INTRODUÇÃO A TEORIA DOS GRAFOS E SEUS ALGORITMOS

Uma visão introdutória da teoria dos grafos

Diogo Cezar Teixera Batista

Robson Gomes de Melo

André Luiz Pires Guedes

Sumário

Lista de Figuras

Capítulo 1

Grafos - Definições Iniciais

Para qualquer conjunto V, denotamos $\binom{V}{2}$ o conjunto de todos pares não-ordenados de elementos de V. Se V tem n elementos então $\binom{V}{2}$ tem $\frac{n(n-1)}{2}$ elementos. Os elementos de $\binom{V}{2}$ terá a forma $\{v, w\}$, sendo v e w dois elementos distintos de V.

Um grafo é um par ordenado de conjuntos finitos (V, E) tal que $E \subseteq \binom{V}{2}$. Isso significa que cada elemento E é um subconjunto de elementos de V ou um binômio de V. Cada elemento de V é chamado de *vértice* do grafo e cada elemento de E é chamado de *aresta* do grafo, dessa forma cada aresta $e \in E$ é um subconjunto de V formado por exatamente dois vértices, ou seja, $e \subseteq V$ e $|e| = 2$.

Todo grafo pode ser representado geometricamente por um diagrama. No plano, desenhamos um ponto para cada vértice e um segmento de curva ligando cada par de vértices que determinam uma aresta (Figura 1.1). Claramente, essa representação geométrica de um grafo não é única.

Exemplo 1: Os conjuntos:

$$\begin{aligned} V &= \{1, 2, 3, 4, 5, 6, 7, 8\} \\ E &= \{\{1, 2\}, \{1, 5\}, \{2, 5\}, \{3, 4\}, \{6, 7\}\} \end{aligned}$$

definem um grafo e um diagrama desse grafo é apresentado na Figura 1.1.

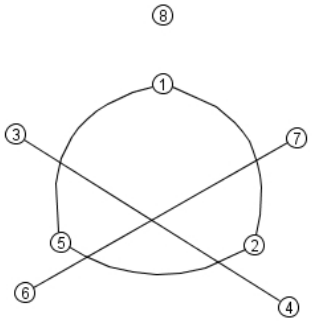

Figura 1.1: Representação geométrica (ou diagrama) do grafo.

Se G denota o grafo que é definido pelo par (V, E) então escrevemos $G = (V, E)$.

No que segue u e v denotam vértices e e e f denotam arestas de um grafo:

- se $v \in e$ então dizemos que a aresta e *incide* em v;
- se u, v é uma aresta, então dizemos que u e v são vértices *adjacentes*;
- se $|e \cap f| = 1$, então dizemos que e e f são arestas *adjacentes*, em outras palavras, se existe um vértice v que é comum (faz a intersecção) às arestas e e f então essas são arestas adjacentes.
- quando $e = v, u$, dizemos que u e v são os extremos da aresta e.

Quando nos referimos a um grafo conhecido G sem especificarmos o conjunto dos vértices e o conjunto das arestas que definem G esses passam a ser referidos como $V(G)$ e $E(G)$, respectivamente. Assim, se $G = (X, Y)$ então $V(G) = X$ e $E(G) = Y$.

Para um grafo G qualquer, chamamos $|V(G)|$ de ordem de G e chamamos $|V(G)|+|E(G)|$ de tamanho de G.

Um expoente em G, quando G é um grafo, denota a ordem de G, assim quando queremos ressaltar que G é um grafo de ordem n, para algum $n \in \mathbb{N}$, escrevemos G^n.

Chamamos $G^0 = (\varnothing, \varnothing)$ de grafo vazio e todo grafo de ordem 1 de grafo trivial.

1.1 Grafo Completo

Um grafo é chamado de completo sobre V se todo par de vértices de V é uma aresta do grafo, ou formalmente $E = \binom{V}{2}$ para cada par de vértices $V(G) = \{x, y\}$ existe uma aresta que os liga. Um grafo completo é denotado por K^n.

Outra definição para **grafo completo** é um grafo que tem todas as arestas possíveis em $E(G)$. $K_n = (V, \binom{V}{2})$ onde $V = \{1, 2, 3, 4, \ldots, n\}$.

São exemplos de grafos completos:

$K_1 = (V_{k1}, E_{k1}) \therefore V_{k1} = \{1\}$ e $E_{k1} = \{\{\varnothing, \varnothing\}\}$

$K_2 = (V_{k2}, E_{k2}) \therefore V_{k2} = \{1, 2\}$ e $E_{k2} = \{1, 2\}$

$K_3 = (V_{k3}, E_{k3}) \therefore V_{k3} = \{1, 2, 3\}$ e $E_{k3} = \{\{1, 2\}, \{2, 1\}, \{1, 3\}\}$

As representações geométricas dos grafos estão dispostas na Figura 1.2 na qual, K_1 é representado por a, K_2 é representado por b e K_3 é representado por c.

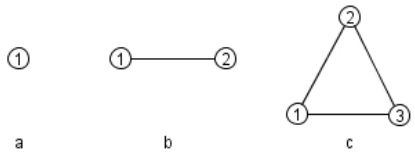

Figura 1.2: Representação geométrica (ou diagrama) dos grafos completos K_1, K_2 e K_3

As definições de K para um n completo representam um grafo com n vértices completo, então essa definição pode ser aplicada a todos os sucessivos grafos até K_n.

1.2 Complemento de um Grafo

O complemento de um grafo G, denotado por $\overline{G}$, é o grafo que tem o mesmo conjunto de *vértices* de G e dois vértices formam uma aresta em $\overline{G}$ se e somente se não formam uma aresta de G:

$$\begin{aligned}
\overline{G} &= (\overline{V}(G), \overline{E}(G)) \\
\overline{V}(G) &= V(G) \\
\overline{G} &= (V(G), \overline{E}(G)) \\
\overline{E}(G) &= \{\{u,v\} \subset V(G) : \{u,v\} \notin E(G)\} \text{ ou} \\
\overline{E}(G) &= \binom{V(G)}{2} \setminus E(G) \text{ ou} \\
\overline{E}(G) &= \{e \in \binom{V(G)}{2} \therefore e \notin E(G)\}
\end{aligned}$$

1.3 Graus de um Grafo

Os graus de um grafo indicam o número de arestas conectadas a ele, por isso é um importante parâmetro para estudo.

Em um grafo definido por:

$$\begin{aligned}
G &= (V(G), E(G)) \\
V(G) &= \{1,2,3,4,5,6\} \\
E(G) &= \{\{1,3\},\{1,6\},\{2,4\},\{5,1\},\{4,3\}\}
\end{aligned}$$

Disposto graficamente na Figura 1.3.

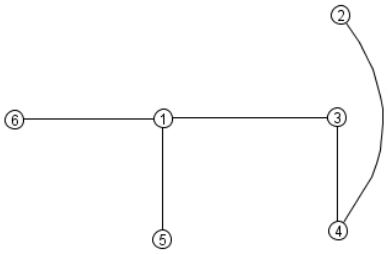

Figura 1.3: Representação geométrica (ou diagrama) do grafo G.

Defini-se como **vértices vizinhos** os vértices que estão conectados as mesmas arestas que o vértice em questão:

$$N_G(u) = \{w \in V(G) \therefore uw \in E(G)\}$$

Ou seja, para descobrir os vértices vizinhos (vizinhança) de um vértice u verifica-se se existe um vértice w que pertence ao conjunto de vértices de G tal que a aresta identificada por uw pertença ao conjunto de arestas de G.

Exemplo 2: Para exemplificar a identificação da vizinhança, utiliza-se o vértice 1 da Figura 1.3:

$$\begin{aligned} N_G(u) &= N_G(1) \\ N_G(1) &= \{6, 5, 3\} \end{aligned}$$

O **grau de um vértice** (denotado por $d_G(u)$) é obtido pelo tamanho do conjunto de sua vizinhança, no exemplo:

$$\begin{aligned} d_G(1) &= |N(1)| \\ d_G(1) &= 3 \end{aligned}$$

Defini-se como **arestas vizinhas** as arestas que incidem no vértice em questão:

$$E_G(u) = \{e \in E(G) \therefore u \in e\}$$

Ou seja, para identificar as arestas vizinhas de um vértice u verifica-se se uma aresta e pertence ao conjunto de arestas de G tal que o vértice u esteja no conjunto de arestas e em questão.

Exemplo 3: Para exemplificar a identificação das arestas vizinhas, utiliza-se o vértices 1 da Figura 1.3:

$$\begin{aligned} E_G(u) &= E_G(1) \\ E_G(1) &= \{\{1, 6\}, \{1, 5\}, \{1, 3\}\} \end{aligned}$$

1.3.1 Vetor de Graus

Um vetor de grau de um grafo, é um vetor contendo os graus de todos os vértices do grafo.

O tamanho do vetor de graus é também o número de vértices do grafo, no exemplo da Figura 1.3, o vetor de grau é dado por:

$$vet_G = \{1, 1, 1, 2, 2, 3\}$$

Pois existem 3 vértices com grau 1, 2 vértices com grau 2 e 1 vértices com grau 3.

Sua notação formal é obtida por:

$$d_G(u) \therefore u \in V(G)$$

Ainda é possível identificar se um vetor de graus é ou não um grafo:

- é um grafo se a somatória dos graus de valor ímpar é par (Teorema 1.4);
- não é um grafo se o maior grau do vetor de graus é igual ou maior que o tamanho do vetor de graus.

1.3.2 Grau Mínimo

Para achar o **grau mínimo** (denotado por δ) em um grafo, deve-se identificar o menor grau do seu vetor de graus.

Dado o grafo da Figura 1.3, temos:

$$\begin{aligned} \delta(G) &= min\{1,1,1,2,2,3\} \\ \delta(G) &= 1 \end{aligned}$$

Definindo-se formalmente, temos:

$$\delta(G) = min\{d_g(u) \therefore u \in V(G)\}$$

1.3.3 Grau Máximo

Para achar o **grau máximo** (denotado por Δ) em um grafo, deve-se identificar o maior grau do seu vetor de graus.

Dado o grafo da Figura 1.3, temos:

$$\begin{aligned} \Delta(G) &= max\{1,1,1,2,2,3\} \\ \Delta(G) &= 3 \end{aligned}$$

Definindo-se formalmente, temos:

$$\Delta(G) = max\{d_g(u) \therefore u \in V(G)\}$$

1.3.4 Grau Médio

Para achar o **grau médio** (denotado por d) em um grafo G determina-se $\frac{1}{|V(G)|}$ (1 sobre a quantidade de vértices) vezes o somatório de todos os graus do vetor de graus ($\sum_{u \in V(G)} d_g(u)$).

Formalmente, temos:

$$d(G) = \frac{1}{|V(G)|} \sum_{u \in V(G)} d_g(u)$$

No exemplo da Figura 1.3, temos:

$$\begin{aligned} d(G) &= \frac{1}{6} \sum_{u \in V(G)} d_g(u) \\ d(G) &= \frac{1}{6}(8) \\ d(G) &= \frac{8}{6} \therefore \frac{4}{3} \end{aligned}$$

Outra forma mais simples para achar o *grau médio* é somar os graus do vetor de graus e dividir pelo tamanho do vetor de graus.

1.3.5 Grafo n-regular

Um grafo G é dito n-regular, para um $n \in \mathbb{N}$ se todos seus vértices têm grau igual a n.

1.4 Teorema da Soma dos Graus

$$\sum_{u \in V} d_g(u) = 2|E(G)|$$

Prove que o somatório dos graus de um vértice do grafo G com $u \in V(G)$ é igual a 2 vezes o tamanho das arestas de G.

Para entender melhor esse teorema vamos usar o seguinte grafo:

$$\begin{aligned} G_t &= (V(G_t), E(G_t)) \\ V(G_t) &= \{1, 2, 3, 4, 5\} \\ E(G_t) &= \{\{1, 4\}, \{1, 3\}, \{4, 3\}, \{3, 5\}, \{3, 2\}\} \end{aligned}$$

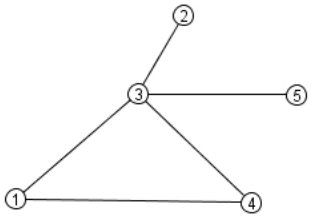

Figura 1.4: Representação geométrica (ou diagrama) do grafo G_t.

Analisando o teorema temos que o somatório dos graus do vértice deve ser igual a 2 vezes o número de arestas do grafo. No caso o somatório dos graus do vértice é $\sum_{u \in V} d_g(u) = 10$ e o número de arestas é 5, então $10 = 2(5)$.

Para cada aresta $e \in E(G)$ removida do grafo, consequentemente remove-se a ligação entre dois vértices, o que diminui 2 graus no grafo todo, logo podemos identificar uma relação de 2 para 1 (dois graus para uma aresta) entre as arestas e graus. O que confirma o teorema.

Demonstração: Seja (V, E) um grafo, podemos definir o conjunto $X = \{(u, e) \in VxE \therefore u \in e\}$. O conjunto X é um conjunto de um vértice u ligado a uma aresta e tal que esse vértice pertence a essa aresta. Ainda poderíamos pensar que o conjunto X, por um momento, relaxa as definições de grafos e duplica as arestas para que cada vértice seja separado do grafo no conjunto X, e leve consigo as arestas a ele conectadas. Podemos contar os elementos de X de duas formas:

1. Cada vértice u participa de $d_g(u)$ dos elementos de X, portanto podemos deduzir que o tamanho de X ($|X|$) é o mesmo que o somatório dos graus do grafo, ou formalmente
$$|X| = \sum_{u \in V} d_g(u)$$
2. Cada aresta e está presente em dois elementos de X, logo: $|X| = 2|E(G)|$.

De 1 e 2 provamos o Teorema na Seção 1.4.

1.4.1 Corolário dos Graus

Em todo grafo o número de vértices com grau ímpar é par.

Demonstração: Seja G um grafo, denote por I o subconjunto formado pelos vértices em $V(G)$ de grau ímpar e denote por P o subconjunto dos vértices de grau par.

$$\begin{aligned} I &= \{v \in V(G) \therefore d_g(v) \text{ impar}\} \\ P &= \{v \in V(G) \therefore d_g(v) \text{ par}\} \end{aligned}$$

Usando que $I \cap P = \varnothing$ e $I \cup P = V(G)$, e o Teorema da Seção 1.4 temos:

$$\sum_{u \in V(G)} d_g(u) = 2|E(G)|$$

$$\sum_{u \in I} d_g(u) + \underbrace{\sum_{u \in P} d_g(u)}_{par} = \underbrace{2|E(G)|}_{par}$$

Portanto devemos ter $\sum_{u \in I} d_g(u)$ par, o que somente é possível quando o tamanho de I ($|I|$) é par.
Entretanto podemos avançar um pouco mais na demonstração e definir:

- Temos um número n qualquer *par* se definido como $n = 2k$;
- Temos um número n qualquer *ímpar* se definido como $n = 2k + 1$.

$$\sum_{u \in I} d_g(u) = \sum_{u \in I} (2kv + 1)$$

Então,

$$\underbrace{2|E(G)|}_{par} = \Big(\underbrace{\sum_{u \in I} 2kv}_{par} + \sum_{u \in I} 1\Big) + \underbrace{\sum_{u \in P} d_g(u)}_{par}$$

Logo, o tamanho do resto do subconjunto dos *ímpares* ($|I|$) deve ser *par*.

Capítulo 2

Subgrafos

Um subgrafo de um grafo G é um grafo H tal que:

- $V(H) \subseteq V(G)$
- $E(H) \subseteq E(G)$
- H deve ser um grafo

Seja um grafo G definido por:

$$\begin{aligned} G_c &= (V(G_c), E(G_c)) \\ V(G_c) &= \{1, 2, 3, 4\} \\ E(G_c) &= \{\{1, 2\}, \{2, 3\}, \{3, 4\}, \{4, 1\}\} \end{aligned}$$

Representado graficamente por pela Figura 2.1.

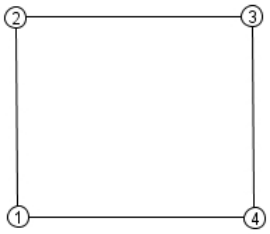

Figura 2.1: Representação geométrica (ou diagrama) do grafo G_c.

São representações de subgrafos?

1. $V(H_c) = \{1, 2, 3\}$
   $E(H_c) = \{\{3, 4\}, \{1, 2\}\}$
   O elemento 1 não é um subgrafo de G_c, pois a aresta $\{3, 4\}$ contém um vértice não definido no conjunto de vértices dado, ou seja, não é um grafo.

2. $V(H_c) = \{1, 2, 3\}$
   $E(H_c) = \{\{1, 2\}\}$
   O elemento 2 é um subgrafo de G_c.

3. $V(H_c) = \{1, 2, 3, 4\}$
   $E(H_c) = \{\varnothing\}$
   O elemento 3 é um subgrafo de G_c.

4. $V(H_c) = \{1, 2, 3, 4\}$
   $E(H_c) = \{\{1, 3\}\}$
   O elemento 4 não é um subgrafo de G_c, pois a aresta definida não existe no grafo original, ou seja, não é um grafo.

2.1 Subgrafo Gerador

H é um subgrafo gerador de G se H é um subgrafo de G e $V(H) = V(G)$. A restrição para um subgrafo ser gerador de algum grafo é que ele deve possuir os mesmos vértices do grafo original.

Por exemplo, a Figura 2.2 mostra dois grafos G e H, no qual H é gerador de G.

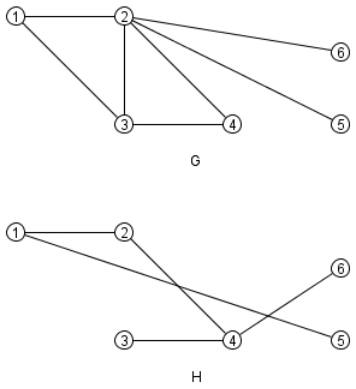

Figura 2.2: H é um subgrafo gerador de G.

Note que o grafo H possui todos os vértices de G, entretanto não possui as mesmas arestas.

2.2 Subgrafo Induzido

Um subgrafo induzido H é um subgrafo de um grafo G, no qual a partir de um conjunto de vértices ou arestas pertencentes ao grafo original G, é possível se obter sua indução, denotada por $G[K]$, onde K é o conjunto de arestas ou vértices que servirá para a indução.

2.2.1 Por um Conjunto de Vértices

Um subgrafo induzido por um conjunto de vértices de G é um subgrafo $G[S]$ de G tal que:

$$
\begin{aligned}
S &\subseteq V(G) \\
G[S] &= (S, E') \\
E' &= \{e \in E(G) \therefore e \subseteq S\} \text{ ou} \\
E' &= \{e \in E(G) \therefore e \in \binom{S}{2}\} \text{ ou} \\
E' &= E(G) \cap \binom{S}{2}
\end{aligned}
$$

As três definições de E' representam todas as arestas possíveis em S que pertencem a $E(G)$.

Em um subgrafo induzido por um conjunto de vértices, seu conjunto de vértices (V(G(S)) passa a ser o próprio conjunto dado, ou seja S. O seu conjunto de arestas é formado por todas as arestas possíveis que interligam os vértices de S, desde que elas também estejam no grafo original G.

Para exemplificar, a extração de um subgrafo induzido por um conjunto de vértices é dado um grafo G, composto por:

$$
\begin{aligned}
G &= (V, E) \\
V &= \{1, 2, 3, 4, 5, 6\} \\
E &= \{\{1, 6\}, \{1, 2\}, \{2, 5\}, \{2, 4\}, \{2, 3\} \\
&\quad \{5, 6\}\}
\end{aligned}
$$

Representado graficamente pela Figura 2.3

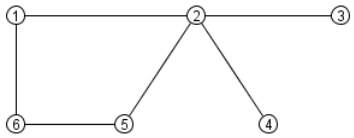

Figura 2.3: Representação gráfica do grafo G.

Demonstração: Obtenha o grafo induzido $G[S]$ com $S = \{1, 2, 5, 6\}$.

Devemos então copiar os vértices passados pelo conjunto S e então ligar todas as arestas possíveis que também estão em G. E assim obtemos o grafo $G[S]$ representado pela Figura 2.4.

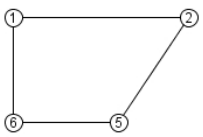

Figura 2.4: Representação gráfica do grafo $G[S]$.

Sua representação formal é dada por:

$$\begin{aligned} G[S] &= (V_s, E_s) \\ V_s &= \{1, 2, 5, 6\} \\ E_s &= \{e \in E(G) \therefore e \subseteq S\} \\ E_s &= \{\{1, 6\}, \{1, 2\}, \{2, 5\}, \{5, 6\}\} \end{aligned}$$

2.2.2 Por um Conjunto de Arestas

Um subgrafo induzido por um conjunto de arestas M de G é um subgrafo $G[M]$ de G tal que:

$$\begin{aligned} M &\subseteq E(G) \\ G[M] &= (V', M) \\ V' &= \bigcup_{e \in M} e \end{aligned}$$

Em um subgrafo induzido pelo conjunto de arestas seu conjunto de arestas (E(G(M)) passa a ser o próprio conjunto dado, ou seja M. O seu conjunto de vértices é formado pela união exclusiva do seu conjunto de arestas, o que resulta em um conjunto com todos os vértices necessários para as ligações entre as arestas passadas como parâmetro.

Demonstração: Utilizando o exemplo da Figura 2.3, obtenha o grafo induzido $G[M]$ com $M = \{\{1, 2\}, \{2, 5\}, \{1, 6\}\}$.
Devemos então verificar o que resulta de $\bigcup_{e \in M} e$.
Obtemos então o conjunto $V' = \{1, 2, 5, 6\}$, sua representação gráfica está na Figura 2.5

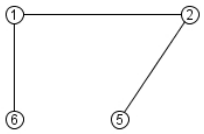

Figura 2.5: Representação gráfica do grafo $G[M]$.

Capítulo 3

Propriedades dos Grafos

3.1 Clique

Em um grafo, podemos encontrar pequenos subgrafos que formam grafos completos (que são chamados de cliques em G).

Um subconjunto de vértices de um grafo G, denotado por U é uma **clique** de G se $G[U]$ é um grafo completo.

Em um mesmo grafo, podemos encontrar cliques de diferentes *ordens*, pois podem existir diferentes subgrafos que podem ser induzidos de G tal que formem um subgrafo completo.

O **tamanho da clique** é denotado pelo tamanho do conjunto de vértices que formam o subgrafo induzido de um determinado G.

Para exemplificar vamos analisar o grafo da Figura 3.1

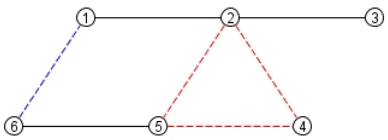

Figura 3.1: Representação gráfica do grafo $G[M]$.

No grafo da Figura 3.1 podemos observar dois destaques, a aresta $\{1, 6\}$ (destaca em azul e pontilhado) forma uma clique de tamanho 2. Isso pois se analisarmos o subgrafo induzido por vértices $G[S]$, com $S = \{1, 6\}$ ele é um grafo completo. Outras cliques de tamanho 2 também estão presentes no grafo. O destaque em vermelho e pontilhado denota uma clique de tamanho 3, pois se analisarmos o subgrafo induzido por vértices $G[S]$, com $S = \{2, 4, 5\}$ ele é também um grafo completo.

3.2 Conjunto Independente

O conjunto independente pode ser definido com o oposto de uma clique em um grafo G qualquer.

Dado um grafo G, e U um conjunto de vértices de $V(G)$, U é dito conjunto independente se $G[U] = (V, \varnothing)$.

Ou seja, um subconunto de vértices que não possui ligação de uma aresta entre eles.

Para exemplificar um conjunto independente podemos analisar o grafo da Figura 3.2

O conjunto U é dado por:

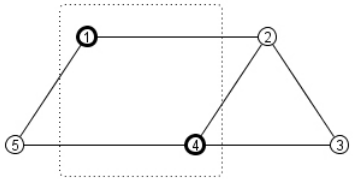

Figura 3.2: Conjunto independe U no grafo G

$$U = \{1, 4\}$$

Assim como na clique o tamanho ou ordem do conjunto independente é dado pelo tamanho do conjunto de vértices do subgrafo induzido que forma o conjunto independente.

3.2.1 Teorema da Clique e Conjunto Independente Máximos

Seja:

$$\begin{aligned} \omega(G) &= \text{max}\{|A| \therefore A \subseteq V(G) \text{ e } A \text{ é clique} \\ \alpha(G) &= \text{max}\{|A| \therefore A \subseteq V(G) \text{ e } A \text{ é um conj. indep.} \end{aligned}$$

Prove que $\omega(G) = \alpha(\overline{G})$.

Prova: Seja S uma clique de G com $|S| = \omega(G)$. Logo $G[S] = (S, \binom{S}{2})$.

Em $\overline{G}$ uma aresta que pertence a G é uma aresta se e somente se não pertence a $E(G)$. Logo, ($e \in E(\overline{G}) \nLeftrightarrow E(G)$). Se S é uma clique em G, em seu grafo complementar $\overline{G}$, S será um conjunto independente. Logo, $\overline{G}[S] = (S, \varnothing)$.

Como $\omega(G) = |S|$ e $\alpha(\overline{G}) \geqslant S$. Ou seja o tamanho do conjunto S criado é o tamanho da clique máxima encontrada em G e como essa clique é um conjunto independente em $\overline{G}$, então existe pelo menos um conjunto independente em $\overline{G}$ do mesmo tamanho ou maior que S.

Logo, $\omega(G) \leqslant \alpha(\overline{G})$.

Seja U um conjunto independente em $\overline{G}$ com $|U| = \alpha(\overline{G})$. Logo $\overline{G}[U] = (U, \varnothing)$.

Logo $G[U] = (U, \binom{U}{2})$.

Em G uma aresta que pertence a $\overline{G}$ é uma aresta se e somente se não pertence a $E(\overline{G})$. Logo, ($e \in E(G) \nLeftrightarrow E(\overline{G})$). Se U é um conjunto independente em G, em seu grafo complementar $\overline{G}$, U será uma clique.

Como $\alpha(\overline{G}) = |U|$ e $\omega(G) \geqslant |U|$. Logo $\omega(G) \geqslant \alpha(\overline{G})$.

Então, como:

$$\begin{aligned} \omega(G) &\geqslant \alpha(\overline{G}) \text{ e} \\ \omega(G) &\leqslant \alpha(\overline{G}) \text{ então} \\ \omega(G) &= \alpha(\overline{G}) \end{aligned}$$

3.3 Grafo Bipartido

Um grafo G é bipartido se existem dois conjuntos independentes A e B em G que particionam $V(G)$, ou seja $A \cup B = V(G)$ e $A \cap B = \varnothing$.

Exemplo 4: Para exemplificar o conceito de grafo bipartido, vamos analisar o grafo da Figura 3.3.

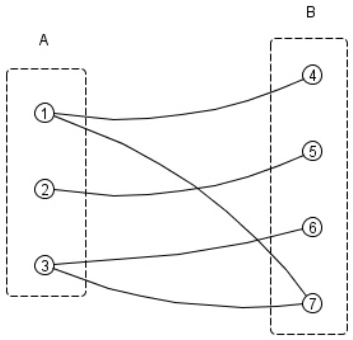

Figura 3.3: Exemplo de Grafo Bipartido.

É possível identificar dois conjuntos no grafo, os elementos do conjunto A são $V(A) = \{1, 2, 3\}$ e apenas se interligam com os elementos do conjunto B dado por $V(B) = \{4, 5, 6, 7\}$.

O Algoritmo 1 mostra um algoritmo para achar dois conjuntos A e B em um grafo bipartido.

Algoritmo 1: Verifica se G é bipartido ou não

```
1  Bipartido(G);
2  início
3      A[] ← lista vazia de vértices;
4      B[] ← lista vazia de vértices;
5      u ←vértice aleatório de V(G);
6      enquanto u não for o último vértice de V(G) faça
7          se N_G(u) está em A e N_G(u) está em B então
8              o grafo G não é bipartido;
9          se N_G(u) está em A então
10             adicione u em B;
11         senão
12             adicione u em A;
13         u ← o próximo vértice vizinho não visitado de u ∈ V(G);
```

No Algoritmo 1, criamos duas listas de vértices A e B que serão os conjuntos que no final da execução conterão os elementos que não se conectam. Inicialmente se escolher um u qualquer, e para todo u do conjunto de vértices se verifica se algum de seus vizinhos está no grupo A, se estiver ele é colocado em B, caso contrário em A.

Exemplo 5: O seguinte G é definido por:

$$\begin{aligned} V(G) &= \{1,2,3,4,5,6,7,8\} \\ E(G) &= \{\{1,6\},\{6,2\},\{3,7\},\{3,8\},\{7,4\}, \\ &\quad \{7,5\}\} \end{aligned}$$

é bipartido, pois $V(G) = A \cup B$ e $V(A) = \{1,2,3,4\}$ e $V(B) = 6,7,8$ e tanto A, como B são conjuntos independentes.

No mesmo grafo ainda é possível identificar outros conjuntos que deixam o grafo bipartido. Seja A' e B', se $V(A') = \{1,2,7,6\}$ e $V(B') = \{2,3,5,4\}$ então A' e B' formam conjuntos independentes.

Um grafo pode ser bipartido formando diferentes conjuntos independentes. Como pode ser observado com os conjuntos A e B e A' e B'.

3.3.1 Subgrafo Bipartido Completo

Um grafo bipartido G com partes A e B é dito **completo** se tem $|A| \cdot |B|$ arestas. ou seja, $E(G) = \{\{a,b\} \subseteq V(G) \therefore a \in A \text{ e } b \in B$. Um grafo bipartido completo com partes de cardinalidade n e m é denotado por $K^{n,m}$.

Todos os elementos do conjunto A devem se ligar a todos elementos do conjunto B e vice versa, como podemos observar no grafo da Figura 3.4.

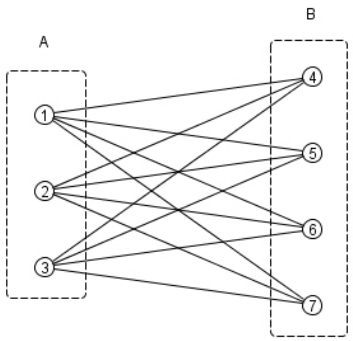

Figura 3.4: Exemplo de Grafo Bipartido Completo.

No grafo de exemplo da Figura 3.4, podemos representar esse subgrafo bipartido completo por $K^{n,m}$.

3.4 Corte

Um corte é um conjunto de arestas que "separam" dois conjuntos de vértices.

Dados $A, B \subseteq V(G)$ e $A \cap B = \varnothing$. Então, $E(A,B) = \{\{u,v\} \in E(G) \therefore u \in A \text{ e } v \in B\}$.

Para exemplificar o conceito de corte vamos usar o grafo G da Figura 3.5.

Então o corte $E(A,B)$ para o grafo G indicado na Figura 3.5, é dado por: $E(A,B) = \{\{1,2\},\{5,4\}\}$.

Para identificar um corte em um grafo G definido por:

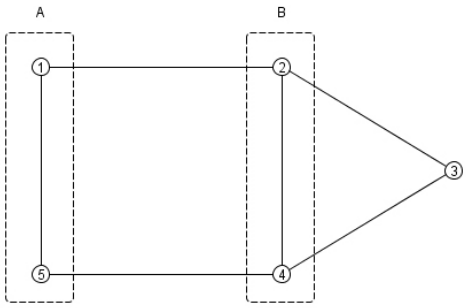

Figura 3.5: Exemplo de Corte em um Grafo.

$$\begin{aligned} V(G) &= \{1,2,3,4,5\} \\ E(G) &= \{\{1,2\},\{2,5\},\{5,4\},\{4,3\},\{4,2\} \\ &\quad \{1,5\}\} \end{aligned}$$

podemos seguir 2 passos:

1. Identificar 2 conjuntos de vértices tais que, se um conjunto for o conjunto $T \subseteq V(G)$, então $\overline{T} = V(G) \setminus T$;

2. Verificar se existem uma ou mais arestas que ligam os elementos do conjunto T com o conjunto $\overline{T}$.

Sejam dois conjuntos T e $\overline{T}$ dados por:

$$\begin{aligned} T &= \{1,2\} \\ \overline{T} &= \{3,4,5\} \end{aligned}$$

Então, um corte do tipo $E(T,\overline{T}) = \{u,v\} \subseteq V(G) \therefore u \in Tev \in \overline{T})$ ou seja, $E(T,\overline{T}) = \{\{3,1\},\{2,4\},\{2,5\}\}$. O grafo G e os cortes (destacados em pontilhado) podem ser observados na Figura 3.6

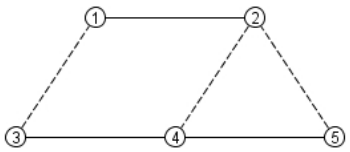

Figura 3.6: Outro Exemplo de Corte em um Grafo.

Observação: Os cortes interessantes são do tipo $E(S,\overline{S})$.

Com o conceito de corte, podemos definir um **grafo bipartido** de outra nova maneira:

G é um grafo bipartido se existe $S \subset V(G)$ não vazio tal que $E(G) = E(S,\overline{S})$ e não exista arestas entre S e $\overline{S}$.

3.5 Isomorfismo

Dizemos que os grafos G e H são **isomorfos**, e nesse caso escrevemos $G \simeq H$, se existe uma função bijetora.

Uma função bijetora garante que cada elemento do conjunto A vai possuir um elemento **correspondente** no conjunto B.

$$f : V(G) \to V(H)$$

tal que,

$$\{u, v\} \in E(G) \Leftrightarrow \{f(u), f(v)\} \in E(H)$$

Para que dois grafos sejam isomorfos, devem ter:

- o mesmo tamanho de arestas;
- o mesmo tamanho de vértices;
- o mesmo vetor de graus;
- uma função bijetora do tipo $f : V(G_1) \to V(H)$.

Exemplo 6: Sejam G_1, G_2 e G_3 os grafos definidos por:

$$\begin{aligned}
V(G_1) &= \{1,2,3,4,5,6\} \\
V(G_2) &= V(G_1) \\
V(G_3) &= V(G_2) \\
E(G_1) &= \{\{1,2\},\{2,3\},\{3,4\},\{4,5\},\{5,6\}, \\
&\quad \{6,1\}\} \\
E(G_2) &= \{\{1,3\},\{3,5\},\{5,1\},\{2,4\},\{4,6\}, \\
&\quad \{6,2\}\} \\
E(G_3) &= \{\{1,3\},\{3,5\},\{5,6\},\{6,2\},\{2,4\}, \\
&\quad \{4,1\}\}
\end{aligned}$$

e dispostos na Figura 3.7.

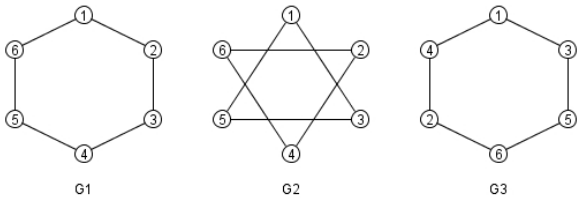

Figura 3.7: Exemplo de Isomorfismo.

Pergunta-se, G_1, G_2 e G_3 são isomorfos entre sí?

Os grafos G_1 e G_3 são isomorfos, pois têm o mesmo formato, além do mesmo vetor de graus, mesmo número de arestas, mesmo número de vértices e pode-se identificar a seguinte função bijetora:

$$f : V(G_1) \rightarrow V(G_3)$$

1	1
2	3
3	5
4	6
5	2
6	4

Pois se testarmos para cada aresta do grafo G_1 existe uma correspondente no grafo G_3. Por exemplo a aresta $\{1, 2\}$ ao aplicarmos a função se transforma na aresta $\{1, 3\}$ que está presente no conjunto de arestas de G_3. Para que eles sejam isomorfos, ao se aplicar a função bijetora em todas as arestas de G_1 devemos obter uma equivalente em G_3.

O grafo G_2 não é isomorfo a G_1 ou G_3, pois em G_2, o conjunto de vértices $\{1, 3, 5\}$ forma um triângulo K_3 que não está presente em G_1 ou G_3.

Grafos isomorfos entre si possuem o mesmo desenho (visualmente), entretanto a localização dos vértices podem ser diferentes como é possível observar em G_1 e G_3. Grafos são quase iguais, o que os torna diferentes é o nome dado aos seus vértices.

Para montar a tabela da função bijetora, devemos testar todas as combinações possíveis entre os vértices dos grafos dados. Para um grafo com um número expressivo de vértices, uma alternativa mais eficiente é:

- identificar e separar vértices do mesmo grau nos dois grafos;
- fazer a permutação entre esses vértices;
- testar as combinações entre todas as permutações dos graus.

Outra definição para **isomorfismo** é que dois grafos G e H são isomorfos se existe um **isomorfismo** entre eles. Em outras palavras, dois grafos são isomorfos se é possível alterar os nomes dos vértices de um deles de tal modo que os dois grafos fiquem iguais.

Para decidir se dois grafos G e H são isomorfos, basta examinar todas as bijeções de $V(G)$ em $V(H)$. Se cada um dos grafos tem n vértices, esse algoritmo consome tempo proporcional a $n!$. Como $n!$ cresce explosivamente com n, esse algoritmo é decididamente insatisfatório na prática. Infelizmente, não se conhece um algoritmo substancialmente melhor.

3.6 Outras Noções de Grafos

Em algumas situações podemos ter um modelo para um problema a ser resolvido e esse modelo seria um grafo se desconsiderassemos algumas peculiaridades da situação. Por exemplo, um mapa rodoviário pode ser modelado definindo-se um vértice para cada cidade e duas cidades formam uma aresta no grafo (modelo) se existe rodovia ligando essas cidades correspondentes aos vértices. Normalmente,

distância é um parâmetro importante nesses mapas e assim as arestas devem ter um comprimento associado a elas.

Entretanto, "comprimento de aresta" não faz parte da definição de um grafo. Num outro exemplo, se estamos interessados em rotas de tráfego dentro de uma cidade podemos definir um vértice por esquina e duas esquinas consecutivas numa mesma rua formam uma aresta. Nesse caso, as ruas têm sentido (mão e contra-mão) e as arestas também deveriam ter mas, novamente, essa característica não faz parte da definição de grafos.

Esses problemas e muitos outros podem ser modelados com "outros tipos" de grafos. Alguns desses outros tipos são:

- **Grafo com pesos nas arestas** é um tripla (V, E, ρ) de modo que (V, E) é um grafo e $\rho : E \to R$ é uma função que assume valores em $R \subseteq \mathbb{R}$. O conjunto R em geral depende do problema sendo considerado. Por exemplo, se os pesos representam distância então tomamos $R + \mathbb{R}^+$ (reais não-negativos). O grafo (V, E) é chamado grafo subjacente do grafo com pesos nas arestas;

- **Grafo orientado** é aquele onde as arestas têm uma orientação (são pares ordenados) de modo que se u e v são vértices, então $(u, v) \neq (v, u)$ e além disso se (u, v) é uma aresta então (v, u) não é aresta. Removendo o sentido das arestas temos o grafo subjacente ao grafo orientado;

- **Grafos dirigido** ou digrafo é dado (V, E) onde $E \subseteq V \times V \setminus (v, v) \therefore v \in V$;

- **Multigrafo** é dado por um conjuntode vértices e podemos ter mais de uma aresta incidente ao mesmo par de vértices. Removendo as arestas repetidas de um multigrafo temos o *grafo subjacente* ao multigrafo.

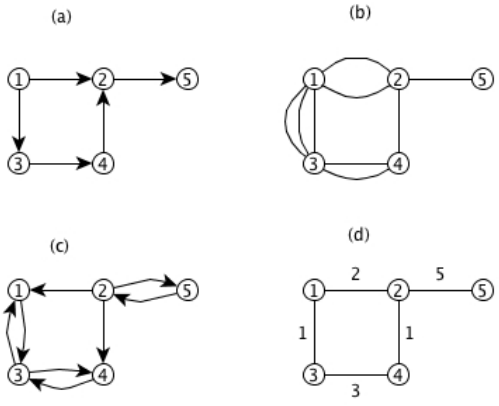

Figura 3.8: (a) Grafo orientado. (b) Multigrafo. (c) Grafo dirigido. (d) Grafo subjacente

Capítulo 4

Representações Computacionais de Grafos

Para representar os grafos estudados em um ambiente computacional devemos estudar qual estrutura devemos utilizar. Em geral, a estrutura de dados adotada está relacionada com qual a função do algoritmo que irá atuar no grafo.

Em casos gerais, com uma estrutura básica deveríamos ser capazes de responder as perguntas:

1. qual é o tamanho do conjunto de vértices de um grafo G? $|V(G)|$
2. qual é o tamanho do conjunto de arestas de um grafo G? $|E(G)|$
3. qual é o grau de um determinado vértice de um grafo G? $d_G(u) \therefore u \in V(G)$
4. uma aresta $\{u, v\}$ pertence ao conjunto de arestas do grafo G? $\{u, v\} \in E(G)$
5. qual a vizinhança de um vértice u do conjunto de vértices do grafo G? $N_G(u) \therefore u \in V(G)$

Algumas perguntas, como a 3, são *perguntas secundárias*, por exemplo o grau de um vértice está ligado com a quantidade de vizinhos desse vértice, outro exemplo de *pergunta secundária* é uma pergunta do tipo: *O grafo G é completo?*, pois também depende de outras informações obtidas por perguntas primárias.

Para representação computacional de um grafo, é necessário trechos de algoritmos que percorram os vizinhos de um determinado vértice, assim respondendo tanto perguntas primárias quanto secundárias.

Uma das perguntas mais recorrentes nos algoritmos que percorrem os nodos de um grafo é: "para todo $u \in N_g(v)$ faça". Essa pergunta pode ser traduzida de duas formas diferentes:

- "para todo $u \in V(G) \therefore \{u, v\} \in E$ faça";
- "$u \leftarrow$ primeiro vizinho (v)" + "$u \leftarrow$ próximo vizinho (v)".

O primeiro laço necessita que se responda a pergunta 4, enquanto que a segunda necessita que se responda a pergunta 5.

Uma questão interessante ao se codificar a estrutura de dados de um grafo é o método de indexação, **vértices x arestas**. Mesmo assim existem casos particulares em que algum tipo de indexação pode ser mais eficiente que outro.

4.1 Lista de Arestas

É uma representação de um grafo na qual é dado um vetor ou lista com pares de vértices representando as arestas desse grafo.

Exemplo 7: Seja um grafo G de exemplo, definido por:

$$\begin{aligned} V(G) &= \{1,2,3,4,5\} \\ E(G) &= \{\{1,2\},\{2,5\},\{5,4\},\{4,3\},\{4,2\}\} \end{aligned}$$

Sua representação por lista de arestas é dada por:

$$\{1,2\},\{2,5\},\{5,4\},\{4,3\},\{4,2\}$$

Para representar os vértices nessa notação utiliza-se um vetor de vértices de 1 a n com $n = |V(G)|$.

$$V = \{1,2,3,4,5\}$$

Com essa estrutura, responder a pergunta 4 ($\{u,v\} \in E(G)$) custa $\Theta(m)$ onde $m = |E(G)|$.

Responder a pergunta 5 ($N_G(u) \therefore u \in V(G)$), também tem o custo $\Theta(m)$ onde $m = |E(G)|$.

Isso porque nos dois casos é necessário percorrer a lista de arestas, no primeiro caso apenas verificando se u e v estão na lista $E_{i,j}$. No segundo caso, procura-se por u, ao se achar anota-se o vizinho, e continua-se da próxima aresta $E_{i,j}$.

4.2 Matriz de Adjacências

É uma matriz n x m tal que:

$$M_{i,j} = \begin{cases} 0 & \text{se} \quad i,j \notin E(G) \\ 1 & \text{se} \quad i,j \in E(G) \end{cases}$$

Essa matriz é simétrica, quadrada e possui sua diagonal preenchida por 0.

O custo para descobrir a pergunta 4 ($\{u,v\} \in E(G)$) custa $\Theta(1)$.

Responder a pergunta 5 ($N_G(u) \therefore u \in V(G)$), tem o custo $\Theta(m)$ onde $m = |V(G)|$.

Segue a representação do Exemplo 4.1 por matriz de adjacências:

$$\begin{pmatrix} \# & 1 & 2 & 3 & 4 & 5 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 1 & 1 \\ 3 & 0 & 0 & 0 & 1 & 0 \\ 4 & 0 & 1 & 1 & 0 & 1 \\ 5 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

4.3 Listas de Adjacência

É um vetor de listas indexado pelos vértices e cada lista representa uma vizinhança.

Na Figura 4.1 podemos observar uma representação por lista de adjacências do Exemplo 4.1.

Responder a pergunta 5 ($N_G(u) \therefore u \in V(G)$), tem o custo $\Theta(m)$ onde $m = |N_G(u)|$.

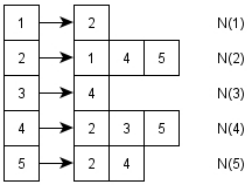

Figura 4.1: Lista de Adjacências.

4.4 Pesos

Para atribuir pesos aos vértices ou arestas, podemos utilizar:

- vértices: criar um vetor associado.
- arestas:
  1. criar um nome para cada aresta e criar também um vetor associado;
  2. na matriz de adjacências armazenar a informação na célula $M_{i,j}$;
  3. na lista de adjacências armazenar uma pequena informação em cada vizinho.

Uma boa prática é a criação de um vetor associado para armazenar os graus de um vértice.

4.5 Vantagens e Desvantagens

As estruturas ilustradas apresentam vantagens e desvantagens, o que é retornado em um melhor tempo por uma estrutura não retornado por outra, por isso deve-se analisar o algoritmo a ser escrito, para que escolha-se a estrutura mais adequada de acordo com as operações que irão ser feitas.

A matriz de adjacências, apresenta um outro problema que é o espaço desperdiçado quando se tem grafos espaços (com poucas arestas), pois a matriz irá ser preenchida por muitos espaços vazios.

Em algumas implementações, pode se tornar interessante preservar as duas estruturas (matriz de adjacências e listas de adjacências) para se obter um melhor desempenho. A desvantagem é manter a persistência em ambas estruturas, e o espaço ocupado duplicado.

Capítulo 5

Percursos em Grafos

Para ilustrar os estudos dos percursos vamos utilizar o grafo disposto na Figura 5.1

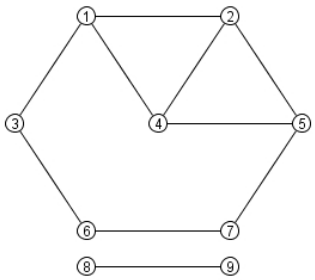

Figura 5.1: Grafo de Exemplo para estudo de Caminhos e Circuitos.

Dado um grafo $G(V, E)$ com uma aresta $v, w \in V(G)$, então podemos definir o conceito de **alcansável**:

O vértice v é alcansável pelo vértice w no grafo G se existe um conjunto de vértices denotado por um conjunto P ou seja $P(v_1, v_2, v_3, \ldots, v_n)$ tal que:

- $v_i \in V(G)$ para $i \in \{1, 2, 3, \ldots, n\}$ Os vértices a serem percorridos estão presentes no conjunto de vértices do grafo dado;
- $v_1 = v$ O primeiro vértice do conjunto se torna o vértice inicial dado, no caso v;
- $v_n = w$ O último vértice do conjunto é o vértice final dado, no caso w;
- $\{v_i, v_{i+1}\} \in E(G)$, para $i \in \{1, 2, 3, \ldots, n\}$ Existe uma aresta que liga o vértice v_i com o vértice v_{i+1} no conjunto de arestas dado.

No exemplo da Figura 5.1, o vértice 4 é alcansável pelo vértice 6. Entretanto o vértice 8 não é alcansável pelo vértice 7.

Um **passeio** em um grafo é uma sequência de arestas do tipo $\{\{v_0, v_1\}, \{v_1, v_2\}, \ldots, \{v_{s-1}, v_s\}\}$ onde s é o comprimento do passeio. Um passeio não possui um limite, é possível passear pelo grafo repetindo vértices e arestas.

No exemplo da Figura 5.1, poderíamos definir um passeio $P_1\{\{6, 3\}, \{3, 1\}, \{1, 4\}, \{4, 1\}\}$.

Um **caminho** em um grafo é um passeio que não possui repetição de vértices. Sua definição formal é dada por:

$$v_i \neq v_j \text{ para } i \neq j$$

O limite de um caminho se refere a situação de estar em determinado vértice v e não poder inserir outro vizinho v por todos já estarem contidos no caminho. Um caminho normalmente visa atingir um determinado objetivo, ou seja ir de um vértice para outro.

No exemplo da Figura 5.1, poderíamos definir um caminho $C_1\{6, 7, 8, 4, 2\}$.

Uma **trilha** em um grafo é um passeio que não possui repetição de arestas. Sua definição formal é dada por:

$$\{v_i, v_{i+1}\} \neq \{v_j, v_{j+1}\} \text{ para } i \neq j$$

No exemplo da Figura 5.1, poderíamos definir uma trilha $C_1\{\{6, 3\}, \{3, 1\}, \{1, 4\}, \{4, 2\}\}$. Note que o vértice 4 foi repetido, entretanto para arestas diferentes.

A *diferença* entre trilha e caminho é que a trilha permite repetição de vértices, desde que não se repitam arestas.

Uma trilha pode ser imaginada como um desenho marcado em determinado grafo, que indica um caminho marcado.

Um **circuito ou ciclo** é um caminho que começa e acaba com o mesmo vértice. Ciclos de comprimento 1 são laços. No grafo de exemplo, $\{1, 4, 5, 2, 1\}$ é um ciclo de comprimento 4 (o comprimento de um ciclo é dado pela contagem de arestas). Sua definição formal é dada por:

$$\begin{aligned} v_1 &\neq v_j, \text{ para } i \neq j \\ v_1 &= v_n \end{aligned}$$

A **distância** de um vértice u a um vértice w é o número de arestas de um caminho mínimo de u a w. Sua definição formal é dada por:

$dist_G(u, w) = min\{k\}$, onde u e w são extremos de um caminho de tamanho k.

Observação Não faz sentido dizer "*a menor distância*" ou "*a distância mínima*": distância já é mínima por definição. Portanto, dizer que a distância de u a w é k significa duas coisas:

1. existe um caminho de u a w com exatamente k arestas;
2. não existe caminho de u a w com menos que k arestas.

Caso u não seja alcansável por w então podemos dizer que a distância $dist_G(u, w) = \infty$.

No exemplo da Figura 5.1, $dist_G(7, 8) = \infty$ e $dist_G(6, 1) = 2$.

5.1 Algoritmo Genérico de Busca

Nessa sessão vamos estudar um algoritmo genérico para percorrer um grafo dado. Para isso iremos tomar como exemplo base o seguinte problema:

Dado um grafo G e um vértice $s \in E(G)$, encontrar $S \subseteq V(G)$ tal que para todo $v \in S$ existe um caminho de s a v.

Com um grafo e um vértice qualquer, devemos descobrir todos os vértices alcansáveis para o vértice passado.

- visite(G,s);
- dado: um grafo G e um vértice $s \in V(G)$;
- devole: um conjunto S de vértices alcansáveis a partir de s;

O algoritmo visite(G,s) está descrito no Algoritmo 2.

Algoritmo 2: Algoritmo de Busca ou Percurso

```
1  visite(G, s);
2  início
3      L[] ← conjunto de vértices vazio;
4      L ← s;
5      marque s como visistado e V(G) ∖ s como não-visistado;
6      marque as arestas de E(G) como não-visistads;
7      enquanto L ≠ ∅ faça
8          escolha v pertencente a L;
9          se em E_G(v) existe uma aresta não visistada a partir de v então
10             escolha {v, w} em E_G(v) não visistada a partir de v;
11             marque {v, w} como visitada a partir de v;
12             se w é não-visistado então
13                 insista w em L;
14                 marque w como visistado;
15         senão
16             remova v de L;
```

5.1.1 Custo do Algoritmo

O número de vezes que um vértice u é escolhido como v é $d_G(u) + 1$. Isso por que ele é escolhido 1 vez para visistar cada aresta incidente a u e uma vez para ser eliminado do conjunto L.

Como cada vértice entra no máximo 1 vez em L, o número de vezes que um vértice é escolhido, é no máximo $\sum_{v \in S} |d_G(v) + 1|$.

$$\begin{aligned} \sum_{v \in S} |d_G(v) + 1| &= \sum_{v \in S} |d_G(v)| + \sum_{v \in S} |1| \\ &= 2|E(G[S])| + |S| \leqslant 2m + n \end{aligned}$$

A complexidade de tempo do algoritmo visite(G,s) é $\Theta(2|E(G[S])| + |S|)$ ou $\Theta(m + n)$, como m sendo o número de arestas e n sendo o número de vértices.

5.2 Tipo de Busca

A forma como a estrutura de dados será implementada indica o tipo de busca que será efetuada, se a estrutura que armazena L for implementada em uma pilha, então a busca se dará em profundidade:

- o algoritmo começa pelo nó raiz e explora tanto quanto possível cada um dos seus ramos, antes de retroceder (*backtracking*);

caso a estrutura que armazena L seja implementada como uma fila, então a busca se dará em largura:

- o algoritmo começa pelo nó raiz e explora todos os nós vizinhos; então, para cada um desses nós mais próximos, explora-se os seus nós vizinhos inexplorados e assim por diante.

5.2.1 Busca em Profundidade

Esse tipo de busca usar um percurso utilizando uma pilha como estrutura de dados.
O algoritmo para uma busca em profundidade é recursivo, como pode ser visto no ALgoritmo 3.

- bp(G,w);
- dado: um grafo G e um vértice $w \in V(G)$;
- devole: uma busca em profundidade em G a partir de w com rótulos *chega*, *sai* e *pai* em cada vértice;

Os rótulos darão origem a uma tabela que representará qual o percurso que o algoritmo tomou a partir de uma marcação *temporal* (baseada no estado dos elmentos).

Algoritmo 3: Algoritmo de Busca em Profundidade

```
1  bp(G, w);
2  início
3      chega[] ← ∅;
4      sai[] ← ∅;
5      pai[] ← ∅;
6      cont ← ∅;
7      cont ← cont + 1;
8      chega[w] ← cont;
9      para cada u, N_G(w) faça
10         se chega[w] = ∅ então
11             pai[u] ← w;
12             bp(G, u);
13     cont ← cont + 1;
14     sai[w] ← cont;
```

A partir do Algoritmo 3 seja $\{w, v\} \in E(G)$:

1. cenário 1: $sai[w] < chega[v]$
2. cenário 2: $chega[w] < chega[v]$ e $sai[v] < sai[w]$
3. cenário 3: $chega[u] < chega[w]$ e $sai[w] < sai[u]$
4. cenário 4: $chega[w] < chega[v]$ e $sai[w] < sai[u]$

Os cenários representam a ordem que w ou v são executados. O primeiro cenário diz que w é executado, termina sua execução e depois u é executado. Isso se torna impossível pois quando o algoritmo está executando w encontra seu vizinho u e chama o algoritmo recursivo para u. Logo w só terminará a sua execução depois de u.

O segundo cenário é o padrão de comportamento do algoritmo, um vértice w invoca a recursividade para seu vizinho u, u termina sua execução e e devolve a vez para w que por sua vez também termina a sua execução.

O terceiro cenário só irá acontecer quando um vértice v estiver procurando por seus vizinhos e identificar que w é seu antecessor.

O quarto cenário é impossível pela explicação dada para o primeiro cenário.

Exemplo 8: Seja um grafo G_1 representado pela Figura 5.2

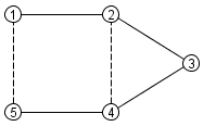

Figura 5.2: Grafo G_1 de Exemplo para estudo de Busca em Profundidade.

A busca em profundidade aplicada ao algoritmo passando bp(G_1,1), retorna a Tabela 5.1:

#	chega	sai	pai
1	1	10	-
2	2	9	1
3	3	8	2
4	4	7	3
5	5	6	4

Tabela 5.1: Tabela de Chega, Sai e Pai para a execução do algoritmo bp(G_1,1).

As arestas $\{1,5\}$ e $\{2,4\}$ não são visitadas de forma explícita pelo algoritmo, então essas arestas são chamadas de **arestas de retorno**, pois relacionam um vértice com um ancestral que não é o seu pai.

A Figura 5.3 mostra uma outra representação do grafo levando em consideração as informações temporais.

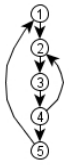

Figura 5.3: Grafo G_1 Ordenado Temporalmente

Uma forma de conseguir atingir as arestas de retorno é a inserção de um "SENÃO" entre as linhas 13 e 14 do Código **??**.

As arestas que formam ligações diretas com seus ancestrais são chamadas de arestas de árvore. Por exemplo a aresta $\{1,2\}$.

Exemplo 9: Seja um grafo G_2 representado pela Figura 5.4

A Figura 5.5 mostra uma outra representação do grafo levando em consideração as informações temporais.

A aresta $\{1,5\}$ são **arestas de retorno**, pois relaciona um vértice com um ancestral que não é o seu pai.

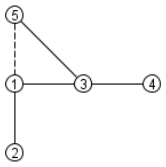

Figura 5.4: Grafo G_2 de Exemplo para estudo de Busca em Profundidade.

#	chega	sai	pai
1	1	10	-
2	2	3	1
3	4	9	1
4	5	6	3
5	7	8	3

Tabela 5.2: Tabela de Chega, Sai e Pai para a execução do algoritmo bp(G_2,1).

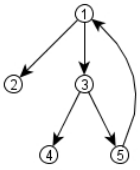

Figura 5.5: Grafo G_1 Ordenado Temporalmente

5.2.2 Busca em Largura

Para implementar a busca em largura basta trocar a escolha de um vértice no algoritmo base (Código **??**) por uma fila.

A busca em largura acontece em camadas (níveis) no grafo.

Exemplo 10: A Figura 5.6 mostra o mesmo grafo da busca em profundidade, entretanto a construção da árvore de busca é dada de forma diferente.

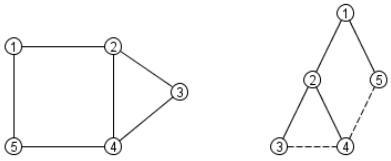

Figura 5.6: Exemplo de Busca em Largura.

As arestas pontilhadas são chamadas de **arestas de cruzamento**, e as demais arestas são chamadas de **arestas de árvore**.

Observações:

- ao entrar na fila, um vértice tem o nível no máximo 1 a mais que o primeiro da fila;

- o nível dos vértices da fila é crescente.

Conclusão:

- os vértices da fila são de no máximo 2 níveis;
- qualquer aresta do primeiro da fila para qualquer vértice na fila, liga vértices do mesmo nível ou 1 nível abaixo.

Teorema: Em uma busca em largura o nível em que se encontra um vértice é a distância para o vértice inicial?

5.2.3 Algoritmos para Identificar Aresta de Ávore ou de Não-árvore

Os algoritmos que identificam as arestas de árvore e de não árvore a partir de um grafo e um vértice inicial, estão dispostas nos Algoritmos 4 e 5, que representam uma busca em largura e em profundidade respectivamente.

Algoritmo 4: Identificar arestas para busca em largura

```
1  bfs(G, u);
2  início
3      lista_arestas_visitadas[] ← ∅;
4      lista_vertices_visitados[] ← ∅;
5      lista_arestas_arvore[] ← ∅;
6      lista_arestas_nao_arvore[] ← ∅;
7      lista_auxiliar[] ← ∅;
8      lista_vizinhos[] ← ∅;
9      adicione v em lista_vertices_visitados;
10     adicione v em lista_auxiliar;
11     enquanto lista_auxiliar ≠ ∅ faça
12         u = último elemento da lista_auxiliar;
13         remove o último elemento da lista_auxiliar;
14         lista_vizinhos ← n_g(u);
15         enquanto lista_vizinhos ≠ ∅ faça
16             w ← n_g(u);
17             se na lista_arestas_visitadas ∉ {u, v} então
18                 adicione u, w em lista_arestas_visistadas;
19                 se na lista_vertices_visitados ∉ w então
20                     adicione w na lista_auxiliar;
21                     adicione w na lista_vertices_visitados;
22                     adicione {u, w} lista_arestas_nao_arvore;
23                 senão
24                     adicione {u, w} lista_arestas_arvore;
```

5.2.4 Aplicações

Dependendo do algoritmo que vá percorrer o grafo, os dois tipos de busca são eficientes e têm o mesmo custo computacional, entretanto algumas particularidades de cada busca pode ser favorável para:

Algoritmo 5: Identificar arestas para busca em profundidade

```
1  dfs(G, u);
2  início
3      lista_arestas_visitadas[] ← ∅;
4      lista_vertices_visitados[] ← ∅;
5      lista_arestas_arvore[] ← ∅;
6      lista_arestas_nao_arvore[] ← ∅;
7      lista_auxiliar[] ← ∅;
8      lista_vizinhos[] ← ∅;
9      enquanto lista_auxiliar ≠ ∅ faça
10         w é o vértice atual;
11         a é a aresta {v, w};
12         se na lista_arestas_visitadas ∉ a então
13             adicione a na lista_arestas_visitadas;
14             se na lista_vertices_visitados ∉ w então
15                 adicione w na lista_vertices_visitados;
16                 adicione a lista_arestas_arvore;
17             senão
18                 adicione a lista_arestas_nao_arvore;
```

- encontrar ciclos: utiliza-se busca em profundidade;
- caminho mínimo: utiliza-se busca em largura.

Capítulo 6

Conexidade

A conexidade diz respeito a quão *conexo* é um grafo.

Um **grafo conexo** ou simplesmente **conexo** ou ainda **1-conexo** é um grafo que não contém partes separadas, ou formalmente: G é conexo se é não vazio e quaisquer 2 vértices de G são extremos de um caminho em G.

Um grafo vazio por definição não é conexo, e um grafo de 1 vértice apenas, por definição é conexo.

6.1 Componente Conexa

Uma componente conexa é um subgrafo maximal conexo de G. Um subgrafo é maximal se não existe outro subgrafo maior que ele que tenha a mesma propriedade (no caso conexidade).

Uma definição formal para maximal é:

X é maximal em uma propriedade P, se X tem a propriedade P e não pertence a Y tal que $X \subset Y$ e Y é P.

Para ilustrar o conceito de componente conexa vamos utilizar o grafo G da Figura 6.1.

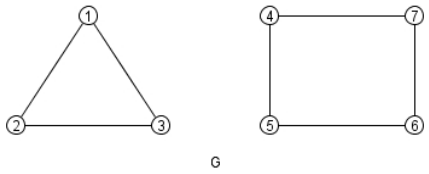

Figura 6.1: Exemplo de componente conexa.

A componente conexa do grafo G é dada pela indução $G[S] \therefore S = \{4, 5, 6, 7\}$. Pois essa é a maior componente conexa possível no grafo G. Se pegássemos um fragmento $S' = \{4, 5\}$ poderíamos dizer que $G[S']$ é conexo entretanto não é uma componente conexa pois existe uma outra indução que também é conexa e maior que ela. Por esse motivo o formato triangular não pode ser considerado componente conexa, pois existe um quadrado que possui mais vértices que o triângulo.

6.2 Relação Binária

Seja $\sim_G$ a relação binária entre dois vértices u e v definida como $u \sim_G v$ se e somente se:

- existe um caminho em G com extremos u e v.

A relação $\sim_G$ é *reflexiva*, *simétrica* e *transitiva* o que define uma relação de equivalência.

6.2.1 Fecho Transitivo

O **fecho transitivo direto** de um vértice v é o conjunto de todos os vértices que podem ser alcansáveis por algum caminho iniciando em v.

Seja o grafo G da Figura 6.2, então:

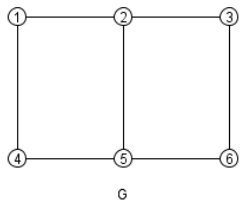

Figura 6.2: Exemplo fecho transitivo.

- O **fecho transitivo** do vértice 5 do grafo G, por exemplo, é o conjunto: $\{1, 2, 3, 4, 5, 6\}$. Note que o próprio vértice faz parte do fecho transitivo já que ele é alcançável partindo-se dele mesmo.

6.3 Definições

1. Se G é um grafo e e é uma aresta de G, então defina $G \setminus e = (V(G), E(G) \setminus \{e\})$.

2. Se G é um grafo e v é um vértice de G, então defina $G \setminus v = (V(G) \setminus \{v\}, E(G) \setminus E_g(v))$.

6.4 Teorema do Ciclo

Se $G = (V, E)$ é conexo então $G \setminus e$ é conexo para toda aresta e que faz parte de um ciclo.

Para provar o teorema vamos ilustrar o teorema com o grafo G da Figura 6.3

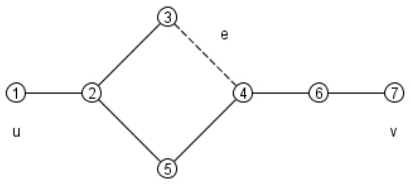

Figura 6.3: Teorema: grafo conexo remoção de aresta em ciclos.

É fácil perceber que se removermos qualquer uma das arestas que está no ciclo formado pelos vértices $\{2, 3, 4, 5\}$ ainda poderemos obter um outro caminho de u a v.

Prova: u e $v \in V(G)$. Em G existe um caminho de u a v, esse caminho é dado por $C = (u = u_1, u_2, \ldots, u_k = v)$.

Podemos deduzir duas situações:

1. Se e não faz parte de C então C também é caminho de u a v em $G \setminus e$. Ou seja, se a aresta retirada não faz parte do meu caminho, não terá influência nesse caminho.
2. Se e faz parte de C, então:
   - Define-se $e = \{x, y\}$;
   - Defini-se $C = (u = u_1, u_2, \ldots, u_i = x, u_{i+1} = y, \ldots, u_k = v)$;
   - Seja $C_1 = (u = u_1, u_2, \ldots, u_i)$;
   - Seja $C_2 = (u_{i+1}, u_{i+2}, \ldots, u_k = v)$;

Como e está em um ciclo $L = (x, w_1, w_2, \ldots, w_n = y, x)$, então existe um caminho P de x a y que não contém a aresta removida.

Seja $T = (u = u_1, u_2, \ldots, u_i = x, w_1, \ldots, w_n = y, u_{i+2}, \ldots, w_k = v)$ um passeio formado por $T = (C_1 + P + C_2)$.

Então existe um caminho entre dois vértices u e v em um passeio que contenha os vértices u e v.

Defini-se C' o caminho que é a simplificação de T, então temos C' um caminho de u a v em $G \setminus e$.

6.5 Articulação

Um vértice v de um grafo conexo G é uma **articulação** (ou vértice de corte) se $G \setminus v$ é não conexo.

O grafo G da Figura 6.4 ilustra um vértice de articulação:

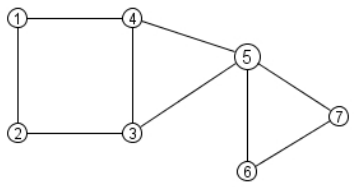

Figura 6.4: Vértice de articulação.

O vértice 5 é um vértice de articulação, pois se o vértice 5 for tirado do grafo o resultado será um grafo não conexo.

6.6 Conexidade

A **conexidade**, é o número de vértices necessários para desconectar um grafo. O grafo da Figura 6.4 é 1-conexo, pois se removermos apenas um vértice (no caso o vértice 5) desconectamos o grafo.

Se $U \subseteq V(G)$ então $G \setminus U = G[V(G) \setminus U]$.

Para todo $K \in \mathbb{N}$ dizemos que G é K-conexo:

- Se $|V(G)|[\geqslant, >]K$;
- Para todo $U \subset V(G)$, se $|U| < K$ então $G \setminus U$ é conexo;

A conexidade de um grafo G é um número $K(G)$ tal que $K(G) = \max\{K \in \mathbb{N} \therefore G \text{ é conexo}\}$.

Observação: Pode-se definir também os mesmos conceitos de conexidade para arestas.

6.7 Grafos Eulerianos

Um grafo *euleriano* é um grafo que pode ser desenhado sem retirar o lápis do papel, dessa forma, uma aresta nunca será repetida.

Definição Formal: Um grafo G é euleriano se em G existe um passeio fechado (começa e termina no mesmo lugar) sem repetições de arestas (trilha fechada) que passa por todas as arestas de G e é conexo.

Uma *trilha fechada* é um ciclo (circuito).

O grafo da Figura 6.5 é um grafo euleriano.

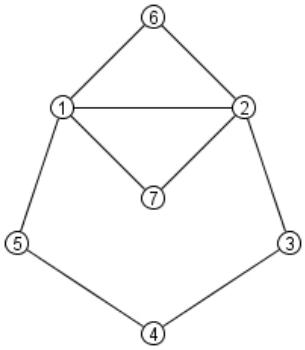

Figura 6.5: Exemplo de Grafo Euleriano.

6.7.1 Teorema de Euler

Um grafo G é *euleriano* se e somente se G é conexo e todo vértice tem grau par:

Prova: Ser euleriano implica em ser conexo e ter grau par ($\rightarrow$).

Se G é euleriano, logo ele também é conexo.

Podemos assumir que existe uma trilha fechada que passa por todas a arestas de G.

Seja v um vértice de G.

Se v aparece na trilha k vezes, então o grau de v é $2k$. Pois existe arestas que chegam e arestas que saem de v.

G é conexo e todos os vértices têm grau par ($\leftarrow$).

Seja $T = (v_0, v_1, v_2, \ldots, v_l)$ uma trilha com o maior número possível de arestas em G. T passa por todas as arestas com extremos v_0 e v_1.

Se $v_0 \neq v_1$ o grau de v_0 deveria ser ímpar.

Como v_0 deve ter grau par, então v_0 deve ser igual a v_l.

Logo, T é fechado.

Suponha que existe uma aresta e que não está em T.

Pode-se assumir que $e = \{v_i, x\}$, já que G é conexo.

Pode-se re-escrever T para que v_i seja o primeiro vértice, portanto a trilha $T = (x, v_i, v_{i+1}, \ldots, v_l = v_0, v_1, v_2, \ldots, v_k)$ tem uma aresta a mais que T.

Logo, não existe aresta fora de T.

Logo, T é fechada e passa por todas as arestas.

Portanto G é euleriano.

6.8 Grafos Hamiltorianos

G é *hamiltoriano* se em G existe um ciclo (circuito) que passa por todos os vértices.

6.8.1 Teorema de Dirac

Para todo grafo G com 3 ou mais vértices se $\delta(G) \geqslant \frac{n}{2}$ então G é hamiltoriano.

Prova: Seja $P = (v_0, v_1, \ldots, v_l)$ o maior caminho em G e defina os conjuntos:

- $A = \{v_i em P \therefore \{v_0, v_{i+1} \in E(G)\}$
- $B = \{v_j em P \therefore \{v_l, v_j \in E(G)\}$

A e B são subconjuntos de $\{v_0, v_i, \ldots, v_{l-1}\}$.

$|A| \geqslant \frac{n}{2}$ e $|B| \geqslant \frac{n}{2}$.

$A \cap B \neq \emptyset$.

Seja $P = \{v_0, \ldots, v_k, v_{k+1}, \ldots, v_l\}$ se retirarmos a aresta $\{v_k, v_{k+1}\}$ e inserirmos uma equivalente u, então podemos re-escrever P, e assim ele não é máximo.

Observação: O problema do caixeiro viajante tem uma relação com os grafos hamiltorianos, pois passar por todas as cidades sem repetições, é também uma característica da propriedade estudada.

6.9 Caminhos Mínimos em Grafos com Pesos nas Arestas

Nessa seção iremos estudar grafos que possuem pesos em suas arestas, ou seja, o fator que deixa um caminho maior ou menos não está apenas na contagem de arestas, mas no somatórios dos pesos de todas as arestas que fazem parte desse caminho.

Uma *definição formal* diz que um grafo com pesos nas arestas é uma terna (V, E, ω) onde V e E são os usuais conjuntos de vértices e arestas, respectivamente, e $\omega : E(G) \to \mathbb{R}^+$ é uma função que atribui peso a cada aresta de $E(G)$.

Por exemplo o grafo G da Figura 6.6.

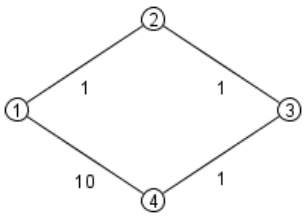

Figura 6.6: Exemplo de Grafo com Pesos nas Arestas.

O *comprimento de um caminho* $C = (x_0, x_1, \ldots, x_k)$ em (V, E, ω) é a soma dos pesos (comprimentos) das arestas do caminho, ou seja:

$$\text{custo}(C) = \sum_{i=0}^{k-1} p(\{x_i, x_{i+1}\})$$

E a *distância* entre dois vértices u e v é dada por:

$$\text{dist}(u, v) = min\{\text{custo}(C)\}$$

Sendo que C é um caminho com extremos em u e v.

Se não existe um caminho de u a v então assume-se que $\text{dist}(u, v) = \infty$.

6.9.1 Algoritmo de Dijkstra para Caminhos Mínimos

Suponha que é dado um grafo com pesos nas arestas do tipo $G = (V, E, \omega)$ e um vértice $s \in V(G)$. Pede-se para determinar a distância de s a todos os vértices de G, ou formalmente: $\text{dist}(u, v) \therefore v \in V(G)$.

As estruturas de dados que serão utilizadas são:

- $P[v]$: Predecessor de v em um caminho *candidato* a ser mínimo de s a v;
- $D[v]$: Custo do caminho *candidato* a ser mínimo;
- S: Conjunto dos vértices para os quais $D[v]$ é equivalente a $\text{dist}(s, v)$.

Inicialmente, as estruturas estão:

- $P[v] = \lambda$ para todo $v \in V(G)$;
- $D[v] = \emptyset$ para todo $v \in V(G) \setminus \{s\}$;
- $S = \emptyset$;

Uma etapa importante chamada de relaxação é definida pelo algoritmo do Algoritmo 6.

Algoritmo 6: Relaxação(u, t)

```
1 Relaxacao(u, t);
2 início
3     se d[t] > d[u] + ω({u, t}) então
4         dt[t] ← d[u] + ω({u, t});
```

Essa função de *Relaxação*(u, v) compara se a distância de determinado vértice a outro é menor, se for atribui a ele a nova distância.

O algoritmo completo de Dijkstra está representado pelo Algoritmo 7.

O algoritmo começa com um conjunto S vazio. A cada iteração do algoritmo, busca-se em uma lista de prioridades o vértice que possui a menor distância. Por esse motivo o primeiro passo é adicionar u ao conjunto S (que tem a distância inicial 0, pois $\text{dist}(u, u) = 0$). A distância para qualquer outro $v \in V(G) = \infty$. Um conjunto $\overline{S}$ contém os demais vértices de $V(G)$. O próximo passo é listar os vizinhos v de u e executar a subrotina de *relaxação* para garantir que se houver um caminho mais curto, esse será o novo $D[v]$, permitindo assim que em uma próxima iteração esse seja o menor na lista de prioridades e seja escolhido.

O custo do algoritmo de Dijkstra para obtenção do caminho mínimo é $\Theta((n + m) \log_n)$.

Algoritmo 7: Dijkstra(G, u)

```
1  início
2      D[u] ← ∞;
3      P[u] ← λ;
4      D[s] ← ∅;
5      S ← {};
6      enquanto existir u ∈ S̄ tal que D[u] ≠ ∞ faça
7          seja u tal que d[u] = min{d[v] ∴ v ∈ S̄};
8          para cada t ∈ N_G(u) faça
9              Relaxação(u, t);
10         S ← S ∪ {u};
```

6.9.2 Algoritmo de Floyd-Warshall para caminhos mínimos

Seja $G = (V, E, \omega)$ um grafo com pesos nas arestas. Apresenta-se um algoritmo para resolver o seguinte problema: dado $G = (V, E, \omega)$ computar $dist_G(i, j)$ para todos $i, j \in V$.

Uma alternativa para realizar essa tarefa é utilizar n vezes com $n = |V(G)|$ o algoritmo de Dijkstra.

Seja a representação de G dada pela matriz:

$$a_{i,j} = \begin{cases} 0 & \text{se} \quad i = j \\ \omega(\{i,j\}) & \text{se} \quad \{i,j\} \in E(G) \\ \infty & \text{caso contrário.} \end{cases}$$

Definições:

1. Para todo $k \in \{0, 1, \ldots, |V|\}$ denotamos por $[k]$ o subconjunto $\{1, 2, \ldots, k\} \subseteq V$, com $[0] = \emptyset$;
2. Dizemos que o caminho $P = (i, v_0, \ldots, v_t, j)$ é um (k)-caminho se o seus vértices internos $\{v_0, \ldots, v_t\}$ pertencem a $[k]$;
3. Uma (k)-distância entre i e j é o comprimento do menor (k)-caminho com extremos em i e j. Denotada por $dist_k(i, j)$, com $dist_0(i, j) = a_{i,j}$.

Exemplo 11: Seja o grafo G representado pela Figura 6.7.

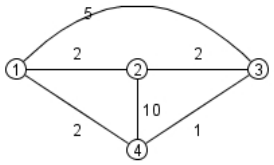

Figura 6.7: Exemplo dos conceitos para algoritmo de Floyd-Warshall.

Para entender melhor vamos analisar o grafo G.

- o único (0)-caminho com extremos 3 e 4 é $\{3, 4\}$;
- os (1)-caminhos com extremos 3 e 4 são $\{3, 4\}$ e $\{3, 1, 4\}$;
- os (2)-caminhos com extremos 3 e 4 são $\{3, 4\}$ e $\{3, 1, 4\}$;

- os (3)-caminhos com extremos 3 e 4 são $\{3, 2, 4\}, \{3, 1, 4\}$;
- os (4) caminhos com extremos 3 e 4 são $\{3, 1, 2, 4\}$, $\{3, 2, 1, 4\}$.

No grafo G temos $dist_0(3, 4) = 5$, $dist_1(3, 4) = 4$ e $dist_2(3, 4) = dist_3(3, 4) = dist_4(3, 4) = 3$. E ainda, $dist_0(1, 2) = dist_1(1, 2) = dist_2(1, 2) = 10$, $dist_3(1, 2) = 4$ e $dist_4(1, 2) = 3$.

6.9.3 Algoritmo de Floyd-Warshall

A idéia principal do algoritmo é que podemos obter $dist_{k+1}$ a partir de $dist_k$ da seguinte maneira: por definição, a $dist[k+1]-dist$ é o comprimento do menor $[k+1]$-caminho e a $[k]$-distância é o comprimento do menor $[k]$-caminho; logo, a diferença entre esses caminhos é que no primeiro, os $[k + 1]$-caminhos, é possível que o vértice $k + 1$ seja o vértice interno e no segundo não, pois $[k + 1] = [k] \cup \{k + 1\}$.

Assim temos:

$$dist_{k+1}(i, j) = min\{dist_k(i, j), (dist_k(i, k + 1) + dist_k(k + 1, j))\}$$

Escrevendo o algoritmo, obtemos o Algoritmo 8.

Algoritmo 8: Floyd-Warshall(G)

Entrada: um grafo G com peso ω nas arestas.
Saída: $dist_G i, j$ para todos $(i, j) \in V(G)$.

```
1 início
2     para cada (i, j) ∈ V(G)² faça
3         dist_0(i, j) ← a_{i,j};
4         para cada k de 1 até |V(G)| faça
5             para cada (i, j) ∈ V(G)² faça
6                 dist_k(i, j) ← min{dist_{k−1}(i, j), (dist_{k−1}(i, k) + dist_{k−1}(k, j))};
```

Capítulo 7

Árvores

As árvores estudadas em grafos podem ser consideradas **árvores genéricas**. Nesse tipo de árvore pouco importa quem é o seu nodo raiz, pois a qualquer momento pode-se eleger um nodo e tornalo raiz. Esse recurso é disponível pela estrutura com que o grafo está disposo.

Uma árvore é um grafo que possui apenas o número suficiente de arestas para que esse grafo não seja desconexo, e ao mesmo tempo não contenha ciclos.

Para introduzir os conceitos de árvores, vamos primeiro definir uma **floresta**.

7.1 Floresta

Uma **floresta** é um grafo cujas componentes conexas não contém circuito (ciclo).

7.2 Árvores

Defini-se como **árvore**, os componentes conexos de uma floresta, ou seja, uma árvore é um grafo conexo e sem circuitos.

7.3 Folha

Uma **folha** é um vértice de grau um na árvore.

Exemplo 12: O Grafo G da Figura 7.1, é uma floresta, e cada uma de suas componentes conexas são árvores. As folhas dessa floresta são: $2, 3, 8, 7, 5, 10, 13, 12$.

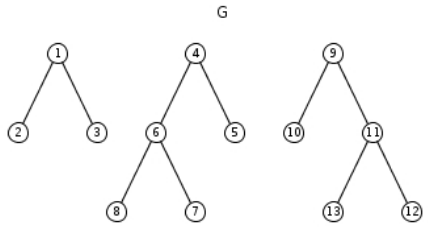

Figura 7.1: Exemplo de Floresta e Árvores.

7.4 Árvore Geradora

Uma árvore geradora de um grafo G é um subgrafo gerador T de G tal que T é gerador e é uma árvore.

7.4.1 Teorema ($n-1$)

Provar que uma árvore possui o número de arestas igual ao número de vértices menos um. Ou formalmente $|E(G) = |V(G)| - 1$. Defini-se $n = |V(G)|$.

Prova: Por definição uma árvore G possui as propriedades:

- G é acíclico, o que implica em $|E(G)| \leqslant n - 1$
- G é conexo, o que implica em $|E(G)| \geqslant n - 1$

Como G possui as duas propriedades, logo:

$$\begin{aligned} |E(G)| &\leqslant n-1 \\ |E(G)| &\geqslant n-1 \\ |E(G)| &= n-1 \end{aligned}$$

7.4.2 Teorema da Folha

Toda árvore com mais de um vértice tem pelo menos uma folha.

Prova: Seja uma árvore G e suponha que $\delta(G) \geqslant 2$.

Todo grafo G contém um caminho de comprimento pelo menos $\delta(G)$ e se $\delta(G) \geqslant 2$ esse grafo também possui um ciclo de tamanho $\delta(G) + 1$.

Logo, G contém ciclos e não é uma árvore.

Colorário da Folha

Todo grafo acíclico com pelo menos uma aresta tem pelo menos um vértice de grau 1.

Prova: Seja G um grafo acíclico com pelo menos 1 aresta, tome uma componente conexa H de G que contenha pelo menos uma aresta.

H é acíclica, pois é subgrafo de G. H contém pelo menos dois vértices, já que possui arestas.

Logo, H é uma árvore e contém folhas.

Lema da Folha

Todo grafo conexo e com $m = n - 1$ tal que $n > 1$, tem um vértice de grau 1. Sendo que m é $|E(G)$ e n é $|V(G)|$.

Prova: Seja G um grafo conexo com $m = n - 1$.

Sabe-se que $\sum_{v \in V(G)} d_G(v) = 2|E(G)|$. Logo podemos deduzir que: $\sum_{v \in V(G)} d_G(v) = 2n - 2$.

$$\sum_{v \in V(G)} d_G(v) = \sum_{v \in V(G)} (d_G(v) - \delta(G)) + n\delta(G) = 2n - 2$$
$$\sum_{v \in V(G)} (d_G(v) - \delta(G)) = n(2 - \delta(G)) - 2$$
$$\sum_{v \in V(G)} (d_G(v) - \delta(G)) \geqslant 0, \text{ pois } d_G(v) \geqslant 0$$

Logo, $n(2 - \delta(G)) - 2 > 0$. Para isso: $\delta(G) < 2$.

Ao mesmo tempo, como G é conexo, temos que $\delta(G) \geqslant 1$, logo $\delta(G) = 1$.

7.4.3 Teorema da Prova dos 3

Seja G um grafo com n vértices e m arestas, se G tem duas das seguintes propriedades, então também tem a terceira:

1. $m = n - 1$;
2. G é conexo;
3. G é acíclico.

Provar que:

I) 1 e 2 $\rightarrow$ 3;

Base: Seja o grafo G trivial ($n = 1$). Obviamente é acíclico.

Hipótese de Indução: Suponha que todo grafo conexo com $n = k - 1$ e $m = k - 2$ seja acíclico.

Passo: Seja G um grafo conexo com $n = k$. Tome uma folha $v \in V(G)$. Considere o grafo $G' = G \setminus v$. Assim temos que:

$$\begin{aligned} |V(G')| &= |V(G)| - 1 = k - 1 \\ |E(G')| &= |E(G)| - d_G(v) = k - 1 - 1 = k - 2 \end{aligned}$$

Como chega-se a $k - 1 = n$ e $k - 2 = m$, descrito pela *Hipótese de Indução*, podemos afirmar que:

G é conexo e $d_G(v) = 1$ logo, G' é conexo.

Pela *Hipótese de Indução* G' é acíclico.

Como $d_G(v) = G$, G também é acíclico.

II) 1 e 3 $\rightarrow$ 2;

Base: Seja o grafo G trivial ($n = 1$). Obviamente é conexo.

Hipótese de Indução: Suponha que todo grafo acíclico com $n = k - 1$ e $m = k - 2$ seja conexo.

Passo: Seja G um grafo acíclico com $n = k$. Tome uma folha $v \in V(G)$. Considere o grafo $G' = G \setminus v$. Assim temos que:

$$\begin{aligned} |V(G')| &= |V(G)| - 1 = k - 1 \\ |E(G')| &= |E(G)| - d_G(v) = k - 1 - 1 = k - 2 \end{aligned}$$

Como chega-se a $k - 1 = n$ e $k - 2 = m$, descrito pela *Hipótese de Indução*, podemos afirmar que:

G é acíclico logo, G' é acíclico.

Pela *Hipótese de Indução* G' é conexo.

Logo, G também é conexo.

III) 2 e 3 $\rightarrow$ 1.

Base: Seja o grafo G trivial ($n = 1$). Obviamente $m = 0$ e $m = n - 1$.

Hipótese de Indução: Suponha que todo grafo acíclico e conexo (árvore) com $n = k - 1$ tenha $m = k - 2$.

Passo: Seja G uma árvore com $n = k$. Tome uma folha $v \in V(G)$. Considere o grafo $G' = G \setminus v$. Assim temos que:

G é conexo e $d_G(v) = 1 \rightarrow G'$ é conexo;

G é acíclico $\rightarrow G'$ é acíclico, pois a remoção de um vértice nunca poderá tornar um grafo acíclico.

$$\begin{aligned} |V(G')| &= |V(G)| - 1 = k - 1 \\ |E(G')| &= \big[|V(G) - 1\big] - 1 = k - 2 \end{aligned}$$

Como G' é acíclico e conexo, e a retirada de uma de suas folhas, comprova a *Hipótese de Indução* por:

$$\begin{aligned} n &= |V(G)| - 1 \\ d_G(v) &= 1 \\ |V(G')| &= n = k - 1 \\ |E(G')| &= n - d_G(v) = k - 2 \\ m &= n - 1 \end{aligned}$$

A demonstração $n = k-1$ prova a primeira *Hipótese de Indução.* e a demonstração $n-d_G(v) = k-2$ prova a segunda *Hipótese de Indução.*

Logo, pode-se inferir $m = n - 1$

7.4.4 Teorema da Prova dos 4

Antes de enunciar e provar uma caracterização define-se a operação:
G é um grafo e $x, y \in V(G)$. Denota-se:

$$G + xy = (V(G), E(G) \cup \{x, y\})$$

As seguintes afirmações são equivalentes para todo grafo $G = (V, E)$:

(1) G é árvore;

(2) para quaisquer $x, y \in V(G)$ existe um único caminho em G com extremos x e y;

(3) G é conexo minimal: G é conexo e $G \setminus e$ é desconexo, para qualquer $e \in E(G)$;

(4) G é acíclico maximal: G é acíclico e $G + xy$ contém um circuito, para quaisquer $x, y \in V(G)$ não adjacentes.

Para provar que essas afirmações são equivalentes devemos mostrar que:

$$(1) \to (2) \to (3) \to (4) \to (1)$$

Provar que $(1) \to (2)$ Como G é conexo, existe um caminho

$$P = (x = x_0, x_1, \ldots, x_n - 1, x_n = y)$$

Suponha a existência de um outro caminho

$$Q = (x = y_0, y_1, \ldots, y_m - 1, y_m = y)$$

Pode ser que esses caminhos tenham seus vértices iniciais contidos em um mesmo caminho, por isso é necessário definir bem os índices para explicitar que pelo menos um ciclo irá ser identificado.

Então, sejam os índices:

$$\begin{aligned} r &= min(i : i \geqslant 0 \text{ e } x_{i+1} \neq y_{i+1}) \\ s &= min(j : i > p \text{ e } x_j = y_l \text{ para algum } l > p) \end{aligned}$$

O índice r identifica o primeiro vértice do caminho P que seja diferente no caminho Q.

O índice s identifica o primeiro vértice maior que o vértice indexado pelo índice r, tal que exista um vértice x_j do caminho P igual a um vértice y_l do caminho Q tal que o indice l seja maior que o indice p definido.

Esses índices estão bem definidos, como os caminhos são distintos, temos: $0 \leqslant r < min\{m, n\}$ e $r < s \leqslant n + 1$.

Dessa forma conseguimos montar um circuito dado por:
$T = (x_r, x_{r+1}, \ldots, x_s = y_l, y_{l-1}, y_{l-2}, y_r)$
Assim temos uma contradição, e podemos afirmar que o caminho com extremos x, y é único.

Demonstração: Pode ser dada por:

Seja $V(G) = X \cup \overline{X} \therefore X \cap \overline{X} = \emptyset$.

Seja C_1 um caminho de um vértice v a um vértice $a \subset X$.

Seja C_2 um caminho de um vértice y a um vértice $b \subset \overline{X}$.

É possível adicionar os sucessores de de a e b ao caminho C_1 e C_2 respectivamente, até que entre a_n e b_n possua apenas um vértice c que possui as arestas $\{a_n, c\}, \{b_n, c\}$.

Dessa forma só existe um caminho de v a a_n pois todos os sucessores de a foram adicionados ao conjunto X. e também só existe um caminho de y a b_n pelo mesmo motivo com $\overline{X}$.

Como o vértice c é o único que liga a_n e b_n, logo só existe um caminho entre v e y.

Provar que $(2) \rightarrow (3)$ Seja G tal que (2) vale, então G é conexo então existe um caminho para qualquer par de vértices x e y tal que $x, y \in V(G)$.

Tome $\{x, y\} \in E(G)$ temos que

$C = (x = v_0, v_1, \ldots, v_{n-1}, v_n = y)$ é o único caminho entre x e y.

Temos que ao remover uma aresta de um caminho único torna o grafo desconexo, então:

$C \setminus e$ é desconexo, para qualquer $e \in E(G)$.

Se existe um único caminho ligando x a y, o grafo é conexo, logo podemos afirmar que $(2) \rightarrow (3)$.

Provar que $(3) \rightarrow (4)$ Suponha que G seja cíclico. Então é possível retirar qualquer aresta pertencente a um ciclo de G e $G \setminus e \therefore e \in E(G)$ continua conexo.

Logo, G é acíclico, pois não é possível retirar qualquer aresta pela definição (3).

Sejam x e y não adjacentes em G, como G é conexo, então existe um caminho $C = (x = v_0, v_1, \ldots, v_{n-1}, v_n = y)$.

Considere o grafo $G' = G + \{x, y\}$. Logo, $C' = (x = v_0, v_1, \ldots, v_{n-1}, v_n = y, x)$. Então C' é um ciclo em G'.

Logo, G é um grafo acíclico maximal.

Como só é possível adicionar a aresta entre as extremidades do caminho achado, o grafo G é acíclico e maximal.

Provar que $(4) \rightarrow (1)$ Suponha que G seja desconexo. x e y são dois vértices entre os quais não há caminho em G.

Seja $G' = G + \{x, y\}$, temos que G' é acíclico, contrariando que G pudesse ser maximal, logo, G é conexo.

Sendo conexo, o grafo G é uma árvore.

7.5 Árvores Geradoras de Custo Mínimo em Grafos com Pesos nas Aresta

Defini-se como custo de um subgrafo H de G conexo e com peso nas arestas $G = (V, E, \omega)$, onde $\omega : E \rightarrow \mathbb{R}$:

$$c(H) = \sum_{e \in E(H)} \omega(e)$$

O problema a ser atacado é a resposta para a pergunta: Qual é o menor custo do subgrafo gerador conexo de G?

Ou formalmente, $S \subseteq E(G)$ que induz a uma árvore geradora de G tal que:

$$c(G[S]) = min\{c(T) : T \text{ é árvore geradora de } G\}$$

Uma árvore geradora de G de custo mínimo também é chamada de **árvore geradora mínima** do grafo G.

Apresenta-se a seguir os algorítmos de *Jarník-Prim* e *Kruskal* para resolver o problema de determinar a árvore geradora mínima.

Esses algoritmos são gulosos, que é uma técnica de projeto de algoritmos para resolver problemas de otimização. os códigos baseam-se na escolha que parece ser a melhor no momento (ótimo local) e terminam com a solução ótima (ótimo global).

7.6 Algoritmo de Jarník-Prim para árvore geradora mínima

Seja X um conjunto de vértices tal que $X \subset V(G)$ e X não é vazio. Temos ainda $\overline{X}$ que é o complemento de X.

7.6.1 Teoremas de Prim

1. Toda árvore geradora possui pelo menos uma aresta de $E(X, \overline{X})$;

2. Uma árvore geradore mínima, possui uma das arestas de menor peso em $E(X, \overline{X})$.

Com os teoremas 1 e 2 podemos estabelecer uma relação com o corte $E(X, \overline{X})$ e as arestas de menor peso da árvore geradora mínima.

O algorítmo de *Jarník-Prim* divide o grafo em dois conjuntos, os vértices que foram visistados U, e os que não foram visistados $\overline{U}$. E a cada iteração adiciona para o conjunto U o vértice que tinha a aresta de menor peso dentre todas as arestas do corte $E(U, \overline{U})$. Esse processo é repetido até que $U = V(G)$. No final do processo $G[U]$ é uma árvore geradora mínima.

O Algorítmo 9 descreve a estratégia de Jarník-Prim.

Algoritmo 9: Jarník-Prim(G)

Entrada: um grafo G com peso ω nas arestas.
Saída: arvore geradora de custo mínimo.

```
1  início
2      escolha v ∈ V(G);
3      U ← {v};
4      S ← ∅;
5      enquanto U ≠ V(G) faça
6          escolha {u, w} ∈ E(U, Ū) de peso mínimo no corte;
7          insira {u, w} em S;
8          v ← {u, w} ∩ Ū;
9          insira v em U;
10     devolva (V, S)
```

O principal gasto computacional está relacionado a escolha das arestas de menor peso. Claramente é inviável que a cada procura para achar a aresta de menor peso, visite-se todas as arestas que incidem no vértice em questão.

Para melhorar essa implementação, propõe-se o uso de uma fila de prioridades que armazena as arestas priorizando o seu peso. Recomenda-se a utilização de uma estrutura *Heap* para melhorar a eficiência do algoritmo.

Então basta encontrar na lista de prioridades uma ocorrência do vértice em questão, e sacar essa aresta.

O Algoritimo 10 é uma implementação da fila de prioridades baseado no Algoritimo 9.

Algoritmo 10: Jarník-Prim(G)

Entrada: um grafo G com peso ω nas arestas.
Saída: arvore geradora de custo mínimo.

```
1  início
2      escolha v ∈ V(G);
3      U ← {v};
4      S ← ∅;
5      L ← lista de prioridades por pesos das arestas;
6      enquanto U ≠ V(G) faça
7          w ← o outro vértice da aresta {u, w} que seja a primeira ocorrência em L;
8          insira {u, w} em S;
9          v ← w;
10         insira v em U;
11     devolva (V, S)
```

7.6.2 Avaliação de Corretude

Teorema de Jarník-Prim: Para $k \in \mathbb{N}$, após k iterações do "enquanto" do algoritmo de *Jarník-Prim*, as arestas escolhidas induzem uma sub árvore de uma árvore geradora mínima.

Definições Para provar o teorema, vamos fazer algumas definições:

- $S_0 = \emptyset$;
- $S_1 = \{v\}$;
- $A_0 = \emptyset$;
- $A_1 = \emptyset$.

Sendo que o conjunto S é o conjunto de vértices visistados, e o conjunto A é o conjunto de arestas que formarão uma árvore geradora mínima.

A cada volta do algoritmo, escolhe-se uma aresta e do corte com peso mínimo.

Então temos:

$$\begin{aligned} e &= \{u, w\} \in E(S_i, \overline{S_i}) \\ u &\in S_i \\ w &\in \overline{S_i} \\ S_{i+1} &= S_i \cup \{w\} \\ A_{i+1} &= A \cup \{\{u, w\}\} \end{aligned}$$

Prova do Teorema de Jarník-Prim

Podemos então reescrever o Teorema 7.6.2 da seguinte maneira: (S_k, A_k) é uma subárvore de uma árvore geradora mínima.

Devemos provar que:

- (S_k, A_k) é uma árvore;
- $(S_k, A_k) \subset T_{min}$;

Sendo que T_{min} é uma árvore geradora mínima.

Para provar que (S_k, A_k) é uma árvore, temos que provar que a cada passo, cada vértice ou aresta adicionados em S e A ainda deixam a árvore conexa e acíclia.

- *Acíclico*: Como S só possui aresta de menor peso entre seus elementos, então os elementos de S não mais mais de uma aresta para qualquer outro elemento de S. O que garante a não existência de ciclos.
- *Conexo*: A conexidade é garantida pois, como S começa com um vértice (por definição conexo), e a cada inserção em S tem-se uma aresta que liga os elementos do corte $E(S, \overline{S})$, então ao final da execução, a árvore geradora mínima permanecerá conexa.

Para provar que $(S_k, A_k) \subset T_{min}$.

Tomamos como **base** um grafo trivial.

Temos a seguinte **hipótese**:

Para $0 \leqslant k \leqslant a$ então $(S_k, A_k) \subseteq T_{min}$.

$(S_{a+1}, A_{a+1}) = (S_a + w \in V(G), A_a + \{u, w\} \in E(G))$

Suponha que $\{u, w\} \notin T_{min}$. Assim temos que $T_{min} + \{u, w\}$ é um ciclo, e $\exists e \in T_{min} \therefore e \in E(S_a, \overline{S_a})$.

E ainda e faz parte do mesmo ciclo que $\{u, w\}$.

Seja $T* = T_{min} + u, w - e$.

Como $\{u, w\}$ e e estão no corte $E(S_a, \overline{S_a})$ e o algoritmo escolheu $\{u, w\}$, logo $c(\{u, w\}) \leqslant c(e)$.

Podemos ainda deduzir que $c(T*) = c(T_{min}) + c(\{u, w\}) - c(e)$.

Logo, $c(\{u, w\}) - c(e) \leqslant 0$.

Logo, $c(T*) \leqslant c(T_{min})$.

Como T_{min} tem o custo mínimo, logo, $c(T*) = c(T_{min})$ e $T*$ é uma árvore geradora mínima.

A cada passo tem-se então uma árvore geradora mínima, logo ao final teremos também a árvore geradora mínima.

A complexidade do algoritmo, se utilizada a estrutura *Heap*, é dada por $\theta(m \log^* n)$.

7.7 Algoritmo de Kruskal para para árvore geradora mínima

A idéia do algorítmo de Kruskal também é bastante simples, a cada passo escolhemos a aresta mais barata dentre as que ainda não foram escolhidas, com a única condição que essa aresta não forme um ciclo com as arestas que já foram escolhidas.

O Algorítmo 11 descreve a estratégia de Kruskal.

No algoritimo nota-se que a procura por um ciclo em um grafo (do ponto de vista computacional) não é algo trivial de ser implementado, por esse motivo, precisamos de estruturas de dados que, dinamicamente, representem e manipulem conjuntos distintos de vértices de modo eficiente.

Algoritmo 11: Kruskal(G)

Entrada: um grafo G com peso ω nas arestas.
Saída: arvore geradora de custo mínimo.

```
1 início
2     S ← ∅;
3     F ← fila das arestas em ordem não-decrescente de peso;
4     para cada e ∈ F faça
5         se S ∪ {e} não induz circuito em G então
6             insira e em S;
7     devolva (V, S)
```

Estruturas como essas são conhecidas como união-e-busca. Essas estrutruas mantêm dinamicamente uma família de subconjuntos distintos com um elemento de cada subconjuto eleito como representando do subconjunto, e temos as operações:

- $faz(x)$ cria o conjunto unitário $\{x\}$, com representante x;
- $busca(x)$ devolve o representante do conjunto ao qual x pertence;
- $uniao(x, y)$ substitui os conjuntos que contém x e y pela união desses conjuntos (e determina um representane para essa união).

Com isso é possível obter o Algoritmo 11 que é uma modificação do Algoritmo 12 com a diferença de implementar a estrutura de união-e-busca.

Algoritmo 12: Kruskal(G)

Entrada: um grafo G com peso ω nas arestas.
Saída: arvore geradora de custo mínimo.

```
1 início
2     S ← ∅;
3     F ← fila das arestas em ordem não-decrescente de peso;
4     para cada e ∈ F faça
5         se busca(u) ≠ busca(v) então
6             insira e em S;
7             uniao(u, v);
8     devolva (V, S)
```

7.7.1 Avaliação de Corretude

Teorema de Kruskal

Kruskal devolve uma árvore geradora mínima.
Suponha que o algoritmo devolve $T = (V, A)$.
Devemos provar que:

- T é uma árvore;
- T é mínima.

Para provar que T é uma árvore, basta provar que o grafo é acíclico e conexo.

- *Acíclico*: Isso é garantido pela restição da seleção das arestas que não formam ciclos.
- *Conexo*: Se a árvore gerada a cada passo é acíclica, tenta-se então mostrar que o árvore deve ter $|E(G)| = |V(G)| - 1$ ou $m = n - 1$. Tendo assim duas propriedades (acíclico e $m = n - 1$) infere-se a terceira, a conexidade. Ao tentar colocar a aresta $n - 1$ conecta-se duas componentes anteriormente desconexas. O ato de conectar as componenetes não gera ciclos, então temos duas propriedades e inferimos a terceira.

Para provar que T é mínima, vamos supor a existência de uma outra árvore geradora mínima denotada por $T*$.

Prova do Teorema de Kruskal

Seja e uma aresta de T que não está em $T*$ e que entrou em T antes.
Seja $(e_1, e_2, \ldots, e_{n-1})$ a sequência em que as arestas entraram na árvore T.
Seja $e_1, e_2, \ldots, e_{j-1} \in E(T*)$.
Seja $e = e_j$ que é a primeira aresta que está em T mas não está em $T*$.
Então $T* + e_j$ tem um ciclo.
E nesse ciclo existe uma aresta f que não está em T.
Logo, f foi analisada depois de e_j e portanto $c(e_j) \leqslant c(f)$.
Seja $T' = T* - f + e$.
Logo, $c(T') \leqslant c(T*)$.
Como $T*$ é mínima.
Então, $c(T') = c(T*)$ e T' é mínima.

Capítulo 8

Emparelhamentos

Um emparelhamento é um conjunto independente de arestas. É um tópico bastante estudado da Teoria dos Grafos, pela ampla variedade de aplicações. Achar o emparelhamento em um grafo, é em sua essência achar pares de vértices que podem ser conectados.

Os pares de vértices que podem ser conectados (com base em um grafo G) formam um aresta.

O conjunto de arestas M de G é chamado de **emparelhamento**. Obviamente duas arestas do conjunto M não incidem em um mesmo vértice.

Todos os vértices da indução $H = G[M]$ tem grau 1, ou seja, H é um grafo *1-regular*. Ou formalmente: $M \subseteq E(G)$ é um emparelhamento se $G[M]$ é *1-regular*.

8.1 Definições

Um vértice é **coberto** por um emparelhamento M se está em um conjunto de arestas em M. Ou seja, $v \in V(G[M])$.

A Figura 8.1 é um exemplo de emparelhamento.

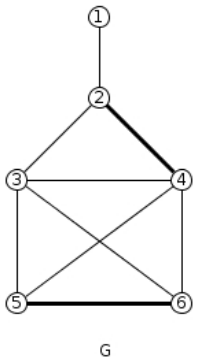

Figura 8.1: $M = \{\{2, 4\}, \{5, 6\}\}$ é um emparelhamento no grafo G.

Os vértices $2, 4, 5, 6$ são ditos cobertos no exemplo da Figura 8.1.

Um emparelhamento é **perfeito** se $V(G[M]) = V(G)$, ou seja, se todos os vértices do grafo original são cobertos por M.

Um emparelhamento é **máximo** se não exsite emparelhamento maior.

Sua notação formal é dada por:

$$\mu(G) = max\{|M| \therefore M \text{ é um emparelhamento em } G\}$$

Temos que o μ máximo para um caminho P^n tal que n é o tamanho do caminho, é $\frac{n}{2}$. Em circuitos emos que o μ máximo C^n tal que n é o tamanho do circuito, é também $\frac{n}{2}$.

Observação: Se o número de vértices de G for ímpar, um emparelhamento nunca será perfeito. Mas o número de vértices par não garante que exista um emparelhamento perfeito.

Como pode ser observado no grafo G da Figura 8.2

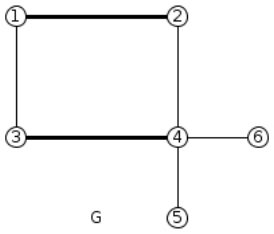

Figura 8.2: Número de vértices par, que não aceitam emparelhamento perfeito.

Isso acontece pois os vértices $2, 4, 5, 6$ competem por uma mesma aresta. E se um está o outro não pode estar.

8.2 Caminho M-Alternante

C é um caminho *M-Alternante* se é um caminho e suas arestas estão alternadamente em M e fora de M.

$$\begin{aligned} C \quad &= \quad (v_0, v_1, \ldots, V_k) \text{ então} \\ &\text{ou} \quad \{V_0, V_1\}, \{V_2, V_3\}, \ldots, \{V_{2_i}, V_{2_i+1}\} \\ &\text{ou} \quad \{V_1, V_2\}, \{V_3, V_4\}, \ldots, \{V_{2_i+1}, V_{2_i+2}\} \end{aligned}$$

são arestas em M.

8.3 Caminho M-Aumentante

Um caminho *M-Alternante* $C = (V_0, V_1, V_2, \ldots, V_k)$ tal que K seja impar e V_0 não seja coberto por M é um caminho *M-Aumentante*

8.4 Teorema do Caminho M-Aumentante

Dado um grafo G e um emparelhamento M em G, se existe um caminho *M-Aumentante* C em G então existe um emparelhamento M' em G tal que o tamanho de M' é o mesmo de M mais um. Ou formalmente $|M'| = |M| + 1$.

Prova do Teorema do Caminho M-Aumentante Para uma aresta $e \in M$ tal que e não usa vértices de C coloque e e, M'. As arestas de C que não são de M entram em M'.

Ou seja, se existe um emparelhamento M-Aumentante, então existe um outro emparelhamento equivalente a esse, entretanto possui uma aresta a mais.

Observações: Não existe uma aresta e que pertence a M e que usa vértices de C que não estejam em C.

As arestas de C que não estão em M não compartilham vértices com as arestas de M que não usam vértices de C.

Para provar a volta do teorema, ou seja, provar que: sendo G um grafo e M um emparelhamento em G, M é máximo se e somente se não existe um caminho *M-Aumentante* em G.

Para isso, definimos como $M*$ um emparelhamento máximo em G.

Seja $H = G[M \cup M*]$, então podemos afirmar que o número máximo de arestas para o subgrafo H é dado por $\Delta(H) \leqslant 2$, pois H pode conter ou arestas vindas de M, ou arestas vindas de $M*$, as arestas que coincidem são desconsideradas.

Com isso pode-se perceber que H contém apenas:

- vértices isolados e desconexos;
- arestas isoladas e desconexas;
- caminhos e ciclos:
  - de tamanho ímpar;
  - de tamanho par;

Como a intenção é provar que o $|M^*| > |M|$ então descarta-se: vértices isolados e arestas isoladas, pois não contribuirão para a contagem.

São componenetes conexas de H: caminhos e ciclos.

Nos ciclos pares o número de arestas de M é igual ao número de arestas de M^* então podem ser descartados.

Nos ciclos ímpares uma aresta sempre ficará de fora, logo também podem ser descartados.

Os caminhos de tamanho par tem o número de arestas de M e de M^* igual, então também podem ser descartados.

Os caminhos de tamanho ímpar podem ter:

1. mais arestas de M^* do que de M;
2. mais arestas de M do que de M^*;

Logo pode-se deduzir que somente a afirmação 1 é verdadeira, pois inicia-se com arestas que estão em M^*, então M^* é aumentante e M não é máximo.

8.5 Diferença Simétrica

A prova do teorema do caminho *M-Aumentante* pode ser denotada como uma operação chamada de **diferença simétrica**.

Sua notação formal é dada por:

$$M \Delta E(P) = (M \cup E(P)) \setminus (M \cap E(P))$$

Onde M é o conjunto de arestas do emparelhamento e $E(P)$ é o caminho *M-Aumentante*.

E como o teorema prova, o resultado dessa operação é um M' que é um outro emparelhamento em G com uma aresta a mais que M.

Capítulo 9

Planaridade

Um grafo diz-se planar se for possível desenhá-lo de tal forma que duas arestas não se cruzem no plano.

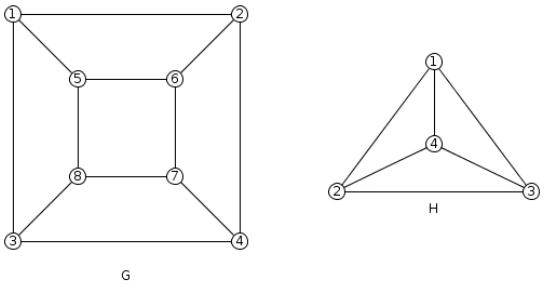

Figura 9.1: Grafos Planares

O grafo G da Figura 9.1, temos um cubo planar. O grafo H é um K^4 desenhado de forma planar.

9.1 Definições

Um grafo planar divide o plano em **regiões**, delimitado por suas arestas. Cada uma destas divisões é denominada por **face** do grafo.

Dois vértices u e v estão na mesma face, se existe uma aresta $\{u, v\}$ que não cruza nenhuma outra aresta do grafo.

No grafo G, existem 6 faces, pois a face "exterior"também é contabilizada. Essa face esta é denominada por face infinita, ou face exterior.

Um grafo I pode ser semelhante a um grafo I' ou formalmente $I\ I'$, se a disposição de seus vértices e arestas não alteram suas faces.

Para exemplificar a semelhança entre os grafos planares, os grafos I e I' da Figura 9.2 mostram um exemplo.

9.2 Teorema de Kuratowski

O teorema diz que G é planar se e somente se não existe H que é subgrafo de G que seja uma subdivisão do grafo completo K_5 ou do grafo $K_{3,3}$.

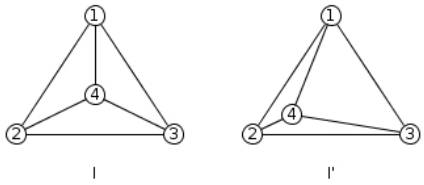

Figura 9.2: Grafos Planares Semelhantes

Com isso, sempre que identificarmos subgrafos do tipo K_5 ou $K_{3,3}$ podemos descatar a possibilidade da construção planar do grafo.

Os grafos K_5 ou $K_{3,3}$ estão dispostos respectivamente como G e H na Figura 9.3.

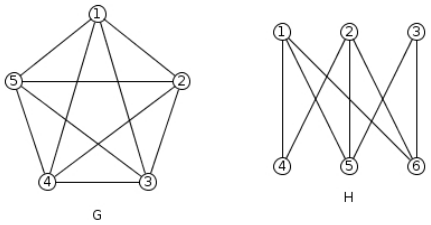

Figura 9.3: Grafos Planares

9.3 Algoritmo de Hopcroft Tarjan

Em 1974, Hopcroft e Tarjan apresentaram o primeiro algoritmo linear para decidir se um dado grafo é planar. Uma das etapas para a execussão do algoritmo é a identificação de componentes biconexas de um grafo.

9.4 Encontrar Blocos Bi-conexos de um Grafo

Define-se como um bloco uma componente bi-conexa ou um caso especial definido por uma aresta e dois vértices.

Para encontrar esses blocos devemos encontrar as articulações de um grafo.

Cada um desses blocos formam uma árvore, e assim podem ser subdivididos.

9.4.1 Encontrar as Articulações

Para encontrar as articulações em um grafo devemos definir duas funções chamadas de f e n tal que:

$$f : V(G) \to V(G)$$

$$n : V(G) \to \mathbb{N}$$

A função f retorna o vértice mais próximo da raiz que pode ser alcançado a partir de v com 0 ou mais arestas de árvore (para baixo) e com apenas 1 aresta de retorno para cima.

A função n é a distância da raiz até o vértice em questão, utilizando somente arestas de árvore.

Para exemplificar a utilização das funções f e n vamos utilizar o grafo da Figura 9.4

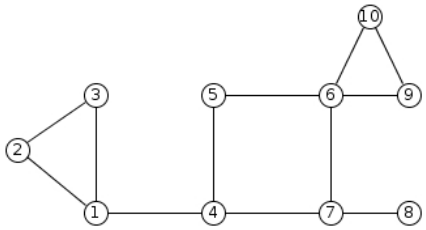

Figura 9.4: Busca por Articulações

A partir desse grafo, podemos gerar o mesmo grafo estruturado de forma diferente com base em uma busca em profundidade. A Figura 9.5 mostra esse novo grafo.

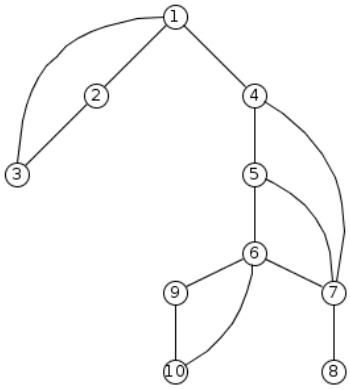

Figura 9.5: Representação Estruturada em uma Árvore

Observações: Se um vértice x e tem mais que um filho então x é uma articulação.

Se x não é raiz e existe pelo menos um ramo de descendentes sem arestas de retorno para ancestrais de x então x é uma articulação.

A Tabela 9.1 mostra as funções f e n preenchidas para o exemplo da Figura 9.5.

Para afirmar se um vértice v é ou não articulação verifica-se:

Se $n(v) \neq \emptyset$ e existe um filho de v que tem $n(f(w)) \geqslant n(v)$ então v é uma articulação. Mesmo que essa comparação funcione apenas para um de seus filhos.

No exemplo os vértices $1, 4, 6, 7$ são articulações.

9.4.2 Como calcular a Função f

Para calcular a função f temos as seguintes opções:

1. atribuir v - caso inicial;

v	$f(v)$	$n(v)$
1	1	0
2	1	1
3	1	2
4	4	1
5	4	2
6	4	3
7	4	4
8	8	5
9	6	4
10	6	5

Tabela 9.1: Tabela com os valores das funções f e n para v.

2. atribuir $f(w) \therefore w$ é filho de v;

3. atribuir u se $\{u, w\}$ é uma aresta de retorno partindo de v.

Um possível problema a ser tratado em um algorítmo recursivo é a questão de como saber o f de meu filho se ainda não foi calculado. Isso é garantido pela pilha de recursão em profundidade.

9.5 Aplicações

Escrever as aplicações de planaridade.

Capítulo 10

Coloração

O problema da coloração em grafos surgiu da necessidade que os desenvolvedores de mapas da antiguidade tinham em obter cores diferentes. Quanto menos cores pudessem usar, melhor. Para colorir os mapas usavam cores de modo que nenhuma região que tivesse fronteira com outra ficasse com a mesma cor. Com isso perceberam que utilizando 4 cores era possível colorir todo o mapa, mas esse questionamento perdurou por muito tempo até que fosse provado em 2004.

Como os mapas formam grafos planares, tem-se então o teorema das 4 cores.

10.1 Teorema das 4 Cores

Se G é um grafo planar, então pode ser colorido com 4 cores.

A prova do teorema é complicada e não se aplica no alvo de estudos desse livro.

10.2 Definição de Coloração

Dado um grafo G, define-se como uma coloração, uma função $c : v(G)$ tal que se $\{u, v\} \in E(G)$ então $c(u) \neq c(v)$.

O número de cores de uma coloração c é o tamanho da imagem de c ou seja: $|c| = |c(V(g))|$.

Observação: Toda coloração c de tamanho k pode ser representada (trocada) por $c' : V(G) \to \{1, \ldots, k\}$. Ou seja, a coloração pode se representada por um número natural.

10.2.1 Número Cromático

O número cromático de um grafo G é definido por $\chi(G)$ que é o menor número de cores de uma coloração de G.

Observações: Um ciclo de tamanho par pode ser colorido com duas cores.

Um ciclo de tamanho ímpar pode ser colorido com 3 cores.

10.2.2 Conjuntos Independentes

Uma coloração C divide o grafo G em conjuntos independentes, para cada cor i o conjunto independente $C^{-1}(i)$, ou seja, todo $v \in V(G) \therefore C(v) = i$ é um conjunto independente. Ou seja, classificar um

grafo em conjuntos independentes é também colori-lo, pois o número de conjuntos independentes de um grafo é também seu número cromático.

10.2.3 Teorema da Coloração para Grafos Bipartidos

Todo grafo bipartido se pinta com duas cores.

Prova: Como um grafo bipartido separa dois conjuntos de vértices que não possuem arestas entre sí, pode-se inferir que os elementos do mesmo conjunto podem estar em uma mesma coloração.

10.2.4 Teorema da Coloração para Grafos K-partidos

Todo grafo k-partido pode ser pintado com $\chi(G) = k$.

Prova: Assim como o grafo bipartido um k-partido possui k conjuntos de vértices que não possuem arestas entre sí, ou seja, k conjuntos independentes. Com isso infere-se k colorações.

10.2.5 Teorema da Clique Máxima

O número cromático de G é maior ou igual a clique máxima de G. Ou formalmente, $\chi(G) \geqslant \omega(G)$.

10.2.6 Teorema do Conjunto Independente Máximo

O número cromático de G é menor ou igual a $\frac{n}{\alpha(G)}$. Ou formalmente, $\chi(G) \leqslant \frac{n}{\alpha(G)}$.

10.3 Grafos Perfeitos

Um grafo é dito perfeito se $\chi(G) = \omega(G)$, ou seja quando seu número cromático é igual a clique máxima em G. A classificação desses grafos é interessante pois, apesar de muitos algoritmo estarem na classe de dificuldade de resolução np, quando se detecta um grafo perfeito é possível resolver muitos dos algoritmos estudados em tempo polinomial.

10.4 Aplicações

Escrever as aplicações de coloração.

Capítulo 11

Fluxo em Redes

Os fluxos em redes, são estudados normalmente em um tipo de grafo não abordado com profundidade nesse livro, os *grafos dirigidos*. Um grafo é dito dirigido se ele possui um par **ordenado** de vértices de tal forma que uma aresta possui um certo sentido. Então podemos dizer que a aresta $\{u, v\}$ se torna diferente (nesse caso) da aresta $\{v, u\}$. Os fluxos tem comportamento ainda melhor com *grafos orientados*. Um grafo é dito orientado se o sentido de suas arestas não forma um ciclo no grafo, ou seja, uma aresta que aponta para um sucessor de um vértice v não terá nenhum de seus sucessores que apontam para um predecessor de v.

Ao se trabalhar com Fluxo em Redes, ainda é necessário atribuir pesos nas arestas de tal forma que um grafo $G = (V, E, \omega)$ tal que $E = \binom{V}{2}$ e E é um par ordenado. $\omega : E \to \mathbb{R}$ é uma função que atribui um peso a cada aresta.

Notação: Arestas em grafos dirigidos são chamadas de arcos.

11.1 Definições

Os fluxos estão intimamente ligados com os pesos nas arestas que incidem em determinado vértice. Suponha dado um grafo dirigido G e uma função ω que atribui um número inteiro a cada arco de G. Diremos que o valor de ω num arco é o *fluxo no arco*. Para qualquer vértice v de G, o *influxo* (denotado por inf) em v é a soma dos fluxos nos arcos que entram em v. O *efluxo* (denotado por ef) de v é a soma dos fluxos nos arcos que saem de v.

O saldo em v é a diferença do *efluxo* com o *influxo*.

$$S = ef(v) - inf(v)$$

Portando, o saldo em v é o que sai de v menos o que entrou em v.

Observação: Não confundir com a diferença $inf(v) - en(v)$.

11.2 Fluxo Máximo

O problema a ser estudado solicita que se encontre o *fluxo máximo*, que pode ser identificado entre dois vértices, aqui nomeados de s e t.

Encontrar o fluxo máximo diz respeito a encontrar o máximo de fluxo que t poderá receber a partir de s levando em consideração os pesos estipulados pela função ω.

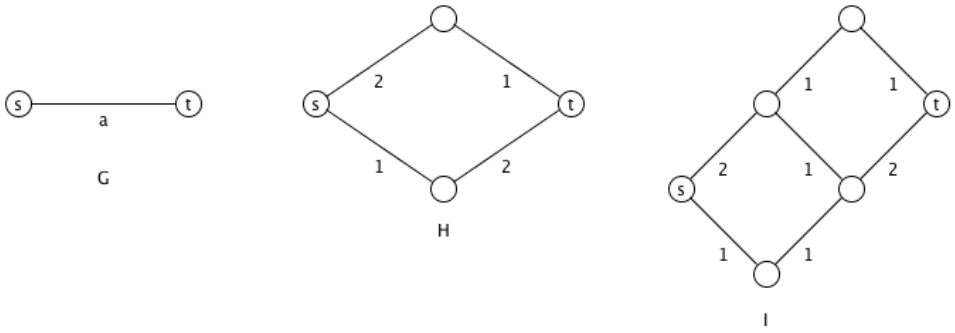

Figura 11.1: caption

A Figura 11.1 mostra exemplos de fluxos.

No grafo G da Figura 11.1 fica claro que o fluxo máximo de t partindo de s é a. No grafo H pode-se observar que o fluxo máximo que chega em s é 2. Pois o **fluxo máximo é limitado pelo menor peso do caminho escolhido entre s e t.** No grafo I podemos observar que o fluxo que chega em t é no máximo 3.

Observação: Os vértices escolhidos para fazer o papel de s e t não precisam ter quaisquer propriedades especiais. Na prática, entretanto, não há mal em supor que s é uma fonte e t é um sorvedouro. Poderíamos até mesmo supor que s é a única fonte e t o único sorvedouro do grafo dirigido.

11.3 Corte Mínimo

Uma outra definição mais restrita para corte se dá por: Dado um grafo dirigido com vértice inicial s e vértice final t, um corte é qualquer partição $E = (A, B)$ do conjunto de vértices tal que s está em A e t está em B.

Em um grafo com peso nas arestas o corte mínimo é definido por um corte $E(S, \overline{S})$ tal que esse corte seja feito onde minimize-se o custo das arestas.

11.3.1 Teorema do Corte

Qualquer corte $E(s, t)$ é maior ou igual ao fluxo entre (s, t). Isso pois em qualquer corte feito no grafo, o número resultante da somatória das arestas do corte nunca poderá ser maior que o fluxo final que t irá receber.

11.3.2 Teorema do Fluxo Máximo

O fluxo máximo é igual ao corte mínimo.

11.4 Algoritmo de Ford e Fulkerson

O algoritmo foi publicado por Ford e Fulkerson em 1962, e propõe que se encontre caminhos de s a t de modo que esses caminhos possam ser modificados de forma a achar o corte mínimo e consequentemente o fluxo máximo.

Apêndice A

Definições de Símbolos

- $\mathbb{N}$ denota o conjunto dos números naturais;
- $\mathbb{Z}$ denota o conjunto dos números inteiros;
- $\mathbb{Q}$ denota o conjunto dos números racionais;
- $\mathbb{R}$ denota o conjunto dos números reais;
- $\mathbb{R}^+$ denota o conjunto dos números reais positivos;
- $|X|$ denota a cardinalidade do conjunto X;
- para $X \subseteq V$ denotamos por $\overline{X}$ o complemento de X em V , isto é, o conjunto $\frac{X}{V}$;
- $\binom{V}{2}$ denota o conjunto dos subconjuntos de V de cardinalidade 2:

$$\binom{V}{2} = X \subseteq V : |X| = 2$$

- $\therefore$ representa *tal que;*