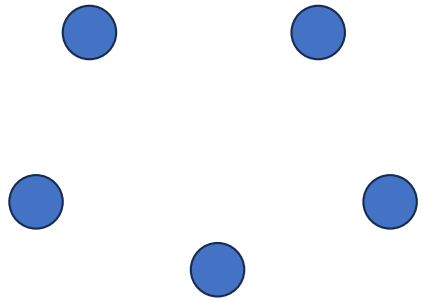
Teoria dos GRAFOS

Aula 03: Tipos de Grafos

Prof. José Alberto S. Torres

Grafo **Nulo** ou **Vazio**

- Um grafo é denominado nulo ou vazio quando não possui nenhuma aresta.



Grafo **simples**

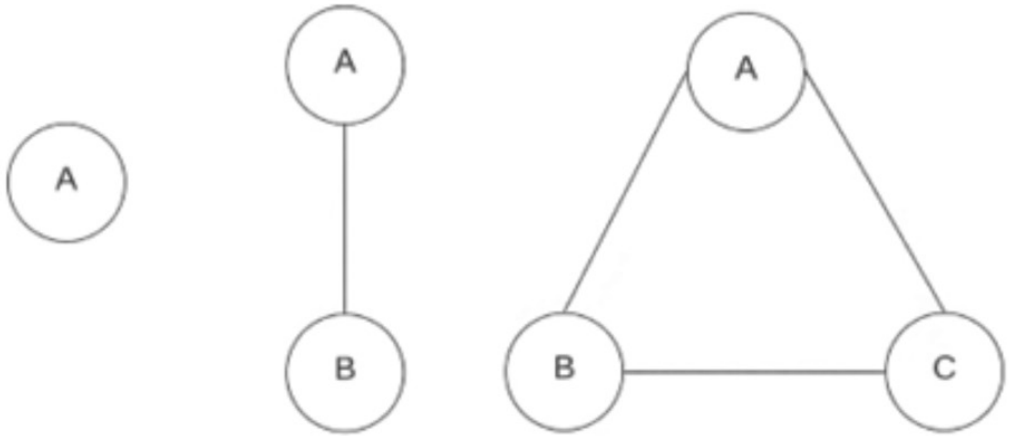
- É possível que uma aresta una os mesmos vértices, ou seja, uma aresta pode formar um laço.
- Além disso, dois vértices u e v podem ser unidos por múltiplas arestas, ou seja, um conjunto de arestas, cada uma tendo u e v como seus pontos extremos.
- Um grafo que **não possui laços ou múltiplas arestas** é chamado de **simples**.

Grafo **Completo**

- É um grafo simples em que **todos os vértices se comunicam entre si** através de alguma aresta, ou seja, todos são adjacentes.
- Um grafo completo com n vértices é comumente denotado como K_n.
- Podemos calcular **o número total de arestas de um grafo completo** com a fórmula:

$$A = \frac{n(n-1)}{2}$$

Grafo **Completo**



a) K1 b) K2 c) K3

Atividade 1

- Crie os seguintes métodos no código que estamos desenvolvendo para os grafos, tanto no GrafoDenso quanto no GrafoEsparso.

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

@abstractmethod
def is_nulo(self):
    pass

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

Grafo **Denso**

```python
def is_simples(self):
    #Percorre a matriz de adjacência para verificar se há laços
    for i in range(self.num_vertices):
        if self.matriz[i][i] != 0:
            return False

    return True

def is_nulo(self):
    if self.numero_de_arestas() == 0 and self.numero_de_vertices() > 0:
        return True
    return False


def is_completo(self):
    if (self.is_simples()) and (
        self.numero_de_arestas() == (self.numero_de_vertices() *
                                     (self.numero_de_vertices() - 1)) // 2):
        return True
    return False
```

IESB
Centro Universitário

Grafo **Esparso**

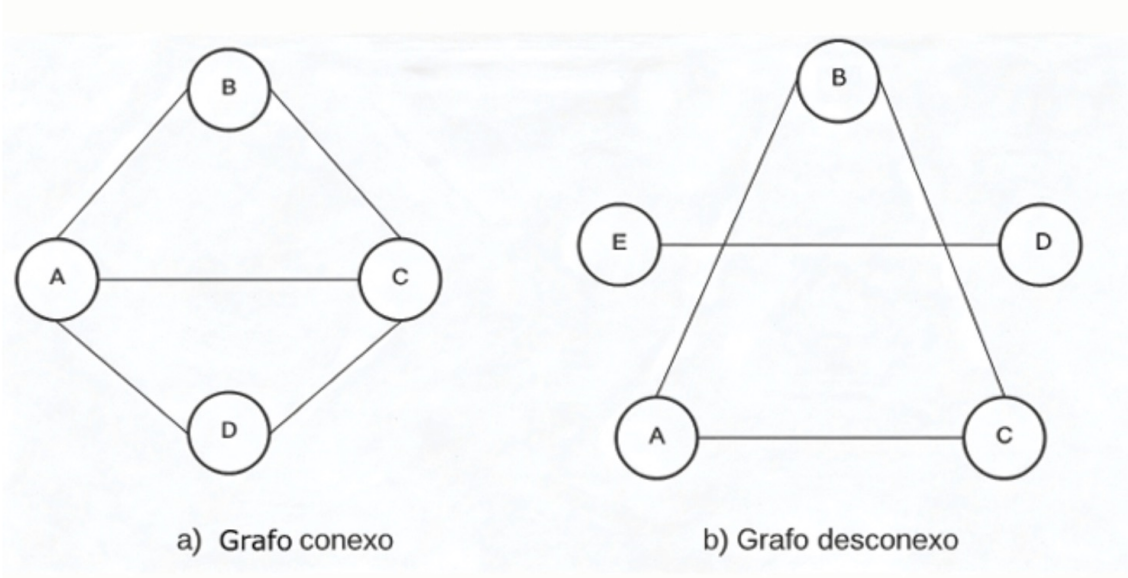
```python
def is_simples(self):
    # Verifica se há laços (loops) na lista de adjacências
    for vertice, vizinhos in self.lista_adj.items():
        if vertice in vizinhos:
            return False
    # Verifica se há arestas duplicadas
    for vertice, vizinhos in self.lista_adj.items():
        if len(vizinhos) != len(set(vizinhos)):
            return False
    # Se não houver laços e arestas duplicadas, o grafo é simples
    return True

def is_nulo(self):
    if self.numero_de_arestas() == 0 and self.numero_de_vertices() > 0:
        return True
    return False


def is_completo(self):
    if (self.is_simples() and self.numero_de_arestas() == (self.numero_de_vertices() *
                                                           (self.numero_de_vertices() - 1)) // 2):
        return True
    return False
```

IESB
Centro Universitário

Grafo **Conexo** e **Desconexo**

- Um grafo é conexo quando, em qualquer par de vértices dele, **consegue-se encontrar algum caminho que os conectam**.
- Dessa forma, **todos os vértices são acessíveis no grafo**.
- Quando essa **característica não ocorre**, então temos um grafo chamado de não conexo ou **desconexo**.

Grafo **Conexo** e **Desconexo**



Subgrafo

- Na teoria dos grafos, um **subgrafo** é, intuitivamente, um grafo que está contido dentro de outro grafo maior.
- Esta noção fundamental permite a análise de estruturas locais, a identificação de padrões e a resolução de uma vasta gama de problemas em diversas áreas do conhecimento.

Subgrafo

- A rigor, temos o seguinte:
  - Um grafo H é um subgrafo de G se $V(H) \subseteq V(G)$ e $E(H) \subseteq E(G)$ tal que para todo $e \in E(H)$ com $e = \{u, v\}$, temos que $u, v \in V(H)$.
  - Quando H é um subgrafo de G, escrevemos $H \subseteq G$.
- Em outras palavras, para que H seja um subgrafo de G, todos os seus vértices e arestas devem também pertencer a G.

IESB
Centro Universitário

Subgrafo **Gerador**

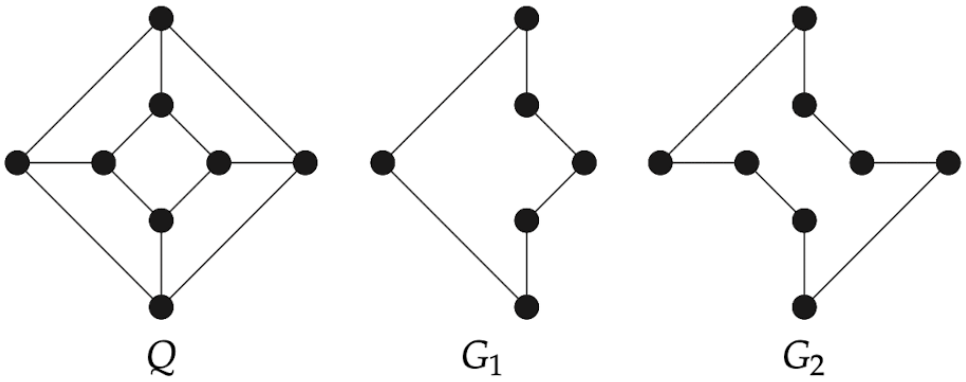
- Um subgrafo H é um **subgrafo gerador** (*spanning subgraph*) de um grafo G se ele **contiver todos os vértices do grafo original**, ou seja, V(H)=V(G).
- A principal característica de um subgrafo gerador é que ele é formado a partir do grafo original pela remoção de um conjunto de arestas, mantendo todos os vértices.

Subgrafo **Induzido**

- É definido por um **subconjunto de vértices do grafo original**.
- Se S é um subconjunto de vértices de um grafo G, o subgrafo induzido por S, denotado como G[S], é o grafo que tem S como seu conjunto de vértices e contém todas as arestas de G que conectam dois vértices em S.
- Em essência, **ao selecionar um grupo de vértices, as arestas entre eles são "herdadas" do grafo original**.

IESB
Centro Universitário

Subgrafos



Atividade 3

- Crie os seguintes métodos no código que estamos desenvolvendo para os grafos, tanto no GrafoDenso quanto no GrafoEsparso.

```python
#Aula 3 - Atividade 2
@abstractmethod
def get_vertices(self):
    pass

@abstractmethod
def get_arestas(self):
    pass

@abstractmethod
def is_subgrafo(self, outro_grafo):
    pass

@abstractmethod
def is_subgrafo_gerador(self, outro_grafo):
    pass

@abstractmethod
def is_subgrafo_induzido(self, outro_grafo):
    pass
```