Big Data - Spark, Hadoop e afins.

Leonardo Gloria

Aula Grafos

Instituto Infnet

Alguns conceitos sobre grafos.

Introdução

* Teoria de grafos é relativamente recente (Século XVIII) na história da Matemática.
* Bem desenvolvida já no século XX é de extrema importância em outras ciências e áreas da matemática.
* Informalmente, pode ser visto como um conjuntos de pontos(a.k.a : vértices) e outro de pares desses pontos chamados arestas. Cada aresta liga um par de pontos que a determina.

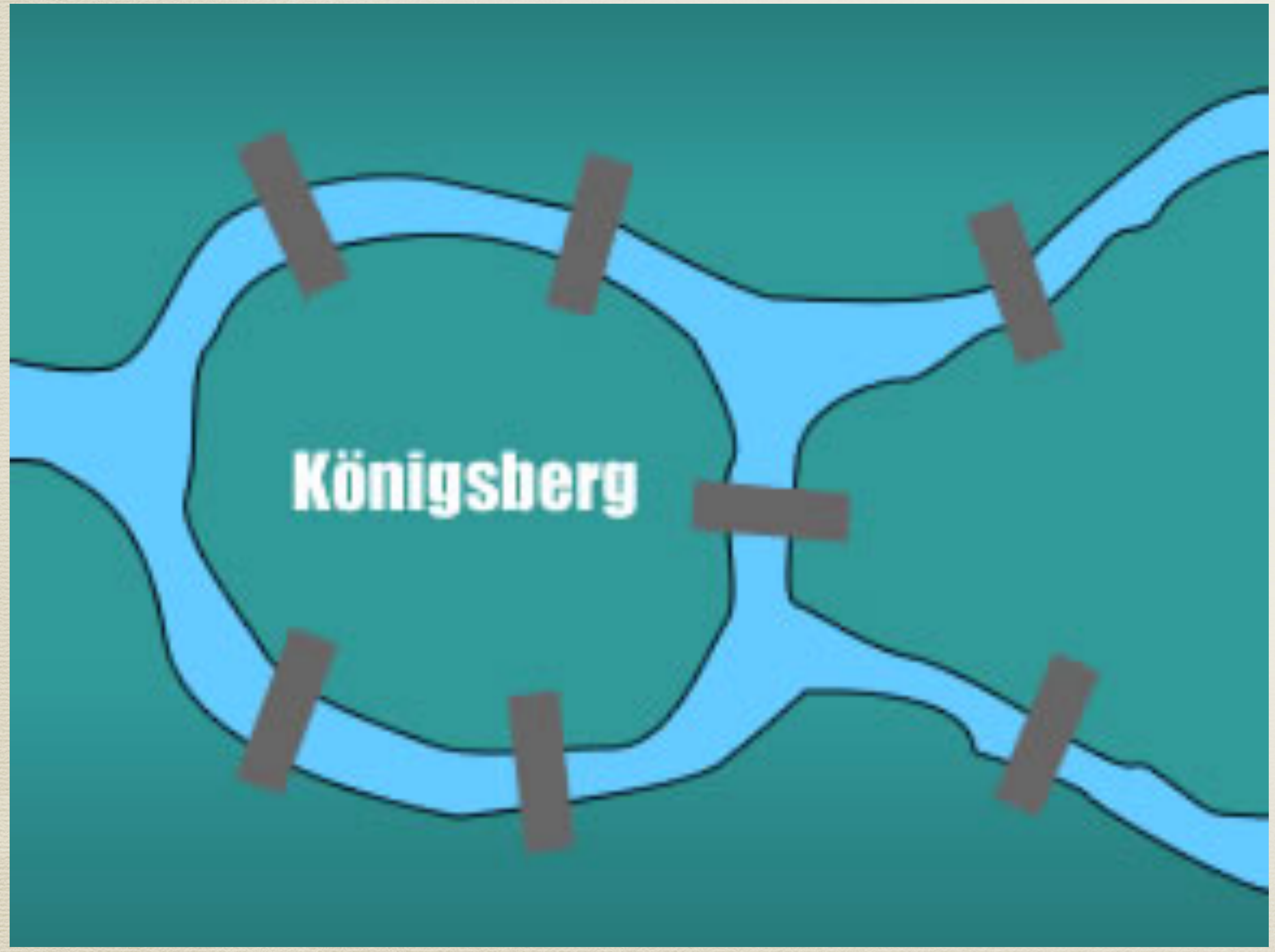
Introdução

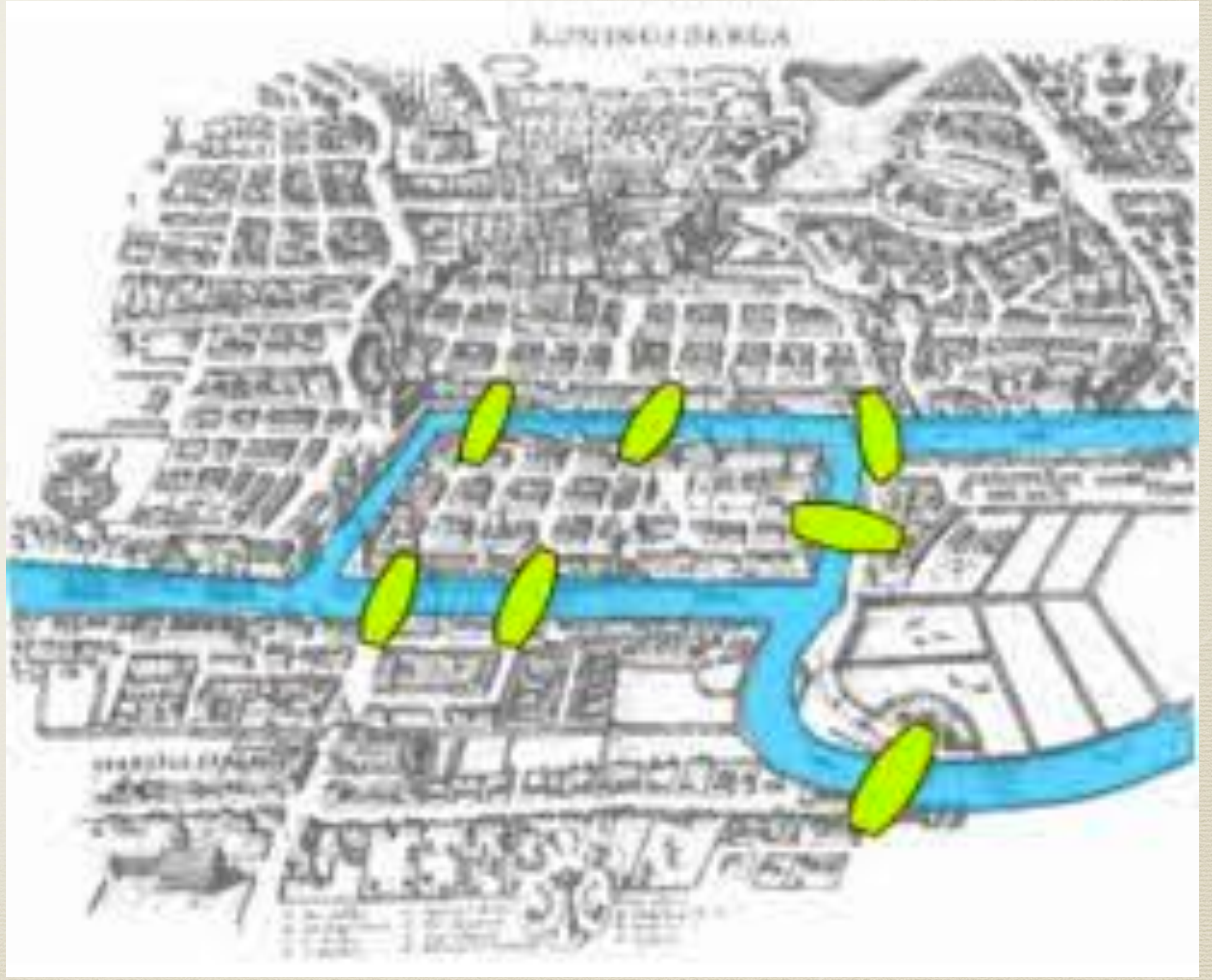
* Esse conceito torna claro que permite a modelagem de situações concretas como, redes de computadores, web, árvores genealógicas, Química orgânica(isômeros)
* Termo grafo foi utilizado pela primeiras fez por James Joseph Sylvester num artigo publicado em 1877, na Nature.
* Apesar disso e da sua definição formal só no séc. XX a resulução de Euler do Problema das pontes de Kønisberg (1976) é referida como a primeira publicação da teoria.

Pontes de Konisberg

* Discutia-se nas ruas a possibilidade de atravessar todas as pontes sem repetir nenhuma.
* Tornou-se uma lenda popular a possibilidade da façanha.
* Leonhard Euler , em 1736 provou que não existia caminho que possibilitasse tais restrições.

Pontes de Konisberg

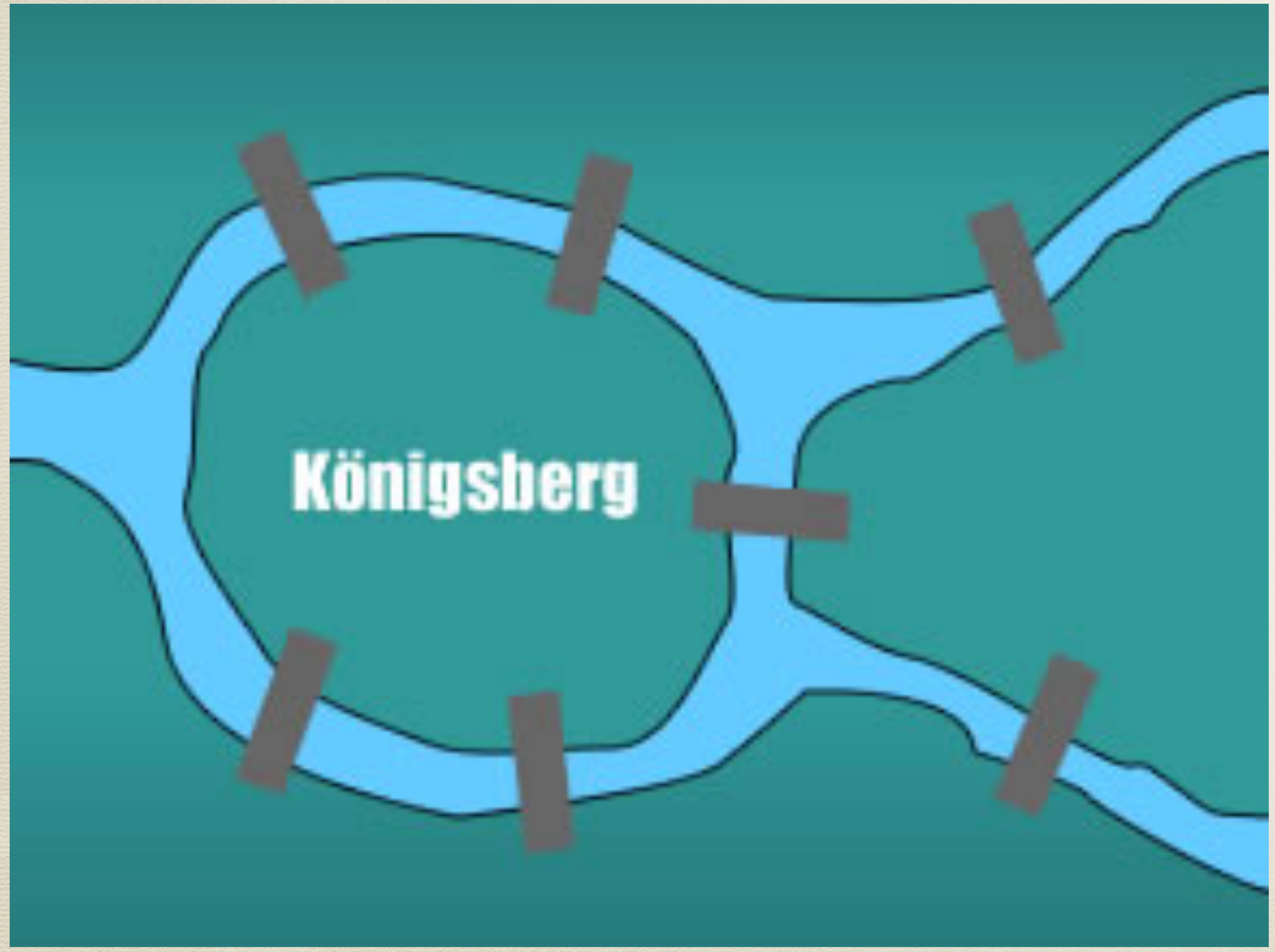


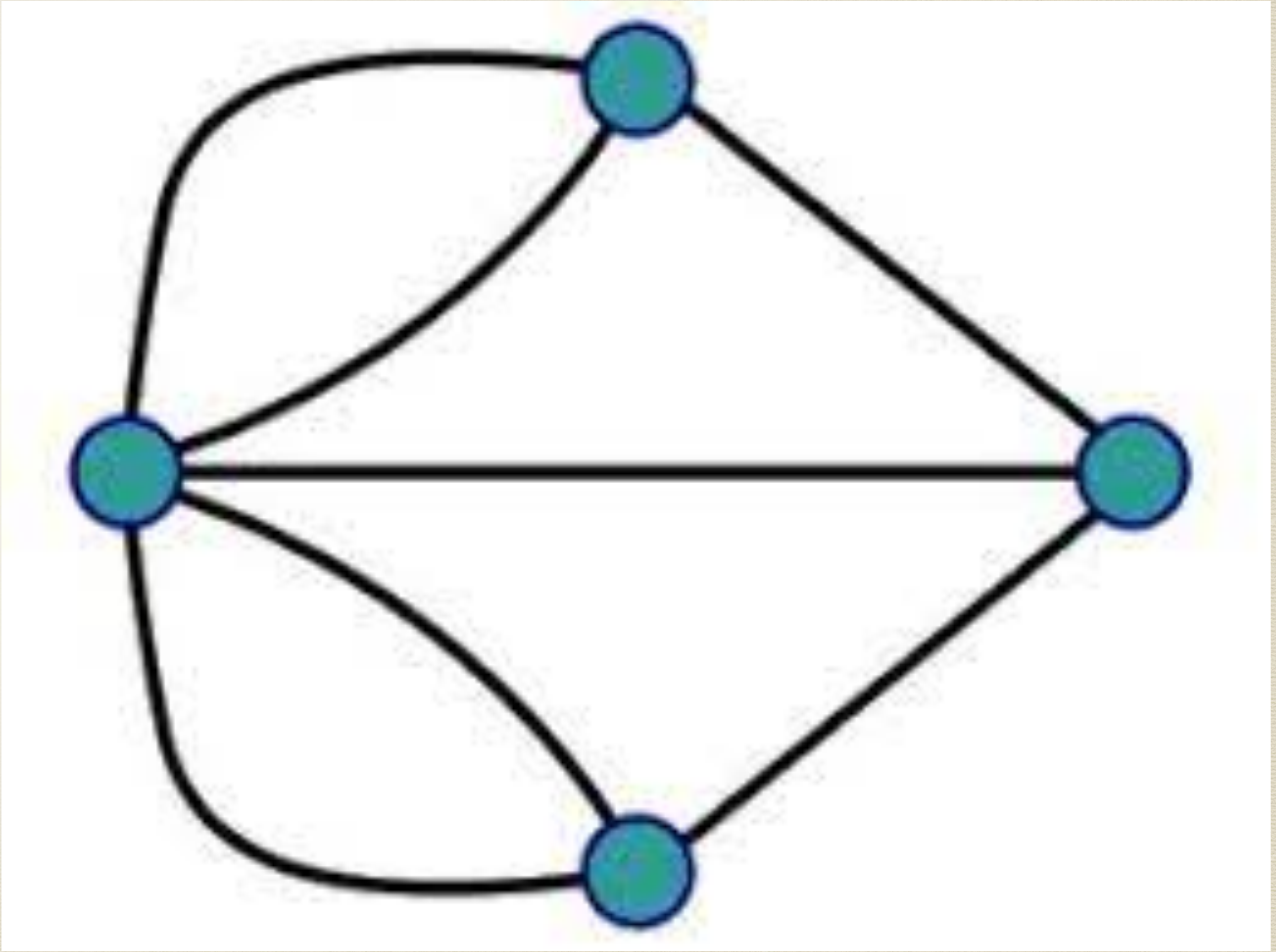


Pontes de Konisberg

* Euler usou um raciocínio muito simples.
* Transformou os caminhos em retas e suas intersecções em pontos.
* Criando possívelmente o primeiro grafo da história.

Pontes de Konisberg





Grafos

* Definição:
  * Grafo G = conjunto finito e não vazio V(G) de objetos chamados vértices juntamente com um conjunto E(G) de pares não ordenados de vértices.
  * E(G) são chamados Arestas.
  * Pode ser chamado de G(V;E) onde V = V(G) e E(G).
* Geralmente grafos são representados por diagramas.

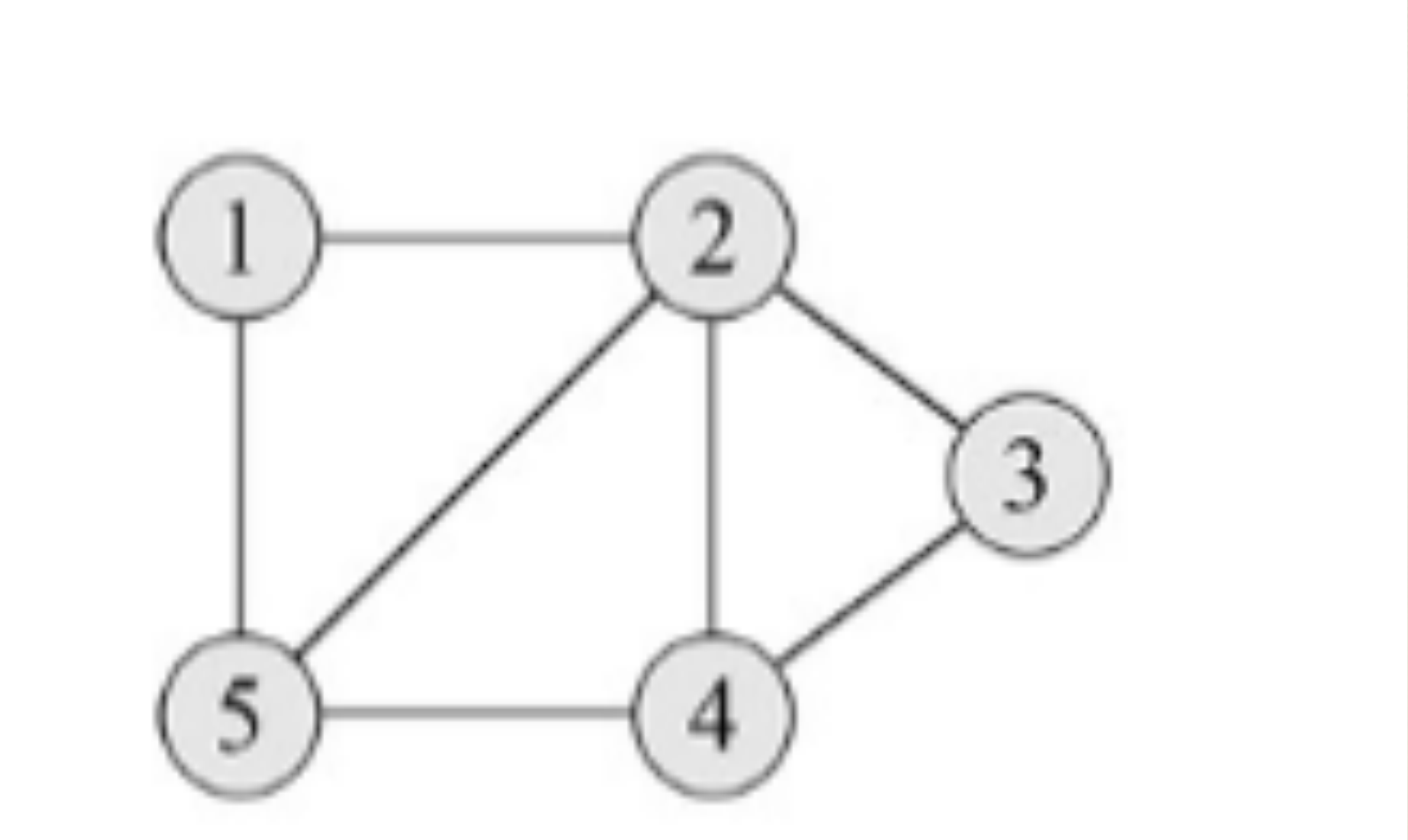
Grafos - Representação computacional

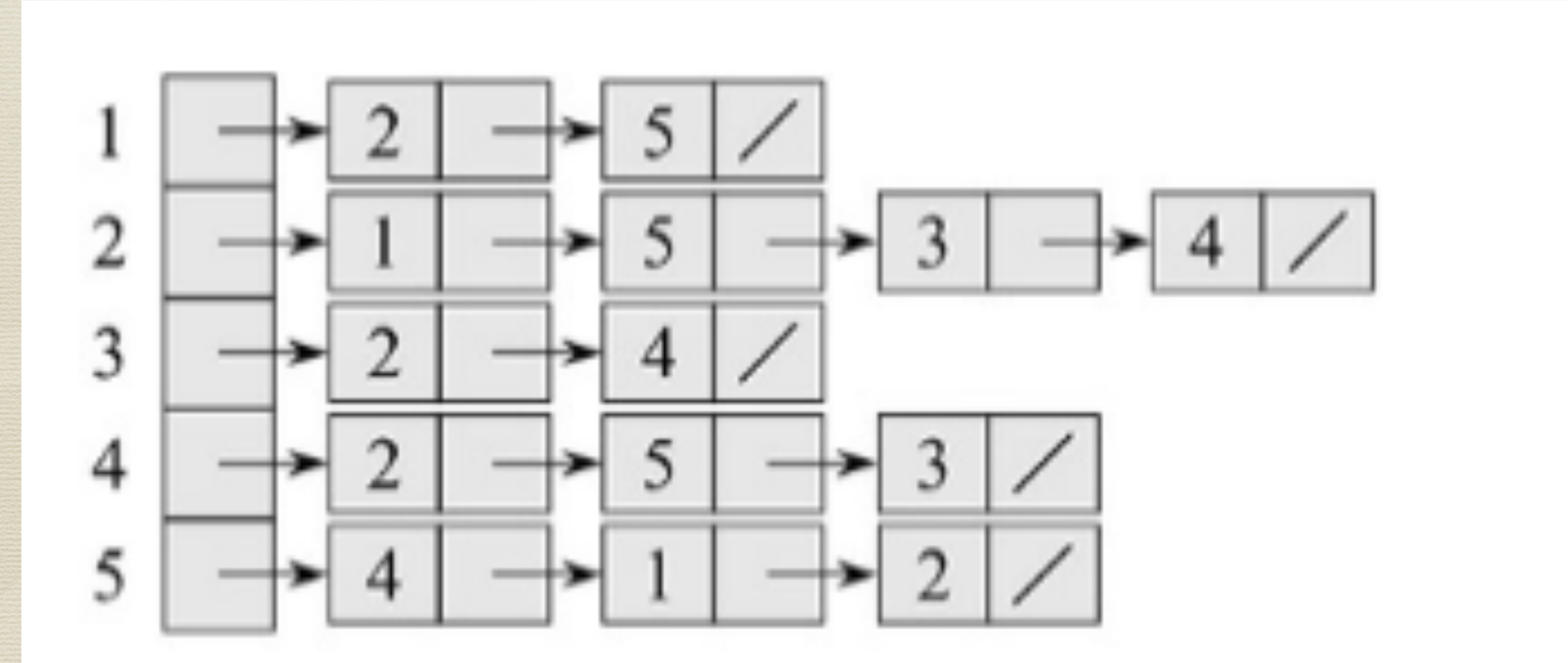
* Podemos escolher entre dois modos padrão para representarmos um grafo $G(V;E)$.
* Como uma lista de adjacências ou como uma matriz de adjacências.
* Qualquer um se aplica a grafos dirigidos e não dirigidos.

Grafos - Listas de Adjacências

* Podemos escolher entre dois modos padrão para representarmos um grafo G(V;E).
* Como uma lista de adjacências ou como uma matriz de adjacências.
* Qualquer um se aplica a grafos dirigidos e não dirigidos.
* Lista de adjacência -> nos dá um modo compacto de armazenar grafos esparsos. (Onde |E| é muito maior que |V^2|). Em geral modo preferido.
* Matriz de adjacências pode ser preferível para os grafos densos. Onde |E| está próximo de |V^2|. Ou quando?

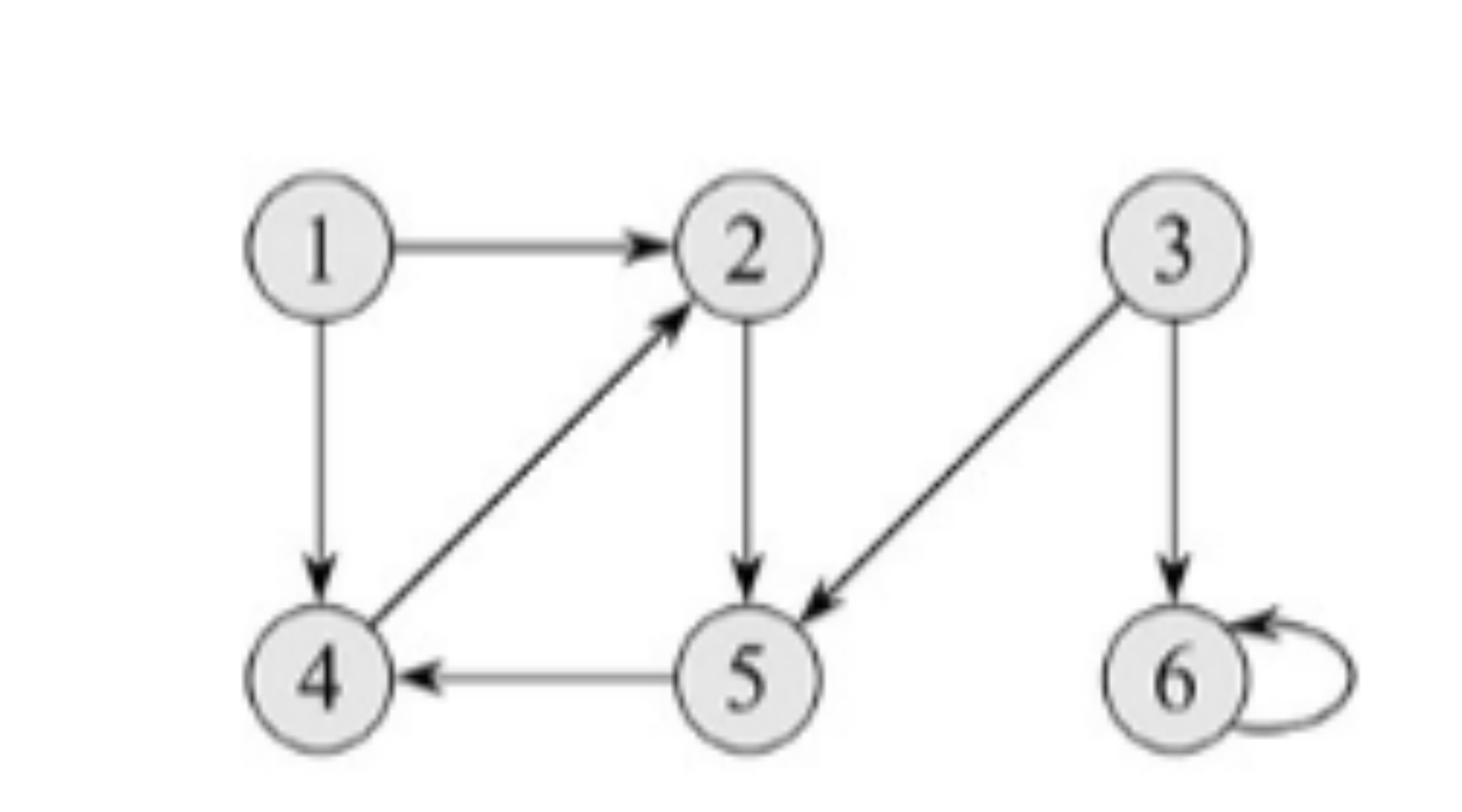
Grafos Não Dirigidos

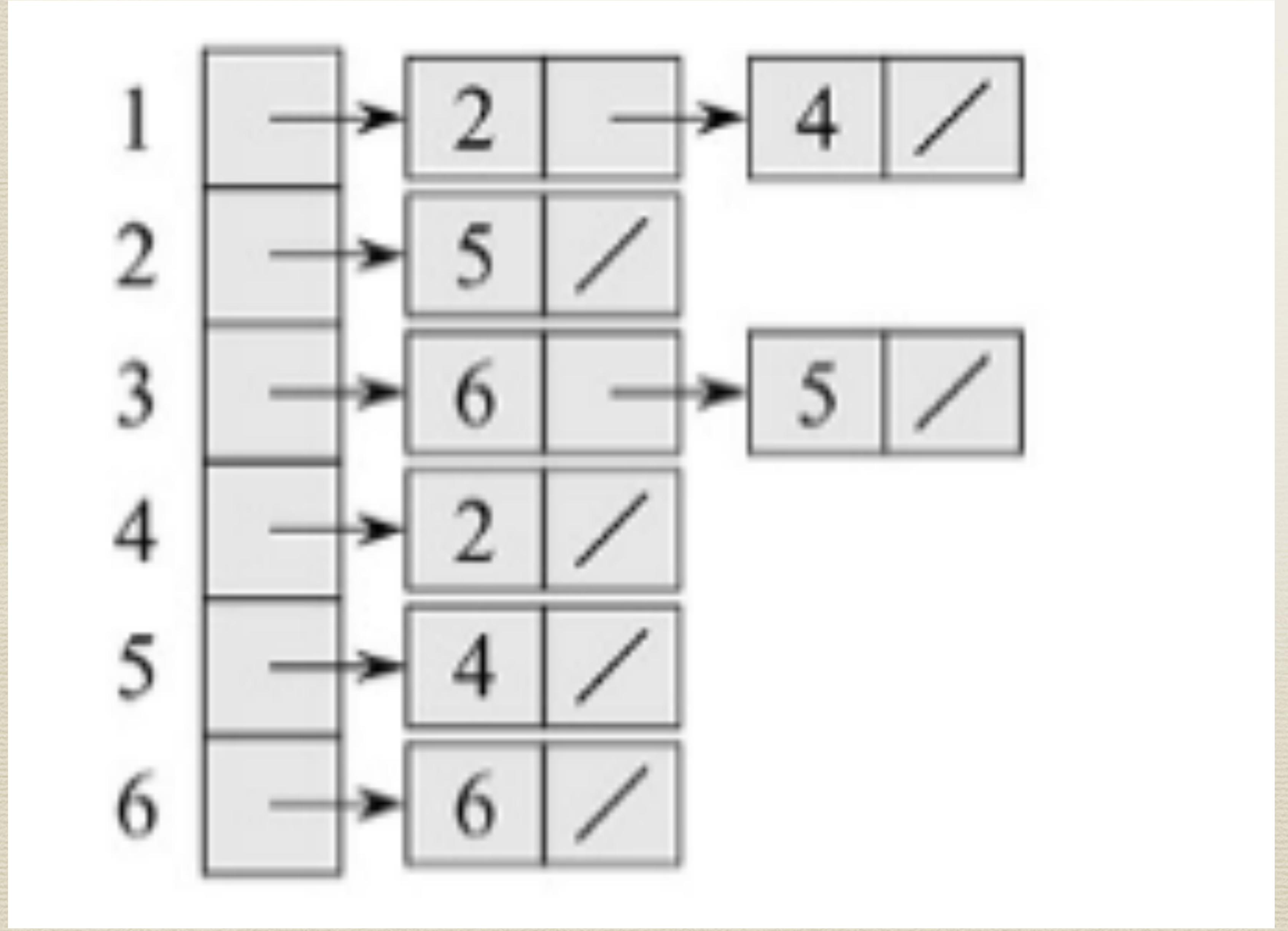




	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

Grafos Dirigidos





	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

Buscas Em Grafos - Busca em Largura

* Um dos algoritmos mais simples para executarmos busca em um grafo.
* Arquétipo de muitos algoritmos importantes.
* Algoritmo de AGM de Prim e Caminhos mínimos de Fonte única de Dijkstra usam idéais semelhantes.

Buscas Em Grafos - Busca em Largura

* Dado um Grafo G(V;E) e uma fonte **s** a busca em largura explora sistematicamente as arestas de de G para “descobrir" cada vértice que pode ser alcançado a partir de s.
* O algoritmo calcula a distancia (menor número de arestas) de s até cada vértice que pode ser alcançado.
* Produz uma árvore de busca em largura com raiz s que contém todos os vértices que podem ser alcançados.

Buscas Em Grafos - Busca em Largura

* Possui esse nome pq expande a fronteira entre vértices descobertos e não descobertos uniformemente ao longo da extensão da fronteira.
* Para controlar o progresso a busca pinta cada vértice de branco , cinza ou preto. No inicio todos os vértices são brancos e depois podem se tornar cinza ou pretos.
* Produz uma árvore de busca em largura com raiz s que contém todos os vértices que podem ser alcançados.

Buscas Em Grafos - Busca em Largura

```
BFS(G, s)
for cada vértice u ∈ V[G] − {s}
    u.cor = BRANCO
    u.d = ∞
    u.π = NIL
s.cor = CINZENTO
s.d = 0
s.π = NIL
Q = ∅
ENQUEUE(Q, s)
while Q ≠ ∅
    u = DEQUEUE(Q)
    for cada v = Adj[u]
        if v.cor == BRANCO
            v.cor == CINZENTO
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.cor = PRETO
```

Buscas Em Grafos - Busca em Largura

```
PRINT-PATH(G, s, v)
    if v == s
        imprimir s
    else if v.π = NIL
        imprimir "não existe nenhum caminho de" s "para v"
    else PRINT-PATH(G, s, v.π)
        imprimir v
```

Buscas Em Grafos - Busca em Profundidade

* Busca mais fundo no grafo, sempre que possível.
* Utiliza pilhas ao invés de filas.

Buscas Em Grafos - Busca em Largura

```
DFS(G)
    for cada vértice u ∈ V[G]
        u.cor = BRANCO
        u.π = NIL
    tempo = 0
    for cada vértice u ∈ V[G]
        if u.cor == BRANCO
            DFS-VISIT(G, u)
```

```
DFS-VISIT(G, u)
    tempo = tempo + 1                 // vértice branco u acabou de ser descoberto
    u.d = tempo
    u.cor = CINZENTO
    for cada v ∈ G.Adj[u]             // explorar aresta (u, v)
        if v.cor == BRANCO
            v.π = u
            DFS-VISIT(G, v)
    u.cor = PRETO                     // pintar u de preto; está terminado
    tempo = tempo + 1
    u.f = tempo
```

Dúvidas?

"Stay hungry, stay foolish."

–Steve Jobs