

# **Algoritmos e Estruturas de Dados II**

## **Grafos – tipo abstrato de dados**

**Thiago A. S. Pardo**

**Profa. M. Cristina**

**Material de aula da Profa.**

**Josiane M. Bueno**

# Grafos

## Tipo Abstrato de Dados

- Última aula: TAD grafo?

# Grafos

## Tipo Abstrato de Dados

- Última aula: TAD grafo?
  - Dados/informação (encapsulados)
    - Estruturas de dados adequadas
  - Operações

# Grafos

## Estruturas de Dados

- A escolha da estrutura de dados certa para a representação de grafos tem um enorme impacto no desempenho de um algoritmo.
- Há duas representações usuais:
  - **Matriz de Adjacências**
  - **Listas de Adjacências**

## Grafos

# Matriz de Adjacências

- Dado um grafo  $G = (V, E)$ , a **matriz de adjacências**  $M$  é uma matriz de ordem  $|V| \times |V|$ , tal que:

$|V|$  = número de vértices

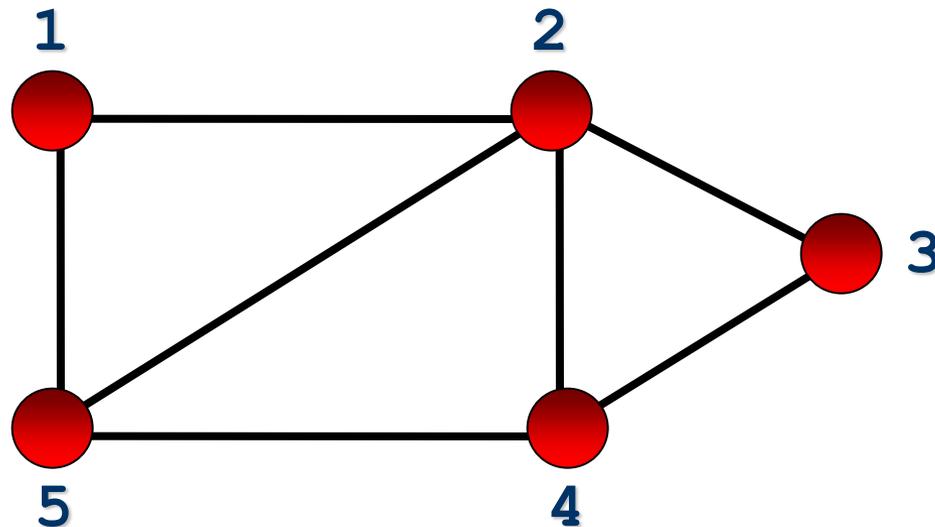
$M[i,j] = 1$ , se existir aresta de  $i$  a  $j$

$M[i,j] = 0$ , se NÃO existir aresta de  $i$  a  $j$

# Grafos

## Matriz de Adjacências

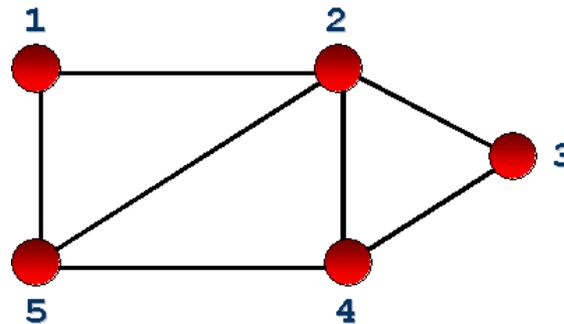
- Qual a matriz de adjacências do grafo a seguir?



# Grafos

## Matriz de Adjacências

- Resposta:



$M =$

	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

← vértices

Grafo não direcionado  
→ Matriz simétrica

# Grafos

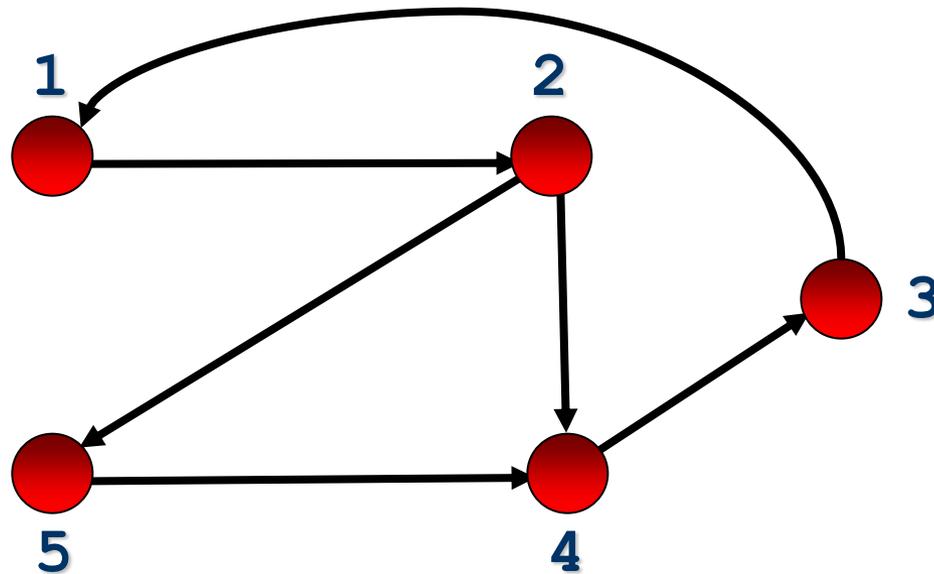
## Matriz de Adjacências

- Se o grafo for **direcionado**
  - $M[i,j]$  deve indicar ou não a presença de uma aresta divergente de  $i$  e convergente em  $j$ , ou seja  $i \rightarrow j$

# Grafos

## Matriz de Adjacências

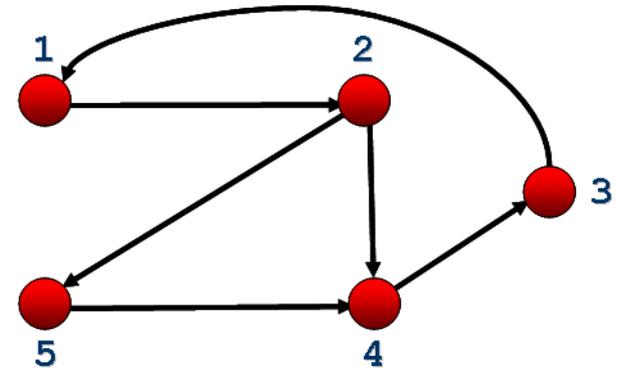
- Qual a matriz de adjacências do dígrafo a seguir?



# Grafos

## Matriz de Adjacências

- Possível resposta:



	1	2	3	4	5
1	0	1	0	0	0
2	0	0	0	1	1
3	1	0	0	0	0
4	0	0	1	0	0
5	0	0	0	1	0

**Matriz  
assimétrica**

# Grafos

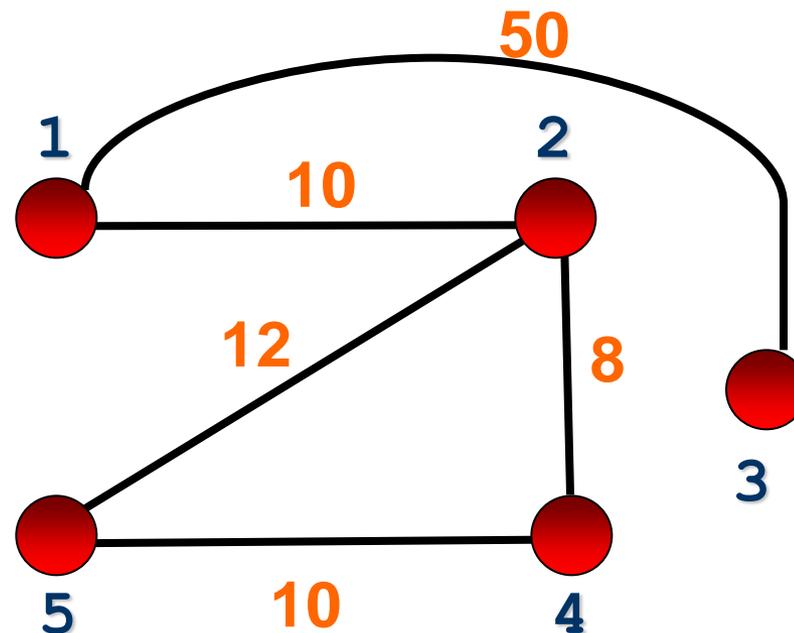
## Matriz de Adjacências

- Se o grafo for **valorado**
  - $M[i,j]$  deve conter o peso associado com a aresta
  - Se não existir uma aresta entre  $i$  e  $j$ , então é necessário utilizar um valor que não possa ser usado como peso (como o valor 0 ou negativo, por exemplo)

# Grafos

## Matriz de Adjacências

- Qual a matriz de adjacências do grafo valorado a seguir? Suponha que o grafo represente a distância em km entre cidades

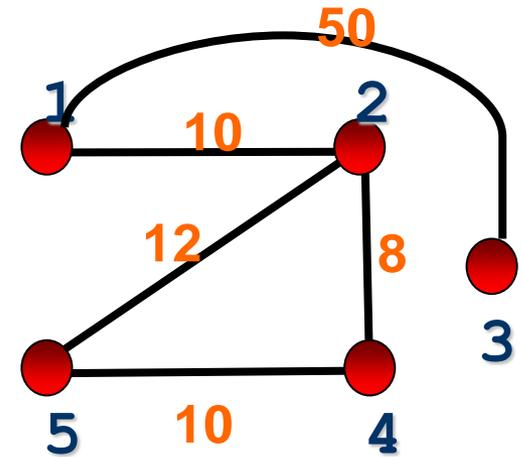


# Grafos

## Matriz de Adjacências

- Possível resposta:

	1	2	3	4	5
1	0	10	50	-1	-1
2	10	0	-1	8	12
3	50	-1	0	-1	-1
4	-1	8	-1	0	10
5	-1	12	-1	10	0



# Grafos

## Matriz de Adjacências

- Forma mais simples de representação
- Propriedades
  - armazenamento: ?
  - teste se aresta  $(i,j)$  está no grafo: ?

# Grafos

## Matriz de Adjacências

- Forma mais simples de representação
- Propriedades
  - armazenamento:  $O(|V|^2)$
  - teste se aresta  $(i,j)$  está no grafo:  $O(1)$

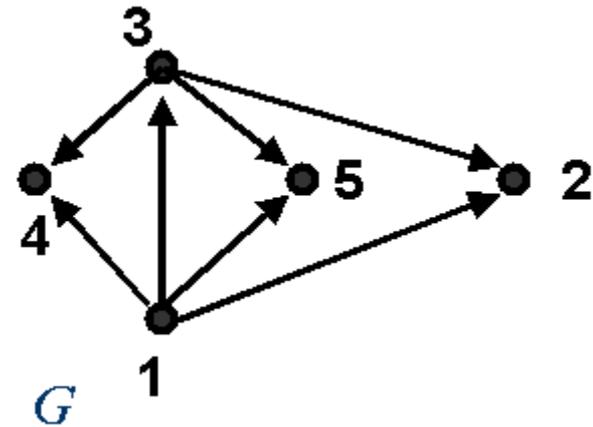
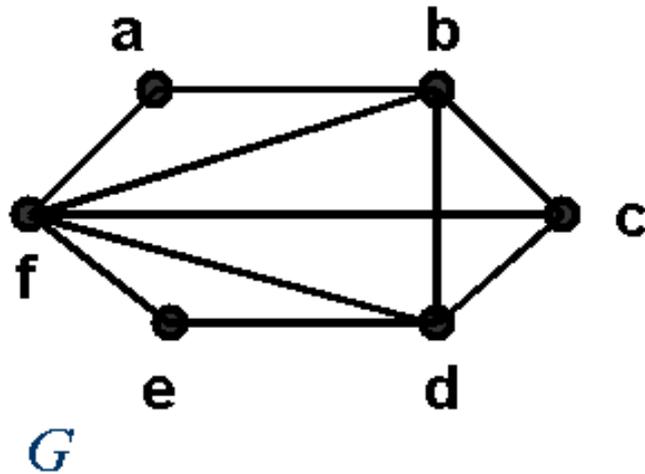
# Grafos

## Matriz de Adjacências

- Representação útil para grafos densos
- Boa para quando desejamos buscar arestas/vértices adjacentes rapidamente
- Ruim quando se necessita examinar a matriz toda:  $O(|V|^2)$
- Inserção e remoção de vértices e arestas: representação boa ou ruim?

## Grafos

# Exercício de Fixação



- Represente os grafos acima utilizando matrizes de adjacências

# Grafos

## Matriz de Adjacências

- Implementação de algumas das operações mais comuns
  - Criar grafo vazio
  - Inserir aresta
  - Dois vértices dados são adjacentes?
  - Retirar aresta
  - Calcular o grau de um vértice (de entrada/saída, se dígrafo)
  - Obter lista de vértices adjacentes a um determinado vértice
  - Imprimir grafo
  - Outras: caminho entre 2 vértices, etc. (não básicas)

# GrafoNaoDirecionado.h

```
#define MaxNumVertices 100

typedef int elem;

typedef struct {
    elem mAdj[MaxNumVertices][MaxNumVertices];
    int NumVertices;
} Grafo;

void criar(Grafo*, int, int*);
void inserir_aresta(Grafo*, int, int, elem, int*);
int existe_aresta(Grafo*, int, int, int*);
void retirar_aresta(Grafo*, int, int, elem*, int*);
int grau_vertice(Grafo*, int, int*);
void imprimir(Grafo*); ...
```

## GrafoNaoDirecionado.c

```
#include "GrafoNaoDirecionado.h"
```

```
/*função que inicializa um grafo com um determinado número de vértices dado pelo usuário */
```

```
void criar(Grafo *G, int NumVertices, int *erro) { }
```

## GrafoNaoDirecionado.c

*/\*função que **insere uma aresta de peso P entre V1 e V2** no grafo\*/*

```
void inserir_aresta(Grafo *G, int V1, int V2, elem P, int *erro) { }
```

## GrafoNaoDirecionado.c

```
/*função que verifica se uma aresta existe entre 2 vértices*/
```

```
int existe_aresta(Grafo *G, int V1, int V2, int *erro) { }
```

## GrafoNaoDirecionado.c

*/\*função que **retira uma aresta do grafo**, retornando seu peso\*/*

```
void retirar_aresta(Grafo *G, int V1, int V2, elem *P, int *erro) { }
```

## GrafoNaoDirecionado.c

```
/*função que calcula o grau de um vértice*/
```

```
int grau_vertice(Grafo *G, int V1, int *erro) { }
```

## GrafoNaoDirecionado.c

***/\*função que retorna todos os vértices adjacentes a um vértice dado\*/***

**Para fazer em casa e testar juntamente com todas as outras**

# Exercício

- **Modifique o TAD anterior para o caso de Dígrafo**

# Grafos

## Matriz de Adjacências

- **Exercício**

- Implementar sub-rotina que encontre a aresta de menor peso em um grafo valorado

# Grafos

## Matriz de Adjacências

- **Questão**
  - Cada grafo associa-se a uma única matriz de adjacência. O inverso é verdade?