Algoritmos e Estrutura de Dados II

Aula 15

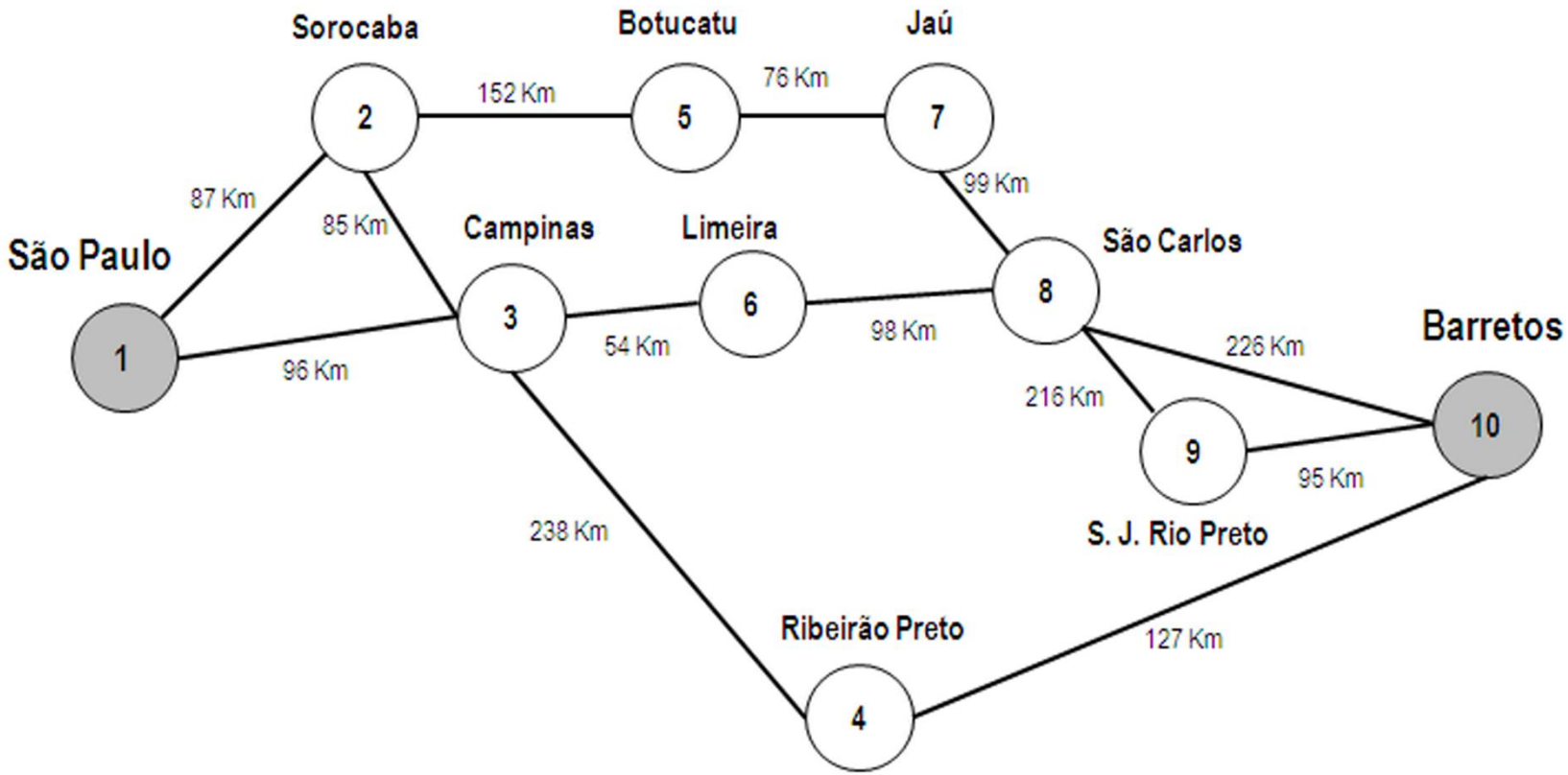
Grafos

Prof. Dr. Dilermando Piva Jr

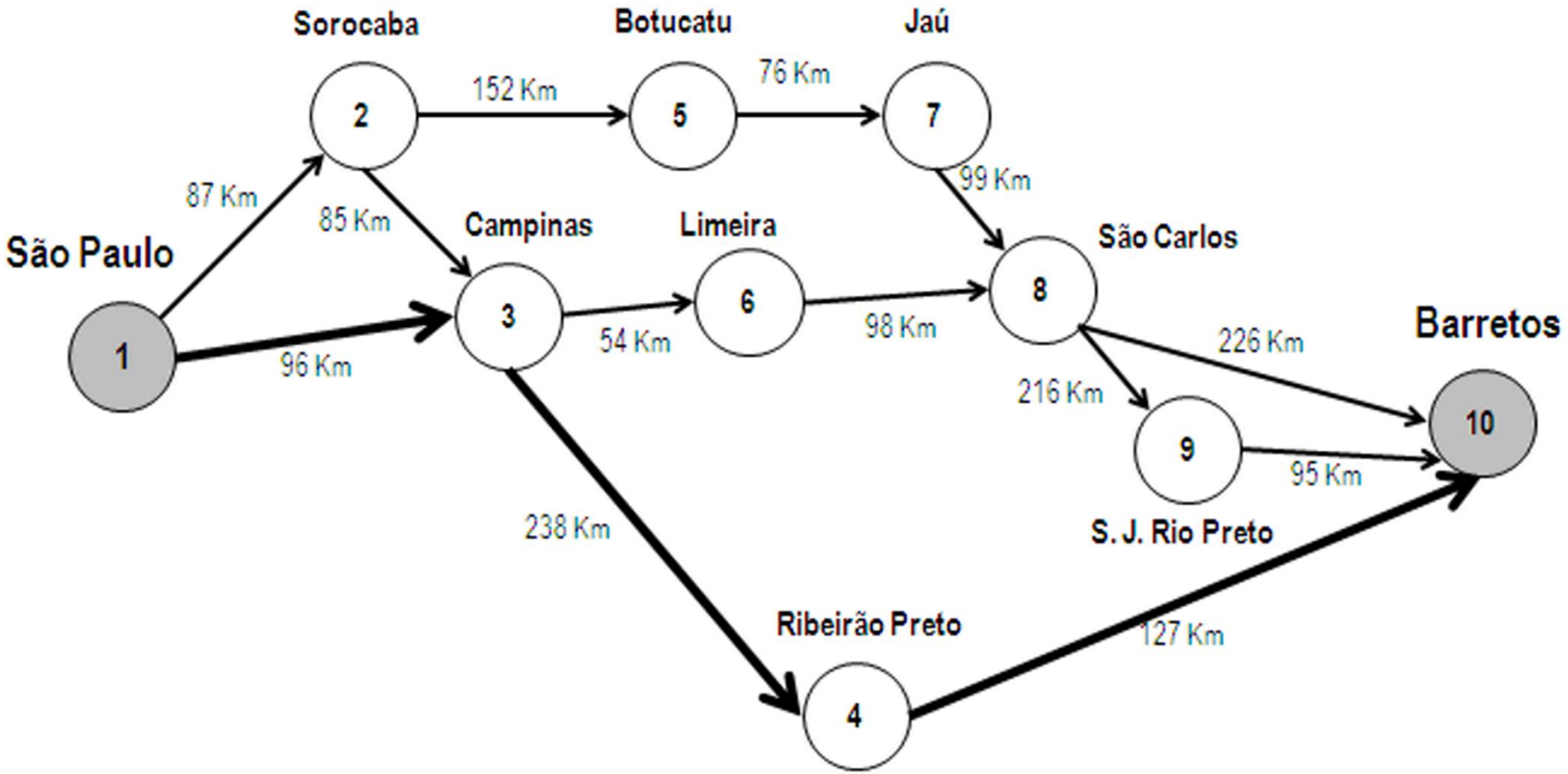
2º Semestre - CDN

Qual caminho devo seguir?

Qual o melhor caminho de São Paulo para Barretos?



Qual o melhor caminho de São Paulo para Barretos?

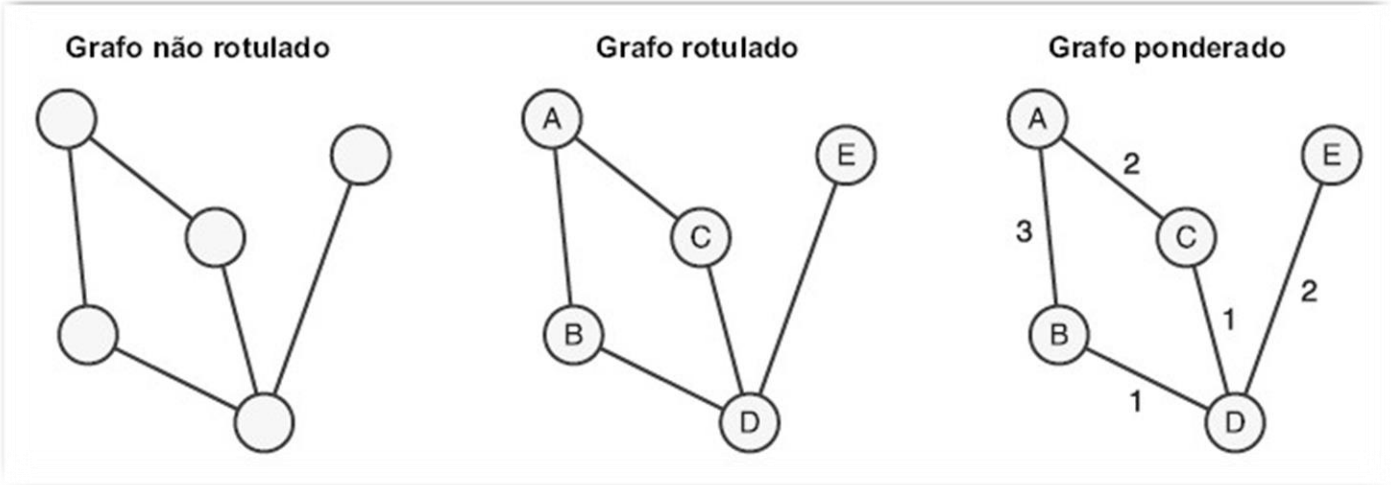


Algumas aplicações dos Grafos...

- Um roteiro.
- Um mapa de rotas aéreas.
- Um layout de um mundo de jogo de aventura.
- Um esquema dos computadores e das conexões que compõem a internet.
- A relação entre alunos e cursos.
- Um diagrama das capacidades de fluxo em uma rede de comunicações ou transporte.
- Redes de distribuição (eletricidade)
- Distribuição de tarefas
- Isômeros Químicos
- Planejamento de tarefas
- Alinhamento de sequências biológicas (DNA → cgat)
- Problemas de caminho mais curto (gps)
- ...

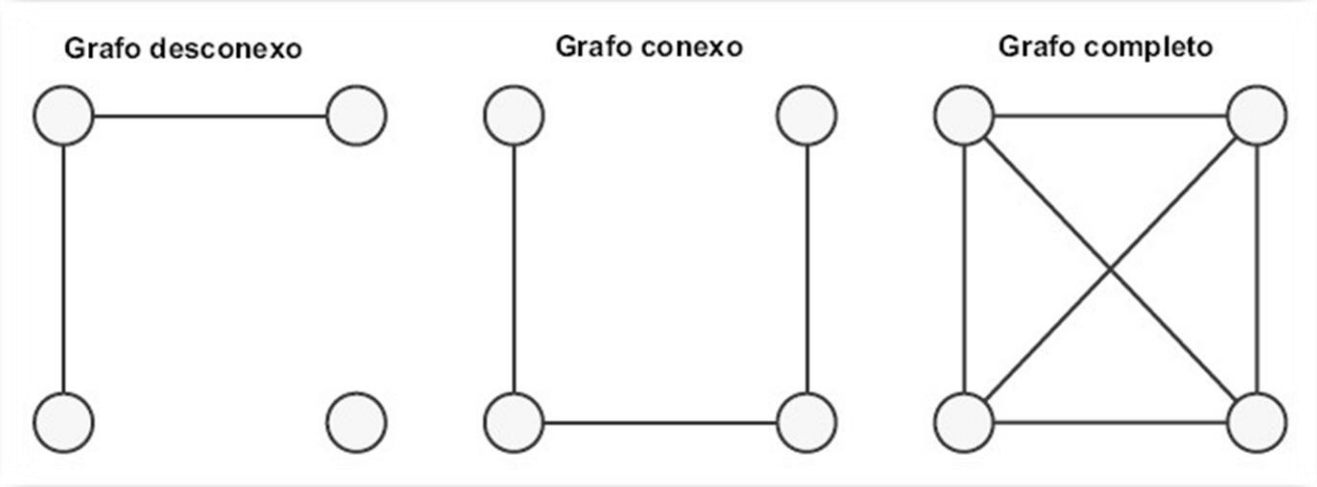
Terminologia

- Um grafo é um conjunto **V** de vértices (ou nós) e um conjunto **E** de arestas:
  - Cada aresta em E conecta dois dos vértices em V.
- Nó
  - Usado como sinônimo de vértice.
- Vértices e arestas podem ser rotulados ou não rotulados:
  - Quando as arestas são rotuladas com números, os números podem ser vistos como pesos, e o grafo é considerado um grafo ponderado.



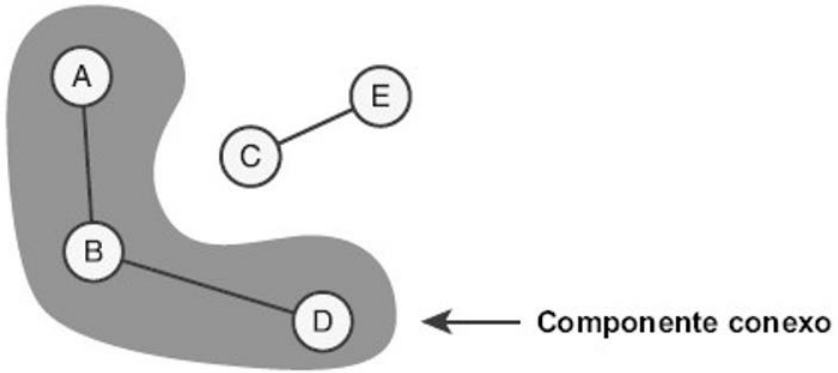
Terminologia

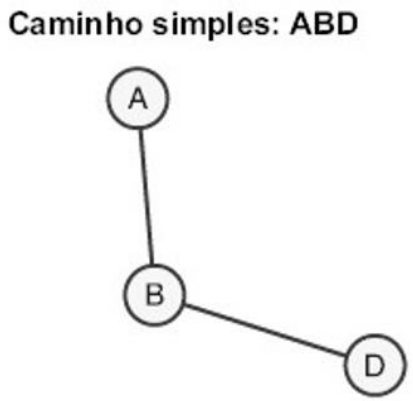
- Um vértice é adjacente a outro vértice se houver uma aresta conectando os dois vértices:
  - Esses dois vértices também são chamados vizinhos.
- Um caminho é uma sequência de arestas que permite que um vértice seja alcançado a partir de outro vértice em um grafo.
- Um vértice pode ser alcançável a partir de outro vértice se e somente se houver um caminho entre os dois:
  - O comprimento de um caminho é o número de arestas no caminho.
- Um grafo é **conexo** se houver um caminho de cada vértice a todos os outros vértices.
- Um grafo está **completo** se houver uma aresta de cada vértice para todos os outros vértices.

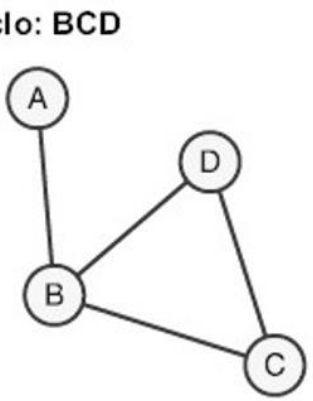


Terminologia

- O **grau de um vértice** é igual ao número de arestas conectadas a ele.
- O **subgrafo** de um grafo consiste em um subconjunto dos vértices desse grafo e nas arestas que conectam esses vértices.
- Um **componente conexo** é um subgrafo que consiste no conjunto de vértices que são alcançáveis a partir de determinado vértice.
- Um **caminho simples** é aquele que não passa pelo mesmo vértice mais de uma vez.
- Um ciclo é um caminho que começa e termina no mesmo vértice.

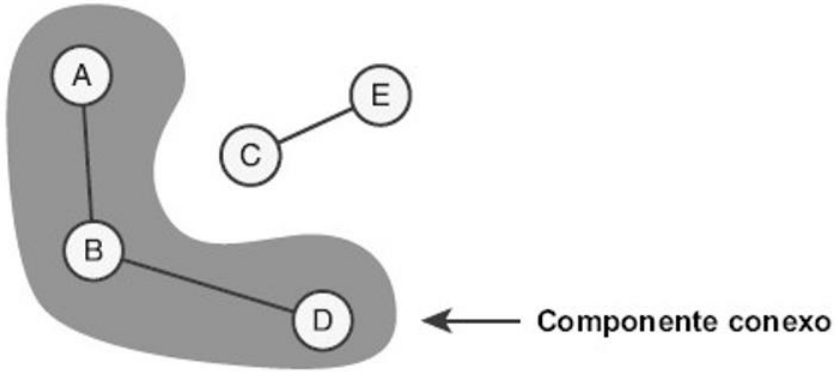


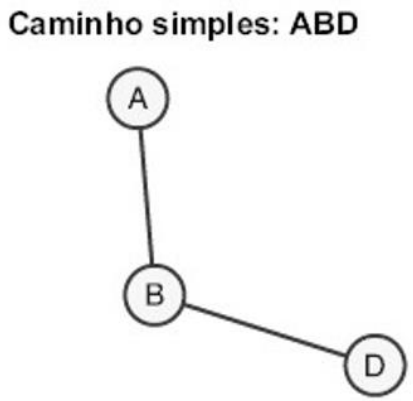


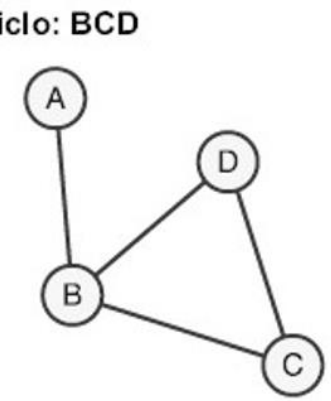


Terminologia

- O **grau de um vértice** é igual ao número de arestas conectadas a ele.
- O **subgrafo** de um grafo consiste em um subconjunto dos vértices desse grafo e nas arestas que conectam esses vértices.
- Um **componente conexo** é um subgrafo que consiste no conjunto de vértices que são alcançáveis a partir de determinado vértice.
- Um **caminho simples** é aquele que não passa pelo mesmo vértice mais de uma vez.
- Um ciclo é um caminho que começa e termina no mesmo vértice.

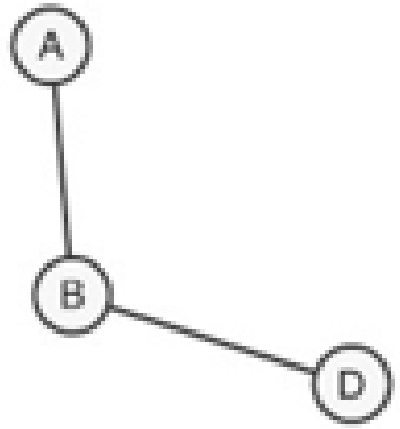




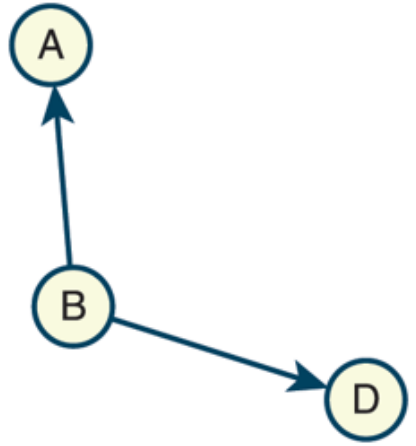


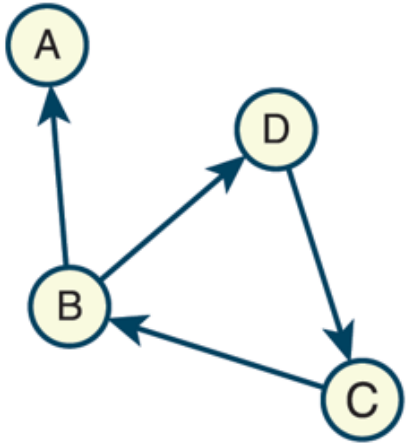
Terminologia

- Nos grafos não direcionados, suas arestas não indicam direção.
- As arestas em um grafo direcionado, ou **dígrafo**, especificam uma direção explícita.



Grafo não direcionado

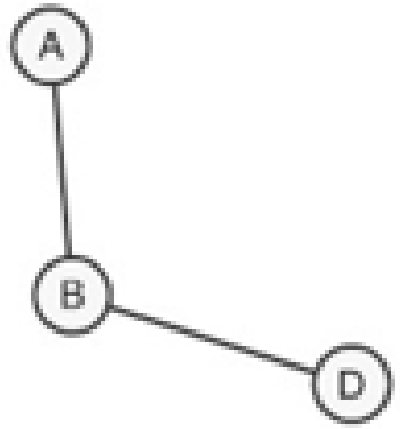




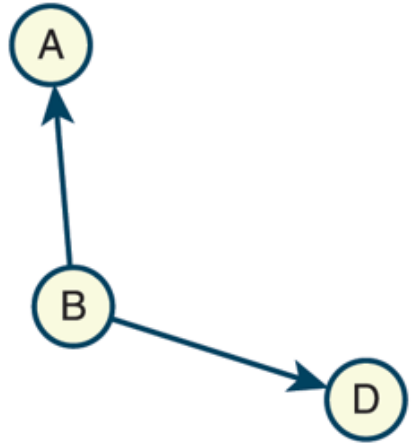
Grafos direcionados

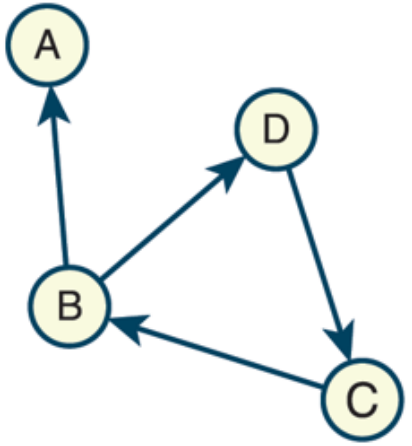
Terminologia

- Nos grafos não direcionados, suas arestas não indicam direção.
- As arestas em um grafo direcionado, ou **dígrafo**, especificam uma direção explícita.



Grafo não direcionado

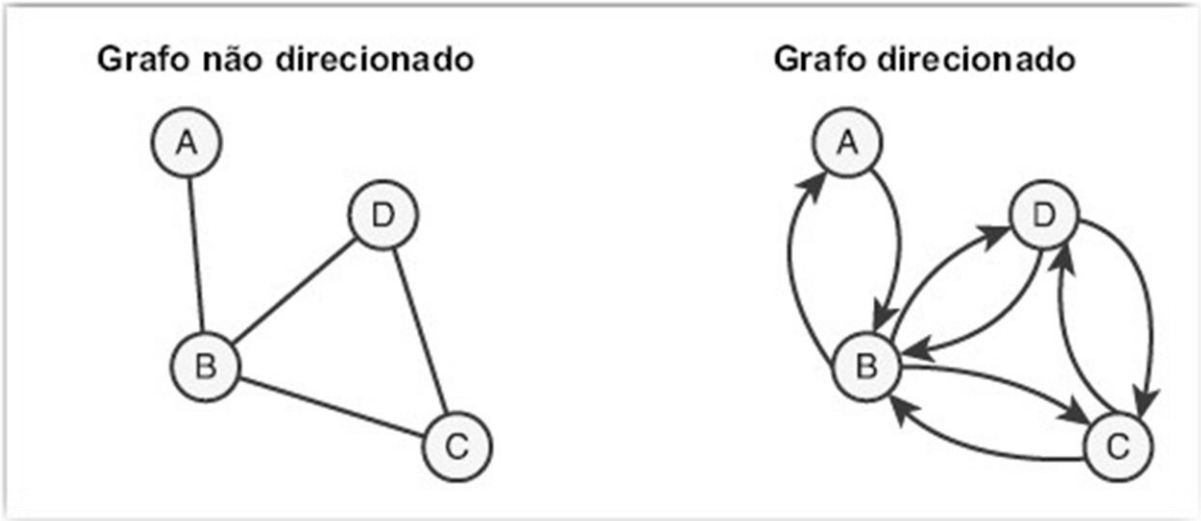




Grafos direcionados

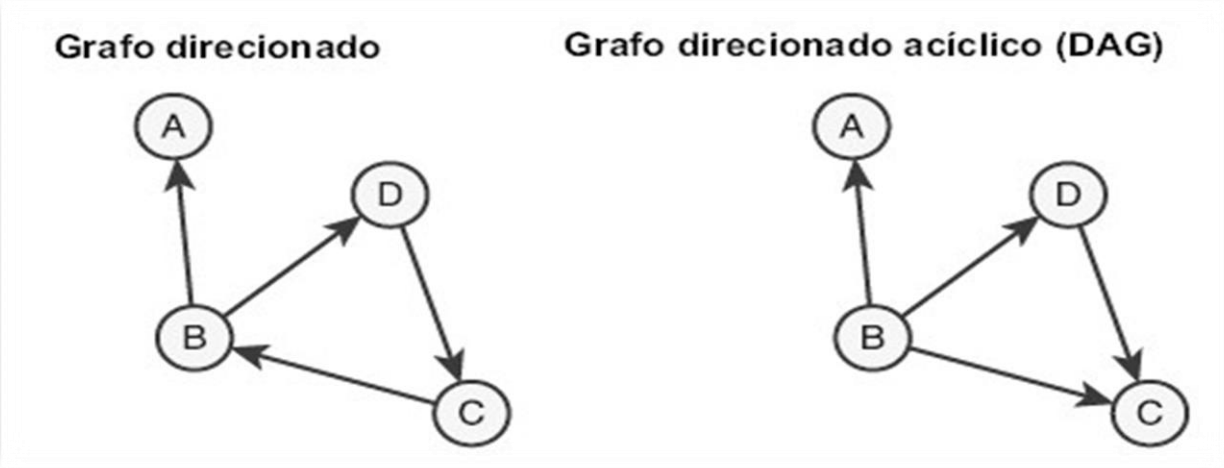
Terminologia

- Cada aresta em um dígrafo chama-se **aresta direcionada**
  - Ela tem um **vértice de origem** e um **vértice de destino.**
- Quando há apenas uma aresta direcionada conectando dois vértices, os vértices estão na relação de predecessor (o vértice de origem) e sucessor (o vértice de destino):
  - A relação de adjacência entre eles é assimétrica.
- As arestas que emanam de determinado vértice de origem são chamadas **arestas incidentes.**



Terminologia

- **Grafo acíclico direcionado**
  - Um caso especial de dígrafo que não contém ciclos.
- Listas e árvores são casos especiais de grafos direcionados.
- **Grafo denso**
  - Um grafo conexo que tem relativamente muitas arestas.
- **Grafo Esparso**
  - Um garfo que tem relativamente poucas arestas.



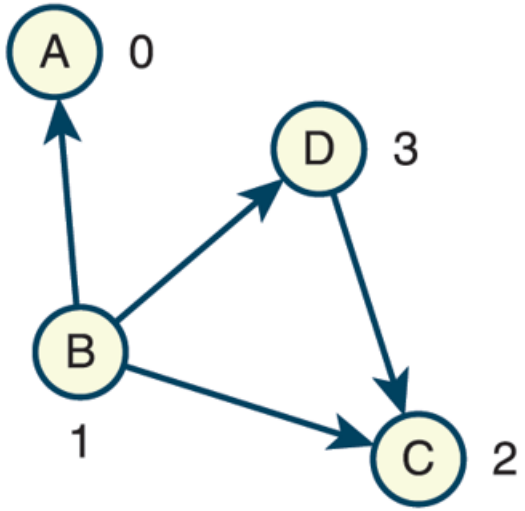
Representação dos grafos

- Para representar grafos,
  - você precisa de uma maneira conveniente de armazenar os vértices e as arestas que os conectam.
- As duas representações de grafos comumente usadas são a **matriz de adjacência** e a **lista de adjacências.**

Representação dos grafos

- Suponha que um grafo tenha *N* vértices marcados com 0, 1, . . ., *N* – 1 e então o seguinte se aplica:
  - A **matriz de adjacência** para o grafo é uma grade *G* com *N* filas e *N* colunas.
  - A célula *G[i] [j]* contém 1 se houver uma aresta do vértice *i* ao vértice *j* no grafo. Do contrário, não há aresta e essa célula contém 0.

Grafo
Direcionado

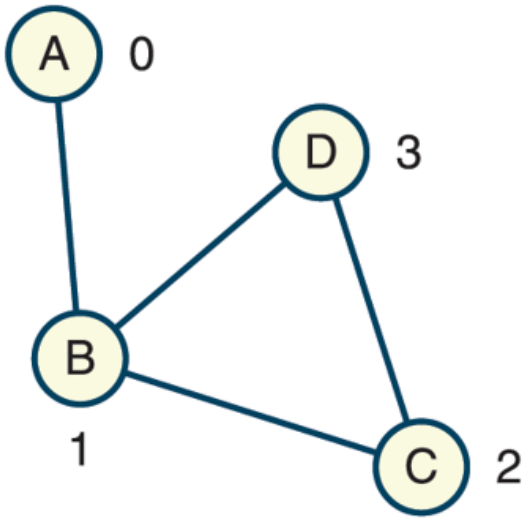


		0 A	1 B	2 C	3 D
0	A	0	0	0	0
1	B	1	0	1	1
2	C	0	0	0	0
3	D	0	0	1	0

Representação dos grafos

- Suponha que um grafo tenha *N* vértices marcados com 0, 1, . . ., *N* – 1 e então o seguinte se aplica:
  - A **matriz de adjacência** para o grafo é uma grade *G* com *N* filas e *N* colunas.
  - A célula *G[i] [j]* contém 1 se houver uma aresta do vértice *i* ao vértice *j* no grafo. Do contrário, não há aresta e essa célula contém 0.

Grafo
Não-Direcionado

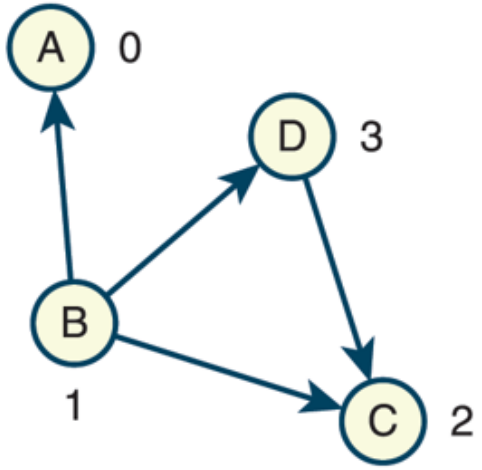


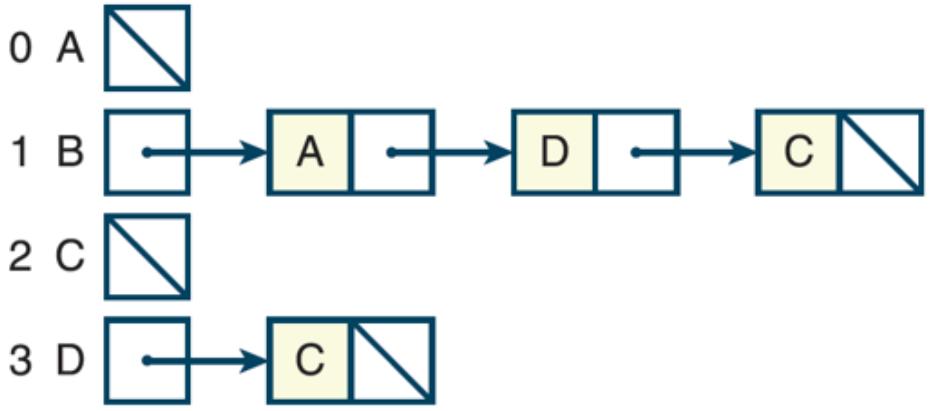
		0 A	1 B	2 C	3 D
0	A	0	1	0	0
1	B	1	0	1	1
2	C	0	1	0	1
3	D	0	1	1	0

Representação dos grafos

- Suponha que um grafo tenha *N* vértices marcados com 0, 1, . . ., *N* – 1 e então o seguinte se aplica:
  - A **lista de adjacências** para o grafo é um array de *N* listas ligadas.
  - A *i-ésima* lista encadeada contém um nó para o vértice *j* se e somente se houver uma aresta do vértice *i* ao vértice *j.*

Grafo
Direcionado

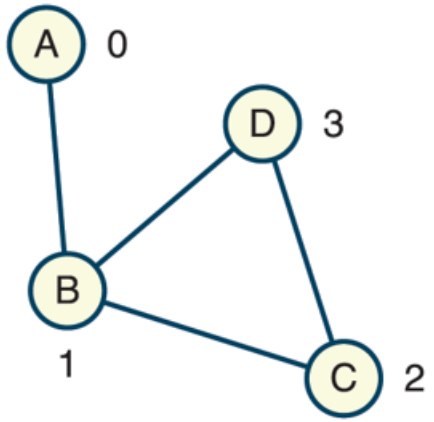


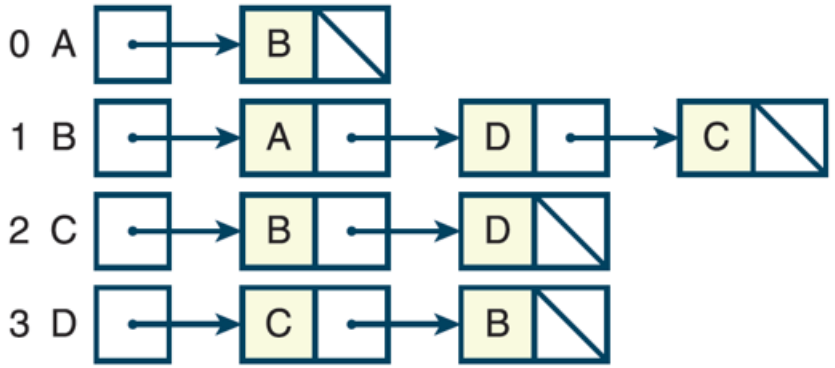


Representação dos grafos

- Suponha que um grafo tenha *N* vértices marcados com 0, 1, . . ., *N* – 1 e então o seguinte se aplica:
  - A **lista de adjacências** para o grafo é um array de *N* listas ligadas.
  - A *i-ésima* lista encadeada contém um nó para o vértice *j* se e somente se houver uma aresta do vértice *i* ao vértice *j.*

Grafo
Não-Direcionado





Principais operações com grafos: Percursos

Importantes operações de processamento de grafo incluem:

- Encontrar o caminho mais curto para determinado item em um grafo.
- Encontrar todos os itens aos quais determinado item está conectado por caminhos.
- Percorrer todos os itens em um grafo.

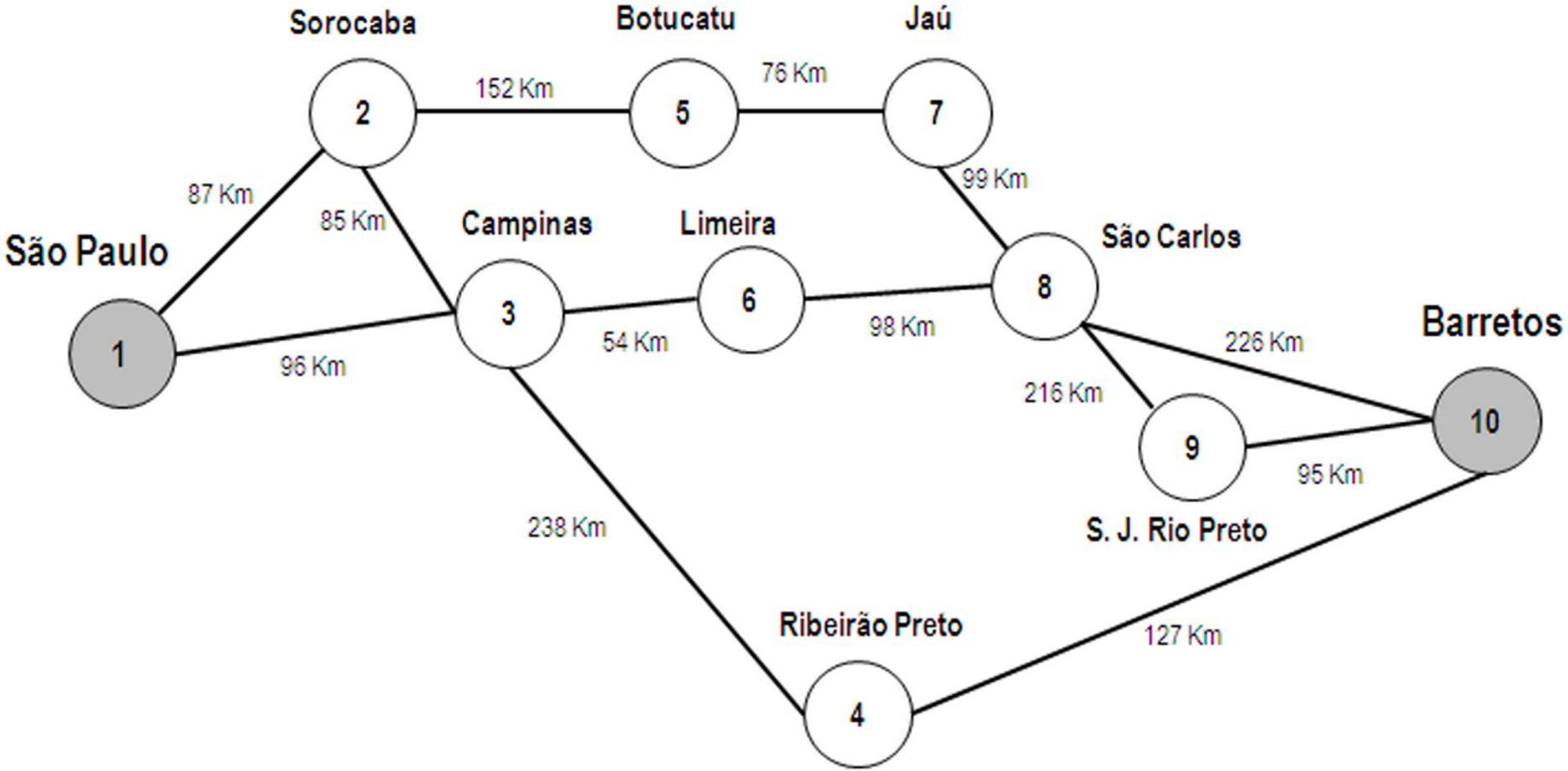
Definição das classes para construir um Grafo...

```
# Classe Vertice ou Nó
class Vertice:
    def __init__(self, rotulo):
        self.rotulo = rotulo
        self.visitado = False
        self.adjacentes = []

    def adiciona_adjacente(self, adjacente):
        self.adjacentes.append(adjacente)

    def mostra_adjacentes(self):
        for i in self.adjacentes:
        print(i.vertice.rotulo, i.custo)
```

Definição das classes para construir um Grafo...

```
# Definição da classe Adjacente ou Aresta
class Adjacente:
    def __init__(self, vertice, custo):
        self.vertice = vertice
        self.custo = custo
```

Definição das classes para construir um Grafo...



Definição das classes para construir um Grafo...

```
# Criação de uma classe Grafo (definindo suas relações)
class Grafo:
    sorocaba = Vertice('Sorocaba')
    saopaulo = Vertice('São Paulo')
    campinas = Vertice('Campinas')
    botucatu = Vertice('Botucatu')
    limeira = Vertice('Limeira')
    ribeiraopreto = Vertice('Ribeirão Preto')
    jau = Vertice('Jaú')
    saocarlos = Vertice('São Carlos')
    sjriopreto = Vertice('S. J. Rio Preto')
    barretos = Vertice('Barretos')
```

Definição das classes para construir um Grafo...

```
# Criação de uma classe Grafo (definindo suas relações)
class Grafo:
...
    sorocaba.adiciona_adjacente(Adjacente(saopaulo, 87))
    sorocaba.adiciona_adjacente(Adjacente(campinas, 85))
    sorocaba.adiciona_adjacente(Adjacente(botucatu, 152))

    saopaulo.adiciona_adjacente(Adjacente(sorocaba, 87))
    saopaulo.adiciona_adjacente(Adjacente(campinas, 95))

    campinas.adiciona_adjacente(Adjacente(saopaulo, 95))
    campinas.adiciona_adjacente(Adjacente(sorocaba, 85))
    campinas.adiciona_adjacente(Adjacente(limeira, 54))
    campinas.adiciona_adjacente(Adjacente(ribeiraopreto, 238))

    botucatu.adiciona_adjacente(Adjacente(sorocaba, 152))
    botucatu.adiciona_adjacente(Adjacente(jau, 76))

    limeira.adiciona_adjacente(Adjacente(campinas, 54))
    limeira.adiciona_adjacente(Adjacente(saocarlos, 98))
```

Definição das classes para construir um Grafo...

```
# Criação de uma classe Grafo (definindo suas relações)
class Grafo:
...

    ribeiraopreto.adiciona_adjacente(Adjacente(campinas, 238))
    ribeiraopreto.adiciona_adjacente(Adjacente(barretos, 127))

    jau.adiciona_adjacente(Adjacente(botucatu, 76))
    jau.adiciona_adjacente(Adjacente(saocarlos, 99))

    saocarlos.adiciona_adjacente(Adjacente(jau, 99))
    saocarlos.adiciona_adjacente(Adjacente(limeira, 98))
    saocarlos.adiciona_adjacente(Adjacente(barretos, 226))
    saocarlos.adiciona_adjacente(Adjacente(sjriopreto, 216))

    sjriopreto.adiciona_adjacente(Adjacente(saocarlos, 216))
    sjriopreto.adiciona_adjacente(Adjacente(barretos, 95))

    barretos.adiciona_adjacente(Adjacente(saocarlos, 226))
    barretos.adiciona_adjacente(Adjacente(ribeiraopreto, 127))
    barretos.adiciona_adjacente(Adjacente(sjriopreto, 95))
```

Definição das classes para construir um Grafo...

```
grafo = Grafo()
```

```
grafo.saopaulo.mostra_adjacentes()
```

```
Sorocaba 87
Campinas 95
```

```
grafo.sorocaba.mostra_adjacentes()
```

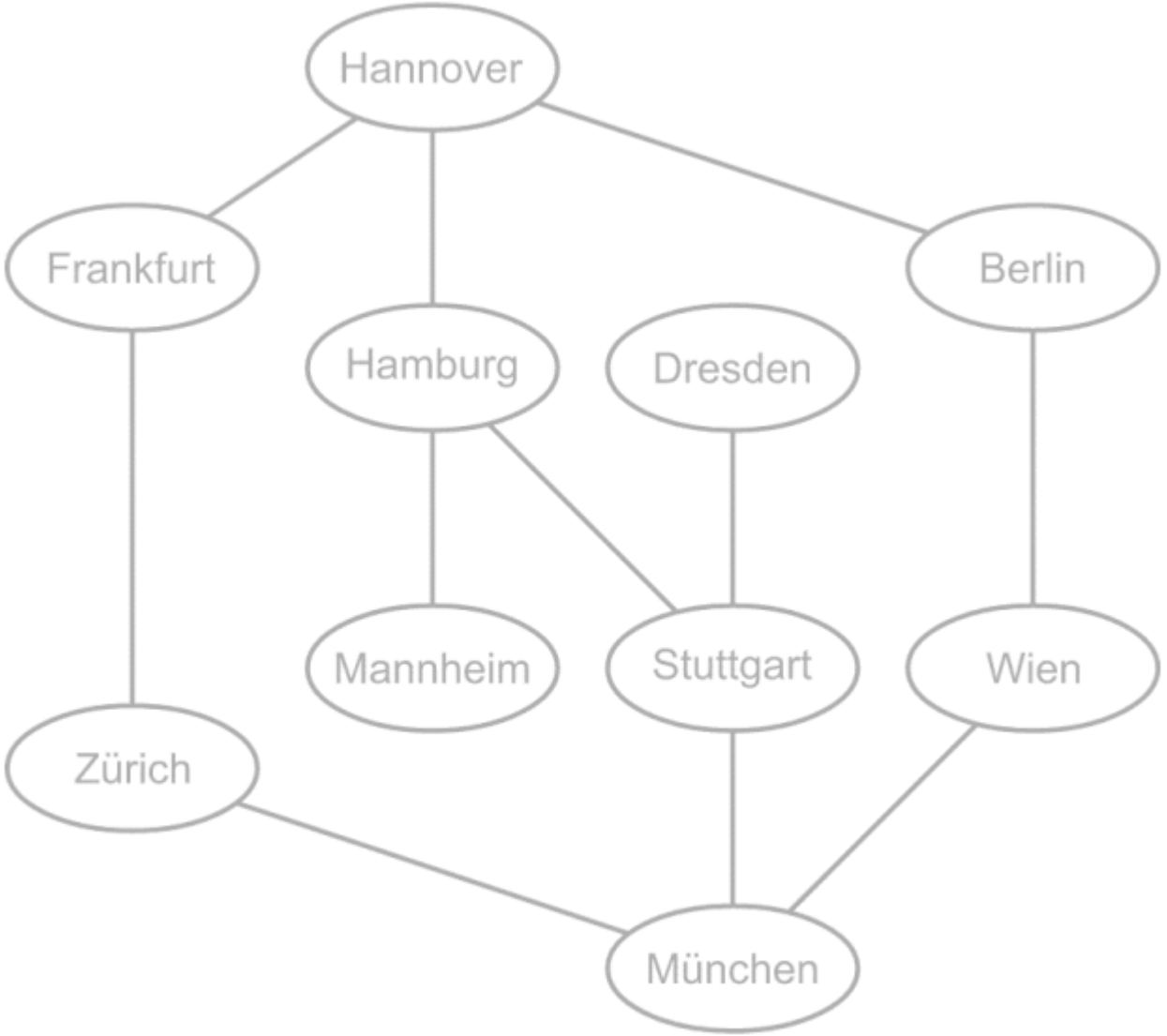
```
São Paulo 87
Campinas 85
Botucatu 152
```

VAMOS PARA A PRÁTICA ?!!!

BUSCA EM PROFUNDIDADE EM GRAFOS
Depth-First Search - DFS

Ideia geral: a cada vértice descoberto, explorar um de seus vizinhos não visitados (sempre que possível).

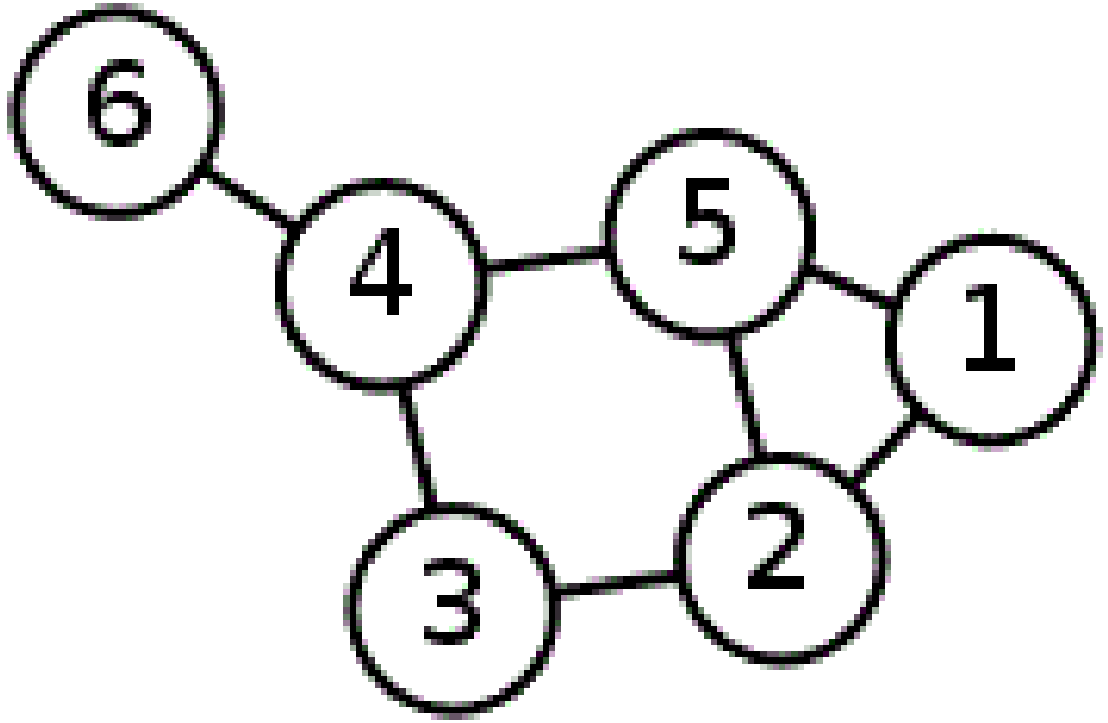
Imita exploração de labirinto, aprofundando sempre que possível.



VAMOS PARA A PRÁTICA ?!!!

BUSCA EM LARGURA EM GRAFOS
Breadth-First Search - BFS

Ideia geral: a cada novo nível descoberto, todos os vértices daquele nível devem ser visitados antes de prosseguir para o próximo nível.



VAMOS PARA A PRÁTICA ?!!!