GRAFOS

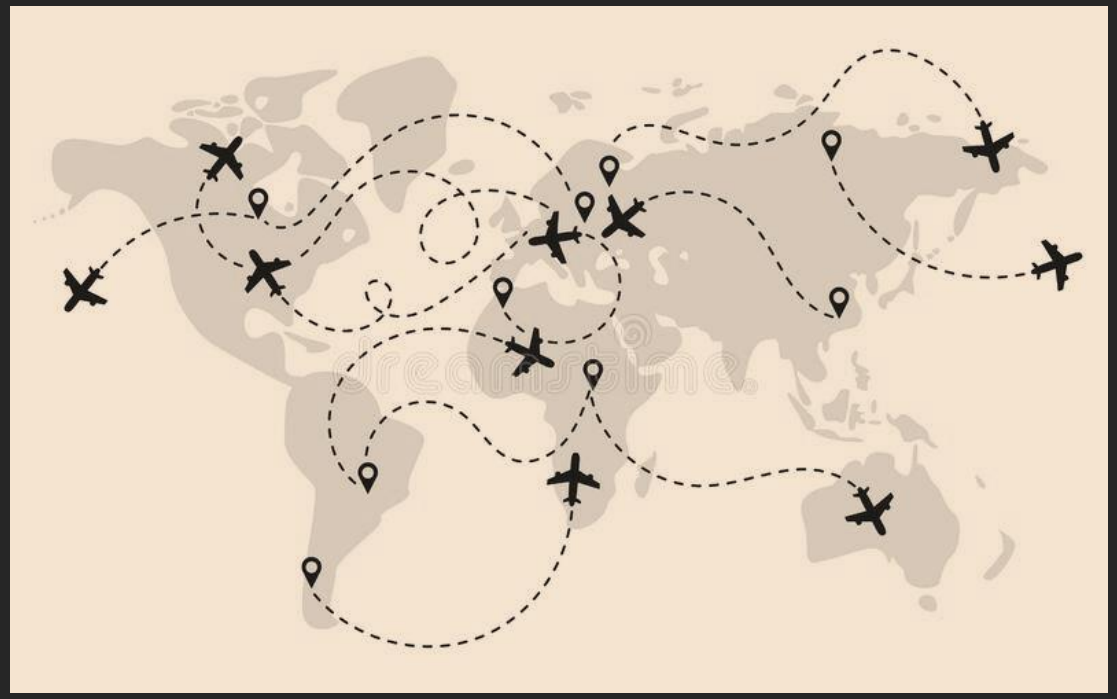
DISCIPLINA DE ESTRUTURA DE DADOS

PROF. DOUGLAS GC

INSTITUTO FEDERAL
Maranhão
Campus Viana

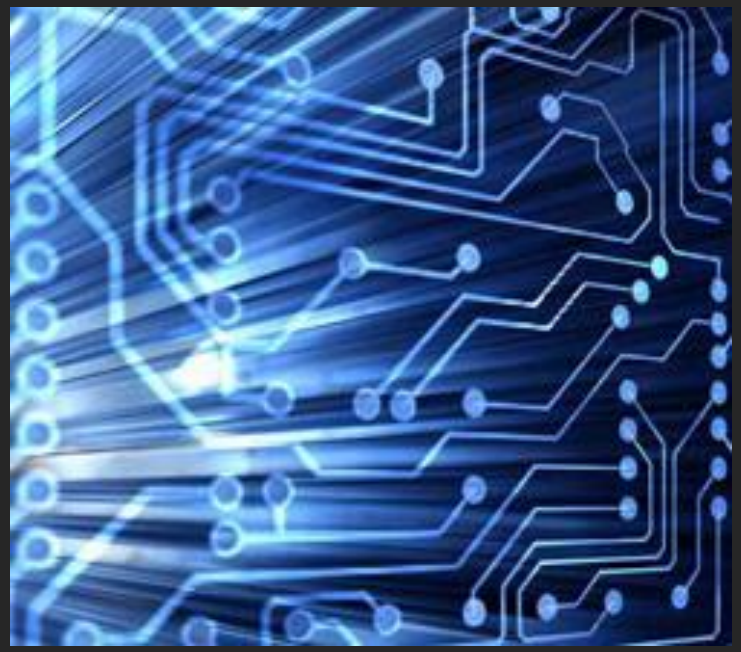
Planejamento de Rotas: sair da cidade A e chegar na Cidade Z.

APLICAÇÕES PRÁTICAS DE GRAFOS



Mapa de tráfego aéreo.

APLICAÇÕES PRÁTICAS DE GRAFOS



Rotas de circuitos eletrônicos.

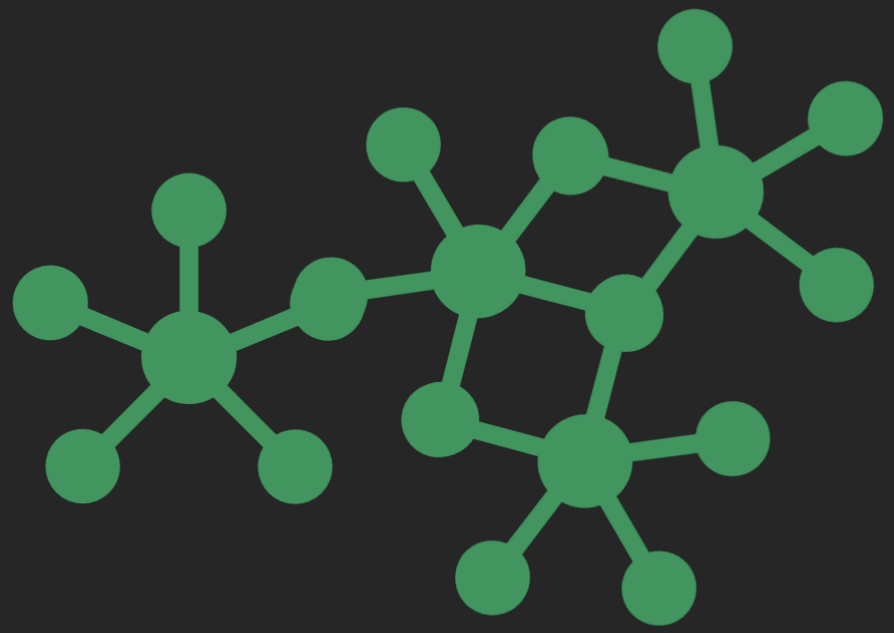
APLICAÇÕES PRÁTICAS DE GRAFOS

Rede de computadores

APLICAÇÕES PRÁTICAS DE GRAFOS

- Redes sociais: descobrir o usuário mais influente em uma rede social;
- Rotas de estação do sistema de metrô.
- Pontes de Konigsberg;

APLICAÇÕES PRÁTICAS DE GRAFOS

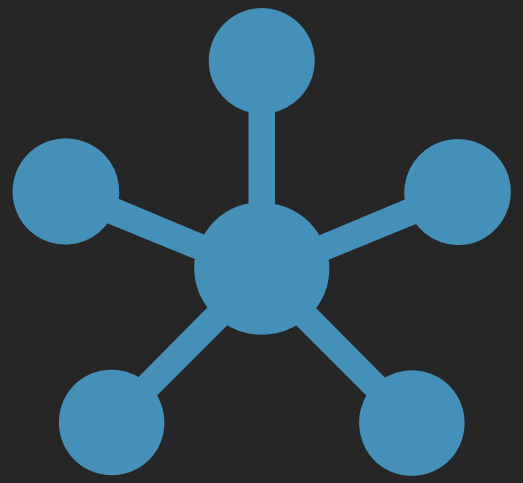


Nó: um nó (vértice/ponto) representa algum objeto do mundo real;

Aresta: uma aresta faz a ligação entre os nós;

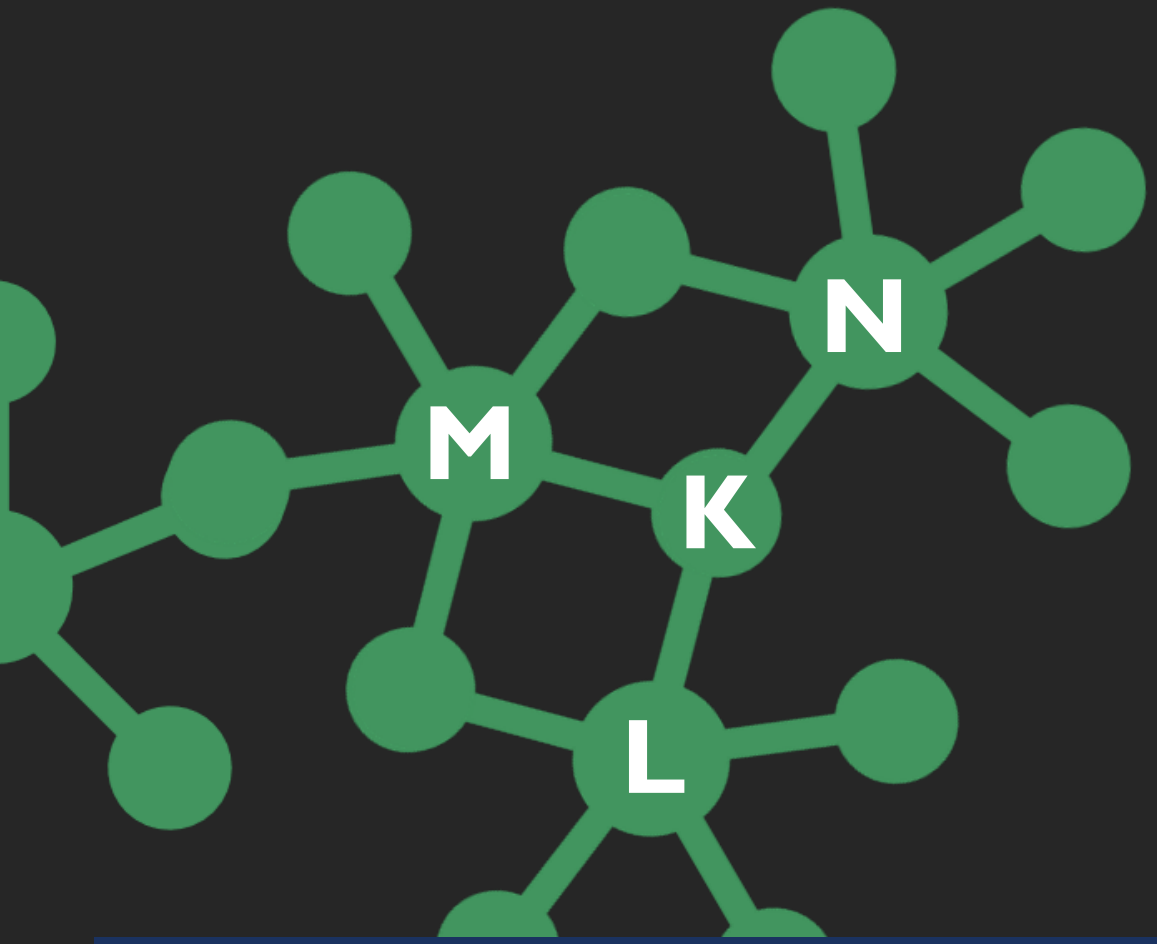
APRESENTAÇÃO GRÁFICA

DEFINIÇÃO CONCEITUAL



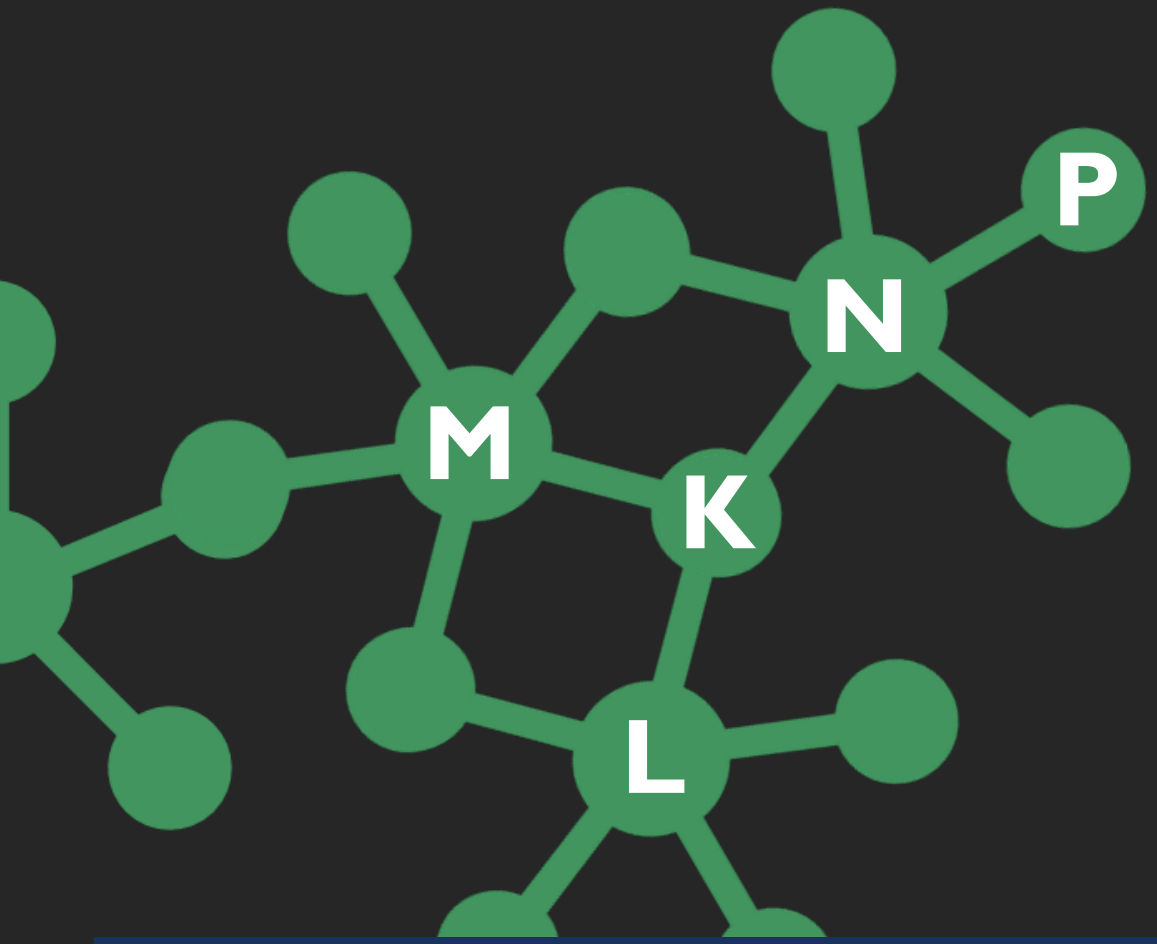
Grafo

[...] é uma forma de representar relacionamentos que existem em pares de objetos. Isto é, um conjunto de objetos, chamados de vértices, juntamente com uma coleção de conexões entre pares de vértices. [...] Grafos têm aplicações em vários domínios diferentes, incluindo mapeamento, transporte, engenharia elétrica e redes de computadores. (GOODRISH e TAMASSIA, 2013, p. 598)



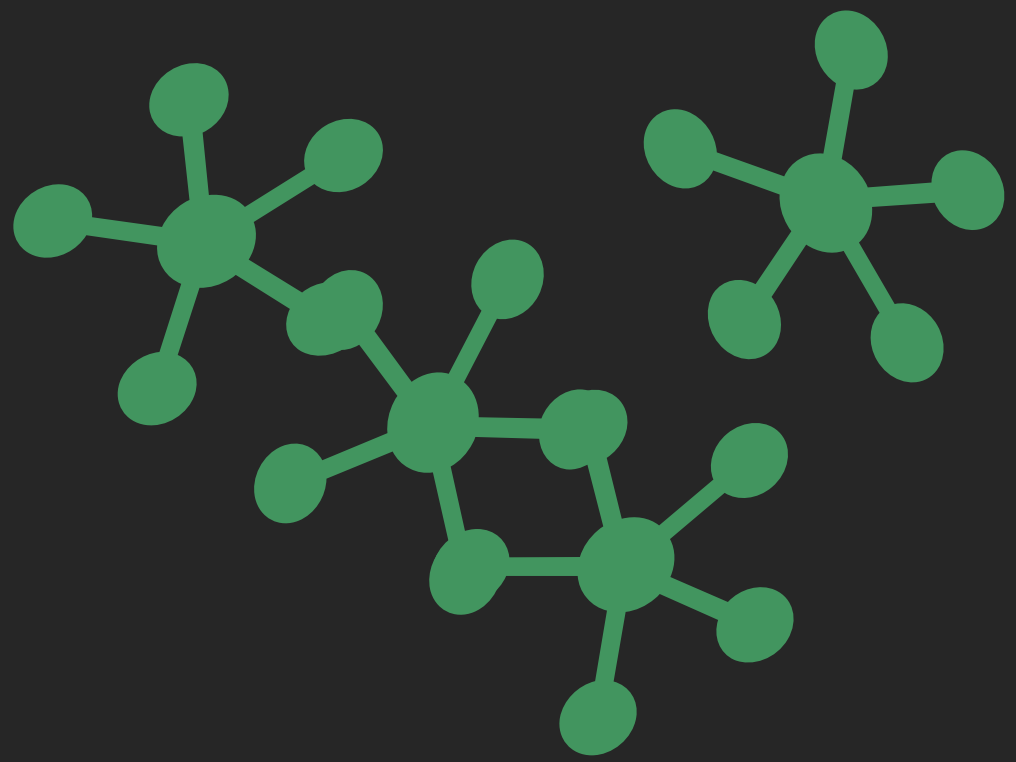
- Adjacência: quando um nó é conectado a outro por uma única aresta;
- Os nós (vértices) L, M e N são adjacentes de K;
- M e L não são adjacentes de N.

TERMINOLOGIA DE GRAFOS



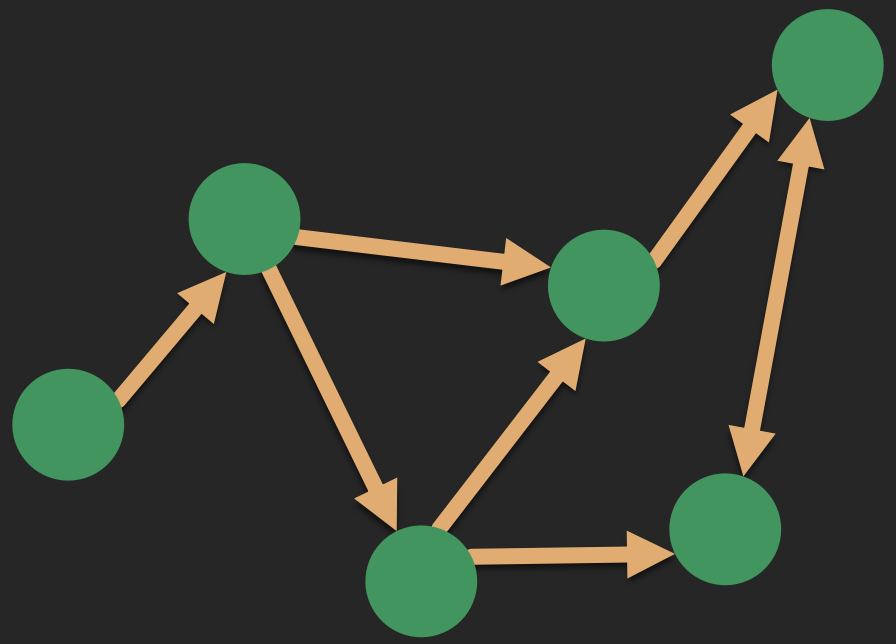
- Caminhos: uma sequência de arestas;
- Entre os nós (vértices) L e P existe um caminho de três arestas.

TERMINOLOGIA DE GRAFOS



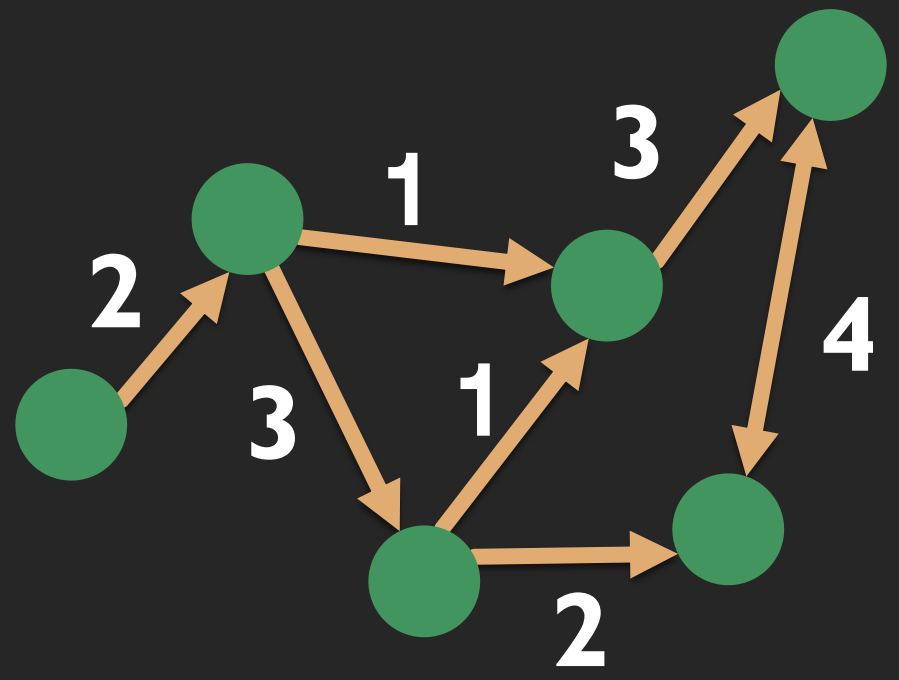
- Grafos conectados e não conectados.

TERMINOLOGIA DE GRAFOS



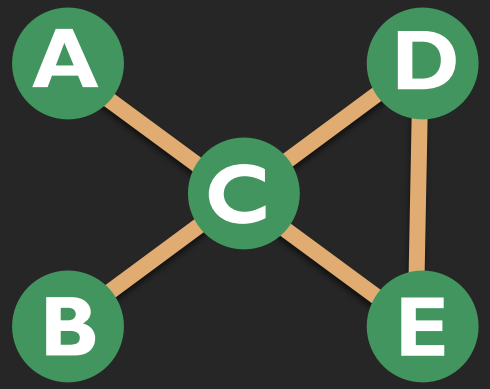
Grafos orientados: possuem setas que configuram uma restrição para a direção.

TERMINOLOGIA DE GRAFOS



Grafos ponderados: as arestas possuem pesos ou custos.

TERMINOLOGIA DE GRAFOS

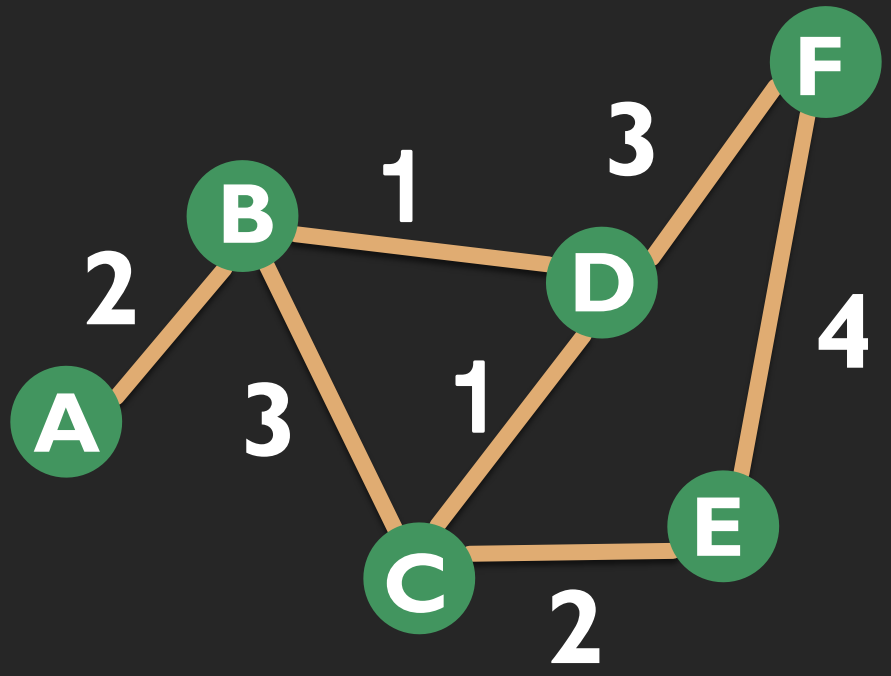


- Matriz de adjacências;
- Lista de adjacências.

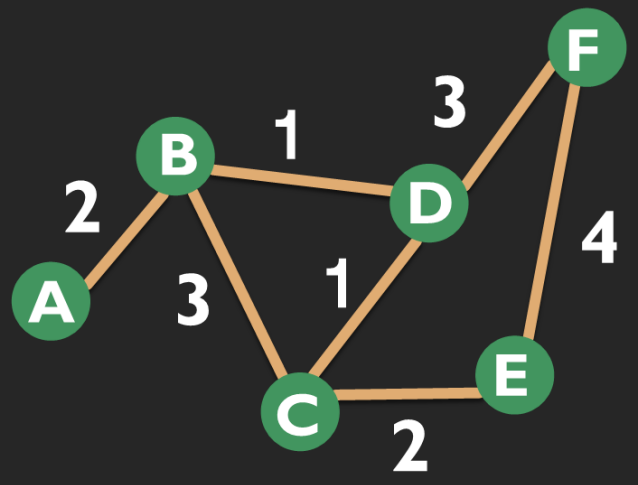
	A	B	C	D	E
A	0	0	1	0	0
B	0	0	1	0	0
C	1	1	0	1	1
D	0	0	1	0	1
E	0	0	1	1	0

REFRESENTAÇÃO DE UM GRAFO

EXERCÍCIO DE EXEMPLO



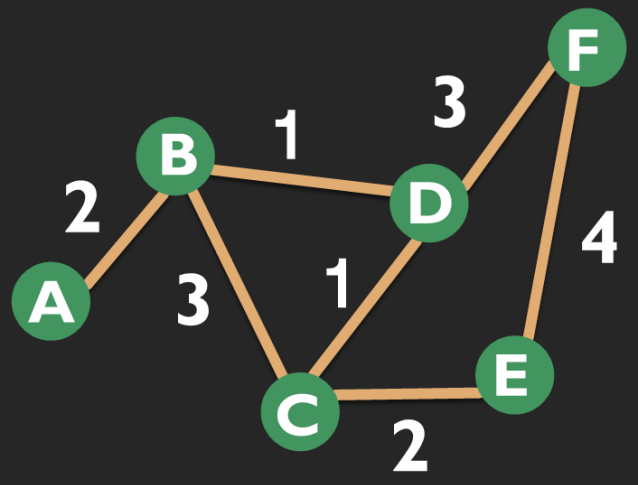
Desenvolva um programa, em qualquer linguagem de programação, para percorrer todos os vértices deste Grafo.



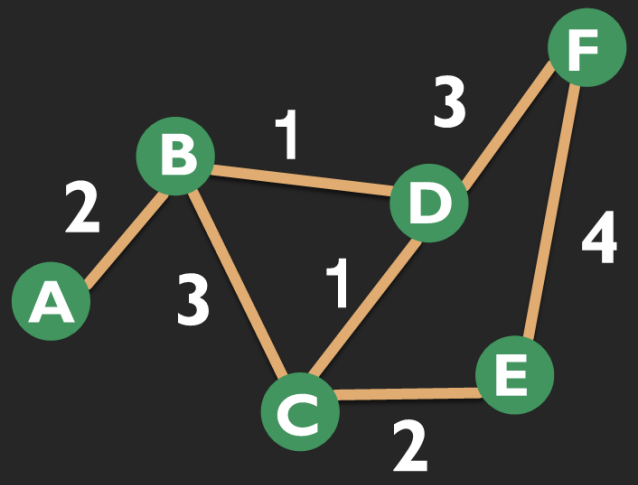
No que tange a disciplina de Estrutura de dados, existem várias estratégias que podem ser adotadas para desenvolver um programa que consiga percorrer todos os vértices deste grafo.

EXERCÍCIO DE EXEMPLO

EXERCÍCIO DE EXEMPLO



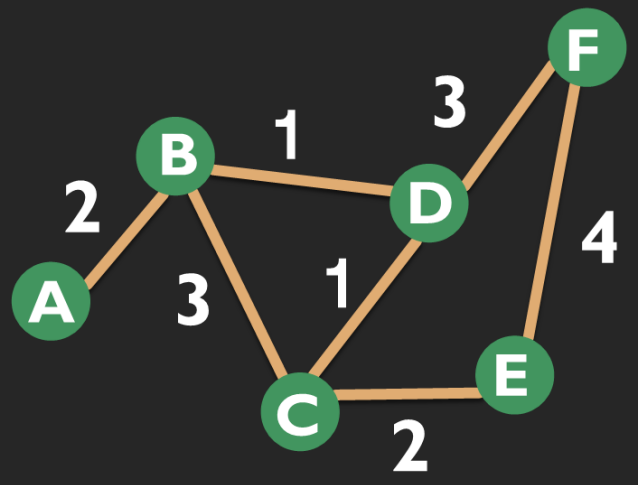
Neste caso, iremos adotar a estratégia da Programação Orientada a Objetos (POO). Além disso, na resolução deste exercício usaremos a linguagem de programação Python.



Em termos de POO, o que podemos abstrair do conceito de grafos?

Em outras palavras, o que podemos representar em termos de classes e objetos?

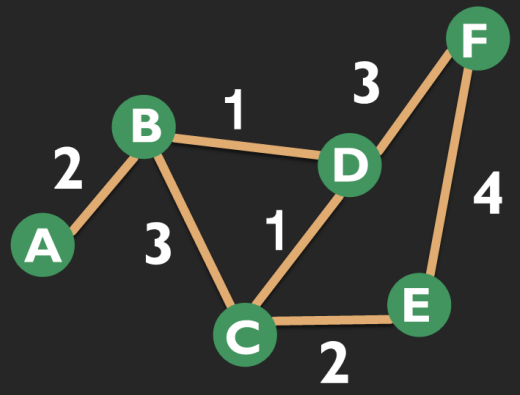
EXERCÍCIO DE EXEMPLO



Será que é possível representar os nós (pontos/vértices) como classe e instanciar esses objetos?

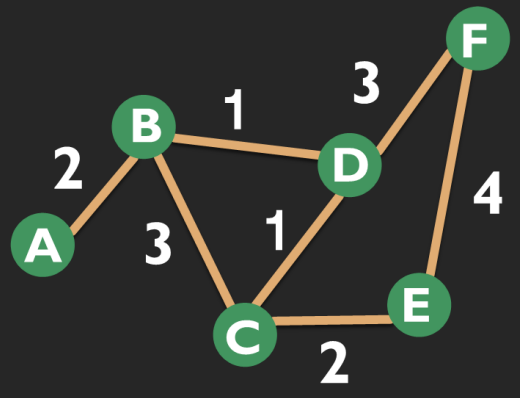
E quanto às adjacências e arestas (ligações/conexões/caminhos)?

EXERCÍCIO DE EXEMPLO



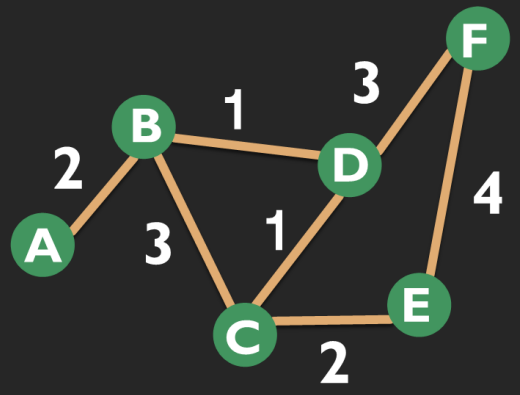

```python
# Classe que representa o Nó
# (ponto/Vértice) de um grafo.
class Noh:
    # Método construtor da classe
    def __init__(self, rotulo):
        # Rótulo é o nome do nó
        self.rotulo = rotulo
        # True se o nó foi visitado
        self.visitado = False
        #  Lista com os nós adjacentes
        self.adjacentes = []
```

EXERCÍCIO DE EXEMPLO



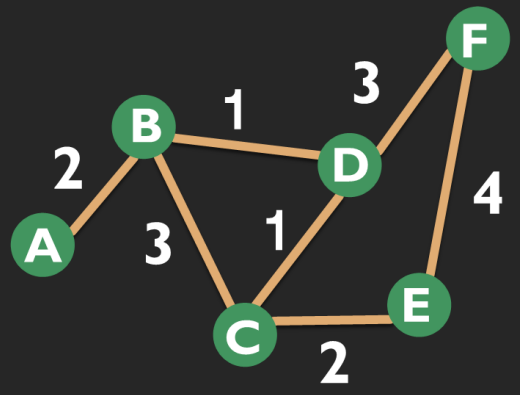

```python
# Classe Adjacente representa o Nó
# imediatamente vizinho de outro nó.
class Adjacente:
    # Método construtor da classe
    def __init__(self, noh, peso):
        # Objeto noh da adjacência
        self.noh = noh
        # A distância ou custo da adjacência
        self.peso = peso
```

EXERCÍCIO DE EXEMPLO



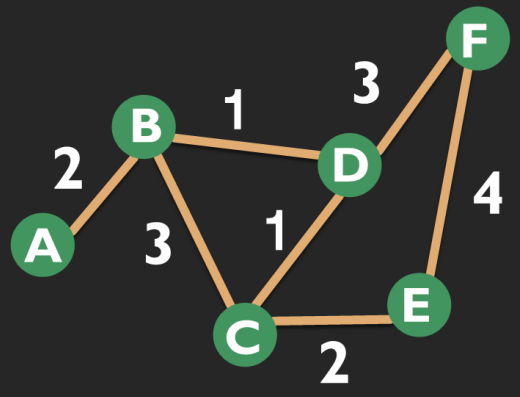

```python
# A classe Noh deve ser capaz de
# identificar seus nós adjacentes
def adiciona_adjacente(self, adjacente):
    # Incluir nó na lista de adjacentes
    self.adjacentes.append(adjacente)

# imprimir a lista de nós vizinhos
def mostra_adjacentes(self):
    # Percorrer a lista de vizinhos
    for i in self.adjacentes:
        # ver nome do nó e seu peso
        print(i.noh.rotulo, i.peso)
```

EXERCÍCIO DE EXEMPLO



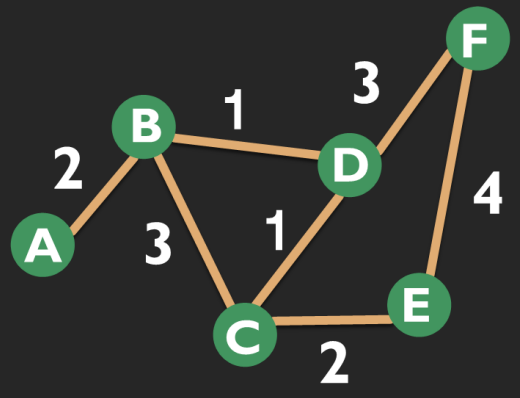
Além da classe Noh (pontos/vértices) e da classe Adjacente (ligações/ conexões/ caminhos) faz-se necessário desenvolver uma classe que irá englobar as classes Noh e Adjacente. Essa será a classe Grafo!

EXERCÍCIO DE EXEMPLO




```python
from Noh import *
from Adjacente import *
# Classe que representa o conceito de grafo
# abarcando a classe de noh (ponto/vértice).
class Grafo:
    noh_A = Noh('A')
    noh_B = Noh('B')
    noh_C = Noh('C')
    noh_D = Noh('D')
    noh_E = Noh('E')
    noh_F = Noh('F')
```

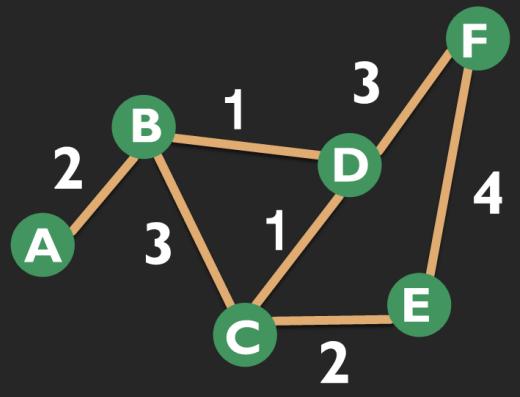
EXERCÍCIO DE EXEMPLO



Até o momento, na classe Grafo, adicionamos apenas a classe Noh.

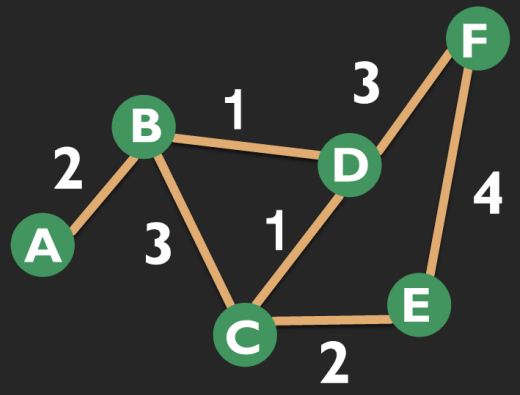
Para ser um grafo é importante haver as adjacências, isto é, incluir os nós adjacentes a cada noh.

EXERCÍCIO DE EXEMPLO




```
13  noh_A.adiciona_adjacente(Adjacente(noh_B, 2))
14  noh_B.adiciona_adjacente(Adjacente(noh_A, 2))
15  noh_B.adiciona_adjacente(Adjacente(noh_C, 3))
16  noh_B.adiciona_adjacente(Adjacente(noh_D, 1))
17  noh_C.adiciona_adjacente(Adjacente(noh_B, 3))
18  noh_C.adiciona_adjacente(Adjacente(noh_D, 1))
19  noh_C.adiciona_adjacente(Adjacente(noh_E, 2))
20  noh_D.adiciona_adjacente(Adjacente(noh_B, 1))
21  noh_D.adiciona_adjacente(Adjacente(noh_C, 1))
22  noh_D.adiciona_adjacente(Adjacente(noh_F, 3))
23  noh_E.adiciona_adjacente(Adjacente(noh_C, 2))
24  noh_E.adiciona_adjacente(Adjacente(noh_F, 4))
25  noh_F.adiciona_adjacente(Adjacente(noh_D, 3))
26  noh_E.adiciona_adjacente(Adjacente(noh_E, 4))
```

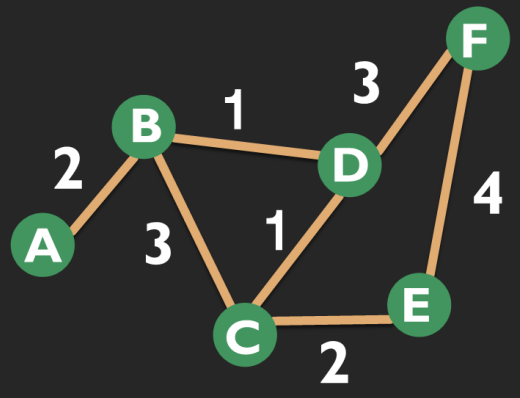
EXERCÍCIO DE EXEMPLO



Agora que já temos concluídas as classes Noh, Adjacente e Grafo, seria interessante testar se o Grafo realmente funciona.

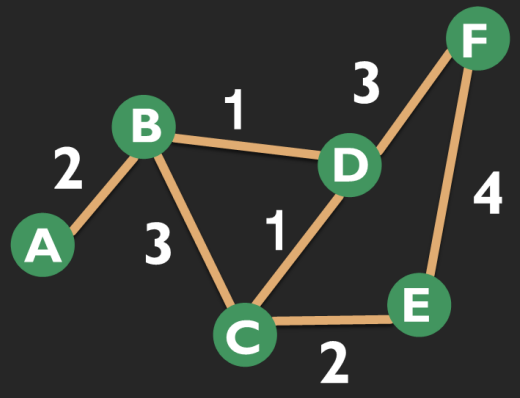
Instancie um objeto Grafo e solicite a um Noh para que mostre seus nós adjacentes.

EXECUTE NO SEU COMPUTADOR



Desenvolver o código que seja capaz de percorrer o nosso grafo que acabamos de criar e imprimir o nome (rótulo) de cada nó que visitar.

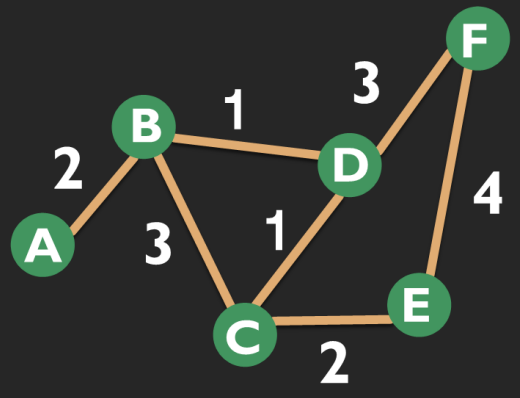
SEGUNDA ETAPA – PERCORRER O GRAFO



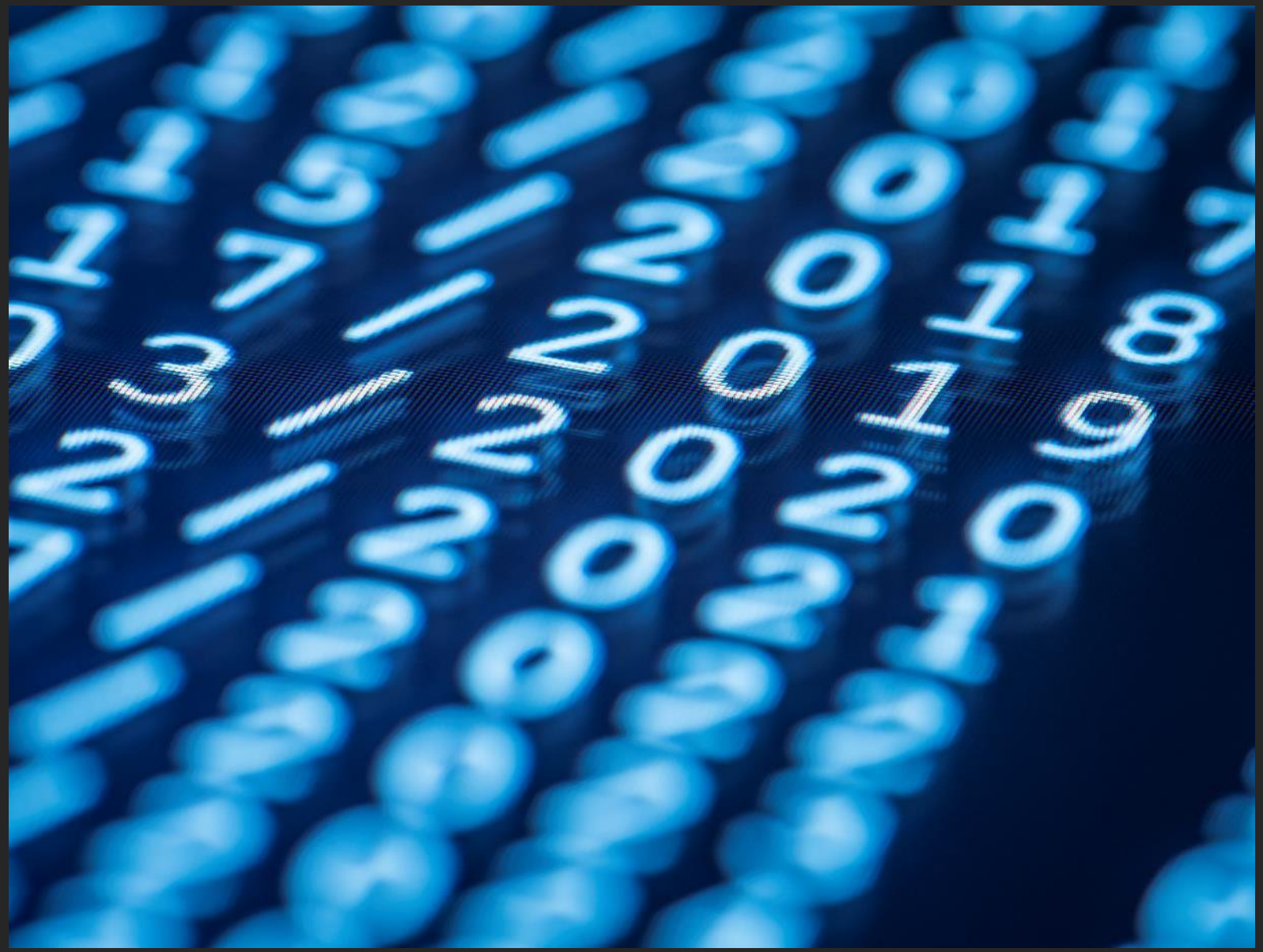
Incrementar o código do desafio A fazendo com que seja capaz de também imprimir o peso de cada aresta que percorrer.

TERCEIRA ETAPA – IMPRIMIR O PESO DE CADA ARESTA VISITADA

DESAFIO



Incrementar o código produzido até agora fazendo com que seja capaz de sempre imprimir os rótulos e pesos/custos pela ordem dos nós de menor custo/peso.



OBRIGADO

DOUGLAS.CARVALHO@IFMA.EDU.BR