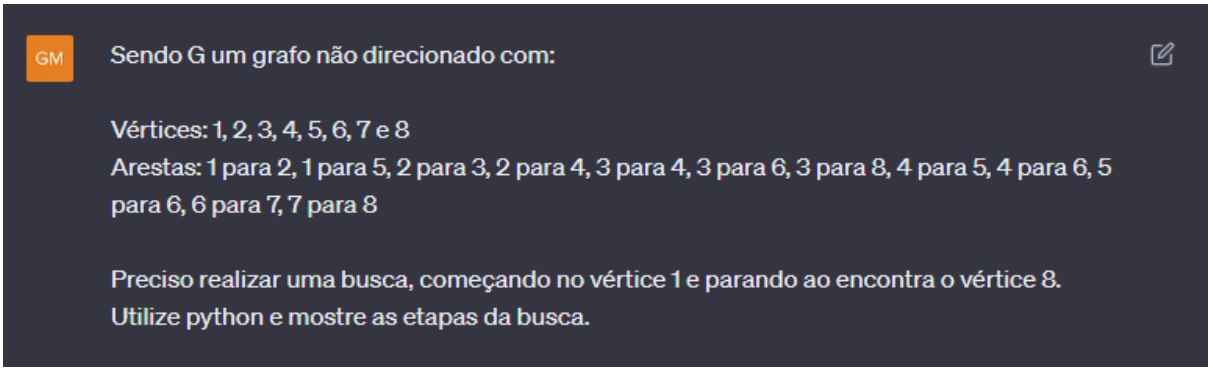
Aula 04 - Conectividade e Buscas em Grafos

Nome: Gustavo Silva Malvestiti

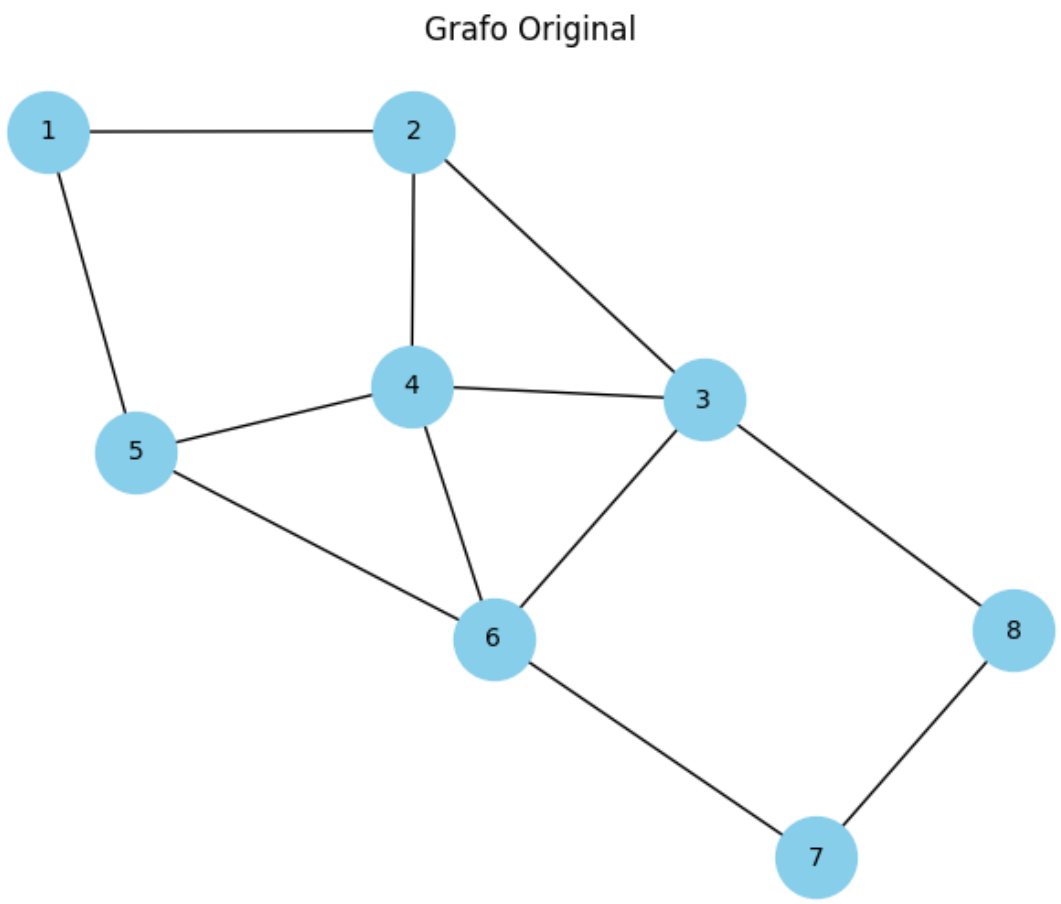
Exercício 01:

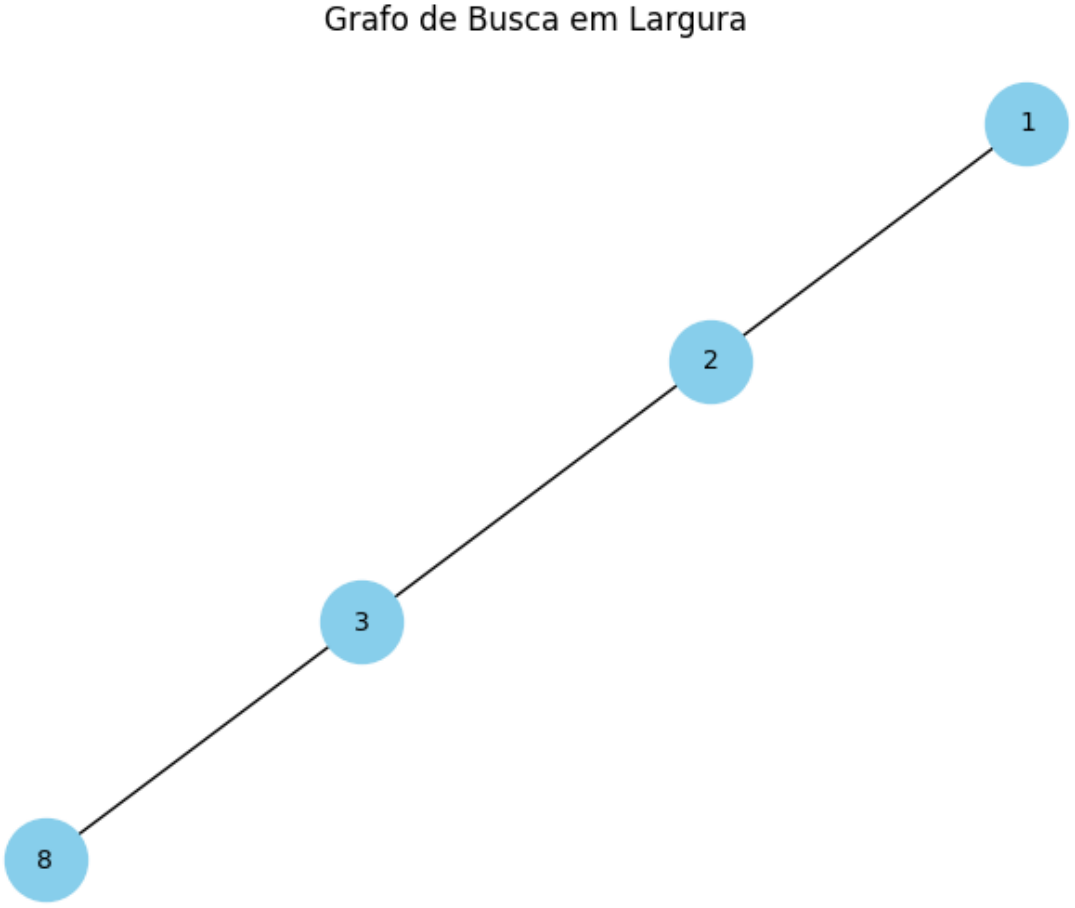


Código:

```python
import networkx as nx
import matplotlib.pyplot as plt
from collections import deque

# Definindo o grafo
edges = [(1, 2), (1, 5), (2, 3), (2, 4), (3, 4), (3, 6), (3, 8),
         (4, 5), (4, 6), (5, 6), (6, 7), (7, 8)]
G = nx.Graph()
G.add_edges_from(edges)

# Definindo a função para busca em largura
def bfs(graph, start, target):
    visited = set()
    queue = deque([(start, [start])])

    while queue:
        node, path = queue.popleft()
        if node == target:
            return path
        if node not in visited:
            visited.add(node)
            neighbors = graph.neighbors(node)
            for neighbor in neighbors:
                if neighbor not in visited:
                    queue.append((neighbor, path + [neighbor]))

    return None
```

```
# Realizando a busca BFS
start_vertex = 1
target_vertex = 8
path = bfs(G, start_vertex, target_vertex)

if path:
    print(f"Caminho encontrado: {path}")
else:
    print("Caminho não encontrado.")

# Plotando o grafo original
pos = nx.spring_layout(G, seed=42)
nx.draw(G,      pos,      with_labels=True,      node_color='lightblue',
font_weight='bold')
plt.title("Grafo Original")
plt.show()

# Plotando o grafo de busca
searched_edges = [(path[i], path[i+1]) for i in range(len(path)-1)]
searched_graph = nx.Graph(searched_edges)
nx.draw(searched_graph, pos, with_labels=True, node_color='lightgreen',
font_weight='bold')
plt.title("Grafo de Busca")
plt.show()
```




Grafo de Busca em Largura
1
2
3
8

Exercício 02:

GM

Sendo G um grafo não direcionado com:

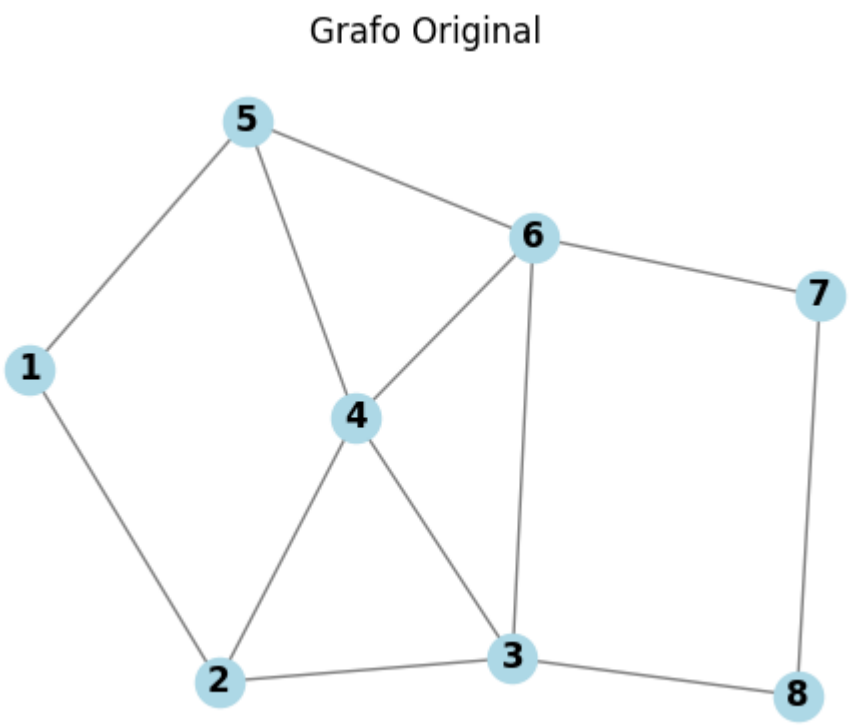
Vértices: 1, 2, 3, 4, 5, 6, 7 e 8
Arestas: 1 para 2, 1 para 5, 2 para 3, 2 para 4, 3 para 4, 3 para 6, 3 para 8, 4 para 5, 4 para 6, 5 para 6, 6 para 7, 7 para 8

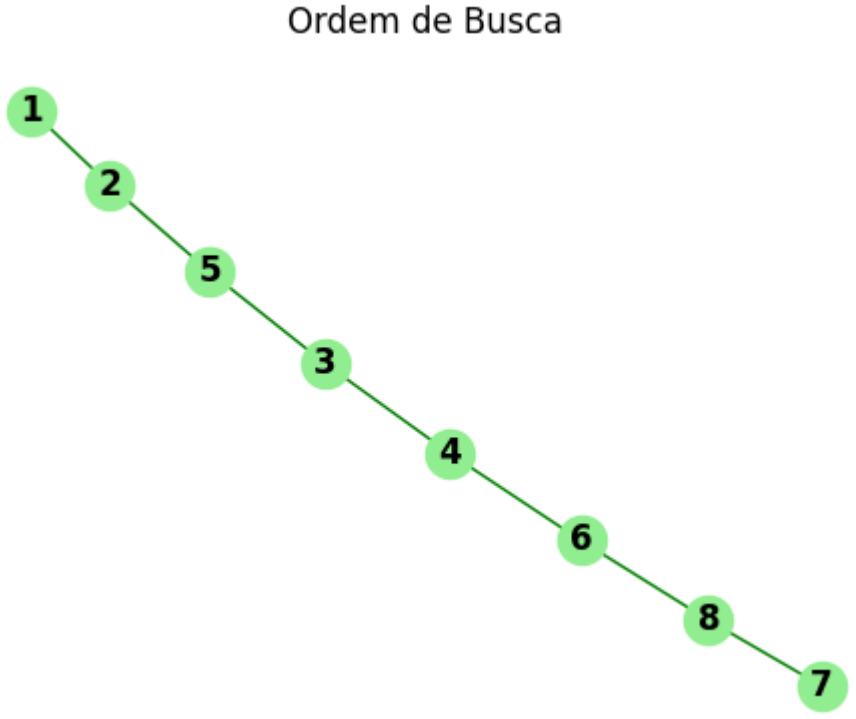
Preciso realizar uma busca, começando no vértice 1 e parar após percorrer todos os vértices, sem repetições. Utilize um algoritmo em python e plote os grafos original e de busca.

Código:

```
from collections import deque
import networkx as nx
import matplotlib.pyplot as plt

# Definindo o grafo
edges = [(1, 2), (1, 5), (2, 3), (2, 4), (3, 4), (3, 6), (3, 8),
         (4, 5), (4, 6), (5, 6), (6, 7), (7, 8)]

G = nx.Graph()
G.add_edges_from(edges)

# Função para realizar a busca em largura
def bfs(graph, start):
    visited = set()
    queue = deque([start])
    order = []

    while queue:
        vertex = queue.popleft()
        if vertex not in visited:
            visited.add(vertex)
            order.append(vertex)
              queue.extend(neigh for neigh in graph[vertex] if neigh not
in visited)

    return order

# Realizando a busca em largura a partir do vértice 1
search_order = bfs(G, 1)
```

```
# Criando um novo grafo para representar a ordem de busca
search_graph = nx.Graph()
for i in range(len(search_order) - 1):
    search_graph.add_edge(search_order[i], search_order[i+1])

# Plotando os grafos
plt.figure(figsize=(10, 4))

plt.subplot(1, 2, 1)
nx.draw(G,              with_labels=True,             font_weight='bold',
node_color='lightblue', edge_color='gray')
plt.title("Grafo Original")

plt.subplot(1, 2, 2)
nx.draw(search_graph,        with_labels=True,        font_weight='bold',
node_color='lightgreen', edge_color='green')
plt.title("Ordem de Busca")

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




Ordem de Busca
1
2
5
3
4
6
8
7

Exercício 03:

GM

Sendo G um grafo não direcionado com:

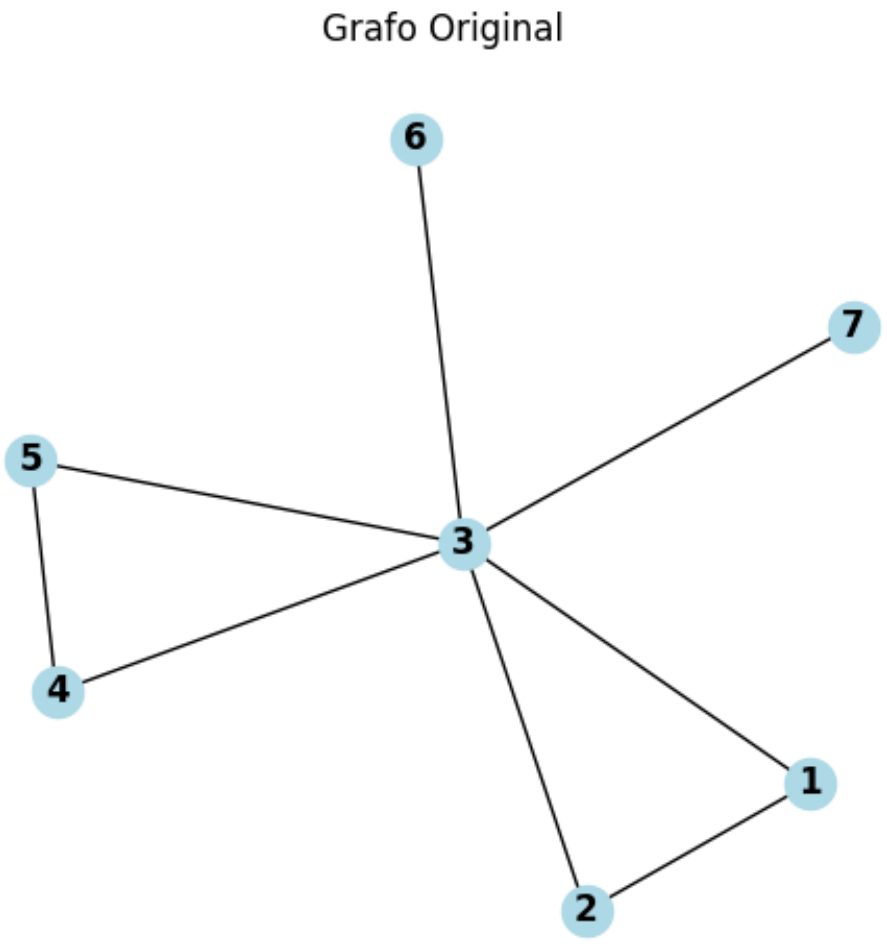
Vértices: 1, 2, 3, 4, 5, 6 e 7
Arestas: 1 para 2, 1 para 3, 2 para 3, 3 para 4, 3 para 5, 3 para 6, 3 para 7 e 4 para 5

Preciso realizar uma busca em profundidade, começando no vértice 1. Utilize um algoritmo python e plote o grafo original e o de busca

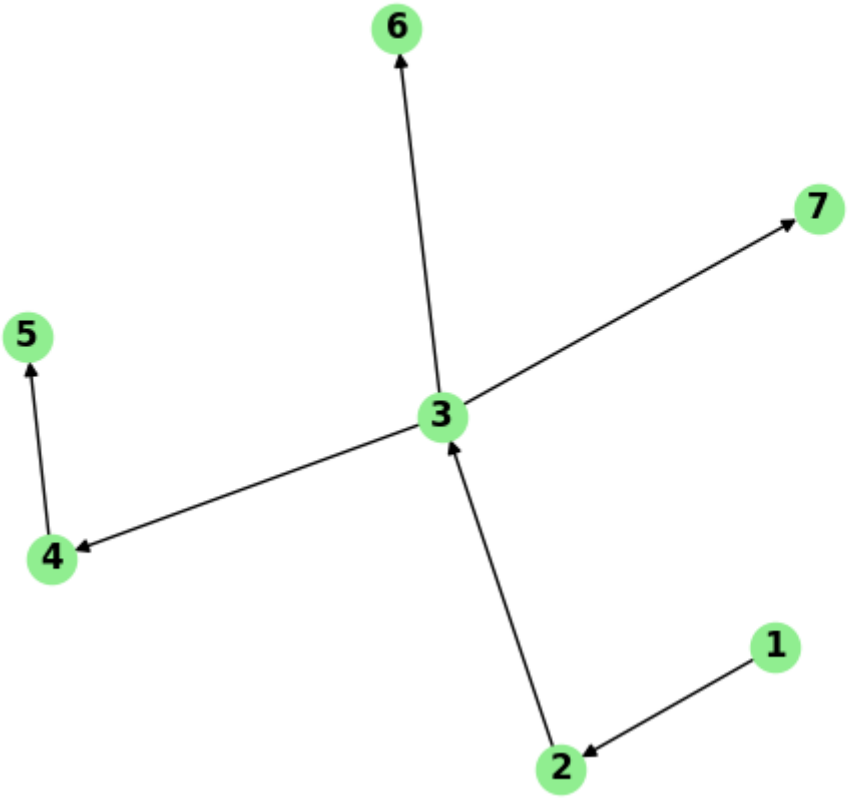
Código:

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

# Definição do grafo
edges = [(1, 2), (1, 3), (2, 3), (3, 4), (3, 5), (3, 6), (3, 7), (4,
5)]
G = nx.Graph()
G.add_edges_from(edges)

# Função de busca em profundidade recursiva
def depth_first_search(graph, node, visited):
    visited[node] = True
    print(node, end=" ")

    for neighbor in graph.neighbors(node):
        if not visited[neighbor]:
            depth_first_search(graph, neighbor, visited)

# Inicialização da busca em profundidade
start_node = 1
visited_nodes = {node: False for node in G.nodes()}

print("Ordem de visita durante a busca em profundidade:")
depth_first_search(G, start_node, visited_nodes)

# Plotagem dos grafos
plt.figure(figsize=(10, 5))

# Grafo original
plt.subplot(1, 2, 1)
pos = nx.spring_layout(G, seed=45)
```

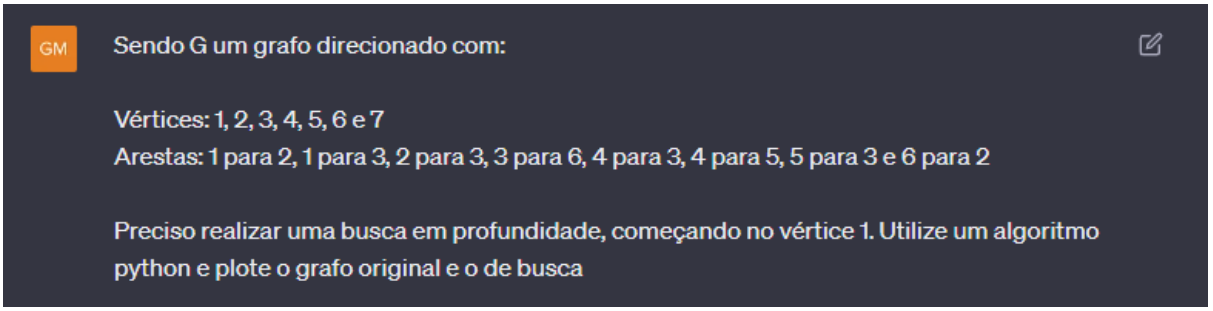
```
nx.draw(G,      pos,      with_labels=True,      node_color="lightblue",
font_weight="bold")
plt.title("Grafo Original")

# Grafo de busca em profundidade
plt.subplot(1, 2, 2)
dfs_tree = nx.dfs_tree(G, source=start_node)
nx.draw(dfs_tree,   pos,   with_labels=True,   node_color="lightgreen",
font_weight="bold")
plt.title("Árvore de Busca em Profundidade")

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



Árvore de Busca em Profundidade



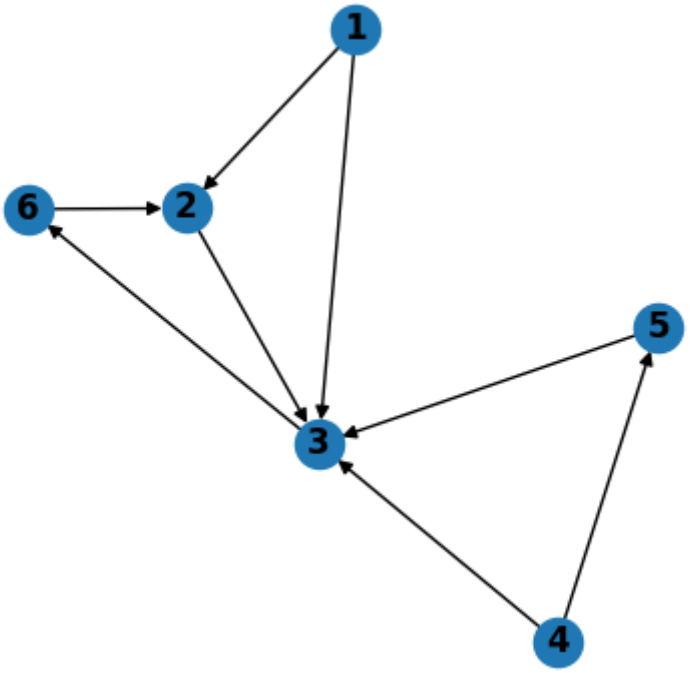
Exercício 04:



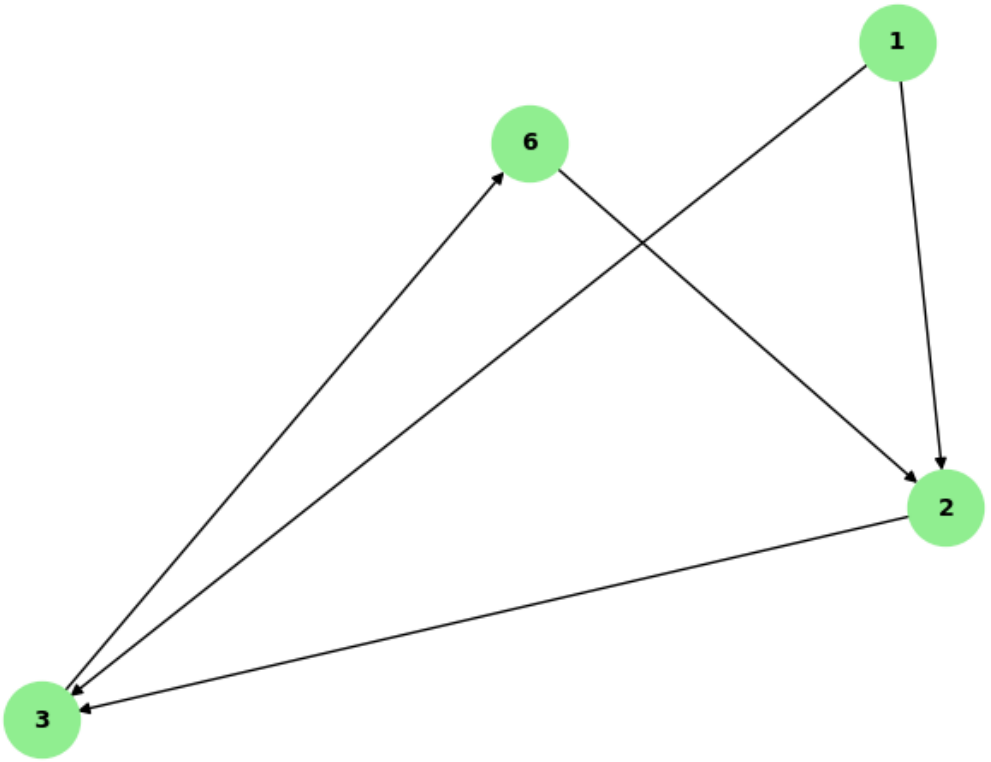
Código:

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

# Definir o grafo
G = nx.DiGraph()
G.add_edges_from([(1, 2), (1, 3), (2, 3), (3, 6), (4, 3), (4, 5), (5, 3), (6, 2)])

def dfs(graph, node, visited, subgraph):
    if node not in visited:
        visited.add(node)
        subgraph.add(node)
        for neighbor in graph[node]:
            dfs(graph, neighbor, visited, subgraph)

# Iniciar a busca em profundidade a partir de cada vértice não visitado
visited_nodes = set()
subgraphs = []

for node in G.nodes():
    if node not in visited_nodes:
        subgraph_nodes = set()
        dfs(G, node, visited_nodes, subgraph_nodes)
        subgraphs.append(subgraph_nodes)

# Criar e mostrar os gráficos dos componentes conectados (árvores de busca)
pos = nx.spring_layout(G, seed=42)

for idx, nodes in enumerate(subgraphs):
    subgraph = G.subgraph(nodes)
```

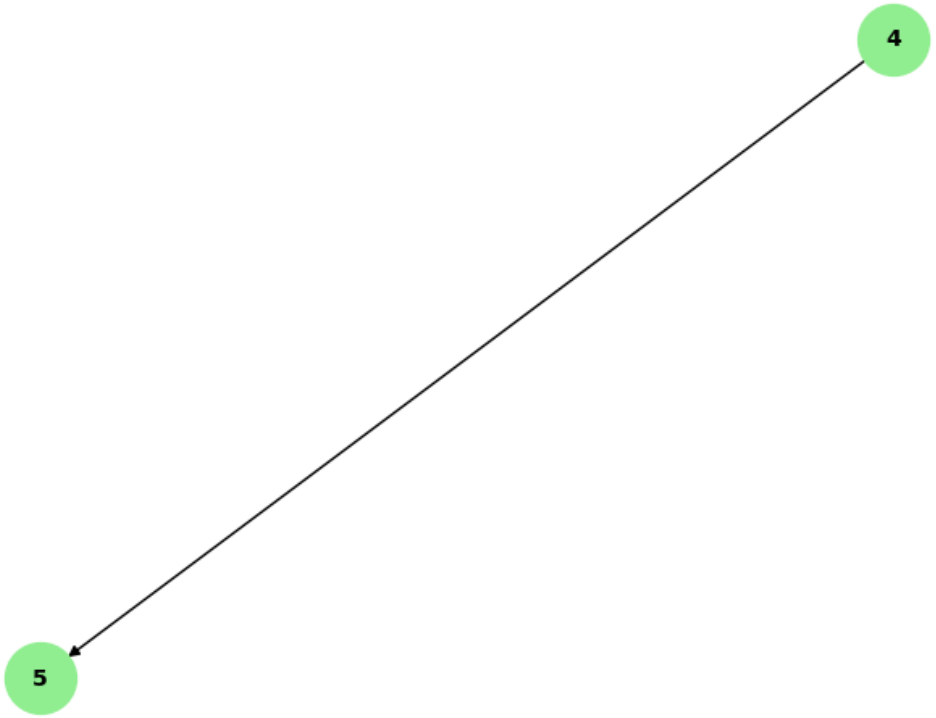
```
            nx.draw(subgraph,   pos,   with_labels=True,   node_size=1000,
node_color='lightgreen',        font_size=10,        font_color='black',
font_weight='bold')
    plt.title(f"Árvore de Busca {idx+1}")
    plt.show()
```



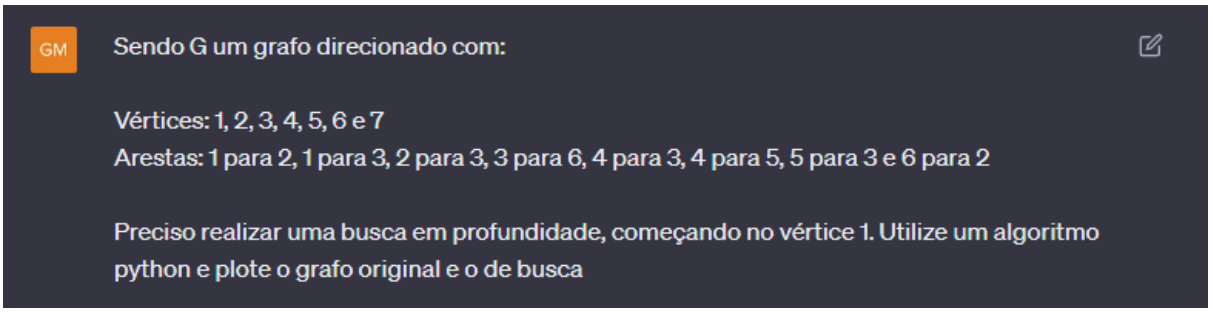
Árvore de Busca 1



Árvore de Busca 2



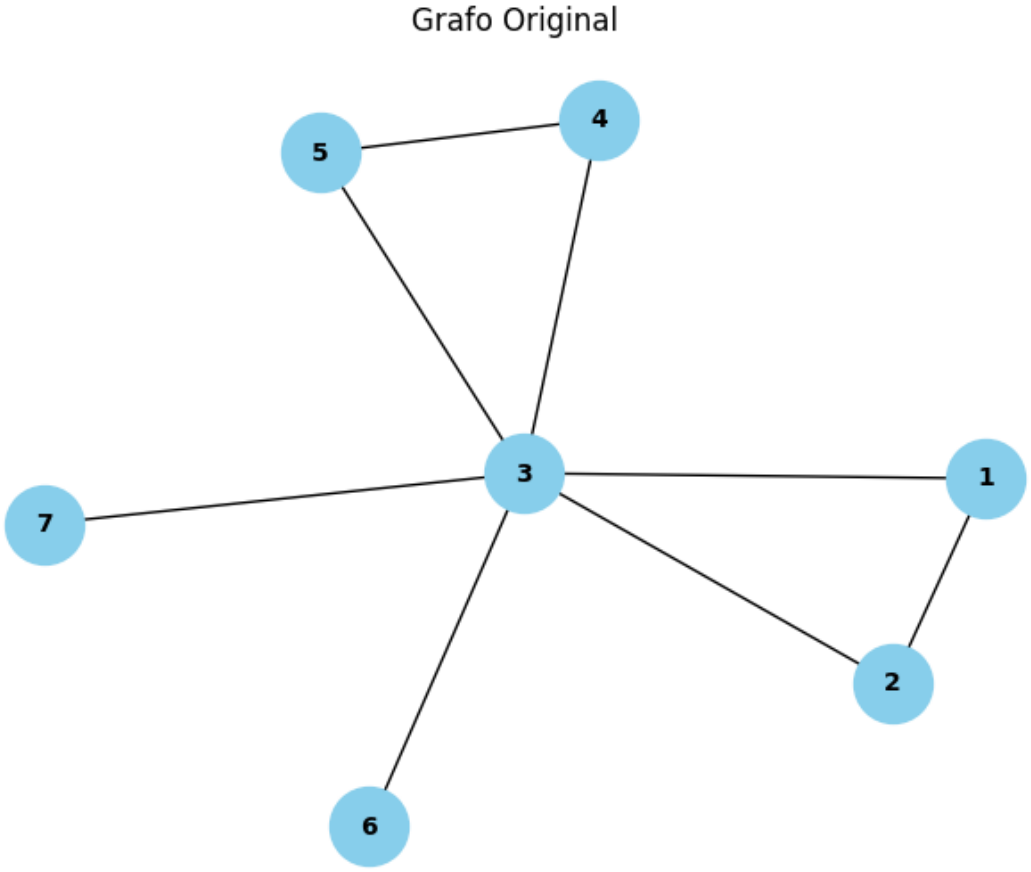
Exercício 05:



Código:

```
import networkx as nx
import matplotlib.pyplot as plt
from collections import deque

# Definindo o grafo
edges = [(1, 2), (1, 3), (2, 3), (3, 4), (3, 5), (3, 6), (3, 7), (4, 5)]
G = nx.Graph()
G.add_edges_from(edges)

def bfs_levels(graph, start_node):
    visited = set()
    queue = deque([(start_node, 0)])  # Adicionando o nível ao par (nó, nível)
    bfs_result = {}

    while queue:
        node, level = queue.popleft()
        if node not in visited:
            visited.add(node)
            if level not in bfs_result:
                bfs_result[level] = []
            bfs_result[level].append(node)
                queue.extend((neighbor, level + 1) for neighbor in graph[node] if neighbor not in visited)

    return bfs_result

# Realizando a busca em largura a partir do vértice 1
bfs_levels_result = bfs_levels(G, 1)
print("Resultado da busca em largura por níveis:", bfs_levels_result)
```

```
# Plotando o grafo original
pos = nx.spring_layout(G)
nx.draw(G, pos, with_labels=True, node_color='skyblue', node_size=1000,
font_size=10)
plt.title("Grafo Original")
plt.show()

# Plotando o subgrafo resultante da busca em largura por níveis
plt.figure(figsize=(10, 6))
for level, nodes in bfs_levels_result.items():
    y_offset = -level  # Ajuste para espaçar os níveis verticalmente
          pos_level  =  {node:  (idx,  y_offset)  for  idx,  node  in
enumerate(nodes)}
             nx.draw(G.subgraph(nodes),  pos_level,  with_labels=True,
node_color='lightgreen', node_size=1000, font_size=10)
plt.title("Resultado da Busca em Largura por Níveis")
plt.show()
```



Resultado da Busca em Largura por Níveis

Exercício 06:

GM

Sendo G um grafo não direcionado com:

Vértices: 1, 2, 3, 4, 5, 6, 7, 8 e 9
Arestas: 1 para 2, 1 para 4, 1 para 5, 2 para 3, 2 para 6, 4 para 7, 5 para 6, 5 para 8, 6 para 9, 7 para 8

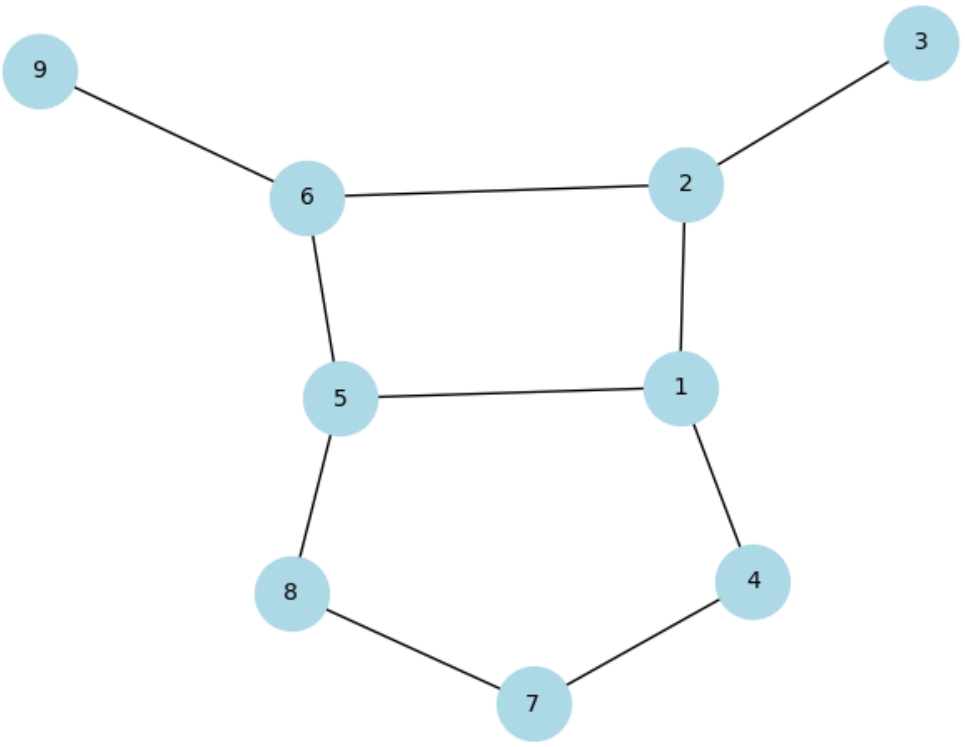
Fazer a busca completa com BFS e DFS no grafo G a partir do vértice 1. Plotar o grafo original e os 2 grafos de busca.

Código:

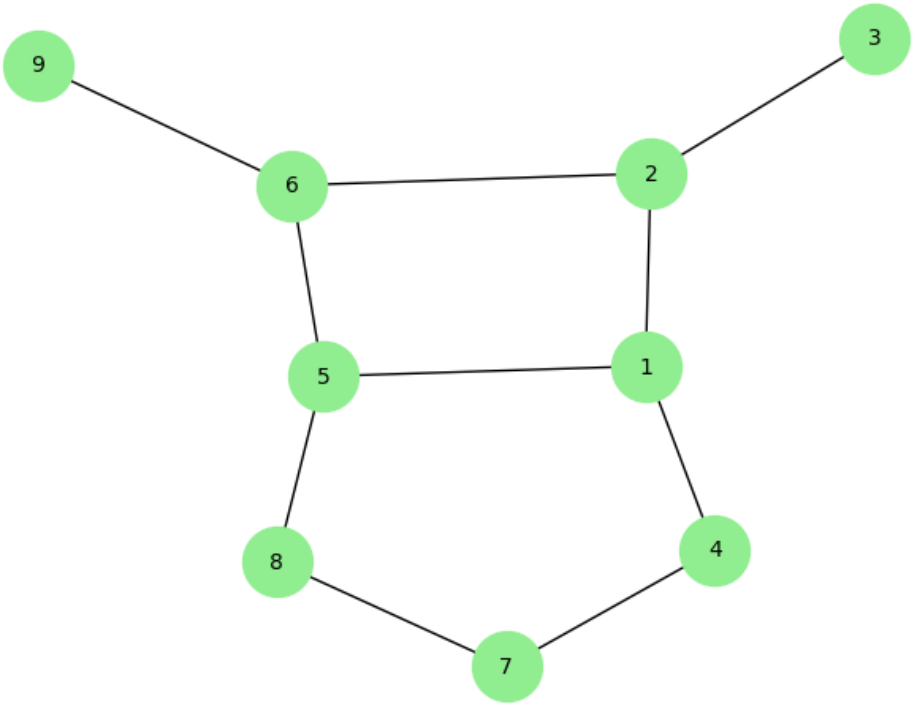
```
import networkx as nx
import matplotlib.pyplot as plt

# Criar o grafo G
G = nx.Graph()
G.add_edges_from([(1, 2), (1, 4), (1, 5), (2, 3), (2, 6), (4, 7), (5, 6), (5, 8), (6, 9), (7, 8)])

# Realizar a busca em largura (BFS)
bfs_nodes = list(nx.bfs_tree(G, source=1).nodes())

# Realizar a busca em profundidade (DFS)
dfs_nodes = list(nx.dfs_tree(G, source=1).nodes())

# Plotar o grafo original
pos = nx.spring_layout(G, seed=42)
nx.draw(G, pos, with_labels=True, node_size=1000, font_size=10, font_color='black', node_color='lightblue')
plt.title("Grafo Original")
plt.show()

# Plotar o grafo da busca em largura (BFS)
bfs_tree = G.subgraph(bfs_nodes)
nx.draw(bfs_tree, pos, with_labels=True, node_size=1000, font_size=10, font_color='black', node_color='lightgreen')
plt.title("Busca em Largura (BFS) a partir do vértice 1")
plt.show()

# Plotar o grafo da busca em profundidade (DFS)
dfs_tree = G.subgraph(dfs_nodes)
```

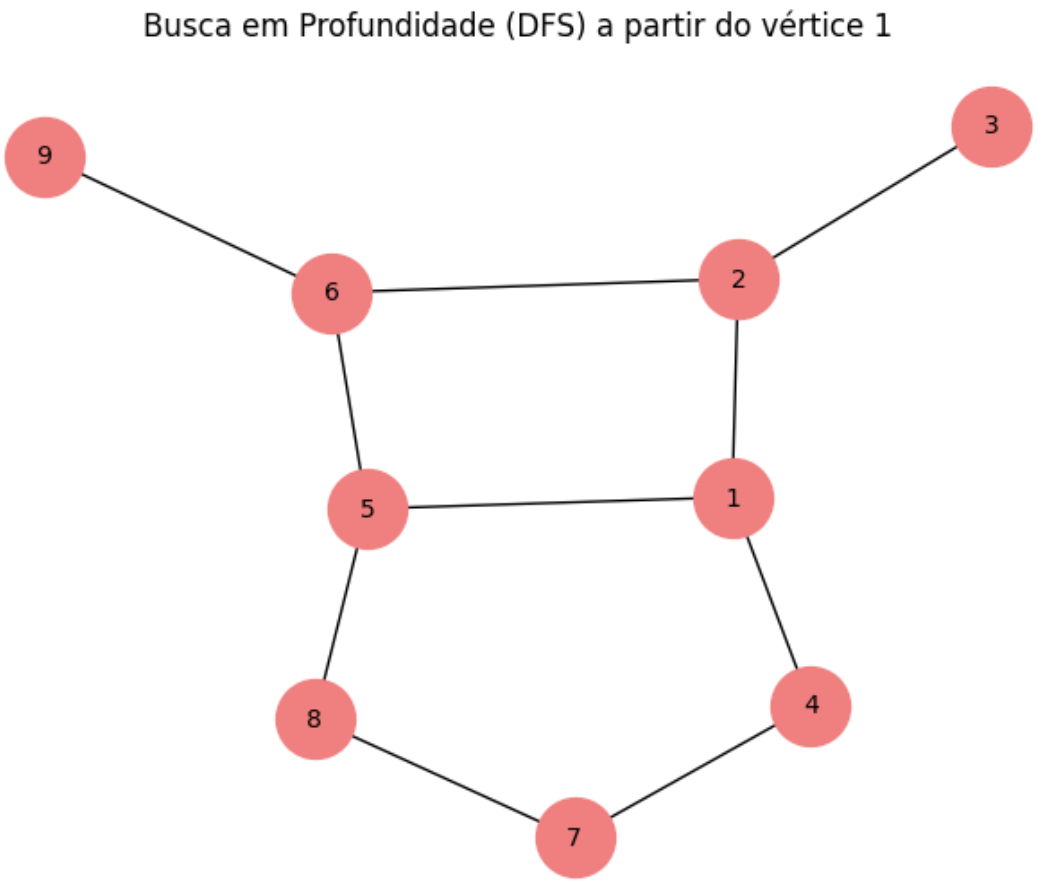
```
nx.draw(dfs_tree, pos, with_labels=True, node_size=1000, font_size=10,
font_color='black', node_color='lightcoral')
plt.title("Busca em Profundidade (DFS) a partir do vértice 1")
plt.show()
```

Grafo Original



Busca em Largura (BFS) a partir do vértice 1




Busca em Profundidade (DFS) a partir do vértice 1
9
3
6
2
5
1
8
4
7