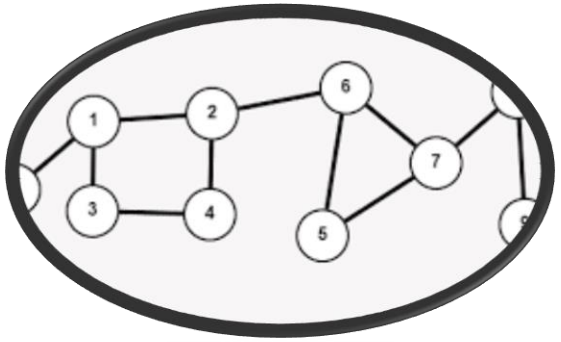
Introdução à Teoria dos Grafos: Cálculo de Conexidade e Distância em Grafos



Implementação em Python

wyden | Faculdade Martha Falcão

by Prof. Ranyere Lima

Tópicos

- Bases e Antibases
- Componentes Conexos e Fortemente Conexos
- Distância, Afastamento, Centro e Raio
- Aplicações e Algoritmos de Conexidade

Introdução aos Grafos

Conceitos Básicos

•**Grafo**: Uma estrutura composta de **nós** (vértices) e **arestas** (conexões entre os nós).

•**Nós**: Representam entidades (pontos, pessoas, objetos, etc.).

•**Arestas**: Representam a relação ou conexão entre dois nós.

Exemplo Visual:

•Um grafo com 4 nós conectados por 3 arestas.

.

Introdução aos Grafos

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

# Criando um grafo simples
G = nx.Graph()
G.add_edges_from([(1, 2), (2, 3), (3, 4)])

# Desenhando o grafo
nx.draw(G, with_labels=True, node_color='lightblue',
font_weight='bold')
plt.show()
```

add_edges_from adiciona as arestas (conexões) entre os nós.

by Prof. Ranyere Lima

Componente Conexo?

O que é um Componente Conexo?

•Um componente conexo é um conjunto de nós onde todos estão conectados diretamente ou indiretamente.

Visualização:

•Um grafo onde todos os nós podem ser alcançados a partir de qualquer outro nó.

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

# Criando um grafo conexo
G_conexo = nx.Graph()
G_conexo.add_edges_from([(1, 2), (2, 3), (3,
4), (4, 5)])

# Desenhando o grafo conexo
nx.draw(G_conexo, with_labels=True,
node_color='lightgreen', font_weight='bold')
plt.show()
```

Todos os nós estão conectados entre si de alguma forma, seja diretamente ou através de outros nós intermediários.

by Prof. Ranyere Lima

Grafo Desconexo

O que é um Grafo Desconexo?

•Em um **grafo desconexo**, há mais de um componente separado, e não há caminho entre alguns dos nós.

```
# Criando um grafo desconexo
G_desconexo = nx.Graph()
G_desconexo.add_edges_from([(1, 2), (2, 3), (4, 5)])

# Desenhando o grafo desconexo
nx.draw(G_desconexo, with_labels=True, node_color='orange',
font_weight='bold')
plt.show()
```

O grafo tem dois subconjuntos de nós que não se conectam entre si.

Componentes Fortemente Conexos (Grafos Direcionados)

O que é um Componente Fortemente Conexo?

•Em um grafo direcionado, um **componente fortemente conexo** é um subconjunto de nós onde todos podem ser alcançados de ida e volta por qualquer caminho.

```
# Criando um grafo fortemente conexo (direcionado)
G_direcionado = nx.DiGraph()
G_direcionado.add_edges_from([(1, 2), (2, 3), (3, 1), (3, 4), (4, 2)])

# Desenhando o grafo direcionado
nx.draw(G_direcionado, with_labels=True, node_color='pink', font_weight='bold', arrows=True)
plt.show()
```

As arestas têm direção (setas), e todos os nós no componente fortemente conexo podem ser alcançados em ambas as direções.

by Prof. Ranyere Lima

Distância e Afastamento

Definindo Distância

•A **distância** entre dois nós é o número mínimo de arestas que devem ser percorridas para ir de um nó a outro.

Afastamento:

•O **afastamento** de um nó é a maior distância entre ele e qualquer outro nó no grafo.

```
# Calculando a distância entre nós
distancia = nx.shortest_path_length(G_conexo, source=1, target=5)
print("Distância entre o nó 1 e o nó 5:", distancia)
```

•O código calcula a distância mínima entre dois nós no grafo conexo.

Centro e Raio

Centro de um Grafo:
•O **centro** do grafo é o nó com o menor afastamento em relação a todos os outros nós.
Raio do Grafo:
•O **raio** é a menor distância máxima de um nó para os outros.

```
# Calculando raio e centro do grafo
raio = nx.radius(G_conexo)
centro = nx.center(G_conexo)
print("Raio do grafo:", raio)
print("Centro do grafo:", centro)
```

O código calcula o nó central e o raio do grafo.

```
Raio do grafo: 2
Centro do grafo: [3]
```

Algoritmos de Conexidade - BFS

Algoritmo de Busca em Largura (BFS):

•O **BFS** percorre o grafo começando de um nó e explora todos os seus vizinhos antes de passar para o próximo nível de nós.

```
# Implementação do algoritmo BFS
def bfs(grafo, inicio):
    visitados = set()
    fila = [inicio]
    while fila:
        no = fila.pop(0)
        if no not in visitados:
            visitados.add(no)
            fila.extend(grafo[no] - visitados)
    return visitados
```

by Prof. Ranyere Lima

Algoritmos de Conexidade

BFS

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

# Implementação do algoritmo BFS
def bfs(grafo, inicio):
    visitados = set()
    fila = [inicio]
    ordem_visita = []  # Para armazenar a ordem em que os nós
foram visitados
    while fila:
        no = fila.pop(0)
        if no not in visitados:
            visitados.add(no)
            ordem_visita.append(no)  # Registra o nó na ordem de
visita
            fila.extend(grafo[no] - visitados)
    return ordem_visita  # Retorna a ordem de visita
```

```
# Criando o grafo com NetworkX
grafo = {
    'A': {'B', 'C'},
    'B': {'A', 'D', 'E'},
    'C': {'A', 'F'},
    'D': {'B'},
    'E': {'B', 'F'},
    'F': {'C', 'E'}
}

G = nx.Graph()

# Adiciona os nós e arestas ao grafo
for no, vizinhos in grafo.items():
    for vizinho in vizinhos:
        G.add_edge(no, vizinho)

# Realiza a busca em largura (BFS)
inicio = 'A'
ordem_visita = bfs(grafo, inicio)

# Desenha o grafo
pos = nx.spring_layout(G)  # Layout para os nós
plt.figure(figsize=(8, 6))
```

Algoritmos de Conexidade

BFS

```
# Desenha os nós e as arestas
nx.draw(G, pos, with_labels=True, node_color='lightblue',
node_size=3000, font_size=15, font_weight='bold',
edge_color='gray')

# Destaca os nós na ordem visitada pelo BFS
colors = ['red', 'orange', 'yellow', 'green', 'blue', 'purple']
for idx, no in enumerate(ordem_visita):
    nx.draw_networkx_nodes(G, pos, nodelist=[no],
node_color=colors[idx % len(colors)], node_size=3000)

plt.title("Busca em Largura (BFS) - Ordem de Visitação")
plt.show()
```

PERGUNTAS ?

Atividade....

by Prof. Ranyere Lima

Exercício Prático

- Crie um grafo com 5 nós e dois componentes desconexos.
- Implemente um código Python para calcular a distância entre dois nós e encontrar o centro do grafo.

Desafio:

•Adicione um nó ao grafo que conecte todos os outros nós e faça com que o grafo se torne fortemente conexo.

•Use o algoritmo BFS para verificar a conectividade.

Obrigado.
Até Próxima Aula !

by Prof. Ranyere Lima