aresta não pertencente a H (i), e a adição de uma aresta pertencente a H (j). Se G' for diferente de H, podemos repetir o processo, obtendo sempre árvores tão baratas quanto G e que possuem cada vez mais arestas em comum com H, até que mais cedo ou mais tarde a árvore obtida seja a própria H. Assim, fica demonstrado que H é tão barata quanto G. ■

Exercício 4.1. *Dado o grafo abaixo, onde os pesos das respectivas arestas estão indicados, encontre a árvore de menor custo utilizando o Algoritmo Guloso.*

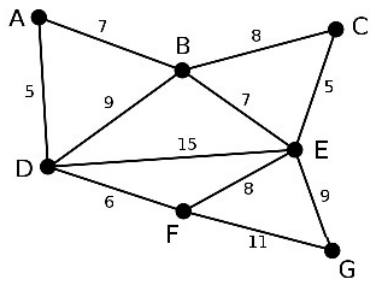


Figura 4.4 Encontrando a árvore gulosa

4.2 O PROBLEMA DO CAXEIRO VIAJANTE

Imagine que agora o dono do lago, por precaução, decida que realizará a obra de modo que para passear por dois lotes quaisquer, haja pelo menos dois caminhos possíveis sem pontes em comum. Para manter o critério de menor custo possível, ainda deverão ser construídas o mínimo de pontes possíveis. Como sabemos que para conectar os n lotes, com apenas um caminho de um lote para outro, são necessárias no mínimo $n-1$ pontes, então neste caso esse número não será mais suficiente. Mas quantas pontes a mais deveriam ser construídas para atender às novas necessidades do proprietário?

Bem, se as pontes construídas formarem um único ciclo pelos lotes do lago, serão necessárias apenas n pontes. A questão é, como selecionar o ciclo de menor custo? Pensando no grafo G que representa este lago, onde os lotes são os vértices e as pontes as arestas cujos custos são indicados por pesos, como encontrar o ciclo C contido em G, passando pelos n vértices, de modo que a soma dos pesos de suas arestas seja a menor possível?

Este é um problema de otimização combinatória, conhecido como *Problema do Caxeiro Viajante*. Seu nome foi inspirado na necessidade de vendedores, conhecidos como caxeiros viajantes, de visitar diversas cidades percorrendo o menor caminho possível, reduzindo o tempo necessário para a viagem e os possíveis custos com transporte e combustível, por exemplo, e depois retornar a sua cidade. Na verdade, este mesmo problema se aplica a diversas situações, como por exemplo a rota de menor custo para coleta de lixo pelos bairros de determinada área ou a melhor rota de entregas para o correio.

Chegar a uma resolução para esse problema não é tão simples, pois não há um algoritmo exato e de simples compreensão que leve à melhor escolha, como é o caso do algoritmo guloso na escolha da árvore gulosa. Na verdade, existem alguns métodos que funcionam na maioria das vezes, mas sua análise e discussão nãos nos convém para esse estudo, onde este problema fica apenas a título de informação como outro exemplo de otimização.

CAPÍTULO 5

SEQUÊNCIA DIDÁTICA

Esta sequência didática tem como objetivo geral fazer com que o aluno seja capaz de analisar tipos específicos de projetos, com a utilização do algoritmo guloso. Como requisito para tal capacidade, algumas atividades que envolvem indiretamente conceitos e aplicações de grafos e árvores serão trabalhadas em sala, associadas a questões contextualizadas para estimular o interesse do aluno. Para finalizar a sequência, será entregue um projeto a ser analisado pelos estudantes com o uso do algoritmo em questão, e tal análise será utilizada como instrumento final de avaliação.

As aulas serão ministradas com alunos do Ensino Médio, e a sequência se dará em cinco aulas de cinquenta minutos cada. Todas as atividades serão realizadas em grupos de cinco alunos, em média, para que possam discutir cada problema proposto. A seguir, cada aula e suas respectivas atividades serão descritas, com detalhamento dos objetivos pretendidos, conteúdos trabalhados e a metodologia utilizada, além dos instrumentos de avaliação.

5.1 AULA 1

5.1.1 Objetivos

- Estimular e direcionar os alunos a utilizarem a representação de grafos sem prévia definição e a partir daí trabalhar com o conceito de grafos e suas principais definições;

- Explicar brevemente como surgiu a Teoria dos grafos a partir do problema das pontes de Königsberg;

- Destacar a importância de grafos na resolução de problemas;

5.1.2 Conteúdos abordados

- Representação de grafos e seus elementos;

- Conceito de grau de vértice no grafo;

5.1.3 Metodologia

Será desenvolvida e trabalhada em sala a seguinte atividade:

Atividade 5.1. *O problema das pontes de Königsberg;*

(Tempo estimado: 40 minutos.)

Com a apresentação do problema das pontes como exposto a seguir (projetado com data show), será pedido aos alunos que façam uma representação da cidade com as pontes de uma maneira simples, mas que seja fiel aos elementos essenciais.

Apresentação do problema: *Há muito tempo, os cidadãos de Königsberg (onde hoje fica Kalingrado, Rússia) propuseram um desafio. A cidade era dividida em quatro distritos por braços do rio Preguel, que eram conectados por 7 pontes. Seria possível realizar um trajeto em todos os distritos passando por todas as pontes uma única vez?*

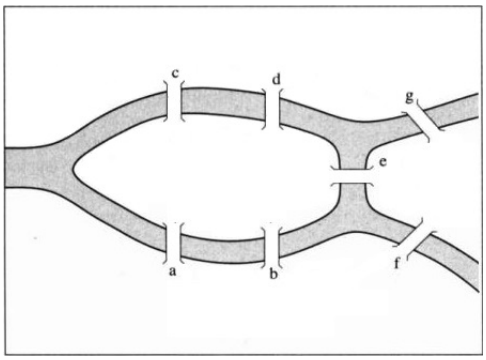


Figura 5.1 Pontes de Königsberg

Em seguida, cada grupo terá 15 minutos para tentar solucionar o desafio, apresentando para a sala argumentos que sustentem a conclusão obtida.

Quando cada equipe tiver apresentado suas considerações, o professor apresentará a real solução do problema, formalizando o conceito de grafos e seus elementos (vértices e arestas), assim como a definição de grau de um vértice, seguido de uma breve explicação da origem da Teoria dos Grafos.

5.1.4 Instrumento de avaliação

A avaliação do aprendizado dos alunos será feita a partir da atividade trabalhada, nos momentos em que cada grupo expõe seus resultados, quando o professor observará os pontos onde apareceram mais dúvidas.

5.2 AULA 2

5.2.1 Objetivos

- Utilizar grafos na resolução de problemas;
- Trabalhar o conceito de grafos dirigidos;
- Definir ciclos e conexidade em grafos;

5.2.2 Conteúdos abordados

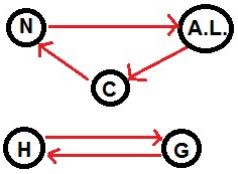
- Grafos;
- Conceitos de ciclos e conexidade em grafos;

5.2.3 Metodologia

Utilizando o recurso data show, será feita uma breve retomada do que foi visto na aula passada, e o seguinte exemplo será trabalhado:

No final do ano, cinco amigos (Neto, Ana Luiza, Hugo, Camila e Geovana) resolvem fazer uma pequena confraternização, onde pretendem realizar um amigo secreto. Geovana, que é muito curiosa, está decidida a descobrir quem cada amigo sorteou. Ela sorteou Hugo, e através dos seus métodos descobriu que Camila sorteou Neto, que por sua vez sorteou Ana Luiza. Se ela sabe que Hugo não sorteou Camila, será que essas informações foram suficientes para Geovana descobrir o amigo secreto de todos? Represente em um grafo.

Desenvolvendo o raciocínio com os alunos, a partir das informações adquiridas, como nem Geovana nem Neto sortearam Camila, apenas Hugo ou Ana Luiza poderiam tê-la sorteado. Mas Geovana sabe que não foi Hugo, logo só pode ter sido Ana Luiza. Sabendo disto, a única pessoa que pode ter sorteado Geovana é Hugo. Após essa conclusão, o seguinte grafo será mostrado:



A partir do exemplo, o professor desenvolverá o conceito de grafos direcionados, ciclos e conexidade. Em seguida, pedirá que os alunos realizem a seguinte atividade:

Atividade 5.2. *O problema das ligações telefônicas;*

Tempo estimado: 10 minutos.

Vinte anos depois da formatura, cinco colegas de turma decidem organizar uma confraternização. Para marcar o dia e o local da confraternização, precisam comunicar-se por telefone. Cada um conhece o telefone de alguns colegas e desconhece o de outros. No quadro abaixo, o número 1 indica que o colega da linha correspondente conhece o telefone do colega da coluna correspondente; o número 0 indica que o colega da linha não conhece o telefone do colega da coluna. Exemplo: Beto sabe o telefone do Dino que não conhece o telefone do Aldo.

	Aldo	*Beto*	*Carlos*	*Dino*	*Ênio*
Aldo	*1*	*1*	*0*	*1*	*0*
Beto	*0*	*1*	*0*	*1*	*0*
Carlos	*1*	*0*	*1*	*1*	*0*
Dino	*0*	*0*	*0*	*1*	*1*
Ênio	*1*	*1*	*1*	*1*	*1*

Construa um grafo representando a situação descrita e responda: Qual é o número mínimo de telefonemas que Aldo deve fazer para se comunicar com Carlos?

Após discutir os resultados da atividade, analisando as respostas dadas por cada grupo, será proposta a hipótese em que um sexto amigo, que não possui o número de ninguém e mudou de número recentemente, ou seja, nenhum dos outros amigos possui o seu número, também faça parte do grupo, para que seja reforçado o conceito de conexidade em grafos.

Como última atividade da aula será proposto o problema a seguir, e após cerca de 10 minutos para que cada equipe o solucione e discuta seus argumentos, os alunos irão expor suas conclusões, mediados pelo professor.

Atividade 5.3. *O problema do aeroporto;*

Tempo estimado: 10 minutos.

Em um país existem 9 aeroportos numerados de 1 até 9. Existe voo direto do aeroporto i para o aeroporto j se $3|ij$, com $ij = 10i + j$. Sabendo disso, existe voo (possivelmente com escalas) do aeroporto 1 para o aeroporto 9?

5.2.4 Instrumento de avaliação

A avaliação do aprendizado dos alunos será similar ao da aula anterior, feita a partir das atividades trabalhadas, onde será observada a participação de cada equipe e seu desempenho nos momentos em que cada uma delas expõe seus resultados.

5.3 AULA 3

5.3.1 Objetivos

- Estimular o interesse dos alunos através de exercícios interessantes envolvendo grafos;
- Definir passeios em um grafo;
- Definir passeio Euleriano e suas propriedades;
- Definir árvores como tipo específico de grafo.

5.3.2 Conteúdos abordados

- Passeios em um grafo;
- Passeio Euleriano;
- Conceito de árvore;

5.3.3 Metodologia

O professor começará a aula com o problema proposto na atividade a seguir. Será dado aos grupos em torno de 15 minutos para resolvê-lo e discutir suas conclusões. Será sugerido aos alunos que representem a planta da casa como um grafo para facilitar a resolução.

Atividade 5.4. *O problema do assassinato do bilionário Van Diamond*

Tempo estimado: 15 minutos.

A figura abaixo é a planta da residência do bilionário Van Diamond, que acaba de ser assassinado. Sherlock Gomes (um conhecido detetive que nas horas vagas é um estudioso

da Teoria de Grafos) foi chamado para investigar o caso. O mordomo alega ter visto o jardineiro entrar na sala da piscina (lugar onde ocorreu o assassinato) e logo em seguida deixar aquela sala pela mesma porta que havia entrado. O jardineiro, contudo, afirma que ele não poderia ser a pessoa vista pelo mordomo, pois ele havia entrado na casa, passado por todas as portas uma única vez e, em seguida, deixado a casa. Sherlock Gomes avaliou a planta da residência (conforme figura) e em poucos minutos declarou solucionado o caso.

a) Quem poderia ser o suspeito indicado por Sherlock Gomes?

b) Qual o raciocínio utilizado pelo detetive para apontar o suspeito?

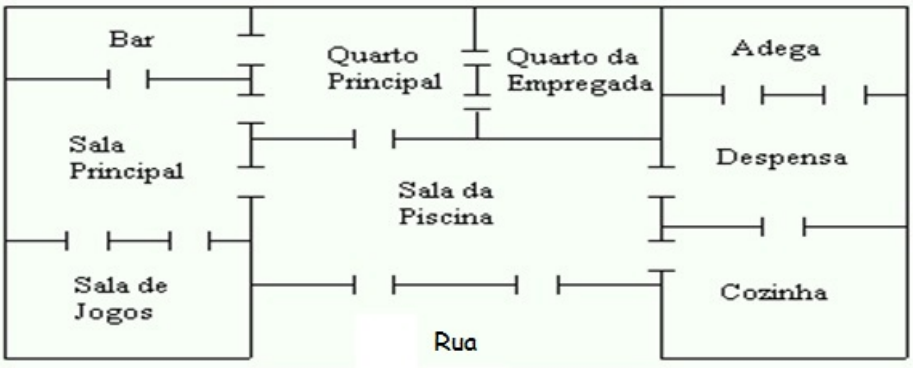


Figura 5.2 Planta da casa

Após a discussão dos resultados obtidos pelos grupos, o professor mostrará a real solução do problema, conduzindo os alunos a chegarem superficialmente ao Teorema 2.1. Formalizando o conceito de passeio em um grafo, o professor definirá passeios Eulerianos baseado na atividade realizada, trabalhando mais aprofundadamente o conteúdo do Teorema 2.1.

Finalizada esta atividade e dando continuidade a aula, cada grupo resolverá o seguinte problema:

Atividade 5.5. *Encontrando o menor grafo*

Tempo estimado: 10 minutos

Em cada um dos três casos abaixo (Figura 5.3), onde foram dados os vértices de um grafo, construa suas arestas de modo que elas estejam em menor número possível e que o grafo seja conexo.

Terminado o tempo previsto para a resolução, o professor discutirá os resultados com a sala e irá direcioná-la a chegar informalmente ao Teorema 3.2, do qual fará a demonstração de maneira simplificada. A partir daí ele trabalhará com a definição de árvores como um tipo específico de grafos. (Será pedido aos alunos que tragam na próxima aula o material produzido)

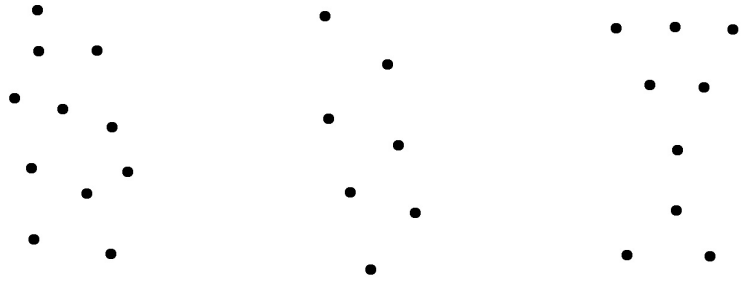


Figura 5.3 Atividade 3

5.3.4 Instrumentos de avaliação

Os alunos serão avaliados no decorrer das atividades desenvolvidas, sendo observados sua participação e interesse.

5.4 AULA 4

5.4.1 Objetivos

- Definir o conceito de árvore enraizada;
- Despertar o interesse do aluno pelo conteúdo através da brincadeira da advinhação;
- Estimular o raciocício lógico dos estudantes.

5.4.2 Conteúdos abordados

- Conceito de árvore enraizada;
- Codificação de árvores (Código de Prüfer);

5.4.3 Metodologia

Será feita uma retomada do conceito de árvores e do Teorema 3.2 (de maneira informal), mostrando que os vértices de uma árvore podem ser numerados para melhor identificação. Em seguida, o professor explicará o que é uma árvore enraizada, associando o vértice numerado como 0 à raiz.

Após estas definições, mostrará como armazenar uma árvore através do código estendido de Prüfer, utilizando para praticar com os alunos as árvores obtidas por eles (após numerar seus vértices) na aula anterior.

Escolhendo aleatoriamente uma das equipes, o professor pedirá que escolha um dos códigos obtidos e que informe o código da linha de baixo até o penúltimo número, e a partir disso advinhará o resto do código.

Despertando o interesse da turma para a brincadeira, explicará que agora ela será realizada entre os próprios grupos, realizando a atividade a seguir.

Atividade 5.6. *Advinhando árvores*
Tempo estimado: 30 minutos.

Escolhendo um componente de cada equipe, o professor explica para eles, em local isolado, como descobrir o resto do código a partir dos números fornecidos. Enquanto isso, pede que cada equipe que permaneceu na sala construa uma árvore, com o número de vértices que desejar, e após numerar seus vértices (associando o 0 *à raiz) anotem o código estendido de Prüfer referente a ela.*

Ao retornarem à sala, após verificação rápida por parte do professor se os códigos obtidos estão corretos, cada componente que estava fora deverá advinhar o código da árvore de sua respectiva equipe, a partir dos números fornecidos por elas (somente até o penúltimo da linha de baixo).

Quando todas as equipes finalizarem, cada componente que fez a advinhação deve explicar aos demais a lógica utilizada. O professor finalizará explicando para a turma em geral a lógica desta advinhação.

5.4.4 Instrumentos de avaliação

A avaliação da aprendizagem do aluno será feita a partir da atividade de advinhação de árvores realizada pelas equipes em todo seu processo: se conseguiram criar uma árvore e armazenar seu código de maneira correta; se conseguiram entender como o código Prüfer é construído e se a advinhação foi correta.

5.5 AULA 5

5.5.1 Objetivos

- Definir algoritmo guloso e mostrar algumas de suas aplicações;
- Fazer com que os alunos sejam capazes de analisar o menor custo de um projeto utilizando o algoritmo guloso.

5.5.2 Conteúdos abordados

- Algoritmo guloso (método de Kruskal) e aplicações.

5.5.3 Metodologia

O professor iniciará a aula com a definição de algoritmo guloso e algumas de suas aplicações na análise do custo de projetos. Explicará o objetivo do algoritmo através do exemplo das pontes do lago (seção 4.1), onde os lotes são os vértices e as pontes as arestas de um grafo, e que dependendo de alguns fatores, cada ponte (ou aresta) terá um custo diferente, onde o objetivo do algoritmo é conectar todos os lotes de um modo que o custo da obra seja o menor possível.

Conduzindo o raciocínio e explicando como o algoritmo realiza suas escolhas (utilizando o método de Kruskal), selecionando as arestas de menor custo e que não formem nenhum ciclo (para que não haja pontes desnecessárias), levará os alunos a deduzirem que o grafo obtido pelo algoritmo será uma árvore, chamada de árvore gulosa.

Em seguida, o professor utilizará o algoritmo - explicando as etapas de seleção de cada aresta - para encontrar a árvore gulosa no exemplo a seguir, que apresenta o grafo

original, onde as arestas estão numeradas em ordem crescente de custo (Figura 5.4).

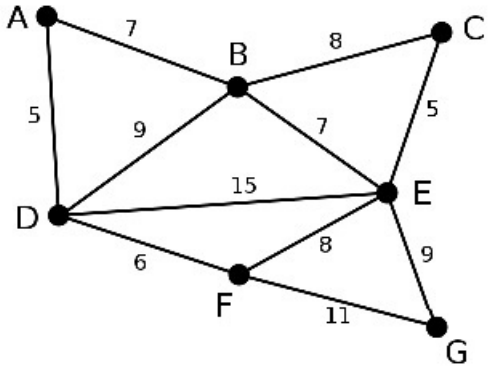


Figura 5.4 Encontrando a árvore gulosa

As arestas AD e CE são as mais leves do grafo, então ambas podem ser selecionadas primeiro. A aresta AD é escolhida aleatoriamente (Figura 5.5).

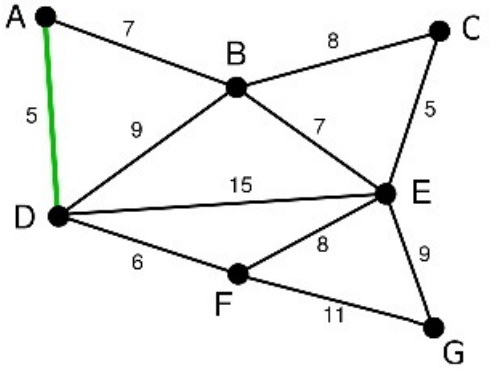


Figura 5.5 Encontrando a árvore gulosa - Passo 1

Agora a aresta CE é a de menor peso. Como ela não forma um ciclo com a aresta AD, ela é selecionada (Figura 5.6).

Agora a aresta de menor peso é a DF (peso 6). Já que não forma um ciclo com as arestas já escolhidas, então ela é selecionada (Figura 5.7).

Agora duas arestas com peso 7 (AB e BE) são as mais leves e não formam ciclo com as demais já escolhidas, então podem ser selecionadas. Escolhendo a aresta AB ao acaso, ela é adicionada. Como a aresta BD forma um ciclo com as arestas AD e AB já selecionadas, então ela é eliminada (indicada de vermelho na Figura 5.8).

Agora a aresta BE é a de menor peso e não forma ciclo com as demais já escolhidas, portando é selecionada. Similarmente ao passo anterior, as arestas BC, DE e EF são eliminadas, pois formariam ciclos com outras arestas já adicionadas (Figura 5.9).

Dentre as arestas restantes a EG é a de menor peso, portando a selecionada. Automaticamente a aresta FG é eliminada, pois se tornou desnecessária (Figura 5.10).

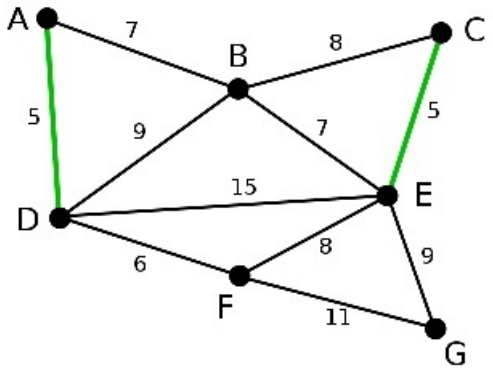


Figura 5.6 Encontrando a árvore gulosa - Passo 2

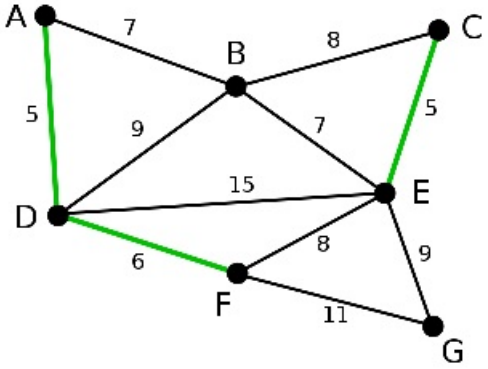


Figura 5.7 Encontrando a árvore gulosa - Passo 3

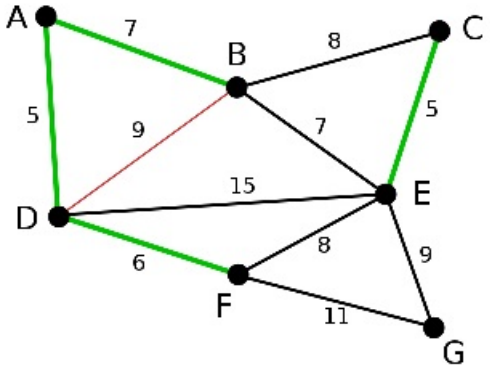


Figura 5.8 Encontrando a árvore gulosa - Passo 4

Retirando as arestas eliminadas da figura, finalmente chegamos à árvore gulosa. Note que como ela é uma árvore, se o grafo inicial possui 7 vértices, então a árvore possui 6 arestas (Figura 5.11).

Após responder a todas as dúvidas que surgirem sobre o algoritmo, o professor realizará a atividade descrita a seguir.

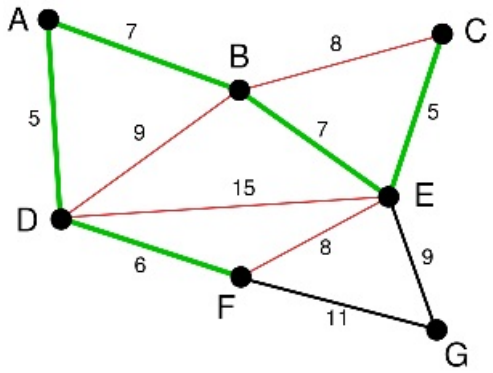


Figura 5.9 Encontrando a árvore gulosa - Passo 5

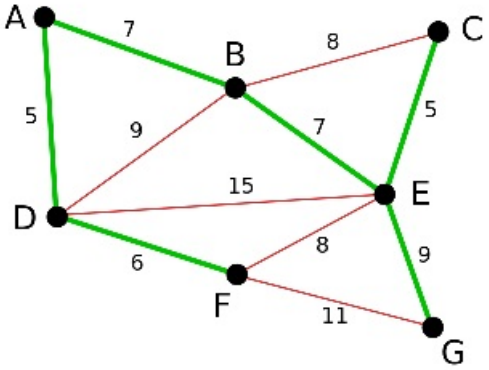


Figura 5.10 Encontrando a árvore gulosa - Passo 6

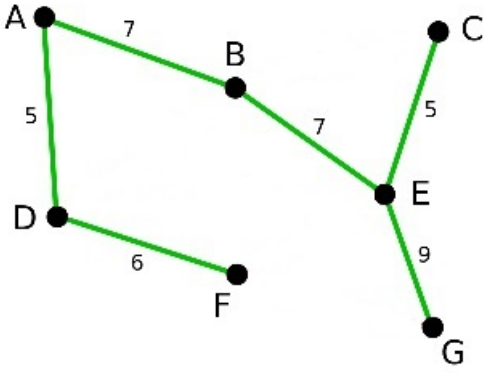
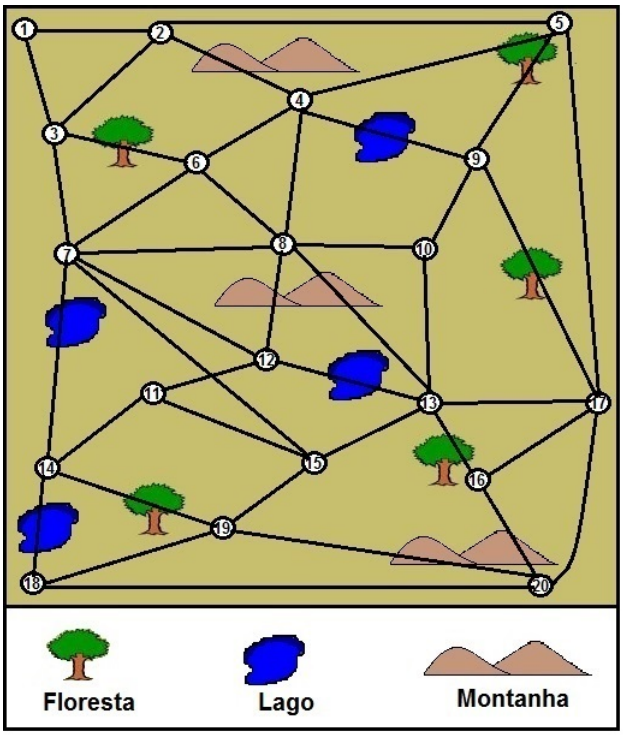


Figura 5.11 Árvore gulosa

Atividade 5.7. *Tempo estimado: 25 minutos.*

Cada um das equipes receberá o projeto de um mapa de um Estado fictício, com 20 cidades que devem ser conectadas por estradas, em forma de grafo, e uma tabela com as distâncias entre as respectivas cidades (Figura 5.12). O professor explicará que cada

uma das cidades será representada por um dos vértices, que estarão numerados de 1 a 20, e as possíveis estradas a serem construídas estão representadas por arestas. Os custos das estradas (arestas) serão diretamente proporcionais às distâncias das cidades que elas conectam. O custo por quilômetro é de 200 reais. Quando a estrada passar por um lago, floresta ou montanha, o custo da estrada, por quilômetro construído, sofrerá aumentos de 15%, 10% *e* 20%, *respectivamente.*



1 - 2: 5 km	8 - 13: 9 km
1 - 3: 4 km	9 - 10: 2,5 km
2 - 3: 6 km	9 - 17: 14 km
2 - 4: 7 km	10 - 13: 6 km
2 - 5: 18 km	11 - 12: 3,5 km
3 - 6: 5 km	11 - 14: 3,5 km
3 - 7: 4 km	11 - 15: 7 km
4 - 5: 12 km	12 - 13: 7 km
4 - 6: 3 km	13 - 15: 4 km
4 - 8: 6 km	13 - 16: 2 km
4 - 9: 8 km	13 - 17: 7 km
5 - 9: 7 km	14 - 18: 5 km
5 - 17: 20 km	14 - 19: 8 km
6 - 7: 6 km	15 - 19: 3 km
6 - 8: 3,5 km	16 - 17: 6,5 km
7 - 12: 12 km	16 - 20: 4 km
7 - 14: 10 km	17 - 20: 10 km
7 - 15: 19 km	18 - 19: 8 km
8 - 10: 5 km	18 - 20: 25 km
8 - 12: 3 km	19 - 20: 16 km

(b)

Figura 5.12 (a) Projeto a ser avaliado pelo Algoritmo Guloso (b)Distância entre cidades

Cada grupo deverá analisar cuidadosamente o mapa e através do algoritmo guloso deve indicar quais estradas seriam contruídas, a distância total e o custo da obra, para posterior comparação entre as equipes.

5.5.4 Instrumento de avaliação

A avaliação final da turma será feita a partir da análise que cada equipe fez do projeto em questão.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] LOVÁSZ, L.; PELIKÁN, J.; VESTERGOMBI, K. *Matemática discreta*. Rio de Janeiro: Sociedade Brasileira de Matemática, 2003.

[2] LUCCHESI, Cláudio Leonardo. *Introdução à Teoria dos grafos*. Rio de Janeiro: Instituto de Matemática pura e aplicada, 1979.

[3] OSTROSKI, Álvaro; MENONCINI, Luciana. *Aplicações práticas da Teoria dos Grafos*. Pato Branco: XIII ERMAC, 2009.