Algoritmos em Grafos

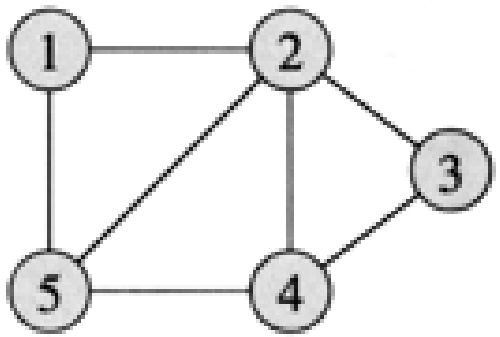
Rodrigo Caetano de Oliveira Rocha

Pontifícia Universidade Católica de Minas Gerais

2015

Matriz de adjacências - Grafos não direcionados

- A matriz de adjacências de um grafo com n vértices é uma matriz $n \times n$, definida como:
  - $M_{u,v} = 1$ se existe uma aresta entre os vértices u e v.
  - $M_{u,v} = 0$ caso contrário.



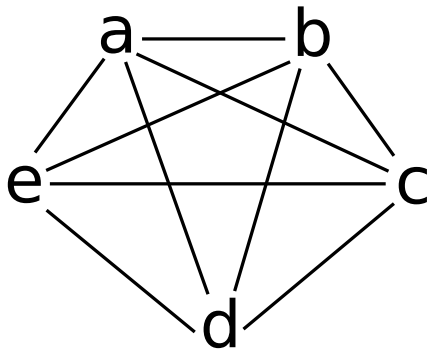
	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

- **Vantagem:** verificar adjacência entre vértices é simples.
- **Desvantagem:** espaço de armazenamento.

Matriz de adjacências - Grafos não direcionados

- Em um grafo completo a matriz é preenchida por 1's em todos os elementos, exceto na diagonal principal.

Exemplo do K_5:

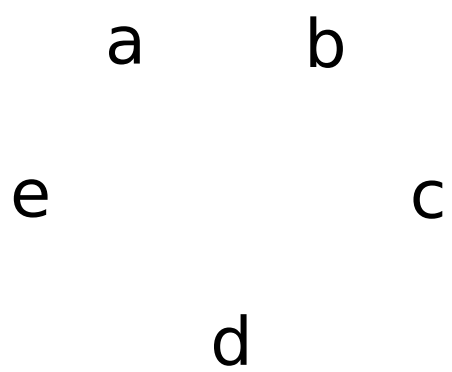


	a	b	c	d	e
a	0	1	1	1	1
b	1	0	1	1	1
c	1	1	0	1	1
d	1	1	1	0	1
e	1	1	1	1	0

Matriz de adjacências - Grafos não direcionados

- Um grafo nulo possui todos os elementos da matriz de adjacências preenchidos por 0's.

Grafo nulo de 5 vértices:

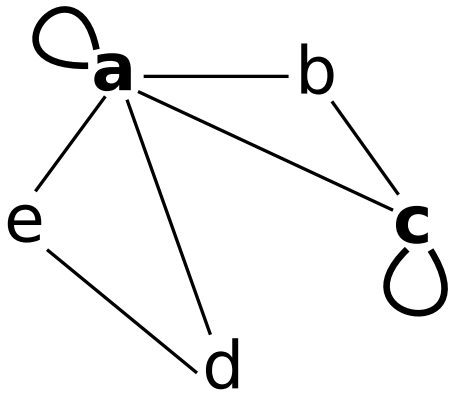


	a	b	c	d	e
a	0	0	0	0	0
b	0	0	0	0	0
c	0	0	0	0	0
d	0	0	0	0	0
e	0	0	0	0	0

Matriz de adjacências - Grafos não direcionados

- Se existe um loop no vértice u, o elemento da diagonal principal na linha e coluna u é preenchido com 1.

Exemplo de um grafo de 5 vértices com um loop em a e um loop em c:

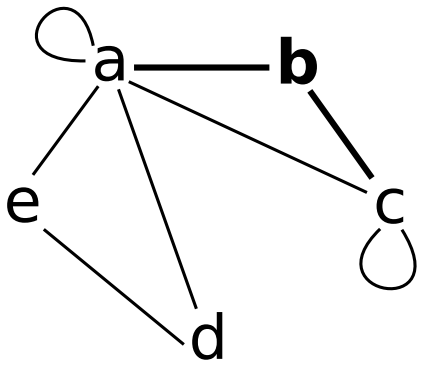


	a	b	c	d	e
a	**1**	1	1	1	1
b	1	0	1	0	0
c	1	1	**1**	0	0
d	1	0	0	0	1
e	1	0	0	1	0

Matriz de adjacências - Grafos não direcionados

- O grau de um vértice pode ser equivalentemente determinado pela coluna ou linha do vértice.

Obtendo o grau do vértice *b* pela coluna ou linha do vértice:



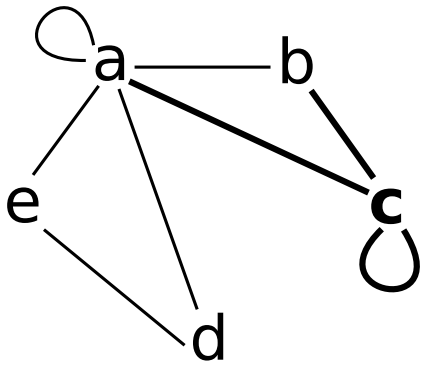
	a	b	c	d	e	
a	1	1	1	1	1	
b	1	0	1	0	0	$d(b) = 2$
c	1	1	1	0	0	
d	1	0	0	0	1	
e	1	0	0	1	0	

$d(b) = 2$

Matriz de adjacências - Grafos não direcionados

- Caso haja um loop, ao calcular o grau do vértice com loop, a entrada da diagonal principal deve ser contada duas vezes.

Obtendo o grau do vértice c, considerando o loop existente:



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

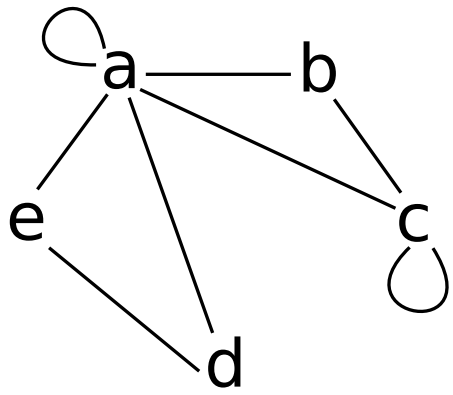
$d(c) = 4$

$d(c) = 4$

Matriz de adjacências - Grafos não direcionados

- O número de arestas de um grafo pode ser calculado utilizando os elementos do triangulo superior da matriz de adjacências, incluindo os elementos da diagonal principal, pois cada aresta é representada por duas entradas simétricas da matriz. De maneira equivalente, o triangulo inferior pode ser utilizado.

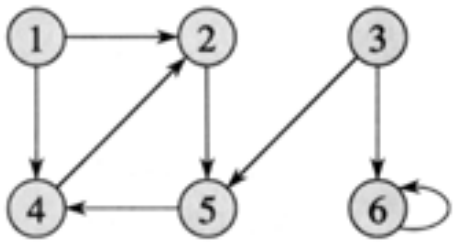
Contagem de arestas em um grafo de 5 vértices:



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

Matriz de adjacências - Grafos direcionados

- A matriz de adjacências de um grafo direcionado com n vértices é uma matriz $n \times n$, definida como:
  - $M_{u,v} = 1$ se existe uma aresta direcionada com **origem** no vértice u e **destino** em v.
  - $M_{u,v} = 0$ caso contrário.



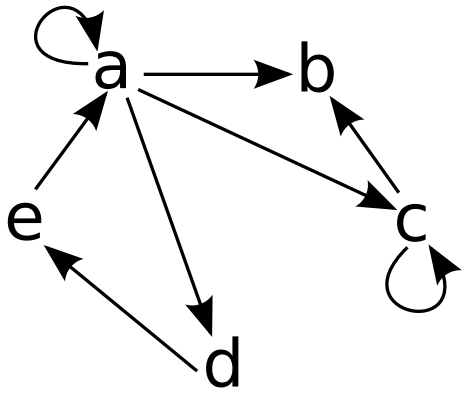
	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

- **Vantagem:** verificar adjacência entre vértices é simples.
- **Desvantagem:** espaço de armazenamento.

Matriz de adjacências - Grafos direcionados

- O número de arestas de um grafo direcionado é calculado somando todos os elementos da matriz de adjacências, pois cada aresta é representada por apenas uma entrada da matriz.

Calculando o número de arestas:

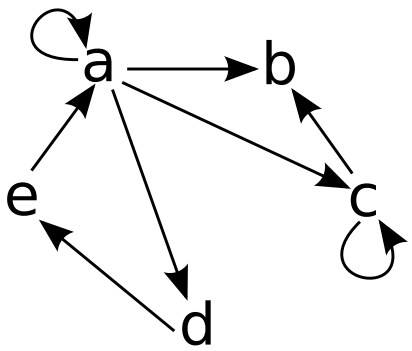


	a	b	c	d	e
a	1	1	1	1	0
b	0	0	0	0	0
c	0	1	1	0	0
d	0	0	0	0	1
e	1	0	0	0	0

Matriz de adjacências - Grafos direcionados

- O grau de saída de um vértice u é obtido somando os elementos da linha u.
- Em grafos direcionados, loops são computados apenas uma vez para o grau de saída e uma para o grau de entrada.

Calculando o grau de saída dos vértices a e c:

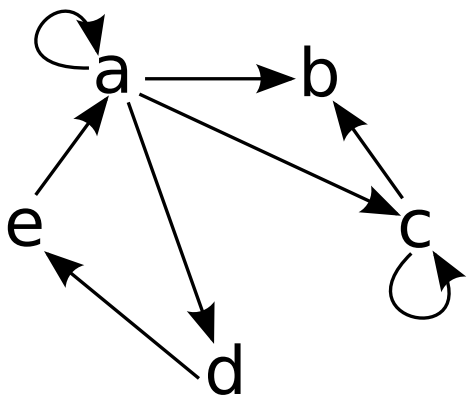


	a	b	c	d	e	
a	1	1	1	1	0	$d^+(a) = 4$
b	0	0	0	0	0	
c	0	1	1	0	0	$d^+(c) = 2$
d	0	0	0	0	1	
e	1	0	0	0	0	

Matriz de adjacências - Grafos direcionados

- O grau de entrada de um vértice u é obtido somando os elementos da coluna u.
- Em grafos direcionados, loops são computados apenas uma vez para o grau de saída e uma para o grau de entrada.

Calculando o grau de entrada do vértice c:



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

$$d^-(c) = 2$$

Matriz de adjacências - Grafos direcionados

```
public class MatrizDigrafo extends Digrafo {
    private int n;
    private int [][]adj;

    public MatrizDigrafo(int n){
        this.n = n;
        this.adj = new int[n][n];
    }
    public void adicionaAresta(int u, int v){
        this.adj[u][v] = 1;
    }
    public void removeAresta(int u, int v){
        this.adj[u][v] = 0;
    }
}
```

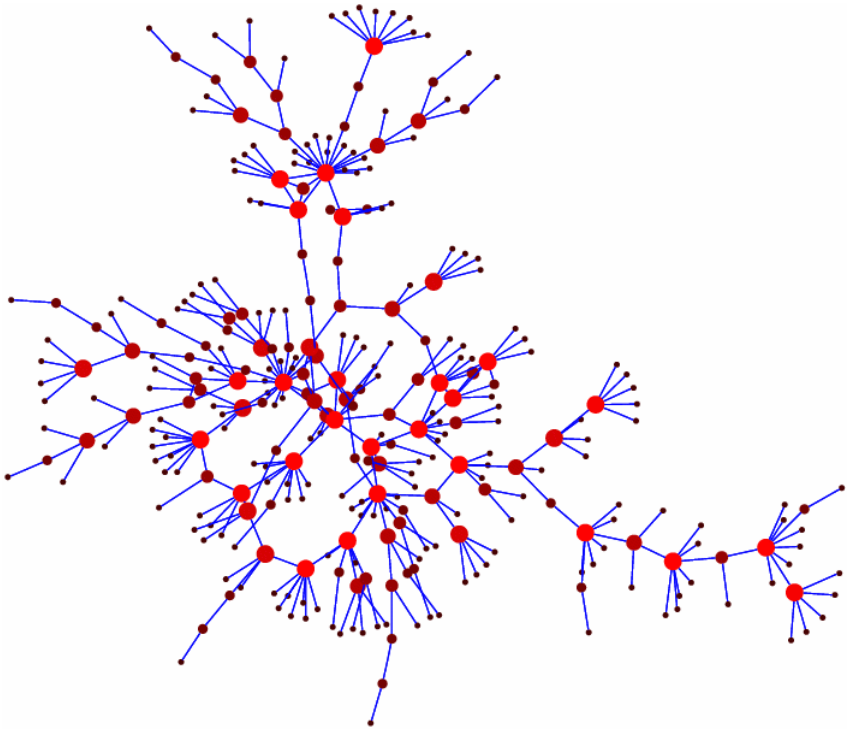
Matriz de adjacências - Grafos direcionados

```
public class MatrizDigrafo extends Digrafo {
    //...
    public boolean existeAresta(int u, int v){
        return (this.adj[u][v] != 0);
    }

    public int numArestas(){
        int arestas = 0;
        for(int u = 0; u<numVertices(); u++){
            for(int v = 0; v<numVertices(); v++){
                if(existeAresta(u,v)){
                    arestas++;
                }
            }
        }
        return arestas;
    }
}
```

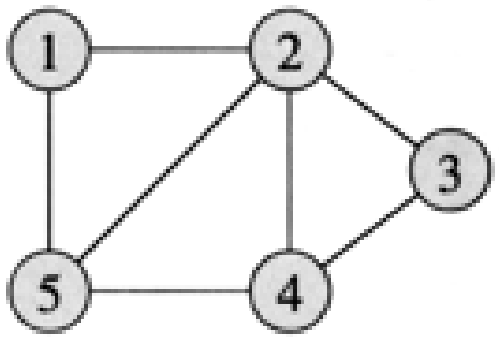
Matriz de adjacências

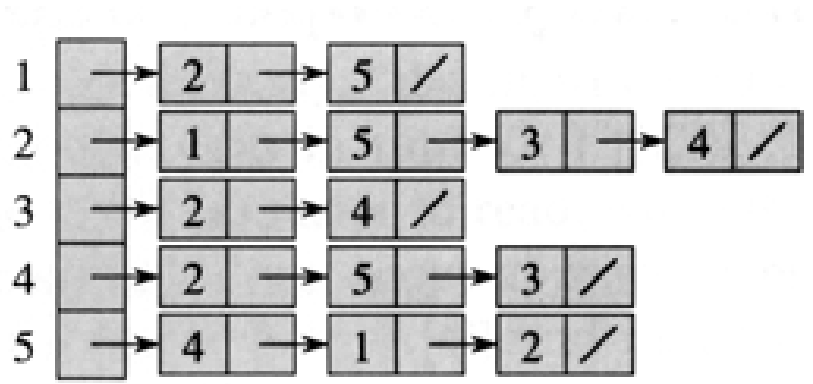
- Para grafos com grandes estruturas, a representação por matriz de adjacências possui uma limitação de espaço de armazenamento, que pode ser agravada em grafos com poucas conexões (grafos esparços), devido ao espaço desnecessariamente utilizado.



Lista de adjacências - Grafos não direcionados

- A lista de adjacências de um grafo com n vértices é representada por n listas, sendo uma para cada vértice.
- Cada lista contém apenas os vértices adjacentes, armazenando apenas os elementos diferentes de zero da matriz de adjacências.

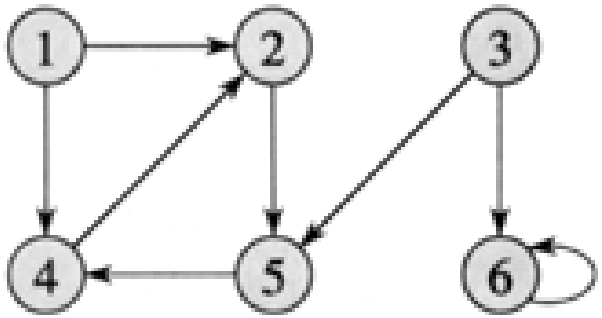


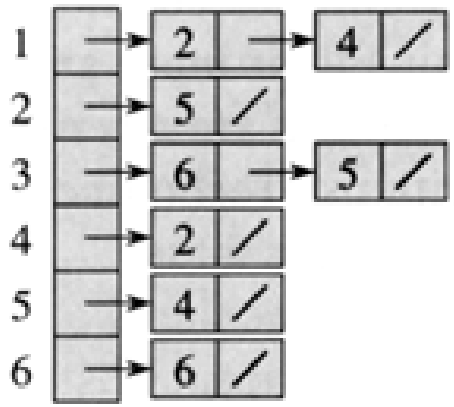


- **Vantagem:** espaço de armazenamento proporcional ao número de arestas.
- **Desvantagem:** para verificar adjacência é necessário percorrer uma lista.

Lista de adjacências - Grafos direcionados

- A lista de adjacências de um grafo com n vértices é representada por n listas, sendo uma para cada vértice.
- Cada lista contém apenas os vértices adjacentes, armazenando apenas os elementos diferentes de zero da matriz de adjacências.



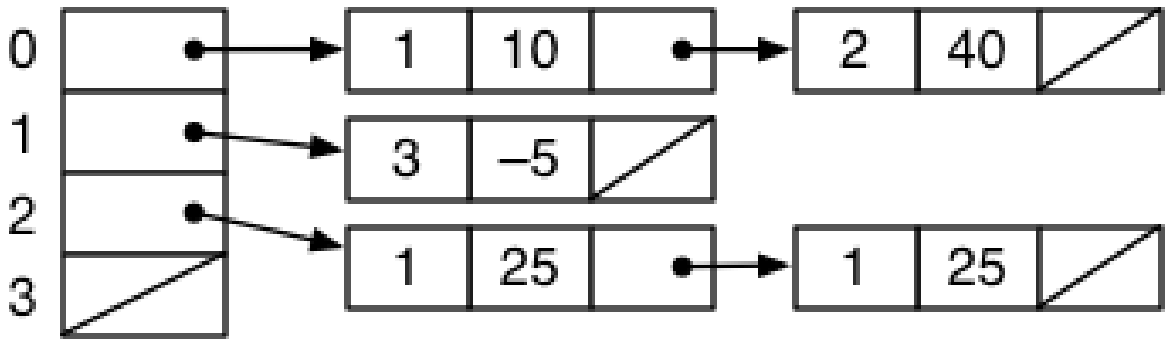


- **Vantagem:** espaço de armazenamento proporcional ao número de arestas.
- **Desvantagem:** para verificar adjacência é necessário percorrer uma lista.

Lista de adjacências

Outras vantagens:

- Melhor representação para arestas paralelas (ou redundantes).
- Melhor representação para arestas com pesos.



Lista de adjacências - Grafos direcionados

```
public class ListaDigrafo extends Digrafo {
    private int n;
    private List<Integer> []adj;

    public ListaDigrafo(int n){
        this.n = n;
        this.adj = (List<Integer>[])new List[n];
        for(int u = 0; u<this.n; u++){
            this.adj[u] = new ArrayList<Integer>();
        }
    }
    public void adicionaAresta(int u, int v){
        this.adj[u].add(new Integer(v));
    }
    public void removeAresta(int u, int v){
        this.adj[u].remove(new Integer(v));
    }
}
```

Lista de adjacências - Grafos direcionados

```
public class ListaDigrafo extends Digrafo {
    //...
    public boolean existeAresta(int u, int v){
        if( this.adj[u].contains(new Integer(v)) ){
            return true;
        }else{
            return false;
        }
    }

    public int numArestas(){
        int arestas = 0;
        for(int u = 0; u<numVertices(); u++){
            arestas += adj[u].size();
        }
        return arestas;
    }
}
```