CES-11

Algoritmos e Estruturas de Dados

Carlos Alberto Alonso Sanches
Juliana de Melo Bezerra

CES-11

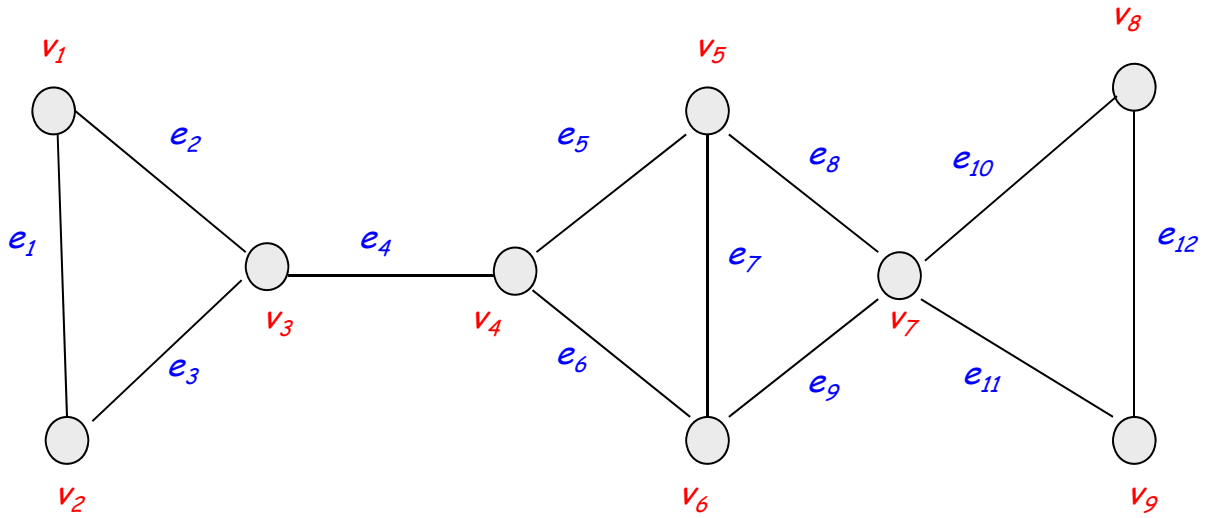
- Grafos
  - Conceitos gerais e representações
- Algoritmos em grafos
  - Exploração sistemática em largura
  - Caminhos mais curtos
  - Exploração sistemática em profundidade
  - Teste de aciclicidade
  - Ordenação topológica
  - Componentes fortemente conexos
  - Vértices e arestas de corte
  - Árvore geradora de custo mínimo

CES-11

- Grafos
  - Conceitos gerais e representações
- Algoritmos em grafos
  - Exploração sistemática em largura
  - Caminhos mais curtos
  - Exploração sistemática em profundidade
  - Teste de aciclicidade
  - Ordenação topológica
  - Componentes fortemente conexos
  - Vértices e arestas de corte
  - Árvore geradora de custo mínimo

Definição

- Um grafo G=(V,E) é formado pelos vértices $V = \{v_1, v_2, ..., v_n\}$ e pelas arestas $E = \{e_1, e_2, ..., e_m\}$.
- Consideraremos sempre que |V| = n e |E| = m.
- Um exemplo:



$V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9\}$ n = 9

$E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}, e_{12}\}$ m = 12

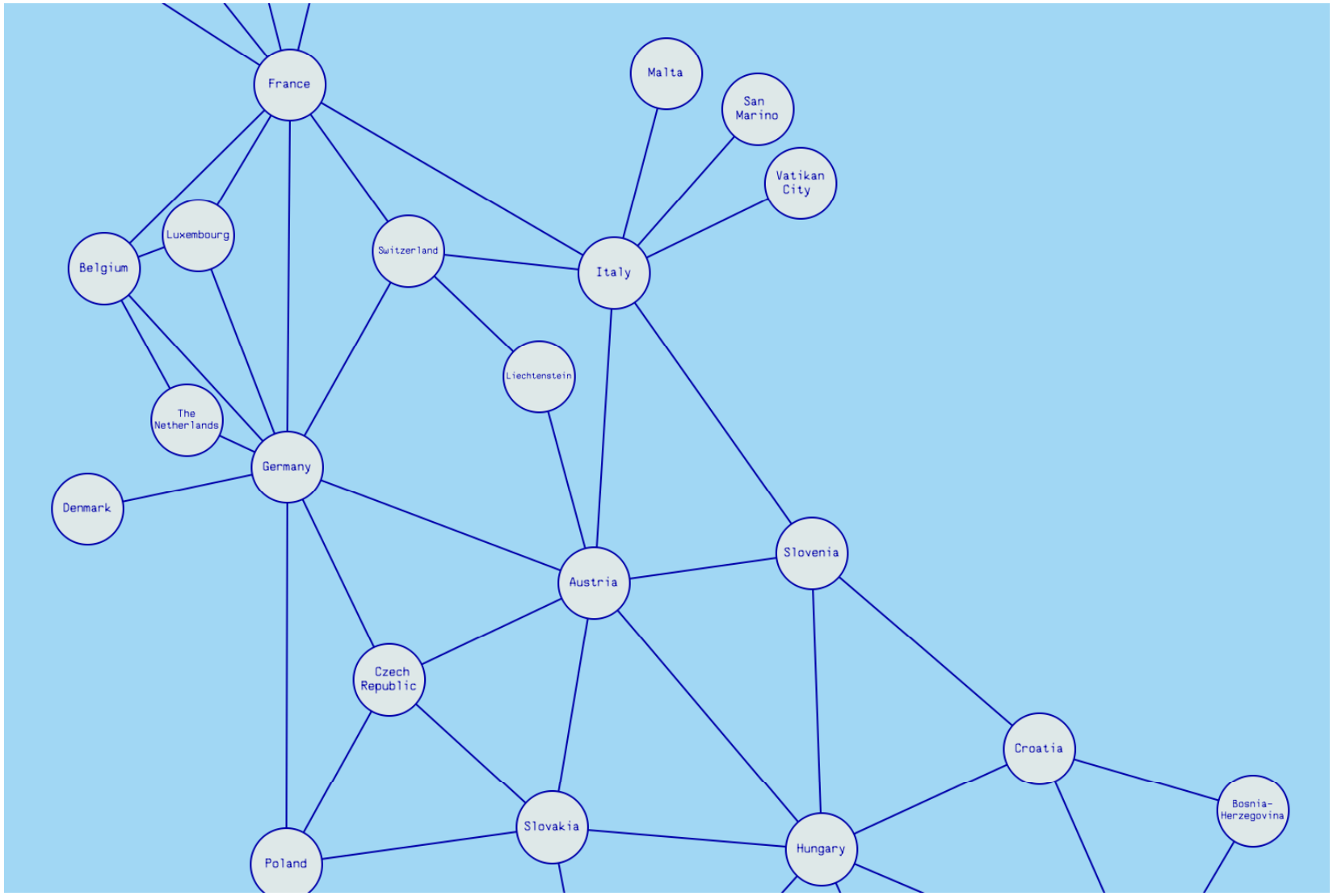
Informações em um grafo

- Pode haver informações nos vértices e/ou nas arestas.
- Ex1: Fornecimento de produtos entre fábricas
  - Vértice (fábrica): nome, localização, número de empregados
  - Aresta (fornecimento): produto transportado, quantidade, custo
- Ex2: Mapa rodoviário
  - Vértice (cidade): nome, número de habitantes, área
  - Aresta (rodovia): distância, qualidade do trecho, eventuais pedágios
- Ex3: Rede social
  - Vértice (pessoa): nome, ocupação, idade, endereço
  - Aresta (colaboração): trabalhos em conjunto, área de atuação, artigos

Algumas aplicações

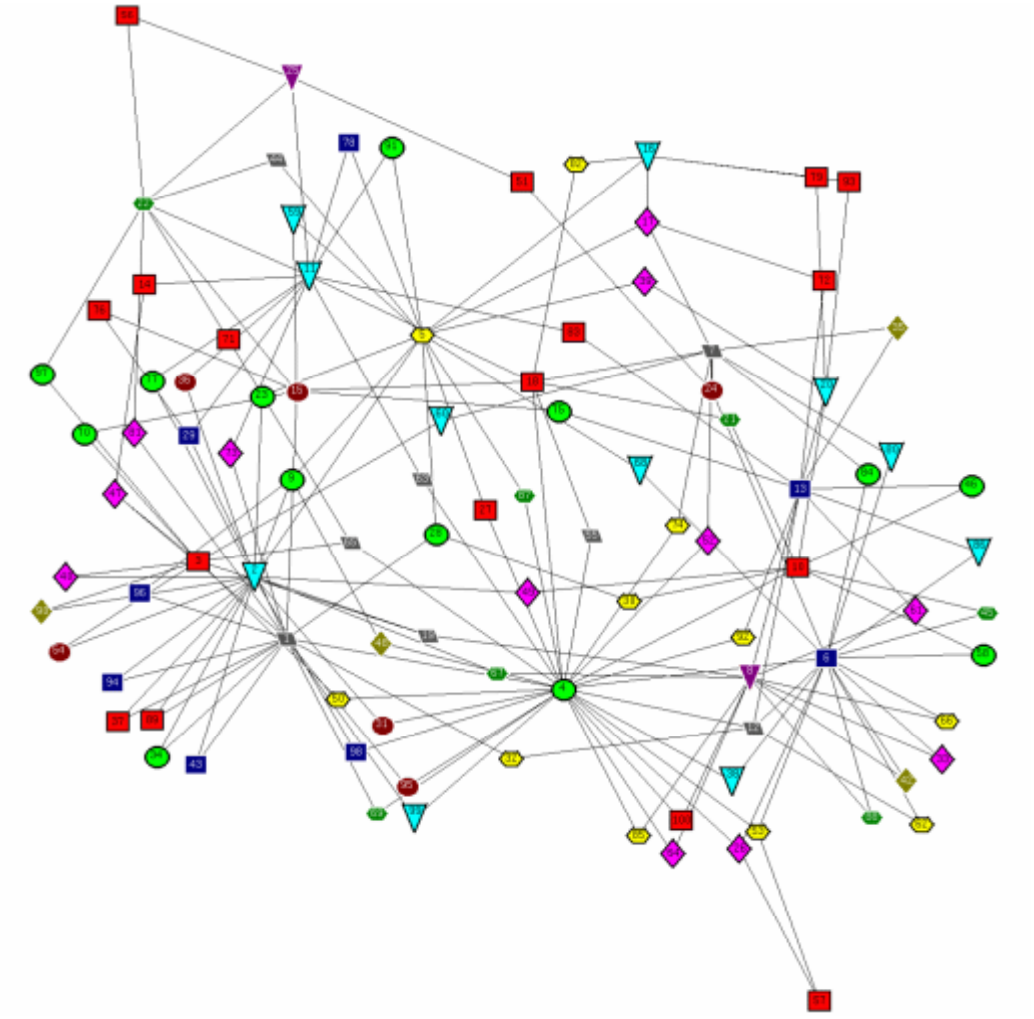
- Administração comercial: diagramas de processos, organogramas
- Bioinformática: mapas genéticos, interação de proteínas
- Bancos de dados: relacionamento de entidades
- Internet: conexões entre redes, projeto e otimização de *sites*
- Problemas em geral: diagramas de evolução, validação de circuitos, autômatos finitos, análise de linguagens, controle de fluxos, etc.

Exemplo: mapeamento de vizinhanças



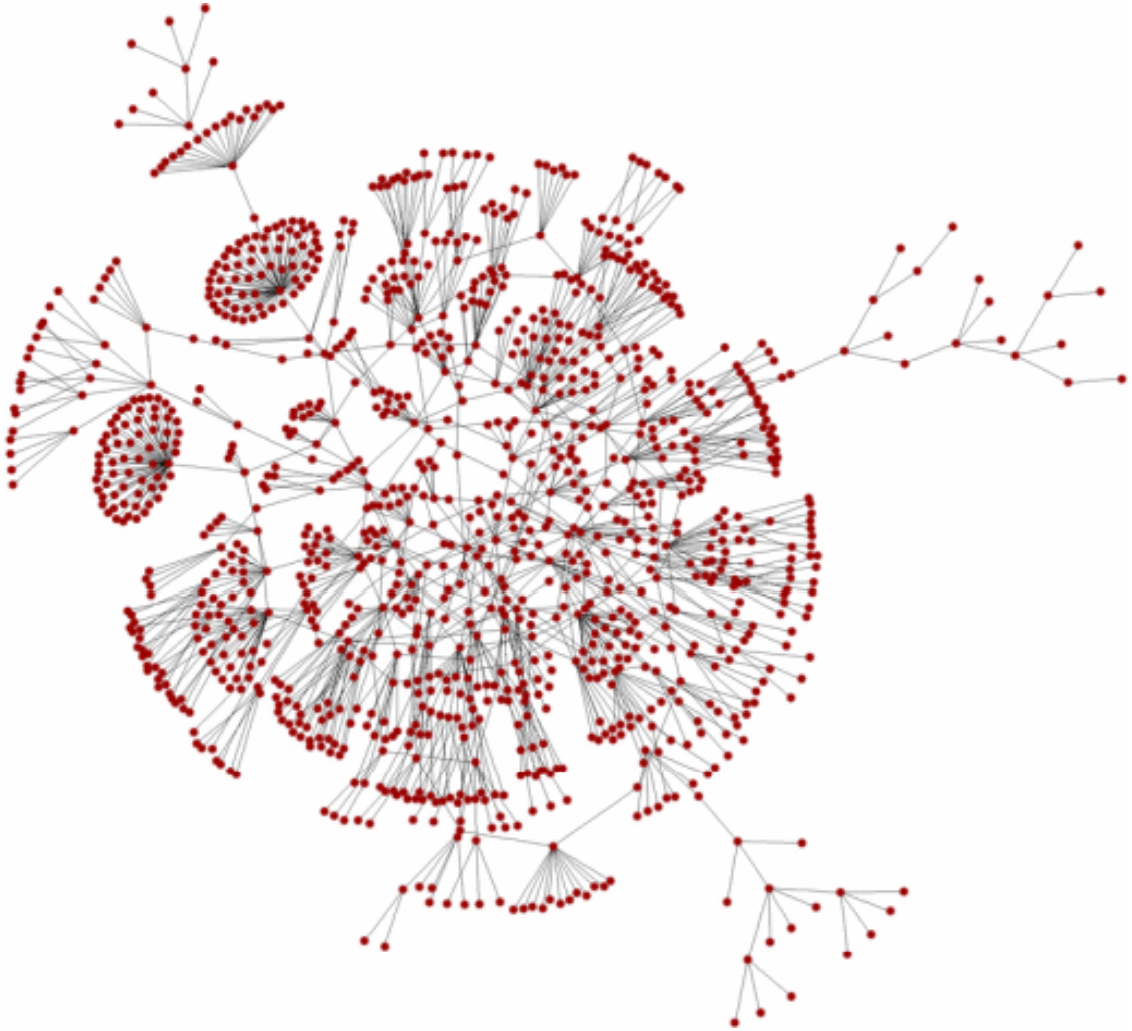
Exemplo: metrô de Moscou

Exemplo: escalonamento de provas



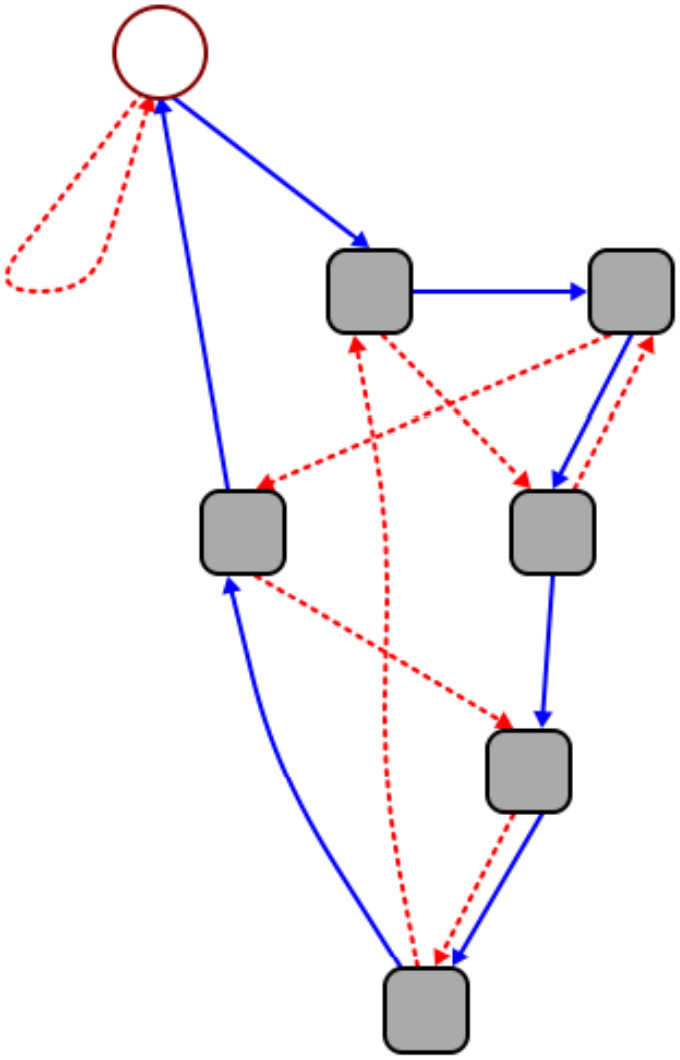
- *Southern Methodist University* (Dallas)
- Vértices representam provas, arestas significam conflitos (pelo menos um aluno em comum)
- Exames de mesma cor poderiam ser realizados simultaneamente (não estão conectados)

Exemplo: mapas de redes P2P



- Criado por *GnuMap*
- Identificação de sub-redes altamente conectadas
- Identificação dos servidores que ficariam desconectados em caso de falha

Exemplo: autômatos e algoritmos



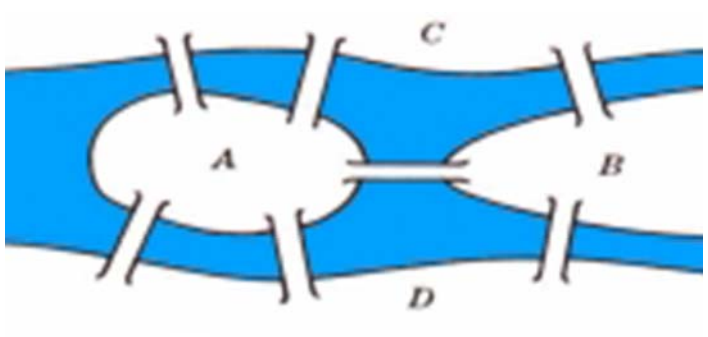
- Algoritmo para verificar a divisibilidade de um número por 7
- Percurso começa no círculo
- Para cada dígito d do número, andar d arestas azuis
- Ao passar para o dígito seguinte, andar 1 aresta vermelha
- Repetir o processo até terminar os dígitos do número
- Se terminar no círculo, o número é divisível por 7

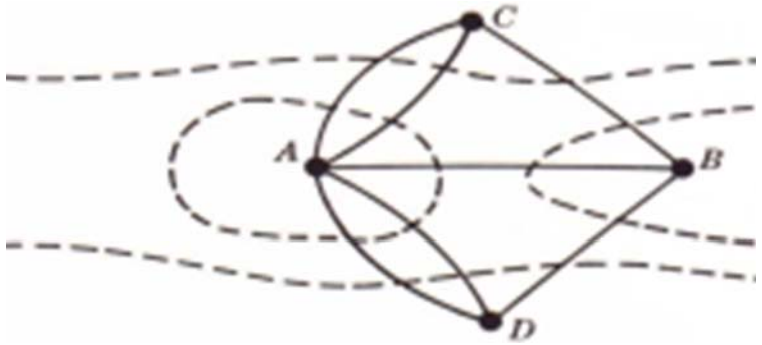
Arestas e vértices

- Uma aresta $e \in E$ é um par não-ordenado (u,v), onde $u,v \in V$.
- Neste caso, dizemos que os vértices u e v são adjacentes entre si, e que a aresta e é incidente em u e em v.
- Uma aresta $e = (u,v)$ é chamada de laço quando $u = v$.
- $d(u)$ é o grau do vértice u, isto é, o número de incidências em u.
- É fácil observar que $\Sigma_{u \in V} d(u) = 2.|E|$

Origem histórica

- Na cidade de Königsberg (atual Kaliningrado), havia sete pontes sobre o rio Pregel:

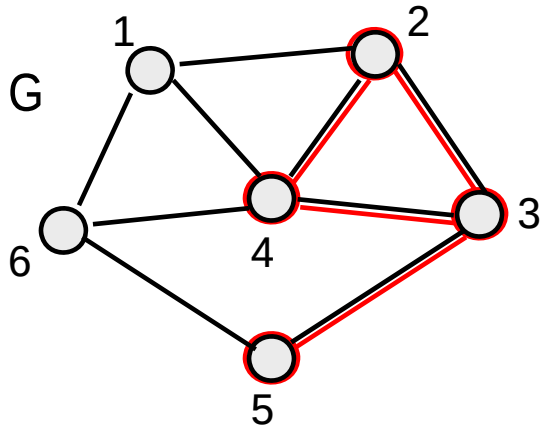




- Será possível fazer um passeio pela cidade, começando e terminando no mesmo local, e passando uma única vez em cada ponte?
- Euler (1736) resolveu o problema: esse passeio só seria possível se cada vértice do grafo tivesse grau par.
- Todo grafo com essa propriedade é chamado de *euleriano*.

Subgrafos

- O grafo $G'=(V',E')$ é um subgrafo de $G=(V,E)$ se $V' \subseteq V$ e $E' \subseteq E$, e todas arestas de E' têm seus vértices em V'.
- Quando $V'=V$, G' é chamado de subgrafo gerador de G.
- Seja $X \subseteq V$ e $E(X)$ o conjunto das arestas de E com ambos os vértices em X. Dizemos que $G(X)=(X,E(X))$ é o subgrafo de G induzido por X.
- Exemplo:



$X = \{2, 3, 4, 5\}$

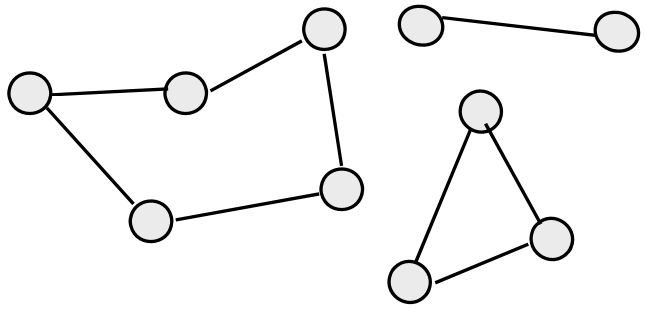
$G(X)$

Sequências de vértices

- Caminho é uma sequência alternada de vértices e arestas, onde cada aresta é incidente tanto ao vértice que a antecede como ao que a segue.
- Caminho simples é um caminho no qual cada vértice aparece uma única vez.
- Comprimento de um caminho é o seu número de arestas.
- Ciclo ou circuito é um caminho que começa e termina no mesmo vértice.

Componentes conexos

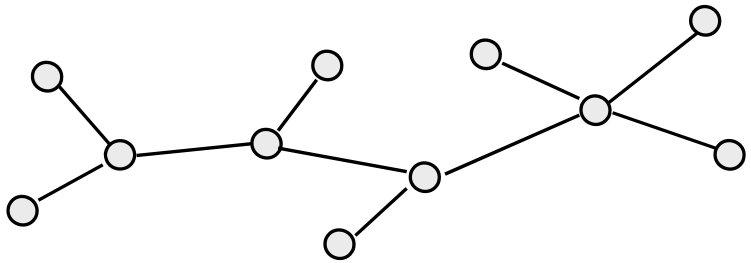
- Dois vértices *v* e *u* são conectados se houver um caminho entre eles. Componentes conexos são subconjuntos maximais de vértices conectados entre si.
- Um grafo é conexo se tiver um único componente conexo, ou seja, se todos seus vértices estiverem conectados entre si.
- Exemplo:



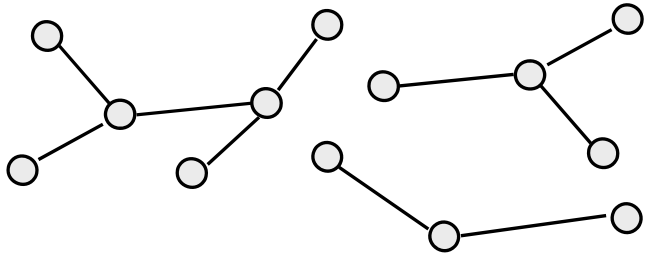
Grafo com 3 componentes conexos

Árvores e florestas

- Árvore é um grafo conexo sem circuitos.
- Exemplo:

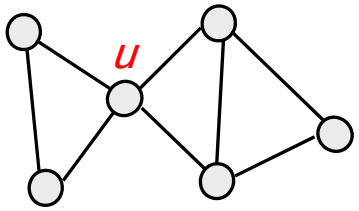


- Portanto, todo caminho simples é uma árvore.
- Floresta é um grafo cujos componentes conexos são árvores.
- Exemplo:

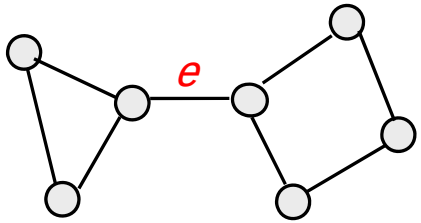


Vértices e arestas de corte

- Em um grafo G, $u \in V$ é chamado de vértice de corte (ou ponto de articulação) se a sua remoção desconecta G.
- Exemplo:

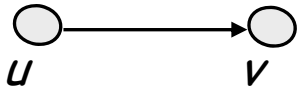


- Se um componente conexo de um grafo não possui vértices de corte, ele é chamado de componente biconexo.
- Analogamente, uma aresta e, cuja remoção ocasiona a desconexão do grafo, recebe o nome de aresta de corte (ou ponte).
- Exemplo:



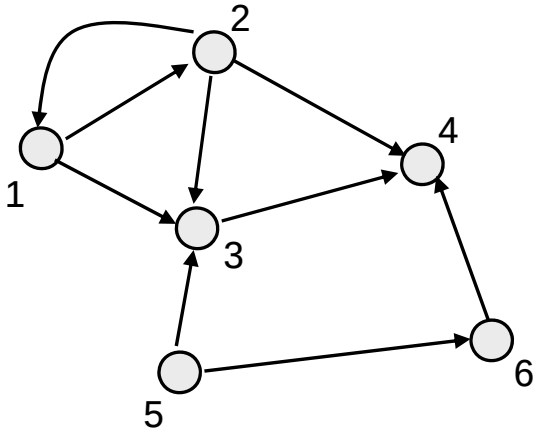
Digrafos ou grafos orientados

- Digrafos são grafos orientados, isto é, suas arestas possuem direção e são chamadas de arcos.
- Em um digrafo $G=(V,E)$, uma aresta $e \in E$ é um par ordenado (u,v), onde $u,v \in V$.



- v é sucessor de u
- u é predecessor de v

- Cada vértice v tem um grau de saída $d^+(v)$ e um grau de entrada $d^-(v)$, que correspondem respectivamente ao total de arcos que saem ou chegam em v.
- Exemplo:



$d^+(4) = 0$

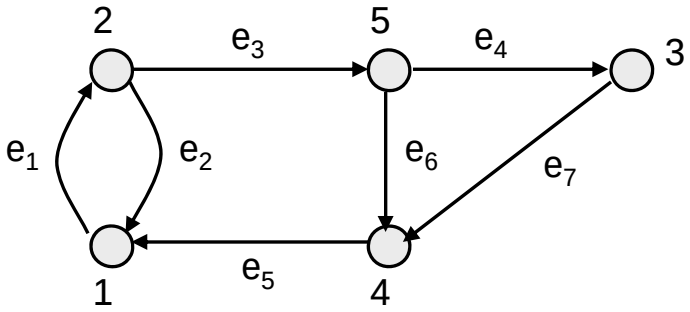
$d^-(4) = 3$

$d^+(6) = 1$

$d^-(1) = 1$

Sequências de arcos

- Caminho é uma sequência de arcos $e_1, e_2, \ldots, e_q$ tal que a extremidade inicial de e_i coincide com a final de e_{i-1}, $1 < i \leq q$.
- Ciclo ou circuito é um caminho que começa e termina no mesmo vértice.
- Exemplo:

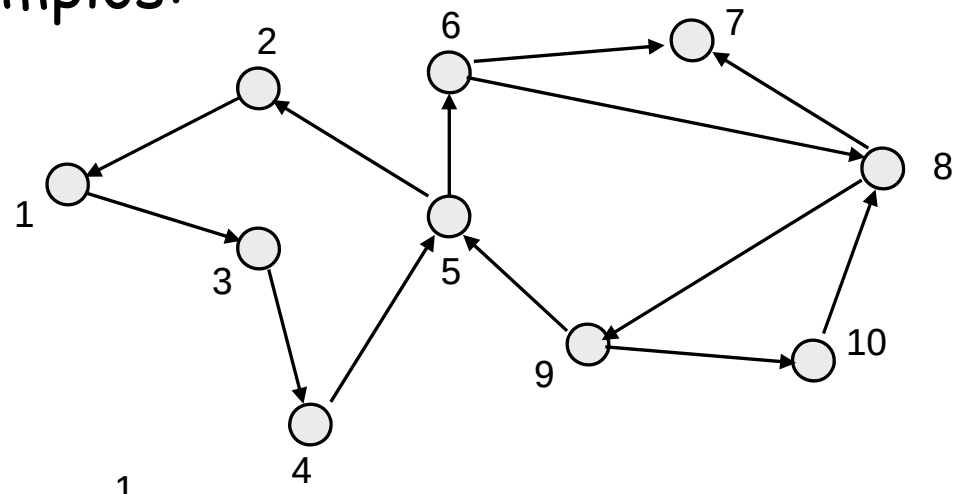


e_3, e_4, e_7, e_5: caminho entre os vértices 2 e 1

e_3, e_6, e_5, e_1: ciclo ou circuito

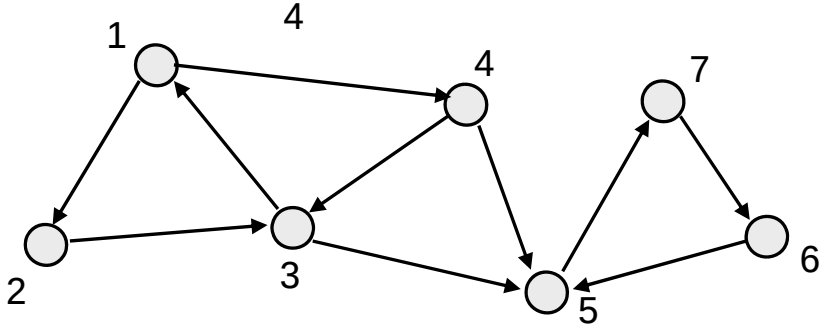
Componentes fortemente conexos

- Em um digrafo, dois vértices v_i e v_j estão conectados entre si se existem caminhos de v_i a v_j e de v_j a v_i.
- Desse modo, surge analogamente o conceito de componente fortemente conexo.
- Exemplos:



{1, 2, 3, 4, 5, 6, 8, 9, 10}

{7}



{1, 2, 3, 4}

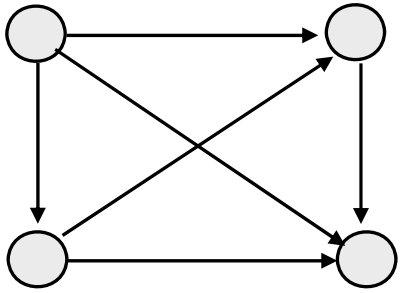
{5, 6, 7}

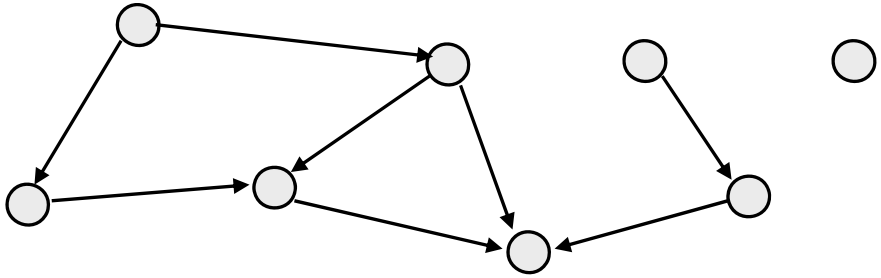
Ordenação topológica

- Uma ordenação topológica de um digrafo $G=(V,E)$ corresponde a uma bijeção $f: V \rightarrow \{1,2, \ldots, n\}$ tal que, para toda aresta $(u,v) \in E$, $f(u) < f(v)$.
- Em outras palavras, deseja-se numerar os vértices de tal modo que, se houver em G um caminho de u até v, então o número de u será menor que o de v.
- É fácil provar que um digrafo G admite uma ordenação topológica se e somente se for acíclico.
- Se os vértices forem alinhados de acordo com uma ordenação topológica, todos os arcos terão uma mesma direção.

Exercício

- Encontrar uma ordenação topológica para os digrafos abaixo.



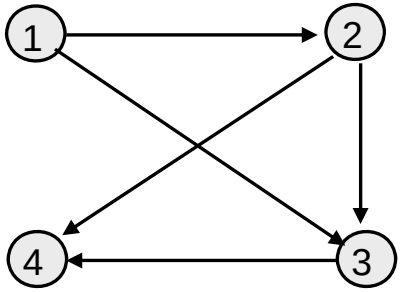


CES-11

- Grafos
  - Conceitos gerais e representações
- Algoritmos em grafos
  - Exploração sistemática em largura
  - Caminhos mais curtos
  - Exploração sistemática em profundidade
  - Teste de aciclicidade
  - Ordenação topológica
  - Componentes fortemente conexos
  - Vértices e arestas de corte
  - Árvore geradora de custo mínimo

Matriz de adjacências

- Matriz de adjacências é formada por n linhas e n colunas.
- A posição a_{ij} dessa matriz indica se o vértice v_j é sucessor ou não do vértice v_i.
- Exemplo:



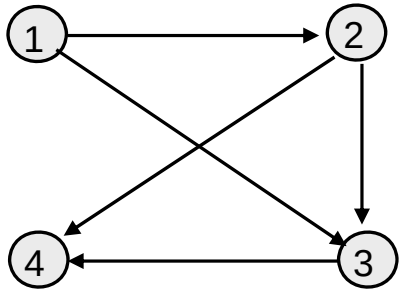
$$A_{n\times n} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

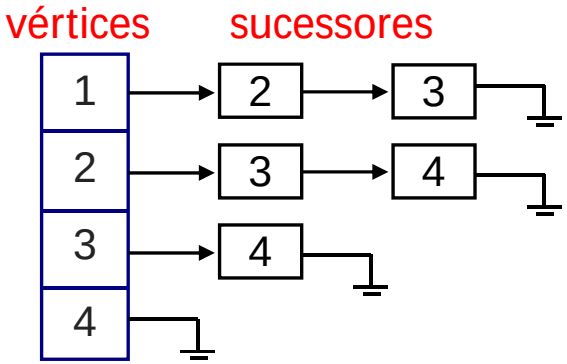
Tamanho da estrutura: $O(n^2)$

Útil quando grafo é denso: $m \sim n^2$

Lista de adjacências

- Lista de adjacências é formada por um vetor de *n* ponteiros, onde cada vértice aponta para seus sucessores (ou predecessores).
- Exemplo:





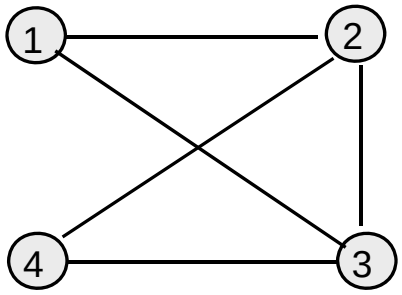
Útil quando grafo é esparso: $m << n^2$

Código para manipulação do grafo é mais complicado

Tamanho da estrutura: $O(n+m)$

Grafos não orientados

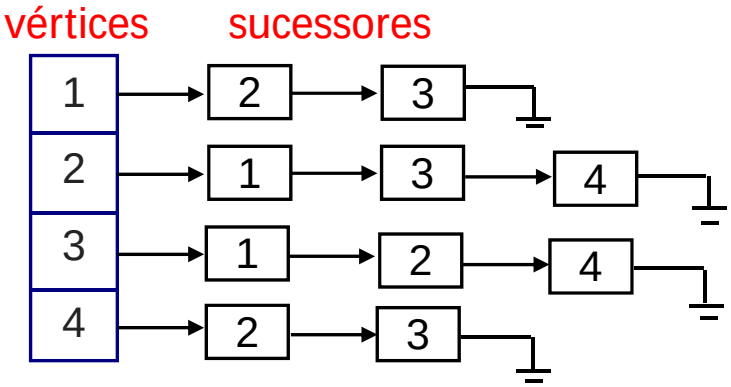
- Exemplo:



Matriz de adjacências
(simétrica em relação
à diagonal principal)

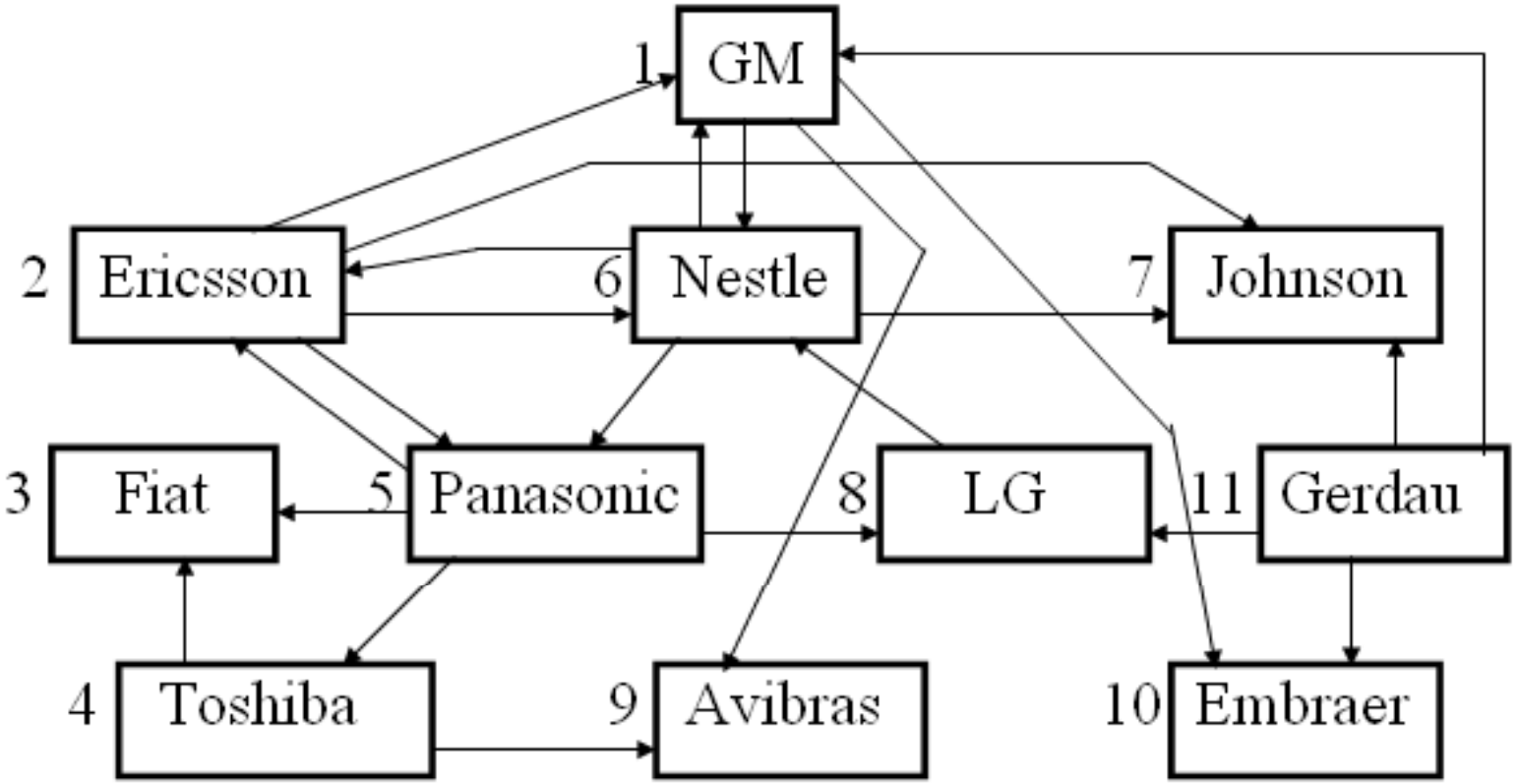
$$A_{n\times n} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Lista de adjacências
(possui 2m nós)



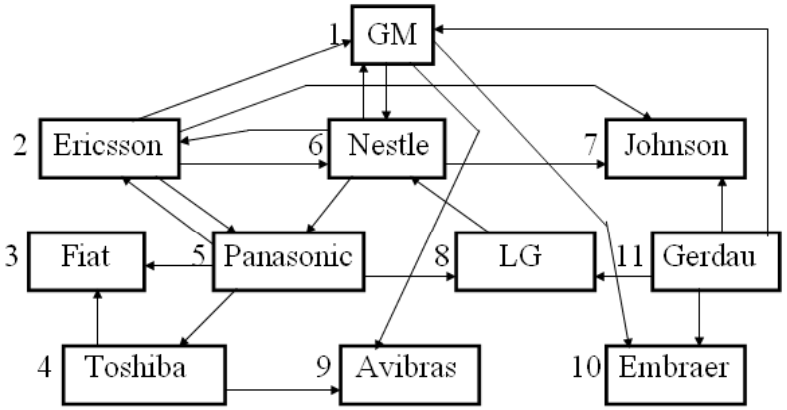
Exemplo 1

- Fornecimento de produtos entre fábricas:



Exemplo 1: matriz de adjacências

```
int maxvert = 100;

struct CelulaVertice  {
  char nome[30];
  bool adjacente[maxvert];
};

struct Digrafo  {
  CelulaVertice vertices[maxvert];
  int nvert;
};

Digrafo Fornecimento;
```



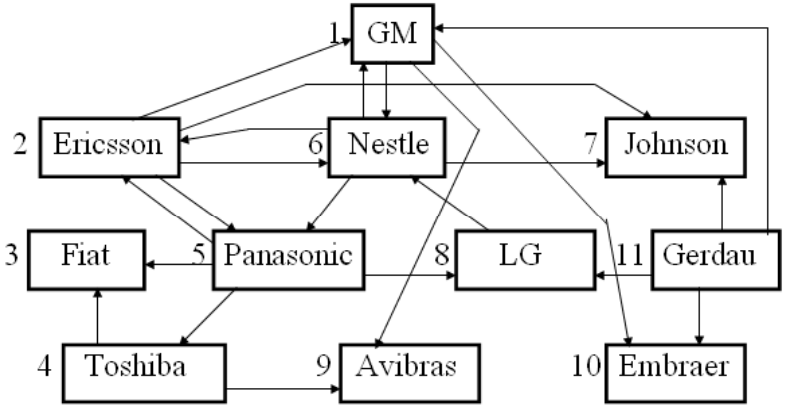
	Fábrica	1	2	3	4	5	6	7	8	9	10	11
1	GM						V			V	V	
2	Ericsson	V				V	V	V				
3	Fiat											
4	Toshiba			V						V		
5	Panasonic		V	V	V				V			
6	Nestle	V	V			V		V				
7	Johnson											
8	LG						V					
9	Avibras											
10	Embraer											
11	Gerdau	V						V	V		V	

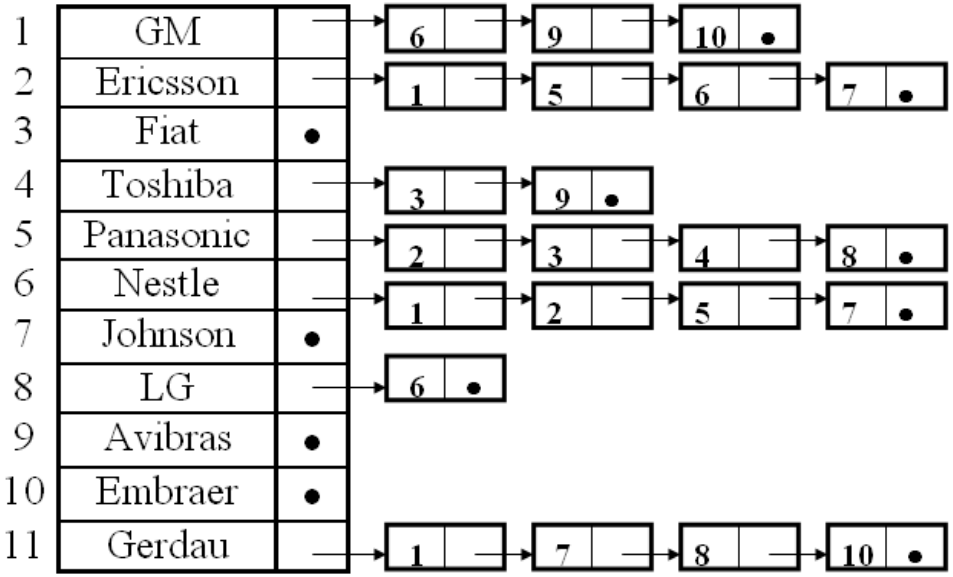
Se houver informação no arco, basta alterar esse tipo (poderia ser até uma estrutura complexa)

Exemplo 1: lista de adjacências

```
int maxvert = 100;
typedef int vertice;

struct CelulaAdj {
  vertice vert;
  CelulaAdj *prox;
};

struct CelulaVertice {
  char infoVert[30];
  CelulaAdj *listaAdj;
};

struct Digrafo {
  int nvert;
  CelulaVertice vertices[maxvert];
};

Digrafo Fornecimento;
```

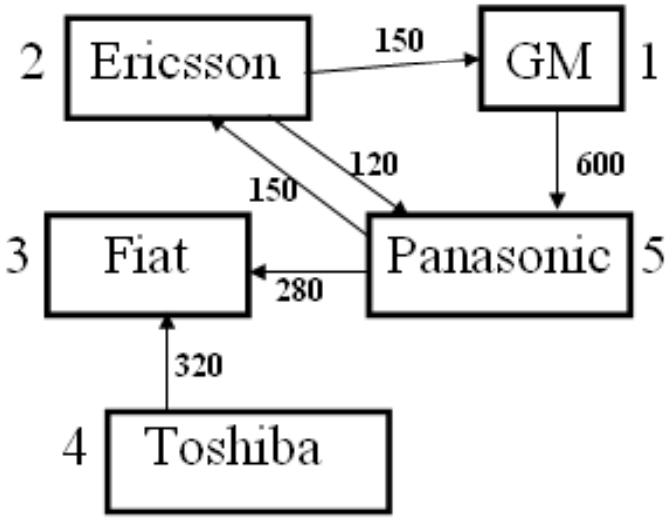
Se houver informação no arco, basta acrescentar um campo aqui

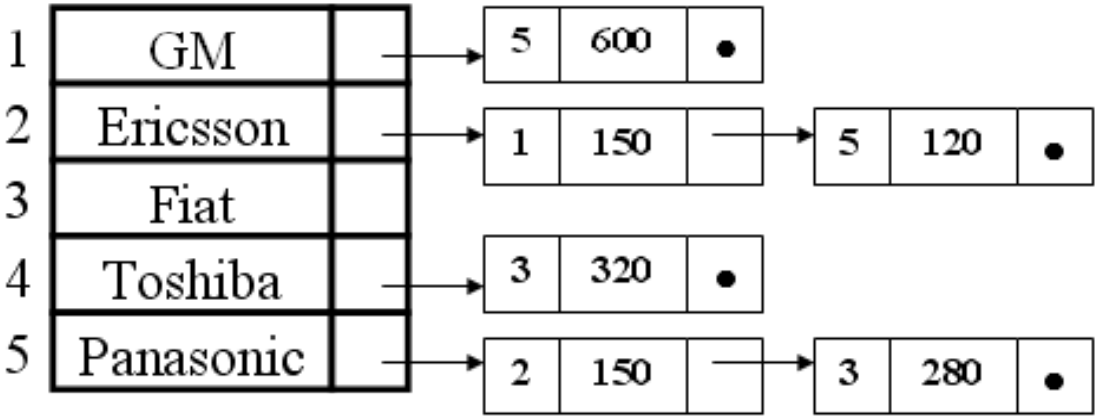




Exemplo 2: armazenamento de digrafo

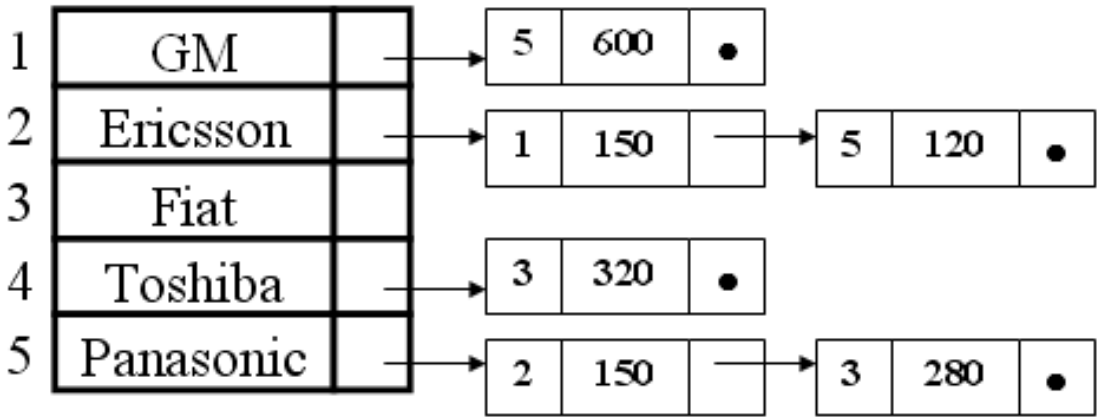
- Seja um digrafo de fornecimento entre fábricas:
  - Arcos: custo de fornecimento mensal (em R$)
- Deseja-se representá-lo através de uma lista de adjacências, onde todas as dimensões são alocadas dinamicamente.





Exemplo 2: armazenamento de digrafo

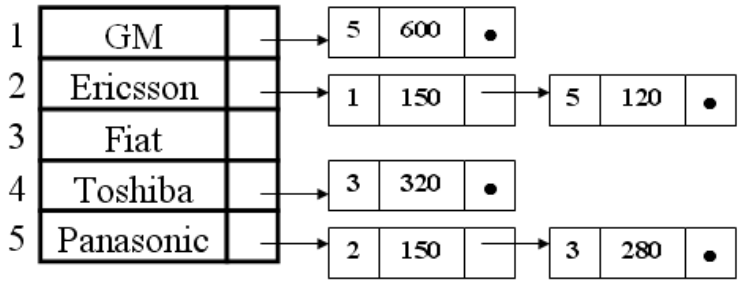
```
typedef int vertice;

struct CelulaAdj {
  vertice vert;
  int custo;
  CelulaAdj *prox;
};

struct CelulaVertice {
  char nome[30];
  CelulaAdj *listaAdj;
};

struct Digrafo {
  int nvert;
  CelulaVertice *vertices;
};

Digrafo G;
```



Vetor de vértices alocados dinamicamente

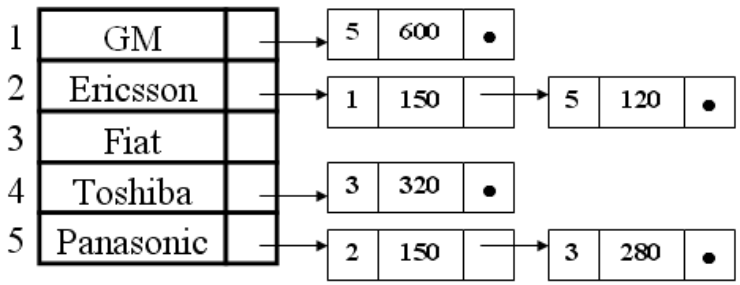
Exemplo 2: armazenamento de digrafo

```
// Função lerGrafo: lê e armazena um digrafo na variável global G

void lerGrafo () {
  int i, j, n;
  CelulaAdj *p;

  // Alocação do vetor de vértices

  printf ("Numero de fabricas: ");
  scanf ("%d", &G.nvert);
  G.vertices=(CelulaVertice*)malloc((G.nvert+1)*sizeof(CelulaVertice));

  // Leitura do nome das fábricas

  for (i = 1; i <= G.nvert; i++) {
    printf ("Nome da fabrica %d: ", i);
    scanf ("%s", &G.vertices[i].nome);
    G.vertices[i].listaAdj = NULL;
  }

  // Continua...
```



Exemplo 2: armazenamento de digrafo

```
// Leitura dos fornecimentos e seus custos

  for (i = 1; i <= G.nvert; i++) {
    printf ("Numero de fornecimentos de %s: ", G.vertices[i].nome);
    scanf("%d", &n);

    for (j = 1; j <= n; j++) {
      printf ("Numero da fabrica e custo:");
      p = G.vertices[i].listAdj;
      G.vertices[i].listaAdj = (CelulaAdj*)malloc(sizeof(CelulaAdj));
      G.vertices[i].listaAdj->prox = p;
      scanf("%d %d", &G.vertices[i].listaAdj->vert,
                              &G.Vertices[i].listaAdj->custo);
    }

  }
}
```



Representação mais adequada

- Critérios para se escolher a melhor representação:
  - Espaço de armazenamento (depende do tamanho do grafo)
  - Teste de pertinência de uma aresta (matriz)
  - Verificação do grau de um vértice (lista)
  - Inserção ou remoção de uma aresta (matriz)
  - Percurso no grafo (lista)
- Geralmente, a lista de adjacências costuma ser mais vantajosa