CI-1065 — Algoritmos e Teoria dos Grafos

Notas de Aula

Renato Carmo André Guedes Murilo da Silva

Primeiro Semestre Letivo de 2021

setembro de 2021 a janeiro de 2022

Sumário

Apresentação

Estas notas de aula foram preparadas para a oferta da disciplina Algoritmos e Teoria dos Grafos (CI-1065) no Segundo Período Especial de Ensino Emergencial Remoto da Universidade Federal do Paraná, na contingência da pandemia de CoViD-19. Desde então, vem sendo aprimoradas.

A modalidade remota da disciplina está organizada em torno deste texto. O texto, por sua vez, está dividido em Tópicos. Cada tópico é complementado por exposições em vídeo. Ao início de cada tópico estão indicados exercícios recomendados. A Lista de Exercícios é o Apêndice A deste texto.

Na página da disciplina você encontra o cronograma que estabelece a distribuição dos tópicos ao longo do período letivo.

Na página você também encontra "links" para todo o material de apoio (vídeos expositivos, "slides", sala de videoconferência etc). Para facilitar sua localização, os nomes dos arquivos de "slides" são iguais e iniciam pelo número do tópico que complementam.

Este arquivo pdf é um hipertexto. Seus "links" são ativos permitindo navegação com um leitor apropriado.

Apesar do esforço para revisar o texto a fim de prepará-lo para uso dos alunos, é praticamente inevitável que erros e imprecisões ainda restem. Caso detecte algum, por favor, comunique enviando mensagem a `ci1065@listas.inf.ufpr.br` para que possamos corrigir.

Os exercícios estão classificados de acordo com a seguinte legenda.

@: exercícios programados para discussão em aula: procure fazê-los antes de serem discutidos em aula.

⋆: exercícios prioritários: na impossibilidade de fazer todos, dê prioridade a estes.

#: exercícios mais difíceis e/ou trabalhosos: não comece por estes.

-: exercícios de interesse marginal: complementam o assunto de alguma forma mas podem ser ignorados sem comprometer o entendimento.

Bom estudo!

Tópico 1

Primeiros Conceitos

Exercícios 1 a 8

Se C é um conjunto e k é um inteiro, então 2^C denota o conjunto dos subconjuntos (ou conjunto das partes) de C e $\binom{C}{k}$ denota o conjunto dos subconjuntos de C com exatamente k elementos, isto é,

$$\begin{aligned} 2^C &:= \{S \mid S \subseteq C\}, \\ \binom{C}{k} &:= \{S \subseteq C \mid |S| = k\}. \end{aligned}$$

Um *grafo* (simples, não–direcionado) G é um par $(V(G), E(G))$ onde $V(G)$ é um conjunto finito e não vazio e $E(G) \subseteq \binom{V(G)}{2}$.

Cada elemento de $V(G)$ é chamado de *vértice* (*"vertex"*) de G e cada elemento de $E(G)$ é chamado de *aresta* (*"edge"*) de G.

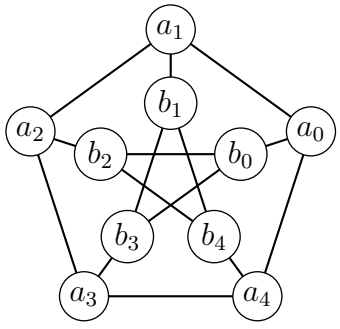


Figura 1.1: Grafo de Petersen

No que segue, G é um grafo e u e v são vértices.

Se $a = \{u, v\}$ é uma aresta de G, dizemos que u e v são as *pontas* da aresta a.

Neste caso dizemos que

- a é *incidente* em u e em v
- os vértices u e v são *vizinhos* (ou *adjacentes*) em G.

Duas arestas são *adjacentes* se tem uma ponta em comum.

A *vizinhança de um vértice* v em G é o conjunto dos vértices que são vizinhos de v em G, isto é,

$$\Gamma_G(v) := \{u \in V(G) \mid u \text{ é vizinho de } v\}.$$

Por extensão, definimos a *vizinhança de um conjunto de vértices* como sendo a união das vizinhanças dos vértices no conjunto, isto é,

$$\Gamma_G(X) := \bigcup_{v \in X} \Gamma_G(v)$$

Um vértice é *isolado* se não tem vizinhos.

A *fronteira de um conjunto* X de vértices de G é o conjunto de arestas de G com uma ponta em X e a outra fora de X, isto é,

$$\partial_G(X) := \{\{x, y\} \in E(G) \mid x \in X \text{ e } y \notin X\}.$$

Convencionamos

$$\partial_G(v) := \partial_G(\{v\}).$$

A fronteira de um vértice v também é chamada de *estrela* do vértice v.

Um *grafo trivial* é um grafo com um único vértice e sem arestas.

Um *grafo completo* é um grafo em que cada vértice é vizinho de todos os outros. É comum indicar por K_n um grafo completo de n vértices.

O *complemento* (ou *grafo complementar*) de um grafo G é o grafo $\overline{G}$ dado por

$$\begin{aligned} V(\overline{G}) &:= V(G), \\ E(\overline{G}) &:= \binom{V(G)}{2} - E(G). \end{aligned}$$

A *matriz de adjacência* do grafo G é a matriz M_G indexada por $V(G) \times V(G)$ dada por

$$M_G[u,v] = \begin{cases} 1, & \text{se } \{u,v\} \in E(G), \\ 0, & \text{se } \{u,v\} \notin E(G). \end{cases}$$

M_G é uma matriz simétrica de $|V(G)|^2$ posições onde os elementos da diagonal principal são todos nulos.

Dependendo do contexto, a matriz de adjacência pode ser encarada como uma matriz inteira ou como uma matriz booleana.

Se G e H são grafos, a união de G e H é o grafo $G \cup H$ dado por

$$\begin{aligned} V(G \cup H) &:= V(G) \cup V(H), \\ E(G \cup H) &:= E(G) \cup E(H). \end{aligned}$$

Dados vértices $u, v \in V(G)$, o grafo $G + \{u,v\}$ é o grafo obtido ao acrescentar ao grafo G a aresta $\{u,v\}$, isto é,

$$\begin{aligned} V(G + \{u,v\}) &:= V(G), \\ E(G + \{u,v\}) &:= E(G) \cup \{u,v\}. \end{aligned}$$

Tópico 2

Isomorfismo entre Grafos

Exercícios 9 a 10

Um *isomorfismo* entre dois grafos é uma bijeção entre seus vértices que preserva vizinhanças.

Mais precisamente, um isomorfismo entre os grafos G e H é uma bijeção $f\colon V(G) \to V(H)$ satisfazendo

$$\{u, v\} \in E(G) \text{ se e somente se } \{f(u), f(v)\} \in E(H).$$

Observe que se $f\colon V(G) \to V(H)$ é isomorfismo, então $f^{-1}\colon V(H) \to V(G)$ também é.

Dizemos que os grafos G e H são *isomorfos* se existe um isomorfismo entre eles.

Um *automorfismo* sobre um grafo G é um isomorfismo entre G e G.

Tópico 3

Grau

Exercícios 12 a 20

O *grau* de um vértice v em um grafo G é o número de arestas de G incidentes em v, isto é,

$$\delta_G(v) := |\partial_G(v)|.$$

Um grafo é

regular se todos os seus vértices tem o mesmo grau.

k***−regular*** se todos os seus vértices tem grau k.

A *matriz de incidência* de um grafo G é a matriz $\overline{M}_G$, indexada por $V(G) \times E(G)$ dada por

$$\overline{M}_G[v,a] = \begin{cases} 0, & \text{se } v \notin a, \\ 1, & \text{se } v \in a. \end{cases}$$

Teorema 1. *Em todo grafo G,*

$$\sum_{v \in V(G)} \delta_G(v) = 2|E(G)|.$$

Demonstração. Seja G um grafo e seja M sua matriz de incidência. Vamos calcular a soma das entradas de M.

$$S(G) = \sum_{a \in E(G), v \in V(G)} M[v,a].$$

$$M: \begin{array}{c|cccc} & a_1 & a_2 & \dots & a_m \\ \hline v_1 & M[v_1,a_1] & M[v_1,a_2] & \dots & M[v_1,a_m] \\ v_2 & M[v_2,a_1] & M[v_2,a_2] & \dots & M[v_2,a_m] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ v_n & M[v_n,a_1] & M[v_n,a_2] & \dots & M[v_n,a_m] \end{array}$$

Somando pelas linhas de M temos

$$S(G) = \sum_{v \in V(G)} \sum_{a \in E(G)} M[v,a],$$

e como

$$\sum_{a \in E(G)} M[v,a] = \delta_G(v),$$

para cada $v \in V(G)$, então

$$S(G) = \sum_{v \in V(G)} \delta_G(v).$$

Somando pelas colunas de M temos

$$S(G) = \sum_{a \in E(G)} \sum_{v \in V(G)} M[v,a],$$

e como

$$\sum_{v \in V(G)} M[v,a] = 2$$

para cada $a \in E(G)$, então

$$S(G) = \sum_{a \in E(G)} 2 = 2|E(G)|.$$

Então,

$$\sum_{v \in V(G)} \delta_G(v) = 2|E(G)|.$$

□

Corolário 2 ("handshaking lemma")**.** *Em todo grafo o número de vértices de grau ímpar é par.*

A *sequência de graus* de um grafo G é a sequência $(\delta(v_1), \ldots, \delta(v_n))$ satisfazendo

$$\delta(v_1) \geq \delta(v_2) \geq \ldots \geq \delta(v_{n-1}) \geq \delta(v_n) \text{ e } \{v_1, \ldots, v_n\} = V(G).$$

Uma sequência não crescente de inteiros $S = (d_1, \ldots, d_n)$ é uma *sequência gráfica* se existe um grafo de n vértices cuja sequência de graus é S.

O *grau máximo* e o *grau mínimo* de um grafo G são denotados, respectivamente,

$$\begin{aligned} \Delta(G) &:= \max\{\delta_G(v) \mid v \in V(G)\}, \\ \delta(G) &:= \min\{\delta_G(v) \mid v \in V(G)\}. \end{aligned}$$

Tópico 4

Grafos Direcionados

Exercícios 22 a 24

Um *grafo direcionado* (*"digraph"*) G é um par $(V(G), A(G))$ onde $V(G)$ é um conjunto finito e $A(G) \subseteq V(G) \times V(G)$. Cada elemento de $A(G)$ é chamado de *arco* (*"arc"*) de G.

Se $a = (u, v)$ é um arco de G, dizemos que u e v são as pontas do arco a.

Dizemos ainda que

- o arco a está *direcionado de u para v*;
- o arco a é um arco de u para v;
- o arco a *incide* em u e v
- o vértice u é a *origem* (*"tail"*) de a;
- o vértice v é o *destino* (*"head"*) de a;
- o arco a *sai* do vértice u e *entra* (ou *chega*) no vértice v;
- v é *vizinho de saída* de u;
- u é *vizinho de entrada* de v;
- u e v são *vizinhos* (*adjacentes*).

Quando suas pontas coincidem, isto é, $u = v$ dizemos que o arco a é um laço (*"loop"*).

Dizemos que o grafo direcionado G é *simétrico* se satisfaz

$$(u, v) \in A(G) \implies (v, u) \in A(G), \text{ para todo } u, v \in V(G).$$

Um grafo direcionado G é *anti–simétrico* se satisfaz

$$(u, v) \in A(G) \implies (v, u) \notin A(G), \text{ para todo } u, v \in V(G).$$

O grafo *transposto* (ou *reverso*) de um grafo direcionado G é o grafo direcionado G^T com o mesmo conjunto de vértices de G onde cada arco de G é substituído por um arco com as mesmas pontas mas a direção invertida, isto é,

$$\begin{aligned} V(G^T) &= V(G), \\ A(G^T) &= \{(v, u) \mid (u, v) \in A(G)\}. \end{aligned}$$

Num grafo direcionado, a vizinhança de um vértice v está particionada em dois conjuntos, chamados de *vizinhança de saída* de v e *vizinhança de entrada* em v, respectivamente,

$$\begin{aligned} \Gamma^+_G(v) &:= \{u \in V(G) \mid (v, u) \in A(G)\}, \\ \Gamma^-_G(v) &:= \{u \in V(G) \mid (u, v) \in A(G)\}, \end{aligned}$$

de modo que

$$\Gamma_G(v) = \Gamma^+_G(v) \cup \Gamma^-_G(v).$$

Um *sumidouro* (ou *sorvedouro*) ("sink") em um grafo direcionado G é um vértice sem vizinhos de saída

Uma *fonte* ("source") em um grafo direcionado G é um vértice sem vizinhos de entrada.

Do mesmo modo, distinguimos *fronteira de saída* e *fronteira de entrada*, respectivamente,

$$\begin{aligned} \partial^+_G(v) &:= \{(v, u) \in A(G) \mid u \in V(G)\}, \\ \partial^-_G(v) &:= \{(u, v) \in A(G) \mid u \in V(G)\}, \end{aligned}$$

de modo que

$$\partial_G(v) = \partial^+_G(v) \cup \partial^-_G(v).$$

e *grau de saída* e *grau de entrada*, respectivamente,

$$\begin{aligned} \delta_G^+(v) &:= |\partial_G^+(v)|, \\ \delta_G^-(v) &:= |\partial_G^-(v)|, \end{aligned}$$

de modo que

$$\delta_G(v) = \delta_G^+(v) + \delta_G^-(v).$$

No caso direcionado, distinguimos os graus máximo e mínimo de entrada e de saída.

$$\begin{aligned} \Delta^+(G) &:= \max\left\{\delta_G^+(v) \mid v \in V(G)\right\}, \\ \Delta^-(G) &:= \max\left\{\delta_G^-(v) \mid v \in V(G)\right\}, \\ \delta^+(G) &:= \min\left\{\delta_G^+(v) \mid v \in V(G)\right\}, \\ \delta^-(G) &:= \min\left\{\delta_G^-(v) \mid v \in V(G)\right\}. \end{aligned}$$

de modo que

$$\begin{aligned} \Delta(G) &= \max\{\delta_G(v) \mid v \in V(G)\}, \\ \delta(G) &= \min\{\delta_G(v) \mid v \in V(G)\}. \end{aligned}$$

A *matriz de incidência* de um grafo direcionado sem laços G é a matriz $\overline{M}_G$, indexada por $V(G) \times A(G)$ dada por

$$\overline{M}_G[v,a] = \begin{cases} 1, & \text{se } a \text{ sai de } v, \\ -1, & \text{se } a \text{ entra em } v, \\ 0, & \text{se } a \text{ não incide em } v. \end{cases}$$

Teorema 3. *Em todo grafo direcionado G,*

$$\sum_{v \in V(G)} \delta^+(v) = |A(G)| = \sum_{v \in V(G)} \delta^-(v).$$

Demonstração. Exercício 22 □

A matriz de adjacência de um grafo direcionado G é dada por

$$M_G[u,v] = \begin{cases} 1, & \text{se } (u,v) \in A(G), \\ 0, & \text{se } (u,v) \notin A(G). \end{cases}$$

A matriz de adjacência não é necessariamente simétrica e cada elemento de sua diagonal principal pode ter tanto 0s como 1s.

A todo grafo direcionado G corresponde um grafo não–direcionado $S(G)$, chamado *grafo subjacente de* G, com o mesmo conjunto de vértices, onde a cada arco (u, v) corresponde a aresta $\{u, v\}$, isto é,

$$\begin{aligned} V(S(G)) &= V(G), \\ E(S(G)) &= \{\{u, v\} \mid (u, v) \in A(G)\}. \end{aligned}$$

Neste caso, convencionamos

$$E(G) := E(S(G)),$$

e cometemos o abuso de chamar os elementos de $E(G)$ de "arestas do grafo G".

Tópico 5

Outros tipos de Grafos

5.1 Grafos Ponderados

Um grafo (direcionado) *com pesos* (ou *ponderado*) (weighted) é um par (G, w) onde G é um grafo (direcionado) e w é uma função que atribui a cada aresta (arco) a de G um *peso* $w(a)$.

Se X é um conjunto de arestas (arcos) de G, o *peso de* X é definido por

$$w(X) := \sum_{a \in X} w(a).$$

A matriz de adjacência de um grafo (direcionado) ponderado (G, w) é dada por

$$M_G[u, v] = \begin{cases} w(\{u, v\}), & \text{se } \{u, v\} \in E(G) \left((u, v) \in A(G)\right), \\ 0, & \text{caso contrário.} \end{cases}$$

5.2 Grafos Orientados

Um *grafo orientado* D é um grafo direcionado obtido a partir de um grafo não direcionado G atribuindo orientação a cada uma das arestas de G. Neste caso dizemos que o grafo direcionado D é uma *orientação* do grafo G.

Observe que, neste caso,

- o grafo G é o grafo subjacente do grafo direcionado D, isto é, $S(D) = G$, e
- o grafo D é anti–simétrico.

Um *torneio* é uma orientação de um grafo completo.

5.3 Grafos Mistos

Um *grafo misto* é um grafo que tem "arestas direcionadas e não–direcionadas".

5.4 Multigrafos

Um *multigrafo* é um grafo em que diferentes arestas podem ter as mesmas pontas (*arestas paralelas*) e também onde as pontas de uma aresta podem coincidir (*laços*).

Um grafo é *simples* se não tem arestas paralelas nem laços.

Tópico 6

Representação Computacional

Exercícios 25 a 27

No caso geral, a descrição de um grafo requer uma lista explícita dos vértices e das arestas.

A descrição de um grafo G consome espaço $\Theta(|V(G)| + |E(G)|)$.

6.1 Listas de Adjacência

O grafo G é representado pelos conjuntos dos vértices de G e pelos conjuntos das vizinhanças de cada vértice de G.

O conjunto $V(G)$ pode ser representado por um vetor ou por uma lista encadeada ou pode ter a representação implícita.

Para cada $v \in V(G)$, o conjunto $\Gamma_G(v)$ é representado por uma lista encadeada $L_G[v]$.

Observe que de qualquer forma temos $|V(G)|$ listas e para cada $v \in V(G)$ o tamanho (número de elementos) da lista dos vizinhos de v em G é

$$|L_G[v]| = |\Gamma_G(v)| = \delta_G(v),$$

e daí,

$$\sum_{v \in V(G)} |L_G[v]| = \sum_{v \in V(G)} \delta_G(v) \overset{\text{T } 1}{=} 2|E(G)|,$$

de maneira que a representação do grafo G consome espaço $\Theta(|V(G)| + |E(G)|)$.

6.2 Matriz de Adjacência

A representação do grafo G por sua matriz de adjacência consome espaço $\Theta(|V(G)|^2)$, mesmo que o grafo não tenha arestas.

6.3 Grafos Direcionados

No caso direcionado, para cada vértice $v \in V(G)$ pode-se usar listas separadas para representar $\Gamma^+(v)$ e $\Gamma^-(v)$ ou indicar a direção do arco em cada nó da lista.

Basta representar uma das vizinhanças.

O consumo de espaço é (assintoticamente) o mesmo do caso não direcionado.

Idem para a matriz de adjacência.

6.4 Grafos Ponderados

Nas listas de adjacência acrescenta-se um campo para o peso.

6.5 Análise Comparativa

1. O espaço consumido pela matriz é $\Theta(|V(G)|^2)$ que é "o maior possível", mesmo que o grafo não tenha arestas.

   Com listas é $\Theta(|V(G)| + |E(G)|)$.

2. Na matriz responde-se se u e v são vizinhos em tempo constante, independente do tamanho do grafo.

   Nas listas o tempo no pior caso é a $\Theta(\min\{\delta_G(u), \delta_G(v)\}) = O(\Delta(G)) = O(|V(G)|)$.

3. Nas listas de adjacência percorre-se a vizinhança de um vértice v em tempo $\Theta(\delta_G(v)) = O(\Delta(G)) = O(|V(G)|)$.

   Na matriz o tempo é proporcional a $|V(G)|$, mesmo para um vértice isolado.

4. Nas listas de adjacência percorre-se todas as arestas em tempo $O(|V(G)| + |E(G)|)$.

   Na matriz o tempo é $\Theta(|V(G)|^2)$, mesmo num grafo sem arestas.

5. Nas listas pode-se determinar o grau de um vértice v em tempo $\Theta(\delta_G(v))$.

   Na matriz o tempo é $\Theta(|V(G)|)$, mesmo quando v é vértice isolado.

6. Tanto com listas como com matrizes, acrescenta-se arestas em tempo $O(1)$.

7. Com listas é possível acrescentar e remover vértices.

   Com matrizes isto depende da possibilidade de realocar memória contígua o que pode não ser possível ou pode envolver cópia de toda a matriz para outra região de memória.

Tópico 7

Subgrafos

Exercícios 28 a 36

Um grafo H é *subgrafo* do grafo G se

$$\begin{aligned} V(H) &\subseteq V(G), \\ E(H) &\subseteq E(G). \end{aligned}$$

Neste caso, G é *supergrafo* de H.

Um subgrafo H de um grafo G é *gerador* ("spanning") de G se tem o mesmo conjunto de vértices de G, isto é, se

$$V(H) = V(G)$$

Dado $X \subseteq V(G)$,

o *subgrafo induzido* por X é o maior subgrafo de G cujo conjunto de vértices é X. Denota-se $G[X]$, isto é,

$$\begin{aligned} V(G[X]) &:= X, \\ E(G[X]) &:= E(G) \cap \binom{X}{2}. \end{aligned}$$

Definimos

$$G - X := G[V(G) - X].$$

e

$$G - v := G - \{v\}.$$

Observe que H é subgrafo induzido por vértices de G se e somente se H pode ser obtido a partir de G por remoção de vértices, isto é, se e somente se $H = G - X$ para algum $X \subseteq V(G)$ (veja o Exercício 31).

Dado $X \subseteq E(G)$,

o *subgrafo induzido* por X é o subgrafo de G cujo conjunto de arestas é X e tal que cada vértice seja ponta de uma aresta de X. Denota-se $G[X]$, isto é,

$$\begin{aligned} V(G[X]) &:= \bigcup_{a \in X} a, \\ E(G[X]) &:= X. \end{aligned}$$

$G - X$ é o grafo dado por

$$\begin{aligned} V(G - X) &:= V(G), \\ E(G - X) &:= E(G) - X. \end{aligned}$$

$$G - a := G - \{a\}.$$

Dizemos que H é um *subgrafo induzido* de G, se H é um subgrafo de G induzido por vértices de G, isto é, se

$$H = G[X], \text{ para algum } X \subseteq V(G).$$

Tópico 8

Conjuntos Independentes e Cliques

Exercícios 37 a 44

Um conjunto $X \subseteq V(G)$ é *independente* (ou *estável*) em G se nenhum vértice de X é vizinho de nenhum outro em G. Noutras palavras, X é independente se não induz arestas em G, isto é, se $E(G[X])$ é vazio. O tamanho máximo de um conjunto independente em G é denotado $\alpha(G)$.

O Problema do Conjunto Independente é o problema de, dados um grafo G e um inteiro k, determinar se G tem conjunto independente de tamanho k.

Teorema 4 (Karp (1972))**.** *O Problema do Conjunto Independente é $\mathcal{NP}$–Difícil.*

Corolário 5. *Determinar o tamanho do maior conjunto independente de um grafo é um problema $\mathcal{NP}$–Difícil.*

Demonstração. Exercício 43 □

Corolário 6. *O problema de, dados dois grafos G e H, determinar se H é isomorfo a um subgrafo induzido de G é um problema $\mathcal{NP}$–Difícil.*

Demonstração. Exercício 44 □

Um conjunto $X \subseteq V(G)$ é uma *clique* em G se todo vértice de X é vizinho de todos os outros em G. Noutras palavras, X é uma clique se induz um grafo completo, isto é, se $G[X]$ é um grafo completo. O tamanho da maior clique em G é denotado $\omega(G)$.

Teorema 7. *Os seguintes problemas são $\mathcal{NP}$–Difíceis.*

1. *Dados um grafo G e um inteiro k, determinar se G tem clique de tamanho k.*

2. *Determinar o tamanho da maior clique de um grafo dado.*

Demonstração. Exercício 37. □

As definições de união de grafos, subgrafo (gerador, induzido etc), supergrafo e conjunto independente adaptam-se naturalmente no caso de grafos direcionados. A definição de clique pode ser adaptada mas cabem variantes dependendo de se laços e arcos nos dois sentidos são exigidos

Tópico 9

Grafos Bipartidos

Exercícios 45 a 47

Uma *bipartição* de um grafo G é uma partição de $V(G)$ em até dois conjuntos independentes. Neste caso, cada um destes conjuntos independentes é chamado de *parte* da bipartição.

Um grafo é *bipartido* se admite bipartição.

Lema 8. *Um grafo G é bipartido se e somente se $E(G) = \partial_G(X)$ para algum $X \subseteq V(G)$. Neste caso, $\{X, V(G) - X\}$ é uma bipartição de G.*

Demonstração. Exercício 45. □

Corolário 9. *Se $\{X, Y\}$ é uma bipartição do grafo G, então G tem no máximo $|X||Y|$ arestas.*

Demonstração. Observe que $E(G) = \partial_G(X)$ e que

$$|\partial_G(X)| = \sum_{x \in X} |\partial_G(x)| = \sum_{x \in X} \delta_G(x) \leq \sum_{x \in X} |Y| = |X||Y|$$

□

Um grafo *bipartido completo* é um grafo bipartido onde cada vértice é vizinho de todos os vértices da outra parte. É usual indicar um grafo bipartido completo cujas partes tem, respectivamente, p e q vértices por $K_{p,q}$.

Corolário 10. *Um grafo bipartido de n vértices tem no máximo $\left\lfloor \frac{n^2}{4} \right\rfloor$ arestas.*

Demonstração. Exercício 47. □

A *Matriz de Bi-adjacência* da bipartição $\{X, Y\}$ de um grafo bipartido G é a matriz M indexada por $X \times Y$ dada por

$$M[x, y] = \begin{cases} 0, & \text{se } \{x, y\} \notin E(G), \\ 1, & \text{se } \{x, y\} \in E(G). \end{cases}$$

No caso de grafos bipartidos ponderados, definimos

$$M[x, y] = \begin{cases} 0, & \text{se } \{x, y\} \notin E(G), \\ w(x, y), & \text{se } \{x, y\} \in E(G). \end{cases}$$

Observe que

- Ao contrário do que acontece com matrizes de adjacência de grafos em geral, qualquer matriz binária pode ser interpretada como a matriz de bi–adjacência de um grafo bipartido.
- Do mesmo modo, qualquer matriz pode ser interpretada como a matriz de bi–adjacência de um grafo bipartido ponderado, o que permite interpretar qualquer problema de processamento de matrizes (por exemplo, imagens digitalizadas) como um problema de processamento de grafos bipartidos.

Tópico 10

Grafos Multipartidos e Coloração de Vértices

Exercícios 49 a 50

Se k é um inteiro, uma *k–partição* de um grafo G é uma partição de $V(G)$ em até k conjuntos independentes. Cada conjunto independente é chamado de *parte* da partição.

Dados um grafo G e um inteiro $k \leq |V(G)|$, dizemos que G é *k–partido* se admite uma k–partição.

Um grafo *k–partido completo* é um grafo k–partido onde cada vértice é vizinho de todos os vértices de todas as partes diferentes da sua. É usual indicar um grafo k–partido completo cujas partes tem, respectivamente, $n_1, \ldots, n_k$ vértices por $K_{n_1,\ldots,n_k}$.

Uma k-partição de G também é chamada de uma k-coloração de G. Neste contexto, cada parte é chamada de *cor* da coloração.

O *número cromático* de G é o menor inteiro k tal que G admite uma k-coloração e é denotado por $\chi(G)$.

Teorema 11 (Karp (1972))**.** *O problema de, dados um grafo G e um inteiro k, decidir se G pode ser colorido com até k cores é $\mathcal{NP}$–difícil.*

Corolário 12. *O problema de determinar o número cromático de um grafo dado é $\mathcal{NP}$–Difícil.*

Demonstração. Exercício 49 □

Tópico 11

Passeios

Exercícios 51 a 53

Um *passeio* ("walk") em um grafo G é uma sequência $(v_0, \ldots, v_n)$ de vértices de G na qual vértices consecutivos são vizinhos em G, isto é,

$$v_{i-1} \text{ e } v_i \text{ são vizinhos, para todo } 1 \leq i \leq n.$$

Os vértices v_0 e v_n são chamados respectivamente de *início* e *fim* do passeio P.

Dizemos que P é um *passeio de v_0 a v_n em G*. O início e o fim de um passeio P são chamados conjuntamente de *pontas* de P e os demais vértices são chamados de *vértices internos* de P.

Um passeio com um único vértice é chamado de *passeio trivial*.

Um passeio não trivial é *aberto* se suas pontas são distintas e é *fechado* se suas pontas coincidem.

O *tamanho* do passeio P é o número "arestas percorridas" e é denotado por $|P|$, isto é,

$$|(v_0, \ldots, v_n)| = n.$$

A *distância* entre os vértices u e v em G é o tamanho do menor passeio de u a v em G, e é denotada por $d_G(u, v)$, isto é,

$$d_G(u, v) := \min \{|P| \mid P \text{ é passeio de } u \text{ a } v \text{ em } G\}$$

Quando não existe passeio de u a v em G convencionamos

$$d_G(u, v) = \infty$$

O *diâmetro* de G é a maior distância entre dois vértices em G e é denotado $\text{diam}(G)$, isto é,

$$\text{diam}(G) := \max\{d_G(u,v) \mid u, v \in V(G)\}.$$

Observe que se existem vértices u e v em G tais que $d_G(u,v) = \infty$, então $\text{diam}(G) = \infty$.

O conjunto dos vértices de um passeio $P = (v_0, \ldots, v_n)$ no grafo G é o conjunto

$$V(P) := \{v_0, \ldots, v_n\},$$

e o conjunto das arestas de P é o conjunto

$$E(P) := \{\{v_{i-1}, v_i\} \mid 1 \leq i \leq n\},$$

O *grafo induzido pelo passeio* P é o grafo $G[P]$ dado por

$$\begin{aligned} V(G[P]) &:= V(P), \\ E(G[P]) &:= E(P). \end{aligned}$$

O passeio P é *gerador* do grafo G se o grafo $G[P]$ é subgrafo gerador de G, ou seja, se o passeio P passa por todos os vértices de G.

O *reverso* do passeio $P = (v_0, \ldots, v_n)$ é o passeio

$$P^{-1} := (v_n, \ldots, v_0)$$

Se $P = (v_0, \ldots, v_n)$ e $Q = (u_0, \ldots, u_m)$ são passeios sobre um grafo G tais que $v_n = u_0$, definimos a *concatenação* de P com Q como o passeio

$$P \cdot Q = (v_0, \ldots, v_n, u_1, \ldots, u_m).$$

Dizemos que um passeio S é um *segmento* de um passeio P, se existem passeios I e F tais que

$$P = I \cdot S \cdot F.$$

Dizemos que S é *segmento próprio* de P quando S é segmento de P e $S \neq P$.

Dizemos que o vértice v é *alcançável* a partir do vértice u em G se existe passeio de u a v em G.

Dizemos que o grafo G é *conexo* se existe passeio de qualquer de seus vértices a qualquer outro.

Um *componente conexo* de G é um subgrafo conexo maximal de G.

O conjunto dos componentes de G será denotado $\mathcal{C}(G)$.

Teorema 13. *Um grafo é bipartido se e somente se não tem passeios de paridades diferentes com as mesmas pontas.*

Exercício 53 Prove que um grafo é bipartido se e somente se não tem passeios de paridades diferentes com as mesmas pontas.

Resposta:

Primeiramente vamos provar que se G é bipartido, então não tem passeios de paridades diferentes com as mesmas pontas.

Seja G um grafo bipartido e sejam P e Q caminhos de u a v em G de tamanhos p e q, respectivamente.

Como G é bipartido, existe (L. 8) $X \subseteq VG$ tal que $\partial(X) = E(G)$.

Considere então o caminho

$$P \cdot Q^{-1} = (u = v_0, v_1, v_2, \ldots, v_{p+q} = u).$$

Sem perda de generalidade, podemos assumir $u \in X$. Como G é bipartido e $\{u, v_1\} = \{v_0, v_1\} \in E(G) = \partial(X)$, temos $v_1 \notin X$; do mesmo modo, concluímos que $v_2 \in X$, e, em geral, concluímos que $v_i \in X$ se e somente se i é par, para todo $0 \leq i \leq p + q$.

Como $v_{p+q} = u \in X$, concluímos que $p + q = |P| + |Q| = |P \cdot Q^{-1}|$ é par, e, portanto, $p = |P|$ e $q = |Q|$ tem a mesma paridade.

Vamos provar agora que se G não tem passeios de paridades diferentes com as mesmas pontas então, G é bipartido.

Sem perda de generalidade, podemos assumir que G é conexo com pelo menos um vértice x e uma aresta $\{u, v\}$.

Considere então o conjunto X dos vértices de G a distância par de x, isto é,

$$X := \{w \in V(G), \mid d_G(x, w) \text{ é par}\}.$$

Vamos provar que G é bipartido, provando que $\partial(X) = E(G)$.

Vamos provar que $\partial(X) = E(G)$ provando que $\{u, v\} \in \partial(X)$.

Sejam, então, P_u e P_v, respectivamente, passeios de tamanho mínimo de x a u e de x a v em G.

Se $\{u, v\} \in E(P_u)$, então as distâncias $d_G(x, u)$ e $d_G(x, v)$ tem paridades diferentes, portanto um deles está em X e o outro não. Consequentemente $\{u, v\} \in \partial(X)$.

Se $\{u, v\} \notin E(P_u)$, então $Q := P_u(u, v)$ é um passeio de x a v.

Como temos, por hipótese, que Q e P_v tem a mesma paridade, concluímos que $|P_u|$ e $|P_v|$ tem paridades diferentes.

Como P_u e P_v são mínimos, temos $|P_u| = d_G(x, u)$ e $|P_v| = d_G(x, v)$.

Então, $d_G(x, u)$ e $d_G(x, v)$ tem paridades diferentes, como antes. Portanto, em qualquer caso, $\{u, v\} \in \partial(X)$.

Tópico 12

Passeios em Grafos Direcionados e Ponderados

No que segue, G é um grafo direcionado.

Um *passeio direcionado* em G é uma sequência $(v_0, \ldots, v_n)$ de vértices de G satisfazendo

$$(v_{i-1}, v_i) \in A(G), \text{ para todo } 1 \leq i \leq n.$$

G é *fortemente conexo* se existe passeio direcionado entre quaisquer de seus vértices.

---

ag: Acho que deveríamos dizer "de qualquer vértice para qualquer outro" – não fica tão claro que esse "entre quaisquer de seus vértices" seja bidirecional.

Um *componente forte* de G é um subgrafo fortemente conexo maximal de G.

O conjunto de componentes fortes de G é denotado $\mathcal{C}(G)$.

O *grafo condensado* de G é o grafo $C(G)$ cujos vértices são os componentes fortes de G, e cujo conjunto de arcos é

$$A(C(G)) := \{(X, Y) \mid \text{ existe arco de } V(X) \text{ para } V(Y) \text{ em } G\}$$

12.1 Passeios em Grafos Ponderados

No que segua, (G, w) é um grafo (direcionado) ponderado.

O peso do passeio $P = (v_0, \ldots, v_n)$ em G é

$$w(P) := \sum_{i=1}^{n} w(\{v_{i-1}, v_i\}).$$

A distância do vértice u ao vértice v é o peso de um passeio de peso mínimo de u a v em (G, w).

Tópico 13

Caminhos

Exercícios 54 a 60

Um *caminho* ("path") é um passeio cujos vértices são todos distintos.

É comum chamar o subgrafo induzido por um caminho também de "caminho" e indicar o grafo induzido por um caminho de n vértices por P_n.

Se u e v são vértices de um caminho P, denotamos por uPv o caminho de u a v que é segmento de P.

Teorema 14. *Se P é um caminho maximal em um grafo G, então todos os vizinhos de suas pontas estão em P.*

Demonstração. Seja $P = (v_0, v_1, \ldots, v_n)$ um caminho maximal em um grafo G.

Se v_0 tivesse um vizinho $u \notin V(P)$, então $(u, v_0, v_1, \ldots, v_n)$ seria caminho em G e P não seria maximal.

Pelo mesmo argumento, v_n não pode ter vizinhos fora de P. □

Teorema 15. *Se P é um caminho direcionado maximal em um grafo direcionado G, então*

- *todos os vizinhos de entrada de seu vértice inicial estão em P, e*
- *todos os vizinhos de saída seu vértice final estão em P.*

Demonstração. Exercício 58. □

Teorema 16. *Todo passeio de tamanho mínimo entre dois vértices de um grafo é um caminho.*

Demonstração. Seja G um grafo e seja $P = (v_0, v_1, \ldots, v_{n-1}, v_n)$ um passeio de tamanho mínimo de v_0 a v_n em G.

Se P não fosse um caminho, teríamos algum vértice repetido em G, isto é,

$$v_i = v_j, \text{ para algum } 0 \leq i < j \leq n,$$

e neste caso, o passeio $(v_0, \ldots, v_{i-1}, v_i, v_{j+1}, \ldots, v_n)$ seria um passeio de v_0 a v_n em G de tamanho $|P| - (j - i + 1) < |P|$. □

Teorema 17. *Todo passeio direcionado de tamanho mínimo entre dois vértices de um grafo direcionado é um caminho direcionado.*

Demonstração. Exercício 59 □

Um *caminho mínimo em um grafo (direcionado)* G é um caminho (direcionado) de tamanho mínimo entre suas pontas em G.

Um *caminho mínimo em um grafo ponderado* é um caminho de peso mínimo no grafo.

Observe que a distância de u a u em G é o tamanho (peso) de um caminho mínimo em G.

Teorema 18. *Todo segmento de caminho mínimo em um grafo G é caminho mínimo em G.*

Demonstração. Exercício 54 □

Teorema 19. *Todo segmento de caminho mínimo em um grafo direcionado G é caminho mínimo em G.*

Demonstração. Exercício 60 □

Tópico 14

Ciclos

Exercícios 61 a 69

Um *ciclo* é um passeio fechado de tamanho maior ou igual a 3 cujos vértices são todos distintos, exceto pelas pontas.

É comum chamar o subgrafo induzido por um ciclo também de "ciclo" ou "ciclo" e indicar um ciclo de n vértices por C_n.

Um grafo sem ciclos é chamado de *acíclico*

Um grafo direcionado sem ciclos é chamado de *grafo direcionado acíclico*

A *cintura* ("girth") de um grafo G é o tamanho um ciclo de tamanho mínimo em G e é denotada por $\gamma(G)$.

Convenciona-se que $\gamma(G) = \infty$ quando G é acíclico.

Uma *corda* em um ciclo C é uma aresta ligando dois vértices que não são vizinhos em $G[C]$.

Teorema 20. *Um grafo tem ciclo se e somente se tem caminhos distintos com os mesmos início e fim.*

Demonstração. Seja G um grafo.

($\Rightarrow$) Se G tem um ciclo $C = (v = v_0, v_1, \ldots, v_n = v)$, Então,

$$(v = v_0, v_1), \ \mathsf{e}$$
$$(v = v_n, v_{n-1}, \ldots, v_1)$$

são caminhos distintos de v a v_1 em G.

($\Leftarrow$) Suponha, por outro lado, que

$$\begin{aligned} P &= (u = v_0, \ldots, v_n = v), \text{ e} \\ P' &= (u = v'_0, \ldots, v'_{n'} = v), \end{aligned}$$

são caminhos distintos de u a v em G, e sejam

m o índice do primeiro vértice em que P e P' diferem, isto é,

$$m := \min \{i \in [0..n] \mid v_i \neq v'_i\}.$$

l o índice do primeiro vértice de P, depois de v_m em $V(P')$, isto é,

$$l := \min \{i \in [m..n] \mid v_i \in V(P')\},$$

l' o índice de v_l em P'

Então

$$C = (v_{m-1}, v_m, \ldots, v_l = v'_{l'}, v'_{l'-1}, \ldots, v'_m, v'_{m-1} = v_{m-1})$$

é um ciclo em G.

□

Corolário 21. *Um grafo é bipartido se e somente se não tem ciclo ímpar.*

Demonstração. ($\Rightarrow$) Seja G um grafo bipartido e seja $C = (v_0, \ldots, v_n = v_0)$ um ciclo em G.

Então (v_0, v_1) e $(v_0, v_{n-1}, \ldots, v_1)$ são caminhos de v_0 a v_1 em G e tem ambos tamanho ímpar (T. 13). Consequentemente C tem tamanho par.

($\Leftarrow$) Seja G um grafo sem ciclo ímpar. Sem perda de generalidade, podemos assumir que G não é vazio e é conexo.

Seja $v \in V(G)$ e seja

$$X := \{u \in V(G) \mid d_G(v, u) \text{ é par}\}$$

Se $\{X, V(G) - X\}$ não fosse bipartição de G, haveria aresta entre dois vértices $x, x' \in X$ (ou $x, x' \in V(G) - X$).

Sejam P e P' caminhos mínimos x a v e de x' a v, respectivamente, em G. Observe que ambos tem a mesma paridade.

Seja w o primeiro vértice comum a P e P'.

Considere então o ciclo $C = xPw \cdot wP'x' \cdot (x', x)$.

Observe que $|vPw| = |vP'w| = d_G(v, w)$ pois ambos são segmentos de caminhos mínimos (T.18).

Consequentemente, $|xPw|$ e $|wP'x'|$ tem mesma paridade e C tem de tamanho ímpar, contradizendo a hipótese inicial.

Logo $\{X, V(G) - X\}$ é bipartição de G.

□

Corolário 22. *Todo grafo acíclico é bipartido.*

Tópico 15

Árvores

Exercícios 70 a 76

Uma *árvore* é um grafo acíclico conexo.

Uma *floresta* é um grafo em que cada componente é uma árvore, isto é, é um grafo acíclico.

Corolário 23. *Um grafo é árvore se e somente se admite um único caminho entre cada par de seus vértices.*

Demonstração. Imediato a partir do T. 20 □

Denotamos por uTv o único caminho de u a v na árvore T.

Se F é uma floresta e os vértices u e v estão no mesmo componente T de F, usamos uFv como sinônimo de uTv.

Um vértice de grau 1 em uma floresta é chamado de *folha*.

Teorema 24. *Toda árvore não trivial tem (pelo menos) duas folhas.*

Demonstração. Seja T uma árvore não trivial e seja $P = (v_0, \ldots, v_n)$ um caminho maximal em T.

Se v_0 tivesse outro vizinho em T além de v_1, este vizinho teria que estar em P (L. 14) e neste caso T teria um ciclo (T. 20).

Logo, v_0 é folha de T.

Por argumento análogo concluímos que v_n também é folha de T. □

Teorema 25. *Toda árvore de n vértices tem $n - 1$ arestas.*

Demonstração. Vamos provar que se G é árvore de n vértices, então G tem $n-1$ arestas, por indução em $|V(G)|$.

Base: Se $n = 1$ então G é um grafo trivial e tem $0 = n - 1$ arestas.

HI: Se G é árvore com a vértices então G tem $a - 1$ arestas.

Passo: Vamos provar que se G é árvore com $a + 1$ vértices, então G tem a arestas.

Seja G uma árvore com $a + 1$ vértices.

Como G é árvore e não é trivial, então (C. 24) G tem uma folha v.

Como G é conexo e v é folha de G, então $G - v$ é conexo.

Como G é acíclico, então $G - v$ é acíclico.

Então $G - v$ é árvore e como

$$|V(G - v)| = |V(G)| - 1 = (a + 1) - 1 = a,$$

então, pela HI, $G - v$ tem $|V(G - v)| - 1 = a - 1$ arestas.

Como v é folha, então

$$|E(G - v)| = |E(G)| - 1,$$

ou seja

$$|E(G)| = |E(G - v)| + 1 = (a - 1) + 1 = a,$$

isto é, G tem a arestas.

□

Teorema 26. *Todo grafo conexo com n vértices e $n - 1$ arestas é árvore.*

Demonstração. Exercício 70. □

Corolário 27. *Um grafo com n vértices é árvore se e somente se é conexo e tem $n - 1$ arestas.*

Corolário 28. *O grafo G é floresta se e somente se*

$$|E(G)| = |V(G)| - |\mathcal{C}(G)|.$$

Demonstração. Exercício 71 □

Teorema 29. *Se T é uma árvore onde u e v não são vizinhos, então $T + \{u, v\}$ tem um único ciclo.*

Demonstração. Seja T uma árvore e sejam $u, v \in V(T)$ tais que $\{u, v\} \notin E(T)$.

Observe que $uTv \cdot (v, u)$ é um ciclo em $T + \{u, v\}$.

Suponha agora que exista um outro ciclo C em $T + \{u, v\}$. Então $\{u, v\} \in E(C)$ pois senão C seria ciclo em T.

Neste caso, $G[C] - \{u, v\}$ seria conexo e existiria caminho P de u a v em $G[C] - \{u, v\}$. Neste caso, P seria um caminho de u a v em T distinto de uTv, e daí (T.20) T teria ciclo. □

Se T é uma árvore e $u, v \in V(T)$ são tais que $\{u, v\} \notin E(T)$, o ciclo $T[uTv \cdot (v, u)]$ é chamado de *ciclo fundamental* de $\{u, v\}$ com relação a T.

Uma *árvore (floresta) geradora* de um grafo G é um subgrafo gerador de G que é árvore (floresta).

Uma *árvore enraizada* é um par (T, r) onde T é uma árvore e r é um vértice de T, chamado *raiz* de T.

O *nível* do vértice v na árvore enraizada (T, r) é a distância de r a v em T, isto é,

$$L_{T,r}(v) := d_T(r, v).$$

Tópico 16

Arborescências

Exercícios 77 a **??**

Uma *arborescência* (*"branching"*, *"arborescence"*) é um grafo direcionado conexo com uma única fonte em que todos os demais vértices tem grau de entrada 1.

A fonte de uma arborescência T é chamada de *raiz* de T e é denotada $r(T)$.

Se T é uma arborescência e

- $(u, v) \in V(T)$, dizemos que u é *pai* de v e que v é *filho* de u em T.
- existe caminho direcionado de u a v em T, dizemos que u é *ancestral* de v e que v é *descendente* de u em T, o que é denotado por $u \leq_T v$. Se $u \neq v$ dizemos ainda que u é *ancestral próprio* de v e que v é *descendente próprio* de u em T o que é denotado por $u <_T v$.
- $v \in V(T)$ não tem filhos, dizemos que v é uma *folha* de T.

Teorema 30. *O grafo subjacente de uma árborescência é uma árvore.*

Demonstração. Exercício 78 □

Corolário 31. *Se T é uma arborescência e u é ancestral de v em T diferente do pai de v, então $T + (v, u)$ tem um único ciclo direcionado.*

Se T é uma arborescência e u é ancestral de v em T, o único ciclo direcionado de $T + (v, u)$ é chamado de *ciclo fundamental* de (v, u) com relação a T.

Uma *floresta direcionada* (*"branching"* para alguns autores) é um grafo direcionado em que cada componente é uma arborescência.

Observe que um componente C de um grafo direcionado G é um subgrafo C de G tal que o grafo subjacente de C é componente do grafo subjacente de G.

Se T é uma floresta direcionada geradora de um grafo direcionado G, um arco $(u, v) \in A(G) - A(T)$ pode ser um

laço: se $u = v$,

arco de retorno: ("backward") se $u >_T v$,

arco de avanço: ("forward") se $u <_T v$,

arco cruzado: ("cross") se $u \not<_T v$.

Seja (T, r) uma árvore enraizada. A arborescência que resulta da orientação das arestas de T do vértice mais próximo para o mais distante de r é chamada de *arborescência induzida por r em T*.

Do mesmo modo, se T é uma arborescência, a árvore enraizada $(S(T), r(T))$ é chamada de *árvore enraizada subjacente a T*.

Em função desta correspondência, a terminologia e a notação acima serão usados indistintamente para árvores enraizadas e arborescências.

Tópico 17

Cortes e Conectividade por Arestas

Exercícios 80 a 82

Um conjunto K de arestas de G é um *corte* de arestas se o grafo $G - K$ tem mais componentes que G, isto é, se

$$|\mathcal{C}(G - K)| > |\mathcal{C}(G)|.$$

Um grafo conexo não trivial e não completo G é k–aresta conexo se não tem corte de arestas de tamanho menor que k.

A *aresta–conexidade* de G é o tamanho do menor corte de arestas de G e é denotada $\lambda(G)$, isto é,

$$\lambda(G) := \min\{k \in \mathbb{N} \mid G \text{ tem corte de arestas de tamanho } k\}.$$

Uma aresta $e \in E(G)$ é uma *aresta de corte* em G se o conjunto $\{e\}$ é um corte em G. Uma aresta de corte em G também é chamada de *ponte* de G.

Dizemos que o corte K *separa* os conjuntos de vértices X e Y em um grafo G se os conjuntos X e Y estão em componentes diferentes de $G - K$.

Teorema 32. *Para todo $X \subseteq V(G)$, a fronteira de X é vazia ou é corte em G.*

Demonstração. Seja G um grafo e seja $X \subseteq V(G)$ tal que $\partial_G(X) \neq \emptyset$.

Seja então $\{x, y\} \in X$ com $x \in X$ e $y \in V(G) - X$.

Se $\partial_G(X)$ não fosse corte em G, então x e y estariam no mesmo componente de $G - \partial_G(X)$ e haveria caminho $P = (x = v_0, \ldots, v_n = y)$ em $G - \partial_G(X)$. Seja então m o índice do primeiro vértice em P fora de X, isto é,

$$m := \min \{i \in [0..n] \mid v_i \notin X\}.$$

Neste caso, temos $v_{m-1} \in X$ e $v_m \notin X$ e, portanto, $\{v_{m-1,v_m}\} \in \partial_G(X)$, o que não pode ser já que P é caminho em $G - \partial_G(X)$. □

As arestas de $\partial_G(X)$ ligam $G[X]$ a $G - X$.

Teorema 33. *Seja G um grafo e seja $X \subseteq V(G)$. Toda aresta de G está em exatamente um dentre os conjuntos $E(G[X])$, $\partial_G(X)$ e $E(G - X)$.*

Demonstração. Seja G um grafo e seja $X \subseteq V(G)$.

É imediato a partir das suas respectivas definições que os conjuntos $E(G[X])$, $\partial_G(X)$ e $E(G - X)$ são dois a dois disjuntos entre si.

Além disso toda aresta que não tem ambas as pontas em X ou em $V(G) - X$ tem uma ponta em cada um destes conjuntos, ou seja, toda aresta que não está nem em $G[X]$ nem em $G - X$ está em $\partial_G(X)$. □

Corolário 34. *O grafo H é componente do grafo G se e somente se H é subgrafo conexo de G com $\partial_G(V(H)) = \emptyset$.*

Demonstração. Seja G um grafo e seja H um subgrafo de G.

Se H é um componente de G, então $\partial_G(V(H)) = \emptyset$.

Se, por outro lado, H é conexo, sejam $h \in V(H)$ e $v \in V(G) - V(H)$.

Para que $G[V(H) \cup \{v\}]$ seja conexo é preciso haver caminho de h a v em G. Pelo argumento da prova do Teorema 32, neste caso $\partial_G(V(H)) \neq \emptyset$

Consequentemente, se H é subgrafo conexo de G com $\partial_G(V(H)) = \emptyset$ então não existe $v \in V(G) - V(H)$ tal que $G[V(H) \cup \{v\}]$ seja conexo.

Então H é um subgrafo maximal conexo de G ou seja, H é componente de G. □

O seguinte resultado generaliza o Teorema 1

Teorema 35. *Para todo $X \subseteq V(G)$,*

$$\sum_{v \in X} \delta_G(v) = 2|E(G[X])| + |\partial_G(X)|.$$

Demonstração. Seja G um grafo e seja $X \subseteq V(G)$. Para cada $v \in X$ temos

$$\delta_G(v) = |\Gamma_G(v) \cap X| + |\Gamma_G(v) - X| = \delta_{G[X]}(v) + |\Gamma_G(v) - X|,$$

ou seja,

$$|\Gamma_G(v) - X| = \delta_G(v) - \delta_{G[X]}(v).$$

Então,

$$\begin{aligned}\sum_{v \in X} |\Gamma_G(v) - X| &= \sum_{v \in X} (\delta_G(v) - \delta_{G[X]}(v)) \\ &= \sum_{v \in X} \delta_G(v) - \sum_{v \in X} \delta_{G[X]}(v) \\ &\overset{\text{T. 1}}{=} \sum_{v \in X} \delta_G(v) - 2|E(G[X])|,\end{aligned}$$

ou seja,

$$\sum_{v \in X} \delta_G(v) = 2|E(G[X])| + \sum_{v \in X} |\Gamma_G(v) - X| = 2|E(G[X])| + |\partial_G(X)|.$$

□

Corolário 36. *Os vértices de um grafo G tem todos grau par se e somente se $|\partial(X)|$ é par para todo $X \subseteq V(G)$.*

Demonstração. Seja G um grafo.

Se $|\partial(X)|$ é par para todo $X \subseteq V(G)$, então $|\partial(v)| = \delta_G(v)$ é par para para todo $v \in V(G)$.

Se, pelo outro lado, todos os vértices de G tem grau par, por consequência direta do Teorema 35, $|\partial(X)|$ é par para todo $X \subseteq V(G)$. □

Tópico 18

Cortes e Conectividade por Vértices

Exercícios 83 a 90

Um conjunto K de vértices de G é um *corte* de vértices se o grafo $G - K$ tem mais componentes que G, isto é, se

$$|\mathcal{C}(G - K)| > |\mathcal{C}(G)|.$$

Um grafo conexo não trivial G é *k–vértice conexo* se não tem corte de vértices de tamanho menor que k.

Um vértice $v \in V(G)$ é um *vértice de corte* em G se o conjunto $\{v\}$ é um corte em G. Um vértice de corte em G também é chamado *articulação* de G.

Dizemos que o corte K *separa* os conjuntos de vértices X e Y em um grafo G se os conjuntos X e Y estão em componentes diferentes de $G - K$.

A *vértice–conexidade* de G é o tamanho do menor corte de arestas (vértices) de G e é denotada $\kappa(G)$, isto é,

$$\kappa(G) := \begin{cases} |V(G)| - 1, & \text{se } G \text{ é completo,} \\ \min\{k \in \mathbb{N} \mid G \text{ tem corte de vértices de tamanho } k\}, & \text{se } G \text{ não é completo.} \end{cases}$$

Um *bloco* em um grafo G é um subgrafo maximal de G sem vértices de corte.

Tópico 19

Trilhas

Exercícios 92 a 102

19.1 Trilhas

Uma *trilha* é um passeio cujas arestas são todas distintas.

Teorema 37. *Se T é uma trilha sobre um grafo G, então o grau de todo vértice interno de T em $G[T]$ é par.*

Demonstração. Seja $T = (v_0, \ldots, v_n)$ uma trilha em um grafo G. Para cada $k \in [1..n-1]$ temos duas arestas distintas de $E(v_{k-1}, v_k, v_{k+1}) \cap \partial_G(v_k)$, a saber, $\{v_{k-1}, v_k\}$ e $\{v_k, v_{k+1}\}$.

---

ag: Acho que deveríamos também provar que estas duas arestas não se repetem em outras ocorrencias de v_k na trilha. Acho que basta afirmar que v_{k-1}, v_k e v_{k+1} são todos distintos. Ou que se $v_i = v_j$, com $i \neq j$, temos que $|i - j| > 1$.

Assim, se o vértice v ocorre l vezes como vértice interno em T, temos $\delta_{G[T]}(v) = 2l$. □

Corolário 38. *Se T é uma trilha fechada sobre um grafo G, então o grau de todo vértice em $G[T]$ é par.*

19.2 Trilhas e Grafos Eulerianos

Uma *trilha euleriana* em G é uma trilha que passa por todas as arestas de G.

Um grafo é *euleriano* se é trivial ou se tem trilha euleriana fechada.

Corolário 39. *Num grafo euleriano todos os vértices tem grau par.*

O seguinte algoritmo é conhecido como Algoritmo de Fleury.

TrilhaEuleriana(G)

Entrada: um grafo conexo G
Saída : uma trilha em G
$x \leftarrow u \leftarrow$ vértice de G
$T \leftarrow (u)$
$F \leftarrow G$
Enquanto $\partial_F(x) \neq \emptyset$
  Se *toda aresta em* $\partial_F(x)$ *é de corte em* F
    $\{x, y\} \leftarrow$ aresta de $\partial_F(x)$
  Senão
    $\{x, y\} \leftarrow$ aresta de $\partial_F(x)$ que não é de corte em F
  remova a aresta $\{x, y\}$ de F
  acrescente y ao final de T
  $x \leftarrow y$
Devolva T

Teorema 40. *Se G é um grafo conexo cujos vértices todos tem grau par, então TrilhaEuleriana(G) é uma trilha euleriana em G.*

Demonstração. Seja G um grafo conexo cujos vértices todos tem grau par e considere a execução de TrilhaEuleriana(G).

Observe que ao início de cada iteração do laço, sempre é verdade que

1. T é trilha de u a x em G, e portanto,
2. exceto quando $u = x$, os vértices u e x são os únicos vértices de grau ímpar em $G[T]$ (T. 37);
3. $F = G - E(T)$ e, consequentemente,
4. $\partial_F(X) = \partial_G(X) - E(T)$, para todo $X \subseteq V(G)$.

O laço termina (em no máximo $|E(G)|$ iterações), pois a cada iteração uma aresta é retirada de F.

Para cada $i \geq 1$, vamos definir F_i, x_i, y_i e T_i como sendo os valores das variáveis F, x, y e T, respectivamente, ao início da i-ésima iteração do laço. Denotamos ainda por k o número de iterações, isto é, a última iteração é a k-ésima iteração, na qual a condição $\partial_F(x) \neq \emptyset$ deixa de ser satisfeita, isto é, $\partial_{F_k}(x_k) = \emptyset$

Como $\partial_{F_k}(x_k) = \emptyset$ e $\partial_{F_k}(x_k) = \partial_G(x_k) - E(T_k)$, concluímos que $\partial_G(x_k) \subseteq E(T_k)$, isto é, todas as arestas incidentes em x_k estão em T_k.

Como $\partial_G(x_k) \subseteq E(T_k)$ e $\delta_G(x_k) = |\partial_G(x_k)|$, concluímos que $\delta_{G[T_k]}(x_k)$ é par. Como $G[T_k]$ não pode ter um único vértice de grau ímpar, concluímos que $x_k = u$ e daí, que T_k é trilha fechada.

Resta provar que $E(T_k) = E(G)$ ou, equivalentemente, que $E(F_k) = \emptyset$, ou ainda, que todos os vértices de G tem grau zero em F_k.

Suponha que não e seja X o conjunto dos vértices de G com grau positivo em F_k.

Observe que todos os vértices de X tem grau par em F_k, pois $F_k = G - E(T_k)$ e todos os vértices de G tem grau par em G bem como todos os de $G[E(T_k)]$ tem grau par em $G[E(T_k)]$. Consequentemente, todo vértice de X tem grau par (e pelo menos 2) em F_k. Além disso, sabemos também (C. 36) que $|\partial_G(X)|$ é par.

Por um lado, $\partial_{F_k}(X) = \emptyset$, pois todas as arestas de F_k tem ambas as pontas em X.

Por outro lado, $\partial_G(X) \neq \emptyset$ por causa do C. 34, já que

1. $X \neq V(G)$, uma vez que todas as arestas de $\partial_G(u)$ estão em T e, portanto, $u \notin X$;

2. G é conexo e (por hipótese) $X \neq \emptyset$,

Considere agora a t-ésima iteração do laço, na qual é retirada a última aresta de $\partial_G(X)$ pelo algoritmo, a saber, a aresta $\{x_t, y_t\}$, sendo $x_t \in X$.

Observe que $F_t[X] = F_k[X]$ (já que a trilha não volta a passar por vértices de X e, portanto, não são mais retiradas arestas de $F_t[X]$) e todo vértice de X tem grau pelo menos 2 em F_k e daí tem grau pelo menos 2 em $F_t[X]$.

Pelo algoritmo, $\{x_t, y_t\}$ é aresta de corte de F_t.

Mas isso não pode ser pois x_t tem outro vizinho z em F_t e a aresta $\{x_t, z\}$ não é de corte em F_t (veja Exercício 102). □

Corolário 41. *Um grafo conexo cujos vértices todos tem grau par é euleriano.*

Teorema 42 (Teorema de Euler (Hierholzer, 1873))**.** *Um grafo conexo é euleriano se e somente se todos os seus vértices tem grau par.*

Corolário 43. *Um grafo conexo tem trilha euleriana aberta se e somente se tem exatamente dois vértices de grau ímpar.*

Demonstração. Exercício 92. □

Tópico 20

Ciclos Hamiltonianos

Exercícios 93 a 101

Um *ciclo hamiltoniano* em um grafo é um ciclo gerador do grafo, ou seja, um ciclo que passa por todos os vértices do grafo.

Um grafo é *hamiltoniano* se tem ciclo hamiltoniano.

Um *caminho hamiltoniano* em um grafo é um um caminho gerador do grafo, ou seja, caminho que passa por todos os vértices do grafo

Teorema 44 (Karp, 1972)**.** *Determinar se um grafo dado tem ciclo hamiltoniano é um problema $\mathcal{NP}$–Difícil.*

Corolário 45. *Determinar o tamanho do maior ciclo de um grafo é um problema $\mathcal{NP}$-difícil.*

Demonstração. Exercício 93. □

Corolário 46. *Decidir se um grafo tem caminho hamiltoniano é um problema $\mathcal{NP}$–difícil.*

Demonstração. Exercício 95 □

Corolário 47. *Determinar um caminho de tamanho máximo em um grafo é um problema $\mathcal{NP}$–difícil.*

Demonstração. Exercício 96 □

20.1 Grafos Direcionados e Ponderados

Um *grafo direcionado hamiltoniano* é um grafo direcionado que tem ciclo hamiltoniano direcionado.

Corolário 48. *O problema de decidir se um grafo direcionado é hamiltoniano é $\mathcal{NP}$–Difícil.*

Demonstração. Exercício 97. □

O problema do Caixeiro Viajante é o problema de encontrar um ciclo hamiltoniano de peso mínimo em um grafo direcionado completo com pesos.

Corolário 49. *O problema do Caixeiro Viajante é $\mathcal{NP}$–difícil.*

Demonstração. Exercício 98. □

Tópico 21

Busca em Grafos

Algoritmos que processam cada vértice e cada aresta de um grafo G dado.

A ideia é particionar os vértices do grafo em 3 partes (V_0, V_1, V_2) tendo um vértice inicial r como ponto de referência. Inicialmente os vértices estão todos em V_0. O vértice r é colocado em V_1 e um a um dos demais vértices vão sendo deslocados para V_1 (são "procesados"), caso tenham um vizinho já em V_1, e para V_2 caso estejam em V_1, não tenham mais vizinhos em V_0 e todas as suas arestas tenham sido processadas.

As arestas são processadas a medida que são encontradas.

Se o grafo não é conexo, é necessário fazer outra busca a partir de outro vértice inicial (ainda em V_0).

Para cada $v \in V(G)$, a variável $v.\mathsf{estado}$ assume os valores 0, 1 ou 2, conforme v esteja em V_0, V_1 ou V_2, respectivamente. A variável $v.\mathsf{pai}$ é o indice do vizinho de v que já estava em V_1 e foi usado para "puxar" v de V_0 para V_1.

Note que V_1 é representado de forma explícita também, pois é necessário

escolher um vértice por vez deste conjunto.

Busca(G)

```
Para v ∈ V(G)
    v.estado ← 0
Para v ∈ V(G)
    Se v.estado = 0
    Busca(G, v)
```

Busca(G, r)

```
V_1 ← ∅
processe r
r.pai ← Λ
acrescente r a V_1
r.estado ← 1
Enquanto V_1 ≠ ∅
    v ← vértice em V_1
    Se não há "próximo" vértice em Γ_G(v)
        retire v de V_1
        v.estado ← 2
    Senão
        w ← "próximo" vértice em Γ_G(v)
        Se w.estado ≠ 0
            Se {v, w} não foi processada
                processe {v, w}
        Senão
            processe {v, w}
            processe w
            w.pai ← v
            acrescente w a V_1
            w.estado ← 1
```

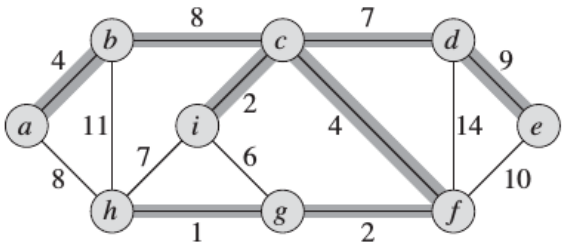


Figura 21.1: (Cormen, Leiserson, Rivest, and Stein, 2009, fig 23.1, p. 625)

Fazendo

$$\begin{aligned}
G_r &:= \text{componente de } G \text{ que contém } r,\\
V_0 &:= \{v \in V(G) \mid v.\mathsf{estado} = 0\},\\
V_1 &:= \{v \in V(G) \mid v.\mathsf{estado} = 1\},\\
V_2 &:= \{v \in V(G) \mid v.\mathsf{estado} = 2\},\\
H &:= G[V_1 \cup V_2],\\
T &:= G[\{(v.\mathsf{pai}, v) \mid v.\mathsf{pai} \neq \Lambda\}] + r,
\end{aligned}$$

temos

1. ao início de cada iteração,
   (a) (T, r) é uma árvore enraizada geradora de H, e portanto,
   (b) H é um subgrafo conexo de G_r.
   (c) $\{V_0, V_1, V_2\}$ é uma partição de $V(G)$,
   (d) os vértices e as arestas de T estão todos processados,
   (e) as arestas de $G[V_2]$ estão todas processadas,
   (f) $\partial_G(V(H)) = \{\{v_0, v_1\} \in E(G) \mid v_0 \in V_0 \text{ e } v_1 \in V_1\}$.
2. ao final do laço
   (a) $V_1 = \emptyset$ e, portanto,
   (b) $\partial_G(V(H)) = \emptyset$ e, portanto,
   (c) como H é conexo, então (C. 34) H é componente de G, e
   (d) como $r \in V(H)$, então $H = G_r$ e,

(e) (T, r) é uma árvore enraizada geradora de G_r,

(f) todos vértices e arestas de G_r foram processados.

3. a cada volta do laço

   (a) um vértice $v \in V_1$ é escolhido e

      i. um novo vizinho w de v é escolhido, ou

      ii. v sai de V_1 e vai para V_2, de forma que

   (b) cada vértice é escolhido como v exatamente $\delta_G(v) + 1$ vezes e

   (c) o laço é executado exatamente

$$\sum_{v \in V(G_r)} (\delta_G(v) + 1) = 2|E(G_r)| + |V(G_r)|$$

   vezes.

Teorema 50. *Sejam G um grafo e $r \in V(G)$. Seja ainda G_r o componente de G que contém o vértice r.*

O Algoritmo Busca(G, r) *devolve uma árvore enraizada (T, r) geradora de G_r, e processa todos os vértices e arestas de G_r. Além disso, o laço do algoritmo é executado $2|E(G_r)| + |V(G_r)|$ vezes.*

Diferentes árvores geradoras de G_r são construídas, dependendo dos vértices escolhidos em cada iteração.

21.1 Componentes Conexos

```
Componentes(G)
  Para cada v ∈ V(G)
      v.estado ← 0
      v.componente ← 0
  c ← 0
  Para cada v ∈ V(G)
      Se v.estado = 0
          v.componente ← ++c
          Componente(G, v)
```

```
Componente(G, r)
  V ← ∅
  acrescente r a V
  r.estado ← 1
  Enquanto V ≠ ∅
      retire um vértice v de V
      Para cada w ∈ Γ_G(v)
          Se w.estado = 0
              w.componente ← r.componente
              acrescente w a V
              w.estado ← 1
      v.estado ← 2
```

Teorema 51. *É possível determinar os componentes de um grafo de n vértices e m arestas em tempo $O(n + m)$.*

Demonstração. Basta implementar o conjunto V no Algoritmo Componentes(G, r) por meio de uma lista encadeada. □

Tópico 22

Busca em Largura

Exercícios 103 a 114

Uma *busca em largura* (*breadth–first search* ou *BFS*) em um grafo a partir de um vértice r é uma busca na qual os vértices são processados em ordem não decrescente de distância de r, isto é, todos os vértices a distância d de r são processados antes de todos os vértices a distância $d + 1$ de r, para

todo $d \geq 0$.

BuscaLargura(G)

Para $v \in V(G)$
    v.estado $\leftarrow 0$
Para $v \in V(G)$
    Se v.*estado* $= 0$
        BuscaLargura(G, v)

BuscaLargura(G, r)

$V \leftarrow$ fila vazia
processe r
r.pai $\leftarrow \Lambda$
enfile r em V
r.estado $\leftarrow 1$
Enquanto V *não está vazia*
    desenfile um vértice v de V
    Para *cada* $w \in \Gamma_G(v)$
        Se w.*estado* $= 1$
            processe $\{v, w\}$ (como aresta fora da árvore)
        Senão, se w.*estado* $= 0$
            processe w
            processe $\{v, w\}$ (como aresta de árvore)
            w.pai $\leftarrow v$
            enfile w em V
            w.estado $\leftarrow 1$
    v.estado $\leftarrow 2$

Teorema 52. *Se F é a floresta direcionada resultante de uma busca em largura sobre um grafo G, então toda aresta fora de F é cruzada com relação a F.*

Demonstração. Exercício 112 □

22.1 Distâncias

Teorema 53. *Se T é a arborescência resultante de uma busca em largura a partir do vértice r em um grafo conexo G, então*

$$d_T(r, v) = d_G(r, v), \text{ para todo } v \in V(G).$$

Demonstração. Exercício 113 □

CaminhosMinimos(G, r)

$V \leftarrow$ fila vazia
r.dist $\leftarrow 0$
enfile r em V
r.estado $\leftarrow 1$
Enquanto $V \neq \emptyset$
    desenfile um vértice v de V
    Para *cada* $w \in \Gamma_G(v)$
        Se w.*estado* $= 0$
            w.pai $\leftarrow v$
            w.dist $\leftarrow w$.pai.dist $+ 1$
            enfile w em V
            w.estado $\leftarrow 1$
    v.estado $\leftarrow 2$

Corolário 54. *Se (T, r) é a árvore enraizada resultante de uma busca em largura em um grafo conexo G, então rTv é um caminho mínimo em G para todo $v \in V(G)$.*

Demonstração. Exercício 106 □

Uma *árvore de caminhos mínimos* do grafo G é uma árvore enraizada (T, r) onde rTv é um caminho mínimo em G para todo $v \in V(G)$.

Corolário 55. *É possível computar uma árvore de caminhos mínimos geradora de um grafo conexo com m arestas em tempo $O(m)$.*

Corolário 56. *É possível computar as distâncias e os caminhos mínimos entre todos os pares de vértices de um grafo com n vértices e m arestas em tempo $O(nm)$.*

Corolário 57. *É possível computar o diâmetro de um grafo conexo com n vértices e m arestas em tempo $O(nm)$.*

22.2 Bipartição

Teorema 58. *Se (T, r) é a árvore enraizada resultante de uma busca em largura sobre um grafo conexo G, então*

$$|d_G(r, u) - d_G(r, v)| \leq 1, \text{ para todo } \{u, v\} \in E(G - T).$$

Demonstração. Exercício 107. □

Teorema 59. *Seja (T, r) a árvore enraizada resultante de uma busca em largura sobre um grafo conexo G. O grafo G é bipartido se e somente se $d(r, u)$ e $d(r, v)$ tem paridades diferentes para toda aresta $\{u, v\} \in E(G - T)$.*

Demonstração. Exercício 108. □

Corolário 60. *É possível decidir se um grafo de n vértices e m arestas é bipartido e, em caso positivo, computar uma sua bipartição em tempo $O(n + m)$.*

Tópico 23

Árvores Geradoras Mínimas e o Algoritmo de Jarník–Prim

Exercícios 115 a 115

Uma *árvore geradora mínima* em um grafo conexo com pesos (G, w) é uma árvore geradora T de G de peso mínimo.

O problema da Árvore Geradora Mínima é o problema de encontrar uma árvore geradora mínima em um grafo conexo com pesos dado.

$\mathsf{AGM}_P(G, w)$

$T \leftarrow$ grafo vazio
acrescente um vértice r de G a T
Enquanto $\partial_G(V(T)) \neq \emptyset$
    escolha uma aresta e em $\partial_G(V(T))$ de peso mínimo
    acrescente a aresta e ao grafo T
Devolva T

Teorema 61. *Ao início de cada iteração do Algoritmo $\mathsf{AGM}_P(G, w)$, o grafo T é subárvore de uma árvore geradora mínima de G.*

Demonstração. Seja (G, w) um grafo conexo com pesos e considere uma execução de $\mathsf{AGM}_P(G, w)$. Seja também T^* uma árvore geradora mínima de G e considere a última iteração para a qual T e $T' := T^*[V(T)]$ coincidem.

Então existe uma única aresta e em $E(T) - E(T')$ e, do mesmo modo, tem que existir uma aresta e' do ciclo fundamental de $T^* + e$ em $\partial_G(V(T))$. Considere então a árvore $T'' := T^* + e - e'$.

Pela escolha de e temos $w(e) \leq w(e')$ e daí

$$w(T'') = w(T^* + e - e') = w(T^*) + w(e) - w(e') \leq w(T^*).$$

Como T^* é árvore geradora mínima de G a conclusão é que T'' também é árvore geradora mínima de G.

Finalmente, como T é subgrafo de T'', é verdade que T é subárvore de uma árvore geradora mínima de G. □

Corolário 62. *O Algoritmo* $\mathsf{AGM}_P(G, w)$ *devolve uma árvore geradora mínima de um grafo conexo com pesos* (G, w).

```
AGM_P(G, w)
Para cada v ∈ V(G)
    v.estado ← 0
V ← ∅
r ← um vértice de G
r.pai ← Λ
r.custo ← 0
acrescente r a V
r.estado ← 1
Enquanto V ≠ ∅
    retire de V um vértice v com v.custo mínimo
    Para cada u ∈ Γ_G(v)
        Se u.estado = 1
            Se w({u, v}) < u.custo
                u.pai ← v
                u.custo ← w({u, v})
        Senão, se u.estado = 0
            acrescente u a V
            u.estado ← 1
            u.pai ← v
            u.custo ← w({u, v})
    v.estado ← 2
```

Corolário 63. *É possível computar uma árvore geradora mínima de um grafo conexo com n vértices e m arestas em tempo $O(m \log n)$.*

Demonstração. Basta implementar o conjunto V do algoritmo $\mathsf{AGM}_P(G, w)$ por meio de uma fila de prioridades.

O trecho do algoritmo anterior ao laço externo pode ser executado em tempo $O(n)$.

Cada iteração retira um elemento v de V, altera o custo de um vizinho u de v e (em alguns casos) acrescenta u a V. Cada iteração executa, portanto, em tempo $O(\log n)$.

O número de iterações agregadas entre os laços externo e interno é $2m$.

Como $|V|$ nunca ultrapassa o número de vértices n do grafo, o tempo de execução total é $O(m \log n)$. □

É possível "enxugar" o algoritmo

```
AGM'_P(G, w)
Para cada v ∈ V(G)
    v.custo ← ∞
V ← V(G)
r ← um vértice de G
r.pai ← λ
r.custo ← 0
Enquanto V ≠ ∅
    retire de V um vértice v com v.custo é mínimo
    Para cada u ∈ Γ_G(v)
        Se w({u, v}) < u.custo
            u.custo ← w({u, v})
            u.pai ← v
```

Tópico 24

Caminhos Mínimos e o Algoritmo de Dijkstra

Exercícios 116 a 118

Teorema 64. *Seja (G, w) um grafo conexo com pesos não negativos. Para todo $v \in V(G)$ existe uma árvore de caminhos mínimos enraizada em v.*

Demonstração. Exercício 116. □

CM(G, w, r)

$T \leftarrow (\{r\}, \emptyset)$
Enquanto $\partial_G(V(T)) \neq \emptyset$
    escolha uma aresta $\{x, y\}$ em $\partial_G(V(T))$ tal que
     $d_T(r, x) + w(\{x, y\})$ é mínimo
    acrescente o vértice y e a aresta $\{x, y\}$ a T
Devolva (T, r)

Ao início de cada iteração, T é uma árvore de caminhos mínimos de (G, w) enraizada em r.

Teorema 65. *Seja (G, w) um grafo conexo com pesos não–negativos e seja (T, r) uma árvore de caminhos mínimos de (G, w).*

Se $\{x, y\} \in \partial_G(V(T))$ é tal que $d_G(r, x) + w(\{x, y\})$ é mínimo, então $T + \{x, y\}$ é uma árvore de caminhos mínimos de (G, w) enraizada em r.

Demonstração. Sejam (G, w), (T, r), e $\{x, y\}$ como acima e seja $P = (r = u_0, \ldots, u_n = y)$ um caminho mínimo de r a y em (G, w). Seja ainda m o

maior índice de um vértice de P em $V(T)$, isto é,

$$m := \max\{k \in [0..n] \mid u_k \in V(T)\},$$

e observe que

1. $u_0 T u_m$ é um caminho mínimo em G e, portanto,
2. $w(u_0 T u_m) = w(u_0 P u_m)$ e
3. $\{u_m, u_{m+1}\} \in \partial_G(V(T))$.

Seja $T' = T + \{x, y\}$. Então, pela escolha de $\{x, y\}$ temos

$$\begin{aligned} w(u_0 T' y) &\leq w(u_0 P u_m) + w(\{u_m, u_{m+1}\}) \\ &\leq w(u_0 P u_m) + w(\{u_m, u_{m+1}\}) + w(u_{m+1} P y) = w(P), \end{aligned}$$

e como P é caminho mínimo, então $u_0 T' y$ também tem que ser. □

Corolário 66. *O Algoritmo* CM(G, w, r) *devolve uma árvore de caminhos mínimos de raiz* r *de* (G, w).

CM′(G, w, r)

```
Para cada v ∈ V(G)
    v.estado ← 0
V ← ∅
r.custo ← 0
r.estado ← 1
acrescente r a V
Enquanto V ≠ ∅
    retire um vértice v ∈ V tal que v.custo é mínimo
    Para cada u ∈ Γ_G(v)
        Se u.estado = 1
            Se v.custo + w({u, v}) < u.custo
                u.pai ← v
                u.custo ← v.custo + w({u, v})
        Senão, se u.estado = 0
            u.pai ← v
            u.custo ← v.custo + w({u, v})
            acrescente u a V
            u.estado ← 1
    v.estado ← 2
```

Corolário 67. *É possível computar uma árvore de caminhos mínimos de um grafo conexo com pesos não negativos de n vértices e m arestas em tempo $O(m \log n)$.*

Demonstração. Basta implementar o conjunto V do algoritmo $\mathsf{CM}'_P(G, w, r)$ por meio de uma fila de prioridades.

O trecho do algoritmo anterior ao laço externo pode ser executado em tempo $O(n)$.

Cada iteração retira um elemento v de V, altera o custo de um vizinho u de v e (em alguns casos) acrescenta u a V. Cada iteração executa, portanto, em tempo $O(\log |V|)$.

O número de iterações agregadas entre os laços externo e interno é $2m$.

Como $|V|$ nunca ultrapassa o número de vértices n do grafo, o tempo de execução total é $O(m \log n)$. □

Corolário 68. *É possível computar as distâncias entre todos os pares de vértices e (consequentemente) o diâmetro de um grafo conexo com pesos não negativos de n vértices e m arestas em tempo $O(nm \log n)$.*

É possível "enxugar" o algoritmo

$\mathsf{CM}'(G, w, r)$

```
Para cada v ∈ V(G)
    v.custo ← ∞
V ← V(G)
r.pai ← λ
r.custo ← 0
Enquanto V ≠ ∅
    retire v ∈ V tal que v.custo é mínimo
    Para cada u ∈ Γ_G(v)
        Se v.custo + w({u, v}) < u.custo
            u.pai ← v
            u.custo ← v.custo + w({u, v})
```

Exercício 118.

Tópico 25

Busca em Profundidade

Exercícios 119 a 122

BuscaProfundidade(G)

Para *cada* $v \in V(G)$
    v.estado $\leftarrow 0$
$t \leftarrow 0$
Para *cada* $v \in V(G)$
    Se v.*estado* $= 0$
        v.pai $\leftarrow \Lambda$
        BuscaProfundidade(G, v)

onde

BuscaProfundidade(G, r)

r.pre $\leftarrow ++t$
r.estado $\leftarrow 1$
Para *cada* $v \in \Gamma_G(r)$
    Se v.*estado* $= 1$ *e* $v \neq r$.*pai*
        processe $\{r, v\}$
    Senão, se v.*estado* $= 0$
        v.pai $\leftarrow r$
        processe $\{r, v\}$
        BuscaProfundidade(G, v)
processe r
r.estado $\leftarrow 2$
r.pos $\leftarrow ++t$

Seja F a floresta direcionada resultante da execução de $\mathsf{BuscaProfundidade}(G)$, então

1. $\mathsf{BuscaProfundidade}(G, r)$ processa a floresta direcionada F e as arestas de $G - F$ que ligam descendentes e ancestrais em F.
2. $v.\mathsf{pre}$ é o *índice de pré–ordem* de v, que indica o "momento" de início do processamento da subárvore de T de raiz v ("*DFS number*"),
3. $v.\mathsf{pos}$ é o *índice de pós–ordem* de v, que indica o "momento" do fim do processamento da subárvore de T de raiz v.
4. ao final, $t = 2|V(G)|$.

Teorema 69. *Se F é a floresta direcionada resultante de uma busca em profundidade em um grafo G e u é ancestral de v em F, então*

$$u.\mathit{pre} < v.\mathit{pre} < v.\mathit{pos} < u.\mathit{pos}.$$

Demonstração. Seja F a floresta direcionada resultante de uma busca em profundidade em um grafo G e sejam $u, v \in V(G)$.

Decorre da definição dos números de pré e pós ordem que

$$\begin{aligned} u.\mathsf{pre} &< u.\mathsf{pos}, \\ v.\mathsf{pre} &< v.\mathsf{pos}. \end{aligned}$$

Além disso, se u é ancestral de v em F, é imediato do Algoritmo $\mathsf{BuscaProfundidade}(G, r)$ que

$$\begin{aligned} u.\mathsf{pre} &< v.\mathsf{pre}, \\ v.\mathsf{pos} &< u.\mathsf{pos}, \end{aligned}$$

e daí, temos

$$u.\mathsf{pre} < v.\mathsf{pre} < v.\mathsf{pos} < u.\mathsf{pos}.$$

□

Teorema 70. *Se F é a floresta direcionada resultante de uma busca em profundidade de um grafo G então toda aresta em $G - F$ é de retorno com relação a F.*

Tópico 26

Busca em Grafos Direcionados

Exercícios 123 a 133

Uma busca em um grafo direcionado funciona da mesma forma que em um grafo não direcionado.

Uma diferença importante é que, ao contrário das buscas em grafos não direcionados, as arborescências da floresta direcionada resultantes da busca podem não ser geradoras de suas respectivas componentes.

Por exemplo, uma busca em profundidade sobre um grafo conexo pode resultar em uma floresta com várias arborescências.

Consequentemente pode haver arcos no grafo que ligam diferentes arbo-

rescências.

Busca(G)

$F \leftarrow$ grafo direcionado vazio
Para $v \in V(G)$
    v.estado $\leftarrow 0$
Para $v \in V(G)$
    Se v.*estado* $= 0$
        $(T, v) \leftarrow$ Busca(G, v)
        acrescente T a F
Devolva F

Busca(G, r)

$V_1 \leftarrow \emptyset$
processe r
r.pai $\leftarrow \Lambda$
acrescente r a V_1
r.estado $\leftarrow 1$
Enquanto $V_1 \neq \emptyset$
    $v \leftarrow$ vértice em V_1
    Se *não há "próximo" vértice em* $\Gamma^+_G(v)$
        retire v de V_1
        v.estado $\leftarrow 2$
    Senão
        $w \leftarrow$ "próximo" vértice em $\Gamma^+_G(v)$
        Se w.*estado* $\neq 0$
            Se (v, w) *não foi processada*
                processe (v, w)
        Senão
            processe (v, w)
            processe w
            w.pai $\leftarrow v$
            acrescente w a V_1
            w.estado $\leftarrow 1$

Corolário 71. *O Algoritmo* Busca(G) *devolve uma floresta direcionada geradora do grafo G e processa todos os vértices e arcos de G. Além disso, o laço do Algoritmo* Busca(G, r) *é executado $2|E(G)| + |V(G)|$ vezes.*

Corolário 72. *Se cada operação no laço do Algoritmo* Busca(G, r) *é executada em tempo $O(1)$, então o tempo de execução do Algoritmo* Busca(G) *é $O(|E(G)| + |V(G)|)$.*

Algumas observações importantes.

1. A busca a partir do vértice v devolve uma arborescência de raiz v em vez de uma árvore enraizada em v.

2. Pode devolver florestas direcionadas com mais de um componente, mesmo quando o grafo é conexo.

3. Pode haver arcos cruzados em uma floresta direcionada gerada por uma busca em profundidade.

4. Se T é a arborescência resultante de uma busca no grafo direcionado G, então $V[T]$ é o conjunto dos vértices de G alcançáveis a partir de v em G.

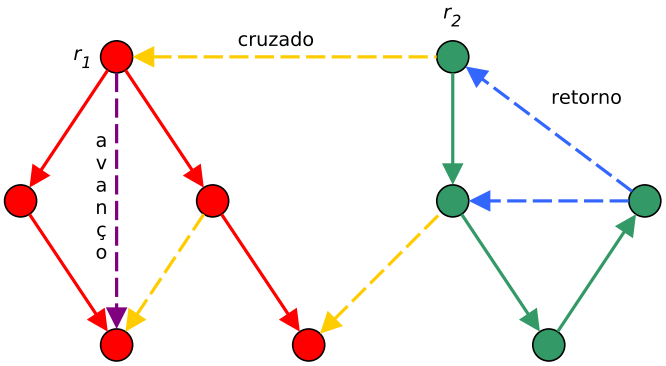


Figura 26.1: Exemplo de busca em profundidade em um grafo direcionado

Seja F uma floresta direcionada.

Uma *pré–ordem* de F é uma permutação de $V(F)$ na qual ancestrais são anteriores a seus descendentes.

Uma *pós–ordem* de F é uma permutação de $V(F)$ na qual ancestrais são posteriores a seus descendentes.

Teorema 73. *Seja F a floresta direcionada resultante da execução de uma busca em profundidade sobre o grafo direcionado G. Para cada $v \in V(G)$ sejam $v.\mathsf{pre}$ e $v.\mathsf{pos}$ os índices de pré–ordem e pós–ordem computados.*

1. *o arco (u, v) é arco de F ou arco de avanço com relação a F, se e somente se $u.\mathsf{pre} < v.\mathsf{pre} < v.\mathsf{pos} < u.\mathsf{pos}$.*

2. *o arco (u, v) é arco cruzado com relação a F, se e somente se $v.\mathsf{pre} < v.\mathsf{pos} < u.\mathsf{pre} < u.\mathsf{pos}$.*

3. *o arco (u, v) é arco de retorno com relação a F, se e somente se $v.\mathsf{pre} < u.\mathsf{pre} < u.\mathsf{pos} < v.\mathsf{pos}$.*

4. *A ordem $<$ induzida sobre $V(G)$ dada por*

$$u < v := u.\mathsf{pre} < v.\mathsf{pre}, \ \mathsf{para\ todo}\ u, v \in V(G),$$

   é uma pré–ordem de F.

5. *A ordem $<$ induzida sobre $V(G)$ dada por*

$$u < v := u.\mathsf{pos} < v.\mathsf{pos}, \ \mathsf{para\ todo}\ u, v \in V(G),$$

   é uma pós–ordem de F.

Demonstração. Exercício 126 □

26.1 Ordenação Topológica

Teorema 74. *O grafo condensado de um grafo direcionado é acíclico.*

Demonstração. Exercício 61. □

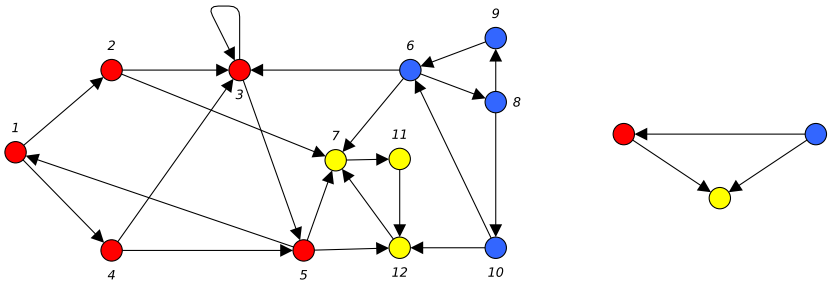


Figura 26.2: Grafo condensado de um grafo direcionado.

Uma *ordenação topológica* de um grafo direcionado G é uma permutação $(v_1, \ldots, v_n)$ de $V(G)$ que "respeita a direção dos arcos" de G, isto é,

$$i < j, \text{ para todo } (v_i, v_j) \in A(G).$$

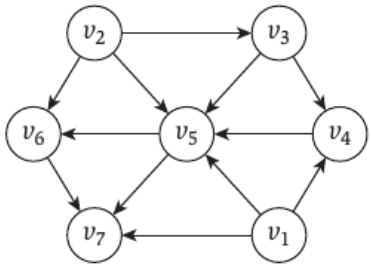


Figura 26.3: isomorfo a Kleinberg and Éva Tardos (2005, fig. 3.7, p. 100)

Teorema 75. *Um grafo direcionado G admite ordenação topológica se e somente se é acíclico.*

Demonstração. Exercício 128 □

Teorema 76. *Um grafo direcionado G é acíclico se e somente se qualquer floresta direcionada resultante de uma busca em profundidade sobre G não tem arcos de retorno.*

Demonstração. Exercício 127 □

Teorema 77. *O reverso da pós–ordem de uma floresta direcionada resultante de uma busca em profundidade em um grafo direcionado acíclico G, é uma ordenação topológica de G.*

Demonstração. Considere uma busca em profundidade sobre um grafo direcionado G e seja $(v_1, \ldots, v_n)$ a permutação de $V(G)$ em ordem decrescente de índices de pós–ordem computados nesta busca, isto é,

$$i < j \text{ se e somente se } v_i.\mathsf{pos} > v_j.\mathsf{pos}, \text{ para todo } 1 \leq i < j \leq n \in A(G),$$

Para provar que esta permutação é uma ordenação topológica de G, basta provar que

$$i < j \text{ para todo } (v_i, v_j) \in A(G),$$

ou seja, que

$$v_i.\mathsf{pos} > v_j.\mathsf{pos} \text{ para todo } (v_i, v_j) \in A(G),$$

ou seja, que

$$v.\mathsf{pos} < u.\mathsf{pos} \text{ para todo } (u, v) \in A(G).$$

Pelo Teorema 73, se tivéssemos $u.\mathsf{pos} < v.\mathsf{pos}$ então (u, v) seria arco de retorno em G.

Como G é acíclico, não tem arco de retorno (T. 76) e portanto,

$$v.\mathsf{pos} < u.\mathsf{pos} \text{ para todo } (u, v) \in A(G),$$

e $(v_1, \ldots, v_n)$ é ordenação topológica de G. □

Ordena(G)

Para *cada* $v \in V(G)$
  $v.\mathsf{estado} \leftarrow 0$
$G.l \leftarrow$ lista vazia
Para *cada* $v \in V(G)$
  Se $v.\mathit{estado} = 0$
    Ordena(G, v)

---

onde

Ordena(G, r)

$r.\mathsf{estado} \leftarrow 1$
Para *cada* $v \in \Gamma^+_G(r)$
  Se $v.\mathit{estado} = 0$
    Ordena(G, v)
acrescente r ao início de $G.l$
$r.\mathsf{estado} \leftarrow 2$

Teorema 78. *É possível determinar uma ordenação topológica de um grafo direcionado acíclico de n vértices e m arcos em tempo $O(n + m)$.*

Tópico 27

Emparelhamentos

Exercícios 134 a 137

Um *emparelhamento* ("matching") em um grafo G é um conjunto de arestas de G sem vértices em comum.

27.1 Coloração de Arestas

Uma *coloração das arestas* de um grafo G é uma partição de $E(G)$ em emparelhamentos. Nesse contexto, cada emparelhamento é chamado de uma *cor* da coloração.

O *índice cromático*, denotado $\chi'(G)$, de G é o menor número de cores distintas necessário para colorir as arestas de um grafo.

Teorema 79 (Vizing, 1964)**.** *Todo grafo G admite uma coloração de suas arestas em $\Delta(G) + 1$ cores e essa coloração pode ser computada em tempo polinomial.*

Teorema 80 (Holyer, 1981)**.** *Decidir se as arestas de um grafo G podem ser coloridas com $\Delta(G)$ cores é um problema $\mathcal{NP}$-difícil.*

27.2 Emparelhamento Máximo

Dizemos que o emparelhamento M *cobre* o vértice v (ou, equivalentemente, que v é *coberto* por M) se v é ponta de alguma aresta de M. Caso contrário, dizemos que v é *descoberto* por M.

Dizemos que um emparelhamento M *cobre* um conjunto X de vértices em um grafo G se todo vértice em X é coberto por M.

Seja M um emparelhamento em um grafo G. O caminho $(v_0, \ldots, v_n)$ é um *caminho M–alternante* em G se suas arestas estão ou não em M de maneira alternada, isto é, se

$$\{v_{i-1}, v_i\} \in M \text{ se e somente se } \{v_i, v_{i+1}\} \notin M, \text{ para todo } 1 \leq i < n.$$

Uma árvore T é M–alternante com relação a um vértice $r \in V(T)$ se o caminho de rTv é M–alternante para todo $v \in V(T)$.

Um caminho M–alternante é chamado de *M–aumentante* se suas pontas não são cobertas por M.

Dados dois conjuntos A e B, defina a diferença simétrica de A e B como

$$A \oplus B := (A \cup B) \setminus (B \cap A).$$

Alternativamente, temos que

$$A \oplus B = (A \setminus B) \cup (B \setminus A)$$

Teorema 81. *Se M é um emparelhamento em um grafo G e C é um caminho M–aumentante em G, então o conjunto $M \oplus E(C)$ é um emparelhamento de tamanho $|M| + 1$ em G.*

Demonstração. Exercício 134 □

Teorema 82 (Berge, 1967)**.** *O emparelhamento M é máximo no grafo G se e somente se não existe caminho M–aumentante em G.*

Demonstração. Seja M um emparelhamento em um grafo G

Se P é um caminho M–aumentante em G, então $M \oplus E(P)$ é um emparelhamento em G com uma aresta a mais que M (T. 81).

Seja agora um emparelhamento M^* em G maior que M e considere o grafo $H := G[M \oplus M^*]$.

Neste caso $\Delta(H) \leq 2$ e daí (Ex. 65), cada componente de H é um ciclo ou um caminho.

Os ciclos tem que ser pares porque M e M^* são emparelhamentos.

Como $|M^*| > |M|$ então ao menos um componente de H tem que ser um caminho P com as pontas cobertas por M^*. Neste caso, P é um caminho M–aumentante em G. □

Teorema 83 (Hall, 1935)**.** *Um grafo bipartido G com bipartição $\{A, B\}$ e seja $X \subseteq A$. O grafo G tem um emparelhamento que cobre X se e somente se*

$$|S| \leq |\Gamma_G(S)|, \text{ para todo } S \subseteq X.$$

Demonstração. Seja G um grafo bipartido com bipartição $\{A, B\}$ e seja $X \subseteq A$.

Se M é um emparelhamento que cobre X, então é evidente que

$$|S| \leq |\Gamma_G(S)|, \text{ para todo } S \subseteq X.$$

Suponha agora que não existe emparelhamento em G que cubra X. Vamos provar que, neste caso, existe um conjunto $S \subseteq X$ para o qual $|S| > |\Gamma_G(S)|$.

Seja M um emparelhamento máximo em G. Seja $x \in X$ um vértice não coberto por M e seja T uma árvore M–alternante maximal com relação a x.

Seja então P o conjunto dos vértices que estão a distância par de x em T, isto é,

$$P := \{v \in V(T) \mid d_T(x, v) \text{ é par}\}.$$

Como T é árvore M–alternante maximal e M é emparelhamento máximo, então x é o único vértice de T não coberto por M e, consequentemente, T tem um número ímpar de vértices. Note que se existisse outro vértice u não coberto, então xTu seria caminho M–aumentante.

Como T é árvore, então $\Gamma_T(P) = V(T) - P$ e como $x \in P$, então

$$|P| > |\Gamma_T(P)|.$$

Isso decorre do fato de todos os vértices de $T - x$ serem cobertos por M e das arestas de M terem como pontas um vértice de P e um de $\Gamma_T(P)$. Ou seja, $|P| = |\Gamma_T(P)| + 1$.

Finalmente, observe que $\Gamma_T(P) = \Gamma_G(P)$ pois caso contrário T não seria árvore M–alternante maximal e portanto,

$$|P| > |\Gamma_G(P)|.$$

□

```
EmparelhamentoMaximo(G)
M ← ∅
Para cada v ∈ V(G)
    Se v não é coberto por M
        P ← CaminhoAumentante(G,M,v)
        Se P não é vazio
            M ← M ⊕ P
Devolva M
```

```
CaminhoAumentante(G, M, v)
T ← ({v}, ∅)
P ← {v}
Enquanto Γ_G(P) − V(T) ≠ ∅
    w ← um vértice em Γ_G(P) − V(T)
    u ← um vizinho de w em T
    acrescente a aresta {u, w} a T
    Se w não está coberto por M
        Devolva vTw
    Senão
        acrescente a T a aresta {w, t} de M que cobre w
        acrescente t a P
Devolva ()
```

Teorema 84. *É possível computar um emparelhamento máximo de um grafo bipartido com n vértices e m arestas em tempo $O(nm)$.*

Demonstração. Seja G um grafo bipartido de n vértices e m arestas. Considere a execução de EmparelhamentoMaximo(G).

O Algoritmo CaminhoAumentante(G, M, v), pode ser implementado por meio de uma busca em largura adaptada. Como todas as operações no seu laço podem ser executadas em tempo $O(1)$, o algoritmo pode ser executado em tempo $O(m)$.

O Algoritmo CaminhoAumentante(G, M, v) é invocado no máximo $\lceil n/2 \rceil$ vezes.

Finalmente, observe que o valor de $M \oplus P$ no laço do Algoritmo EmparelhamentoMaximo(G) também pode ser computado em tempo $O(m)$. □

Tópico 28

O Algoritmo de Kruskal

Exercícios 138 a 140

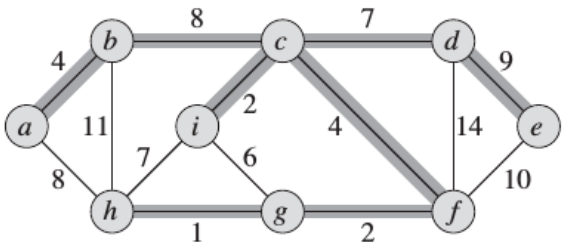


Figura 28.1: (Cormen et al., 2009, fig 23.1, p. 625)

AGM$_K(G, w)$

$F \leftarrow (V(G), \emptyset)$
$E \leftarrow E(G)$
Enquanto $|E(F)| < |V(F)| - 1$
    remova de E uma aresta $\{u, v\}$ de peso mínimo
    Se *u e v estão em componentes distintos de F*
        acrescente $\{u, v\}$ a F
Devolva F

Teorema 85. *A execução de AGM(G, w) devolve uma árvore geradora de peso mínimo do grafo conexo com pesos (G, w).*

Demonstração. Seja (G, w) um grafo conexo com pesos e seja $n = |V(G)|$.

Ao início do laço, é sempre verdade que o grafo F é uma floresta geradora de G.

O laço só termina quando $|E(F)| = |V(F)| - 1$, ou seja (C. 27), quando F é árvore.

Como $|V(F)| = |V(G)| = n$, então, ao fim do laço, F é árvore geradora de G.

Logo, o algoritmo devolve uma árvore geradora de G.

Resta provar que esta árvore tem peso mínimo.

Seja, então, T uma árvore geradora mínima de G com número máximo de arestas em comum com F. Vamos provar que $T = F$ e, portanto, F é árvore geradora mínima de G.

Seja $(f_1, \ldots, f_{n-1})$ a sequência de arestas acrescentadas a F ao longo da execução do algoritmo.

Suponha que $E(T) \neq E(F)$ e seja m o índice da primeira aresta de F que não está em T, isto é,

$$m := \min \{i \in [1..n-1] \mid f_i \notin E(T)\}.$$

Então $T + f_m$ tem um único ciclo (T. 29). Além disso, neste ciclo tem que existir uma aresta $t \notin F$, pois F é árvore.

Note que t não pode ter sido testada pelo algoritmo antes de f_m, senão seria escolhida.

Por causa da escolha de f_m na execução de AGM(G,w) temos

$$w(f_m) \leq w(t),$$

e, consequentemente,

$$w(T - t + f_m) = w(T) - w(t) + w(f_m) \leq w(T).$$

Como $T - t + f_m$ é árvore, pois é acíclico e tem o mesmo número de arestas que T, e T é árvore geradora mínima, somos obrigados a concluir

$$w(T - t + f_m) = w(T),$$

o que não pode ser pois, neste caso, $T - t + f_m$ seria uma árvore geradora de peso mínimo com mais arestas em comum com F do que T. □

Veremos agora como testar se dois vértices estão em componentes distintos de F.

Para cada $v \in V(G)$, a variável $v.\mathsf{componente}$ tem o valor de um vértice de G no mesmo componente de v.

Para cada componente C de G, existe um único vértice $r \in V(C)$ tal que $r.\mathsf{componente} = r$. Este vértice é chamado de *representante* do componente C.

Se r é o representante de um componente, então $r.\mathsf{nivel}$ é um limite superior para o tamanho da maior sequência

$$(v, v.\mathsf{componente}, v.\mathsf{componente}.\mathsf{componente}, \ldots, r)$$

terminando em r.

```
AGM_K(G, w)
F ← (V(G), ∅)
Para cada v ∈ V(F)
    v.componente ← v
    v.nivel ← 1
E ← E(G)
Enquanto |E(F)| < |V(F)| − 1
    remova de E uma aresta {u, v} de peso mínimo
    Se Representante(F, u) ≠ Representante(F, v)
        acrescente {u, v} a F
        UneComponentes(F, u.componente, v.componente)
Devolva F
```

```
Representante(G, v)
l ← lista vazia
Enquanto v.componente ≠ v
    acrescente v a l
    v ← v.componente
Enquanto l não é vazia
    retire um vértice u de l
    u.componente ← v
Devolva v
```

```
UneComponentes(G, u, v)
Se u.nivel < v.nivel
    u.componente ← v
Senão
    v.componente ← u
    Se u.nivel = v.nivel
        ++u.nivel
```

Teorema 86. *É possível computar uma árvore geradora de peso mínimo de um grafo com pesos (G, w) com n vértices e m arestas em tempo $O(m \log n)$.*

Demonstração. Seja (G, w) um grafo conexo com n vértices e m arestas e considere uma implementação do Algoritmo $\mathsf{AGM}_K(G, w)$ onde o conjunto E é implementado por meio de uma lista das arestas em ordem não decrescente de peso.

A inicialização do conjunto E pode ser feita em tempo $O(m \log m)$.

O laço do algoritmo executa no máximo m iterações.

O tempo total de até m execuções de $\mathsf{UneComponentes}(F, u, v)$ e $\mathsf{Representante}(F, u)$ combinadas é $O(m\alpha(m))$ (Cormen et al., 2009, sec. 21.4), onde $\alpha(m)$ denota a função inversa da *função de Ackermann*, que satisfaz

$$\alpha(n) \leq 4, \text{ para todo } n \leq 2^{2048}.$$

Como $m < n^2$, então

$$\log m < \log n^2 = 2 \log n = O(\log n),$$

e o tempo de execução de $\mathsf{AGM}_K(G, w)$ é $O(m \log n)$. □

Tópico 29

Distâncias entre Todos os Pares de Vértices

Exercícios 141 a 145

Seja (G, w) um grafo direcionado com pesos e seja $(v_1, \ldots, v_n)$ uma ordenação de $V(G)$.

Dados $i, j \in [1..n]$ e $k \in [0..n]$, um (i, j, k)–*caminho* em G é um caminho de v_i a v_j em G cujos vértices internos estão em $\{v_1, \ldots, v_k\}$.

A k–*distância* de v_i a v_j em G é o peso de um (i, j, k)–caminho de peso mínimo e será denotada $d_{G,w}(i, j, k)$.

Observe que

1. se P é um (i, j, k)–caminho em (G, w) então P também é um $(i, j, k{+}1)$–caminho em (G, w).

2. a 0–distância de v_i a v_j é

$$d_{G,w}(i, j, 0) \begin{cases} 0, & \text{se } i = j, \\ w(v_i, v_j), & \text{se } i \neq j \text{ e } (v_i, v_j) \in A(G), \\ \infty, & \text{se se } i \neq j \text{ e } (v_i, v_j) \notin A(G). \end{cases}$$

3. a n–distância de v_i a v_j é a distância de v_i a v_j em G, isto é,

$$d_{G,w}(i, j, n) = d_{G,w}(v_i, v_j), \text{ para todo } i, j \in [1..n].$$

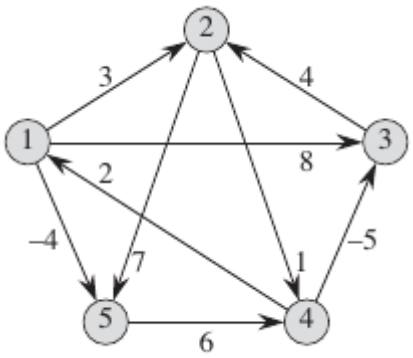


Figura 29.1: Cormen et al. (2009), fig. 25.1

Teorema 87. *Seja (G, w) um grafo direcionado com pesos nas arestas, sendo $V(G) = \{v_1, \ldots, v_n\}$. Para todo $i, j \in [1..n]$ e todo $k \in [0..n]$,*

$$d_{G,w}(i,j,k) = \begin{cases} 0, & \textsf{se } i = j, \\ w(v_i, v_j), & \textsf{se } i \neq j \textsf{ e } k = 0 \textsf{ e } (v_i, v_j) \in A(G), \\ \infty, & \textsf{se } i \neq j \textsf{ e } k = 0 \textsf{ e } (v_i, v_j) \notin A(G), \\ \min\{d_{G,w}(i,j,k-1), & \\ d_{G,w}(i,k,k-1) + d_{G,w}(k,j,k-1)\}, & \textit{caso contrário.} \end{cases}$$

Demonstração. Exercício 141. □

29.1 Algoritmo Recursivo

```
Distancias_R(G, w)
    Para i ← 1 até |V(G)|
        Para j ← 1 até |V(G)|
            d[i, j] ← Distancia_R(G, w, i, j, |V(G)|)
    Devolva d
```

```
Distancia_R(G, w, i, j, k)
    Se i = j
        Devolva 0
    Se k = 0
        Se (v_i, v_j) ∈ A(G)
            Devolva w(v_i, v_j)
        Devolva ∞
    D ← Distancia_R(G, w, i, j, k − 1)
    D' ← Distancia_R(G, w, i, k, k − 1) + Distancia_R(G, w, k, j, k − 1)
    Se D' < D
        Devolva D'
    Devolva D
```

O Algoritmo $\mathsf{Distancias}_R(G, w)$ tem complexidade $\Theta(n^2 3^n)$ devido a diversas repetições de subproblemas.

29.2 Programação Dinâmica: O Algoritmo de Floyd–Warshall

Distancias(G, w)

$d \leftarrow$ matriz $|V(G)| \times |V(G)| \times (|V(G)| + 1)$
Para $k \leftarrow 0$ *até* $|V(G)|$
    Distancias(G, w, d, k)
Devolva d

Distancias(G, w, d, k)

Para $i \leftarrow 1$ *até* $|V(G)|$
    Para $j \leftarrow 1$ *até* $|V(G)|$
        $d[i, j, k] \leftarrow$ Distancia(G, w, d, i, j, k)
Devolva d

O Algoritmo Distancias(G, w, d, k) devolve as k–distâncias entre os vértices de G de forma que

$$d[i, j, k] = d_{G,w}(v_i, v_j, k), \text{ para todo } 1 \leq i, j \leq |V(G)|,$$

assumindo que

$$d[i, j, k-1] = d_{G,w}(v_i, v_j, k-1), \text{ para todo } 1 \leq i, j \leq |V(G)|.$$

Distancia(G, w, d, i, j, k)

Se $i = j$
    Devolva 0
Se $k = 0$
    Se $(v_i, v_j) \in A(G)$
        Devolva $w(v_i, v_j)$
    Devolva ∞
$D \leftarrow d[i, j, k-1]$
$D' \leftarrow d[i, k, k-1] + d[k, j, k-1]$
Se $D' < D$
    Devolva D'
Devolva D

O Algoritmo Distancia(G, w, d, i, j, k) devolve a k–distância de v_i a v_j em (G, w) assumindo que

$$\begin{aligned} d[i, j, k-1] &= d_{G,w}(v_i, v_j, k-1), \text{ e} \\ d[i, k, k-1] &= d_{G,w}(v_i, v_k, k-1), \text{ e} \\ d[k, j, k-1] &= d_{G,w}(v_k, v_j, k-1). \end{aligned}$$

Para cada $k \in [1..n]$, as k-ésima linha e coluna de $d[,,k-1]$ e $d[,,k]$ são iguais, isto é,

$$\begin{aligned} d[k,j,k] &= d[i,k,k-1], \text{ e} \\ d[i,k,k] &= d[k,j,k-1], \end{aligned}$$

para todo $i,j \in [1..n]$, pois

$$\begin{aligned} d[k,j,k] &= \min\{d[k,j,k-1], d[k,k,k-1] + d[k,j,k-1]\} \\ &= \min\{d[k,j,k-1], 0 + d[k,j,k-1]\} = d[k,j,k-1], \end{aligned}$$

e

$$\begin{aligned} d[i,k,k] &= \min\{d[i,k,k-1], d[i,k,k-1] + d[k,k,k-1]\} \\ &= \min\{d[k,j,k-1], d[i,k,k-1] + 0\} = d[i,k,k-1]. \end{aligned}$$

Para cada $k \in [1..n]$, para computar $d[i,j,k]$ precisamos apenas de $d[i,j,k-1]$ e da k-ésima linha e coluna de $d[,,k-1]$, pois

$$d[i,j,k] = \min\{d[i,j,k-1], d[i,k,k-1] + d[k,j,k-1]\}$$

Distancias(G,w)

$d \leftarrow$ matriz $|V(G)| \times |V(G)|$
Para $k \leftarrow 0$ *até* $|V(G)|$
    Distancias(G,w,d,k)
Devolva d

Distancias(G,w,d,k)

Para $i \leftarrow 1$ *até* $|V(G)|$
    Para $j \leftarrow 1$ *até* $|V(G)|$
        $d[i,j] \leftarrow$ Distancia(G,w,d,i,j,k)
Devolva d

O Algoritmo Distancias(G,w,d,k) devolve as k–distâncias entre os vértices de G de forma que

$$d[i,j] = d_{G,w}(v_i, v_j, k), \text{ para todo } 1 \leq i,j \leq |V(G)|,$$

assumindo que, ao início,

$$d[i,j] = d_{G,w}(v_i, v_j, k-1), \text{ para todo } 1 \leq i,j \leq |V(G)|.$$

```
Distancia(G, w, d, i, j, k)
Se k = 0
    Se (v_i, v_j) ∈ A(G)
        Devolva w(v_i, v_j)
    Devolva ∞
D ← d[i, j]
D' ← d[i, k] + d[k, j]
Se D' < D
    Devolva D'
Devolva D
```

O Algoritmo Distancia(G, w, d, i, j, k) devolve $d_{G,w}(v_i, v_j, k)$ assumindo que

$$\begin{aligned} d[i,j] &= d_{G,w}(v_i, v_j, k-1), \text{ e} \\ d[i,k] &= d_{G,w}(v_i, v_k, k-1), \text{ e} \\ d[k,j] &= d_{G,w}(v_k, v_j, k-1). \end{aligned}$$

O Algoritmo Distancias(G, w, d, k) devolve as k–distâncias entre os vértices de G de forma que

$$d[i,j] = d_{G,w}(v_i, v_j, k), \text{ para todo } 1 \leq i, j \leq |V(G)|,$$

assumindo que, ao início,

$$d[i,j] = d_{G,w}(v_i, v_j, k-1), \text{ para todo } 1 \leq i, j \leq |V(G)|.$$

e que

$$\begin{aligned} d[i,j] &= d_{G,w}(v_i, v_j, k-1), \text{ e} \\ d[i,k] &= d_{G,w}(v_i, v_k, k-1), \text{ e} \\ d[k,j] &= d_{G,w}(v_k, v_j, k-1). \end{aligned}$$

Teorema 88. *É possível computar as distâncias entre todos os pares de vértices de um grafo direcionado com pesos de n vértices em tempo $O(n^3)$.*

Demonstração. Seja (G, w) um grafo direcionado com pesos de n vértices.

É possível implementar o algoritmo de maneira que cada execução de Distancia(G, w, d, i, j, k) toma tempo $O(1)$.

Assim, cada execução de Distancia(G, w, d, k) toma tempo $O(n^2)$ e a execução de Distancias(G, w) toma tempo $O(n^3)$. □

É comum apresentar o Algoritmo de Floyd-Warshall de forma mais compacta como abaixo.

```
Distancias(G, w)
d ← matriz |V(G)| × |V(G)|
Para i ← 1 até |V(G)|
    Para j ← 1 até |V(G)|
        d[i, j] ← ∞
        Se i = j
            d[i, j] ← 0
        Senão
            Se (v_i, v_j) ∈ A(G)
                d[i, j] ← w(v_i, v_j)
Para k ← 1 até |V(G)|
    Para i ← 1 até |V(G)|
        Para j ← 1 até |V(G)|
            D ← d[i, k] + d[k, j]
            Se D < d[i, j]
                d[i, j] ← D
Devolva d
```

Tópico 30

Planaridade

Exercícios 146 a 152

Neste tópico e no seguinte os grafos podem não ser simples, isto é, diferentes arestas podem ter pontas coincidentes.

Um *desenho (ou imersão) de um grafo* G é um grafo isomorfo a G cujos vértices são pontos do $\mathbb{R}^2$ e cujas arestas são curvas contínuas sem auto–interseções ligando suas pontas. O desenho é dito *planar* se suas arestas interseptam-se somente nos vértices que tem em comum.

Um grafo é *planar* se admite desenho planar.

30.1 Dualidade

Uma *face* de um desenho planar G de um grafo é uma região conexa do $\mathbb{R}^2$ delimitada por arestas de G.

O *grafo dual* de um desenho planar G é o grafo G^* cujos vértices são as faces de G e que tem uma aresta e^* para cada aresta e de G, cujas pontas são as faces que e delimita em G. Por extensão, dizemos que um grafo dual de um desenho planar de um grafo G é um grafo dual de G.

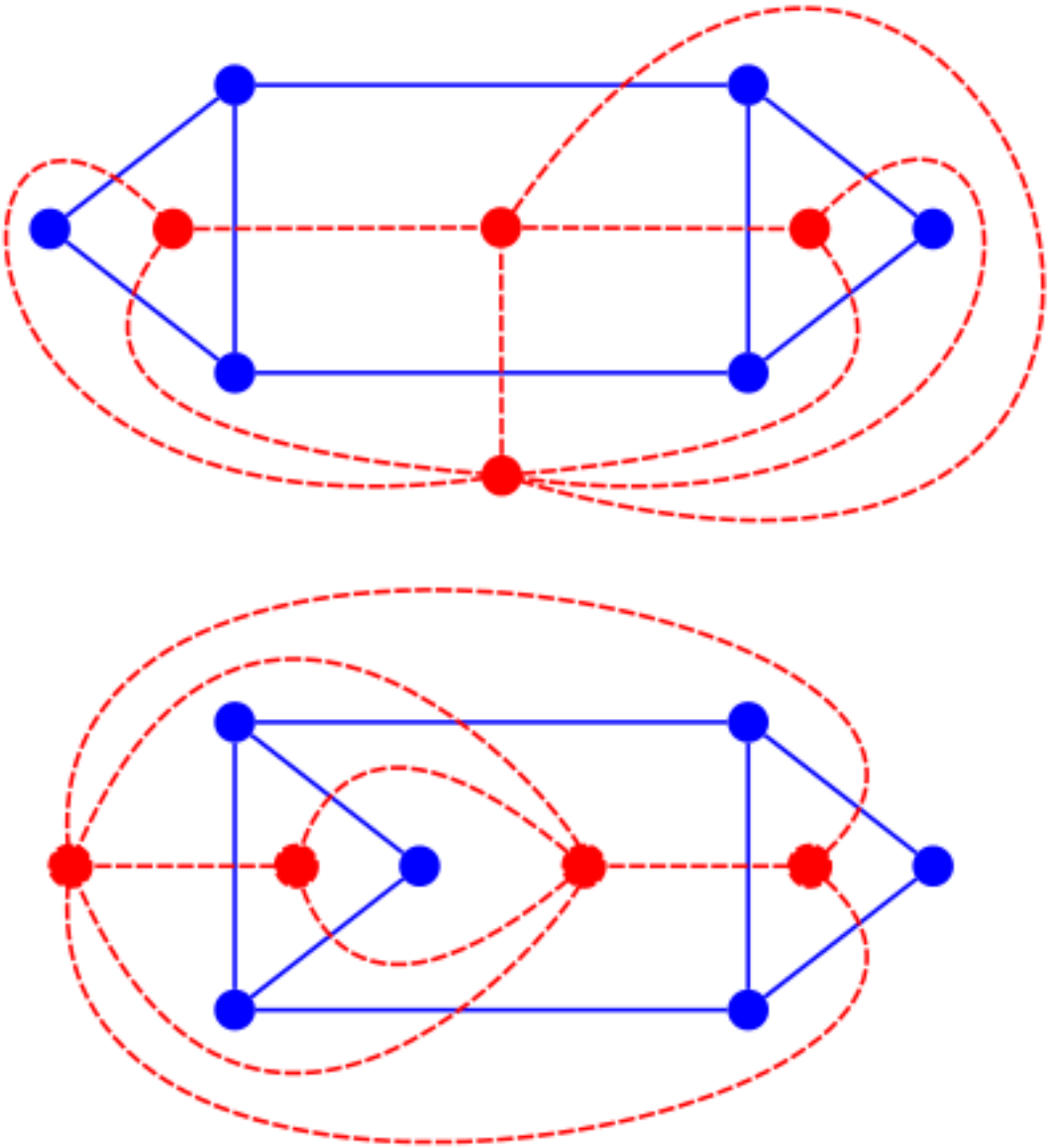


Figura 30.1: grafos duais não são únicos (https://en.wikipedia.org/wiki/Dual_graph)

Teorema 89. *Um grafo dual de um grafo planar também é planar.*

Teorema 90. *Se G^* é um grafo dual de G, então G é isomorfo a um grafo dual de G^*.*

Teorema 91. *Se G é um desenho planar de um grafo conexo e $F(G)$ é o conjunto de suas faces, então,*

$$|V(G)| + |F(G)| = |E(G)| + 2.$$

Demonstração. Por indução em $|F(G)|$.

Base: $|F(G)| = 1$.

Neste caso G é árvore e, consequentemente,

$$|E(G)| = |V(G)| - 1,$$

e daí,

$$|V(G)| + |F(G)| = |E(G)| + 2.$$

H.I.: Seja $f \in \mathbb{N}$ tal que

$$|V(G)| + |F(G)| = |E(G)| + 2,$$

para todo desenho planar G de grafo conexo com $|F(G)| \leq f$.

Passo: Vamos provar que

$$|V(G)| + |F(G)| = |E(G)| + 2$$

para todo desenho planar G de grafo conexo com $|F(G)| = f + 1$.

Seja G um desenho de grafo planar conexo com $f + 1$ faces e seja G' o desenho de grafo resultante de remover uma aresta que separa duas faces distintas de G.

Então G' tem f faces e, pela HI,

$$|V(G')| + |F(G')| = |E(G')| + 2$$

Como

$$\begin{aligned} V(G) &= V(G'), \\ |E(G)| &= |E(G')| + 1, \\ |F(G)| &= |F(G')| + 1, \end{aligned}$$

então

$$|V(G)| + |F(G)| = |V(G')| + |F(G')| + 1 = |E(G')| + 2 + 1 = |E(G)| + 2.$$

□

Corolário 92. *Se G é um grafo planar conexo, e G^* é um dual de G, então,*

$$|V(G)| + |V(G^*)| = |E(G)| + 2 = |E(G^*)| + 2$$

Teorema 93. *Se G é um grafo planar simples com ciclos, então*

$$|E(G)| \leq \frac{\gamma(G)}{\gamma(G) - 2}(|V(G)| - 2).$$

Demonstração. Observe que basta provar a desigualdade para grafos conexos.

Seja então G um grafo planar conexo simples, com ciclos e considere um dual G^* de G.

Como G é simples, cada face f de um desenho planar de G está delimitada por pelo menos $\gamma(G)$ arestas e, consequentemente, $\delta_{G^*}(f) \geq \gamma(G)$.

Então (T. 1)

$$2|E(G^*)| = \sum_{f \in V(G^*)} \delta_{G^*}(f) \geq \gamma(G)|V(G^*)|$$

Por outro lado, (C. 92),

$$\begin{aligned} 2|E(G^*)| &= 2|E(G)| \\ &\geq \gamma(G)|V(G^*)| = \gamma(G)(|E(G)| - |V(G)| + 2) \\ &= \gamma(G)|E(G)| - \gamma(G)(|V(G)| - 2), \end{aligned}$$

e, portanto,

$$(\gamma(G) - 2)|E(G)| \leq \gamma(G)(|V(G)| - 2),$$

isto é,

$$|E(G)| \leq \frac{\gamma(G)}{\gamma(G) - 2}(|V(G)| - 2).$$

□

Corolário 94. *Todo grafo planar simples com $n \geq 3$ vértices tem no máximo $3n - 6$ arestas.*

Demonstração. Exercício 146. □

Corolário 95. *Um grafo completo de cinco vértices não é planar.*

Corolário 96. *Todo grafo bipartido planar simples com $n \geq 3$ vértices tem no máximo $2n - 4$ arestas.*

Demonstração. Exercício 147. □

Corolário 97. *Um grafo bipartido completo com três vértices em cada parte não é planar.*

Corolário 98. *Se G é um grafo planar simples, então $\delta(G) \leq 5$.*

Demonstração. Exercício 148. □

30.2 Subdivisões

Dizemos que o grafo H é *subdivisão* do grafo G quando $V(G) \subseteq V(H)$ e a cada aresta $\{u, v\}$ de G corresponde um caminho de u a v em H e os vértices internos destes caminhos tem grau 2 em H.

Teorema 99. *Um grafo G é planar se e somente se toda subdivisão de G é planar.*

Teorema 100 (Kuratowski, 1930)**.** *Um grafo G é planar se e somente se não tem subdivisão de K_5 ou $K_{3,3}$ como subgrafo.*

Teorema 101 (das Quatro Cores)**.** *Todo grafo planar é 4–colorível*

Demonstração. Appel and Haken (1977) e Appel, Haken, and Koch (1977). □

Tópico 31

Algoritmo de Teste de Planaridade

Corolário 102. *Um grafo é planar se e somente se cada um dos seus blocos é planar.*

Demonstração. Toda subdivisão de K_5 e $K_{3,3}$ é um bloco. □

Seja G um grafo e C um ciclo de G. Um *segmento* de G com relação a C é uma corda de C, ou um componente S de $G - C$, juntamente com as arestas que ligam S a C.

Um ciclo de C é *separador* em um grafo G se G tem mais de um segmento com relação a C.

Teorema 103. *Se o único segmento de um grafo G com relação a um ciclo C não for um caminho, então G tem ciclo separador.*

Demonstração. Seja C um ciclo em um grafo G e seja S um segmento de G com relação a C que não é um caminho. Sejam ainda

u, v**:** vértices de $V(C) \cap V(S)$ tais que existe um caminho P em C de u a v sem vértices de S,

Q**:** o caminho de v a u em C,

R**:** um caminho de u a v em S.

Neste caso $C' = Q \cdot R$ é um ciclo separador em G. Um dos segmentos de G com relação a C' é o caminho P.

Como S não é caminho, então S tem ao menos uma aresta e fora de R. Esta aresta também não está em C' e, consequentemente, o seu componente em $G - C'$ é um segmento S' de G com relação a C'. Como não há vértices de S em P, o segmento S' tem que ser distinto de P. □

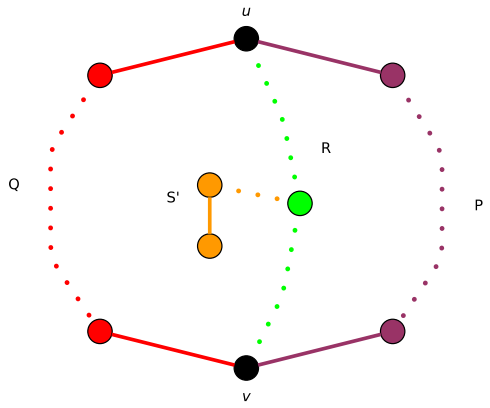


Figura 31.1: Ciclo C com segmento em relação a C que não é um caminho.

Teorema 104. *Todo grafo sem ciclo separador é planar.*

Demonstração. Seja G um grafo sem ciclo separador.

No caso acíclico os blocos de G são vértices isolados ou arestas (pontes). Pelo Corolário 102, G é planar.

Suponha então que C é um ciclo de G. Como C não é separador, G tem no máximo um segmento S com relação a C. Pelo Teorema 103 S, se existir, é um caminho. Um desenho planar de G pode ser contruído com um desenho de C de forma circular, e um desenho de S como um caminho entre dois vértices de C posicionado internamente ao círculo de C. □

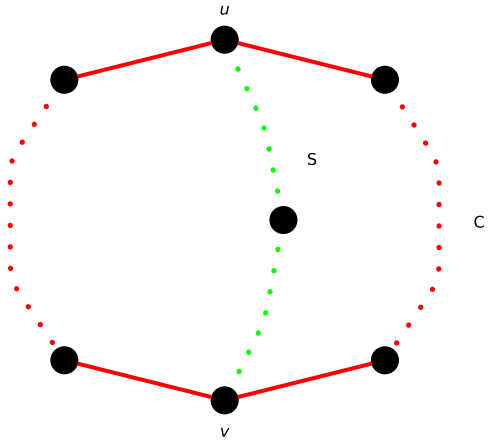


Figura 31.2: Ciclo C com segmento S em relação a C e S é um caminho.

Dados um grafo G, um ciclo C em G e dois segmentos S e T de G com relação a C, dizemos que S e T são *compatíveis* se existe caminho em $G[C]$ contendo $V(S) \cap V(C)$ sem nenhum vértice de T.

Seja então $X(G, C)$ o grafo cujos vértices são os segmentos de G com relação a C, ligados por uma aresta quando não são compatíveis, isto é,

$$E(X(G,C)) := \{\{S,T\} \subseteq V(X(G,C)) \mid S \text{ e } T \text{ não são compatíveis}\}$$

Teorema 105. *Um grafo G é planar se e somente se,*

1. *é acíclico, ou*
2. *para todo ciclo C de G,*
   (a) *o grafo $G[C \cup S]$ é planar para cada segmento S de G com relação a C, e*
   (b) *o grafo $X(G, C)$ é bipartido.*

```
Planar(G)
  Se |V(G)| ≤ 4 ou G é acíclico ou G não tem ciclo separador
     Devolva sim
  Se |E(G)| > 3|V(G)| − 6
     Devolva não
  Se G tem mais de um bloco
     Para cada bloco B de G
        Se não Planar(B)
           Devolva não
     Devolva sim
  C ← ciclo separador em G
  X ← grafo vazio
  Para cada segmento S de G com relação a C
     P ← caminho minimal em C contendo V(S) ∩ V(C)
     Se não Planar(P ∪ S)
        Devolva não
     acrescente o vértice S ao grafo X
     Para cada T ∈ V(X) − {S}
        Se S e T não são compatíveis
           acrescente a aresta {S, T} ao grafo X
           Se o grafo X não é bipartido
              Devolva não
  Devolva sim
```

Teorema 106 (Hopcroft, Tarjan (1974))**.** *É possível implementar o Algoritmo* Planar(G) *de maneira que sua execução sobre um grafo de n vértices leva tempo $O(n)$.*

Apêndice A

Exercícios

A.1 Fundamentos

1⋆. Exiba a fronteira do conjunto $\{d_0, d_1, d_3\}$ no grafo da direita do Exercício 9.

2⋆. Quantas arestas tem um grafo completo de n vértices? Justifique sua resposta.

3⋆. Se um vértice tem k vizinhos em G, quantos vizinhos ele tem em $\overline{G}$? Justifique sua resposta.

4. Exercício intencionalmente deixado em branco

5⋆. Quantos grafos diferentes existem com o conjunto de vértices $\{1, \ldots, n\}$? Justifique.

6⋆. Seja G um grafo. Dados $X \subseteq V(G)$ e $n \in \mathbb{N}$ vamos definir $\Gamma^n_G(X)$ como segue.

$$\Gamma^n_G(X) := \begin{cases} X, & \text{se } n = 0, \\ \Gamma_G(\Gamma^{n-1}_G(X)), & \text{se } n > 0. \end{cases}$$

(a) Exiba os conjuntos $\Gamma^n_G(\{a_0, b_2\})$ para $n \in \{1, 2, 3, 4, 5\}$, onde G é o grafo da esquerda no Exercício 9.

(b) É verdade que para qualquer grafo G e qualquer conjunto X de seus vértices, existe um valor de $n \in \mathbb{N}$ tal que $\Gamma^n_G(X) = V(G)$? Justifique.

(c) É verdade que para qualquer grafo G e qualquer conjunto X de seus vértices, existe um valor de $n \in \mathbb{N}$ tal que $\Gamma^n_G(X) = \Gamma^{n+1}_G(X)$? Justifique.

7⋆. Seja $\mathcal{F} = \{A_1, \ldots, A_n\}$ uma família de subconjuntos de um conjunto A. O *grafo de interseção de* $\mathcal{F}$ é o grafo $\mathcal{I}_\mathcal{F}$ dado por

$$\begin{aligned} V(\mathcal{I}_\mathcal{F}) &:= \mathcal{F}, \\ E(\mathcal{I}_\mathcal{F}) &:= \{\{A_i, A_j\} \mid A_i \cap A_j \neq \emptyset\}. \end{aligned}$$

(a) Seja G o grafo interseção de $\binom{\{1,\ldots,5\}}{2}$. Desenhe G.

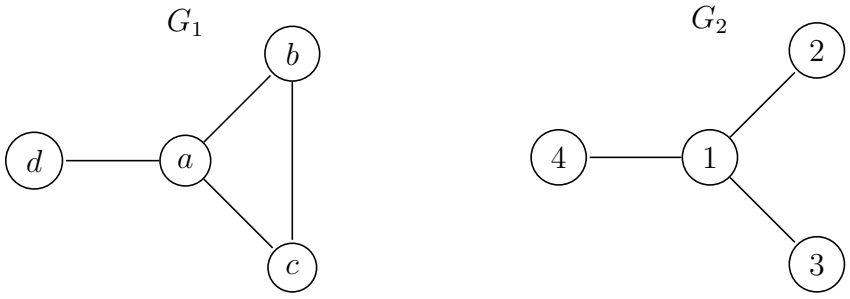
(b) Desenhe $\overline{G}$.

(c) Verifique que $\overline{G}$ é (isomorfo a)o Grafo de Petersen.

Dados inteiros n e k, o complemento do grafo interseção de $\binom{\{1,\ldots,n\}}{k}$ é conhecido como *grafo de Kneser* com parâmetros n e k.

8⋆. Um grafo G é um *grafo de intervalo* se é o grafo de interseção[1] de um conjunto de intervalos fechados de $\mathbb{R}$.

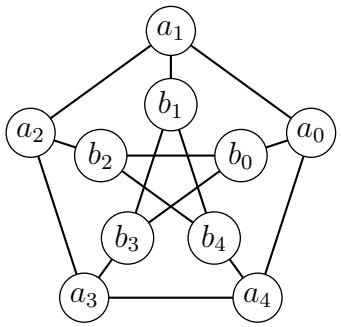
(a) Os grafos G_1 e G_2 abaixo são grafos de intervalo? Justifique.

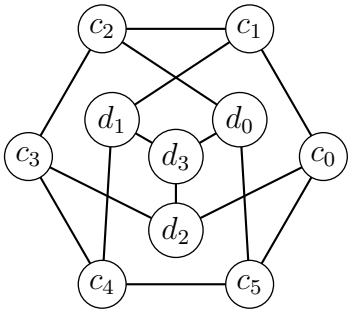


(b) Dê um exemplo de um grafo que **não é** grafo de intervalo e prove que seu exemplo está correto.

[1] Veja o Exercício 7

9⋆. Prove que os grafos abaixo são isomorfos.





10⋆. Um grafo é *auto–complementar* se é isomorfo ao seu complemento.

(a) Dê um exemplo de um grafo auto–complementar não trivial.

(b) Prove se G é auto–complementar, então $|V(G)|$ ou $|V(G)| - 1$ é múltiplo de 4.

11. Exercício intencionalmente deixado em branco

12#. Seja G um grafo com $V(G) = \{v_1, \ldots, v_n\}$ tal que $\delta_G(v_i) = i$ para cada $1 \leq i < n$. O objetivo deste exercício é determinar os possíveis valores de $d := \delta_G(v_n)$, com base no seguinte roteiro.

Para cada $0 \leq i < n/2$ seja G_i o grafo dado por

$$G_i = \begin{cases} G, & \text{se } i = 0, \\ G_{i-1} - \{u_{i-1}\} - \{t_{i-1}\}, & \text{se } i > 0, \end{cases}$$

onde u_i é o vértice de grau 1 em G_i e t_i é o vértice que é vizinho de todos os demais em G_i.

(a) Dê uma expressão para $n_i := |V(G_i)|$ em função de n_{i-1}, isto é, uma recorrência descrevendo n_i.

(b) Dê uma expressão para $\delta_{G_i}(v_n)$.

(c) Descreva o grafo G_i para $i = \left\lfloor \frac{n}{2} \right\rfloor$ (n ímpar) e $i = \frac{n}{2} - 1$ (n par) e conclua daí os possíveis valores para d.

13⋆. (a) Se G é um grafo de 14 vértices e 25 arestas cujos vértices tem graus 3 ou 5, quantos vértices tem grau 3 e quantos tem grau 5?

(b) Generalize o raciocínio para um grafo de n vértices e m arestas cujos vértices tem graus d_1 ou d_2.

14⋆. Prove que se G é um grafo satisfazendo

$$\begin{aligned} \delta(G) &> 0, \\ |E(G)| &< |V(G)|, \end{aligned}$$

então G tem (pelo menos) dois vértices de grau 1.

15⋆. Existe algum grafo não trivial em que todos os vértices tem graus distintos? Justifique.

16⋆. Assim como se define o grau mínimo e o grau máximo, é possível definir o *grau médio* de um grafo G.

(a) Denotando por $\overline{\delta}(G)$ o grau médio de um grafo G, proponha uma expressão para $\overline{\delta}(G)$ em função de $|V(G)|$ e $|E(G)|$.

(b) Prove que, para todo grafo G,

$$\delta(G) \leq \frac{2|E(G)|}{|V(G)|} \leq \Delta(G).$$

17. Exercício intencionalmente deixado em branco

18⋆. (a) A sequência $(5, 4, 4, 3, 3, 2)$ é gráfica? Justifique.

(b) A sequência $(4, 3, 3, 2, 2, 2, 2)$ é gráfica? Justifique.

(c) A sequência $(5, 4, 4, 3, 2)$ é gráfica? Justifique.

(d) A sequência $(4, 4, 4, 3, 1)$ é gráfica? Justifique.

19⋆. Proponha uma definição para o conceito de *densidade* de um grafo não trivial de tal maneira que se $\rho(G)$ é a densidade de um grafo não trivial G, então

(a) $\rho(G) = 0$ se e somente se $E(G) = \emptyset$.

(b) $\rho(G) = 1$ se e somente se G é completo.

(c) $\rho(G) = 1 - \rho(\overline{G})$, para todo grafo G.

(d) fixado o número de vértices de um grafo, $\rho(G)$ é estritamente crescente em função do número de arestas, isto é, se $|V(G)| = |V(G')|$, então $\rho(G) > \rho(G')$ se e somente se $|E(G)| > |E(G')|$.

Prove que sua definição satisfaz cada uma destas propriedades.

20⋆. Prove que, para todo grafo G temos

$$M_G^2[v, v] = \delta_G(v), \text{ para todo } v \in V(G).$$

21. Exercício intencionalmente deixado em branco

22⋆. Prove que, se G é um grafo direcionado, então

$$\sum_{v \in V(G)} \delta^+(v) = |A(G)| = \sum_{v \in V(G)} \delta^-(v),$$

23⋆. Prove que

$$M_{G^T} = (M_G)^T,$$

para todo grafo direcionado G, onde M^T denota a matriz transposta da matriz M.

24⋆. Prove que, para todo grafo direcionado G temos

$$M_G M_G^T[v, v] = \delta_G^+(v), \text{ para todo } v \in V(G).$$

A.2 Representação Computacional

25⋆. Descreva como computar eficientemente o grafo transposto de um grafo direcionado G, quando G é representado

(a) por listas de adjacência, ou

(b) pela matriz de adjacência.

Expresse a eficiência de ambas as soluções em termos assintóticos em função do tamanho do grafo G.

26⋆. Considere a idéia de evitar o desperdício de espaço na representação da matriz de adjacência de um grafo G com $V(G) = \{1, \ldots, n\}$ usando um vetor m contendo somente os elementos abaixo da diagonal principal de M_G, de tal maneira que

$$M_G[u, v] = \begin{cases} m[f(u, v)], & \text{se } u > v, \\ 0, & \text{se } u = v, \\ m[f(v, u)], & \text{se } u < v, \end{cases}$$

onde $f\colon \{(u, v) \in V(G) \times V(G) \mid u < v\} \to \{0, \ldots, N(n) - 1\}$ é a função que faz a correspondência entre os elementos de M_G e m, e $N(n)$ é o tamanho do vetor m.

(a) Dê uma expressão[2] para $N(n)$.

(b) Dê uma expressão[3] para $f(u, v)$.

(c) Dê uma expressão[4] para $f^{-1}(k)\colon 0 \leq k < N(n)$.

(d) Escreva a função

```
unsigned int vizinho(unsigned int *m, unsigned int u, unsigned int v);
```

que devolve o valor de $M_G[u, v]$, onde M_G é representado pelo vetor `m` tal como descrito acima.

27. Suponha que a representação de um número inteiro consome i bytes e a representação de um apontador consome a bytes e considere o problema de representar um grafo ponderado de n vértices e m arestas nesta linguagem.

[2] **Sugestão:** Descreva $N(n)$ por meio de uma recorrência e daí resolva esta recorrência.
[3] **Sugestão:** Observe que $f(u, 1) = N(u - 1)$ e que $f(u, v) = f(u, 1) + v - 1$.
[4] **Sugestão:** Observe que $f(u, 1) \leq f(u, v) \leq f(u, u - 1)$.

(a) Expresse o total de memória consumido na representação do grafo em função de i, a, n e m ao usar

   i. matriz de adjacência

   ii. listas de adjacência

(b) Indique como decidir, em função de i, a, n e m, qual das duas representações ocupa menos memória.

A.3 Subgrafos

28⋆. Dê um exemplo de um grafo G com subgrafos H_1, H_2 e H_3 tais que

(a) o grafo H_1 é um subgrafo induzido por vértices;

(b) o grafo H_2 é um subgrafo induzido por arestas mas não é um subgrafo induzido por vértices;

(c) o grafo H_3 não é um subgrafo induzido por arestas nem por vértices.

29. Exercício intencionalmente deixado em branco

30. Exercício intencionalmente deixado em branco

31⋆. Seja G um grafo e seja X um conjunto de vértices de G. É verdade que todo subgrafo de G induzido por vértices pode ser obtido a partir de G pela remoção de um conjunto de vértices?

32. Se G é um grafo e α e β são arestas de G, é verdade que

$$(G - \alpha) - \beta = (G - \beta) - \alpha?$$

Justifique.

?

33⋆. Seja G um grafo e seja X um conjunto de arestas de G. É verdade que

$$G - X = G[E(G) - X]?$$

Justifique.

É verdade que todo subgrafo de G induzido por arestas pode ser obtido a partir de G por uma sequência de remoções de arestas?

34. Exercício intencionalmente deixado em branco

35⋆. Seja G um grafo e seja v um vértice de G. Prove que

$$|E(G - v)| = |E(G)| - \delta_G(v).$$

36⋆. Seja G um grafo e sejam $A \subseteq E(G)$ e $V = \bigcup_{\alpha \in A} \alpha$.

É verdade que $G[A] = G[V]$? Justifique.

37⋆. Prove que se G é um grafo, então,

(a) Um conjunto X é independente em G se e somente se X é clique em $\overline{G}$.

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

38⋆. Considere o seguinte algoritmo guloso para computar um conjunto independente de um grafo G.

```
Independente(G)
I ← ∅
Enquanto V(G) ≠ ∅
    v ← vértice de grau mínimo em G
    acrescente v a I
    remova v e Γ_G(v) de G
Devolva I
```

(a) Sejam n e Δ, respectivamente, o número de vértices no grafo G e seu grau máximo ao início do algoritmo. Prove que, ao início do laço, sempre é verdade que[5] $n \leq |V(G)| + |I|(\Delta + 1)$.

[5] **Sugestão:** Para facilitar a expressão do seu raciocínio, defina G_i e I_i como os valores das variáveis G e I no algoritmo ao início da i-ésima iteração

(b) Conclua daí que $\alpha(G) \geq \dfrac{|V(G)|}{\Delta(G)+1}$, para todo grafo G,

(c) Conclua também que $\omega(G) \geq \dfrac{|V(G)|}{|V(G)|-\delta(G)+1}$, para todo grafo G,

39$^{\#}$. Prove que todo grafo com pelo menos 6 vértices tem uma clique ou um conjunto independente com 3 vertices.
[6]

40*. Prove que para todo grafo G, $\omega(G) \leq \Delta(G)+1$.

41*. Seja H um subgrafo de um grafo G. Decida se as afirmações abaixo são verdadeiras ou não, justificando sua resposta em cada caso.

(a) $\alpha(H) \leq \alpha(G)$?
(b) $\alpha(G) \leq \alpha(H)$?
(c) $\omega(G) \leq \omega(H)$?
(d) $\omega(H) \leq \omega(G)$?

42. Exercício intencionalmente deixado em branco

43$^{@}$. Prove que o problema de determinar o tamanho máximo de um conjunto independente de um grafo é um problema $\mathcal{NP}$–Difícil[7].

44*. Prove que o problema de determinar se um grafo dado é subgrafo induzido de outro grafo também dado é um problema $\mathcal{NP}$–Difícil[8].

[6] **Sugestão:**
Pense na seguinte formulação, equivalente à do enunciado: qualquer coloração das arestas de um grafo completo de 6 vértices com 2 cores vai ter um triângulo monocromático como subgrafo.

[7] **Sugestão:** Pense em como resolver o Problema do Conjunto Independente a partir de um algoritmo para este problema

[8] **Sugestão:** Pense em como resolver o Problema do Conjunto Independente a partir de um algoritmo para este problema

45⋆. Prove que um grafo G é bipartido se e somente se $E(G) = \partial_G(X)$ para algum $X \subseteq V(G)$ e que, neste caso, $\{X, V(G) - X\}$ é uma bipartição de G.

46. Exercício intencionalmente deixado em branco

47⋆. Prove que um grafo bipartido de n vértices tem no máximo $\left\lfloor \frac{n^2}{4} \right\rfloor$ arestas.

48. Prove que o complemento de um grafo bipartido G é conexo se e somente se G não é (bipartido) completo.

49⋆. Prove que o problema de determinar o número cromático de um grafo é $\mathcal{NP}$–Difícil.

50⋆. Prove que o algoritmo abaixo nem sempre devolve uma coloração mínima do grafo G.

Colore(G)

$\mathcal{C} \leftarrow \emptyset$
Enquanto *existe vértice de G que não pertence a nenhum conjunto em $\mathcal{C}$*
  $v \leftarrow$ vértice de G que não pertence a nenhum conjunto em $\mathcal{C}$
  Se *v não tem vizinho em algum conjunto $K \in \mathcal{C}$*
    acrescente v ao conjunto K
  Senão
    acrescente o conjunto $\{v\}$ a $\mathcal{C}$
Devolva $\mathcal{C}$

A.4 Passeios, Caminhos e Ciclos

51⋆. Um grafo e seu complemento podem ser

(a) ambos conexos?

(b) ambos desconexos?

Justifique.

52⋆. Descreva em palavras os grafos k–regulares para $k \in \{0, 1, 2\}$.

53@. Prove que um grafo é bipartido se e somente se não tem passeios de paridades diferentes com as mesmas pontas.

54⋆. Prove que todo segmento de caminho mínimo em um grafo G é caminho mínimo em G.

55⋆. Prove que se P e Q são dois caminhos de tamanho máximo em um grafo conexo G, então P e Q tem um vértice em comum.

56⋆. Seja G um grafo conexo e seja P um caminho de tamanho máximo em G. Prove[9] que o tamanho de um caminho de tamanho máximo em $G - V(P)$ é menor que $|P|$.

57⋆. Seja G um grafo direcionado sem laços e seja $m := \min\{\delta^-(G), \delta^+(G)\}$. Prove que

(a) o grafo G tem caminho direcionado de tamanho m.

(b) se $m > 0$, então G tem ciclo direcionado de tamanho pelo menos $m + 1$.

58⋆. Prove que se P é um caminho direcionado maximal em um grafo direcionado G, então

- todos os vizinhos de entrada de seu vértice inicial estão em P, e

[9] **Sugestão:** use o Exercício 55

- todos os vizinhos de saída seu vértice final estão em P.

59⋆. Prove que todo passeio direcionado de tamanho mínimo entre dois vértices de um grafo direcionado é um caminho direcionado.

60⋆. Prove que todo segmento de caminho direcionado mínimo em um grafo G é caminho direcionado mínimo em G.

61⋆. Prove que o grafo condensado de um grafo direcionado é um grafo direcionado acíclico.

62⋆. É verdade que existe ciclo em um grafo G se e somente se existem passeios distintos com as mesmas pontas em G? Justifique.

63⋆. É verdade que existe ciclo num grafo se e somente se existe um passeio fechado nesse grafo? Justifique.

64⋆. Prove que $\gamma(G) > 3$ se e somente se as vizinhanças de u e v são disjuntas para toda aresta $\{u, v\} \in E(G)$.

65⋆. Prove que os componentes de um grafo G são todos caminhos ou ciclos se e somente se $\Delta(G) \leq 2$.

66⋆. Prove que todo grafo G tem

(a) caminho de tamanho (pelo menos) $\delta(G)$ e,

(b) ciclo de tamanho pelo menos $\delta(G) + 1$, se $\delta(G) \geq 2$.

67⋆. Prove que[10], se $k > 1$, então todo grafo k–regular tem

(a) um caminho de tamanho k;

(b) um ciclo de tamanho pelo menos $k + 1$.

[10] **Sugestão:** Use o Exercício 66.

68*. Prove que se um grafo G não é acíclico, então

$$\operatorname{diam}(G) \geq \left\lfloor \frac{\gamma(G)}{2} \right\rfloor .$$

69*. Prove que todo grafo direcionado acíclico tem fonte.

A.5 Árvores, Florestas e Arborescências

70⋆. Prove que todo grafo conexo com n vértices e $n-1$ arestas é árvore[11].

71⋆. Prove que o grafo G é uma floresta se e somente se

$$|E(G)| = |V(G)| - |\mathcal{C}(G)|.$$

72⋆. Prove que toda árvore T tem (pelo menos) $\Delta(T)$ folhas.

73⋆. Prove que se T é uma árvore então

$$|\mathcal{C}(T - v)| = \delta_T(v),$$

para todo $v \in V(T)$.

74⋆. É verdade que todo grafo de n vértices com n (ou mais) arestas tem ciclo? Justifique.

75. Um vértice v é *central* em um grafo G se a maior distância de v a qualquer outro vértice de G é a menor possível, isto é, se

$$\max\{d_G(v,u) \mid u \in V(G)\}$$

é mínimo.

(a) Seja T uma árvore com pelo menos 3 vértices e seja $T' = T - F$, onde F é o conjunto das folhas de T. Prove que T e T' tem os mesmos centros,

(b) Conclua daí que toda a árvore tem um único centro ou tem dois centros vizinhos.

76. Prove que uma sequência $(d_1, d_2, \ldots, d_n)$ de inteiros positivos é sequência gráfica [12] de uma árvore se e somente se $\sum_{i=1}^{n} d_i = 2(n-1)$.

[11] **Sugestão:** Use o Exercício 14.
[12] cfr. Exercício 18

77⋆. Prove que um grafo direcionado G tem arborescência geradora se e somente se G tem um vértice r a partir do qual todo vértice de G é alcançável.

78⋆. Prove que o grafo subjacente de uma árborescência é uma árvore.

79. Exercício intencionalmente deixado em branco

A.6 Cortes e Conectividade

80⋆. É possível que toda aresta de um grafo não trivial seja de corte? Justifique sua resposta e, em caso positivo, caracterize tais grafos.

81⋆. Prove que um grafo G é conexo se e somente se

$$\partial_G(X) \neq \emptyset, \text{ para todo } \emptyset \subset X \subset V(G).$$

82⋆. Prove que um grafo direcionado G é fortemente conexo se e somente se

$$\partial_G^+(X) \neq \emptyset, \text{ para todo } \emptyset \subset X \subset V(G).$$

83⋆. É possível que todo vértice de um grafo não trivial seja de corte? Justifique sua resposta e, em caso positivo, caracterize tais grafos.

84⋆. Dê um exemplo de um grafo conexo G para o qual

$$\kappa(G) < \lambda(G) < \delta(G).$$

85⋆. É verdade que

$$\kappa(G) \leq \lambda(G) \leq \delta(G),$$

para todo grafo conexo G?

Justifique.

86⋆. Seja G um grafo, T uma árvore geradora de G e $v \in V(G)$.

(a) É verdade que se v tem grau maior que 1 em T, então v é vértice de corte em G? Justifique.

(b) É verdade que se v é vértice de corte em G então v tem grau maior que 1 em T? Justifique.

87⋆. Prove que todo vértice de corte em um grafo é vértice de corte em qualquer árvore geradora deste grafo.

88⋆. Considere o jogo em que o jogador recebe um grafo conexo. O objetivo é remover os vértices deste grafo, um a um, sem desconectar o grafo em nenhum momento. O jogador vence se conseguir remover todos os vértices.

(a) Sempre é possível vencer o jogo[13]? Justifique.

(b) Descreva um algoritmo para vencer o jogo nos casos em que é possível vencer.

89⋆. Prove que um vértice de um grafo G faz parte de dois blocos distintos de G se e somente é vértice de corte.

90⋆. É verdade que se u e v são vértices de corte vizinhos em um grafo então a aresta $\{u, v\}$ é de corte neste grafo? Justifique.

91⋆. Um vértice de corte em um grafo conexo pode ser folha de uma árvore geradora deste grafo? Justifique.

[13] **Sugestão:** Use o ex. 83

A.7 Grafos Eulerianos e Hamiltonianos

92⋆. Prove que um grafo conexo tem trilha euleriana aberta se e somente se tem exatamente dois vértices de grau ímpar.

93⋆. Prove que determinar o tamanho do maior ciclo de um grafo é um problema $\mathcal{NP}$-difícil.

94⋆. Seja G um grafo não vazio e seja $v \in V(G)$. Seja G_v o grafo obtido ao acrescentar-se a G três novos vértices, v', u e w e as arestas $\{u, v\}$, $\{w, v'\}$ e $\{\{v', r\} \mid r \in \Gamma_G(v)\}$. Prove que G é hamiltoniano se e somente se G_v tem caminho hamiltoniano.

95⋆. Prove que decidir se um grafo tem caminho hamiltoniano é um problema $\mathcal{NP}$–difícil[14].

96⋆. Prove que o problema de determinar um caminho direcionado de tamanho máximo em um grafo direcionado é um problema $\mathcal{NP}$–difícil.

97⋆. Seja G um grafo e seja $D(G)$ o grafo direcionado dado por

$$\begin{aligned} V(D(G)) &= V(G), \\ A(D(G)) &= \bigcup_{\{u,v\} \in E(G)} \{(u, v), (v, u)\}. \end{aligned}$$

Prove que G é hamiltoniano se e somente se $D(G)$ é hamiltoniano e conclua, a partir daí, que o problema de decidir se um grafo direcionado é hamiltoniano é $\mathcal{NP}$–Difícil.

98⋆. Prove que o problema do Caixeiro Viajante é $\mathcal{NP}$–difícil.

99⋆. Seja G um grafo e seja G' o grafo que se obtém ao acrescentar a G um novo vértice v e uma aresta ligando v a cada vértice de G.

Prove que G' é hamiltoniano se e somente se G tem caminho hamiltoniano.

[14] **Sugestão:** Use o Exercício 94

100⋆. Seja G um grafo direcionado e seja (G', w) um grafo direcionado com pesos nas arestas completo com o mesmo conjunto de vértices de G onde w é dada por

$$w(u,v) = \begin{cases} 0, & \text{se } (u,v) \in A(G), \\ 1, & \text{se } (u,v) \notin A(G) \end{cases}$$

Prove que G é hamiltoniano se e somente se a resposta da instância (G', w) do problema do Caixeiro Viajante tem peso 0.

101⋆. Considere o seguinte algoritmo para o Problema do Caixeiro Viajante

```
CV(G, w)
(i, f) ← arco de peso mínimo em G
P ← (i, f)
Enquanto V(P) ≠ V(G)
    u ← origem de um arco de peso mínimo em ∂⁻(i) fora de V(P)
    v ← destino de um arco de peso mínimo em ∂⁺(f) fora de V(P)
    Se w(u, i) ≤ w(f, v)
        i ← u
        acrescente i ao início de P
    Senão
        f ← v
        acrescente f ao final de P
    acrescente i ao final de P
    Devolva P
```

(a) Mostre que o algoritmo está **errado**, exibindo uma instância do Problema do Caixeiro Viajante para a qual o algoritmo não computa uma resposta correta.

(b) Mostre que o algoritmo está **muito errado**, provando que sua resposta pode ficar arbitrariamente longe da resposta correta, isto é, prove que para todo $n \in \mathbb{N}$ existe uma instância (G, w) do Problema do Caixeiro Viajante tal que

$$\mathrm{OPT}(G, w) < w(\mathsf{CV}(\mathsf{G}, \mathsf{w})) - n,$$

onde $\mathrm{OPT}(G, w)$ denota o peso de uma solução da instância (G, w).

102⋆. Um grafo conexo onde todos os vértices tem grau par pode ter aresta de corte[15]? Justifique.

[15] **Sugestão:** considere um componente de $G - e$ onde G é um grafo como no enunciado

e e é uma aresta de corte em G.

A.8 Busca em Largura

103⋆. Sejam

$$\begin{aligned} A_1 &= \{1,4,8\}, \\ A_2 &= \{1,5,9\}, \\ A_3 &= \{2,4,6\}, \\ A_4 &= \{2,5,7\}, \\ A_5 &= \{3,6,10\} \text{ e} \\ A_6 &= \{3,7,8\} \end{aligned}$$

e seja G o grafo de interseção[16] de $\{A_1, \ldots, A_6\}$.

(a) Faça um desenho deste grafo.

(b) Faça uma busca em largura em G a partir do vértice A_1 e apresente

i. os níveis de cada vértice, e

ii. as arestas que não estão na árvore enraizada resultante desta busca.

(c) Baseado nesta busca, diga se G é conexo. Justifique sua resposta.

(d) Baseado nesta busca, diga se G é bipartido. Justifique sua resposta.

104. Exercício intencionalmente deixado em branco

105⋆. Seja T uma árvore de distâncias mínimas de raiz r de um grafo G. Prove que

$$\text{diam}(T) \leq 2\text{diam}(G).$$

106⋆. Prove que se (T, r) é a árvore enraizada resultante de uma busca em largura em um grafo conexo G, então rTv é um caminho mínimo em G para todo $v \in V(G)$.

107⋆. Prove que se (T, r) é árvore enraizada resultante de uma busca em largura sobre um grafo conexo G, então

$$|d_G(r,u) - d_G(r,v)| \leq 1, \text{ para todo } \{u,v\} \in E(G-T).$$

[16] Veja o Exercício 7.

108⋆. Seja (T, r) a árvore enraizada resultante de uma busca em largura sobre um grafo conexo G. Prove que G é bipartido se e somente se $d(r, u)$ e $d(r, v)$ tem paridades diferentes para toda aresta $\{u, v\} \in G - E(T)$.[17]

109⋆. Proponha um algoritmo que recebe um grafo de n vértices e m arestas e responde se este grafo é bipartido em tempo $O(n + m)$.

110⋆. Seja (T, r) a árvore enraizada resultante de uma busca em largura sobre um grafo conexo G. Prove que G é bipartido se e somente se $d(r, u) \neq d(r, v)$ para toda aresta $\{u, v\} \in G - E(T)$.

111⋆. Proponha um algoritmo baseado no algoritmo de busca em largura para detectar um ciclo de tamanho ímpar em um grafo, caso exista.

112⋆. Prove que se F é a floresta direcionada resultante de uma busca em largura então toda aresta em $G - S(F)$ é cruzada com relação a F.

113⋆. Prove que se T é a árvore enraizada de raiz r resultante de uma busca em largura em um grafo conexo G, então

$$d_T(r, v) = d_G(r, v), \text{ para todo } v \in V(G).$$

114⋆. Caracterize as árvores enraizadas resultantes de uma busca em largura em um grafo bipartido completo.

A.9 Os Algoritmos de Prim e Dijkstra

115⋆. É verdade que se um grafo com pesos (G, w) é tal que os pesos das arestas são todos distintos entre si então o grafo tem uma única árvore geradora mínima? Justifique.

116⋆. Prove que se (G, w) é um grafo conexo com pesos não negativos então existe uma árvore de caminhos mínimos geradora de G enraizada em v para todo $v \in V(G)$.

[17] **Sugestão:** Use os resultados dos Exercícios 106 e 107

117⋆. Seja T uma árvore de distâncias mínimas de raiz r de um grafo com pesos (G, w). Prove que

$$\text{diam}(T) \leq 2\text{diam}(G).$$

118⋆. O seguinte algoritmo foi proposto para determinar um caminho de comprimento máximo em um grafo conexo com pesos positivos. O algoritmo está correto? Justifique.

CaminhoMaisComprido(G, w)

$P \leftarrow$ caminho vazio
Para *cada* $r \in V(G)$
  $(T, r) \leftarrow$
    árvore enraizada obtida pelo Algoritmo de Dijkstra a partir do vértice r
  $u \leftarrow$ vértice mais distante de r em T
  $v \leftarrow$ vértice de $V(T) - V(rTu)$ mais distante de r em T
  Se $|uTv| > |P|$
    $P \leftarrow uTv$
Devolva P

A.10 Busca em Profundidade

119⋆. Caracterize

(a) as árvores enraizadas produzidas por busca em largura em um grafo completo, e

(b) as árvores enraizadas produzidas por busca em profundidade em um grafo completo.

120⋆. Caracterize as árvores enraizadas resultantes de uma busca em profundidade de um grafo bipartido completo.

121⋆. Um estudante propõe o seguinte algoritmo para encontrar um caminho de comprimento máximo em um grafo.

```
CaminhoMaisLongo(G)
encontre um vértice r de G que não é de corte
(T, r) ←
  árvore enraizada de busca em profundidade em G a partir de r
Devolva caminho de distância máxima de r a uma folha de T
```

O algoritmo está correto? Justifique.

122⋆. Considere a seguinte ideia para determinar se um grafo de n vértices tem ciclo hamiltoniano.

> Se o grafo tem ciclo hamiltoniano, então uma busca em profundidade "iniciada pelo vértice certo" vai encontrá-lo, pois haverá um vértice v a distância $n-1$ do vértice inicial r e v será vizinho de r.
>
> Para garantir que o ciclo hamiltoniano seja encontrado, será suficiente executar n buscas em profundidade, cada uma iniciando por um dos vértices do grafo. Se nenhuma delas localizar um ciclo hamiltoniano é porque o grafo não tem ciclo hamiltoniano.

A ideia está correta? Justifique sua resposta.

A.11 Busca em Grafos Direcionados

123*. Explique como decidir em tempo $O(n + m)$ se um grafo direcionado com n vértices e m arcos é acíclico e, em caso negativo, encontrar um ciclo direcionado.

124*. Apresente um algoritmo que recebe um grafo direcionado G e devolve a arborescência resultante de uma busca em largura em G^T. Para cada $v \in V(G)$, as vizinhanças de entrada e saída de v estão disponíveis.

125*. Apresente um algoritmo que recebe um grafo direcionado G e devolve a arborescência resultante de uma busca em profundidade em G^T. Para cada $v \in V(G)$, as vizinhanças de entrada e saída de v estão disponíveis.

126*. Seja G um grafo direcionado e considere uma execução do algoritmo $\mathsf{BuscaProfundidade}(G)$. Seja F a floresta direcionada induzida pelos valores de $v.\mathsf{pai} \mid v \in V(G)$ e, para cada $v \in V(G)$ sejam $v.\mathsf{pre}$ e $v.\mathsf{pos}$ os índices de pré–ordem e pós–ordem computados.

Prove que

(a) o arco (u, v) é arco de F ou arco de avanço com relação a F, se e somente se $u.\mathsf{pre} < v.\mathsf{pre} < v.\mathsf{pos} < u.\mathsf{pos}$.

(b) o arco (u, v) é arco cruzado com relação a F, se e somente se $v.\mathsf{pre} < v.\mathsf{pos} < u.\mathsf{pre} < u.\mathsf{pos}$.

(c) o arco (u, v) é arco de retorno com relação a F, se e somente se $v.\mathsf{pre} < u.\mathsf{pre} < u.\mathsf{pos} < v.\mathsf{pos}$.

(d) A ordem $<$ induzida sobre $V(G)$ dada por

$$u < v := u.\mathsf{pre} < v.\mathsf{pre}, \text{ para todo } u, v \in V(G),$$

é uma pré–ordem de F.

(e) A ordem $<$ induzida sobre $V(G)$ dada por

$$u < v := u.\mathsf{pos} < v.\mathsf{pos}, \text{ para todo } u, v \in V(G),$$

é uma pós–ordem de F.

127⋆. Prove que um grafo direcionado G é acíclico se e somente se qualquer floresta direcionada resultante de uma busca em profundidade sobre G não tem arcos de retorno.

128⋆. Prove que um grafo direcionado G admite ordenação topológica se e somente se é acíclico.

129⋆. Modifique o algoritmo de Ordenação Topológica discutido em aula de maneira que ele receba como entrada um grafo direcionado G e devolva,

(a) uma ordenação topológica de G, se G é acíclico, ou

(b) um ciclo direcionado de G, caso contrário.

130⋆. O seguinte algoritmo, conhecido como *Algoritmo de Kahn*, que recebe um grafo direcionado G e devolve (uma lista com) a ordenação topológica de G ou um subgrafo de G sem fontes, caso G seja cíclico.

Qual o seu tempo de execução no pior caso (em termos assintóticos) em função de $|V(G)|$ e $|E(G)|$?

Com relação ao desempenho de pior caso (em termos assintóticos) como

ele se compara ao algoritmo discutido em aula?

```
OrdenaTopologica(G)
  Se V(G) = ∅
     Devolva ()
  Se G não tem fonte
     Devolva G
  v ← fonte de G
  R ← Ordena(G − v)
  Se R é uma lista
     acrescente v ao início de R
  Devolva R
```

```
Ordena(G)
  Q ← lista vazia
  Enquanto V(G) ≠ ∅
     Se G não tem fonte
        Devolva G
     v ← fonte de G
     remova v de G e enfile em Q
  Devolva Q
```

131⋆. Um estudante argumenta que um algoritmo mais simples do que o estudado para ordenar topologicamente um grafo direcionado seria o que devolve a pré–ordem induzida por uma busca em profundidade sobre o grafo a partir de suas fontes. O estudante está correto? Justifique.

132⋆. Prove que $(v_1, \ldots, v_n)$ é uma ordenação topológica de um grafo direcionado acíclico G se e somente se $(v_n, \ldots, v_1)$ é uma ordenação topológica de G^T.

133⋆. Seja G um grafo direcionado e seja $r \in V(G)$. Sejam ainda T e T', respectivamente, arborescências maximais de G e G^T com raiz r. Prove que $G[V(T) \cap V(T')]$ é componente forte conexo de G.

A.12 Emparelhamentos

134⋆. Seja M um emparelhamento em um grafo G e seja P um caminho M–aumentante. Prove que

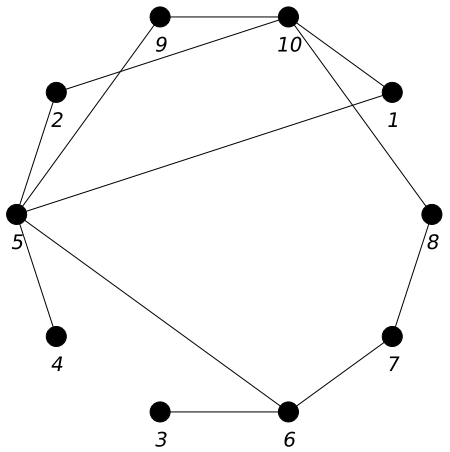
(a) O conjunto $M \oplus E(P)$ é um emparelhamento em G.

(b) $|M \oplus E(P)| = |M| + 1$.

135⋆. Seja G um grafo bipartido e seja $X \subseteq V(G)$. O Teorema de Hall afirma que, se não é possível cobrir o conjunto X por um emparelhamento, então $|\Gamma_G(S)| < |S|$ para algum $S \subseteq X$.

Explique como modificar o algoritmo que computa um emparelhamento máximo num grafo bipartido discutido em aula de maneira que, se o emparelhamento devolvido não cobre todos os vértices de G, então o algoritmo devolve também um conjunto $S \subseteq V(G)$ satisfazendo $|\Gamma_G(S)| < |S|$.

136⋆. Prove, usando o Teorema de Hall, que o grafo abaixo não tem emparelhamento que cobre todos os vértices.



137⋆. Seja G um grafo e sejam M_1 e M_2 dois emparelhamentos em G. É verdade que o grafo $G[M_1 \cup M_2]$ é bipartido? Justifique.

A.13 Algoritmo de Kruskal

138⋆. Considere a seguinte modificação do Algoritmo de Kruskal.

```
FGM(G, w)
F ← (V(G), ∅)
E ← E(G)
Enquanto F tem mais de 1 componente e E não está vazia
    remova de E uma aresta {u, v} de peso mínimo
    Se u e v estão em componentes diferentes de F
        acrescente {u, v} a F
Devolva F
```

É verdade que a execução de FGM(G, w) sempre devolve uma floresta geradora de peso mínimo de G? Justifique.

139⋆. Proponha uma modificação do Algoritmo de Kruskal que computa uma floresta geradora de peso máximo de um grafo dado. Prove que seu algoritmo está correto.

140⋆. Seja $\{u, v\}$ uma aresta de peso mínimo de um grafo com pesos conexo (G, w). Prove que $\{u, v\}$ é aresta de alguma árvore geradora mínima de (G, w).

A.14 Distâncias entre Todos os Pares de Vértices

141⋆. Seja (G, w) um grafo com pesos nas arestas, sendo $V(G) = \{v_1, \ldots, v_n\}$. Prove que

$$d_{G,w}(i,j,k) = \begin{cases} \infty, & \text{se } k = 0 \text{ e } (v_i, v_j) \notin G, \\ w(v_i, v_j), & \text{se } k = 0 \text{ e } (v_i, v_j) \in G, \\ \min\{d_{G,w}(v_i, v_j, k-1), & \\ \quad d_{G,w}(v_i, v_k, k-1) + d_{G,w}(v_k, v_j, k-1)\}, & \text{caso contrário.} \end{cases}$$

para todo $i, j \in [1..n]$ e todo $k \in [0..n]$, onde $d_{G,w}(i, j, k)$ denota o peso de um caminho mínimo de v_i a v_j em (G, w) cujos vértices internos estão em $\{v_1, \ldots, v_k\}$.

142⋆. Seja G um grafo direcionado com pesos nas arestas, sendo $V(G) = \{v_1, \ldots, v_n\}$, e seja M a matriz indexada por $[1..n] \times [1..n] \times [0..n]$ dada por

$$M[i,j,k] = \begin{cases} 0, & \text{se } k = 0 \text{ e } (v_i, v_j) \notin G, \\ 1, & \text{se } k = 0 \text{ e } (v_i, v_j) \in G, \\ M[i,j,k-1] \vee (M[i,k,k-1] \wedge M[k,j,k-1]), & \text{caso contrário.} \end{cases}$$

Prove que $M[i,j,k] = 1$ se e somente se existe caminho direcionado de v_i a v_j em G cujos vértices internos estão em $\{1, \ldots, k\}$, para todo $i, j \in [1..n]$ e todo $k \in [0..n]$.

143⋆. Uma certa biblioteca de rotinas para grafos implementa os algoritmos de Dijkstra e de Floyd–Warshall. Após alguma experimentação um programador concluiu que

$$\begin{aligned} dm \lg n &\leq T_D(n,m), \\ T_F(n) &\leq fn^3, \end{aligned}$$

onde d e f são constantes e $T_D(n,m)$ e $T_F(n)$ são os tempos de execução das implementações dos algoritmos de Dijkstra e de Floyd–Warshall, respectivamente, para um grafo com n vértices e m arestas.

O programador deseja implementar uma nova rotina para computar as distâncias entre todos os pares de vértices de um grafo que escolhe, em função do número de arestas e vértices do grafo, qual dos algoritmos usar.

Como deve ser feito o teste para tomar esta decisão? Justifique sua resposta.

144⋆. Seja G um grafo direcionado com pesos e seja $(v_1, \ldots, v_n)$ uma ordenação de $V(G)$.

Dados $i, j \in [1..n]$ e $k \in [0..n]$, definimos o grafo

$$G(i,j,k) := G[\{v_i, v_j\} \cup \{v_1, \ldots, v_k\}]$$

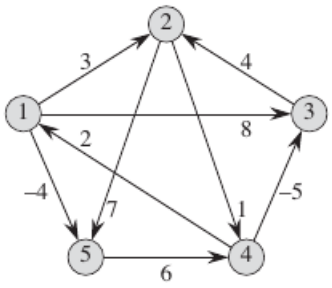
(a) É verdade que os grafos $G(i,j,k)$ são conexos para todo $i, j \in [1..n]$ e todo $k \in [0..n]$? Justifique.

(b) É verdade que P é (i, j, k)-caminho em G se e somente se é caminho em $G(i, j, k)$ para todo $i, j \in [1..n]$ e todo $k \in [0..n]$? Justifique.

145*. Seja G um grafo direcionado conexo com pesos e seja $(v_1, \ldots, v_n)$ uma ordenação de $V(G)$. Dados $i, j \in [1..n]$ e $k \in [0..n]$, definimos o grafo

$$G(i, j, k) := G[\{v_i, v_j\} \cup \{v_1, \ldots, v_k\}]$$

(a) Desenhe os grafos $G(v_1, v_5, k)$ para $0 \leq k \leq 5$, onde G é o grafo abaixo.



(b) É verdade que os grafos $G(i, j, k)$ são conexos para todo $i, j \in [1..n]$ e todo $k \in [0..n]$? Justifique.

A.15 Planaridade

146⋆. Prove que todo grafo planar simples com $n \geq 3$ vértices tem no máximo $3n - 6$ arestas.

147⋆. Prove que todo grafo bipartido planar simples com $n \geq 3$ vértices tem no máximo $2n - 4$ arestas.

148⋆. Prove que todo grafo planar simples tem um vértice de grau no máximo 5.

149⋆. Dê um exemplo de um grafo não trivial, planar e conexo cujo complemento também é planar e conexo.

150⋆. Qual o maior número de vértices que um grafo pode ter para que tanto ele como seu complemento possam ser planares? Justifique.

151⋆. A *densidade*[18] de um grafo G é

$$\rho(G) := \frac{2|E(G)|}{|V(G)|(|V(G)| - 1)}.$$

Prove que a densidade de um grafo planar de n vértices é menor que $6/n$ para todo $n \in \mathbb{N}$

152⋆. Prove que o grafo dual de um grafo euleriano planar é bipartido.

[18] cfr. Exercício 19

Referências Bibliográficas

K. Appel and W. Haken. Every planar map is four colorable. part i: Discharging. *Illinois J. Math.*, 21(3):429–490, September 1977. URL http://projecteuclid.org/euclid.ijm/1256049011.

K. Appel, W. Haken, and J. Koch. Every planar map is four colorable. part ii: Reducibility. *Illinois J. Math.*, 21(3):491–567, September 1977. URL http://projecteuclid.org/euclid.ijm/1256049012.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 3 edition, 2009. ISBN 978-0-262-03384-8. URL http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&tid=11866.

Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller, James W. Thatcher, and Jean D. Bohlinger, editors, *Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations, held March 20–22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, and sponsored by the Office of Naval Research, Mathematics Program, IBM World Trade Corporation, and the IBM Research Mathematical Sciences Department*, pages 85–103, Boston, MA, 1972. Springer US. ISBN 978-1-4684-2001-2. doi: 10.1007/978-1-4684-2001-2_9. URL https://doi.org/10.1007/978-1-4684-2001-2_9.

Jon Kleinberg and Éva Tardos. *Algorithm Design*. Addison-Wesley, 2005. ISBN 0-321-29535-8.