Grafos: DFS e BFS

Disciplina: Laboratório de Programação II

Professor Luciano Brum
Email: lucianobrum18@gmail.com

Assunto da aula de hoje:

Grafos: BFS e DFS

Tópicos

- Revisão de grafos e respectiva estrutura.
- Algoritmo de BFS.
- Algoritmo de DFS.
- Resumo.
- Exercícios.

Grafos: Revisão

- Muitas aplicações necessitam considerar conjunto de conexões entre pares de objetos:
  - Existe um caminho de um objeto à outro?
  - Qual a menor distância entre esses objetos?
  - Quantos objetos podem ser alcançados de determinado objeto?

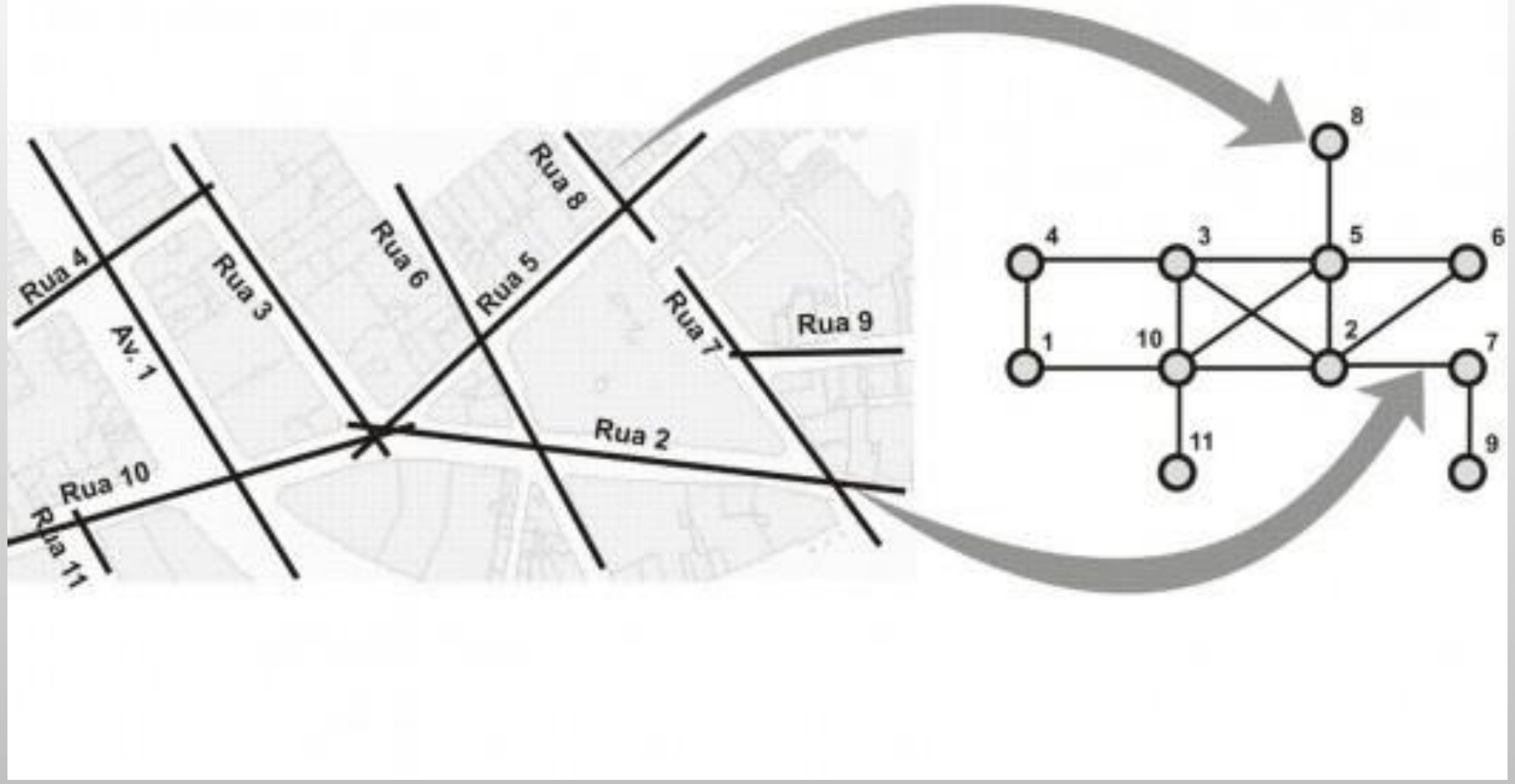
Grafos: Revisão

- Grafos são estruturas de dados muito importantes e recorrentes.

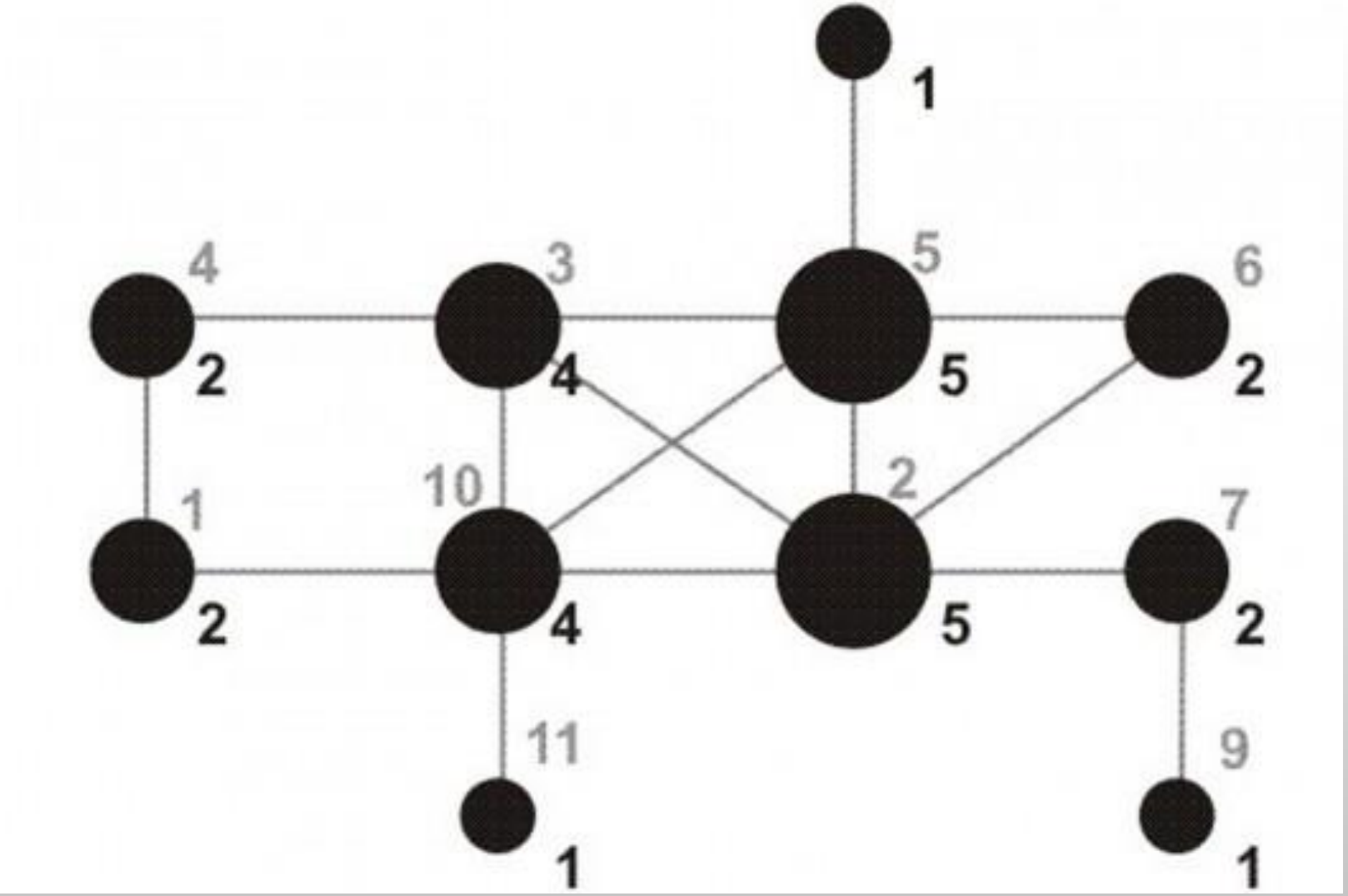
  Representam arbitrárias relações entre elementos.

- Grafos podem ser utilizados para modelar:
- Malhas viárias;
- Dutos (oleodutos, gasodutos, etc);
- Circuitos eletrônicos;
- Redes (elétricas, de computadores, etc);
- Programas;

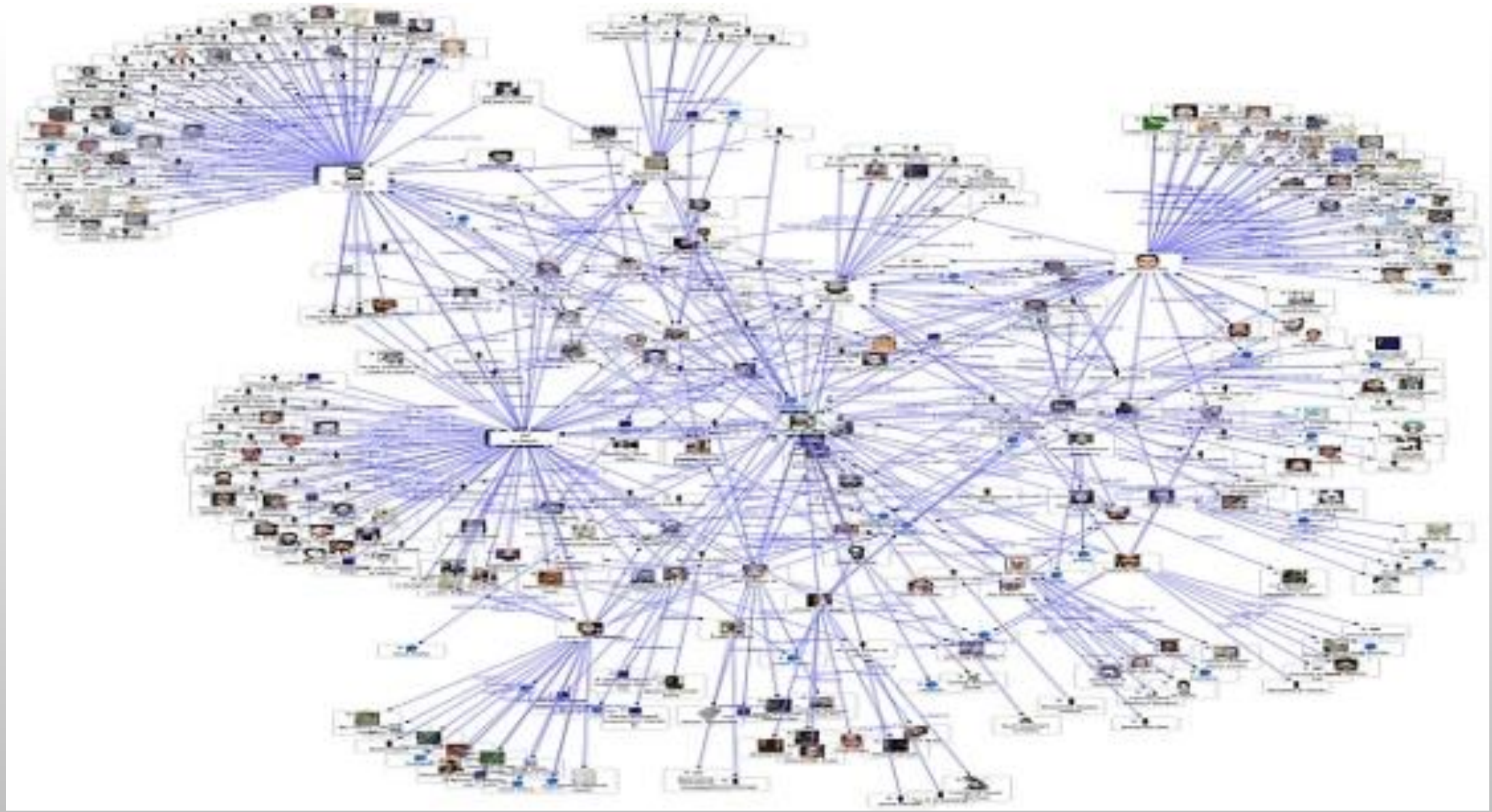
Grafos: Revisão



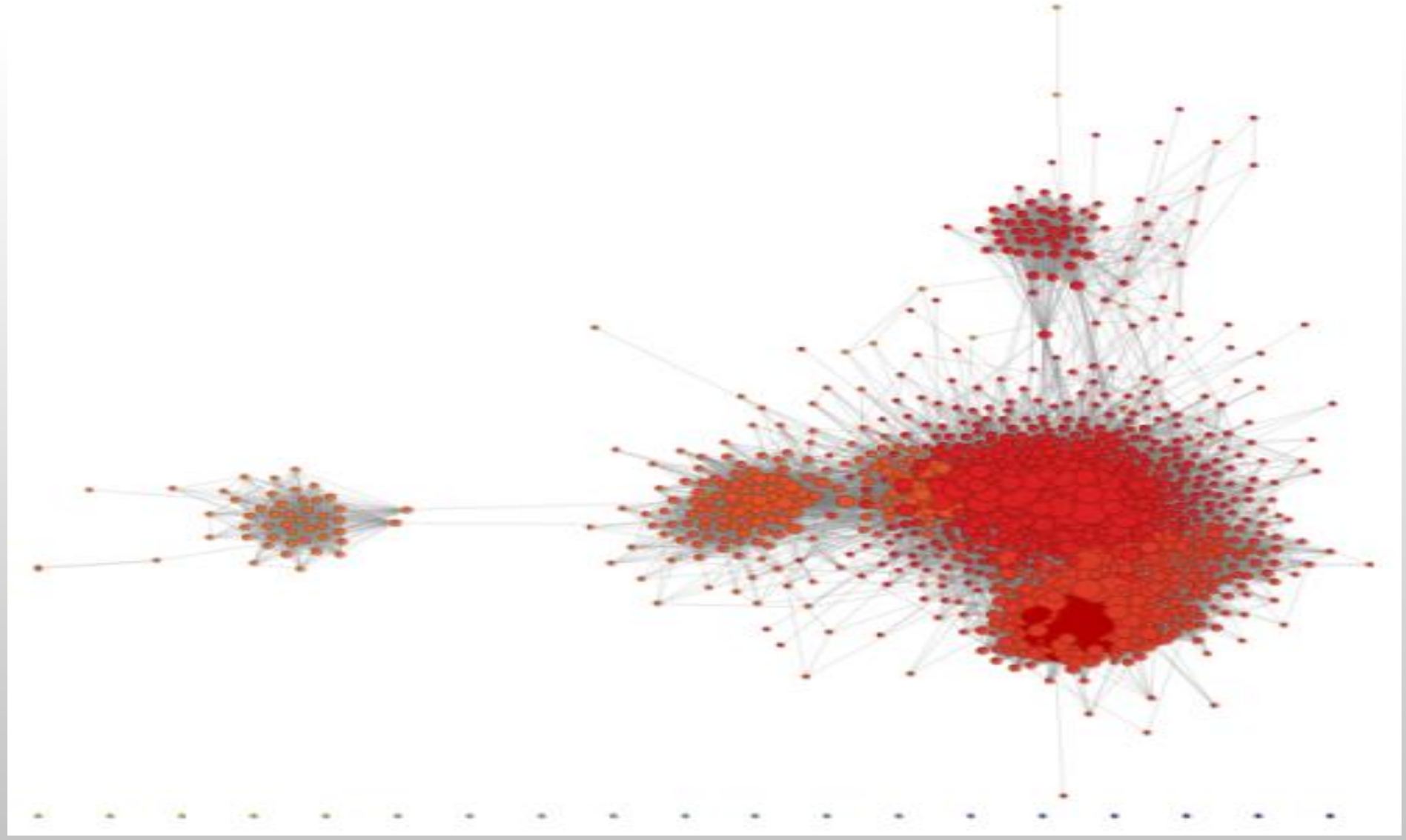
Grafos: Revisão



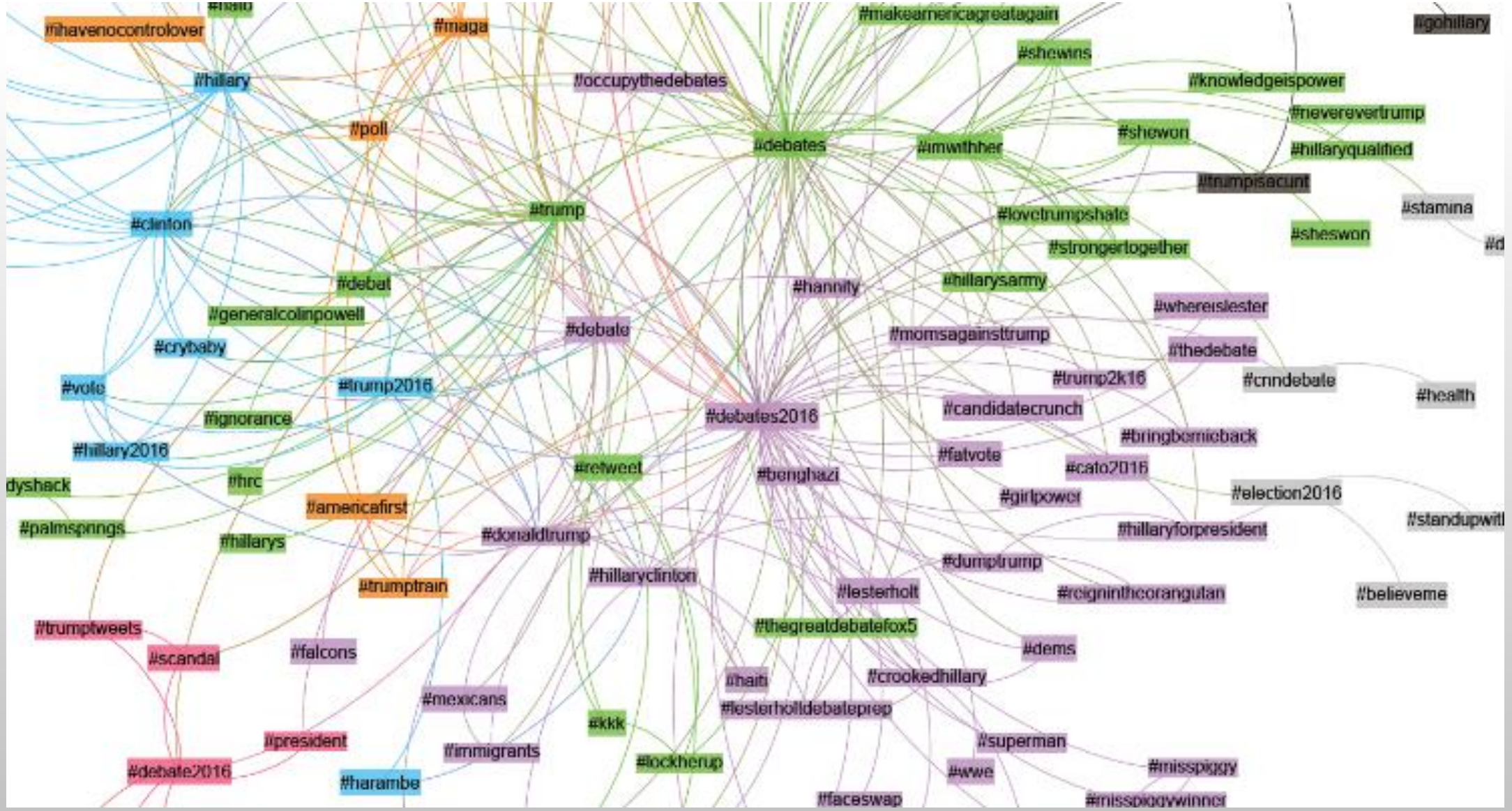
Grafos: Revisão



Grafos: Revisão

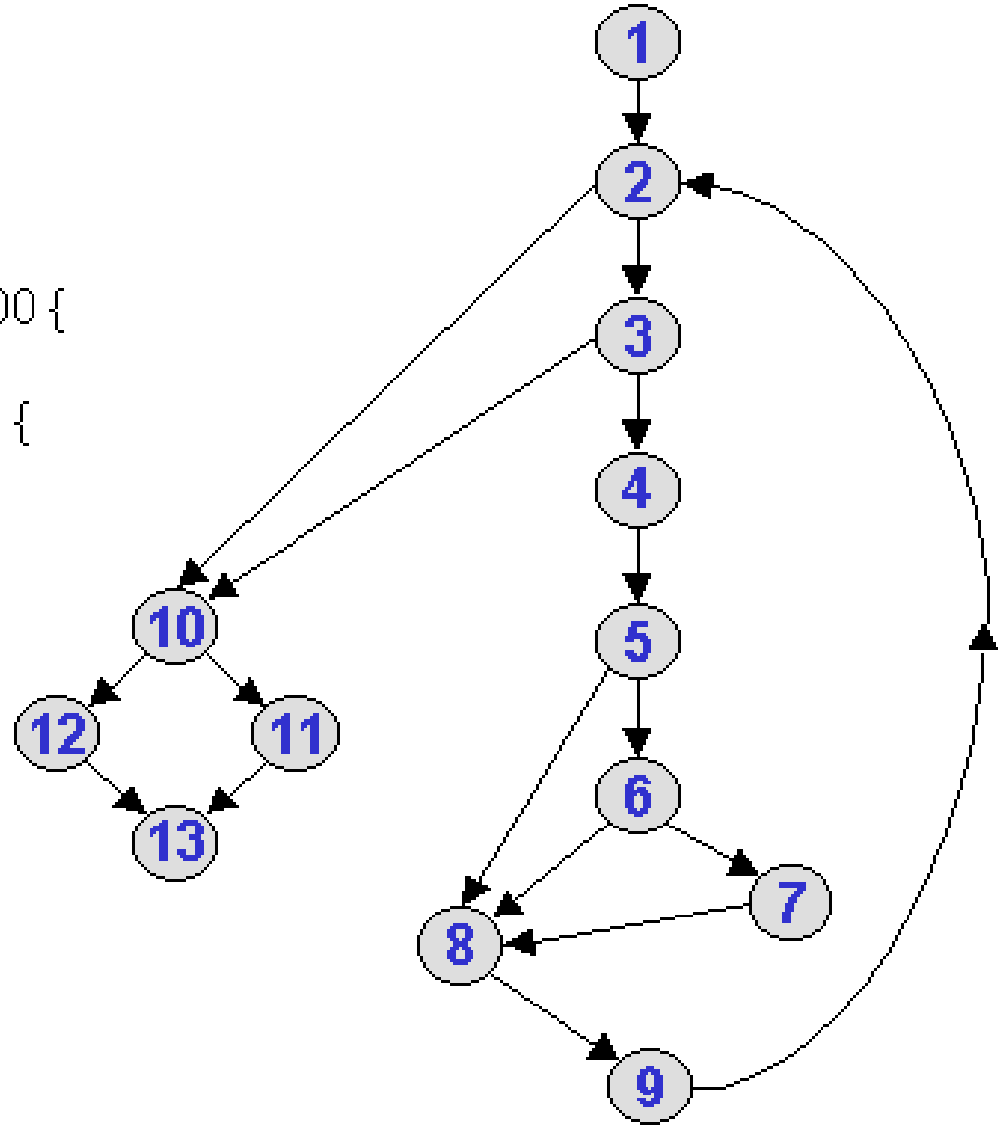


Grafos: Revisão



Grafos: Revisão

```
1   i = 1;
    total_entrada = total_valido = 0;
    soma= 0;                                    3
                                2
    while valor[i] != -999 && total_entrada < 100 {
4       total_entrada++;                5                   6
        if valor[i] ? minimo && valor[i] ? maximo {
                        7       total.valido++;
                                soma = soma + valor[i];
        }
8       i++;
9   }
10  if total_valido > 0
                        11      media = soma/total_valido;
    else
                        12      media = -999;
13  }
```



Grafos: Revisão

- Grafos podem ser descritos como um par G = (V, E), sendo V não-vazio, representando os vértices ou nós e sendo E o conjunto de arestas do grafo G.

- As arestas podem ter valores ou pesos associados a elas.

- Os grafos podem ser não-direcionados ou direcionados (dígrafos).

Grafos: Revisão

- Os grafos ainda podem ser: ponderados ou não-ponderados.

- O menor caminho em um grafo ponderado é a menor distância de um nó origem até um nó destino.

- O menor caminho em um grafo não-ponderado é o caminho com menor número de arestas entre um nó origem e destino.

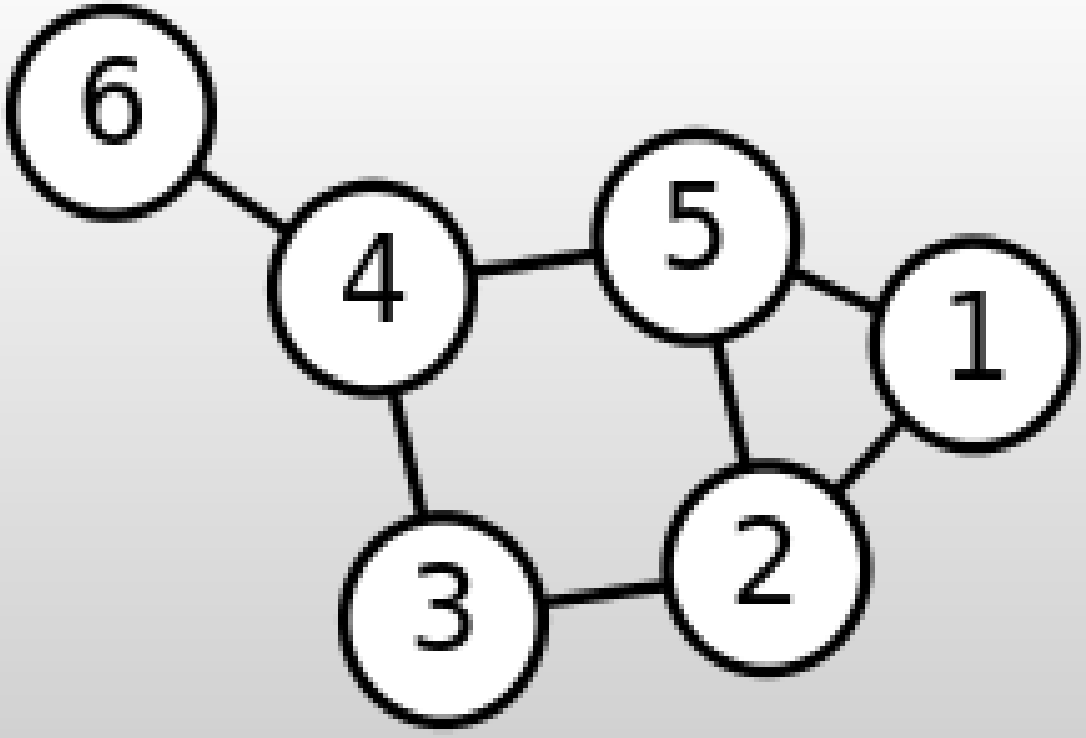
Grafos: Revisão

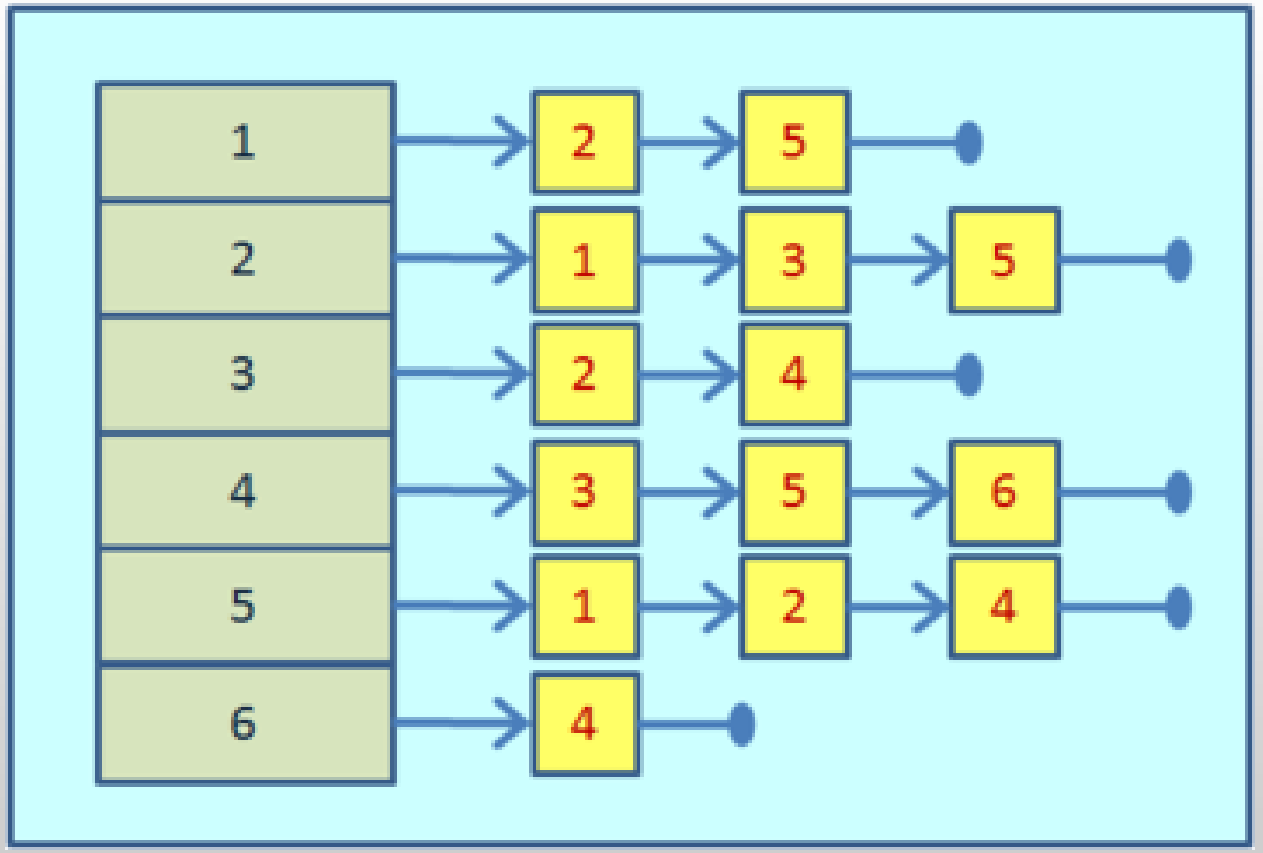
- Grafos que têm |E| muito menor que $|V|^2$ são ditos **esparsos**.
- Grafos que têm |E| muito próximo de $|V|^2$ são ditos **densos**.
- Diferentes estruturas de dados podem ser utilizadas para representar um grafo, de acordo com a sua densidade.

Grafos: Revisão

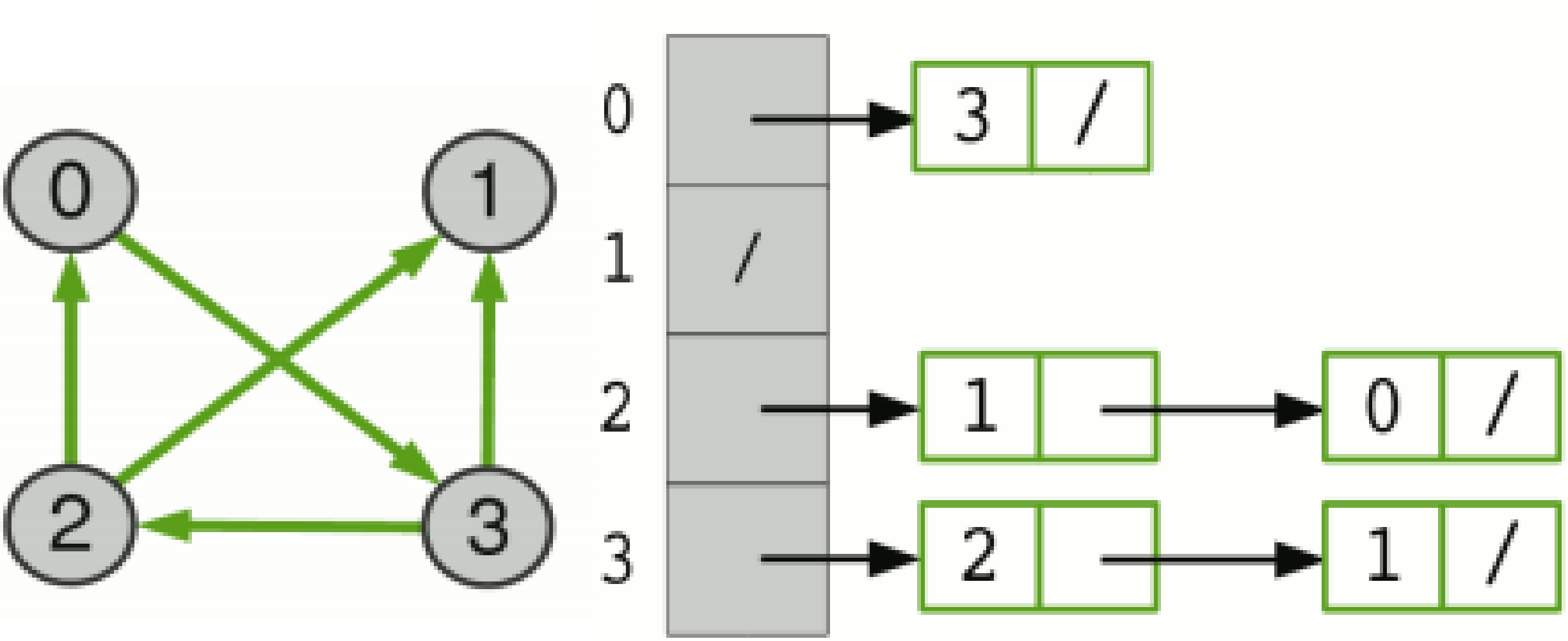
- Temos 2 formas de representação de grafos:
  - Listas de adjacências;
  - Matriz de adjacências;

Grafos: Listas de Adjacência.

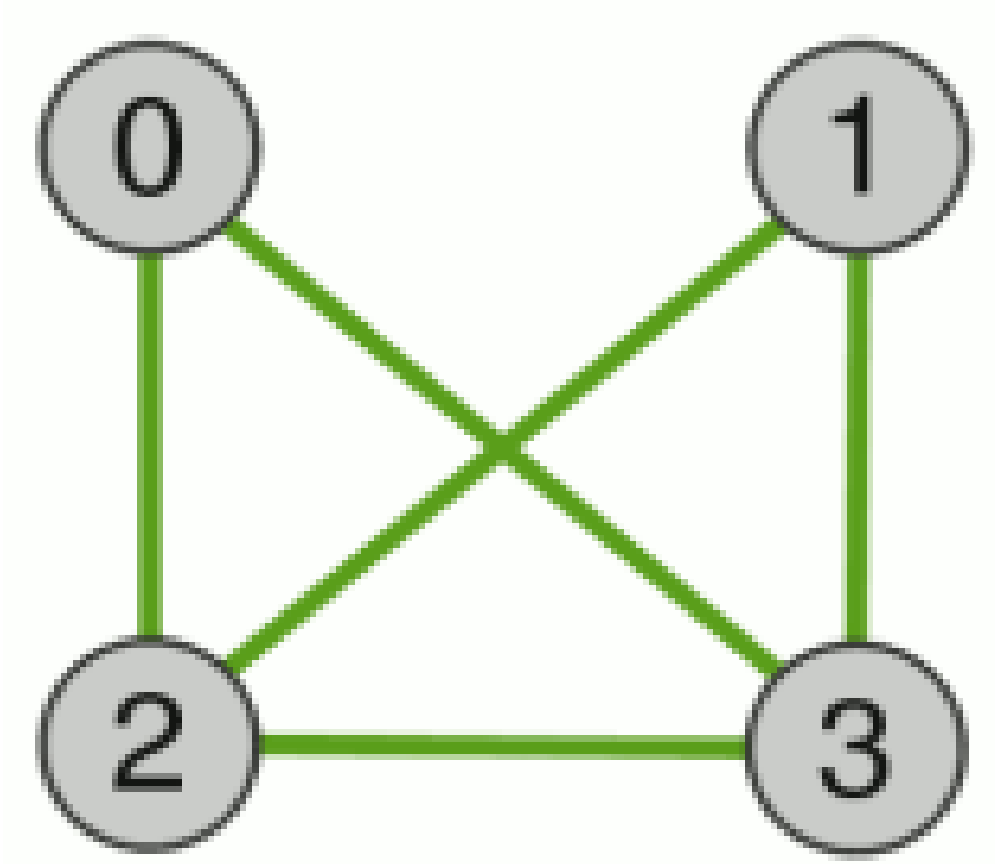




Grafos: Listas de Adjacência.

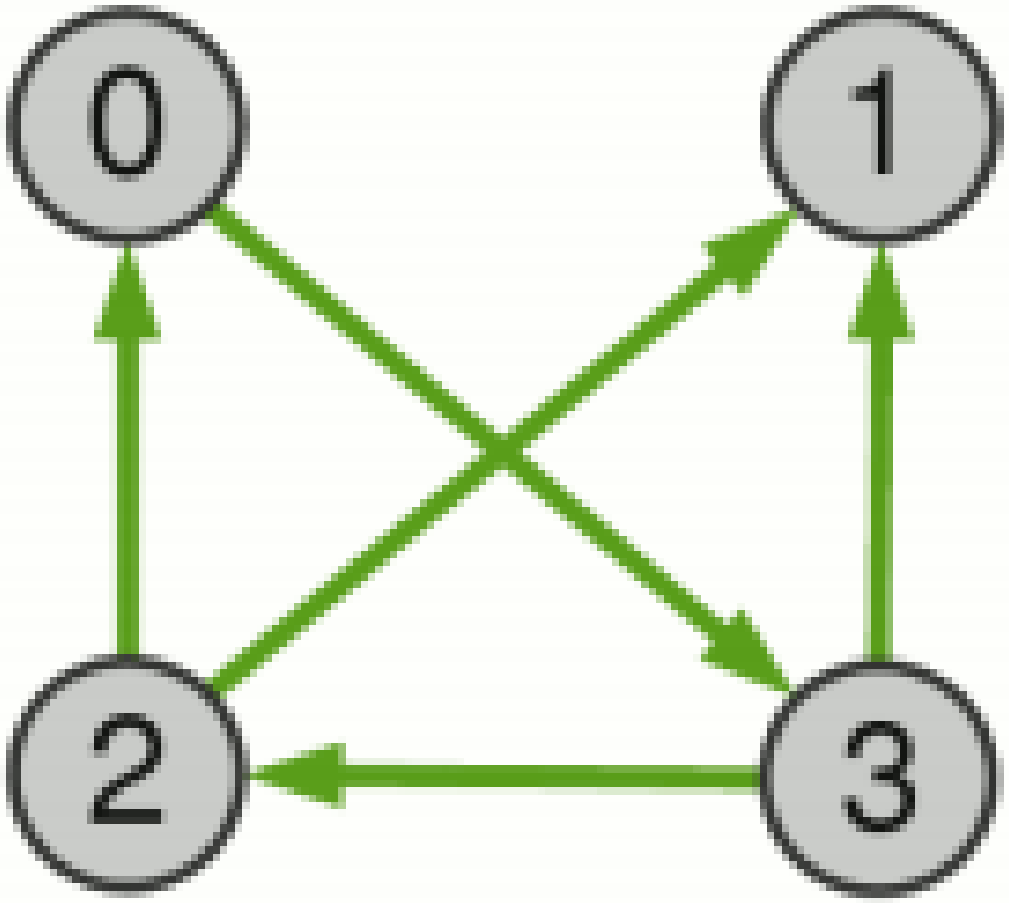


Grafos: Matriz de Adjacência.



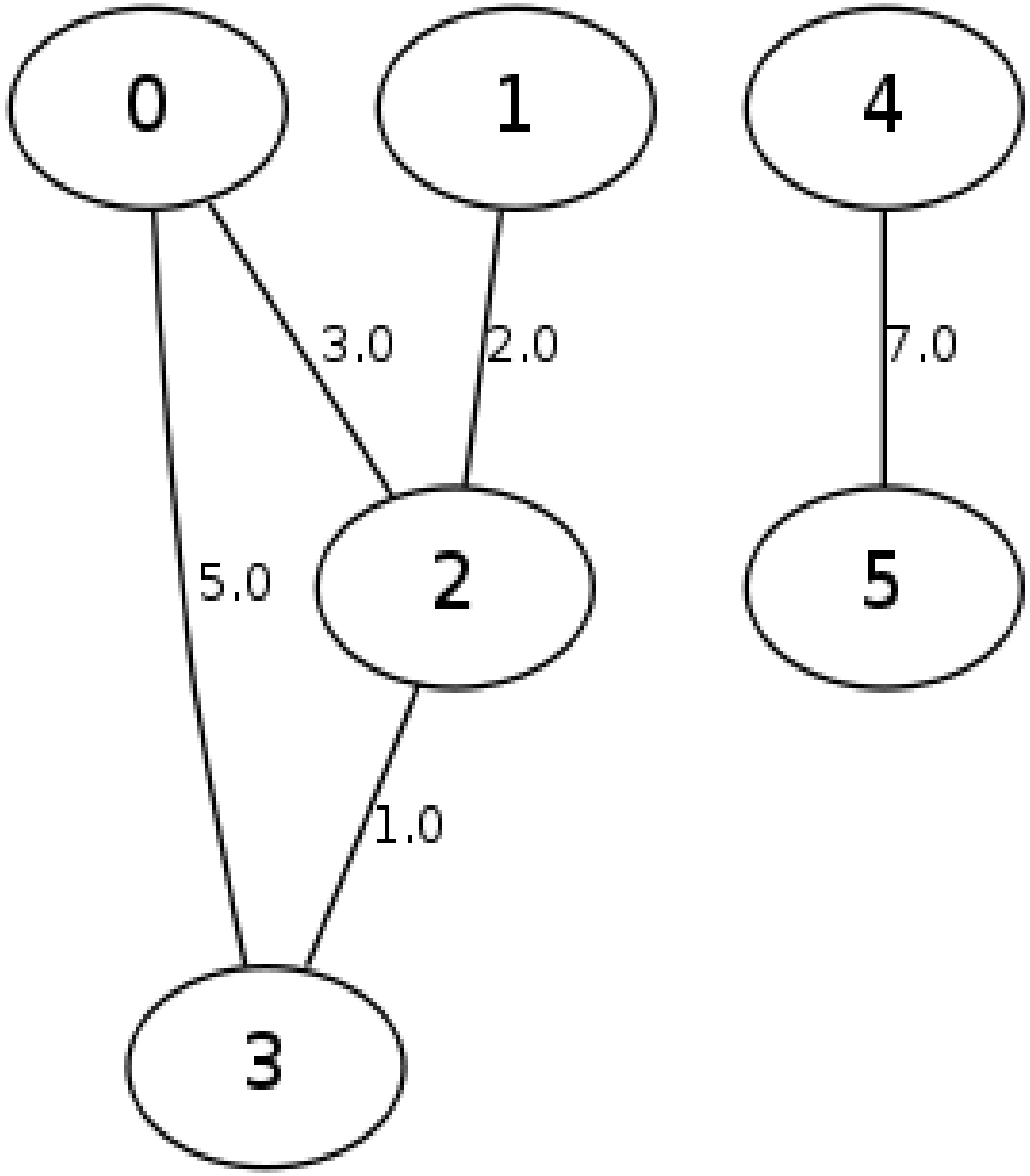
	0	1	2	3
0	0	0	1	1
1	0	0	1	1
2	1	1	0	1
3	1	1	1	0

Grafos: Matriz de Adjacência.



	0	1	2	3
0	0	0	0	**1**
1	0	0	0	0
2	**1**	**1**	0	0
3	0	**1**	**1**	0

Grafos: Matriz de Adjacência.



	0	1	2	3	4	5
0	∞	∞	3	5	∞	∞
1	∞	∞	2	∞	∞	∞
2	3	2	∞	1	∞	∞
3	5	∞	1	∞	∞	∞
4	∞	∞	∞	∞	∞	7
5	∞	∞	∞	∞	7	∞

Tópicos

- ~~Revisão de grafos e respectiva estrutura.~~
- Algoritmo de BFS.
- Algoritmo de DFS.
- Resumo.
- Exercícios.

Grafos: BFS

- Um dos algoritmos de busca utilizado em grafos é o BFS (Breadth-First Search ou Busca em Largura).

- Começamos pesquisando no vértice origem ‘o’ e exploramos todos seus vértices vizinhos. Para cada um dos próximos vértices vizinhos, exploramos os seus vizinhos, e assim, sucessivamente, até não encontrar mais vértices no grafo.

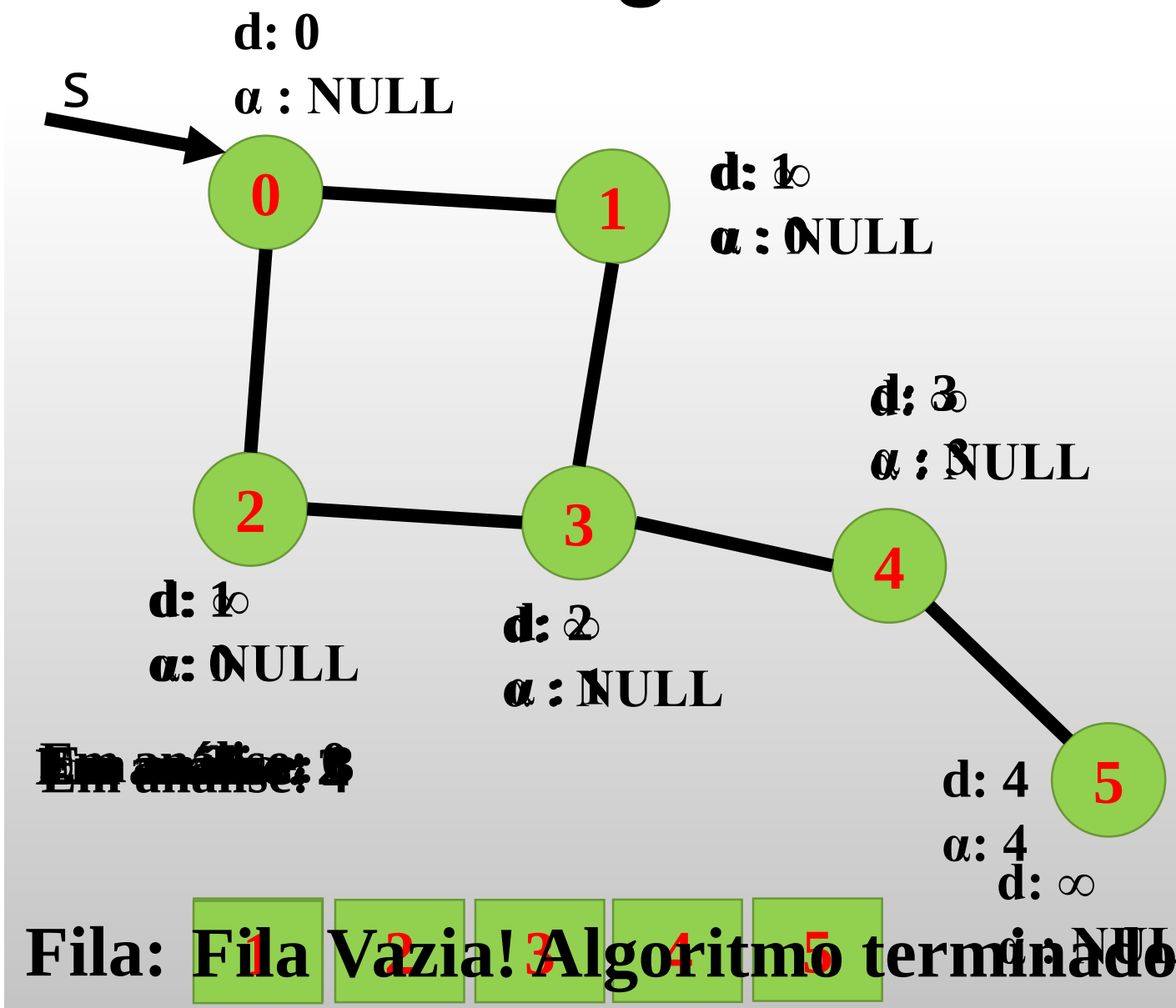
Grafos: BFS

- Cada vértice adjacente que é descoberto pela primeira vez é marcado e é inserido em uma fila.

- Quando todos os adjacentes do vértice analisado forem descobertos, marcamos este vértice e retiramos o próximo da fila e repetimos o procedimento até não encontrar mais vértices.

Grafos: Busca em Largura

```
BFS(G, s)
 1  for cada vértice u ∈ V[G] – {s}
 2        do cor[u] ← BRANCO
 3           d[u] ← ∞
 4           α[u] ← NIL
 5  cor[s] ← CINZA
 6  d[s] ← 0
 7  α[s] ← NIL
 8  Q ← 0
 9  ENQUEUE(Q, s)
10   while Q ≠ 0
11      do u ← DEQUEUE(Q)
12         for cada v ← Adj[u]
13            do if cor[v] = BRANCO
14                  then cor[v] ← CINZA
15                       d[v] ← d[u] + 1
16                       π[v] ← u
17                       ENQUEUE(Q, v)
18         cor[u] ← PRETO
```



Grafos: BFS

- Objetivo: obter-se o menor caminho (menor número de arestas) entre o vértice inicial da busca (origem) e todos os outros vértices em grafos não ponderados.

- Complexidade: $O\ (V + E)$.

Tópicos

- ~~Revisão de grafos e respectiva estrutura.~~
- ~~Algoritmo de BFS.~~
- Algoritmo de DFS.
- Resumo.
- Exercícios.

Grafos: DFS

- Um dos algoritmos de busca utilizado em grafos é o DFS (Depth-First Search ou Busca em Profundidade).

- A estratégia do algoritmo é buscar “o mais fundo possível” no grafo.

- As arestas são exploradas a partir do vértice v mais recentemente descoberto que ainda possui arestas não exploradas saindo dele.

Grafos: DFS

- Para acompanhar o progresso do algoritmo, cada vértice é colorido de branco, cinza ou preto.

- Todos os vértices são iniciados em branco.

- Quando um vértice é descoberto pela primeira vez, ele torna-se cinza e é tornado preto quando sua lista de adjacentes tiver sido completamente examinada.

Grafos: DFS

- **antecessor[v]**: antecessor de v.
- **d[v]**: tempo de descoberta de v.
- **f[v]**: tempo de término do exame da lista de adjacentes de v.

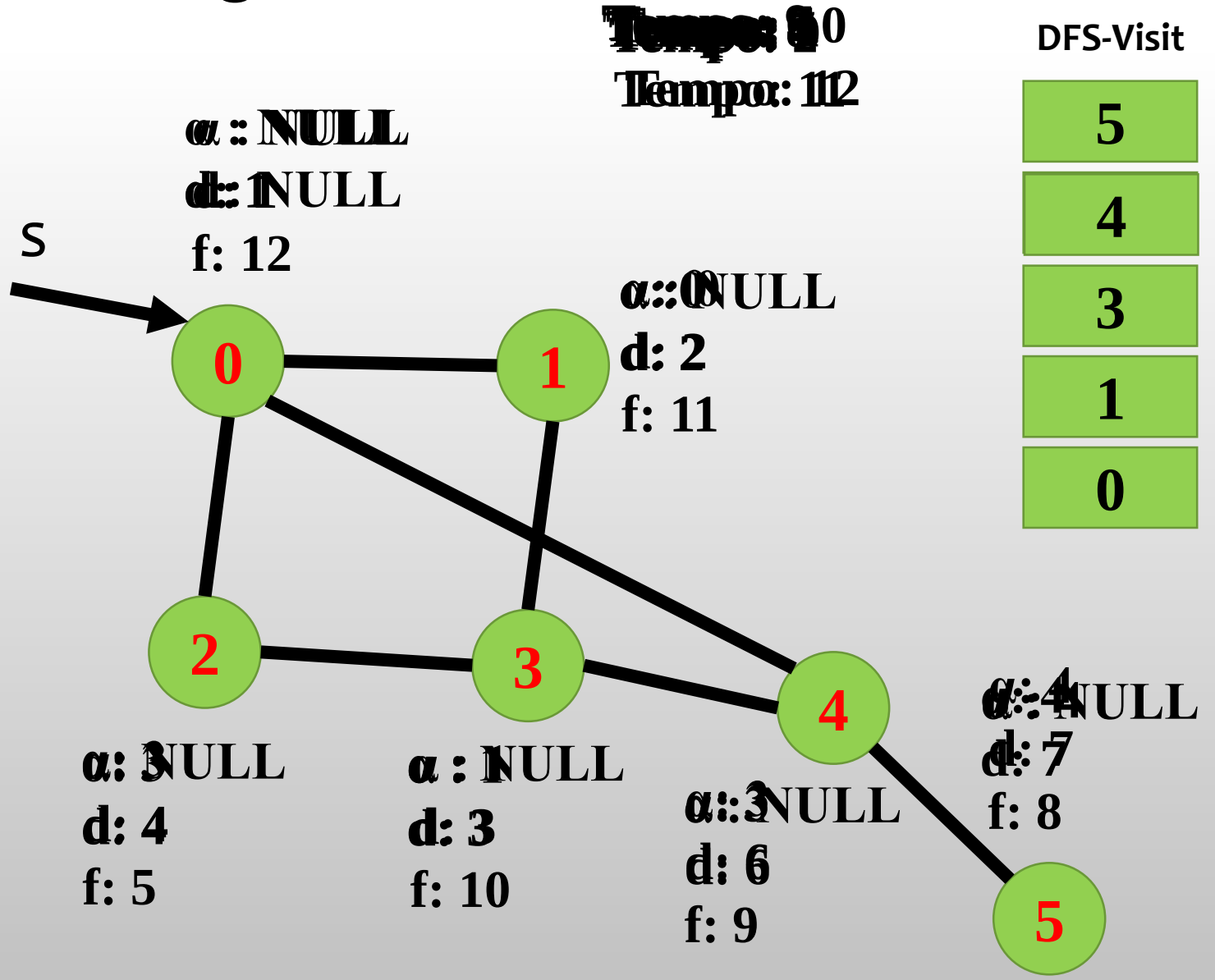
- Estes registros são inteiros entre 1 e 2|V|, pois existem um evento de descoberta e um evento de término para cada um dos |V| vértices.

- Para todo vértice v, d[v] < f[v].

```
DFS(G)
1  for cada vértice u ← V[G]
2      do cor[u] ← BRANCO
3         π[u] ← NIL
4  tempo ← 0
5  for cada vértice u ∈ V[G]
6      do if cor[u] = BRANCO
7           then DFS-VISIT(u)

DFS-VISIT(u)
1  cor[u] ← CINZA
2  tempo ← tempo + 1
3  d[u] ← tempo
4  for cada v ∈ Adj[u]
5      do if cor[u] = BRANCO
6           then π[v] ← u
7                DFS-VISIT(v)
8  cor[u] ← PRETO
9  f[u] ← tempo ← tempo + 1
```

Algoritmo Terminado !!



Grafos: DFS

- O algoritmo de busca em profundidade é base para a solução de vários outros problemas de grafos:
  - Descobrir se um grafo orientado é acíclico;
  - Descobrir se um grafo é conexo;
  - Achar componentes fortemente conectados;
  - Ordenação Topológica;

Tópicos

- ~~Revisão de grafos e respectiva estrutura.~~
- ~~Algoritmo de BFS.~~
- ~~Algoritmo de DFS.~~
- Resumo.
- Exercícios.

Resumo

- Foram vistos algoritmos de busca em grafos:
  - Busca em Largura (BFS);
  - Busca em Profundidade (DFS);
- A seguir, vamos utilizar os algoritmos de busca em grafos para solucionar outros problemas conhecidos em grafos.

Tópicos

- ~~Revisão de grafos e respectiva estrutura.~~
- ~~Algoritmo de BFS.~~
- ~~Algoritmo de DFS.~~
- ~~Resumo.~~
- Exercícios.

Exercício

- Implemente todas as funções básicas de busca em um grafo (BFS e DFS):
  - Criação do grafo;
  - Inserção de arestas;
  - Busca DFS e BFS;
  - Impressão dos Resultados;
  - Sair;

```
struct grafo{
        int v; (número de vértices)
        int matriz[][]; (matriz adjacência)
};
typedef struct grafo Grafo;

struct resultado{
        int *cor;
        int *predecessor;
        int *distancia;
};
typedef struct resultado Resultado
```

- Use matriz ou lista de adjacências e crie uma versão para cada busca.

Referências Bibliográficas

- CORMEN, Thomas H. **Algoritmos: teoria e prática**. 3° ed. Rio de Janeiro: Elsevier, 2012. xvi, 926 p.

Sugestão de Leitura

- CORMEN, Thomas H. **Algoritmos: teoria e prática**. Parte VI, Capítulo 22, páginas 419-431.

Dúvidas?

Professor Luciano Brum
email: lucianobrum18@gmail.com
https://sites.google.com/view/brumluciano