# Universidade Federal de Itajubá

Nome: Cheyenne Cattani Pereira Matrícula: 2021001943

Link do GitHub: https://github.com/cheycattani/SIN110---Atividades/tree/main/Atividade_3

Relatório Atividade 3 – 12/09

## Função 1 – Tipo do grafo:

Para determinar qual é o tipo do grafo, foi implementada o seguinte código:

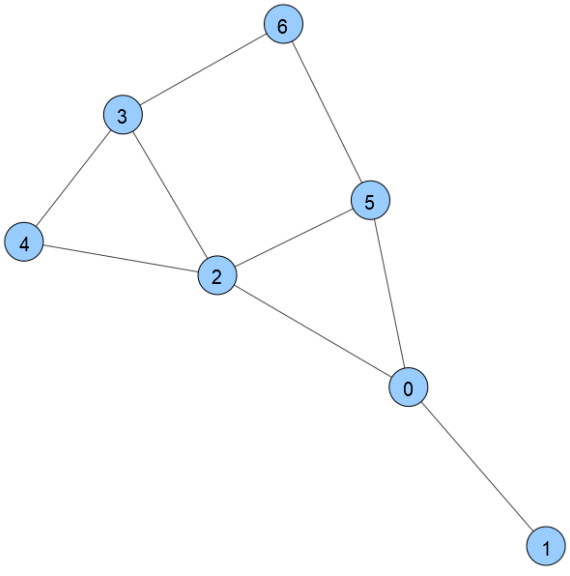
```python
'''Tipo Grafo: Retorna o tipo do grafo representado por uma dada matriz de adjacências
Entrada: matriz de adjacências
Saida: Integer (0 - simples, 1 - digrafo, 2 - multigrafo, 3 - pseudografo)
'''
def tipoGrafo(matriz):
    direcionado = not simetrica(matriz)
    tem_laco = laco(matriz)
    paralela = arestas_paralelas(matriz)

    # Condicional para identificar o grafo
    if not direcionado and not tem_laco and not paralela:  # simples
        return 0
    elif direcionado and not tem_laco and paralela:  # digrafo
        return 1
    elif not direcionado and not tem_laco and paralela:  # multigrafo
        return 2
    else:
        return 3  # pseudografo

# verificar se a matriz é simétrica
def simetrica(matriz):
    ehsimples = True
    for i in range(matriz.shape[0]):
        for j in range(matriz.shape[1]):
            if matriz[i][j] != matriz[j][i]:
                ehsimples = False
                break
    return ehsimples

# verifica se a matriz tem laços
def laco(matriz):
    tem_laco = False
    for i in range(matriz.shape[0]):
        tem_laco = matriz[i][i] != 0
        if tem_laco:
            break
    return tem_laco
```

```python
# verifica se a matriz tem arestas paralelas
def arestas_paralelas(matriz):
    for i in range(matriz.shape[0]):
        for j in range(matriz.shape[1]):
            if matriz[i][j] > 1:
                return True
    return False
```

Para a demonstração de um grafo simples, foi utilizado o dataset “exemplo”. Segue abaixo seu retorno do console.

```
Digite qual arquivo deseja verificar: exemplo
Vertices 0 e 1 são adjacentes? True

Nome da instância: exemplo

Matriz:
 [[0 1 1 0 0 1 0]
 [1 0 0 0 0 0 0]
 [1 0 0 1 1 1 0]
 [0 0 1 0 1 0 1]
 [0 0 1 1 0 0 0]
 [1 0 1 0 0 0 1]
 [0 0 0 1 0 1 0]]

IGRAPH U--- 7 9 --
+ attr: label (v)
+ edges:
0 -- 1 2 5     2 -- 0 3 4 5   4 -- 2 3       6 -- 3 5
1 -- 0         3 -- 2 4 6     5 -- 0 2 6

Tipo do grafo: 0
```

O grafo foi estruturado da seguinte maneira:



**Função 2 – Verificar adjacência:**

Para verificar a adjacência do grafo foi utilizado o código disponibilizado pelo professor.

```
'''Verifica Adjacência: Função que verifica se os vértices vi e vj são
adjacentes.
Entrada: matriz de adjacências (numpy.ndarray), vi (Integer), vj (Integer)
Saída: 0 (Integer) se vi e vj NÃO são adjacentes; 1 se vi e vj são
adjacentes'''

def verificaAdjacencia(matriz, vi, vj):
    if matriz[vi][vj] > 0:  # Se célula M[vi][vj] for maior que 0 existe uma
ou mais arestas
        verticesAdjacentes = True
    else:
        verticesAdjacentes = False
    print('Vertices', vi, 'e', vj, 'são adjacentes?', verticesAdjacentes,
'\n')
    return verticesAdjacentes
```

**Função 3 – Calcular densidade:**

Para calcular a densidade do grafo foi feita a seguinte implementação:

```
'''Calcula Densidade: Retorna o valor da densidade do grafo
Entrada: matriz de adjacências
Saida: Float (valor da densidade com precisão de três casas decimais)'''

def calcDensidade(matriz):
    # Verifica qual é o tipo do grafo
    tipo = tipoGrafo(matriz)

    # Contador
    cont = 0

    # Contando as arestas
    for i in range(matriz.shape[0]):
        for j in range(matriz.shape[1]):
            if matriz[i][j] != 0:
                cont += 1
    # Simples
    if tipo == 0:
        cont /= 2
        return round((2 * cont) / (matriz.shape[0] * (matriz.shape[0] - 1)),
3)

    # Dígrafo
    elif tipo == 1:
        return round(cont / (matriz.shape[0] * (matriz.shape[0] - 1)), 3)
    else:
        return - 1
```

Para a demonstração foi utilizado o dataset “exemplo”, do tipo 0, ou seja, simples. Segue abaixo seu retorno do console com o nome “Densidade”.

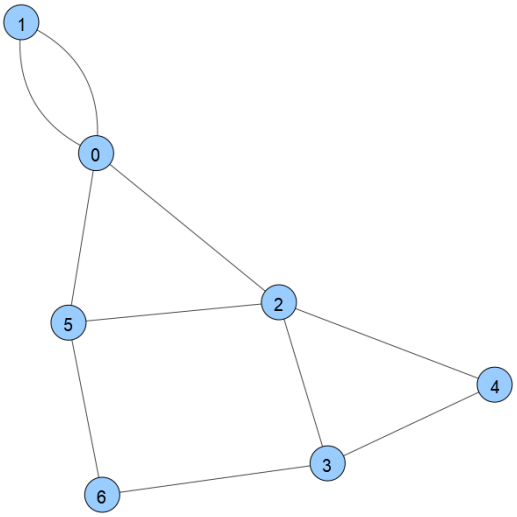
```
Digite qual arquivo deseja verificar: exemplo
Vertices 0 e 1 são adjacentes? True

Nome da instância: exemplo

Matriz:
 [[0 1 1 0 0 1 0]
 [1 0 0 0 0 0 0]
 [1 0 0 1 1 1 0]
 [0 0 1 0 1 0 1]
 [0 0 1 1 0 0 0]
 [1 0 1 0 0 0 1]
 [0 0 0 1 0 1 0]]

IGRAPH U--- 7 9 --
+ attr: label (v)
+ edges:
0 -- 1 2 5     2 -- 0 3 4 5   4 -- 2 3       6 -- 3 5
1 -- 0         3 -- 2 4 6     5 -- 0 2 6

Tipo do grafo: 0
Densidade: 0.429
```

**Função 4 e Função 6 – Insere aresta e Remove aresta:**

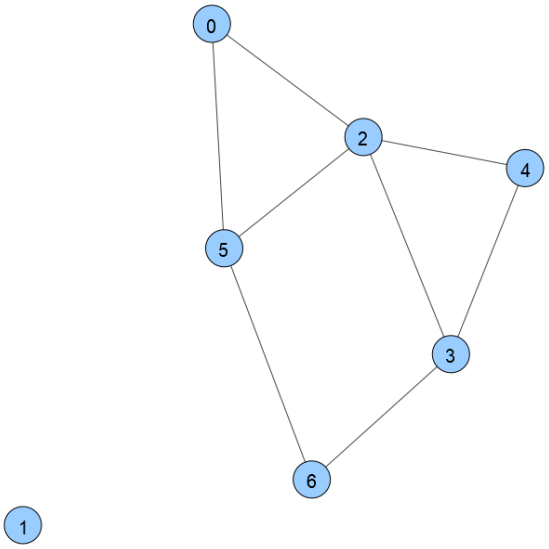
Para a inserção e remoção uma aresta foi feita a seguinte implementação:

```
'''Insere Aresta: Insere uma aresta no grafo considerando o par de vértices
vi e vj
Entrada: Matriz de adjacêcnias, vi e vj (ambos são números inteiros que
indicamo id do vértice
Saída: void'''

def insereAresta(matriz, vi, vj):
    matriz[vi][vj] += 1

''' Remove Aresta: Remove uma aresta no grafo considerando o par de vértices
vi e vj
Entrada: Matriz de adjacêcnias, vi e vj (ambos são números inteiros que
indicamo id do vértice
Saída: boolean (True se remoção Ok, False se a aresta não existe)'''


def removeAresta(matriz, vi, vj):
    matriz[vi][vj] = 0
```

Utilizando o dataset “exemplo”, para a implementação do insereAresta entre 0 e 1 o grafo ficou da seguinte forma:




O grafo da implementação removeAresta entre 0 e 1, ficou da seguinte forma:




O código do console ficou estruturado da seguinte maneira:

```
Digite a coordenada x: 0
Digite a coordenada y: 1


Inserindo arestas...
 [[0 2 1 0 0 1 0]
 [1 0 0 0 0 0 0]
 [1 0 0 1 1 1 0]
 [0 0 1 0 1 0 1]
 [0 0 1 1 0 0 0]
 [1 0 1 0 0 0 1]
 [0 0 0 1 0 1 0]]

Aperte enter!
Removendo arestas...
 [[0 0 1 0 0 1 0]
 [1 0 0 0 0 0 0]
 [1 0 0 1 1 1 0]
 [0 0 1 0 1 0 1]
 [0 0 1 1 0 0 0]
 [1 0 1 0 0 0 1]
 [0 0 0 1 0 1 0]]
```

Função 5 e Função 7 – Insere Vértice e Remove Vértice:

Para a inserção e remoção um vértice foi feita a seguinte implementação:

```
'''Insere Vertice: Insere um vértice no grafo
Entrada: matriz de adjacências, vi(número inteiro que indica o id do vértice)
Saída: Boolean (True se remoção OK, False se a aresta não existe)'''

def insereVertice(matriz, vi):
    shape = matriz.shape

    # Será criada uma nova matriz de zeros
    nMatriz = numpy.zeros((shape[0] + 1, shape[1] + 1))

    # Esse for irá transferir os valores da matriz antiga para a nova.
    for i in range(0, matriz.shape[0]):
        for vj in range(0, matriz.shape[0]):
            nMatriz[vi][vj] = matriz[vi][vj]

    return nMatriz  # retorna a nova matriz

'''Remove Vertice: Remove um vértice do grafo
Entrada: matriz de adjacências, vi (número inteiro que indica o id do vértice
Saída: Boolean (True se remoção OK, False caso o vértice vi, não exista)'''

def removeVertice(matriz, vi):
    pass
```

**Dificuldade:**

Tive dificuldade na separação dos arquivos em pastas, mas consegui. Minha maior dificuldade foi começar a atividade, porque não tenho contato com python e também com as funções 5 e 7, acredito que não estejam corretas. Vou aguardar o professor disponibilizar o arquivo e ver o que errei e tentar entender o que foi feito.