Algoritmos e Estruturas de Dados III
Grafos – tipo abstrato de dados

Anderson Canale Garcia

Material de aula de
Cristina D. Aguiar, Thiago A. S. Pardo, M. Cristina de Oliveira, Josiane M. Bueno e Elaine P. M. de Souza

Tipo Abstrato de Dados (TAD)

- TAD grafo
  - Dados/informação (encapsulados)
    - Estruturas de dados adequadas
  - Operações

- A escolha da estrutura de dados adequada para a representação de grafos tem um enorme impacto no desempenho de um algoritmo

Grafos
Estruturas de Dados

- Há duas representações usuais
  - Matriz de Adjacências
  - Listas de Adjacências

- Independência de implementação
  - permite alterar a implementação do tipo abstrato de dados sem ter que alterar a implementação das aplicações que o utilizam

Matriz de Adjacências

Matriz de Adjacências

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

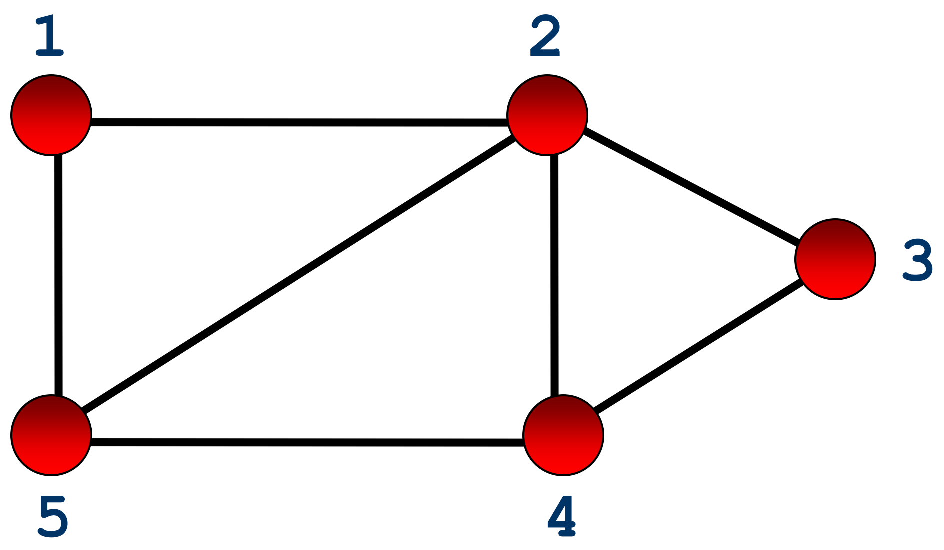
|V| = número de vértices

M[u,v] = 1, se existir aresta de u a v

M[u,v] = 0, se NÃO existir aresta de u a v

Grafos
Matriz de Adjacências

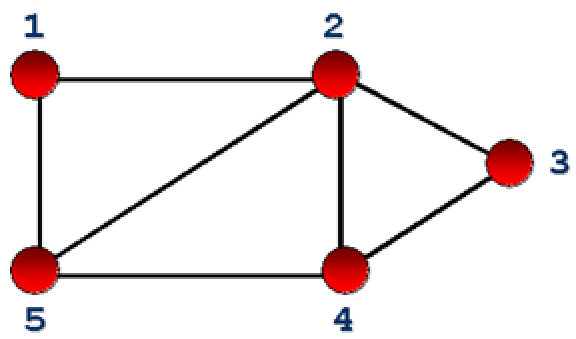
- **Qual a matriz de adjacências do grafo a seguir?**



Grafos

Matriz de Adjacências

- **Resposta:**



$$
M = \begin{array}{c} \\ \\ \end{array}
\begin{array}{ccccc|c}
1 & 2 & 3 & 4 & 5 & \\
\hline
0 & 1 & 0 & 0 & 1 & 1 \\
1 & 0 & 1 & 1 & 1 & 2 \\
0 & 1 & 0 & 1 & 0 & 3 \\
0 & 1 & 1 & 0 & 1 & 4 \\
1 & 1 & 0 & 1 & 0 & 5
\end{array}
$$

vértices

Grafo simétrico

Grafos

Matriz de Adjacências

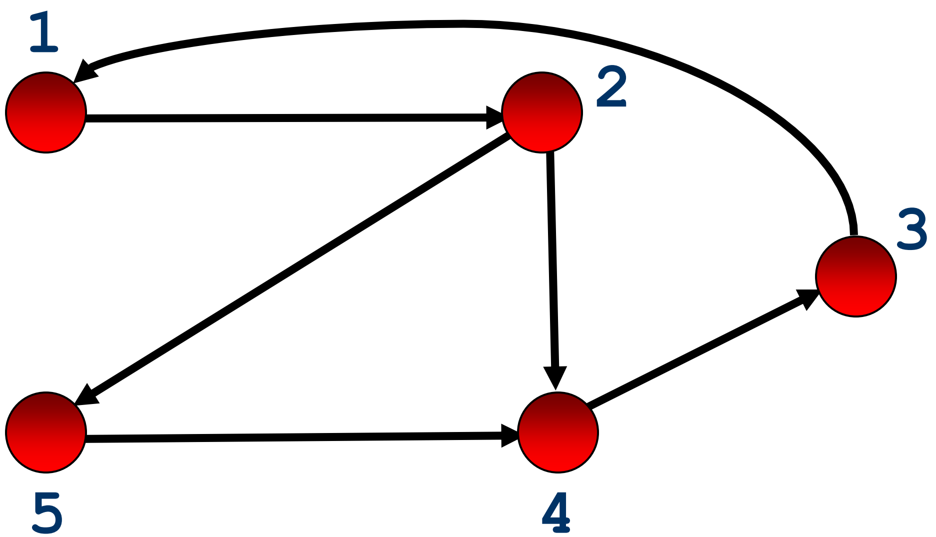
- **Se o grafo for direcionado**

  - **M[u,v] indica uma aresta que <u>sai</u> do vértice u e <u>chega</u> no vértice v, ou seja u □ v**

Grafos

Matriz de Adjacências

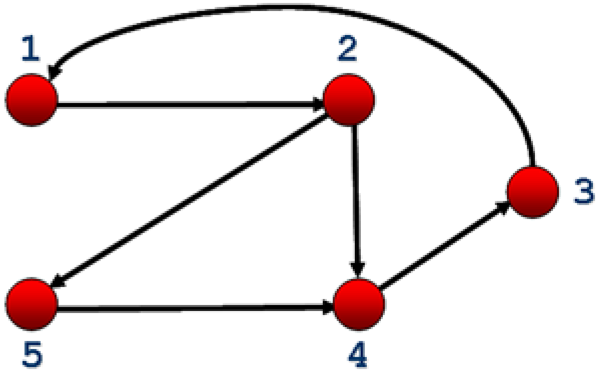
- **Qual a matriz de adjacências do dígrafo a seguir?**



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

Grafo assimétrico

Grafos

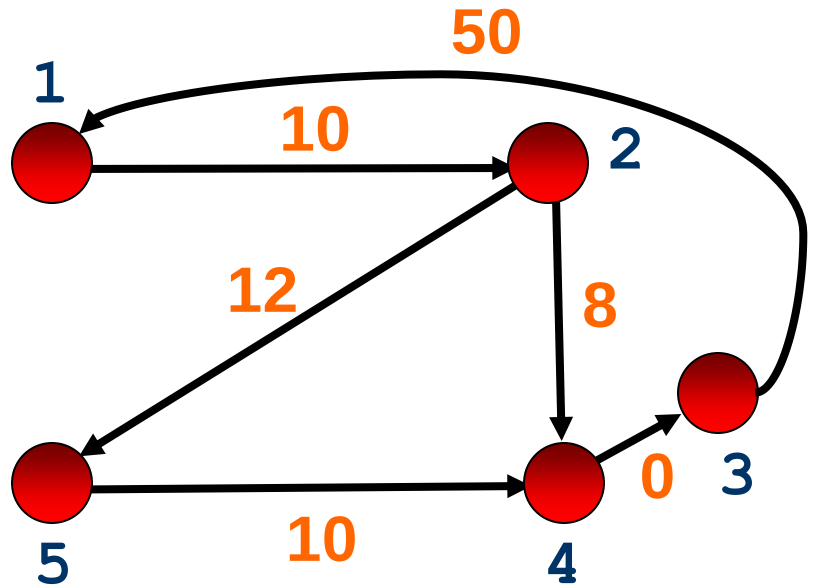
Matriz de Adjacências

- **Se o grafo for ponderado**
  - **M[u,v] deve conter o peso associado com a aresta**
  - **Se não existir uma aresta entre u e v, então é necessário utilizar um valor que não possa ser usado como peso (como o valor 0 ou -1, por exemplo)**

Grafos

Matriz de Adjacências

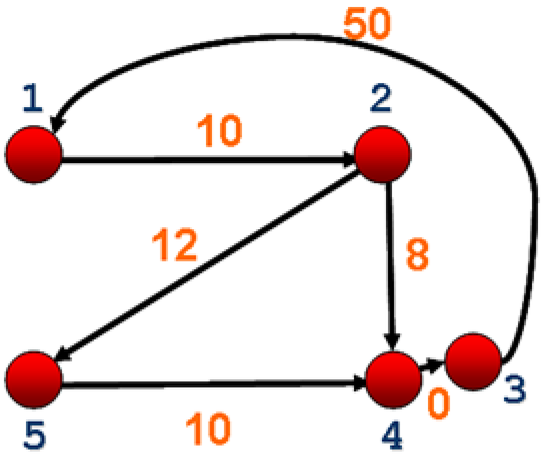
- **Qual a matriz de adjacências do grafo direcionado e ponderado a seguir? Suponha que o grafo represente a distância em km entre cidades**



Grafos

Matriz de Adjacências

- **Possível resposta:**



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

Grafo simétrico ou assimétrico?

Grafos
Matriz de Adjacências

- **Característica**
  - forma mais simples de representação de grafos
- **Propriedades**
  - espaço de armazenamento: $O(|V|^2)$
    - matriz é simétrica para grafos não direcionados, sendo que aproximadamente metade do espaço pode ser economizado representando a matriz triangular superior ou inferior
  - teste se aresta (u,v) está no grafo: $O(1)$
    - tempo necessário para acessar um elemento é independente de |V| ou |A|

Matriz de Adjacências

- Vantagens
  - representação útil para grafos densos, nos quais |A| é próximo a $|V|^2$
  - boa quando se deseja buscar arestas rapidamente
- Desvantagens
  - ruim quando se necessita examinar a matriz toda: $O(|V|^2)$

- Perguntas
  - inserção e remoção de vértices: representação boa ou ruim?
  - inserção e remoção de arestas: representação boa ou ruim?

Listas de Adjacências

Grafos

Listas de Adjacências

- **Dado um grafo $G = (V, A)$, as listas de adjacências L são um conjunto de |V| listas $L(v)$, uma para cada vértice v pertencente a V**

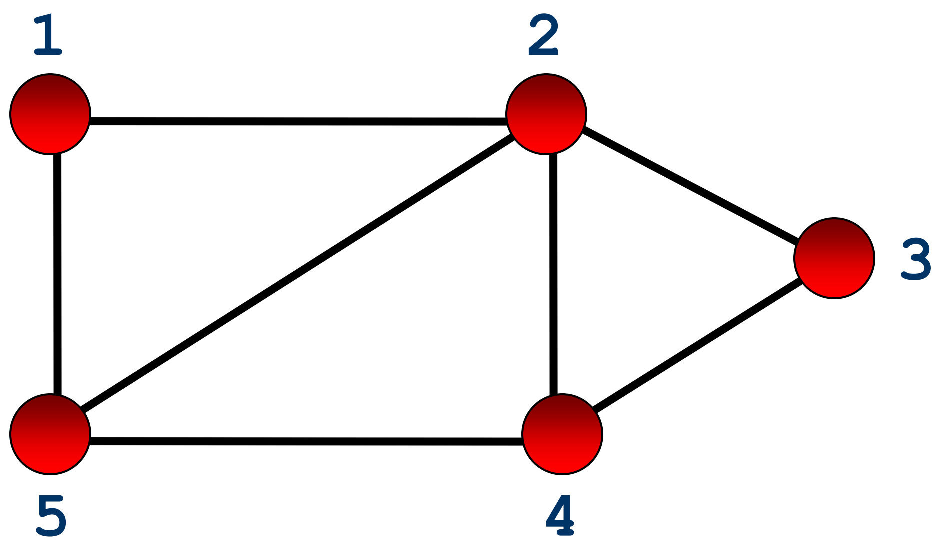
- **Cada lista $L(v)$ é denominada lista de adjacências do vértice v e contém os vértices w adjacentes a v em G**

- **Ou seja, as listas de adjacências consistem tradicionalmente em um vetor de |V| elementos que são capazes de apontar, cada um, para uma lista linear**
  - **O i-ésimo elemento do vetor aponta para a lista linear das arestas que são adjacentes ao vértice i**

Grafos

Listas de Adjacências

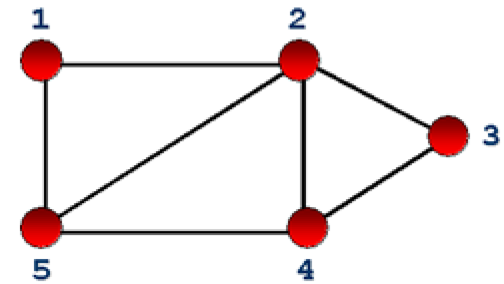
- **Como são as listas de adjacências do grafo a seguir?**

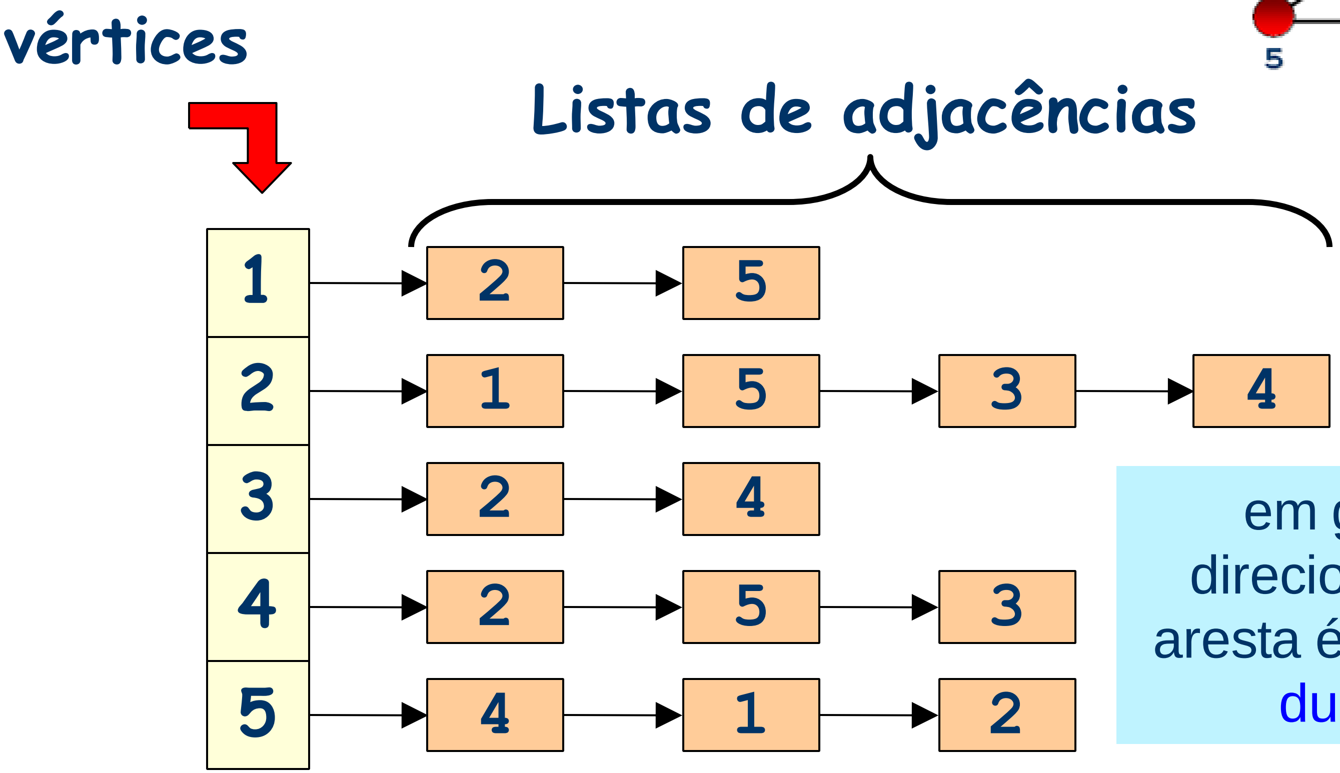


Grafos

Listas de Adjacências

- **Possível resposta:**



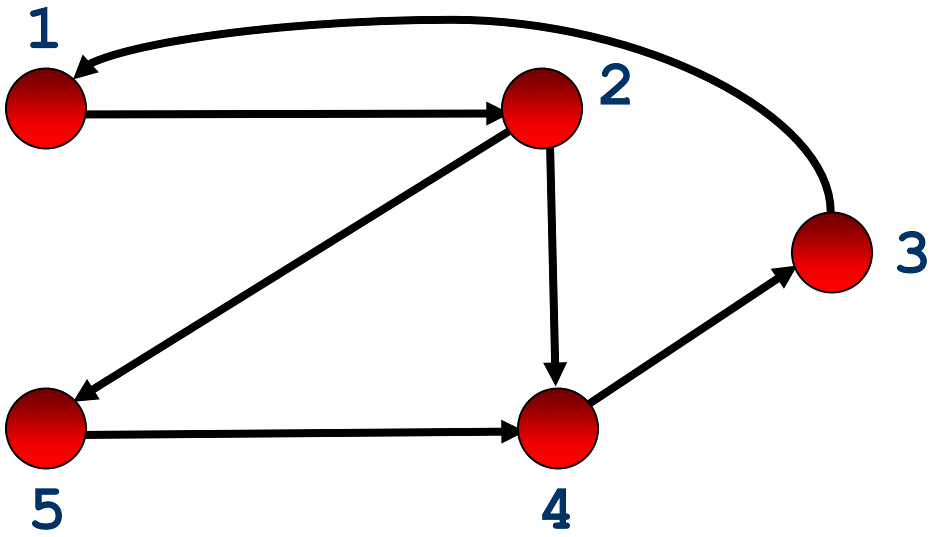


em grafos não direcionados, cada aresta é representada duas vezes

Grafos

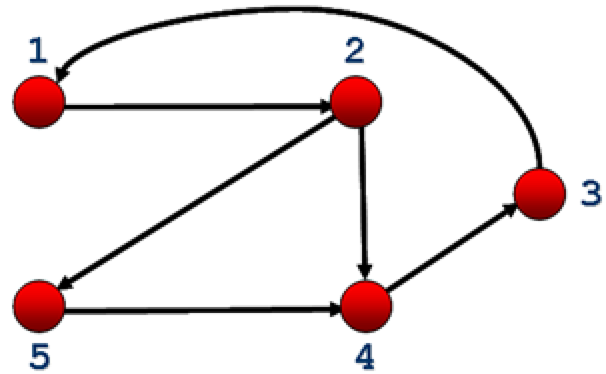
Listas de Adjacências

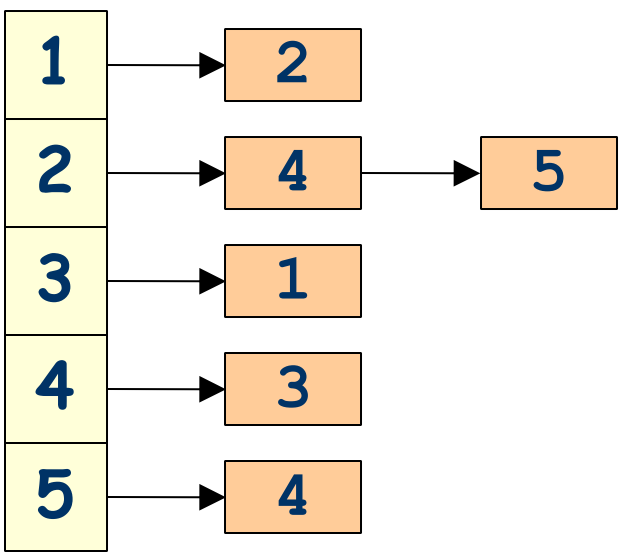
- Como representar o dígrafo abaixo?



Grafos
Listas de Adjacências

- **Possível resposta:**

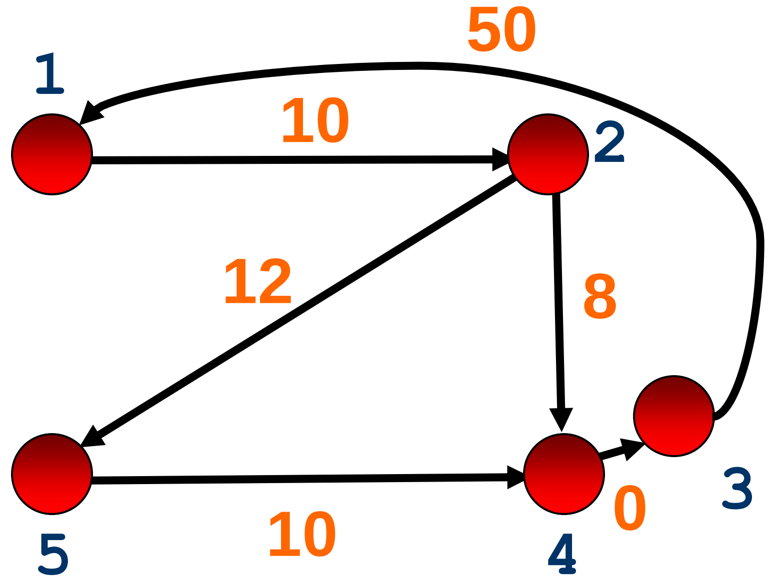




em grafos direcionados, cada vértice aponta para os seus vértices adjacentes

Grafos
Listas de Adjacências

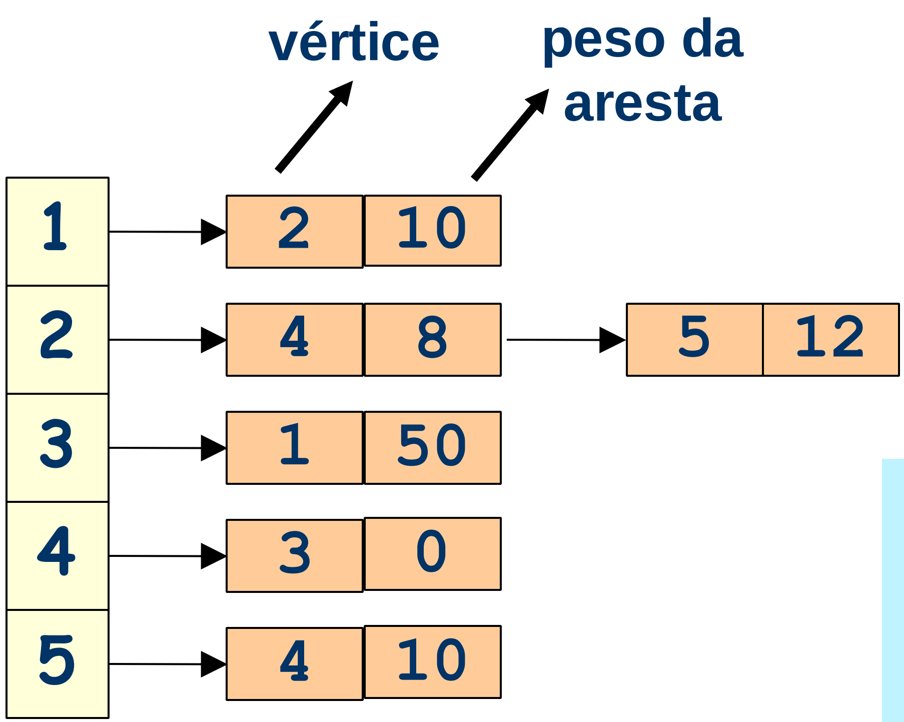
- **Como representar o grafo direcionado e ponderado abaixo?**

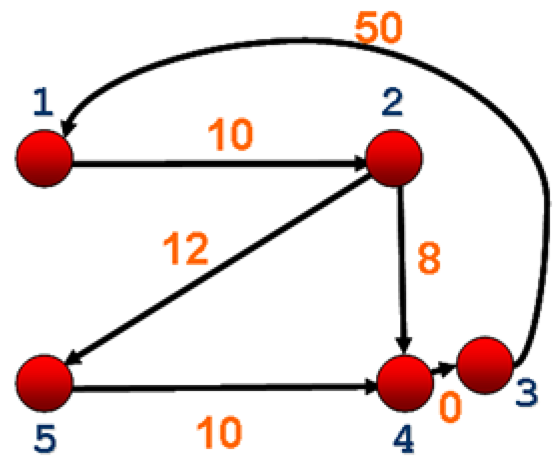


Grafos

Listas de Adjacências

- **Possível resposta:**





em grafos ponderados, cada elemento da lista armazena o rótulo do vértice e o peso da aresta correspondente

Grafos
Listas de Adjacências

- **Características**
  - **maior complexidade na representação de grafos**
- **Propriedades**
  - **espaço de armazenamento: $O(|V|+|A|)$**
  - **teste se aresta (u,v) está no grafo: $O(d_u)$**
    - **grafos não direcionados => d_u = grau do vértice u**
    - **grafos direcionados => d_u = grau de saída do vértice u**
    - **$d_u \approx |V|$ para vértices com muitas arestas**

Grafos
Listas de Adjacências

- **Vantagens**
  - **representação útil para grafos esparsos, nos quais |A| é muito menor do que $|V|^2$**
  - **representação compacta**
- **Desvantagens**
  - **tempo O(|V|) para determinar se existe uma aresta entre u e v**
    - **podem haver |V| elementos na lista de adjacências de u**

listas de adjacências: representação geralmente usada na maioria das aplicações

Grafos
Listas de Adjacências

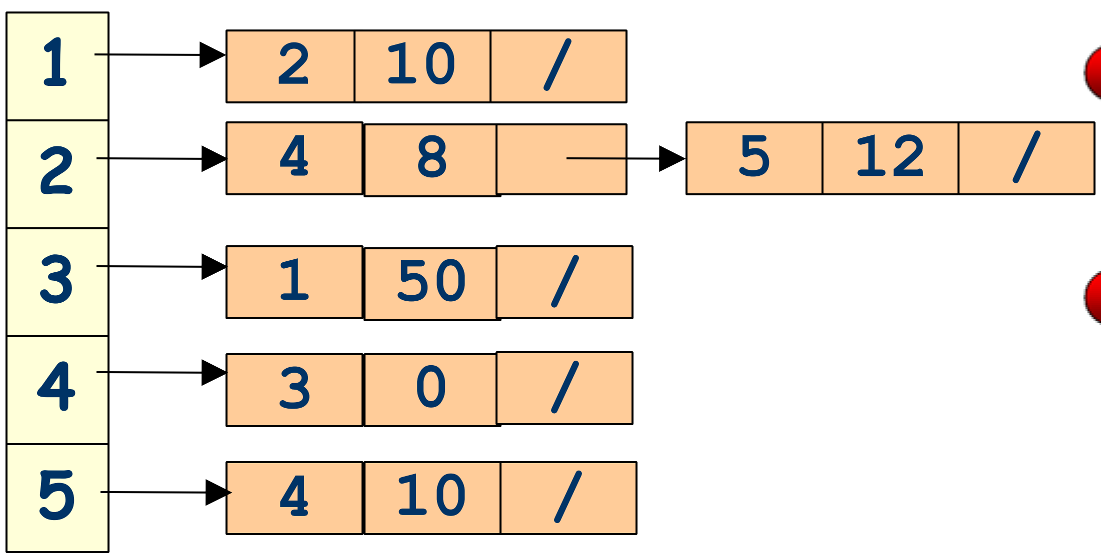
- **Observações**
  - **Os vértices adjacentes a um vértice i podem ser armazenados na lista de adjacências de i em ordem arbitrária ou não**
    - **usualmente armazenados de forma arbitrária**
  - **Como em qualquer estrutura de dados, há liberdade para haver variações na representação**
    - **vetor de vetores**
    - **vetor de listas ligadas**
    - **...**

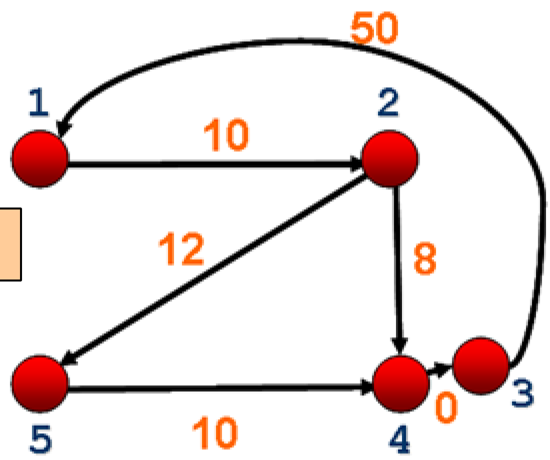
implementação muito comum:
vetor de ponteiros com listas encadeadas dinâmicas

Grafos

Listas de Adjacências

- **Exemplo: representação usando vetor de ponteiros com listas encadeadas dinâmicas**

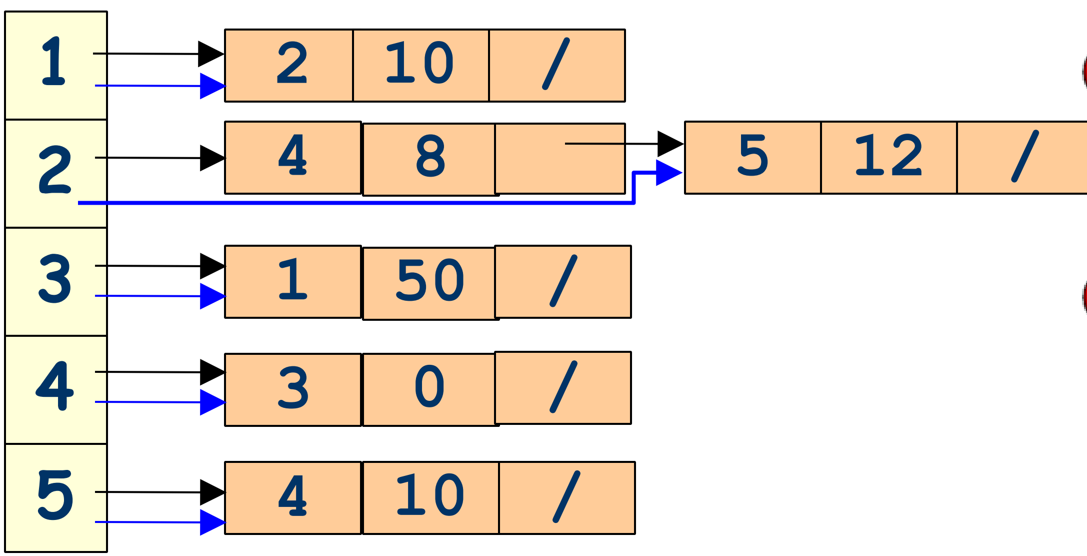


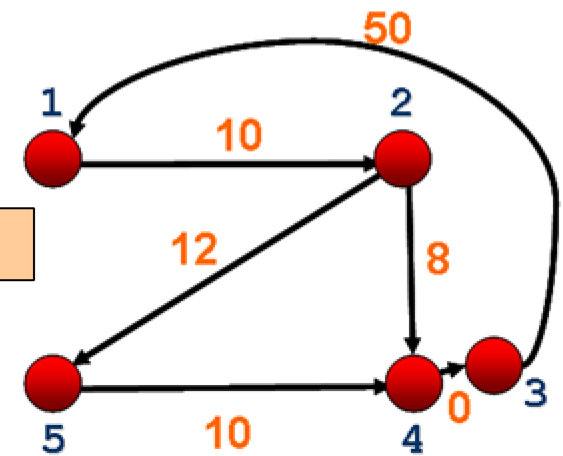


Grafos

Listas de Adjacências

- **Exemplo: representação usando vetor de ponteiros com listas encadeadas dinâmicas**





Pergunta: seria interessante armazenar um ponteiro para o último elemento de cada lista?

Grafos
Comparação

Comparação	Vencedor
Rapidez para saber se (x,y) está no grafo	Matriz de adjacências
Rapidez para determinar o grau de um vértice	Listas de adjacências
Grafos esparsos	Listas de adjacências

Grafos

Comparação

Comparação	Vencedor
Grafos densos	Matriz de adjacências
Inserção/remoção de arestas	Matriz: O(1) Listas: O(d)
Melhor na maioria dos problemas	Listas de adjacências
Rapidez para percorrer o grafo	Listas: $O(\|V\| + \|A\|)$ Matriz: $O(\|V\|^2)$

TAD

Operações

Grafos

Operações

- Exemplos de operações básicas sobre um grafo G
  - **Criar grafo**: cria *G* composto de um conjunto de vértices
  - **Inserir aresta**: insere uma aresta e seu peso em *G*
  - **Remover aresta**: remove uma aresta de *G* e retorna seu peso
  - **Verificar a existência de aresta**: retorna verdadeiro se a aresta existe e falso caso contrário
  - **Imprimir grafo**: imprimir os vértices e arestas de *G*
  - **Liberar grafo**: libera o espaço ocupado por um grafo
  - **Transpor grafo**: libera o espaço ocupado por um grafo

Grafos
Operações

- Exemplos de operações básicas sobre um grafo G
  - Gerenciar vértices adjacentes
    - Verificar a existência de um vértice adjacente ao vértice *v*
    - Verificar se a lista de vértices adjacentes está vazia
    - Retornar o primeiro vértice da lista
    - Retornar o próximo vértice adjacente da lista

Exemplos em C

TAD Grafos – Exemplos em C

Matriz de Adjacências

```c
#define MAX_VERTICES 100

// Estrutura do Grafo usando matriz de adjacência
typedef struct {
    int numVertices;
    int matriz[MAX_VERTICES][MAX_VERTICES];
} GrafoMatriz;

// Operações
void inicializarGrafoMatriz(GrafoMatriz* grafo, int numVertices);
void inserirArestaMatriz(GrafoMatriz* grafo, int origem, int destino);
void removerArestaMatriz(GrafoMatriz* grafo, int origem, int destino);
void imprimirGrafoMatriz(GrafoMatriz* grafo);
int verificarConexaoMatriz(GrafoMatriz* grafo, int origem, int destino);
```

TAD Grafos – Exemplos em C

Lista de Adjacências

```c
// Estrutura para um nó na lista de adjacência
typedef struct No {
    int vertice;
    struct No* prox;
} No;

// Estrutura para a lista de adjacência de cada vértice
typedef struct {
    int numVertices;
    No** listaAdjacencia;  // Ponteiro para uma array de listas
} GrafoLista;

// Protótipos das operações
GrafoLista* criarGrafoLista(int numVertices);
void destruirGrafoLista(GrafoLista* grafo);
void inserirArestaLista(GrafoLista* grafo, int origem, int destino);
void removerArestaLista(GrafoLista* grafo, int origem, int destino);
```

Referências

CORMEN, T.H.; LEISERSON, C.E.; RIVEST, R.L.; STEIN, C. **Algoritmos: Teoria e Prática**. Campus. 2002.

ZIVIANI, N.; **Projeto de Algoritmos com Implementações em Pascal e C**, 2 edição, Pioneira Thonsom Learning, 2004.