# Atividade 2 – Parte 1

- Vamos implementar a estrutura de armazenamento de um **grafo não direcionado não ponderado**.
- Inicialmente, precisamos criar uma interface chamada Grafo, com os seguintes métodos abstratos e seus parâmetros:
  - numero_de_vertices() – retorna o número de vértices
  - numero_de_arestas() – retorna o número de arestas
  - sequencia_de_graus() – retorna a sequência de graus
  - adicionar_aresta(u, v) – adiciona uma nova aresta
  - remover_aresta(u, v) – remove uma aresta
  - imprimir() – imprime a matriz/ lista ligada do grafo
- Onde u e v são vértices do grafo - $\{u, v\} \in V$.
- O peso é aplicado apenas quando trata-se de grafos ponderados.

# Atividade 2

```python
from abc import ABC, abstractmethod

class Grafo(ABC):
    @abstractmethod
    def numero_de_vertices(self):
        pass

    @abstractmethod
    def numero_de_arestas(self):
        pass

    @abstractmethod
    def sequencia_de_graus(self):
        pass

    @abstractmethod
    def adicionar_aresta(self, u, v):
        pass

    @abstractmethod
    def remover_aresta(self, u, v):
        pass

    @abstractmethod
    def imprimir(self):
        pass
```

# Atividade 2 – Parte 2

- Vamos implementar a classe GrafoDenso, que implementa a interface Grafo utilizando uma representação por matriz.
- Para isso, será preciso instanciar cada um dos métodos abstratos definidos na interface Grafo na classe **GrafoDenso**.
- Inicie criando o método para instanciar a classe (constructor).
- O grafo pode ser criado pelo número de vértices, com os rótulos sendo atribuídos em ordem numérica de 0 até a quantidade de vértices-1, ou por rótulos, onde a quantidade de vértices é a quantidade de rótulos.
- Com base nestas informações, deve-se inicializar a matriz base com valores iniciais igual a 0.

# Atividade 2 – Parte 3

- Implemente os métodos:
  - numero_de_vertices() – retorna o número de vértices
  - numero_de_arestas() – retorna o número de arestas
  - sequencia_de_graus() – retorna a sequência de graus

# Atividade 2 – Parte 4

- Crie os métodos para adicionar e remover arestas, e imprimir o Grafo.
  - adicionar_aresta(u, v) – adiciona uma nova aresta
  - remover_aresta(u, v) – remove uma aresta
  - imprimir() – imprime a matriz
- O grafo permite apenas uma aresta entre dois vértives u,v.

# Atividade 2 – Parte 5

- Instancie um Grafo com vértices com rótulos V = {A,B,C,D,E}.
- Adicione as arestas E = {(A,B), (A,C), (C,D), (C,E), (B,D)}
- Imprima o grafo, a quantidade de vértices, a quantidade de arestas, a sequência de graus.
- Remova a aresta e = (A,C)
- Imprima novamente o grafo.

# Atividade 2 – Parte 5

```
Aresta adicionada entre A e B.
Aresta adicionada entre A e C.
Aresta adicionada entre C e D.
Aresta adicionada entre C e E.
Aresta adicionada entre B e D.

Matriz de Adjacência:
   A  B  C  D  E
_____________________
A | 0  1  1  0  0
B | 1  0  0  1  0
C | 1  0  0  1  1
D | 0  1  1  0  0
E | 0  0  1  0  0

Número de vértices: 5
Número de arestas: 5
Sequência de graus: [1, 2, 2, 2, 3]
Aresta removida entre A e C.

Matriz de Adjacência:
   A  B  C  D  E
_____________________
A | 0  1  0  0  0
B | 1  0  0  1  0
C | 0  0  0  1  1
D | 0  1  1  0  0
E | 0  0  1  0  0
```