Grafos: Introdução

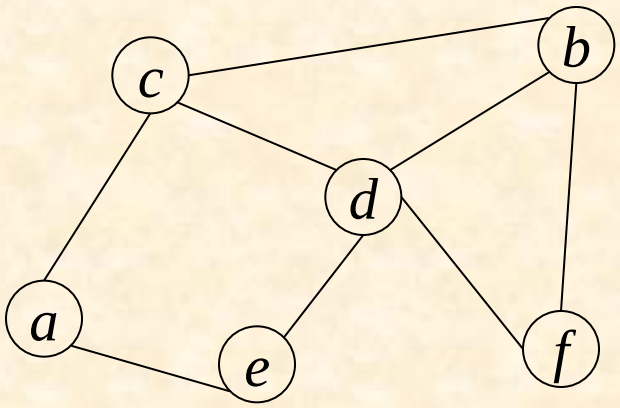
Grafos

Um ***grafo não orientado*** G é um par (V, E), onde V é um conjunto de ***vértices*** e E é um conjunto de ***arestas***; cada aresta é um par não ordenado de vértices.

Seja $(v, w) \in E$; v e w são denominados ***extremidades*** da aresta; v e w são ***adjacentes*** ou ***vizinhos***. Uma aresta unitária, da forma (v, v), é denominada ***laço***.

$|V| = n$ e $|E| = m$. Um grafo é ***trivial*** se $|V| = 1$.

Um ***multigrafo*** é um grafo que admite arestas paralelas, isto é, com as mesmas extremidades.

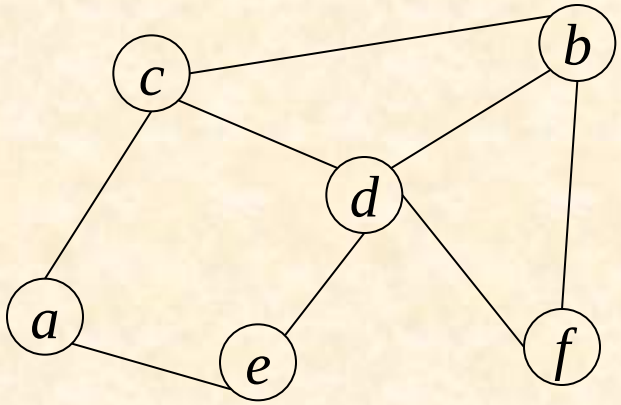


$G = (V, E)$

$V = \{a, b, c, d, e, f\}$

$E = \{(a,c), (a,e), (b,c), (b,d), (b,f), (c,d), (d,e), (d,f)\}$

Grafos



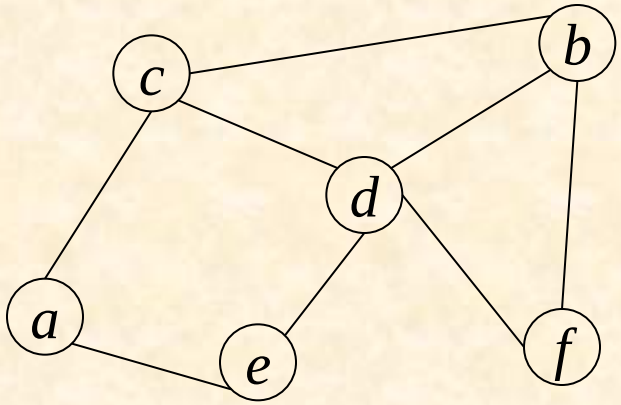
$G = (V, E)$
$V = \{a, b, c, d, e, f\}$
$E = \{(a,c), (a,e), (b,c), (b,d), (b,f), (c,d), (d,e), (d,f)\}$

Conjunto de adjacência de um vértice $v \in V$:
$Adj(v) = \{w \in V \mid (v, w) \in E\}$.

$|Adj(v)|$: ***grau*** de v, notado $d(v)$. Se o grau de um vértice é nulo, o vértice é chamado ***isolado***.

$Adj(a) = \{c,e\}$ $Adj(b) = \{c,d,f\}$ $Adj(c) = (a,b,d\}$
$Adj(d) = \{b,c,e,f\}$ $Adj(e) = \{a,d\}$ $Adj(f) = \{b,d\}$

Grafos



$G = (V, E)$

$V = \{a, b, c, d, e, f\}$

$E = \{(a,c), (a,e), (b,c), (b,d), (b,f), (c,d), (d,e), (d,f)\}$

$Adj(a) = \{c,e\}$ $Adj(b) = \{c,d,f\}$ $Adj(c) = (a,b,d\}$

$Adj(d) = \{b,c,e,f\}$ $Adj(e) = \{a,d\}$ $Adj(f) = \{b,d\}$

Teorema:

Para qualquer grafo $G = (V, E)$ vale a igualdade:

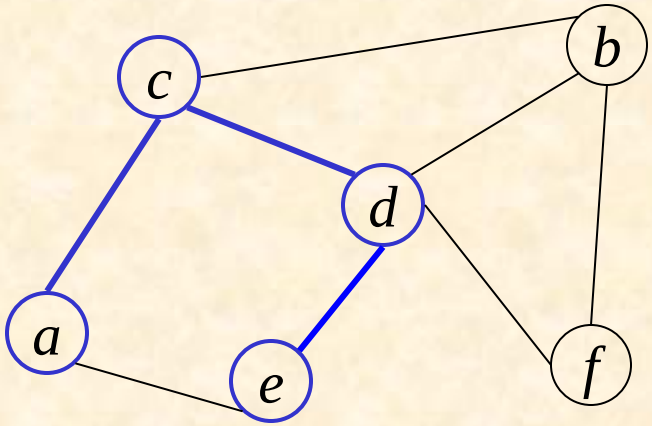
$$\sum_{v \in V} |Adj(v)| = 2m.$$

Caminhos

Um ***caminho*** em $G = (V, E)$ é uma seqüência de $k > 0$ vértices $[v_1, v_2, ..., v_k]$ tal que ou $k = 1$ e o caminho é unitário ou $k > 1$ e $(v_i, v_{i+1}) \in E$, para $i = 1, ..., k - 1$.

O ***comprimento*** do caminho é $k - 1$.

caminho simples: se nenhum vértice se repete.



caminho [*a*, *c*, *d*, *e*]

$k = 4$

comprimento do caminho: 3

Caminhos

Um caminho em $G = (V, E)$ é uma seqüência de $k > 0$ vértices $[v_1, v_2, ..., v_k]$ tal que ou $k = 1$ e o caminho é unitário ou $k > 1$ e $(v_i, v_{i+1}) \in E$, para $i = 1, ..., k - 1$.

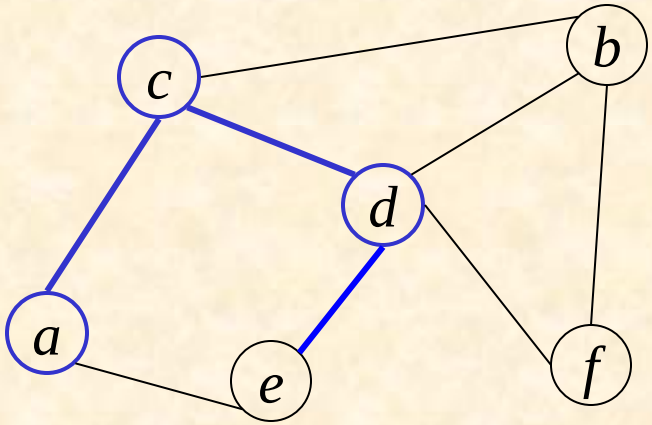
O comprimento do caminho é $k - 1$.

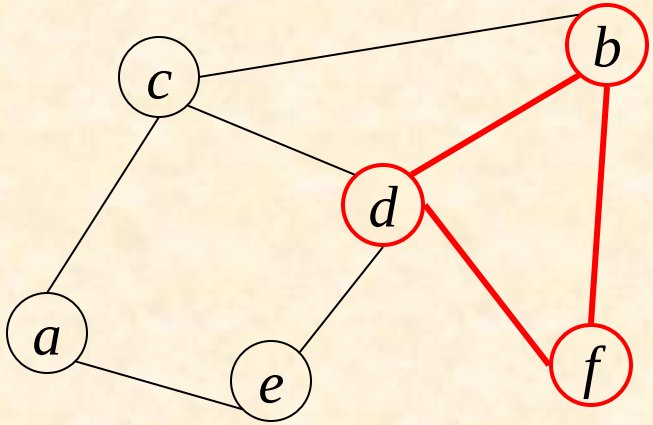
caminho simples: se nenhum vértice se repete.

ciclo: um caminho, de comprimento maior ou igual a 3, onde o primeiro e último vértices coincidem.

ciclo simples: é um caminho, de comprimento maior ou igual a 3, onde apenas o primeiro e o último vértices coincidem, sendo distintos os demais.

Um grafo que não possui ciclos é dito ***acíclico***.





caminho [*a*, *c*, *d*, *e*]
$k = 4$
comprimento do caminho: 3

ciclo [*d*, *b*, *f*, *d*]
comprimento do ciclo: 3

Grafos

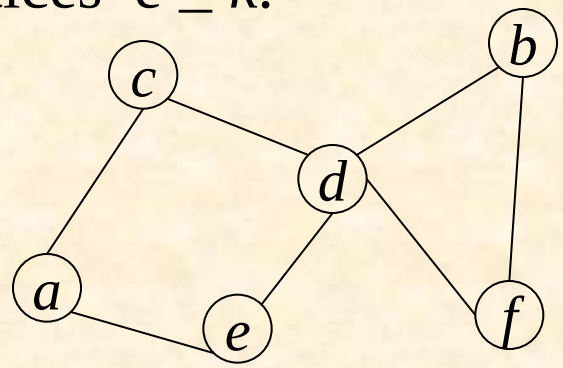
grafo ***conexo***: quando existir pelo menos um caminho entre cada par de vértices; do contrário, o grafo é dito ***desconexo***.

corte de vértices: subconjunto minimal de vértices $V' \subset V$ cuja remoção desconecta o grafo ou o torna trivial.

conectividade de vértices: cardinalidade do menor corte de vértices (corte mínimo).

um grafo G é ***k-conexo em vértices*** quando G é conexo e sua conectividade de vértices é $\geq k$.

articulação



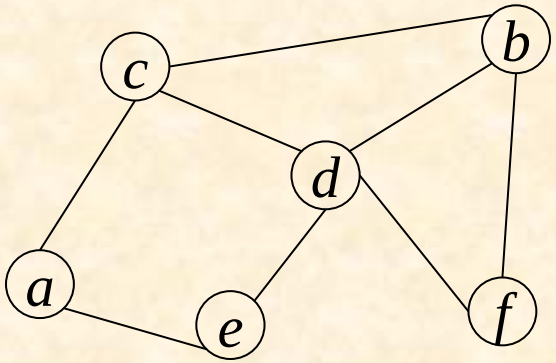
Grafos

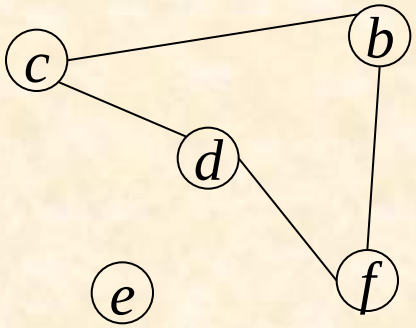
subgrafo de G: grafo $G' = (V', E')$ tal que $V' \subseteq V$ e $E' \subseteq E$.

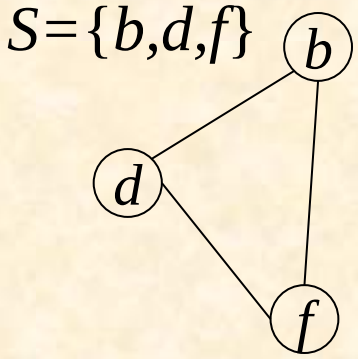
Dado $S \subseteq V$, $G[S] = (S, \{(x,y) \in E \mid x \in S \wedge y \in S\})$ é denominado ***subgrafo de G induzido por S***.

subgrafo gerador de G: qualquer subgrafo de G que possua V como conjunto de vértices.

O conjunto S é uma ***clique*** quando $G[S]$ for um ***grafo completo***, isto é, possuir todas as arestas possíveis. Uma t-clique, $t \geq 1$, é uma clique de cardinalidade t.







Digrafos

Um ***grafo direcionado,*** ou ***digrafo***, G é um par (V, E), onde V é um conjunto de ***vértices*** e E é um conjunto de ***arestas***; cada aresta é um par ordenado de vértices.

Seja $G = (V, E)$ um digrafo. A aresta (v,w) é dita deixando o vértice v e chegando ao vértice w.

grau de entrada de v: número de arestas que chegam à v.

grau de saída de v: número de arestas que deixam v.

grafo subjacente: multigrafo obtido ao se remover a direção das arestas de G.

G é ***fracamente conexo*** ou ***desconexo*** conforme seu grafo subjacente seja conexo ou desconexo.

G é ***fortemente conexo*** quando para todo par de vértices $v, w \in V$ existem caminhos em G de v para w e de w para v.

Famílias de Grafos

O conjunto de todos os grafos que satisfazem uma dada propriedade constitui uma ***família de grafos***.

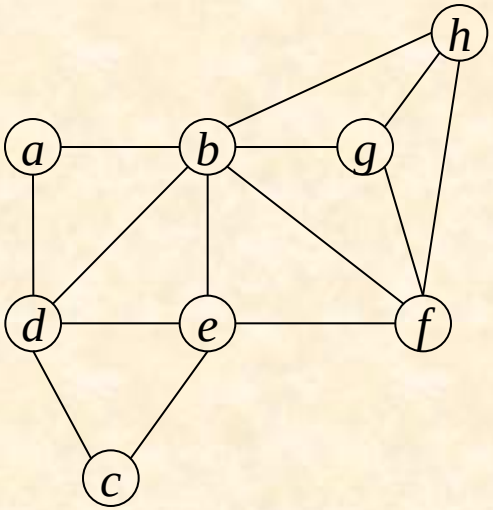
A ***definição*** de uma família consiste em mencionar a propriedade satisfeita pelos grafos que a constituem. A propriedade pode ser acrescida de novas condições, originando propriedades mais específicas, que definem ***subfamílias***.

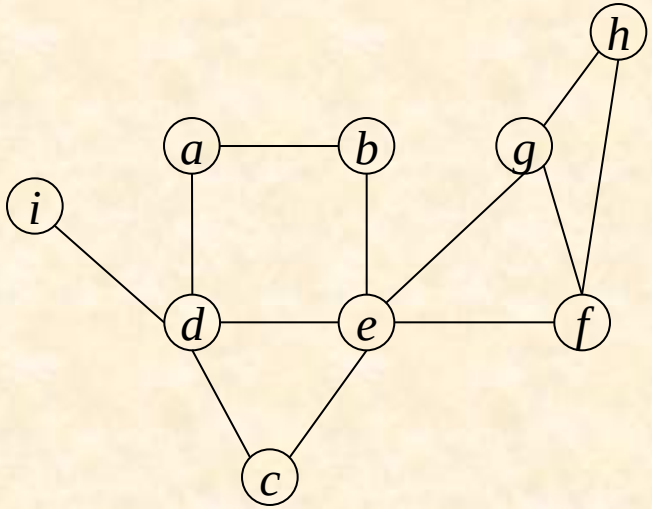
Problema computacional a ser resolvido: ***reconhecimento***.

Exemplos de Famílias

Grafos planares: grafos que podem ser desenhados sobre uma superfície plana sem cruzamento de arestas.

Grafos periplanares: grafos planares tais que todos os seus vértices podem situar-se em uma única face da representação.

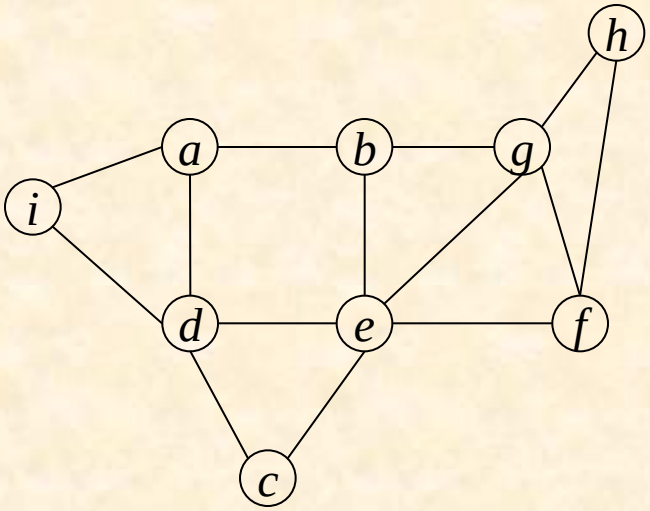


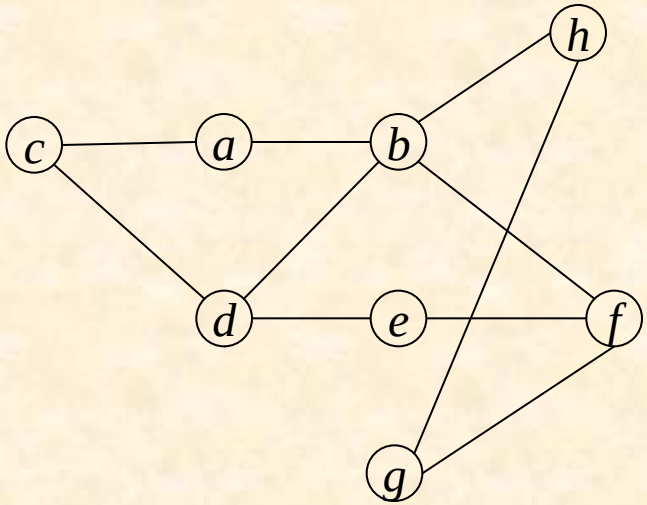


Exemplos de Famílias

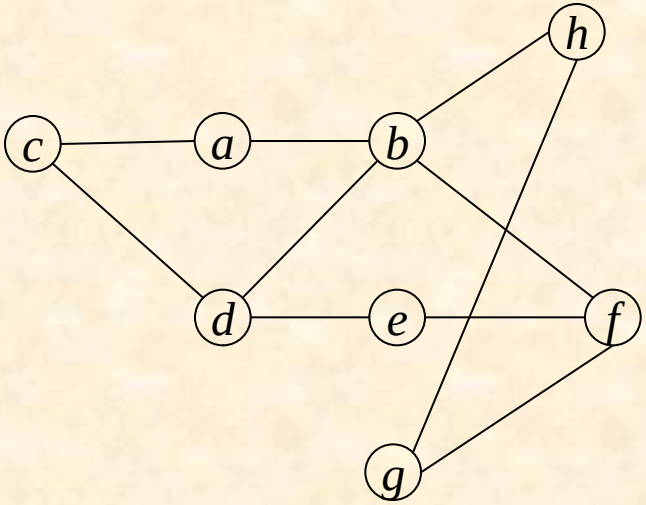
Grafos hamiltonianos: grafos que possuem um ciclo simples contendo todos os seus vértices.

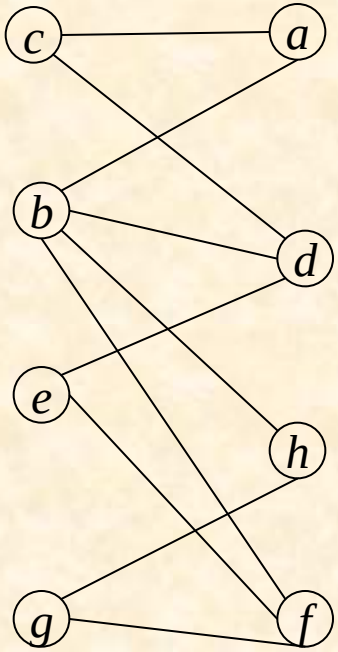
Grafos bipartidos: grafos tais que seu conjunto de vértices pode ser particionado em dois conjuntos, *X* e *Y*, tais que todas as arestas têm uma extremidade em *X* e e outra em *Y*.





Grafos bipartidos





Árvores

Uma ***árvore*** é um grafo conexo e acíclico.

Teorema:
Seja $G = (V, E)$ um grafo. As seguintes afirmativas são equivalentes:
(1) G é uma árvore.
(2) G é conexo e $|E|$ é mínima.
(3) G é conexo e $|E| = |V| - 1$.
(4) G é acíclico e $|E| = |V| - 1$.
(5) G é acíclico e para todo $v, w \in V$ a adição da aresta (v, w) produz um grafo contendo exatamente um ciclo (ciclo fundamental).

Árvores

Seja uma árvore $T = (V, E)$.
árvore enraizada: quando é eleito um vértice $r \in V$ como raiz da árvore.

ancestrais de um vértice v: todos os vértices no (único) caminho simples entre a raiz r e v (incluindo o próprio vértice v).
A raiz é um ancestral comum a todos os vértices.

nível de um vértice: número de ancestrais que ele possui.
A raiz tem nível 1.

um vértice w é ***descendente*** de v se, e somente se, v for ancestral de w.

Uma ***árvore geradora*** de um grafo conexo $G = (V, E)$ é qualquer subgrafo gerador de G que seja acíclico e conexo.

Esquemas de Representação para Grafos

definição: através de um par de conjuntos, o conjunto de vértices e o conjunto de arestas.

esquema de representação: regras que conduzem a uma maneira alternativa de individualizar grafos, na qual se evita a menção explícita dos conjuntos de vértices e arestas.

representação: resultado obtido pela aplicação de um esquema a um grafo.

Esquema de Representação Geométrico

Uma ***representação geométrica*** de um grafo $G = (V, E)$ é obtida pela aplicação das seguintes regras:

- escolha uma superfície sobre a qual o grafo será representado;

- escolha n pontos distintos sobre esta superfície, nomeando-os de acordo com os vértices especificados em V;

- para cada aresta mencionada em E, una os pontos correspondentes às suas extremidades através de um segmento arbitrário de curva sobre a superfície escolhida.

A representação de um grafo através do esquema geométrico é, portanto, um desenho, a partir do qual vértices e arestas podem ser visualizados.

Esquema de Representação por Conjuntos de Adjacência

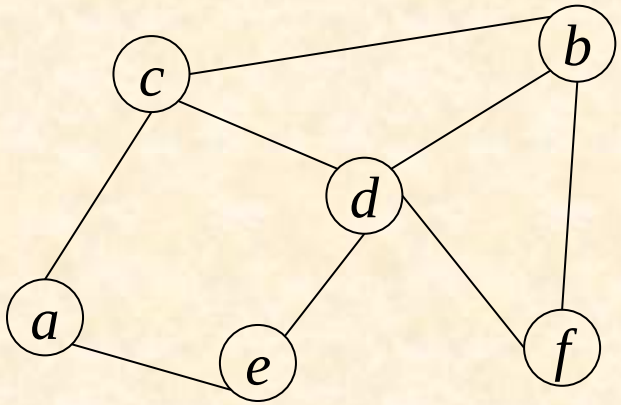
Consiste em mencionar, para todo vértice $v \in V$, o conjunto $Adj(v)$, construindo um conjunto de pares da forma

$$\{(v, Adj(v)) \mid v \in V\}.$$

São mencionados:

$$\sum_{v \in V} [1 + |Adj(v)|] = n + 2m \text{ símbolos.}$$

Exemplo



Definição: $G = (V, E)$

$V = \{a, b, c, d, e, f\}$

$E = \{(a,c), (a,e), (b,c), (b,d), (b,f), (c,d), (d,e), (d,f)\}$

Conjuntos de Adjacência: $\{(a,\{c,e\}), (b,\{c,d,f\}), (c,\{a,b,d\}), (d,\{b,c,e,f\}), (e,\{a,d\}), (f, \{b,d\})\}$

$Adj(a) = \{c,e\}$

$Adj(b) = \{c,d,f\}$

$Adj(c) = \{a,b,d\}$

$Adj(d) = \{b,c,e,f\}$

$Adj(e) = \{a,d\}$

$Adj(f) = \{b,d\}$

Esquema de Representação por Matriz de Adjacência

Uma ***matriz de adjacência*** é uma matriz simétrica 0-1 $A_{n\times n}$ obtida da seguinte maneira:

- defina uma bijeção ω: $\{1,...,n\} \rightarrow V$, que corresponde a numerar seqüencialmente os vértices a partir de 1;

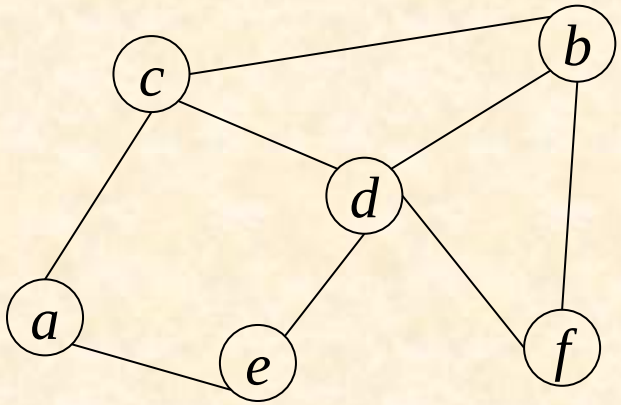
- para $1 \leq i < j \leq n$, defina

$$A(i,j) = A(j,i) = 1, \text{ se } (\omega(i), \omega(j)) \in E$$

$$A(i,j) = A(j,i) = 0, \text{ se } (\omega(i), \omega(j)) \notin E$$

Utilizados $n^2 + n$ símbolos.

Exemplo



Definição: $G = (V, E)$

$V = \{a, b, c, d, e, f\}$

$E = \{(a,c), (a,e), (b,c), (b,d), (b,f), (c,d), (d,e), (d,f)\}$

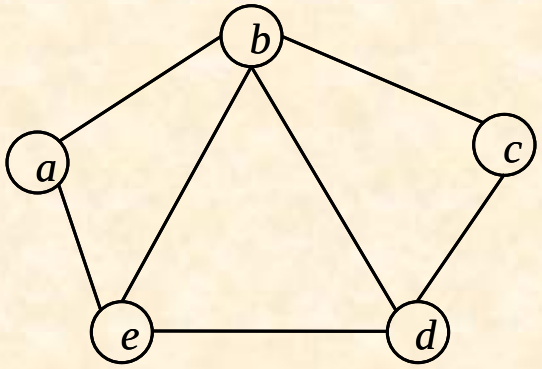
Conjuntos de Adjacência: $\{(a,\{c,e\}), (b,\{c,d,f\}), (c,\{a,b,d\}), (d,\{b,c,e,f\}), (e,\{a,d\}), (f, \{b,d\})\}$

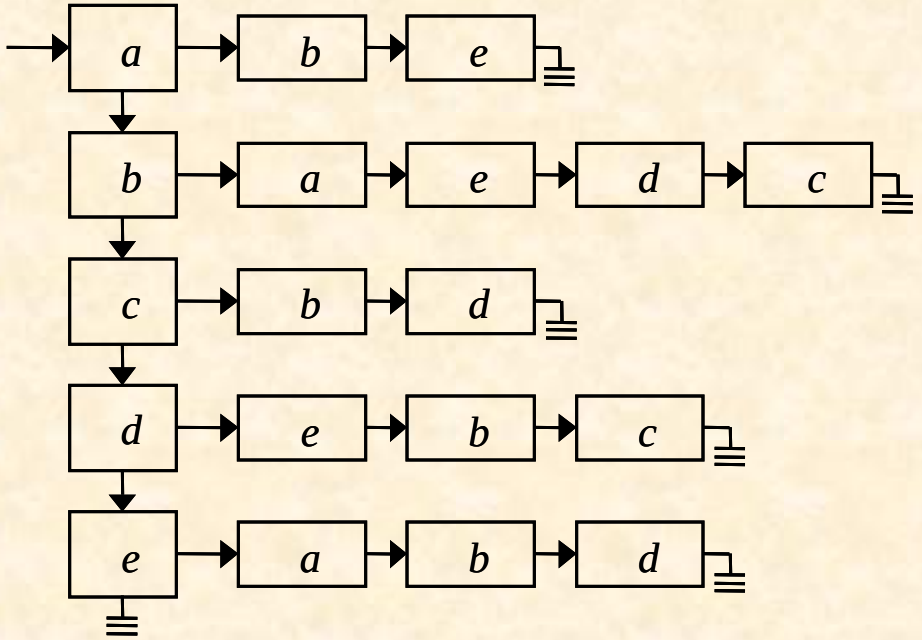
Matriz de Adjacência:

	a	*b*	*c*	*d*	*e*	*f*
a	0	0	1	0	1	0
b	0	0	1	1	0	1
c	1	1	0	1	0	0
d	0	1	1	0	1	1
e	1	0	0	1	0	0
f	0	1	0	1	0	0

Armazenamento em Memória Principal

Por ***listas de adjacência***: decorre da representação por conjuntos de adjacência.





Leitura de um Grafo para a Memória

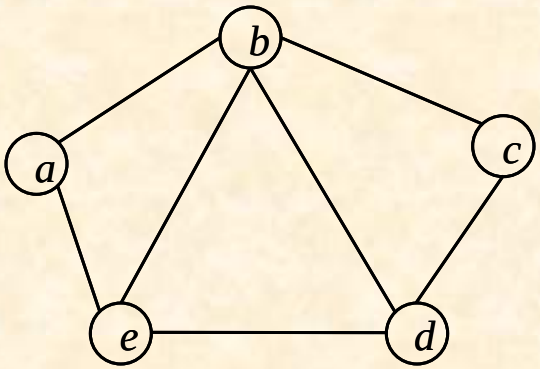
Leitura de um grafo + inicializações das estruturas que o armazenam

```
para i ← 1, …, n faça  lista-de-pont[i] ← λ;
para i ← 1, …, m faça
        ler aresta (v,w);
        alocar-nó(pt);
        pt ↑.adj ← w;   pt ↑.prox ← lista-de-pont[v];
       lista-de-pont[v] ← pt;
       alocar-nó(pt);
       pt ↑.adj ← v;   pt ↑.prox ← lista-de-pont[w];
       lista-de-pont[w] ← pt;
```

Complexidade de tempo: $O(n+m)$

Armazenamento em Memória Principal

Por ***matriz de adjacência***: decorre da representação por matriz de adjacência.



	a	*b*	*c*	*d*	*e*
a	0	1	0	0	1
b	**1**	0	1	1	1
c	**0**	**1**	0	1	0
d	**0**	**1**	**1**	0	1
e	**1**	**1**	**0**	**1**	0

Armazenamento compacto:

A_{21}	A_{31}	A_{32}	A_{41}	A_{42}	A_{43}	A_{51}	A_{52}	A_{53}	A_{54}

Leitura de um Grafo para a Memória

Leitura de um grafo + inicializações das estruturas que o armazenam

```
para i ← 1, …, n faça
        para j ← 1, …, n faça
                mat[i,j] ← 0;
para i ← 1, …, m faça
        ler-aresta(v,w);
        mat[v,w] ← 1;
        mat[w,v] ← 1;
```

Complexidade de tempo: $n^2+m = O(n^2)$

Solução: Matriz de adjacência + Lista de certificados

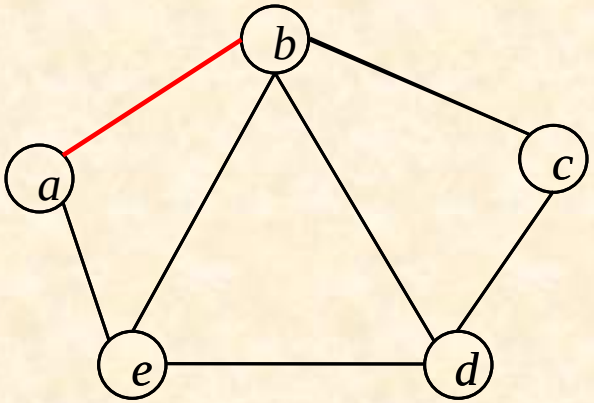
Elementos da Lista de Certificados: subconjuntos de dois vértices

Leitura de um grafo + inicializações das estruturas que o armazenam

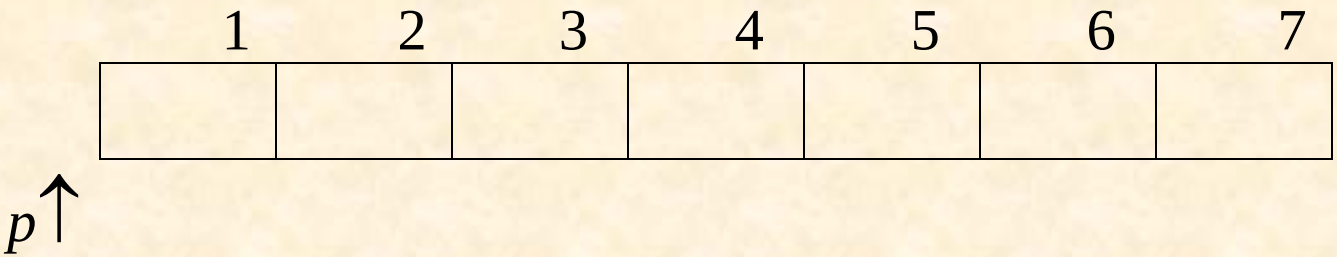
Algoritmo de inicialização:

$p \leftarrow 0$;
para $i \leftarrow 1, \ldots, m$ faça
    ***ler-aresta*(*v*,*w*);**
    $p \leftarrow p + 1$;
    $M(v,w) \leftarrow M(w,v) \leftarrow p$;
    ***Lista*(*p*) $\leftarrow$ (*v*, *w*);**

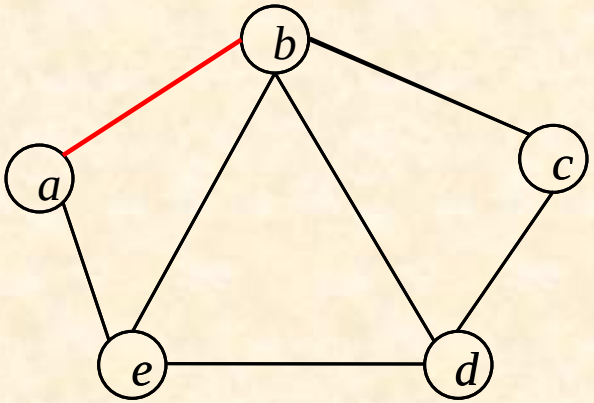
Exemplo



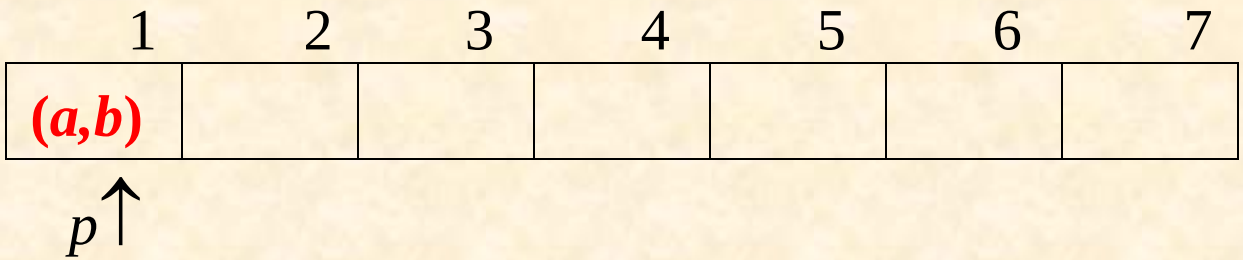
	a	*b*	*c*	*d*	*e*
a	&	?	?	#	$
b	@	×	?	@	&
c	%	#	♥	$	♥
d	&	?	@	%	$
e	%	?	×	#	♥



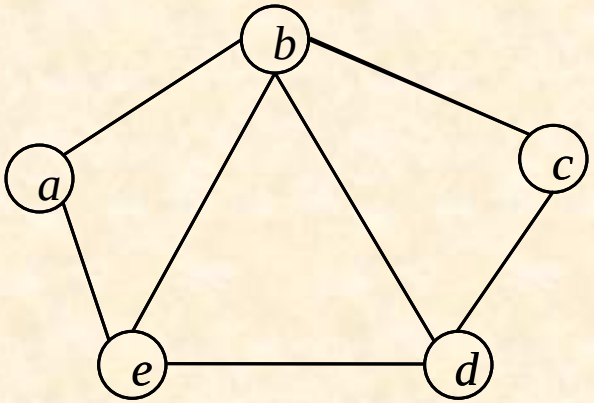
Exemplo



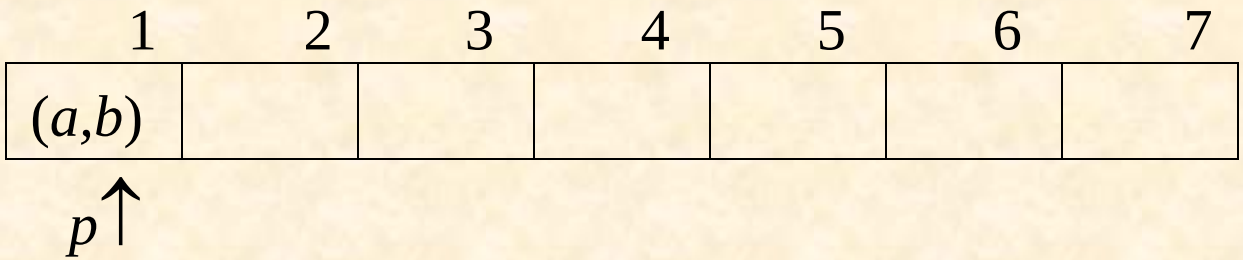
	a	*b*	*c*	*d*	*e*
a	&	**1**	?	#	$
b	**1**	×	?	@	&
c	%	#	♥	$	♥
d	&	**?**	@	%	$
e	%	?	×	#	♥



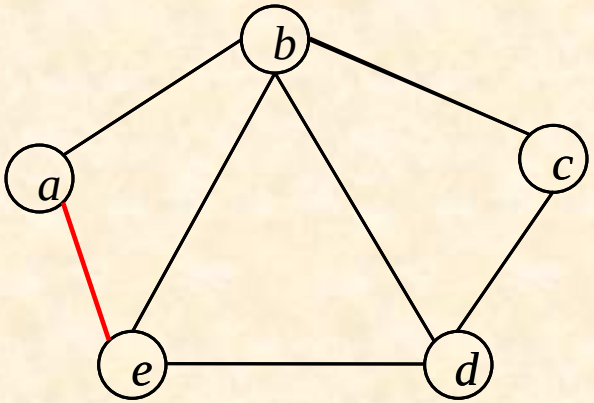
Exemplo



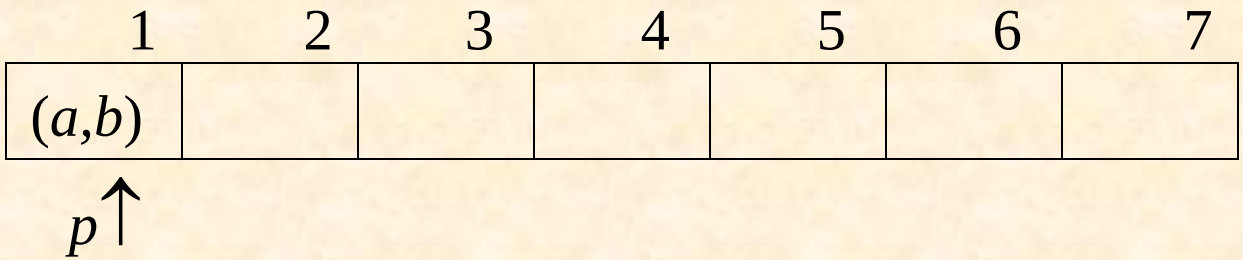
	a	*b*	*c*	*d*	*e*
a	&	**1**	?	#	$
b	**1**	×	?	@	&
c	%	#	♥	$	♥
d	&	**?**	@	%	$
e	%	?	×	#	♥



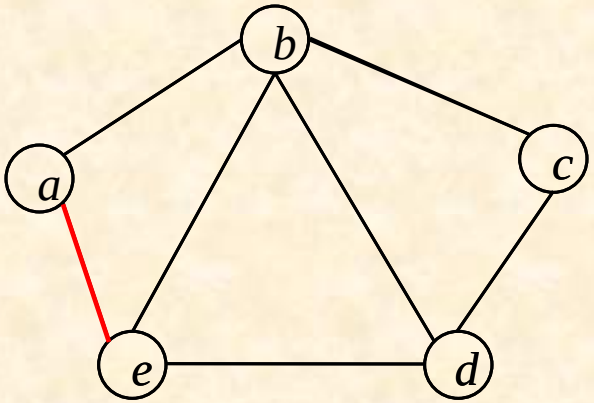
Exemplo



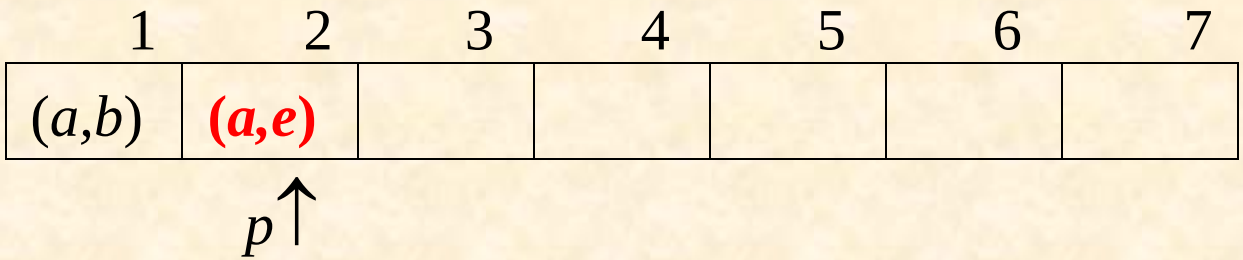
	a	*b*	*c*	*d*	*e*
a	&	**1**	?	#	$
b	**1**	×	?	@	&
c	%	#	♥	$	♥
d	&	**?**	@	%	$
e	%	?	×	#	♥



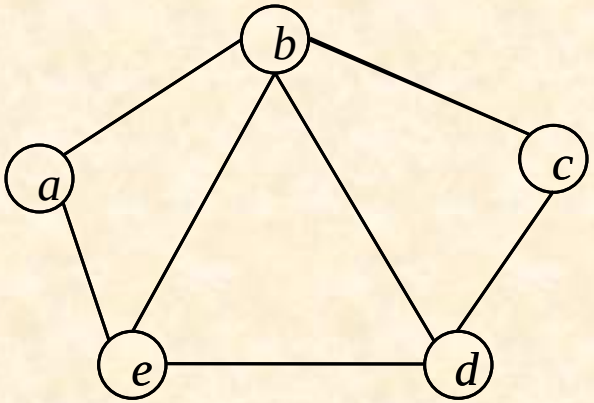
Exemplo



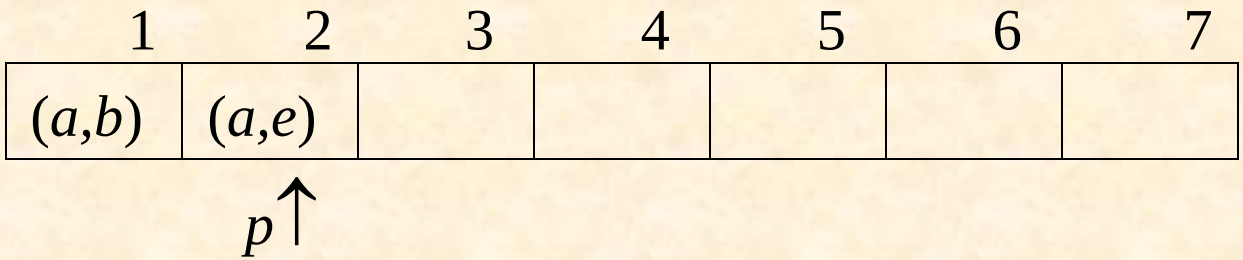
	a	*b*	*c*	*d*	*e*
a	&	1	?	#	**2**
b	1	×	?	@	&
c	%	#	♥	$	♥
d	&	**?**	@	%	$
e	**2**	?	×	#	♥



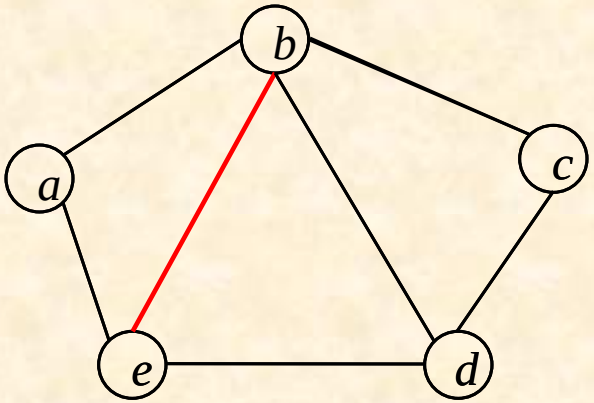
Exemplo



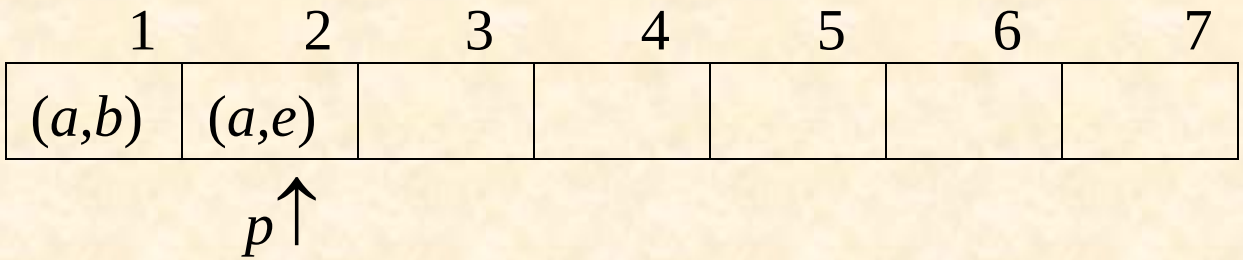
	a	*b*	*c*	*d*	*e*
a	&	**1**	?	#	**2**
b	**1**	×	?	@	&
c	%	#	♥	$	♥
d	&	**?**	@	%	$
e	**2**	?	×	#	♥



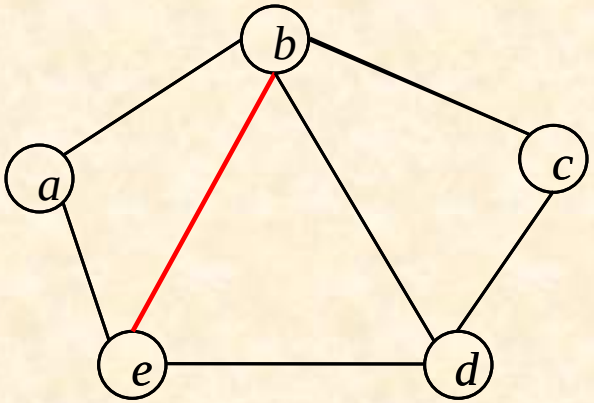
Exemplo



	a	*b*	*c*	*d*	*e*
a	&	**1**	?	#	**2**
b	**1**	×	?	@	&
c	%	#	♥	$	♥
d	&	**?**	@	%	$
e	**2**	?	×	#	♥



Exemplo

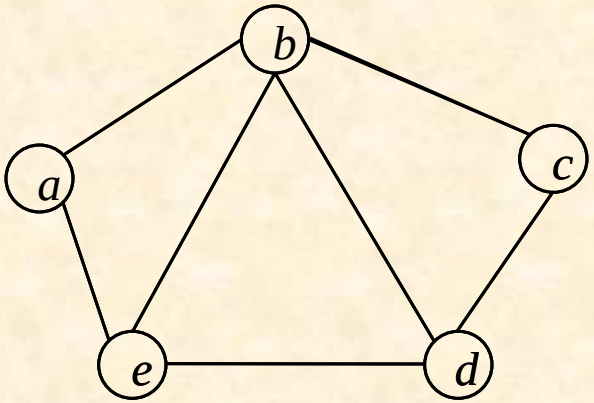


	a	*b*	*c*	*d*	*e*
a	&	**1**	?	#	**2**
b	**1**	×	?	@	**3**
c	%	#	♥	$	♥
d	&	**?**	@	%	$
e	**2**	**3**	×	#	♥

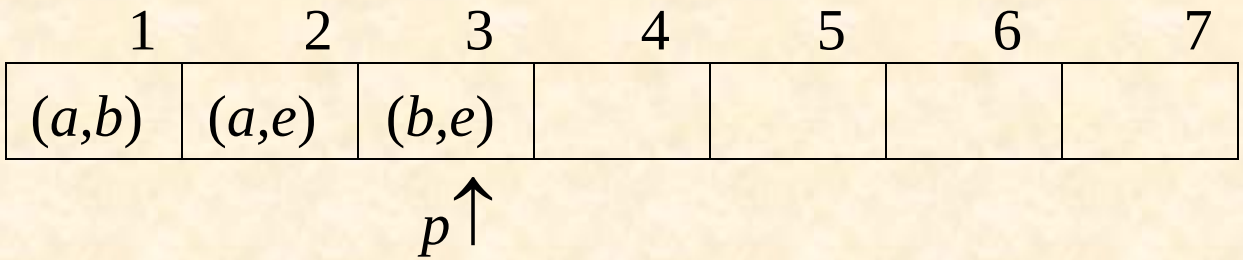
1	2	3	4	5	6	7
(*a*,*b*)	(*a*,*e*)	(***b***,***e***)				

p↑

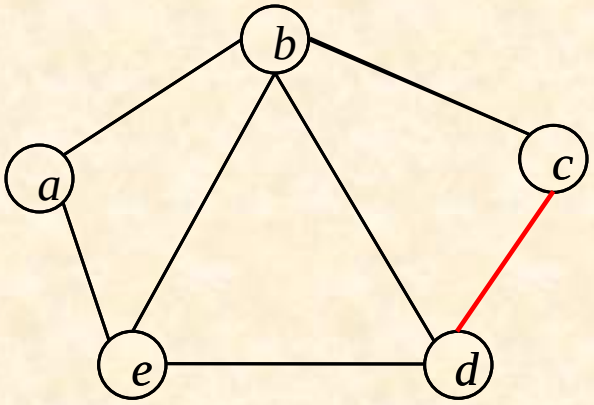
Exemplo



	a	*b*	*c*	*d*	*e*
a	&	**1**	?	#	**2**
b	**1**	×	?	@	**3**
c	%	#	♥	$	♥
d	&	**?**	@	%	$
e	**2**	**3**	×	#	♥



Exemplo

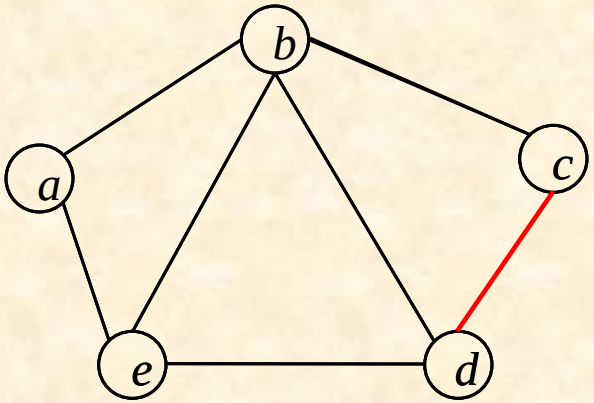


	a	*b*	*c*	*d*	*e*
a	&	**1**	?	#	**2**
b	**1**	×	?	@	**3**
c	%	#	♥	$	♥
d	&	**?**	@	%	$
e	**2**	**3**	×	#	♥

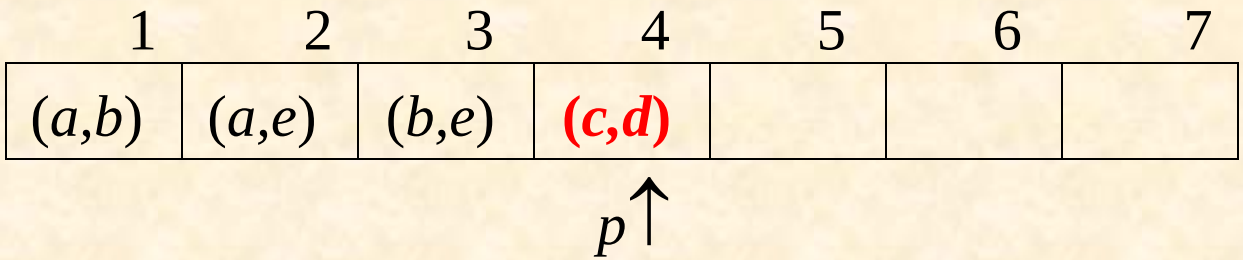
1	2	3	4	5	6	7
(*a*,*b*)	(*a*,*e*)	(*b*,*e*)				

p↑

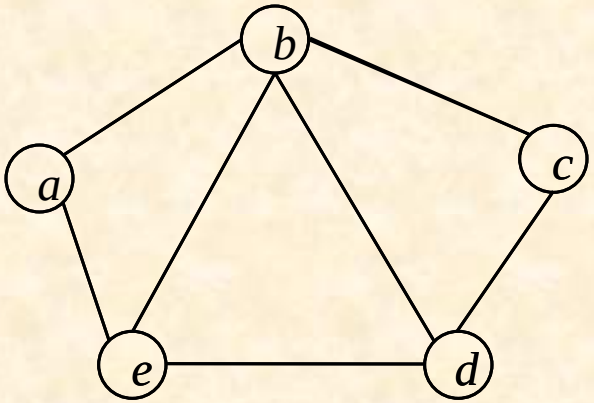
Exemplo



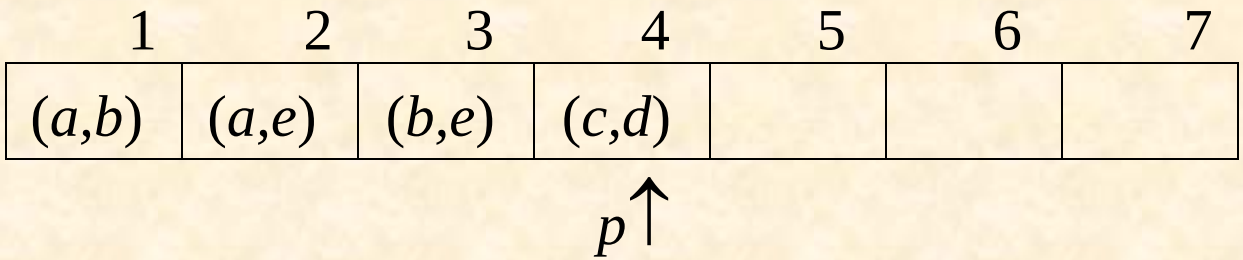
	a	*b*	*c*	*d*	*e*
a	&	**1**	?	#	**2**
b	**1**	×	?	@	**3**
c	%	#	♥	**4**	♥
d	&	**?**	**4**	%	$
e	**2**	**3**	×	#	♥



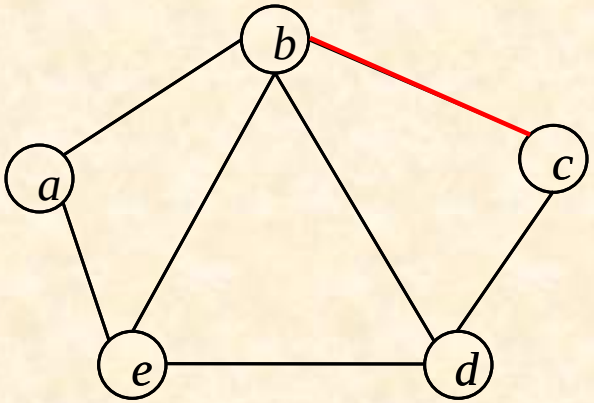
Exemplo



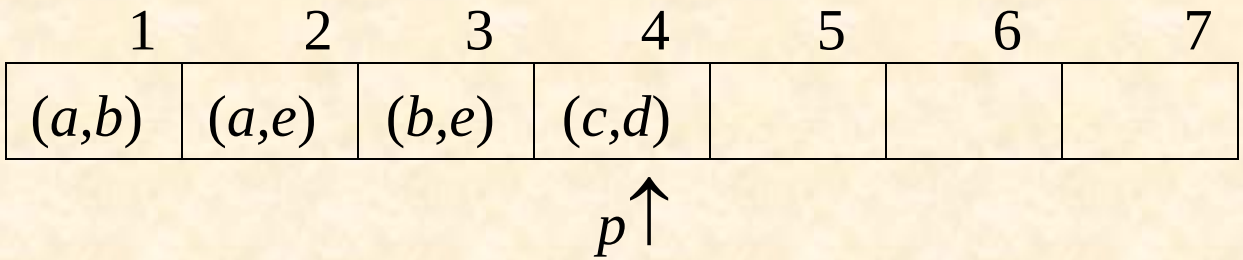
	a	*b*	*c*	*d*	*e*
a	&	**1**	?	#	**2**
b	**1**	×	?	@	**3**
c	%	#	♥	**4**	♥
d	&	**?**	**4**	%	$
e	**2**	**3**	×	#	♥



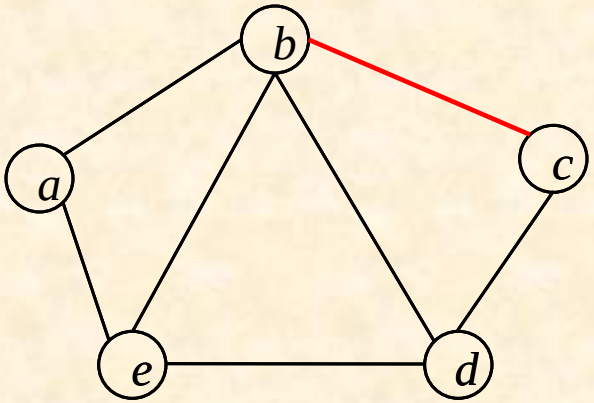
Exemplo



	a	*b*	*c*	*d*	*e*
a	&	**1**	?	#	**2**
b	**1**	×	?	@	**3**
c	%	#	♥	**4**	♥
d	&	**?**	**4**	%	$
e	**2**	**3**	×	#	♥



Exemplo

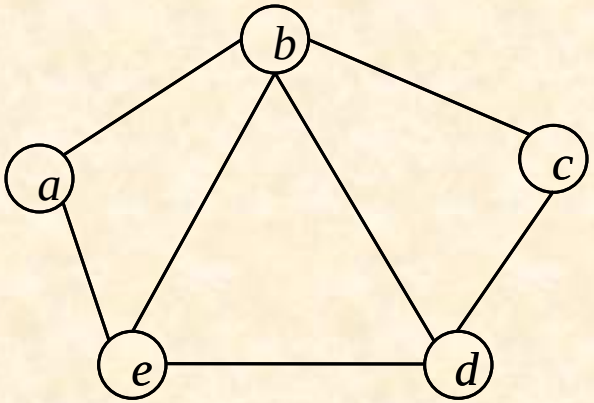


	a	*b*	*c*	*d*	*e*
a	&	**1**	?	#	**2**
b	**1**	×	**5**	@	**3**
c	%	**5**	♥	**4**	♥
d	&	**?**	**4**	%	$
e	**2**	**3**	×	#	♥

1	2	3	4	5	6	7
(*a*,*b*)	(*a*,*e*)	(*b*,*e*)	(*c*,*d*)	(***b***,***c***)		

p↑

Exemplo

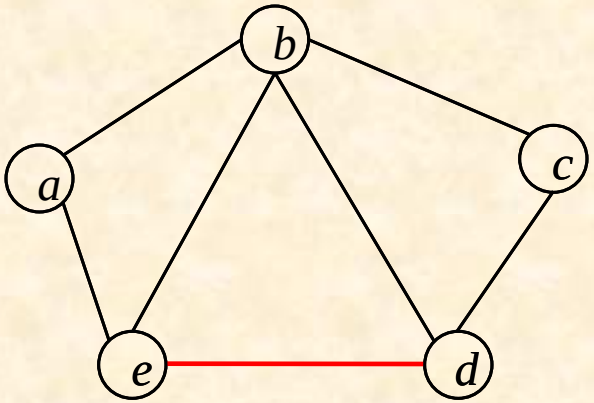


	a	*b*	*c*	*d*	*e*
a	&	**1**	?	#	**2**
b	**1**	×	**5**	@	**3**
c	%	**5**	♥	**4**	♥
d	&	**?**	**4**	%	$
e	**2**	**3**	×	#	♥

1	2	3	4	5	6	7
(*a*,*b*)	(*a*,*e*)	(*b*,*e*)	(*c*,*d*)	(*b*,*c*)		

p↑

Exemplo

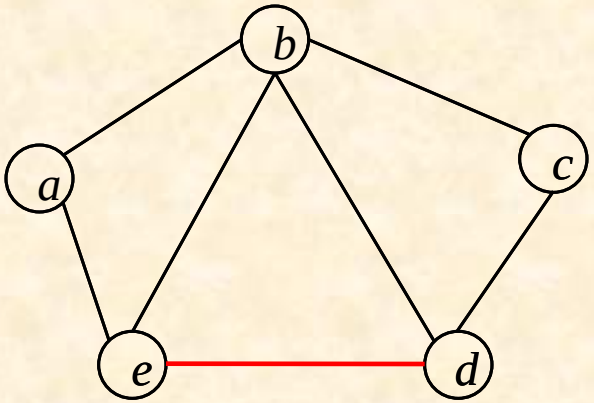


	a	*b*	*c*	*d*	*e*
a	&	**1**	?	#	**2**
b	**1**	×	**5**	@	**3**
c	%	**5**	♥	**4**	♥
d	&	**?**	**4**	%	$
e	**2**	**3**	×	#	♥

1	2	3	4	5	6	7
(*a*,*b*)	(*a*,*e*)	(*b*,*e*)	(*c*,*d*)	(*b*,*c*)		

p↑

Exemplo

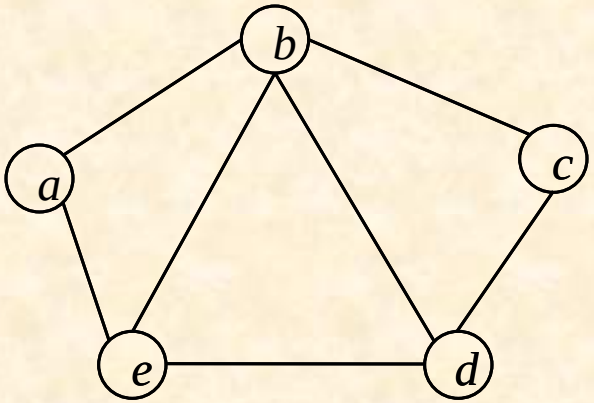


	a	*b*	*c*	*d*	*e*
a	&	**1**	?	#	**2**
b	**1**	×	**5**	@	**3**
c	%	**5**	♥	**4**	♥
d	&	**?**	**4**	%	**6**
e	**2**	**3**	×	**6**	♥

1	2	3	4	5	6	7
(*a*,*b*)	(*a*,*e*)	(*b*,*e*)	(*c*,*d*)	(*b*,*c*)	**(*d*,*e*)**	

p↑

Exemplo

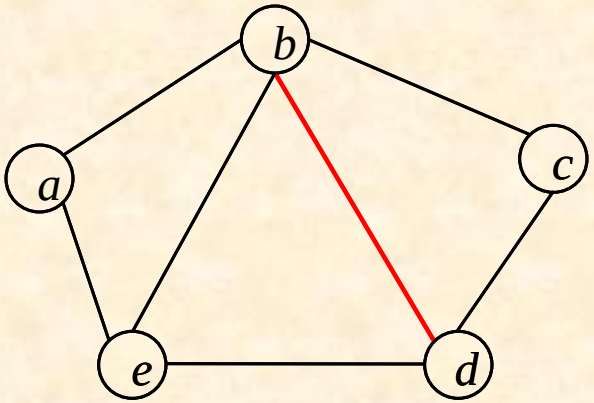


	a	*b*	*c*	*d*	*e*
a	&	**1**	?	#	**2**
b	**1**	×	**5**	@	**3**
c	%	**5**	♥	**4**	♥
d	&	**?**	**4**	%	**6**
e	**2**	**3**	×	**6**	♥

1	2	3	4	5	6	7
(*a*,*b*)	(*a*,*e*)	(*b*,*e*)	(*c*,*d*)	(*b*,*c*)	(*d*,*e*)	

p↑

Exemplo

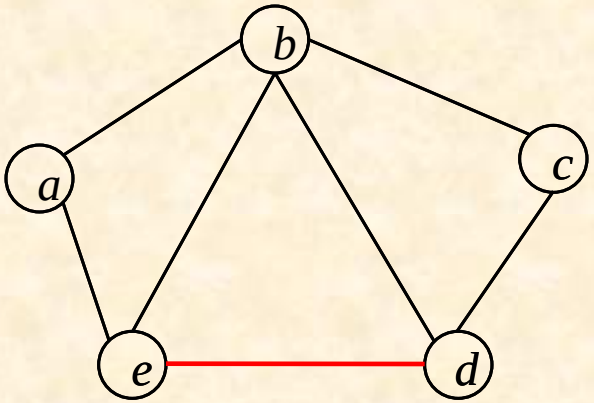


	a	*b*	*c*	*d*	*e*
a	&	**1**	?	#	**2**
b	**1**	×	**5**	@	**3**
c	%	**5**	♥	**4**	♥
d	&	**?**	**4**	%	**6**
e	**2**	**3**	×	**6**	♥

1	2	3	4	5	6	7
(*a*,*b*)	(*a*,*e*)	(*b*,*e*)	(*c*,*d*)	(*b*,*c*)	(*d*,*e*)	

p↑

Exemplo

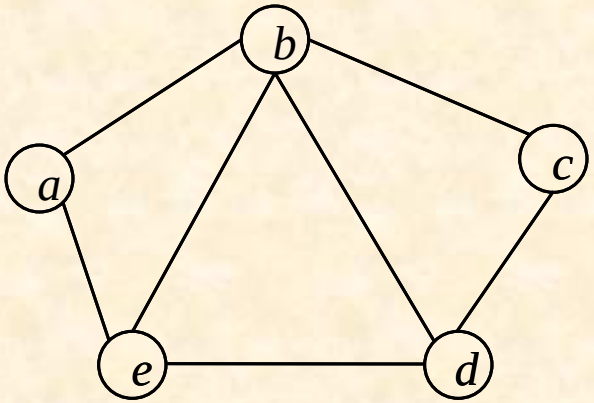


	a	*b*	*c*	*d*	*e*
a	&	**1**	?	#	**2**
b	**1**	×	**5**	@	**3**
c	%	**5**	♥	**4**	♥
d	&	**?**	**4**	%	**6**
e	**2**	**3**	×	**6**	♥

1	2	3	4	5	6	7
(*a*,*b*)	(*a*,*e*)	(*b*,*e*)	(*c*,*d*)	(*b*,*c*)	**(*d*,*e*)**	

p↑

Exemplo

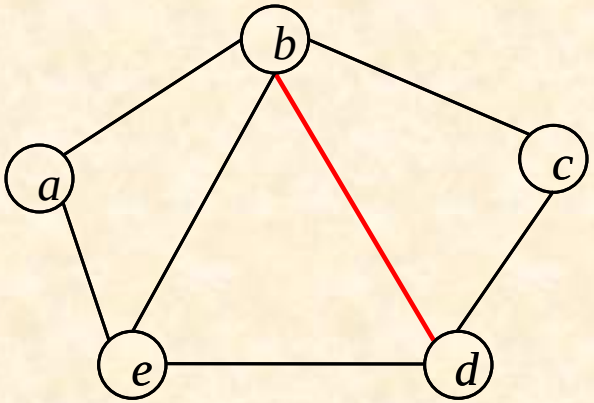


	a	*b*	*c*	*d*	*e*
a	&	**1**	?	#	**2**
b	**1**	×	**5**	@	**3**
c	%	**5**	♥	**4**	♥
d	&	**?**	**4**	%	**6**
e	**2**	**3**	×	**6**	♥

1	2	3	4	5	6	7
(*a*,*b*)	(*a*,*e*)	(*b*,*e*)	(*c*,*d*)	(*b*,*c*)	(*d*,*e*)	

p↑

Exemplo

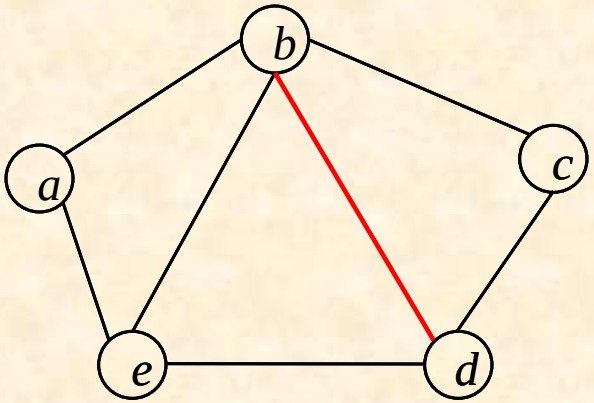


	a	*b*	*c*	*d*	*e*
a	&	**1**	?	#	**2**
b	**1**	×	**5**	@	**3**
c	%	**5**	♥	**4**	♥
d	&	**?**	**4**	%	**6**
e	**2**	**3**	×	**6**	♥

1	2	3	4	5	6	7
(*a*,*b*)	(*a*,*e*)	(*b*,*e*)	(*c*,*d*)	(*b*,*c*)	(*d*,*e*)	

p↑

Exemplo

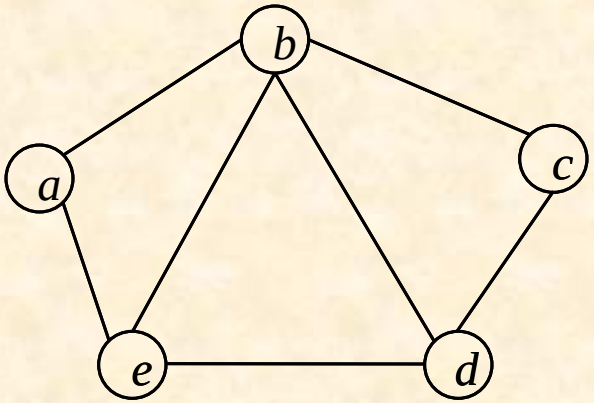


	a	*b*	*c*	*d*	*e*
a	&	**1**	?	#	**2**
b	**1**	×	**5**	**7**	**3**
c	%	**5**	♥	**4**	♥
d	&	**7**	**4**	%	**6**
e	**2**	**3**	×	**6**	♥

1	2	3	4	5	6	7
(*a*,*b*)	(*a*,*e*)	(*b*,*e*)	(*c*,*d*)	(*b*,*c*)	(*d*,*e*)	**(*b*,*d*)**

p ↑

Exemplo



	a	*b*	*c*	*d*	*e*
a	&	**1**	?	#	**2**
b	**1**	×	**5**	**7**	**3**
c	%	**5**	♥	**4**	♥
d	&	**7**	**4**	%	**6**
e	**2**	**3**	×	**6**	♥

1	2	3	4	5	6	7
(*a*,*b*)	(*a*,*e*)	(*b*,*e*)	(*c*,*d*)	(*b*,*c*)	(*d*,*e*)	(*b*,*d*)

p↑

Exemplo

Dado um digrafo $G = (V, E)$ calcular o grau de entrada e o grau de saída de seus vértices.

```
para v ∈ V faça
        grau-de-entrada[v] ← 0; grau-de-saida[v] ← 0;
para v ∈ V faça
        para w ∈ Adj(v) faça
                grau-de-entrada[w] ← grau-de-entrada[w] + 1;
                grau-de-saida[v] ← grau-de-saida[v] + 1;
```

Complexidade de tempo: ???