Grafos

Grafo

Um <u>grafo</u> é um par (V, E), onde

- V é o conjunto finito de elementos chamados <u>vértices</u>

- E é um conjunto finito $E \subseteq V \times V$ de pares não ordenados de elementos chamados <u>arestas</u>

Exemplo de Grafos

$$G = (V, E)$$

$$V = \{v_0, v_1, v_2, v_3, v_4, v_5\}$$

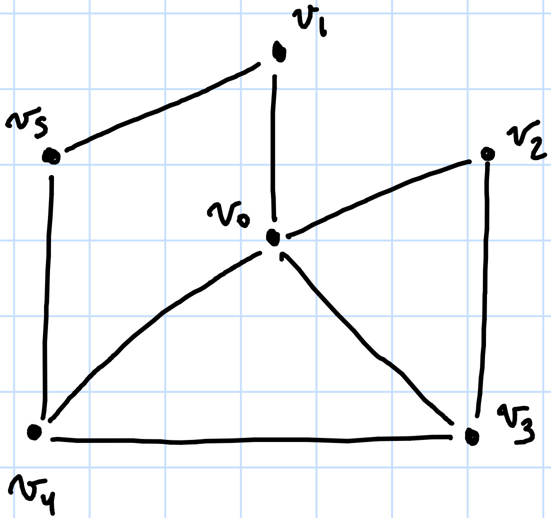
$$E = \{ \{v_0 v_1\}, \{v_0, v_2\}, \{v_0, v_3\}, \{v_0, v_4\},$$
$$\{v_1, v_5\}, \{v_2, v_3\} \{v_3, v_4\}, \{v_4, v_5\} \}$$

Desenho de um Grafo

Grafos possuem uma representação gráfica amigável

- Círculos representam vértices
- Segmentos de retas ligando dois círculos (vértices) representam arestas

Desenho de um Grafo

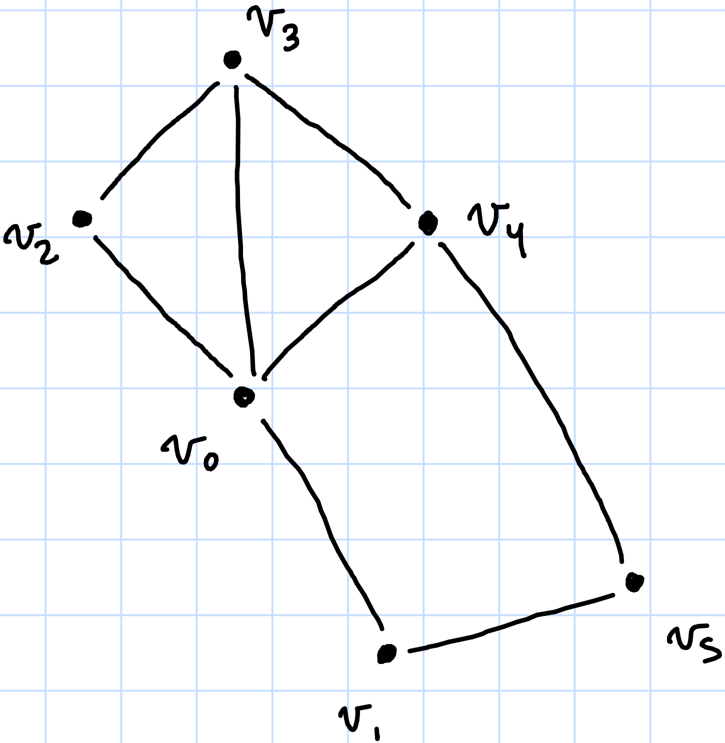


$$G = (V, E)$$

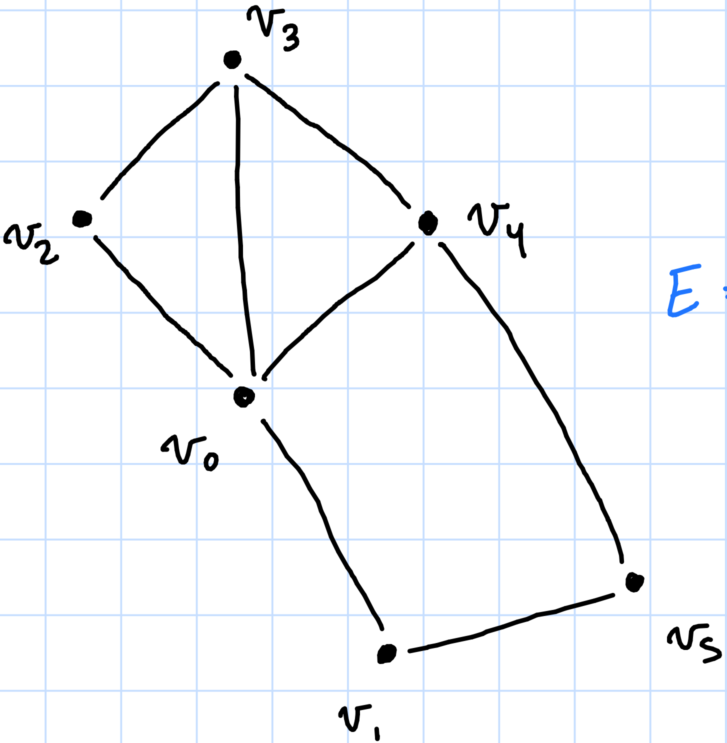
$$V = \{v_0, v_1, v_2, v_3, v_4, v_5\}$$

$$E = \{ \{v_0 v_1\}, \{v_0, v_2\}, \{v_0, v_3\}, \{v_0, v_4\},$$
$$\{v_1, v_5\}, \{v_2, v_3\} \{v_3, v_4\}, \{v_4, v_5\} \}$$

É comum definirmos um grafo pelo seu desenho



É comum definirmos um grafo pelo seu desenho



$$G = (V, E)$$

$$V = \{v_0, v_1, v_2, v_3, v_4\}$$

$$E = \{ \{v_0, v_1\}, \{v_0, v_2\}, \{v_0, v_3\}, \{v_0, v_4\}, \{v_1, v_5\}, \{v_2, v_3\}, \{v_3, v_4\}, \{v_4, v_5\} \}$$

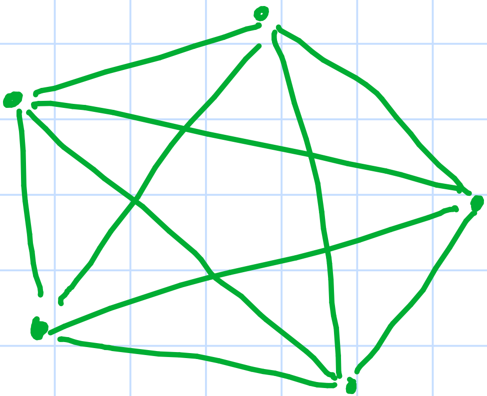
Nomenclatura

Dado um grafo $G=(V,E)$, definimos

- $V(G) = V$
- $E(G) = E$
- $v(G) = |V(G)|$
- $e(G) = |E(G)|$
- Denotemos uma aresta $\{u,v\}$ por uv (ou vu)
- um <u>n-grafo</u> é um grafo com n vértices

Proposição Se G é um n-grafo, então

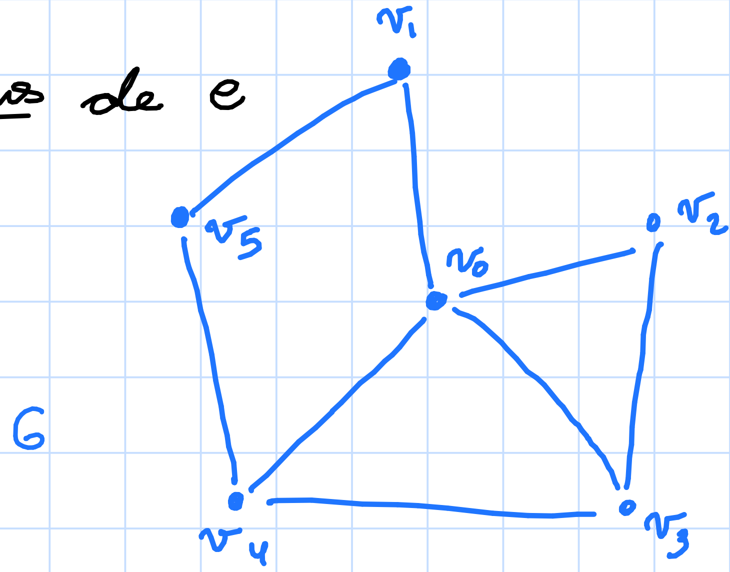
$$e(G) \leq \binom{n}{2} = \frac{n(n-1)}{2}$$



Adjacência e Vizinhança

Se $e \in E(G)$ e se $e = uv$, dizemos:

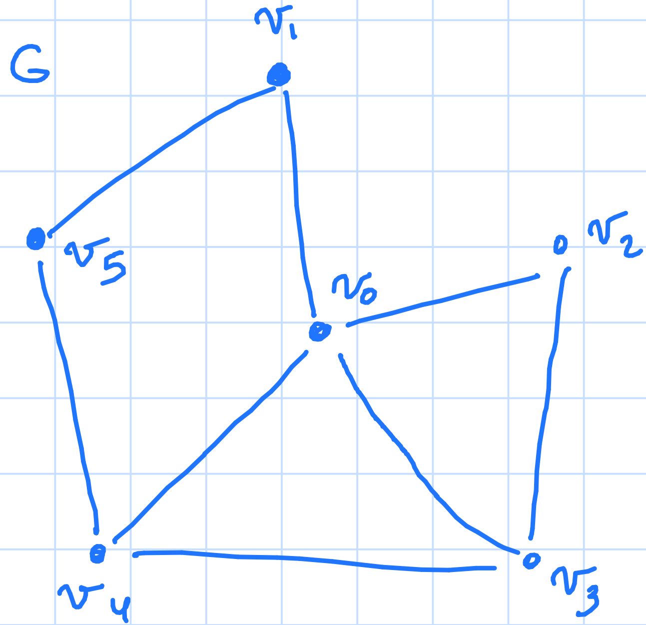
- u e v são <u>vizinhos</u> ou <u>adjacentes</u>
- u é <u>adjacente</u> a v (vice-versa)
- u é <u>vizinho</u> de v
- u e v são <u>extremos</u> de e
- e <u>incide</u> em u



Vizinhança

A <u>vizinhança</u> de um vértice u de um grafo G é

$$N_G(u) = \{ v \in V(G) : uv \in E(G) \}$$

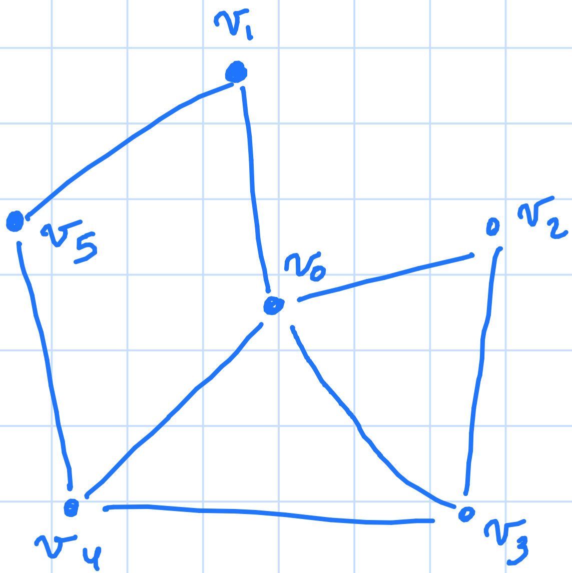


$$N_G(v_3) = \{ v_2, v_0, v_4 \}$$

$$N_G(v_5) = \{ v_1, v_4 \}$$

Grau

O <u>grau</u> de um vértice v de um grafo G, denotado por $d_G(v)$, é o número de arestas incidentes* a v



* laço contar duas vezes

$d_G(v_3) = 4$

$d_G(v_5) = 4$

* Note que $d_G(v) = |N_G(v)|$

* Se $d(v) = 0$, então dizemos que v é um vértice isolado.

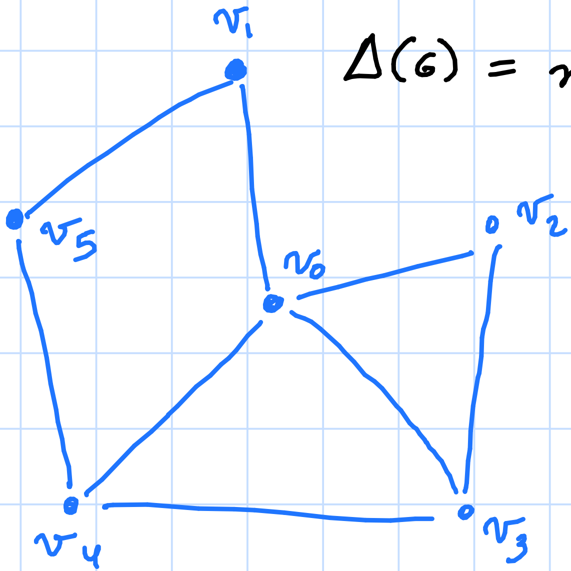
Grau mínimo e máximo

* O **grau mínimo** de um grafo G, denotado por $\delta(G)$, é

$$\delta(G) = \min\{d_G(u) : u \in V(G)\}$$

* O **grau máximo** de um grafo G, denotado por $\Delta(G)$, é

$$\Delta(G) = \max\{d_G(u) : u \in V(G)\}$$



$$\delta(G) = 2$$

$$\Delta(G) = 4$$

Simplificação de Notação

Quando o grafo G é claro pelo contexto

equivalente

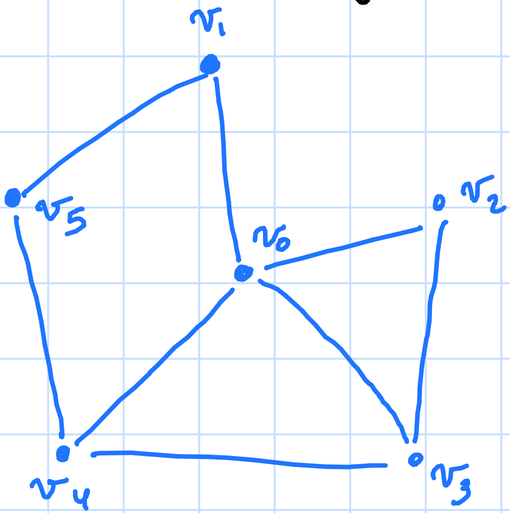
$$N(v) \equiv N_G(v)$$

$$d(v) \equiv d_G(v)$$

Teorema (Aperto de mãos) Para todo grafo G, vale

$$\sum_{v \in V(G)} d_G(v) = 2e(G)$$

Corolário Todo grafo possui um número par de vértices de grau ímpar

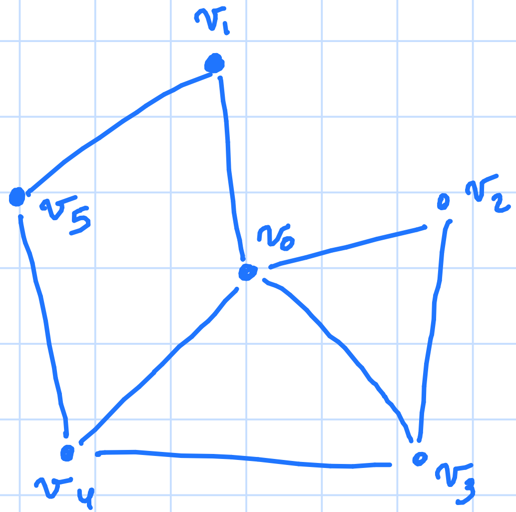


$e(G) =$

$\sum_{v \in V(G)} d_G(v) =$

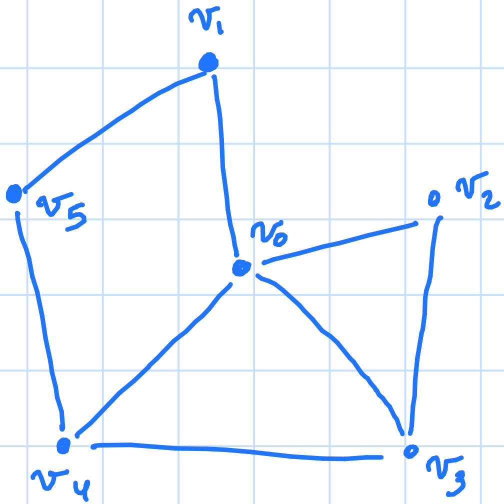
vértices pares =

Teorema Todo grafo possui ao menos dois vértices com o mesmo grau

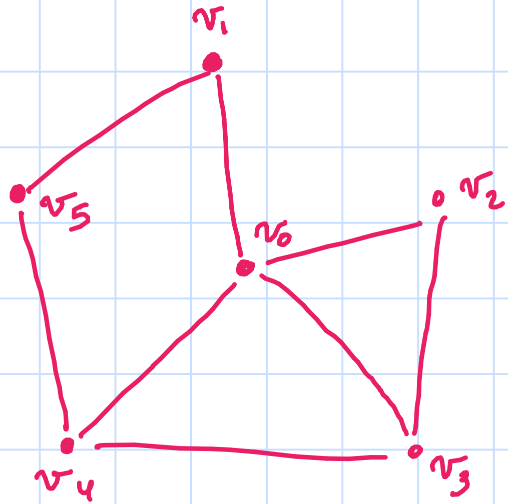


Igualdade Entre grafos

Dois grafos $G=(V, E)$ e $H=(A, B)$ são <u>idênticos</u> se $V = A$ e $E = B$

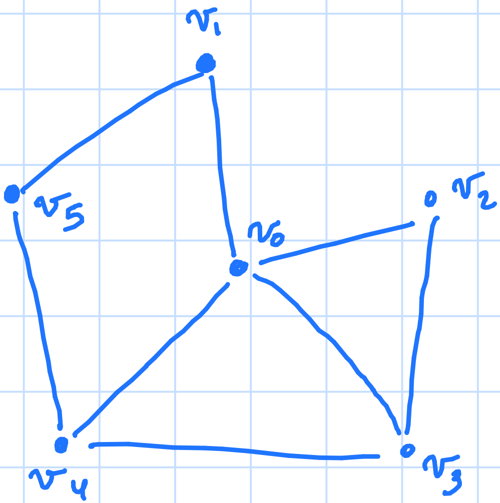


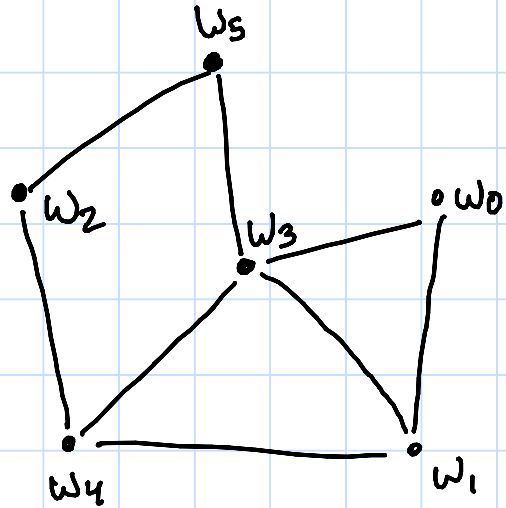
G



H

Isomorfismo



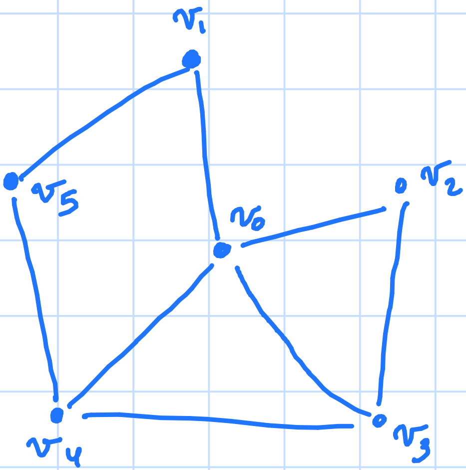


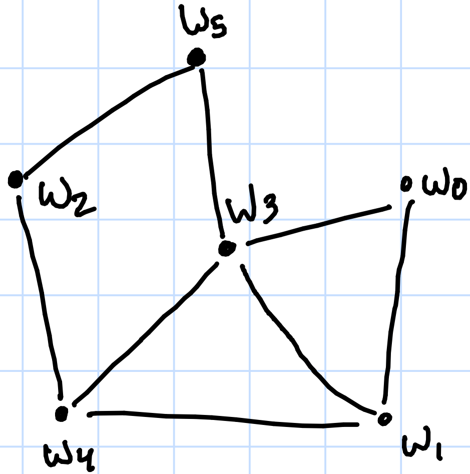
Se os grafos G e H apresentam estruturas idênticas quando ignorados os rótulos dos vértices e arestas, então dizemos que G e H são <u>isomorfos</u> (iso = igual, morfo = forma).

Isomorfismo

Dois grafos simples $G = (V, E)$ e $H = (A, B)$ são <u>isomorfos</u> se existe uma bijeção $\pi : V \to A$ tal que $uv \in E$ sse $\pi(u)\pi(v) \in B$.

G

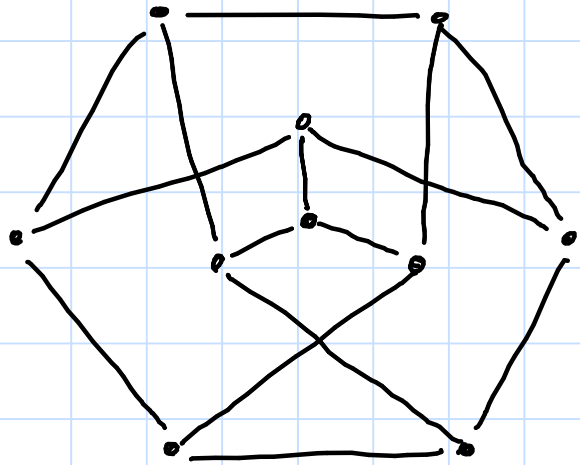




H

$$\pi = \begin{pmatrix} v_0 & v_1 & v_2 & v_3 & v_4 & v_5 \\ w_3 & w_5 & w_0 & w_1 & w_4 & w_2 \end{pmatrix}$$

* Às vezes desenhamos um grafo sem rótulos, para representar qualquer grafo isomorfo ao desenho



* Escrevemos $G \simeq H$ para denotar que o grafo G é isomorfo ao grafo H.

Subgrafos

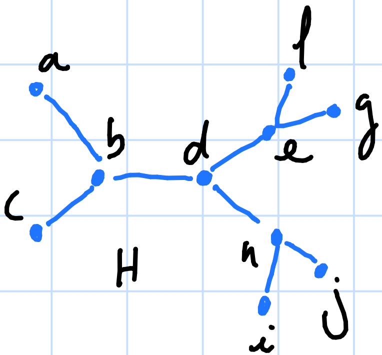
Um grafo H é <u>subgrafo</u> de um grafo G, denotado por $H \subseteq G$, se $V(H) \subseteq V(G)$ e $E(H) \subseteq E(G)$

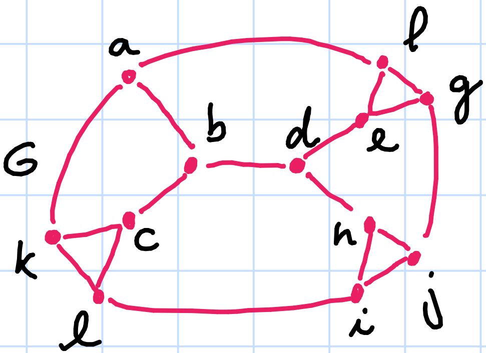
$V(H) = \{a, b, \dots, j\}$

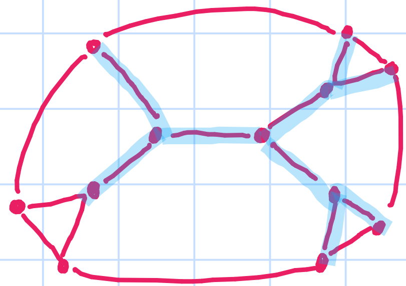
$E(H) = \{ab, bc, bd, de, dh, ef, eg, hi, hj\}$

$V(G) = \{a, b, \dots, l\}$

$E(G) = E(H) \cup \{lk, ak, af, fg, gj, ji, il, ck, cl\}$





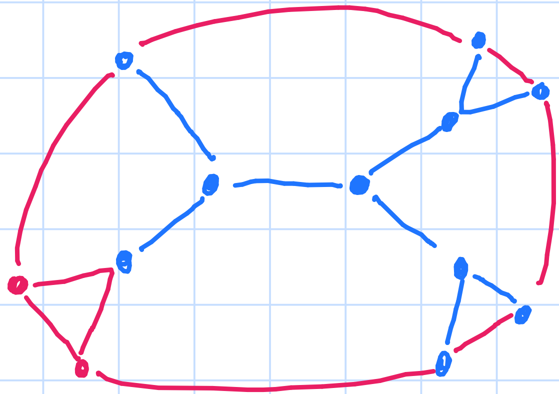


Note que todo grafo é um subgrafo de si.

Subgrafos

As seguintes frases são equivalentes:

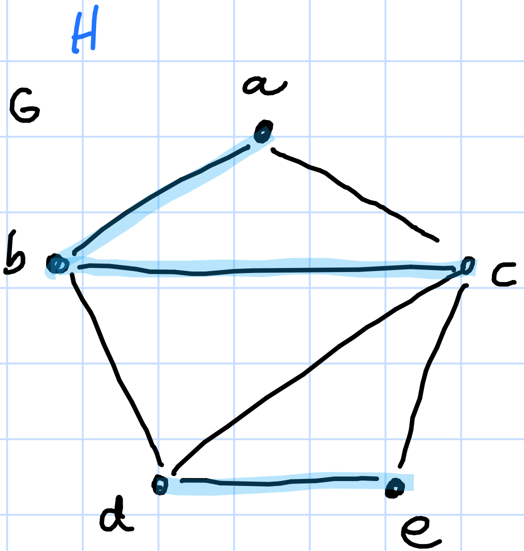
- H é um subgrafo de G
- $H \subseteq G$
- G contém (o grafo) H



Subgrafos

Um subgrafo H de um grafo G é **gerador** se

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



$V(H) = \{a, b, c, d, e\}$
$E(H) = \{ab, bc, de\}$

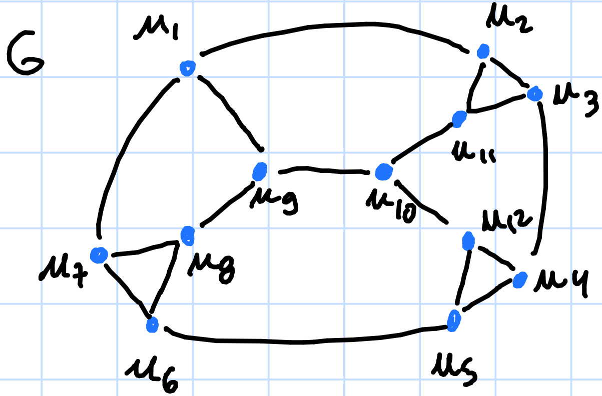
$V(G) = \{a, b, c, d, e\}$
$E(G) = \{ab, ac, bc, bd, cd, ce, de\}$

Operações

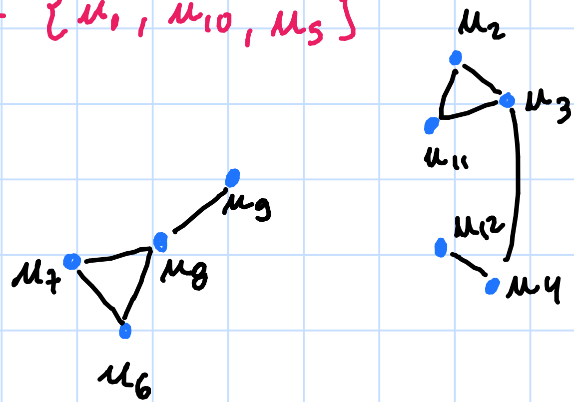
Se S é um subconjunto de vértices de um grafo G, definimos $G - S$ como sendo o grafo

$$V(G-S) = V(G) \setminus S$$

$$E(G-S) = \{uv \in E(G) : u, v \in V(G) \setminus S\}$$



$G - \{u_1, u_{10}, u_5\}$



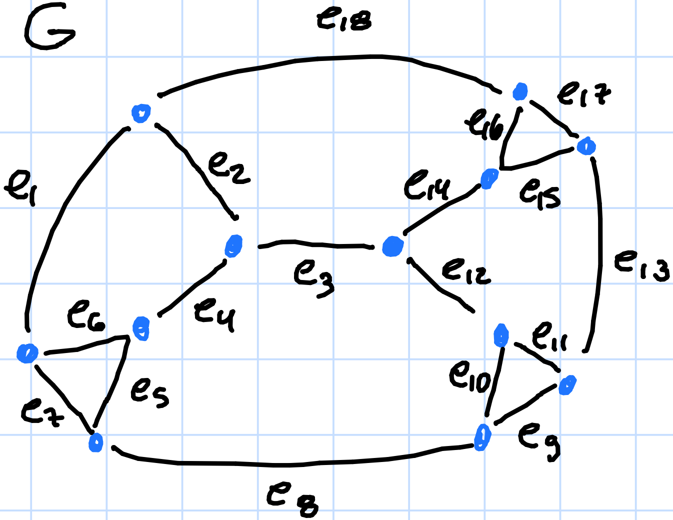
Operações

Se S é um subconjunto de arestas de um grafo G, definimos $G \setminus S$ como sendo o grafo

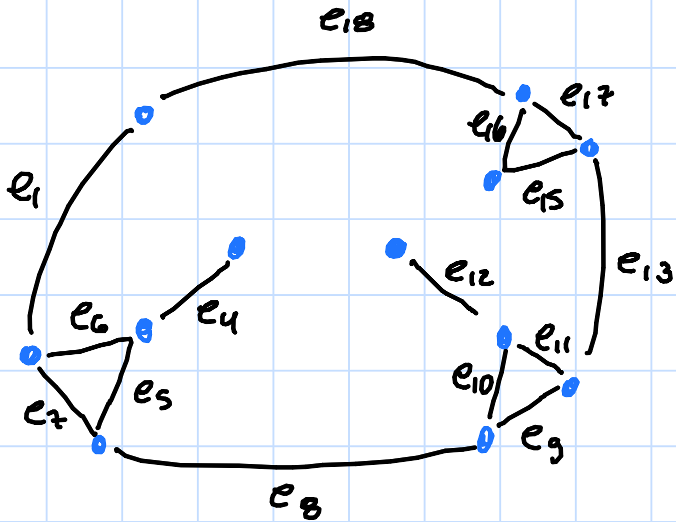
$$V(G\setminus S) = V(G)$$

$$E(G\setminus S) = E(G)\setminus S$$

G



$G \setminus \{e_2, e_3, e_{14}\}$



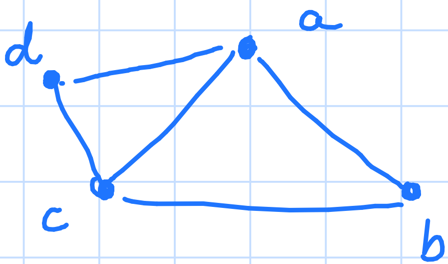
Quando $S = \{x\}$, i.e., S é um conjunto unitário, simplificamos a notação da seguinte maneira

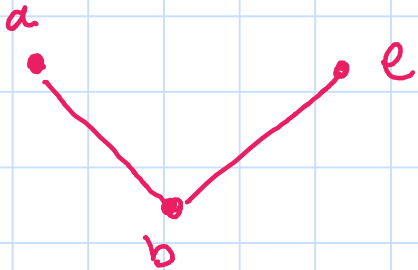
$$G - x \equiv G - S$$
$$G \setminus x \equiv G \setminus S$$

Dados dois grafos $G=(V,E)$ e $H=(A,B)$, definimos a <u>união</u> dos grafos G e H, denotada por $G+H$, como sendo o grafo

$$V(G+H) = V \cup A \qquad E(G+H) = E \cup B$$

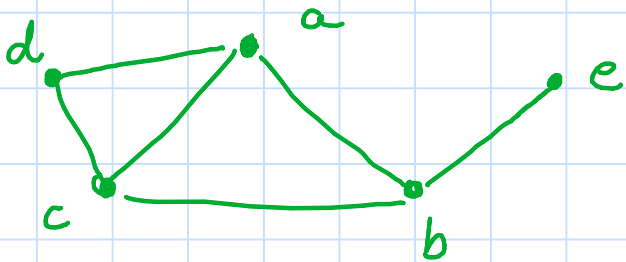
G





H

G+H



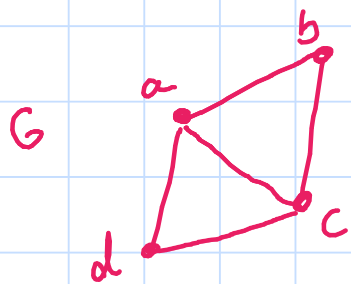
Abuso de notação: é comum tratermos um conj. de arestas $S = \{u_1v_1, u_2v_2, \ldots, u_\ell v_\ell\}$ como sendo o grafo G_S tal que

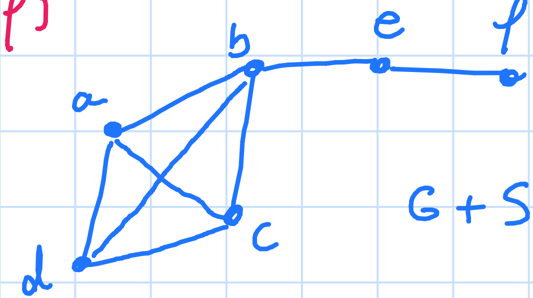
$$V(G_S) = \{u_1, u_2, \ldots, u_\ell, v_1, v_2, \ldots, v_\ell\}$$

$$E(G_S) = S$$

Assim podemos escrever $G+S$ para denotar $G+G_S$

$S = \{bd, be, ef\}$



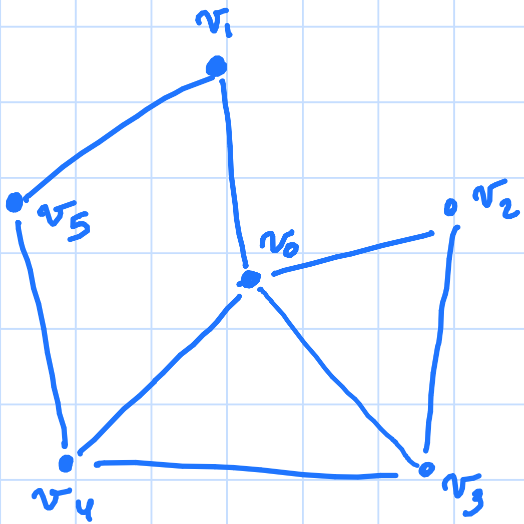


Passeio

Um __passeio__ é uma sequência de vértices $u_0, u_1, u_2, \ldots, u_\ell$ tal que

(i) $u_i \in V(G)$, para $0 \leq i \leq \ell$

(ii) $u_i u_{i+1} \in E(G)$, para $0 \leq i < \ell$



$P = v_0\, v_2\, v_3\, v_0\, v_1\, v_5\, v_1$

Passeio

Se $P = u_0, u_1, u_2, \dots, u_k$ é um passeio, então

- u_0 e u_k são os (vértices) externos / extremos
- u_i, $0 < i < k$, são os <u>vértices internos</u>;
- o <u>comprimento</u> de P é k;
- P é <u>fechado</u> se $u_0 = u_k$ e <u>aberto</u> caso contrário.
- É comum ver P como o grafo (V, E), onde

$$V = \{u_0, u_1, \dots, u_k\}$$

$$E = \{u_0u_1, u_1u_2, u_2u_3, \dots, u_{k-1}u_k\}$$

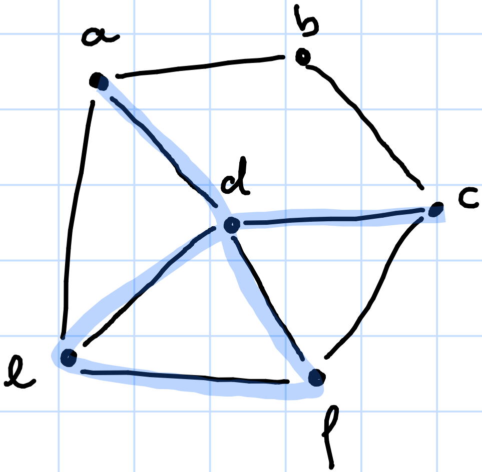
Portanto podemos usar a notação de grafos para P, tais como $V(P)$ e $E(P)$.

Trilha, Passeio e Ciclo

Seja $P = u_0, u_1, u_2, \ldots, u_\ell$ um passeio. Dizemos que

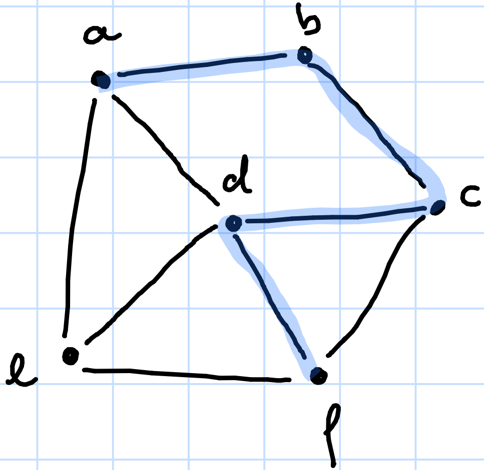
- P é uma trilha se P não repete arestas, i.e., $u_i u_{i+1} \neq u_j u_{j+1}$ para qualquer $i \neq j$
- P é um caminho se P não repete vértices, i.e., $u_i \neq u_j$ para todo $i \neq j$
- P é um ciclo se P $u_i \neq u_j$ para todo $0 \leq i < j < \ell$, $u_0 = u_\ell$ e $\ell \geq 3$.

Trilha, Passeio e Ciclo



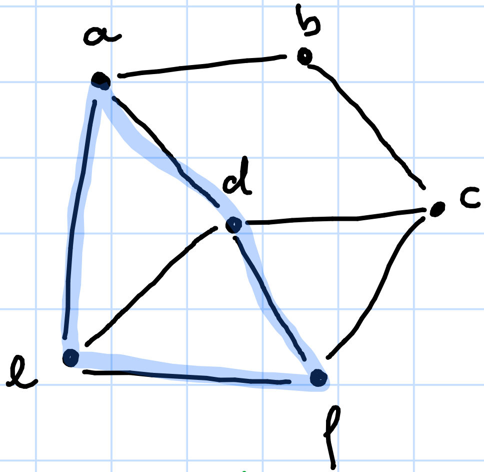
P = a, d, e, f, d, c

Trilha



P = a, b, c, d, f

Passeio

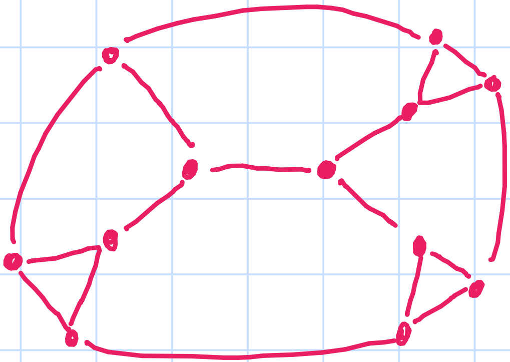


P = a, d, f, e, a

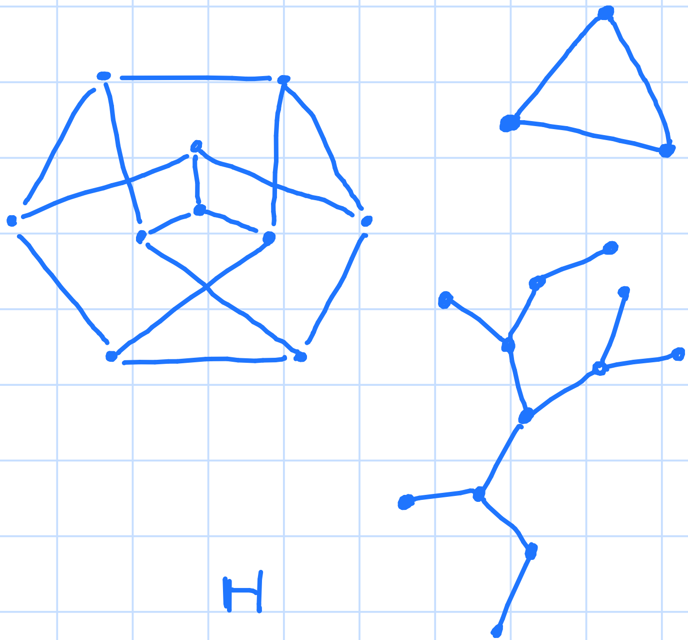
ciclo

Conexidade

Um grafo no qual existe um caminho entre todo par de vértices é chamado de conexo. Um grafo que não é conexo é dito desconexo.



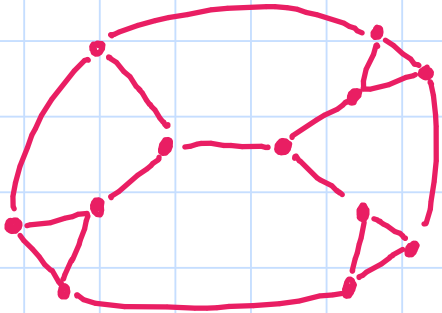
G



H

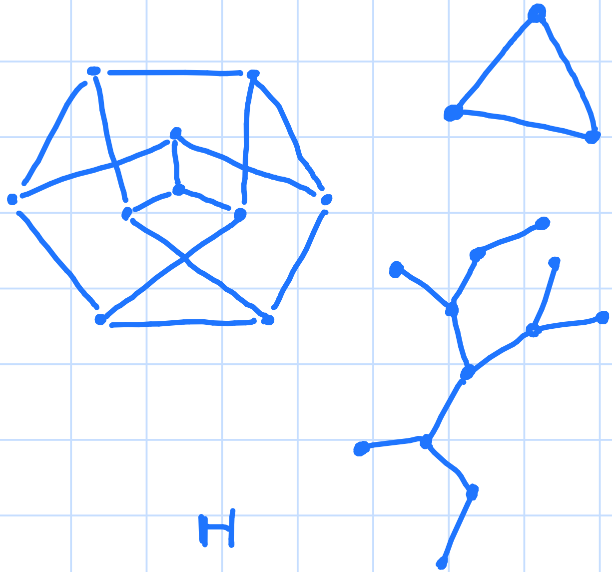
Conexidade

Uma componente conexa é um subgrafo conexo maximal.



G

1 componente conexa



H

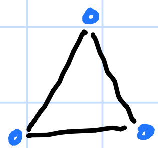
3 componentes conexas

Grafo Completo

O grafo completo de ordem n, denotado por K_n, é o grafo simples tal que $V(K_n)$ é uma clique.

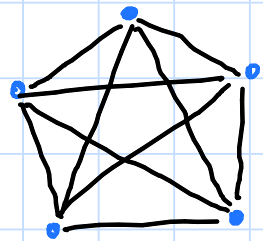
K_1

K_2

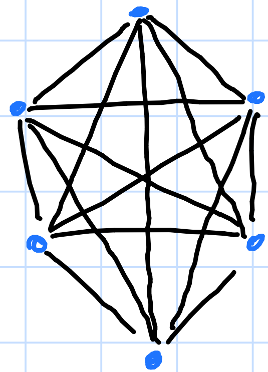


K_3

triângulo



K_5

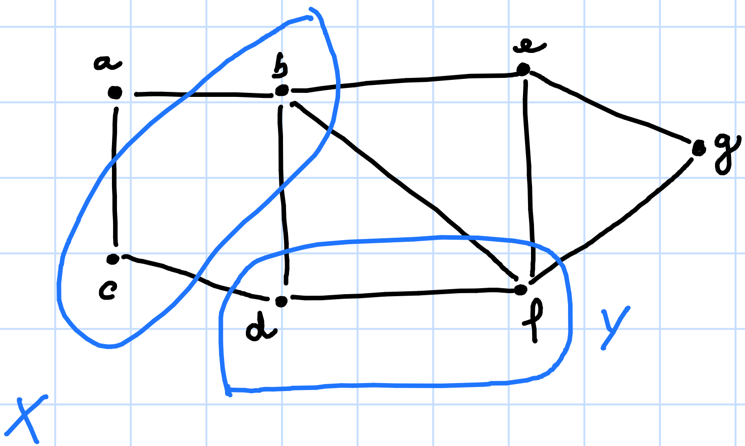


K_6

Corte (X,Y)

Dados um grafo G e dois conjuntos $X, Y \subseteq V(G)$ tais que $X \cap Y = \emptyset$, o corte (X,Y) de G, denotado por $E(X,Y)$, é o conjunto

$$\{xy \in E(G) : x \in X \text{ e } y \in Y\}$$



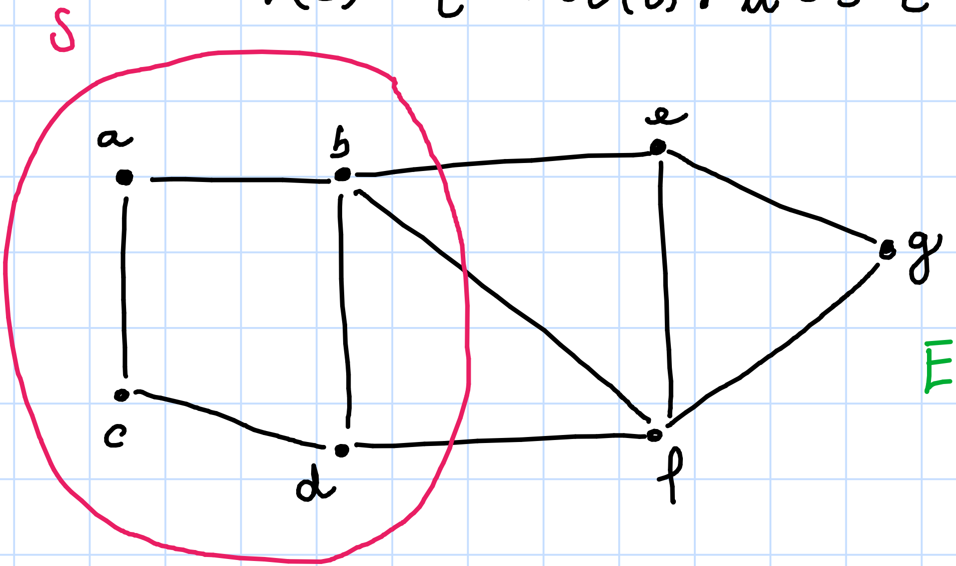
$X = \{b, c\}$, $Y = \{d, f\}$

$E(X,Y) = \{bf, bd, cd\}$

Corte

Dados um grafo G e um conjunto $S \subseteq V(G)$, definimos

$$E(S) = \{uv \in E(G) : u \in S \text{ e } v \in V(G) \setminus S\}$$



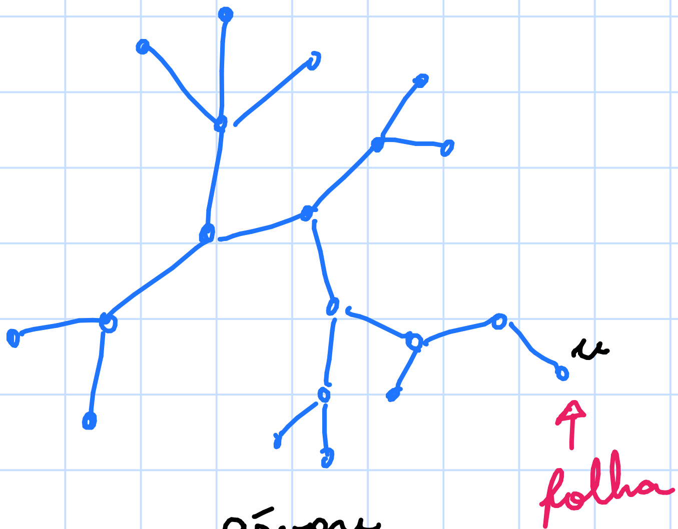
$S = \{a, b, c, d\}$

$E(S) = \{be, bf, df\}$

Dizemos que $E(S)$ é o <u>corte</u> de S

Árvore

dizemos que um grafo é uma __árvore__ se ele não contém ciclos e é conexo.



árvore

- não tem raiz, mas pode ser enraizada

- uma __folha__ é um vértice de grau 1

Teorema Seja G um grafo com n vértices e m arestas. As seguintes afirmações são equivalentes:

ⓐ G é uma árvore

ⓑ Existe um único caminho entre qualquer par de vértices de G

ⓒ G é conexo e para toda aresta $e \in E(G)$, o grafo $G-e$ é desconexo (G é conexo minimal)

ⓓ G é conexo e $m = n-1$

ⓔ G é acíclico e $m = n-1$

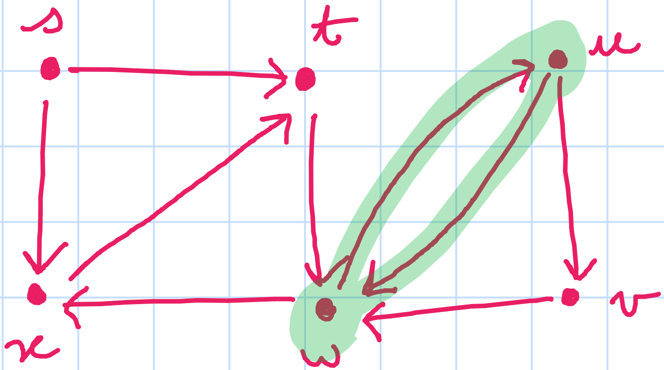
ⓕ G é acíclico e para todo par de vértices $u, v \in V(G)$, o grafo $G+uv$ tem exatamente um ciclo (G é acíclico maximal).

Grafo Direcionado

Grafo Direcionado

Um grafo direcionado é definido de forma semelhante a um grafo, exceto que as arestas consistem de pares ordenados

- chamamos os grafos direcionados simplesmente de dígrafos
- muitas vezes chamamos suas arestas de arcos



DiGrafo

Um digrafo é um par (V, E), onde

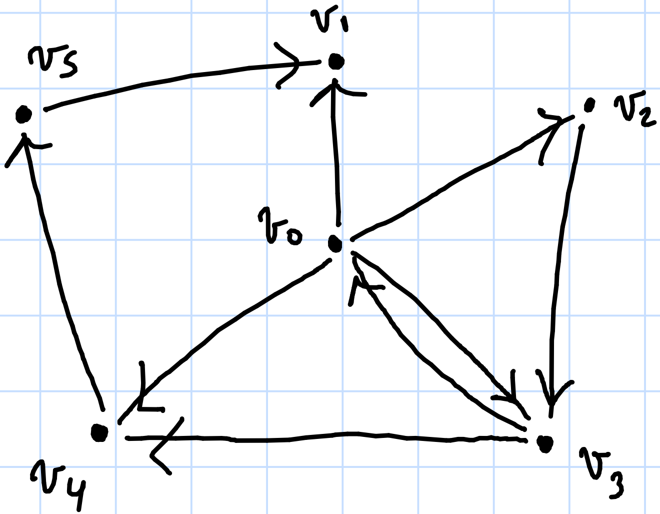
- V é o conjunto finito de elementos chamados vértices

- $E \subseteq V \times V$ é um conj. finito de pares ordenados de elementos chamados arestas

Exemplo de Desenho de um DiGrafo

$G = (V, E)$, onde $V = \{v_0, v_1, v_2, v_3, v_4, v_5\}$,

$E = \{(v_0, v_1), (v_0, v_2), (v_0, v_3), (v_0, v_4),$
$(v_2, v_3), (v_3, v_0), (v_3, v_4), (v_4, v_5),$
$(v_5, v_1)\}$

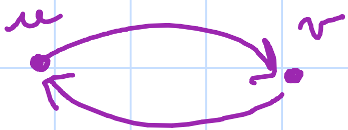


Nomenclatura

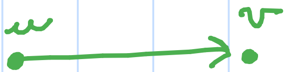
Dado um digrafo $G=(V,E)$, definimos

- $V(G) = V$
- $E(G) = E$
- $v(G) = |V|$
- $e(G) = |E|$

- Denotemos uma aresta (u,v) por uv ~~(ou vu)~~



permitido



$\equiv uv$

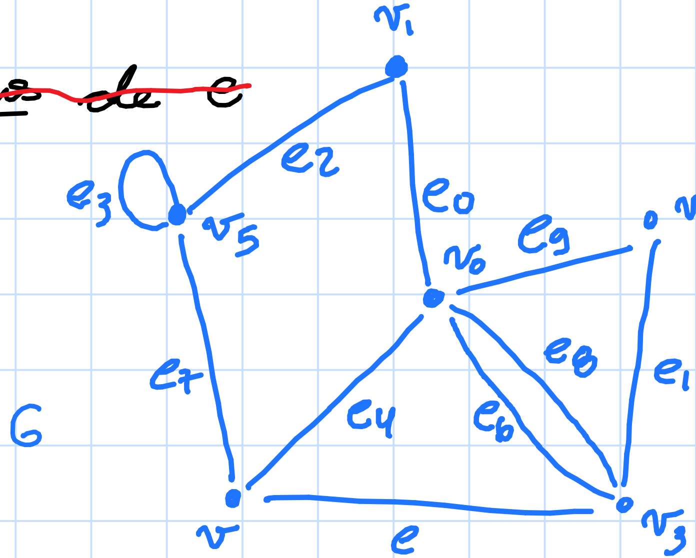
$\equiv uv$ ou vu

Adjacência e Vizinhança

dígrafo

Se $e \in E(G)$ e se $e = uv$, dizemos:

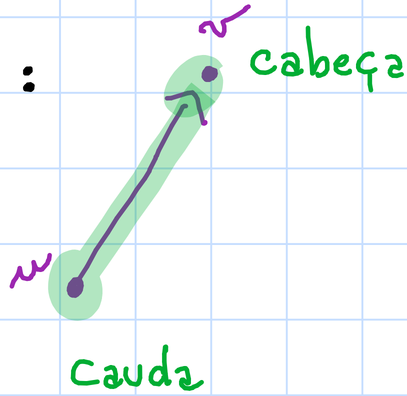
- ~~u e v são vizinhos ou adjacentes~~
- ~~u é adjacente a v (vice-versa)~~
- ~~u é vizinho de v~~
- ~~u e v são extremos de e~~
- ~~e incide em u~~



Adjacência, Vizinhança e grau

dígrafo

Se $e \in E(G)$ e se $e = uv$, dizemos:



- e <u>sai</u> de u e <u>entra</u> em v
- u é <u>cauda</u> de e
- v é <u>cabeça</u> de e

Temos dois tipos de grau para dígrafos

- <u>grau de saída</u> $d^+(u)$ é o número de arestas que saem de u
- <u>grau de entrada</u> $d^-(u)$ é o número de arestas que entram em u

Teorema Para todo digrafo $G=(V,E)$, temos que

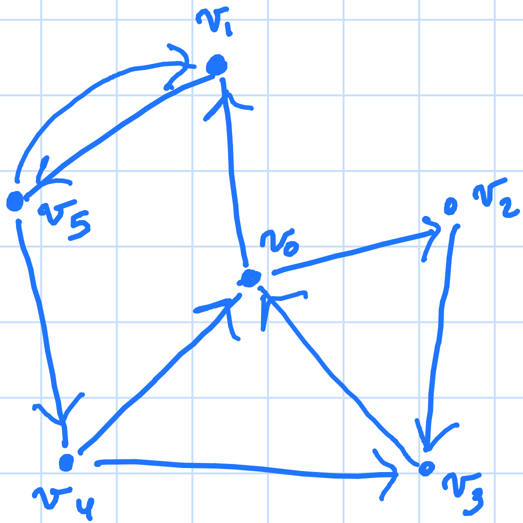
$$\sum_{v\in V} d^+(v) = \sum_{v\in V} d^-(v) = |E|$$

Passeio em Digrafo

Um <u>passeio</u> é uma sequência de vértices $u_0, u_1, u_2, \ldots, u_\ell$ tal que

(i) $u_i \in V(G)$, para $0 \leq i \leq \ell$

(ii) $u_i u_{i+1} \in E(G)$, para $0 \leq i < \ell$



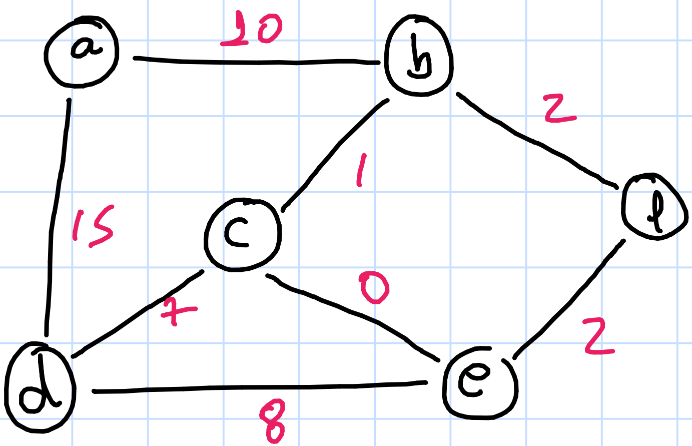
$P = v_0\, v_2\, v_3\, v_0\, v_1\, v_5\, v_1$

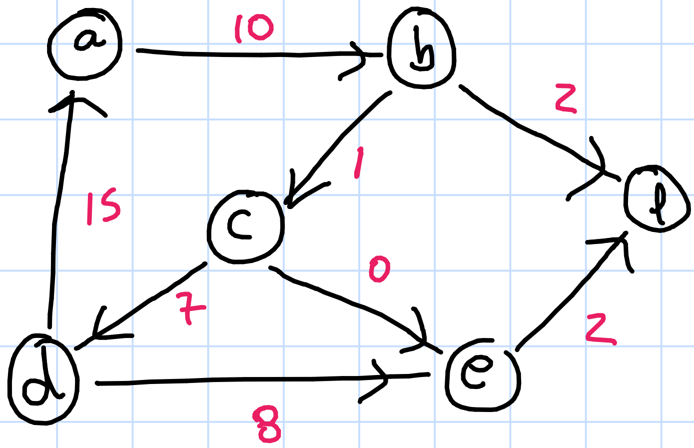
As definições de trilha caminho, ciclo e subgrafo para digrafos também são idênticas as de grafos

(Di)Grafo Ponderado

Dizemos que um (di)grafo é **ponderado** se cada aresta e do (di)grafo está associada a um valor real $w(e)$, denominado **peso** da aresta

- também dizemos que $w(e)$ é o **custo** da aresta





Analisando o Tempo de Execução de alguns Algoritmos Simples de Grafos

Coloração

Uma k-coloração de um grafo G é uma função

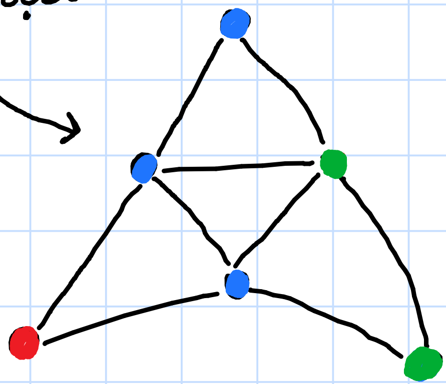
$$c : V(G) \to \{1, 2, \dots, k\}$$

nº representam k cores

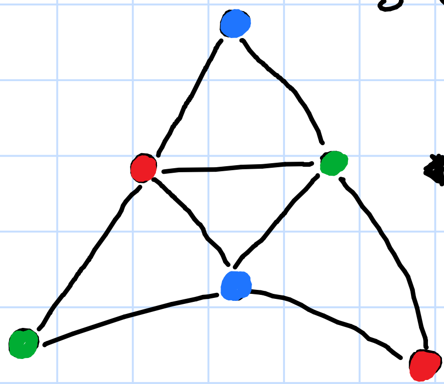
Uma k-coloração $c : V(G) \to \{1, 2, \dots, k\}$ é própria se

$$c(u) \neq c(v) \quad \forall\, uv \in E(G)$$

3-coloração



3-coloração própria



Coloração

Uma k-coloração de um grafo G é uma função

$$c : V(G) \to \{1, 2, \dots, k\}$$

nº representam k cores

Uma k-coloração $c : V(G) \to \{1, 2, \dots, k\}$ é própria se

$$c(u) \neq c(v) \quad \forall\, uv \in E(G)$$

Uma coloração (coloração própria) é uma k-coloração (k-coloração própria) no qual o k ñ é importante.

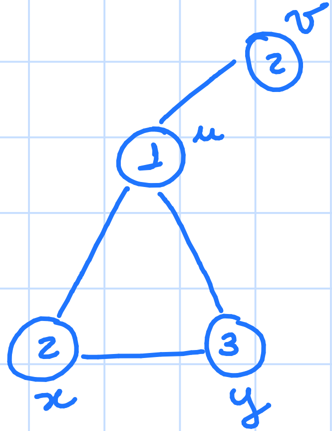
Vc ñ se importa p/ o nº de cores usadas

Verificando se é uma K-coloração Própria

ehColoraçãoPropria (G , c[V(G)], k) — vetor indexado pelos vértices de G

- Um grafo G
- a coloração $c: V(G) \to \{1..k\}$ é dada por um vetor

```
Para u ∈ V(G) Faça
    Se c[u] < 1 ou c[u] > k
        Retorna Falso
Para u ∈ V(G) Faça
    Para v ∈ N(u) Faça
        Se c[u] == c[v] Então
            Retorna Falso
Retorna Verdadeiro
```



	u	v	x	y
c	1	2	2	3

Analisando o tempo: Alg. p/ grafos

- É comum que o pseudo-código de um algoritmo de grafos esteja em um nível mais alto do que o adequado para a análise de tempo

* Isso pode esconder armadilhas

* 1° coisa a se fazer:

Definir como o grafo será representado

- matriz de adj.
- Lista de adj.

Analisando o tempo: Alg. p/ grafos

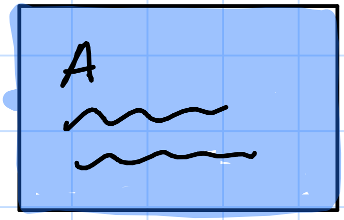
Pseudo-Código Alto nível	Grafo representado como Matriz de adj.
Para $v \in N(u)$ Faça	Para $(v=0; v<n; v{+}{+})$
	Se $G[u][v] == 1$
A	A

	a	u	b	c
a				
u	1	0	0	1
b				
c				

Analisando o tempo: Alg. p/ grafos

Pseudo-Código Alto nível

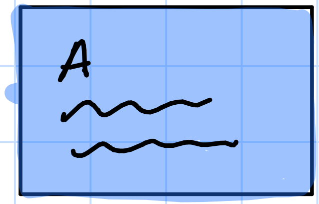
Para $v \in N(u)$ Faça



Grafo representado como Matriz de adj.

Para $(v=0;\ v<n;\ v{+}{+})$

Se $G[u][v] == 1$



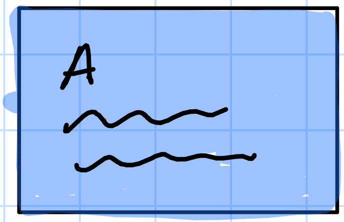
$T(A) = \Theta(1)$

$T(\text{Trecho}) = \Theta(V)$

Analisando o tempo: Alg. p/ grafos

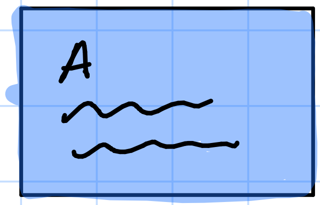
Pseudo-Código Alto nível

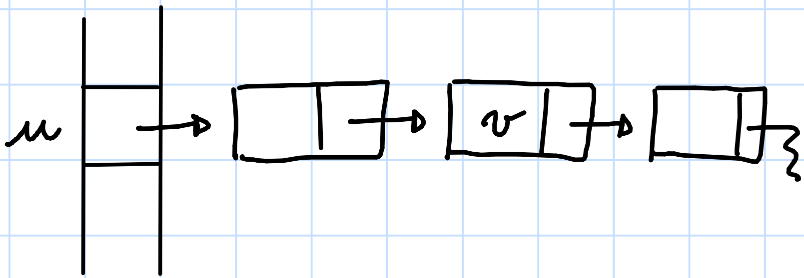
Para $v \in N(u)$ Faça



Grafo representado como Lista de adj.

Para ($v = G[u]$; $v \neq Nil$; $v = v.next$)

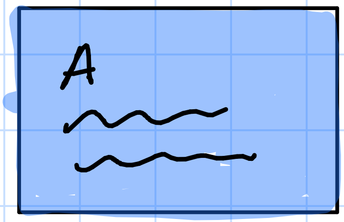




Analisando o tempo: Alg. p/ grafos

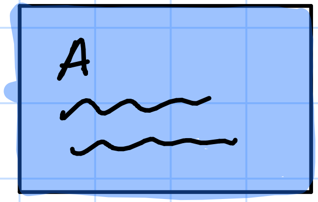
Pseudo-Código Alto nível

Para $v \in N(u)$ Faça



Grafo representado como Lista de adj.

Para ($v = G[u]$; $v \neq Nil$; $v = v.next$)



$T(A) = \Theta(1)$

$T(\text{Trecho}) = \Theta(d(u))$

Analisando o tempo: Alg. p/ grafos

Pseudo-Código Alto nível

Para $v \in N(u)$ Faça

A

Grafo representado como Matriz de adj.

Para ($v=0$; $v<n$; v++)

Se $G[u][v] == 1$

A

$T(A) = \Theta(1)$

$T(\text{Trecho}) = \Theta(V)$

Grafo representado como Lista de adj.

Para ($v = G[u]$; $v \neq Nil$; $v = v.next$)

A

$T(A) = \Theta(1)$

$T(\text{Trecho}) = \Theta(d(u))$

Analisando o tempo: Alg. p/ grafos

Pseudo-Código Alto nível

Para $uv \in E(G)$

A

Grafo representado como Matriz de adj.

Para $(u=0\,;\, u<n\,;\, u{+}{+})$

Para $(v=u+1\,;\, v<n\,;\, v{+}{+})$

Se $G[u][v] == 1$

A

$T(A) = \Theta(1)$

$T(\text{Trecho}) = \Theta(V^2)$

Grafo representado como Lista de adj.

Para $(u=0\,;\, u<n\,;\, u{+}{+})$

Para $(v = G[u]\,;$
$v \neq Nil\,;\, v = v.next)$

A

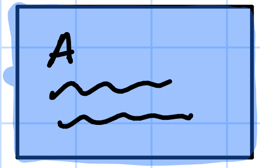
$T(A) = \Theta(1)$

$T(\text{Trecho}) = \Theta(E+V)$

Analisando o tempo: Alg. p/ grafos

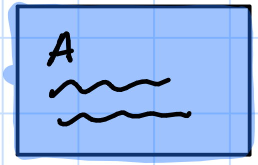
Pseudo-Código Alto nível

Para $u \in V(G)$



Grafo representado como Matriz de adj.

Para ($u=0$; $u<n$; u++)

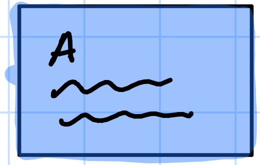


$T(A) = \Theta(1)$

$T(\text{Trecho}) = \Theta(V)$

Grafo representado como Lista de adj.

Para ($u=0$; $u<n$; u++)



$T(A) = \Theta(1)$

$T(\text{Trecho}) = \Theta(V)$

Tempo de Execução: Lista de Adjacências

```
ehColoraçãoPropria ( G , c[V(G)], k)
Para u ∈ V(G) Faça
    Se c[u] < 1 ou c[u] > k
        Retorna Falso
Para u ∈ V(G) Faça
    Para v ∈ N(u) Faça
        Se c[u] == c[v] Então
            Retorna Falso
Retorna Verdadeiro
```

Tempo de Execução: Lista de Adjacências

```
ehColoraçãoPropria ( G , c[V(G)], k )
    Para u ∈ V(G) Faça
        Se c[u] < 1 ou c[u] > k
            Retorna Falso
    Para u ∈ V(G) Faça
        Para v ∈ N(u) Faça
            Se c[u] == c[v] Então
                Retorna Falso
    Retorna Verdadeiro
```

$A = O(V)$

$B = O(V+E)$

$C = \Theta(1)$

$$T(G) = A + B + C = O(V+E)$$

Análise Formalizada

- Nessa análise, vamos assumir que o grafo está representado por uma lista de adjacências.

- O laço da linha 1 executa no máximo V vezes. Logo o tempo gasto ao longo da execução com as linhas 1-3 é $O(V)$

- O laço da linha 4 executa no máximo V vezes. Portanto o tempo gasto com essa linha é $O(V)$.

- Para um u fixo, o laço da linha 5 executa $O(d(u))$ vezes, já que G é representado por uma lista de adjacências. Assim, ao longo da execução do programa, o laço da linha 5 executa

quero essa conta.

$$\sum_{u \in V(G)} O(d(u)) \leq \sum_{u \in V(G)} c \cdot d(u) = c \sum_{u \in V(G)} d(u) = c\, 2E$$
$$= O(E) \text{ vezes}$$

- O tempo para executar as linhas 5-7 uma única vez é $O(1)$. Como o laço da linha 5 executa $O(E)$ vezes, temos que o tempo gasto com esse trecho é $O(1) \cdot O(E) = O(E)$.

- O custo com a linha 8 é $O(1)$.

- Assim, o custo total do código é $O(V+E)$.

$\square$

Tempo de Execução: Matriz de Adjacências

```
ehColoraçãoPropria ( G , c[V(G)], k)
Para u ∈ V(G) Faça
    Se c[u] < 1 ou c[u] > k
        Retorna Falso
Para u ∈ V(G) Faça
    Para v ∈ N(u) Faça
        Se c[u] == c[v] Então
            Retorna Falso
Retorna Verdadeiro
```

Tempo de Execução: Matriz de Adjacências

```
ehColoraçãoPropria ( G, c[V(G)], k )
Para u ∈ V(G) Faça
    Se c[u] < 1 ou c[u] > k          A = O(V)
        Retorna Falso
Para u ∈ V(G) Faça                   B = O(V)
    Para v ∈ N(u) Faça               C = O(V)·O(V) = O(V²)
        Se c[u] == c[v] Então        D = O(V²)
            Retorna Falso
Retorna Verdadeiro                   E = O(1)
```

$$T(G) = A + B + C + D + E = O(V^2)$$

Análise Formalizada

- Nessa análise, vamos assumir que o grafo está representado por uma matriz de adjacências.

- O laço da linha 1 executa no máximo V vezes. Logo o tempo gasto ao longo da execução com as linhas 1-3 é $O(V)$

- O laço da linha 4 executa no máximo V vezes. Portanto o tempo gasto com essa linha é $O(V)$.

- Para um u fixo, o laço da linha 5 executa $O(V)$ vezes, já que G é representado por uma matriz de adjacências. Assim, ao longo da execução do programa, o laço da linha 5 executa

$$\sum_{u \in V(G)} O(V) \leq \sum_{u \in V(G)} c \cdot V = c\, V \sum_{u \in V(G)} 1 = c\, V^2 = O(V^2) \text{ vezes}$$

- O tempo para executar as linhas 5-7 uma única vez é $O(1)$. Como o laço da linha 5 executa $O(V^2)$ vezes, temos que o tempo gasto com esse trecho é $O(1) \cdot O(V^2) = O(V^2)$.

- O custo com a linha 8 é $O(1)$.

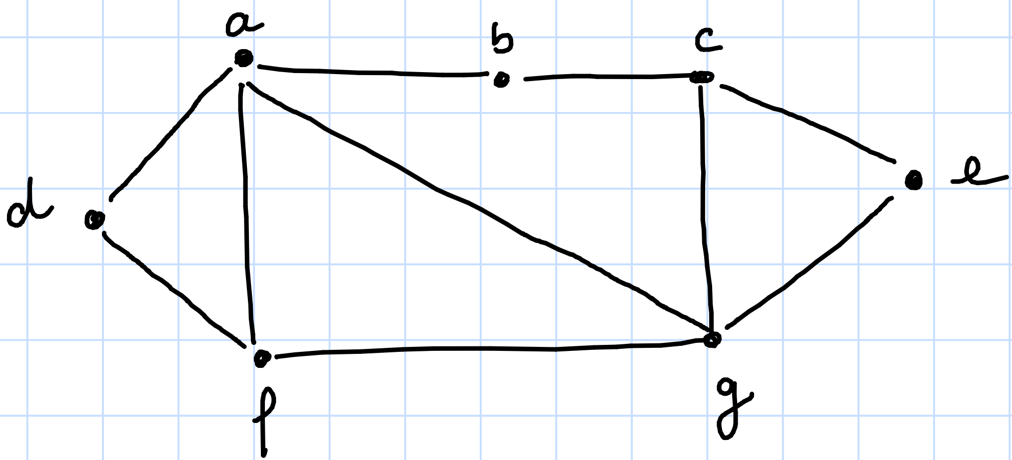
- Assim, o custo total do código é $O(V^2)$.

$\square$

Distância

Dados dois vértices u e v em um grafo G, a distância entre u e v, denotado por $\mathrm{dist}_G(u,v)$, é o menor comprimento de um caminho entre u e v.

número de arestas

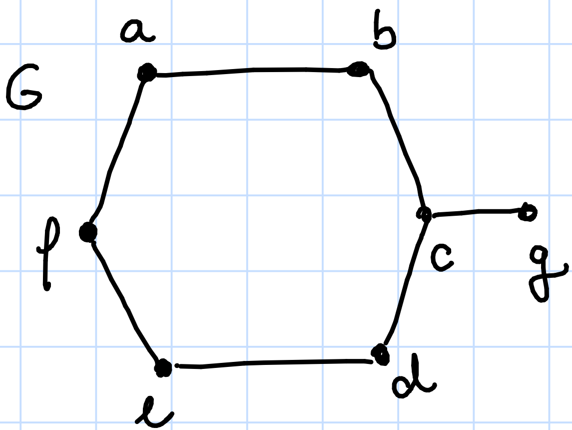


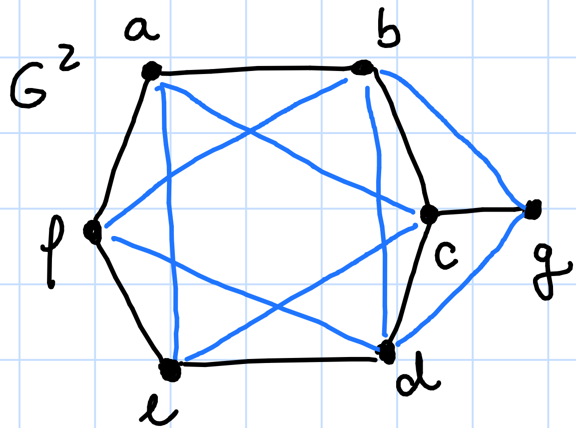
$\mathrm{dist}_G(a,e) = 2$

G^2

dado um grafo G, o grafo G^2 é o grafo definido como

$$V(G^2) = V(G)$$
$$E(G^2) = \{uv : \text{dist}_G(u,v) \leq 2\}$$





Algoritmo

Quadrado(G):

G^2 = copiaGrafo(G)
para $u \in V(G)$
    para $v \in N_G(u)$
        para $w \in N_G(v)$
            $G^2 = G^2 + uw$
Devolva G^2

Análise

Quadrado(G):

G^2 = copiaGrafo(G)
para $u \in V(G)$
    para $v \in N_G(u)$
        Para $w \in N_G(v)$
            $G^2 = G^2 + uw$
Devolva G^2

Análise: lista de Adjacências

$$T(G) = A + B + C + E + F = O(V + E + VE)$$

$$= O(VE)$$

Quadrado(G):

1 $G^2 = \text{copiaGrafo}(G)$ | $A = O(V+E)$
2 para $u \in V(G)$ | $B = O(V)$
3 para $v \in N_G(u)$ | $C = O(E)$
4 Para $w \in N_G(v)$ | $D = O(VE)$
5 $G^2 = G^2 + uw$ | $E = O(1) \cdot O(VE) = O(VE)$
6 Devolva G^2 | $F = O(1)$

$$D = \sum_{u \in V} \sum_{v \in N(u)} d(v) \leq \sum_{u \in V} \sum_{v \in V} d(v) = \sum_{u \in V} 2E$$

$$= 2EV$$

Formalização da análise

- Nossa análise assume que o grafo está representado por uma lista de adjacências.
- Como G esta representado por listas de adjacências, podemos implementar copiaGrafo(G) para executar em $O(V+E)$. Assim, o custo da linha 1 é $O(V+E)$.
- O tempo de execução da linha 2 é $O(V)$

Formalização da análise

- O tempo de execução da linha 3 é

$$\sum_{u \in V} O(d(u)) \leq \sum_{u \in V} C\, d(u) = C \sum_{u \in V} d(u) = 2C\,E$$
$$= O(E)$$

- O laço da linha 4 executa $\sum_{u \in V} \sum_{v \in N(u)} O(d(v))$ vezes.

Assim o número de execuções da linha 4 é

$$\sum_{u \in V} \sum_{v \in N(u)} O(d(v)) \leq C \sum_{u \in V} \sum_{v \in N(u)} d(v) \leq C \sum_{u \in V} \sum_{v \in V} d(v) = CVE$$
$$= O(VE)$$

Formalização da análise

- O custo para inserir uma aresta em um grafo G^2 (linha 5) é $O(1)$. Portanto o tempo gasto com o trecho das linhas 4-5 é $O(VE)$.

- Linha 6 é $O(1)$

- Assim, o custo total do código é $O(VE)$ $\square$