Aula 03 - Árvores e Florestas

Nome: Gustavo Silva Malvestiti

Exercício 01:

GM

Preciso de um código em Python para plotar um grafo direcionado com pesos nas arestas, em que os pesos são empréstimos de uma pessoa para outra.

Também quero o print dos saldos de cada pessoa.

Cada vértice deve ser representado pela INICIAL do nome de cada pessoa.

O sinal dos valores dos saldos de + ou - devem vir ANTES do R$.

Gerar também uma matriz de adjacência ANTES de TUDO.

Colocar em Negrito os maiores e menores saldos.

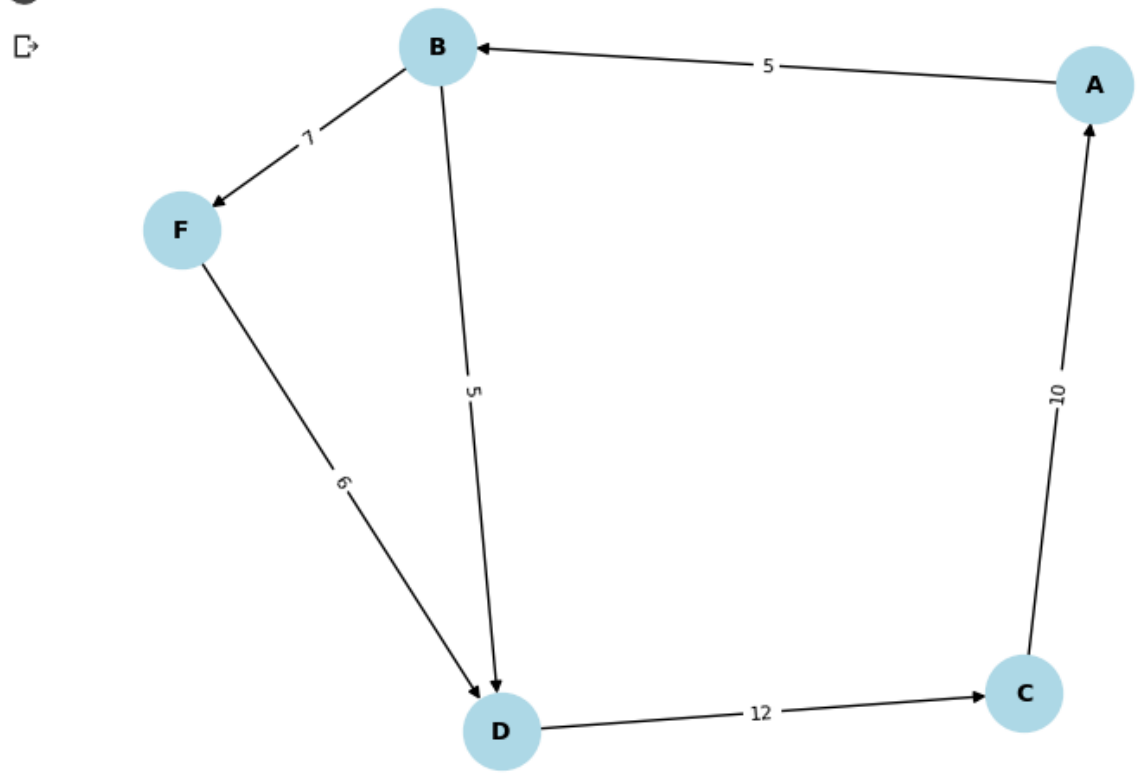
Problema:

Alberto emprestou R$5 para Beltrano, Ciclano emprestou R$10 para Alberto, Beltrano emprestou R$7 para Fulana, Fulana emprestou R$6 para Deltrana, Beltrano emprestou R$5 para Deltrana e Deltrana emprestou R$12 para Ciclano.

Código:

```python
import networkx as nx
import matplotlib.pyplot as plt

# Definindo os empréstimos
emprestimos = {
    ('Alberto', 'Beltrano'): 5,
    ('Ciclano', 'Alberto'): 10,
    ('Beltrano', 'Fulana'): 7,
    ('Fulana', 'Deltrana'): 6,
    ('Beltrano', 'Deltrana'): 5,
    ('Deltrana', 'Ciclano'): 12
}

# Criando o grafo
G = nx.DiGraph()

# Adicionando vértices e arestas com pesos
for (devedor, credor), valor in emprestimos.items():
    G.add_edge(devedor[0], credor[0], weight=valor)
```

```
# Plotando o grafo
pos = nx.spring_layout(G, seed=42)  # Definindo a posição dos vértices
nx.draw(G,         pos,         with_labels=True,         node_size=1000,
node_color='lightblue', font_weight='bold', font_size=10)
labels = nx.get_edge_attributes(G, 'weight')
nx.draw_networkx_edge_labels(G, pos, edge_labels=labels, font_size=8)
plt.show()

# Calculando os saldos de cada pessoa
saldos = {}
for (devedor, credor), valor in emprestimos.items():
    saldos[devedor] = saldos.get(devedor, 0) - valor
    saldos[credor] = saldos.get(credor, 0) + valor

# Imprimindo os saldos
print("Saldos:")
for pessoa, saldo in saldos.items():
    if saldo == max(saldos.values()):
            saldo_str = f"** +R${abs(saldo)}" if saldo > 0 else f"**
-R${abs(saldo)}"
    elif saldo == min(saldos.values()):
            saldo_str = f"** +R${abs(saldo)}" if saldo > 0 else f"**
-R${abs(saldo)}"
    else:
                saldo_str = f"+R${abs(saldo)}" if saldo > 0 else
f"-R${abs(saldo)}"
    print(f"{pessoa}: {saldo_str}")

# Matriz de adjacência
adj_matrix = nx.adjacency_matrix(G, nodelist=sorted(G.nodes)).todense()
print("\nMatriz de Adjacência:")
print(adj_matrix)
```




```
Saldos:
Alberto: ** +R$5
Beltrano: ** -R$7
Ciclano: +R$2
Fulana: +R$1
Deltrana: -R$1

Matriz de Adjacência:
[[ 0  5  0  0  0]
 [ 0  0  0  5  7]
 [10  0  0  0  0]
 [ 0  0 12  0  0]
 [ 0  0  0  6  0]]
```

Exercício 02:

GM

Sendo G um grafo não direcionado com:

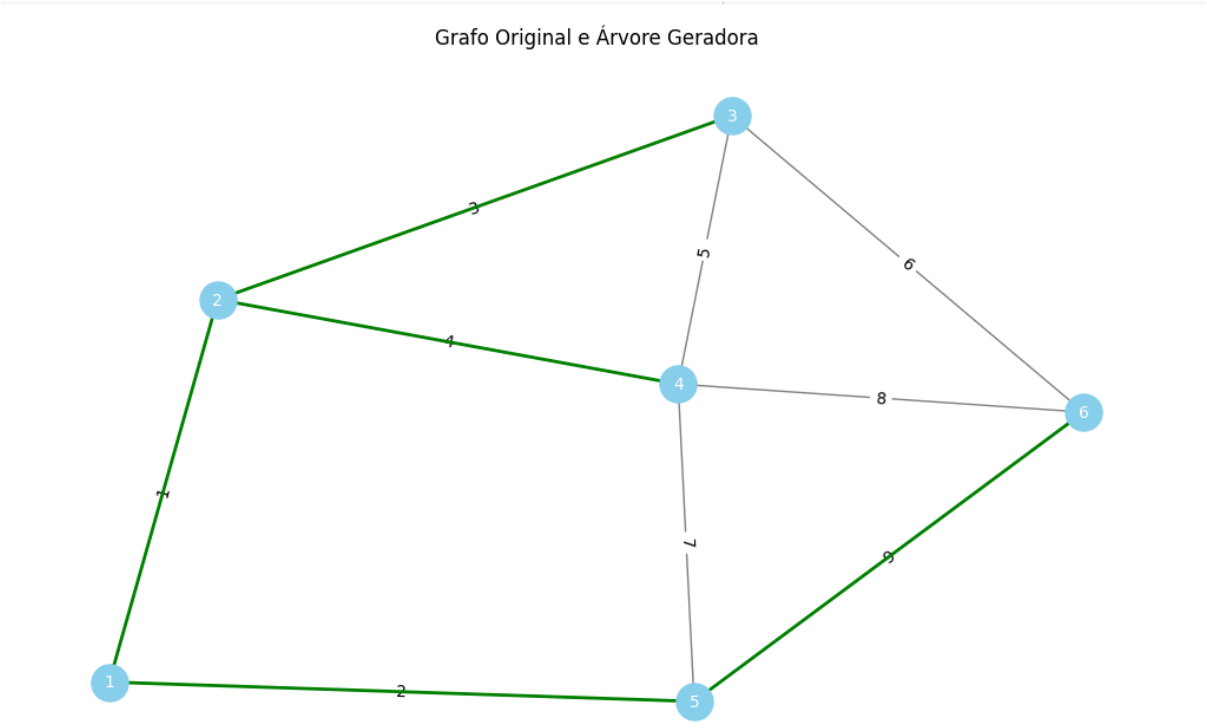
Arestas: 1, 2, 3, 4, 5 e 6
Vértices: 1 para 2, 1 para 5, 2 para 3, 2 para 4, 3 para 4, 3 para 6, 4 para 5, 4 para 6 e 5 para 6

E sendo uma árvore um grafo conexo e acíclico, quero gerar e plotar uma das árvores geradoras de G com um código python.

Código:

```
import networkx as nx
import matplotlib.pyplot as plt

# Definindo as arestas e vértices do grafo G
edges = [(1, 2), (1, 5), (2, 3), (2, 4), (3, 4), (3, 6), (4, 5), (4, 6), (5, 6)]

# Criando um grafo usando NetworkX
G = nx.Graph()
G.add_edges_from(edges)

# Gerando uma árvore geradora
spanning_tree = nx.minimum_spanning_tree(G)

# Plotando o grafo original e a árvore geradora
pos = nx.spring_layout(G)  # Define a posição dos nós para melhor visualização

plt.figure(figsize=(10, 6))

# Grafo original
nx.draw(G, pos, with_labels=True, node_size=500, font_size=10, font_color='white', node_color='skyblue', edge_color='gray')
nx.draw_networkx_edge_labels(G, pos, edge_labels={(u, v): str(i) for i, (u, v) in enumerate(edges, start=1)})

# Árvore geradora
nx.draw_networkx_edges(spanning_tree, pos, edge_color='green', width=2)

plt.title("Grafo Original e Árvore Geradora")
plt.axis('off')
```

```
plt.show()
```



Exercício 03:

GM

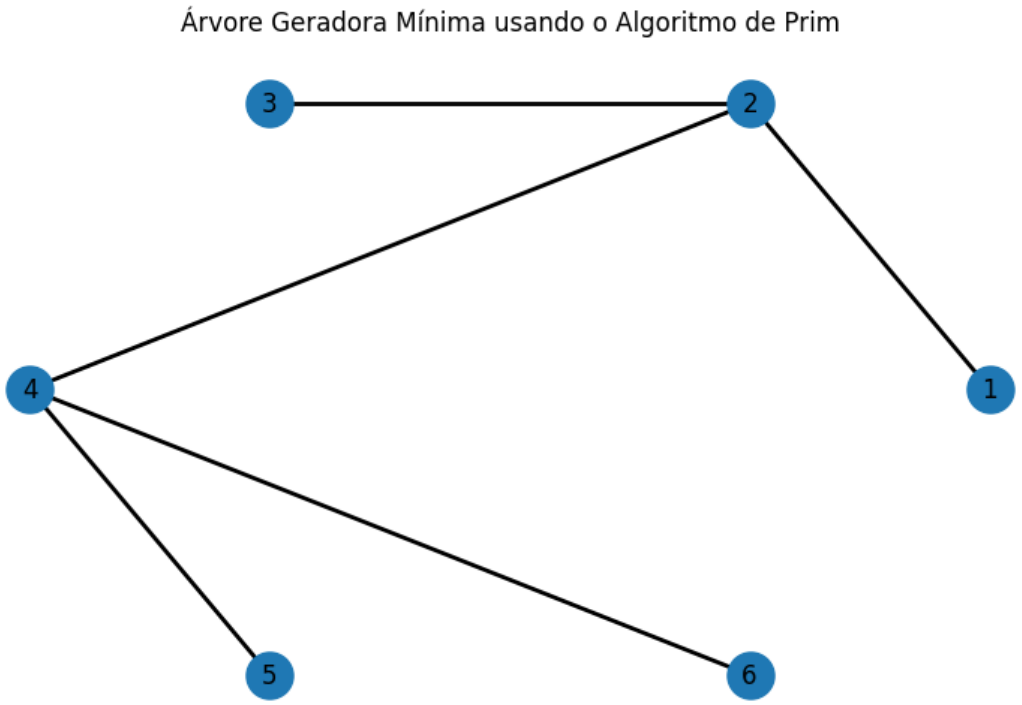
Sendo G um grafo não direcionado com:

Arestas: 1, 2, 3, 4, 5 e 6
Véstices e Pesos: 1 para 2 (1), 1 para 3 (3), 2 para 3 (1), 2 para 4 (1), 2 para 5 (4), 3 para 4 (3), 3 para 5 (2), 4 para 5 (-2), 4 para 6 (1) e 5 para 6 (2)

Preciso de um código em python que execute o algoritmo de Prim para obter e plotar a árvore geradora mínima do grafo G a partir do vértice inicial 1.

Código:

```
import networkx as nx
import matplotlib.pyplot as plt

# Definindo as arestas, vértices e pesos
edges = [(1, 2, 1), (1, 3, 3), (2, 3, 1), (2, 4, 1), (2, 5, 4), (3, 4, 3), (3, 5, 2), (4, 5, -2), (4, 6, 1), (5, 6, 2)]
G = nx.Graph()
G.add_weighted_edges_from(edges)

# Executando o algoritmo de Prim
mst = nx.minimum_spanning_tree(G, algorithm='prim')

# Plotando da árvore geradora mínima
pos = nx.circular_layout(G)  # Posições dos vértices para a plotagem

plt.figure(figsize=(10, 6))

# Árvore Geradora Mínima
nx.draw_networkx_edges(mst, pos, width=2.0, alpha=1)
nx.draw_networkx_nodes(mst, pos, node_size=500)
nx.draw_networkx_labels(G, pos, font_size=12, font_color='black')

# Exibindo o plot
plt.title("Árvore Geradora Mínima usando o Algoritmo de Prim")
plt.axis('off')
plt.show()
```


Árvore Geradora Mínima usando o Algoritmo de Prim
3
2
4
1
5
6

Exercício 04:

GM

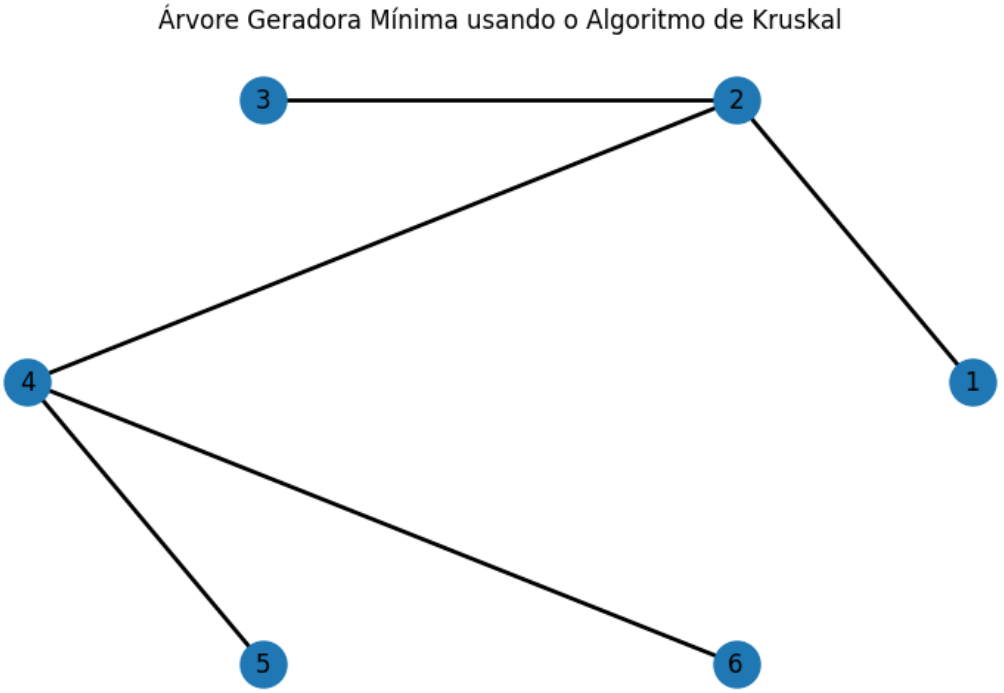
Sendo G um grafo não direcionado com:

Arestas: 1, 2, 3, 4, 5 e 6
Véstices e Pesos: 1 para 2 (1), 1 para 3 (3), 2 para 3 (1), 2 para 4 (1), 2 para 5 (4), 3 para 4 (3), 3 para 5 (2), 4 para 5 (-2), 4 para 6 (1) e 5 para 6 (2)

Preciso de um código em python que execute o algoritmo de Kruskal para obter e plotar a árvore geradora mínima do grafo G a partir do vértice inicial 1.

Código:

```
import networkx as nx
import matplotlib.pyplot as plt

# Definindo as arestas, vértices e pesos
edges = [(1, 2, 1), (1, 3, 3), (2, 3, 1), (2, 4, 1), (2, 5, 4), (3, 4,
3), (3, 5, 2), (4, 5, -2), (4, 6, 1), (5, 6, 2)]
G = nx.Graph()
G.add_weighted_edges_from(edges)

# Executando o algoritmo de Prim
mst = nx.minimum_spanning_tree(G, algorithm='kruskal')

# Plotando da árvore geradora mínima
pos = nx.circular_layout(G)  # Posições dos vértices para a plotagem

plt.figure(figsize=(10, 6))

# Árvore Geradora Mínima
nx.draw_networkx_edges(mst, pos, width=2.0, alpha=1)
nx.draw_networkx_nodes(mst, pos, node_size=500)
nx.draw_networkx_labels(G, pos, font_size=12, font_color='black')

# Exibindo o plot
plt.title("Árvore Geradora Mínima usando o Algoritmo de Kruskal")
plt.axis('off')
plt.show()
```


Árvore Geradora Mínima usando o Algoritmo de Kruskal
3
2
4
1
5
6

Exercício 05:

GM

Sendo G um grafo não direcionado com:

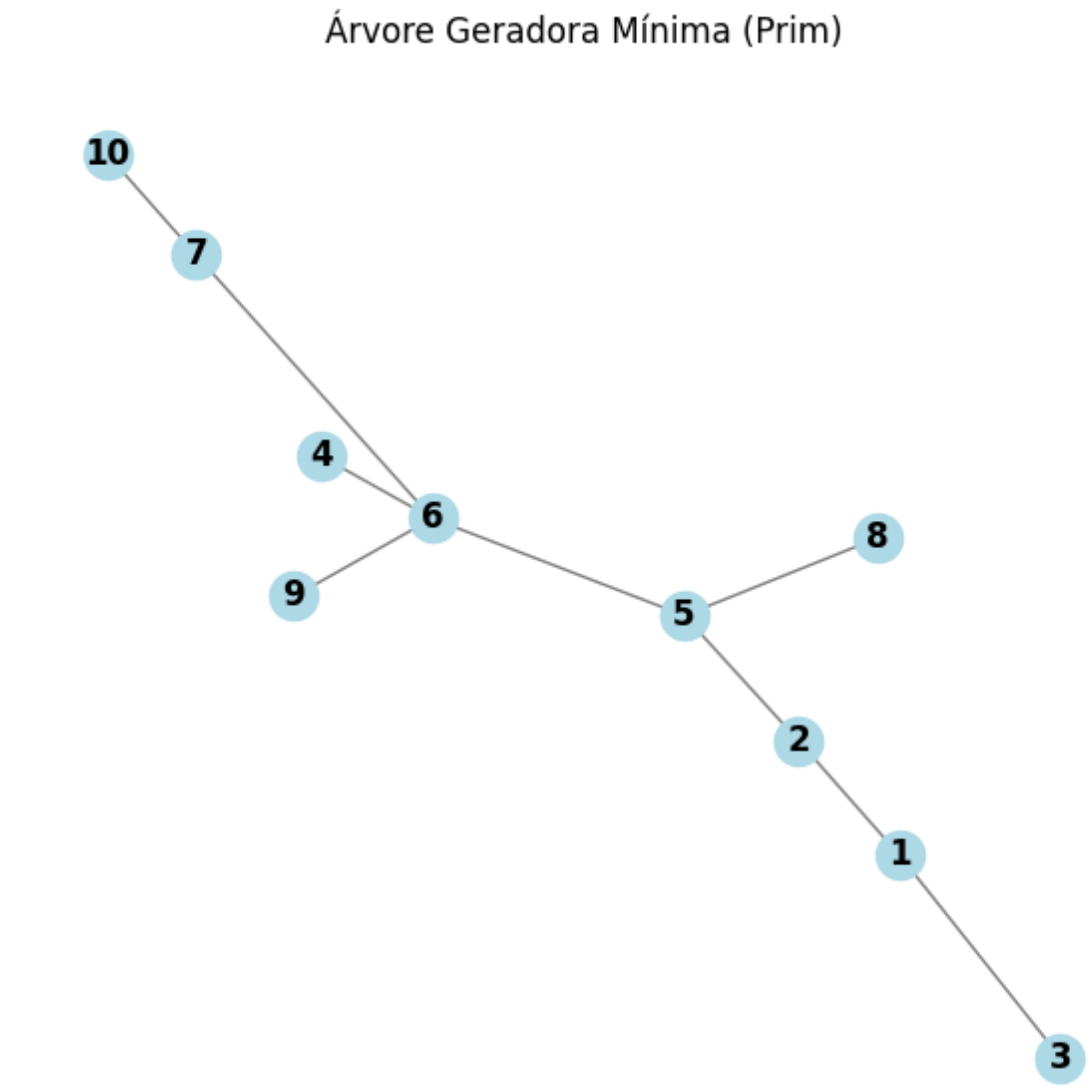
Arestas: 1, 2, 3, 4, 5 e 6
Véstices e Pesos: 1 para 2 (6), 1 para 3 (1), 1 para 4 (14), 2 para 3 (10), 2 para 5 (6), 2 para 6 (8), 3 para 4 (7), 3 para 6 (7), 4 para 6 (6), 4 para 7 (6), 5 para 6 (4), 5 para 8 (3), 6 para 7 (1), 6 para 8 (6), 6 para 9 (5), 7 para 10 (5), 8 para 9 (6), 9 para 10 (6)

Preciso de um código em python que execute os algoritmos de Prim e Kruskal para obter e plotar a árvore geradora mínima do grafo G a partir do vértice inicial 1.

Código:

```
import networkx as nx
import matplotlib.pyplot as plt

# Definindo as arestas e pesos
edges = [
    (1, 2, 6), (1, 3, 1), (1, 4, 14),
    (2, 3, 10), (2, 5, 6), (2, 6, 8),
    (3, 4, 7), (3, 6, 7),
    (4, 6, 6), (4, 7, 6),
    (5, 6, 4), (5, 8, 3),
    (6, 7, 1), (6, 8, 6), (6, 9, 5),
    (7, 10, 5),
    (8, 9, 6),
    (9, 10, 6)
]

# Criando o grafo
G = nx.Graph()
G.add_weighted_edges_from(edges)

# Algoritmo de Prim
prim_tree = nx.minimum_spanning_tree(G)

# Algoritmo de Kruskal
kruskal_tree = nx.minimum_spanning_tree(G, algorithm='kruskal')

# Plotando os grafos resultantes
plt.figure(figsize=(12, 6))
```

```
plt.subplot(1, 2, 1)
nx.draw(prim_tree,          with_labels=True,          font_weight='bold',
node_color='lightblue', edge_color='gray', font_color='black')
plt.title("Árvore Geradora Mínima (Prim)")

plt.subplot(1, 2, 2)
nx.draw(kruskal_tree,        with_labels=True,        font_weight='bold',
node_color='lightgreen', edge_color='gray', font_color='black')
plt.title("Árvore Geradora Mínima (Kruskal)")

plt.tight_layout()
plt.show()
```



Árvore Geradora Mínima (Kruskal)