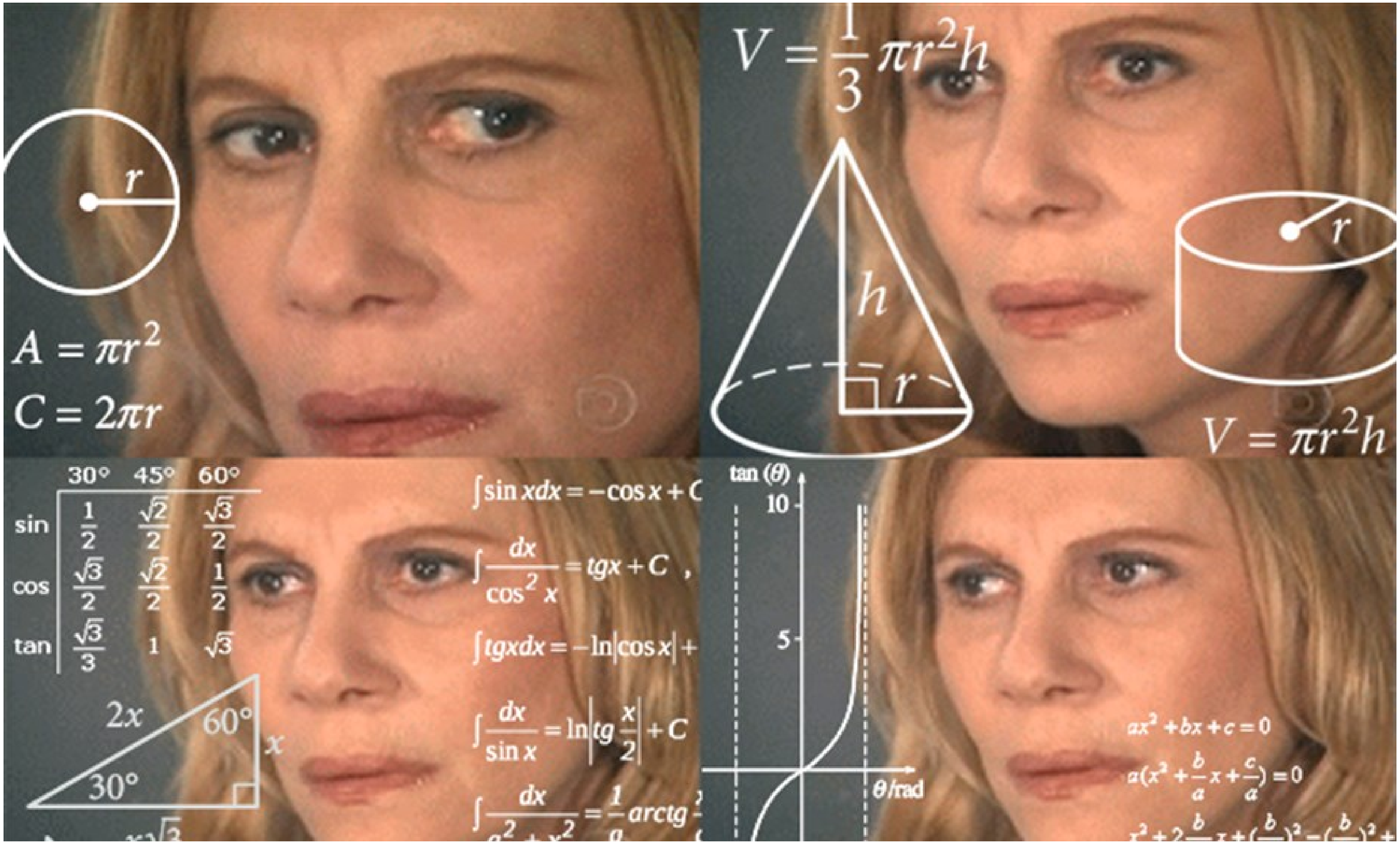
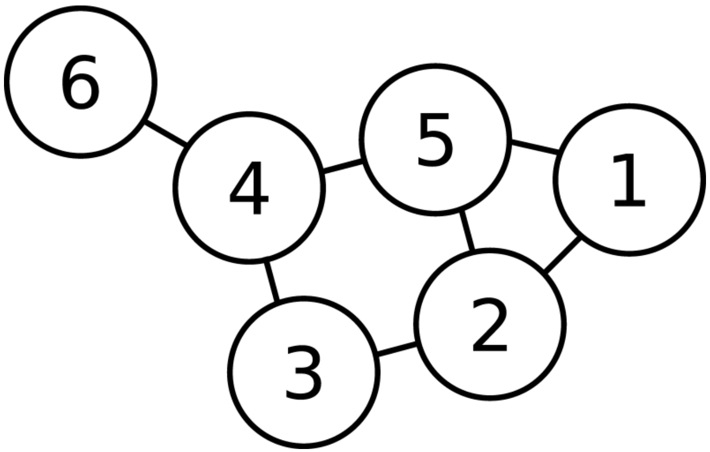
Grafos



$A = \pi r^2$
$C = 2\pi r$
$V = \frac{1}{3}\pi r^2 h$
h
r
$V = \pi r^2 h$
30° 45° 60°
sin $\frac{1}{2}$ $\frac{\sqrt{2}}{2}$ $\frac{\sqrt{3}}{2}$
cos $\frac{\sqrt{3}}{2}$ $\frac{\sqrt{2}}{2}$ $\frac{1}{2}$
tan $\frac{\sqrt{3}}{3}$ 1 $\sqrt{3}$
2x
60°
x
30°
$\int \frac{dx}{\cos^2 x} = tgx + C$
$\int \frac{dx}{\sin x} = \ln\left|tg\frac{x}{2}\right| + C$
tan (θ)
10
5
θ/rad
$ax^2 + bx + c = 0$
$a(x^2 + \frac{b}{a}x + \frac{c}{a}) = 0$

Grafos

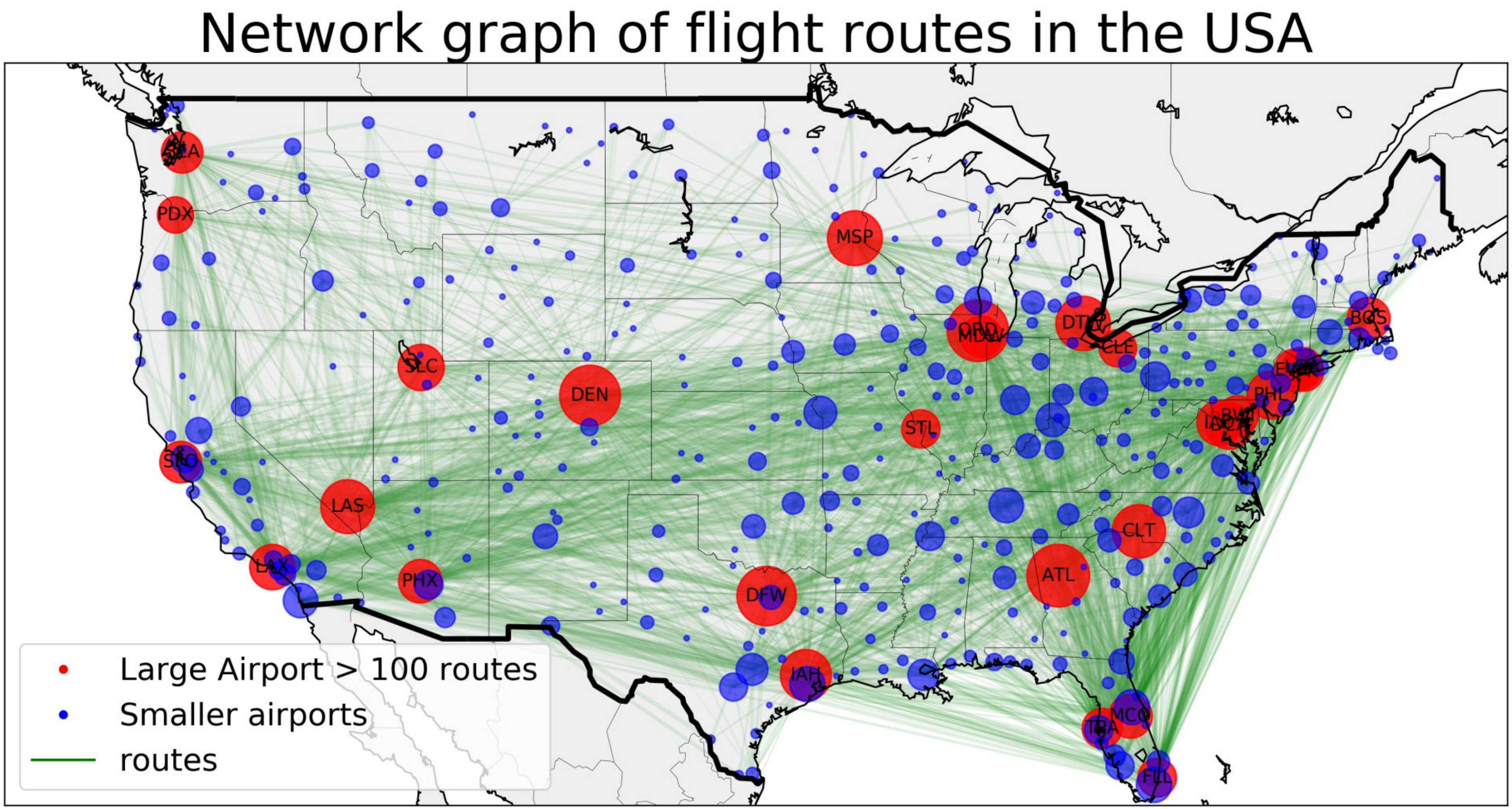
- **Definição (Grafo):** Um grafo **G = G(V, E)** é uma estrutura matemática constituída pelos seguintes conjuntos:
  1. **V:** denominado conjunto de vértices **i** (ou nós do grafo).
  2. **E:** denominado conjunto de arestas **(i,j)** (ou arcos), que conectam dois vértices **i** e **j** do conjunto **V**.



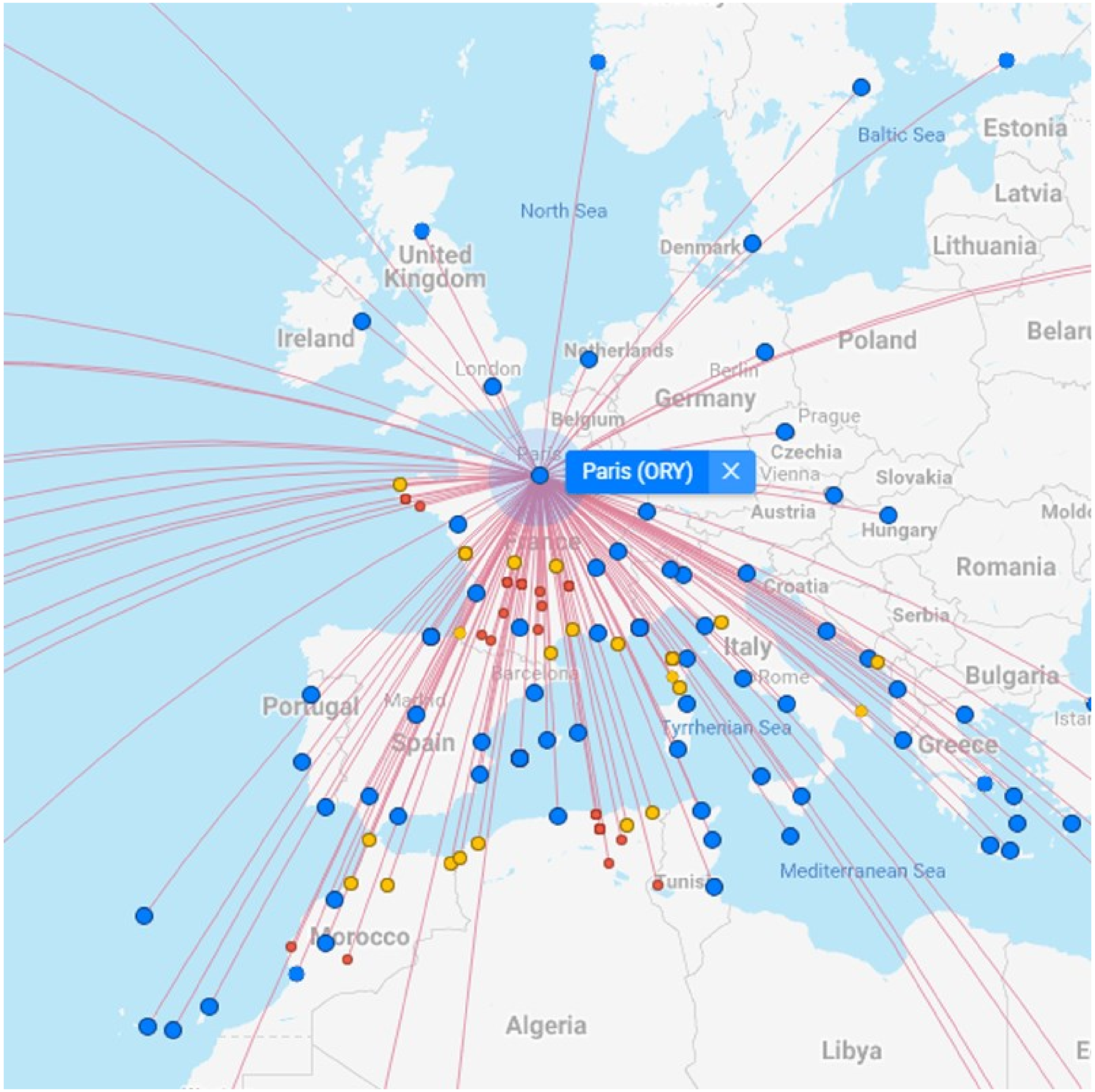
Exemplo:

- V = {1,2,3,4,5,6}
- A = {(1,2), (1,5), (2,3), (2,5), (3,4), (5,4), (4,6)}

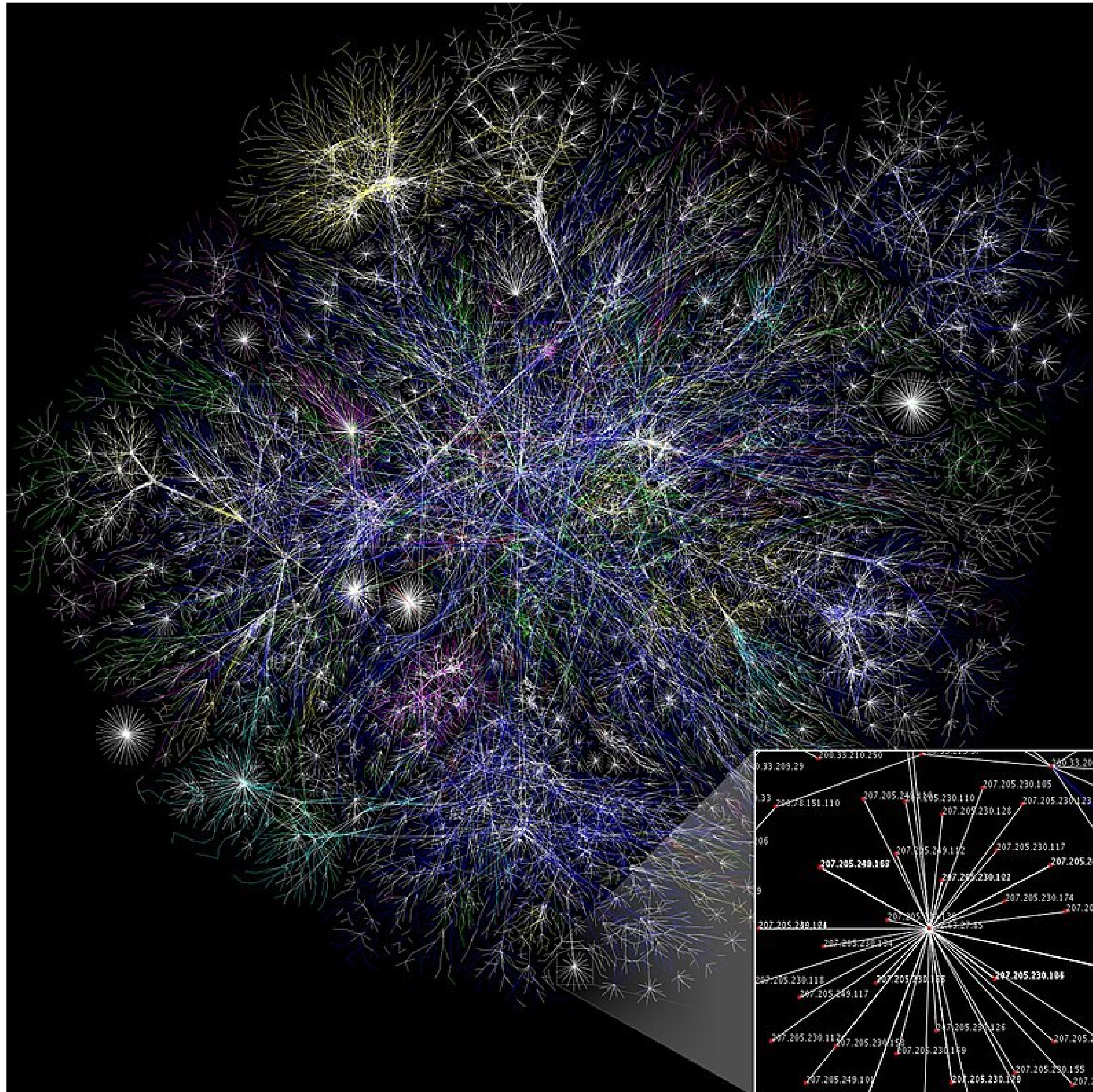
Grafos – motivação



- # Redes de conexões aéreas



- **Mapa (parcial) da internet em 2005**
- Link: en.wikipedia.org/wiki/Small-world_network#/media/File:Internet_map_1024.jpg



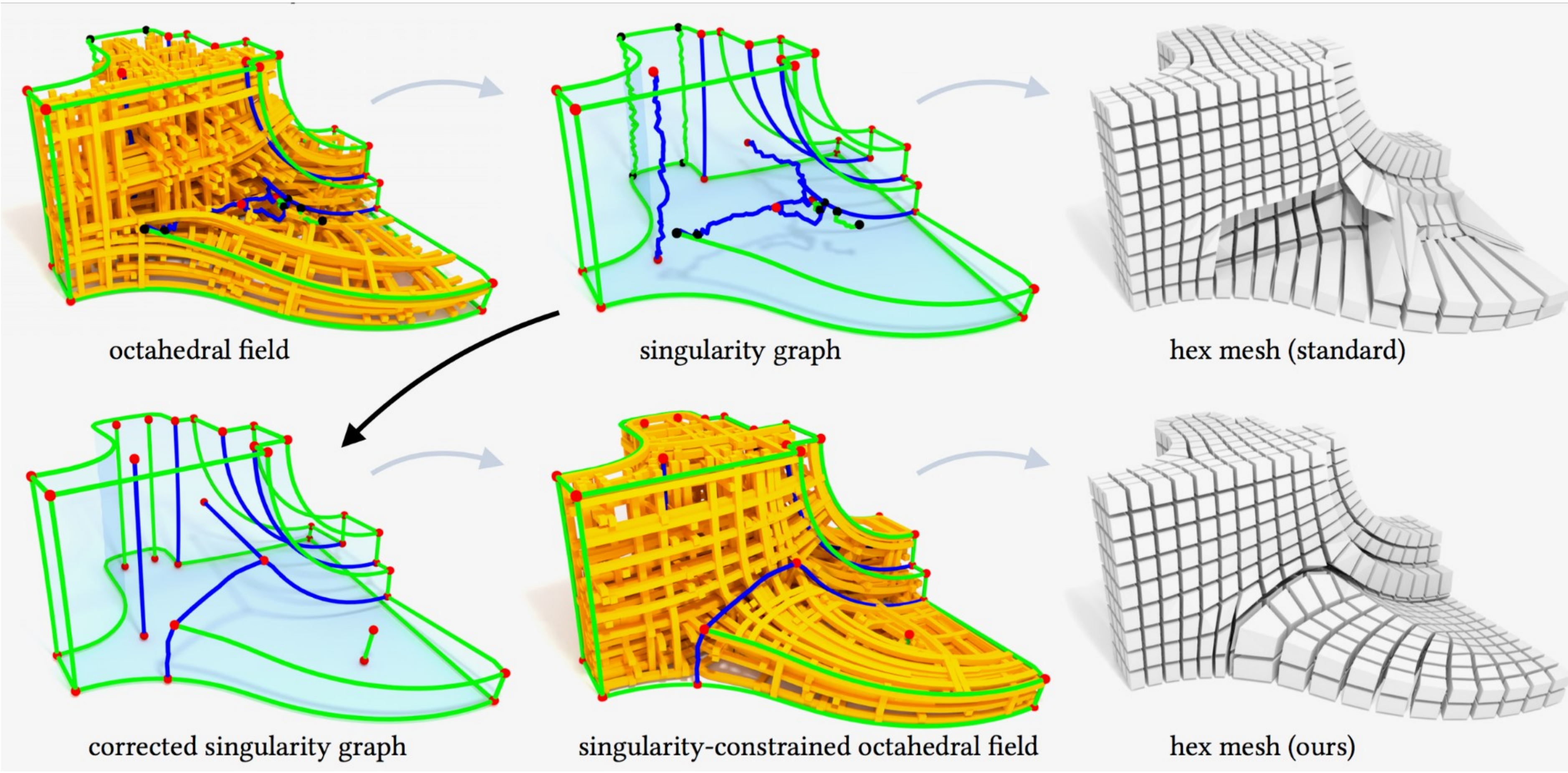
Grafos – motivação

- **Rede sociais**

A map of the global audience for Facebook, created by Paul Butler, visualizes the geographic spread of its user base. (Source: Facebook)

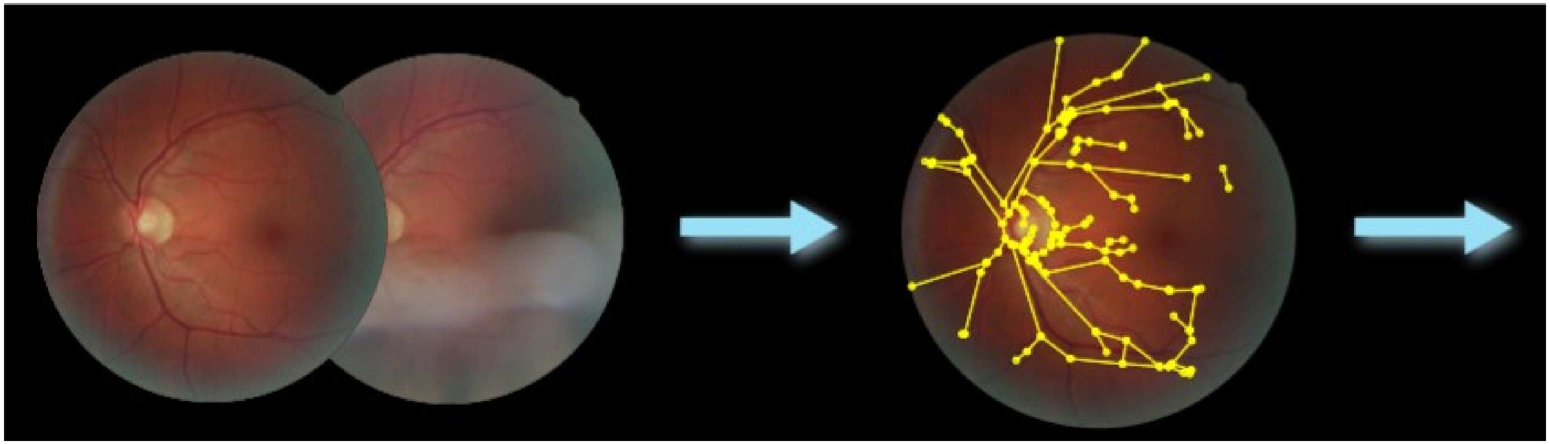
Grafos – motivação

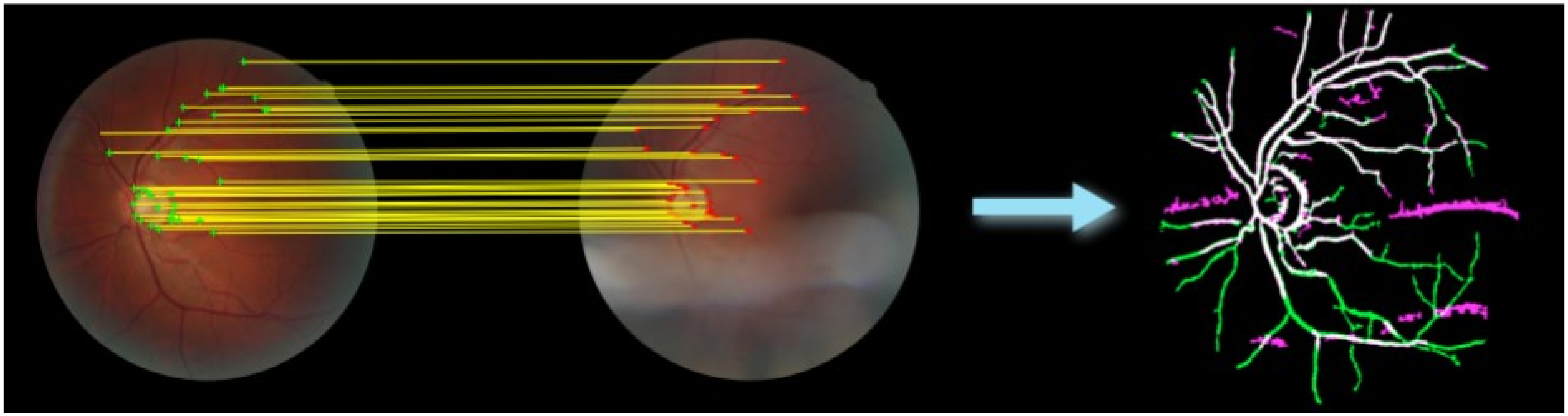
- **Simulação numérica e computação gráfica**



Grafos – motivação

- **Tratamento de imagens de retina (do fundo do olho)**

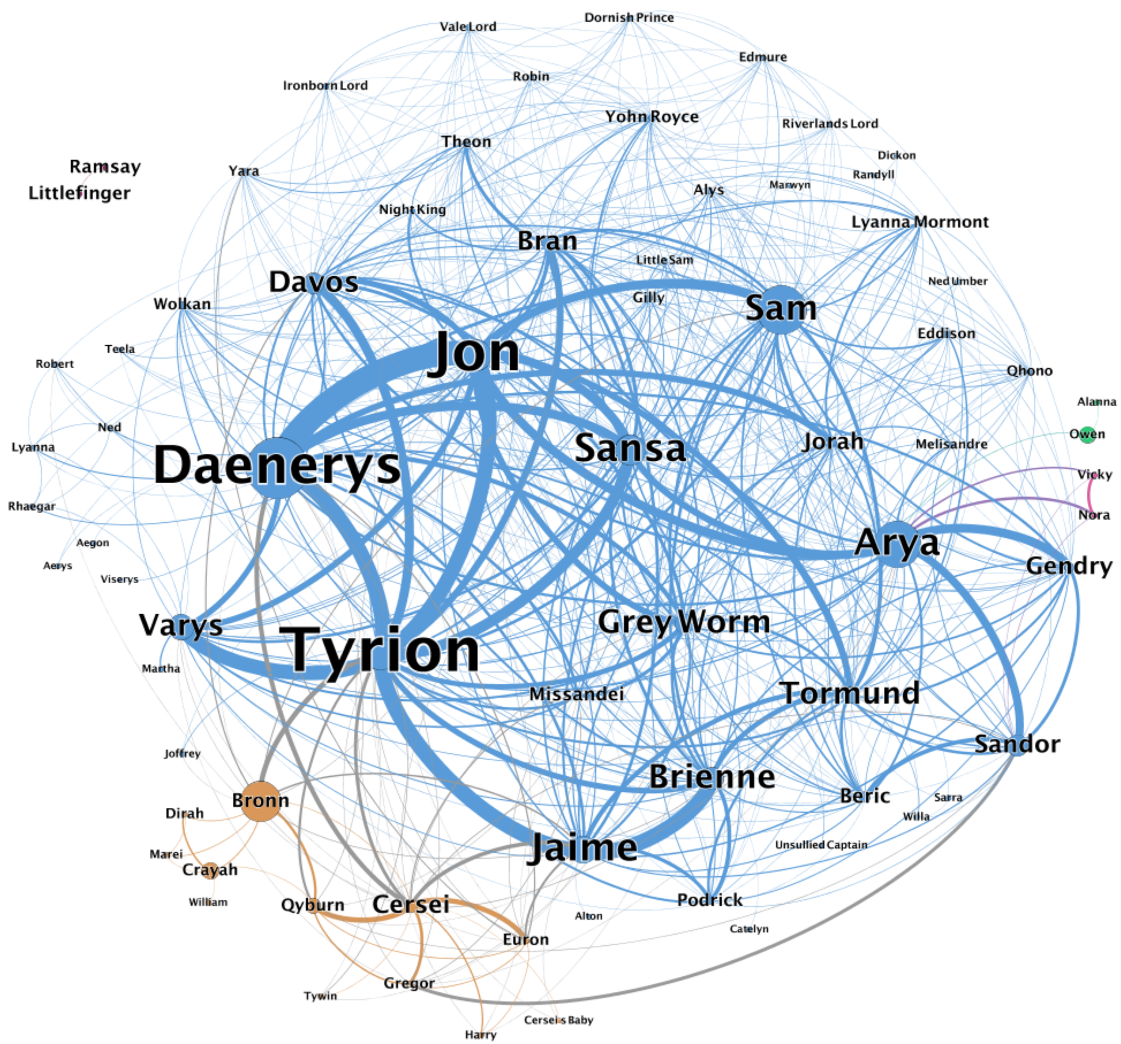




Danilo Motta, Wallace Casaca, Afonso Paiva, Vessel Optimal Transport for Automated Alignment of Retinal Fundus Images, IEEE Transactions on Image Processing, 2019.

Grafos – motivação

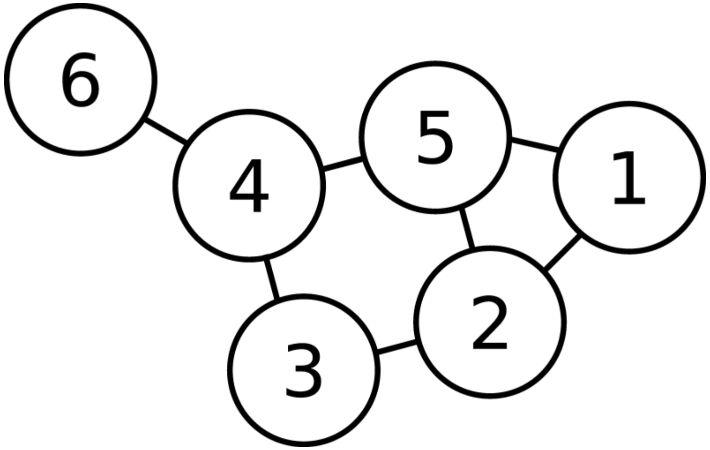
- **Grafo de relação da série Game of Thrones**




EU VEJO GRAFOS
(COM QUE FREQUÊNCIA?)
O TEMPO INTEIRO

Grafos

- **Definição (Grafo):** Um grafo **G = G(V, E)** é uma estrutura matemática constituída pelos seguintes conjuntos:
  1. **V:** denominado conjunto de vértices **i** (ou nós do grafo).
  2. **E:** denominado conjunto de arestas **(i,j)** (ou arcos), que conectam dois vértices **i** e **j** do conjunto **V**.



Exemplo:

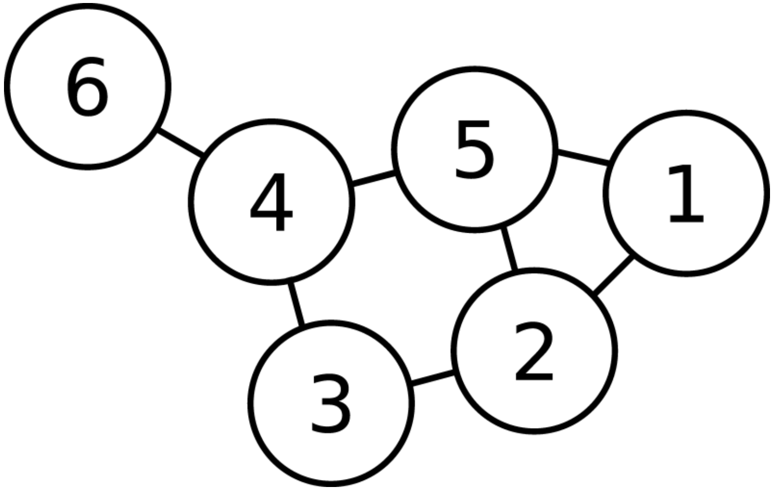
- V = {1,2,3,4,5,6}
- A = {(1,2), (1,5), (2,3), (2,5), (3,4), (5,4), (4,6)}

Grafos – definições relacionadas

- **Grafos:** adjacência e grau.
  - **Vértices adjacentes:** são vértices conectados por uma aresta.
    - Dizemos que as arestas são incidentes à um vértice.
  - **Grau de um vértice:** número de arestas incidentes.

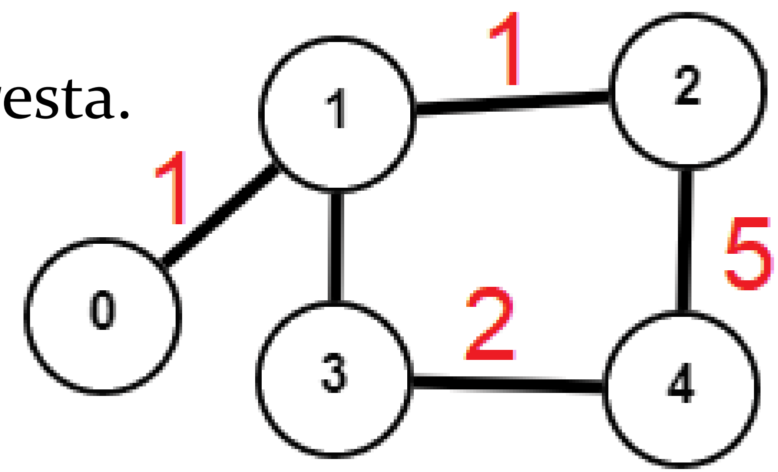
Exemplo:

- 6 e 4 são vértices adjacentes.
- O nó 4 tem grau = 3.



Grafos – definições relacionadas

- **Grafos:** vizinhança e pesos nas arestas
  - **Vizinhança** (de um dado vértice **vi**): subconjunto formado pelos vértices **vj** que estão conectados a **vi** por uma aresta.
  - **Peso:** valor atribuído a uma aresta.

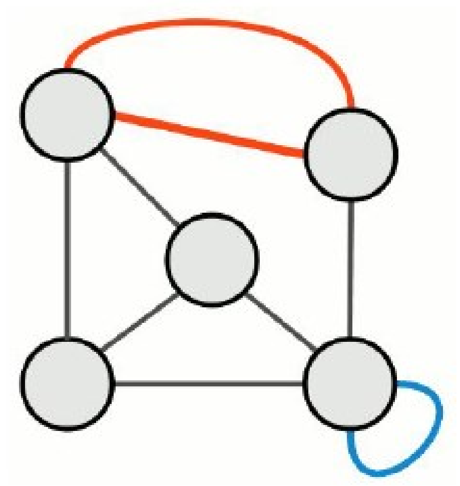


Exemplo:

- A vizinhança do nó = 1 é o subconjunto Viz(1) = {0,2,3}
- A aresta (3, 4) tem peso = 2

Grafos – definições relacionadas

- **Grafos:** laços e arestas múltiplas.
  - Um **laço** (**loop**) é uma aresta que conecta um vértice a ele mesmo.
  - Já as **arestas múltiplas** ocorrem quando existe a possibilidade de mais de uma aresta conectar o mesmo par de vértices.

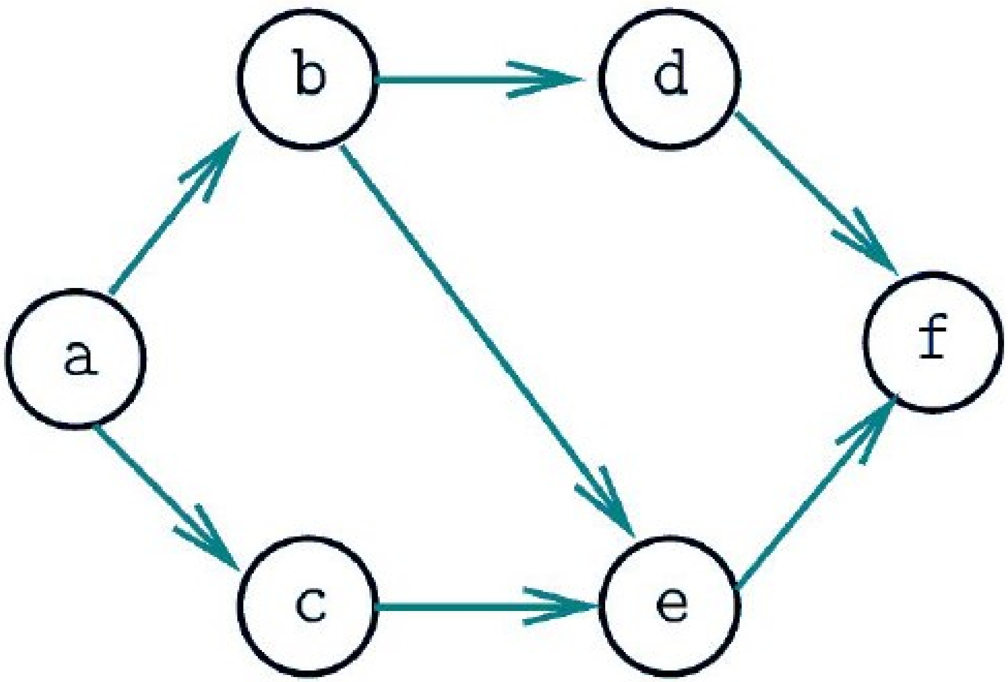


Exemplo:

- laço (em azul).
- arestas múltiplas (em vermelho).

Grafos – definições relacionadas

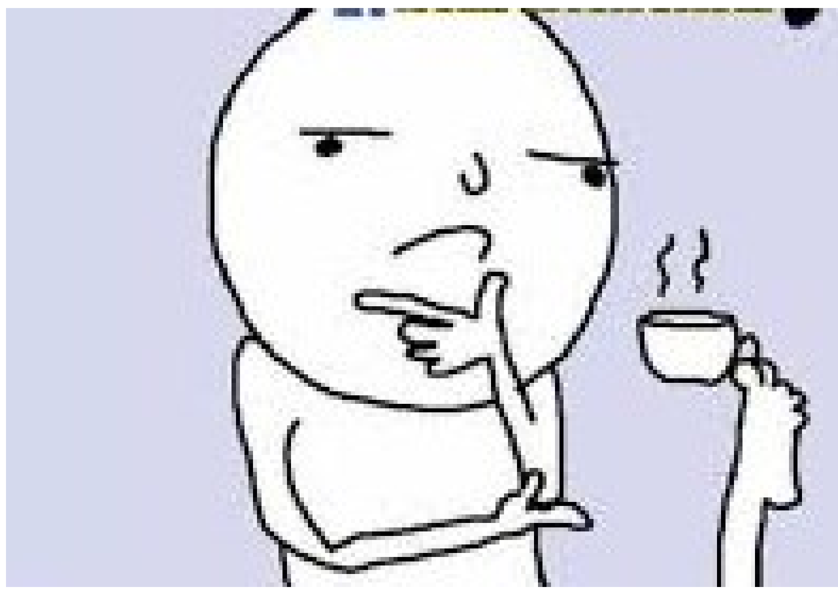
- **Dígrafo:** É um **grafo direcionado**, cujas arestas possuem direções específicas (são na verdade flechas).
  - O primeiro vértice do par ordenado é a **ponta inicial do arco**, e o segundo, **a ponta final**.



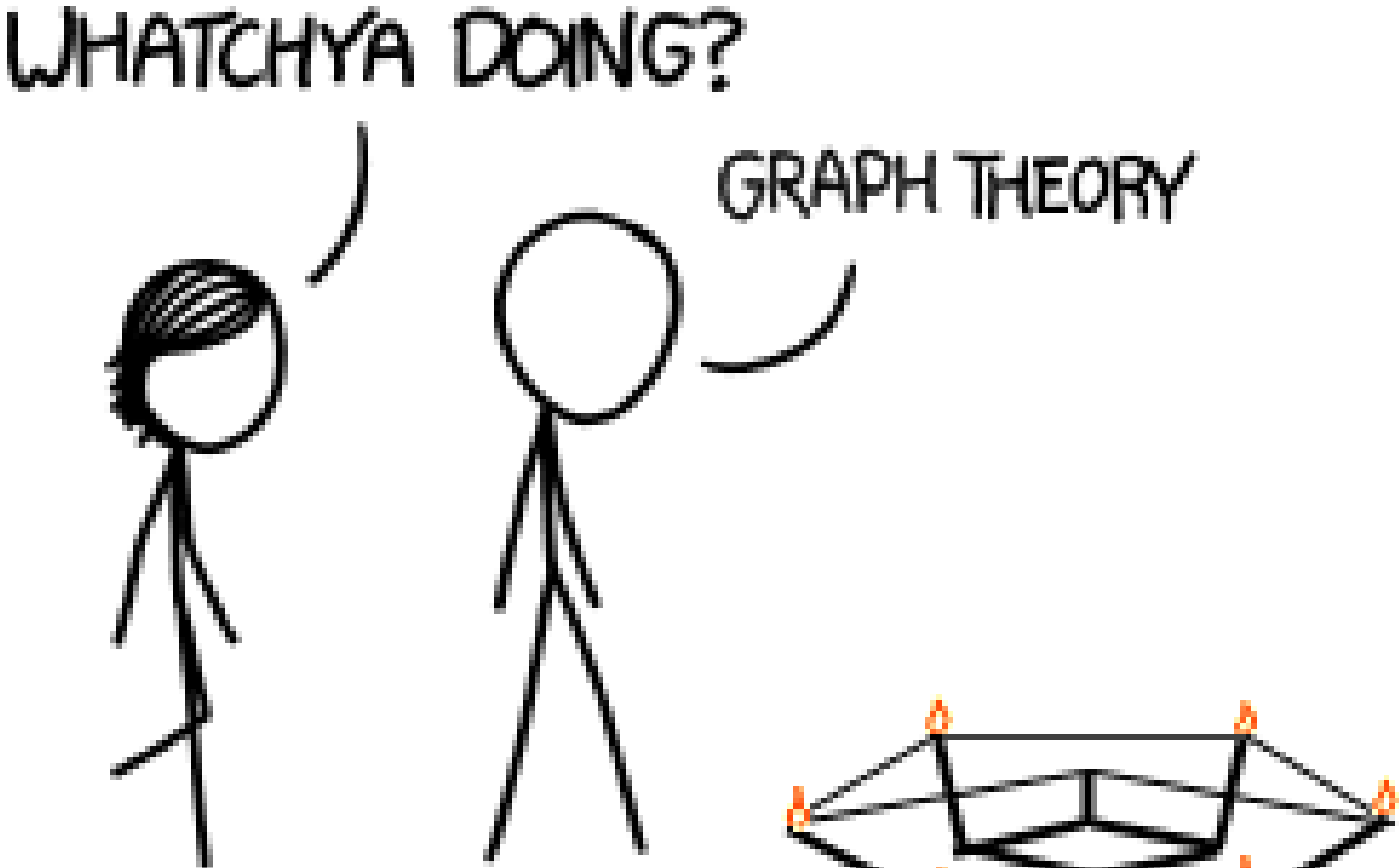
Grafos – observações

- O conjunto **E** determina o grau de relação (**conexão**) entre os nós do conjunto **V**.
- As arestas podem ter pesos (**valores**) associados.
- Grafos possibilitam modelar não somente um conjunto de objetos, mas também **a relação entre esses elementos**.
- **Teoria dos Grafos:** é uma área de matemática que envolve uma série de resultados importantes obtidos principalmente a partir do século XVII.

Perguntas?

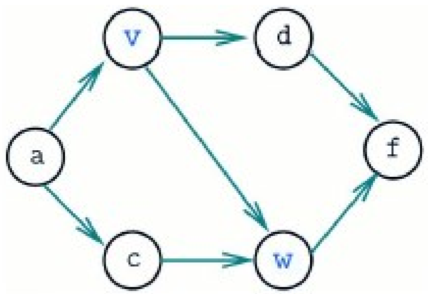


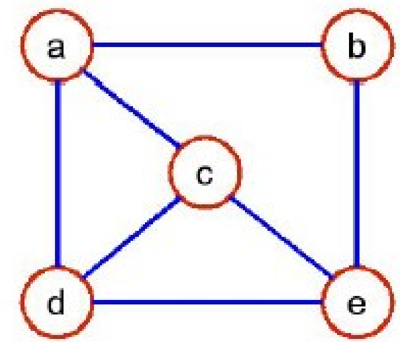
Grafos – TADs

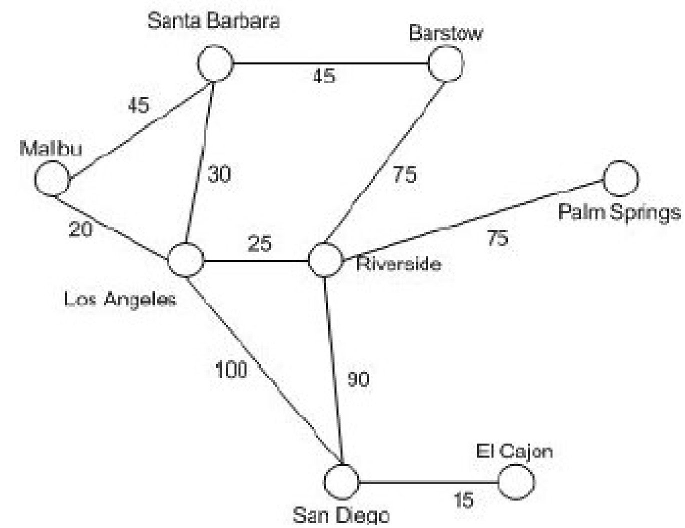


Grafos – TADs

- **Perguntas iniciais:**
  1. Como representar um TAD para um grafo?
  2. Precisamos de representações distintas para grafos e dígrafos?
  3. E se o grafo possuir arestas com pesos?
  4. Como ler arestas e vértices e armazenar na memória?
     - Vamos começar pelo último ponto!



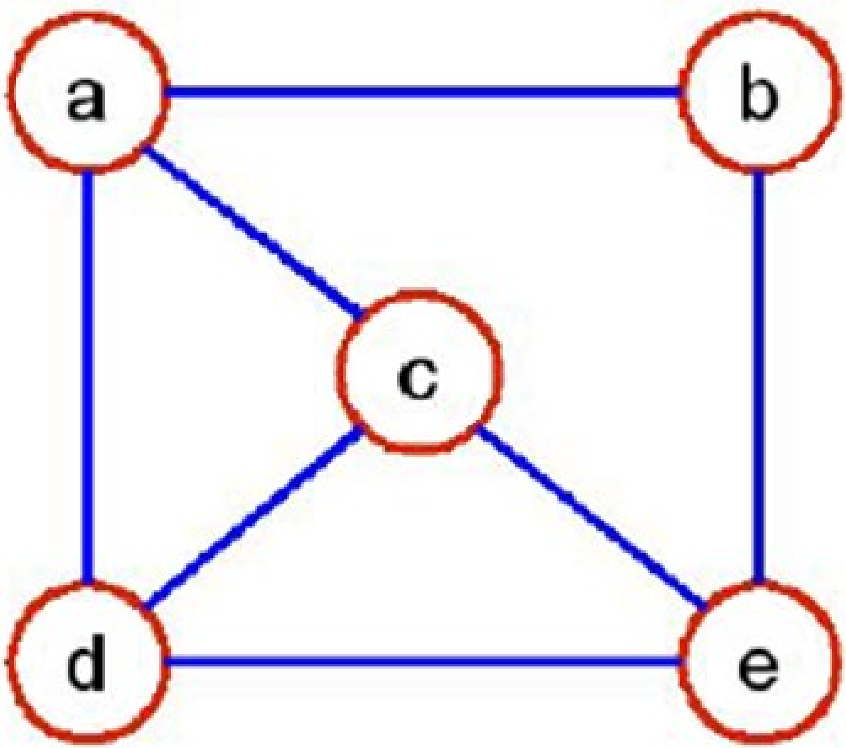




Grafos – lendo vértices e arestas

- **Representação** do grafo a partir de um **padrão pré-definido** em um arquivo de texto.

5
a b
a c
a d
b e
c d
c e
d e



- **Representação** do grafo a partir de um **padrão pré-definido** em um arquivo de texto.

8

0 1 45

0 2 20

1 2 30

1 3 45

2 4 25

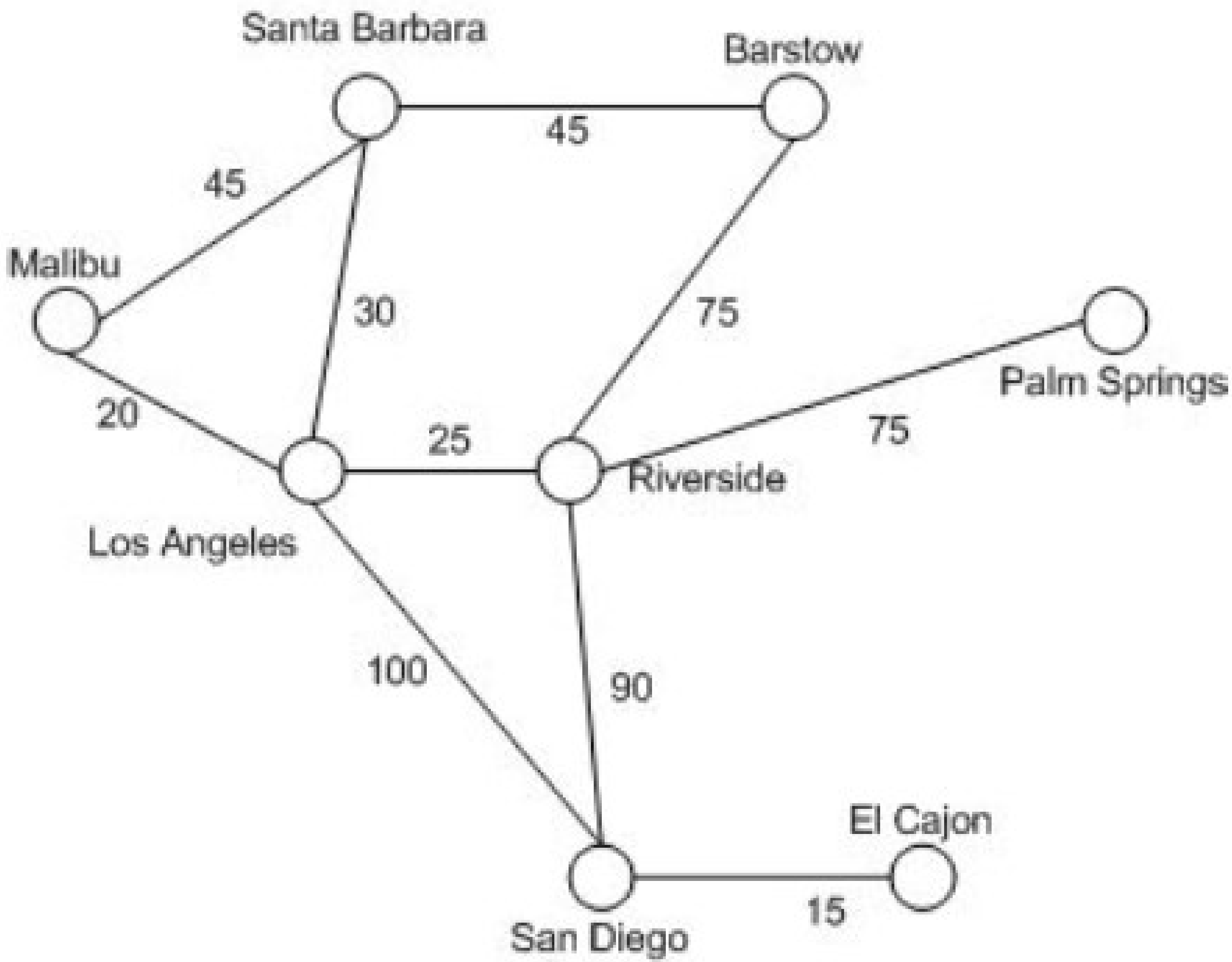
2 5 100

3 4 75

4 5 90

4 6 75

5 7 15



Grafos x Dígrafos – TAD

- É necessário diferenciar a estrutura de dados entre um grafo e um dígrafo?

- As EDs não precisam ser necessariamente diferentes (porém, em alguns casos **requer ajustes**!).

- O que muda: em um grafo, ao contrário do dígrafo, ambos os arcos v-w e w-v necessitarão estar representados na ED.

Grafos – TADs mais comuns

- Lista de arcos (ou vértices).
- Lista de adjacências.
- Matriz de adjacência.
- Matriz de incidência.

Grafos – TADs mais comuns

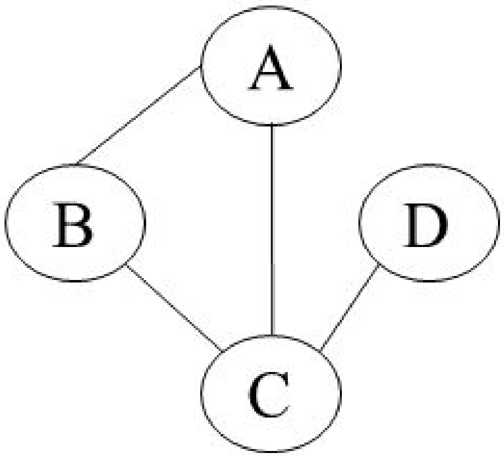
- Lista de arcos (ou vértices).
- Lista de adjacências.
- Matriz de adjacência.
- Matriz de incidência.

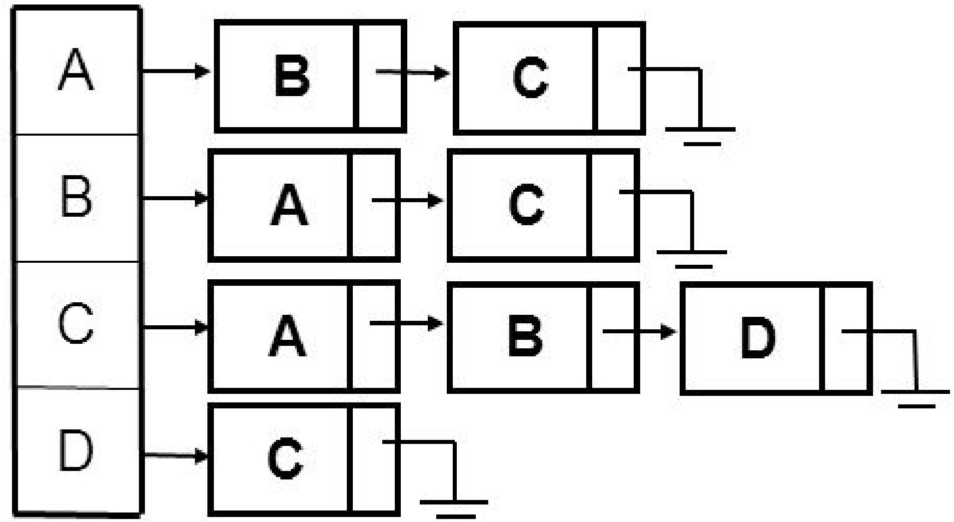
Grafos – lista de adjacência

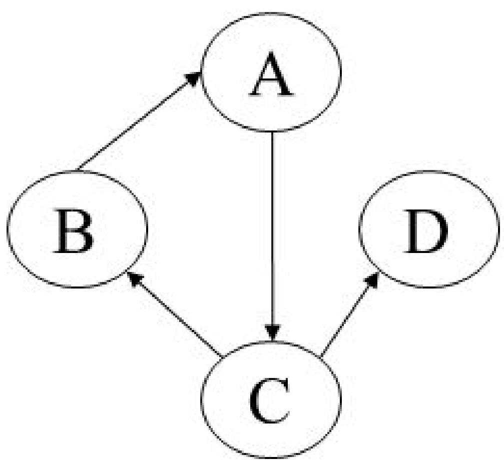
- Especifica os **vértices adjacentes a cada vértice** do grafo.

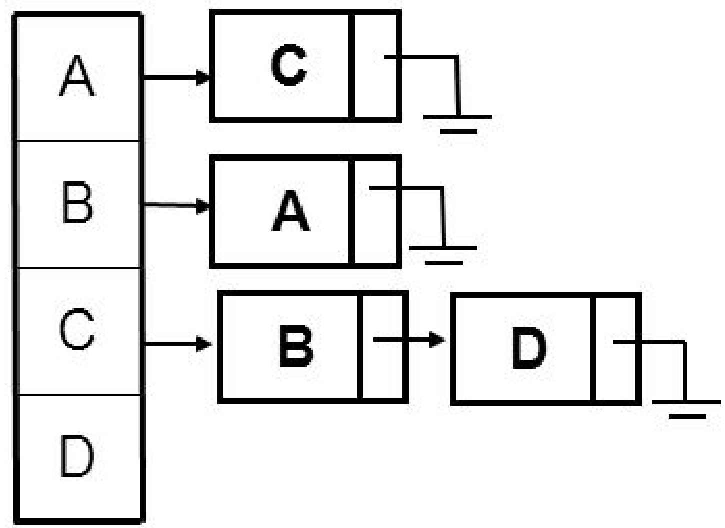
- **Implementações possíveis:**
  1. Implementação via tabela.
  2. Lista encadeada de vértices, com ponteiro para uma lista de adjacências (que também pode ser encadeada).
  3. Um *array* de vértices (estático), com ponteiro para uma lista de adjacências (também uma lista encadeada).

Grafos – lista de adjacência









Grafos – lista de adjacência

- Estrutura ideal para armazenar grafos esparsos.
  - por que? (pense em uma matriz, ao invés de uma lista).

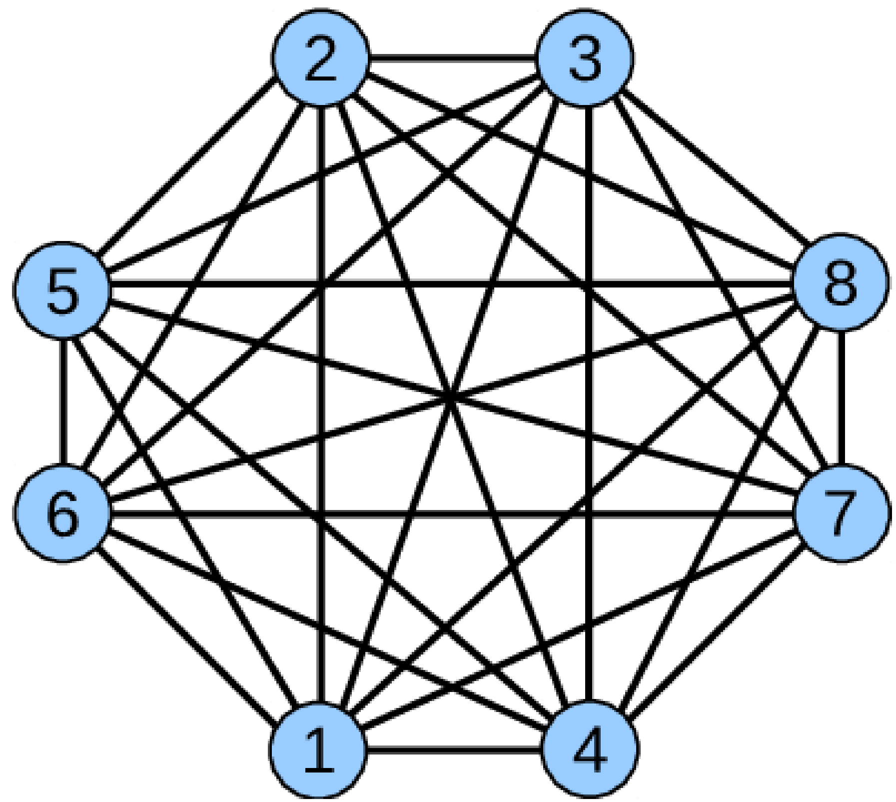
Grafos – lista de adjacência

- **Grafo completo (cheio):** é um grafo simples, em que todo vértice é adjacente a todos os outros vértices. Notação: K_n

- Grafo completo K_8

- Número de nós de K_n

$$\binom{n}{2} = \frac{n(n-1)}{2} = \mathrm{O}(n^2)$$



Grafos – lista de adjacência

- **Conclusão:** um grafo simples pode ter, no máximo, **n(n-1)/2 arestas**.
- Mas em problemas com muitos nós e poucas arestas?
- **Facebook:**
  - 2.9 bilhões de usuários por mês ativos (Abril 2023).
  - Matriz ou tabela (p/ representar relação de amizade): 4.20 10^18 elementos (525 mil teras!).
    - Duas pessoas amigas levaria O(1).
    - Imprimir todos os amigos levaria O(n).
      - Percorrer 2.9 bilhões posições na matriz!
      - Usuário em geral não tem isso tudo de amigo!!

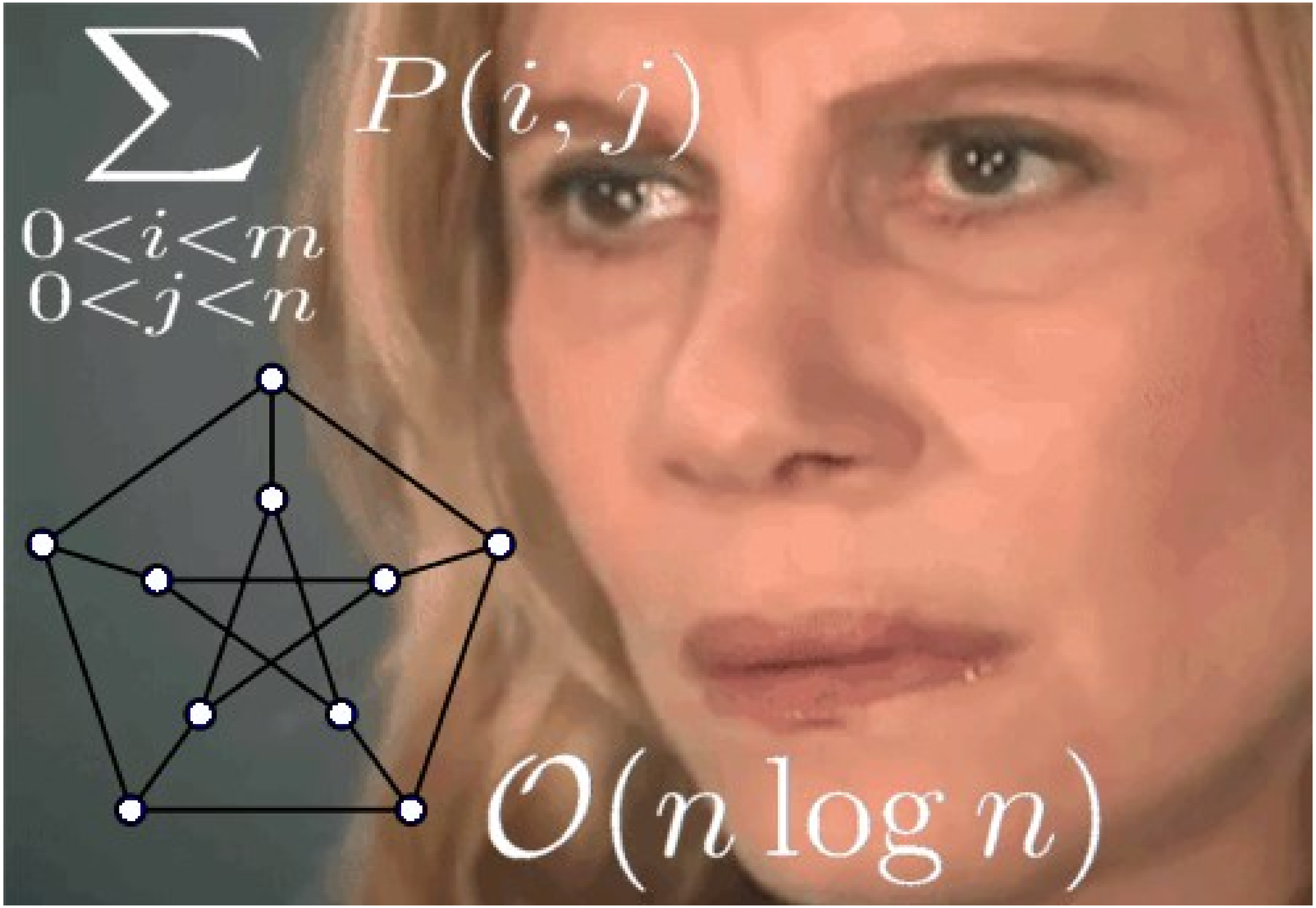
Grafos – lista de adjacência

- **Grafo esparso:** Se o número de arestas é da mesma ordem de V.

- **Facebook:**
  - Cada usuário (padrão) tem, no máximo, 5.000 amigos.
  - Máximo de arestas da rede será 7.25 10^12 (bem menos que 4.20 10^18 da matriz cheia do grafo completo).
  - Grafos com grau máximo d (constante).
    - Número de arestas: d (n/2) = O(n) << O(n^2).

- **Melhor representar o grafo como uma lista de adjacência!**

TAD Grafos – lista de adjacência



```c
//EDs para nó e grafo
//-----------------------------------
struct no {
    int id;
    int val;
    struct no *prox;
};

typedef struct no *No;

struct grafo {
    int id;
    int nNo; //nro de nós
    No vertices; //array de vértices
};

typedef struct grafo *Grafo;
```

```c
No criaNo(int id, int val){
    No n = (No) malloc(sizeof(struct no));
    n->id = id;
    n->prox = NULL;
    n->val = val;
    return n;
}
 void addNo(No n, int id, int val){
    No novo = criaNo(id, val);
    if(n == NULL){
        return;
    }
    while(n->prox != NULL){
        n = n->prox;
    }
    n->prox = novo;
 }
```

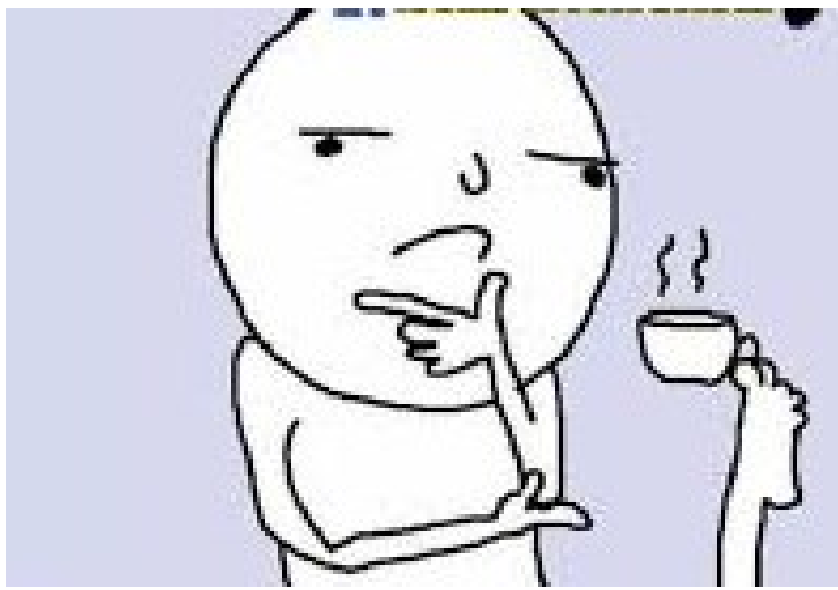
TAD Grafos – lista de adjacência

```
Grafo criaGrafo(){
    Grafo G = (Grafo) malloc(sizeof(struct grafo));
    G->vertices = NULL;

    return G;
}
```

TAD Grafos – lista de adjacência

```
//Primeira linha do arquivo indica o numero de vertices
fgets(buffer,bsize, fp);
sscanf(buffer,"%d", &G->nNo); //Salva o numero de vertices
G->vertices = (No) malloc(G->nNo * sizeof(struct no)); // Cria o vetor de vertices
//Primeiro elemento de cada lista identifica o vertice que a lista representa
for(i = 0; i < G->nNo; i++){
    (G->vertices + i)->id = i;
    (G->vertices + i)->val = -1; // Valor default (nao utilizado)
    (G->vertices + i)->prox = NULL;

}
//Percorre o arquivo
while(!feof(fp)){
    fgets(buffer,bsize, fp);
    sscanf(buffer,"%d %d %d", &o, &d, &v);
    //Adiciona um novo vertice com id = d e valor = v na lista da posicao o do vetor
    addNo((G->vertices + o), d, v);
}
```

Perguntas?

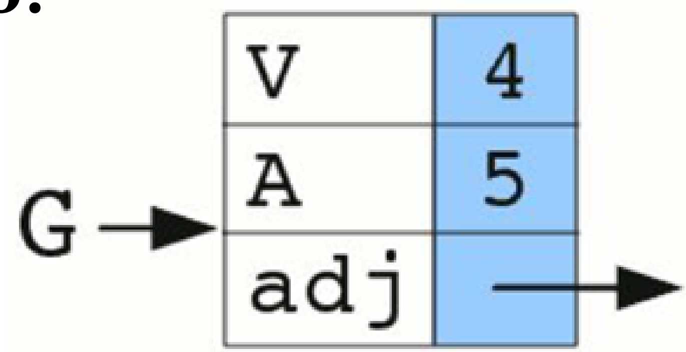


Grafos – matriz de adjacência

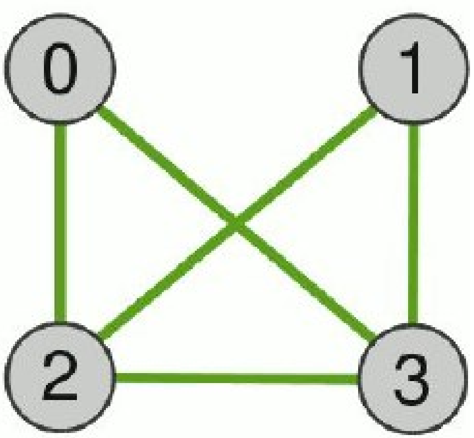
- **Definição (Matriz de Adjacência)**: Matriz binária, de tamanho $\#V \times \#V$, tal que cada entrada $d_{i,j}$ é:
  - 1, se existe uma aresta (v_i, v_j).
  - 0, caso contrário.
- **Observação 1:** pode ser generalizada para multigrafos se, ao invés de utilizarmos 0's e 1's, indicarmos $d_{i,j}$ = número de arestas entre v_i e v_j.
- **Observação 2:** se o grafo tiver laços, podemos colocar valores na diagonal principal.

Grafos – matriz de adjacência (grafo)

- **Exemplo:**

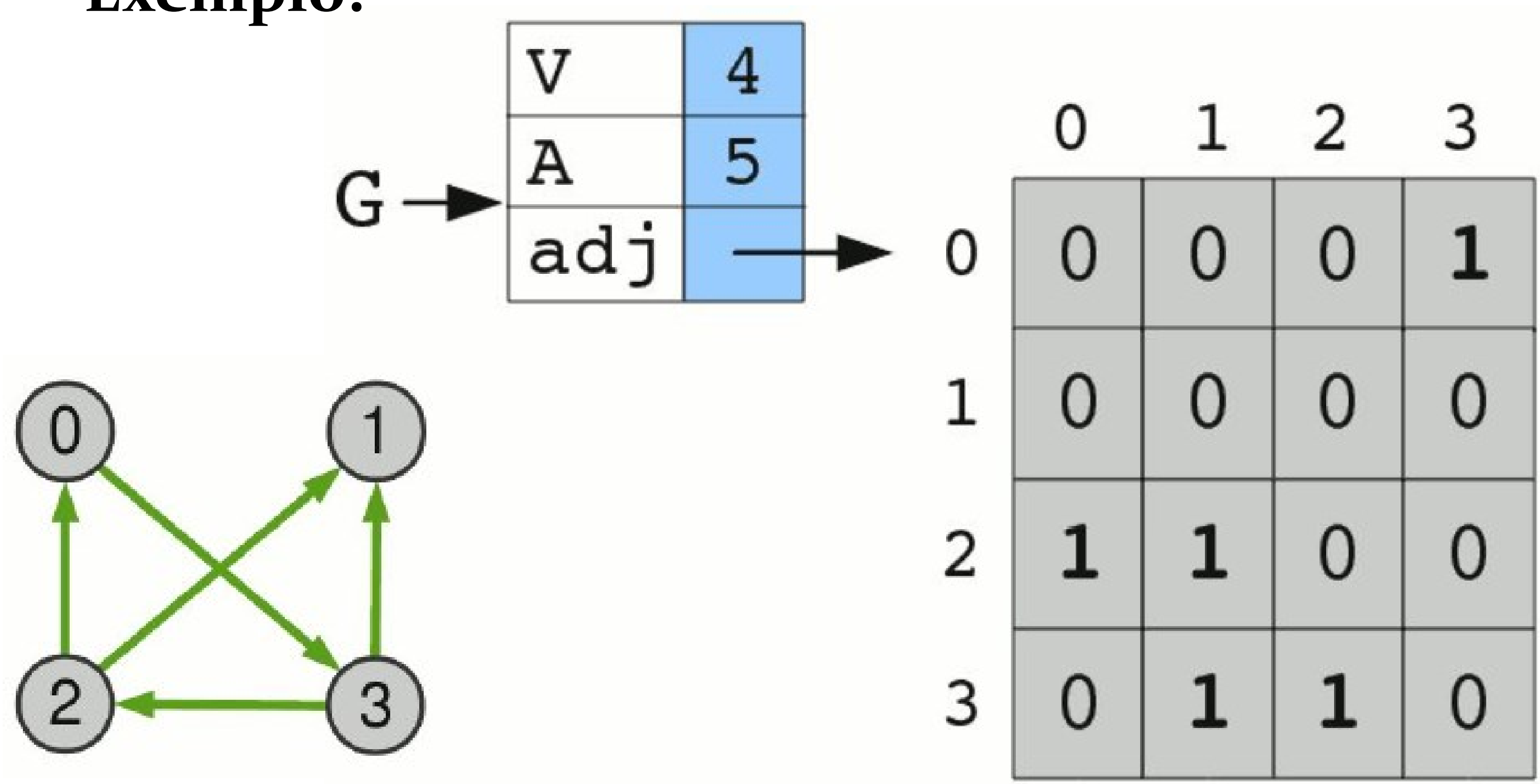


	0	1	2	3
0	0	0	1	1
1	0	0	1	1
2	1	1	0	1
3	1	1	1	0



Grafos – matriz de adjacência (dígrafo)

- **Exemplo:**



TAD Grafos – matriz de adjacência

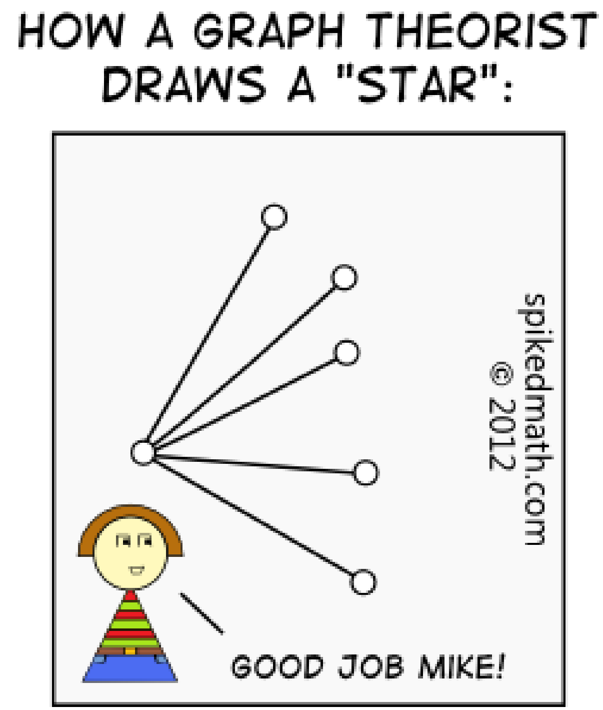
```c
typedef struct grafo{
    int nNo; //nro de nós
    Matrix adj; //Matriz de adj
} Grafo;



Grafo criaGrafo(){
    Grafo G = (Grafo) malloc(sizeof(struct grafo));
    G->ajd = NULL;

    return G;
}
```

TAD Grafos – matriz de adjacência

```
boolean readGraph(Grafo G, const char *filename){
    FILE *fp;
    int bsize = 20;
    int a,b;
    char buffer[bsize];
    fp = fopen(filename,"r");

    fgets(buffer,bsize, fp);
    sscanf(buffer,"%d", &G->nNo);
    G->adj = zeros(G->nNo, G->nNo);

    while(!feof(fp)){
        fgets(buffer,bsize, fp);
        sscanf(buffer,"%d %d", &a, &b);
        setVal(G->adj, b, a, 1.0);
    }

    fclose(fp);
    return True;
}
```

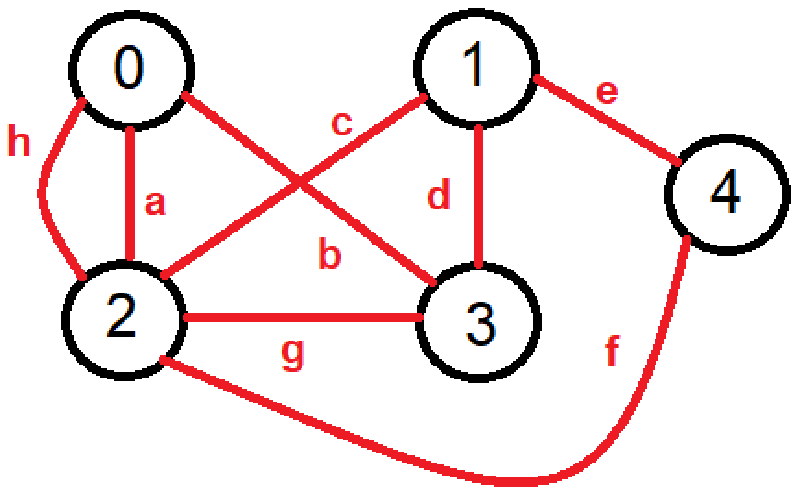
Grafos – matriz de incidência

- **Definição (Matriz de Incidência)**: É uma matriz de tamanho $\#V \times \#A$, baseada na incidência de vértices e arestas, tal que cada entrada $c_{i,j}$ é:
  - 1, se a aresta a_j é incidente com o vértice v_i.
  - 0, caso contrário.



Grafos – matriz de incidência (grafo)

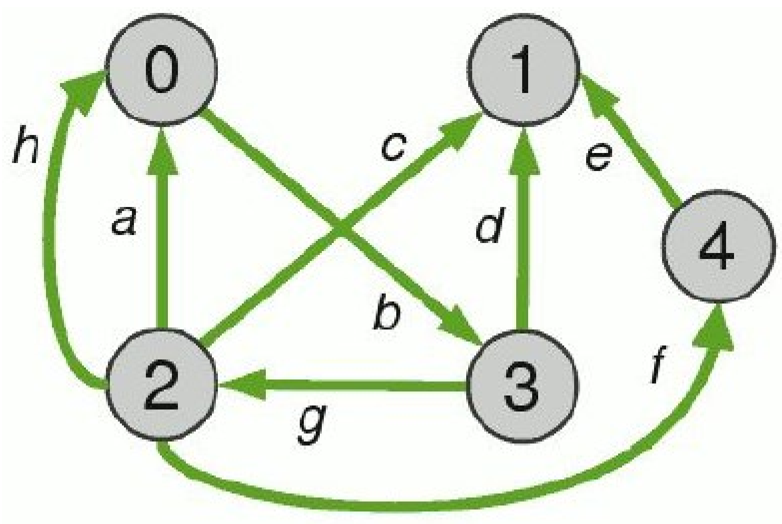
- **Exemplo:**



	a	b	c	d	e	f	g	h
0	**1**	**1**	0	0	0	0	0	**1**
1	0	0	**1**	**1**	**1**	0	0	0
2	**1**	0	**1**	0	0	**1**	**1**	**1**
3	0	**1**	0	**1**	0	0	**1**	0
4	0	0	0	0	**1**	**1**	0	0

Grafos – matriz de incidência (dígrafo)

- **Exemplo:**



	a	b	c	d	e	f	g	h
0	**1**	**–1**	0	0	0	0	0	**1**
1	0	0	**1**	**1**	**1**	0	0	0
2	**–1**	0	**–1**	0	0	**–1**	**1**	**–1**
3	0	**1**	0	**–1**	0	0	**–1**	0
4	0	0	0	0	**–1**	**1**	0	0

Grafos – comparação matriz x lista

- Espaço para o armazenamento:
  - Via Matriz: $O(|V|^2)$ – adjacência; $O(|V||E|)$ – incidência
  - Via Listas: $O(|V| + |E|)$

- Tempo

Operação	Matriz	Lista
Inserção	O(1)	O(1)
Remoção	O(1)	O(d(v))
Checa se há aresta	O(1)	O(d(v))
Percorrer vizinhança	O(1)	O(d(v))

- As duas permitem representar grafos e dígrafos!
- Qual usar?

  Depende das operações usadas e se o grafo é esparso

Perguntas?