

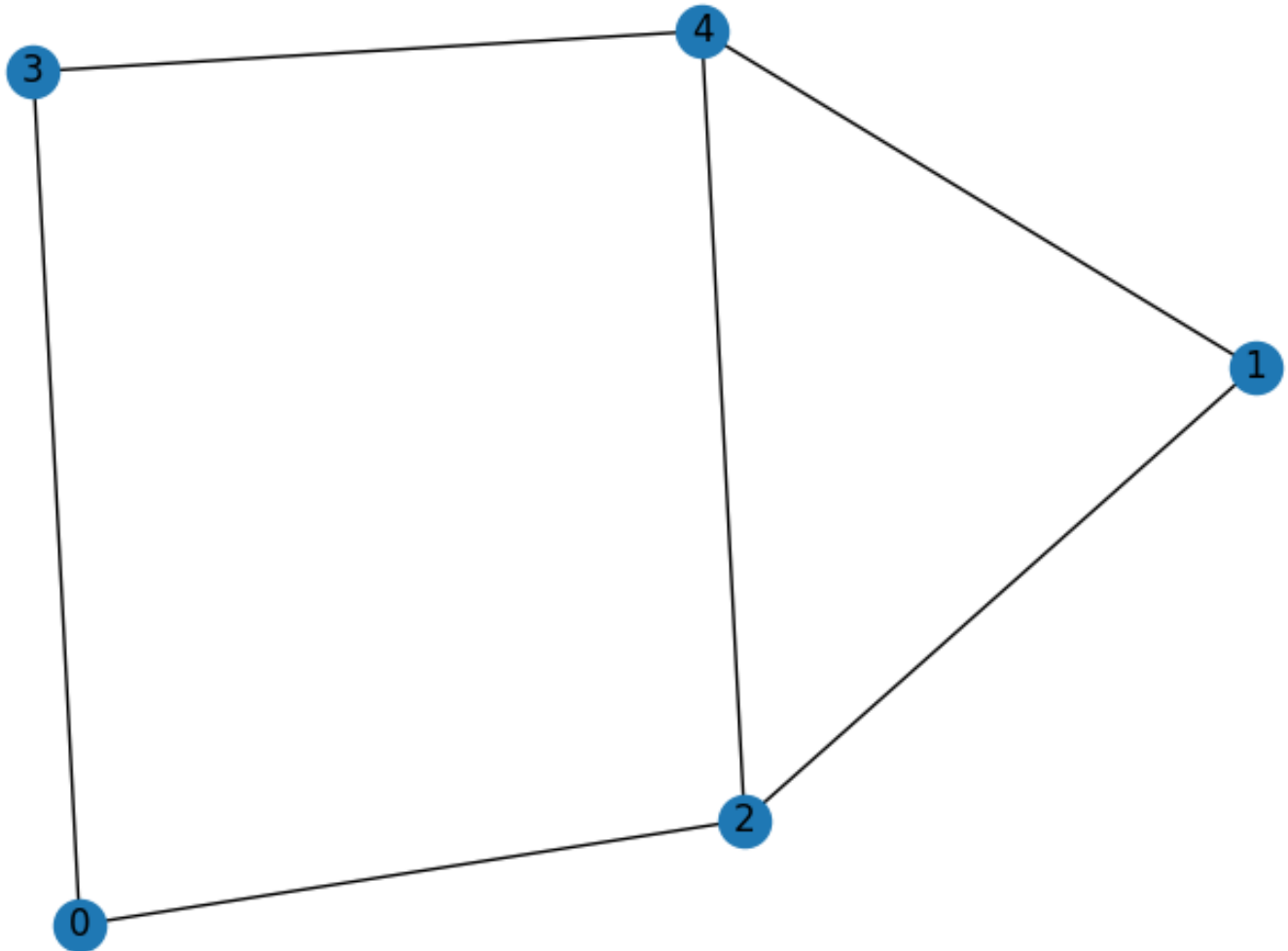
Aula 03



Rodrigo G. Araújo

26/07/2019

Introdução a Teoria dos Grafos



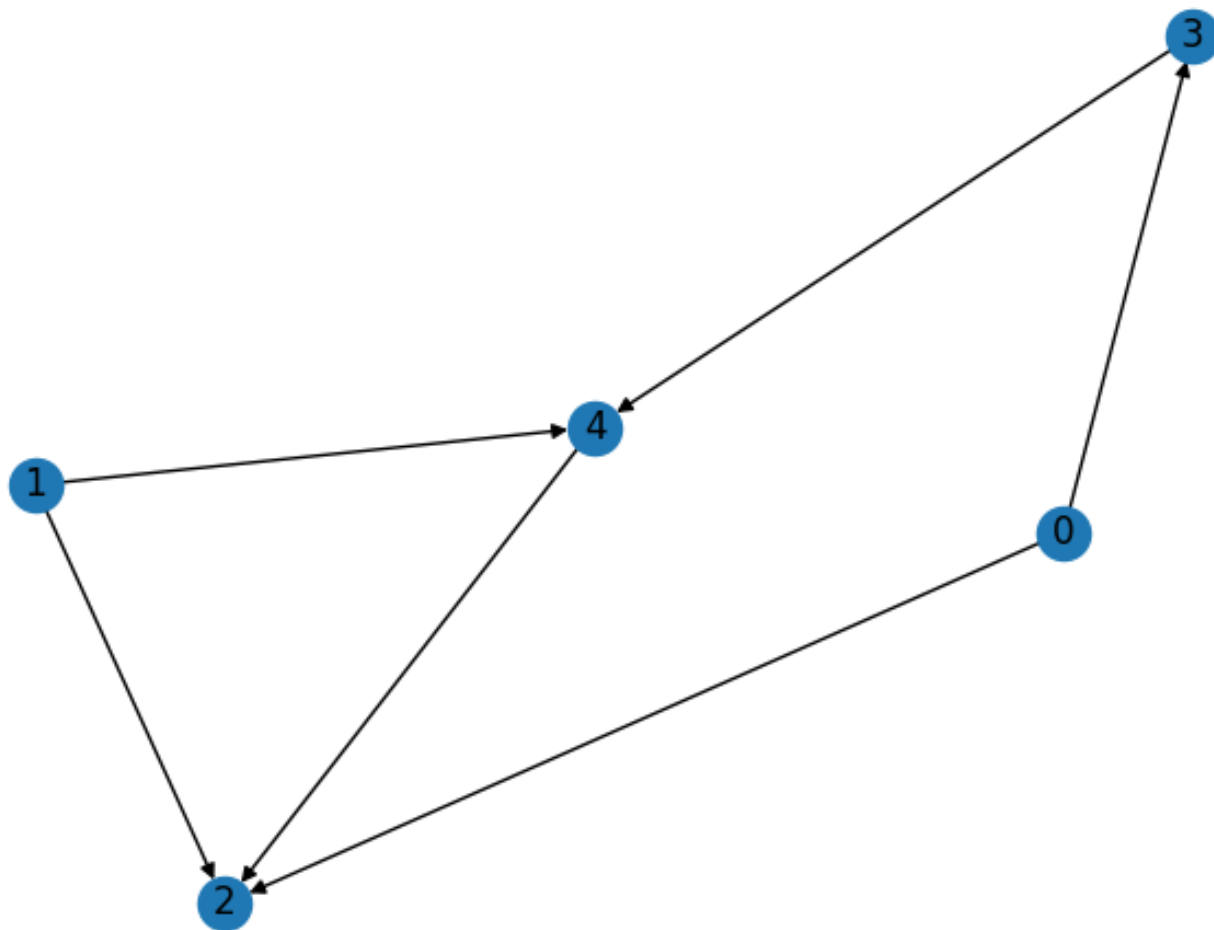
Lista de Adjacência

```
lista_adj = {  
    0: [2, 3],  
    1: [2, 4],  
    2: [1, 4],  
    3: [0, 4],  
    4: [1, 2, 3]  
}
```

Matriz de Adjacência

```
matriz_adj = [  
    [1, 0, 1, 1, 0],  
    [0, 1, 1, 0, 1],  
    [0, 1, 1, 0, 1],  
    [1, 0, 0, 1, 1],  
    [0, 1, 1, 1, 1],  
]
```

Grafo Direcionado

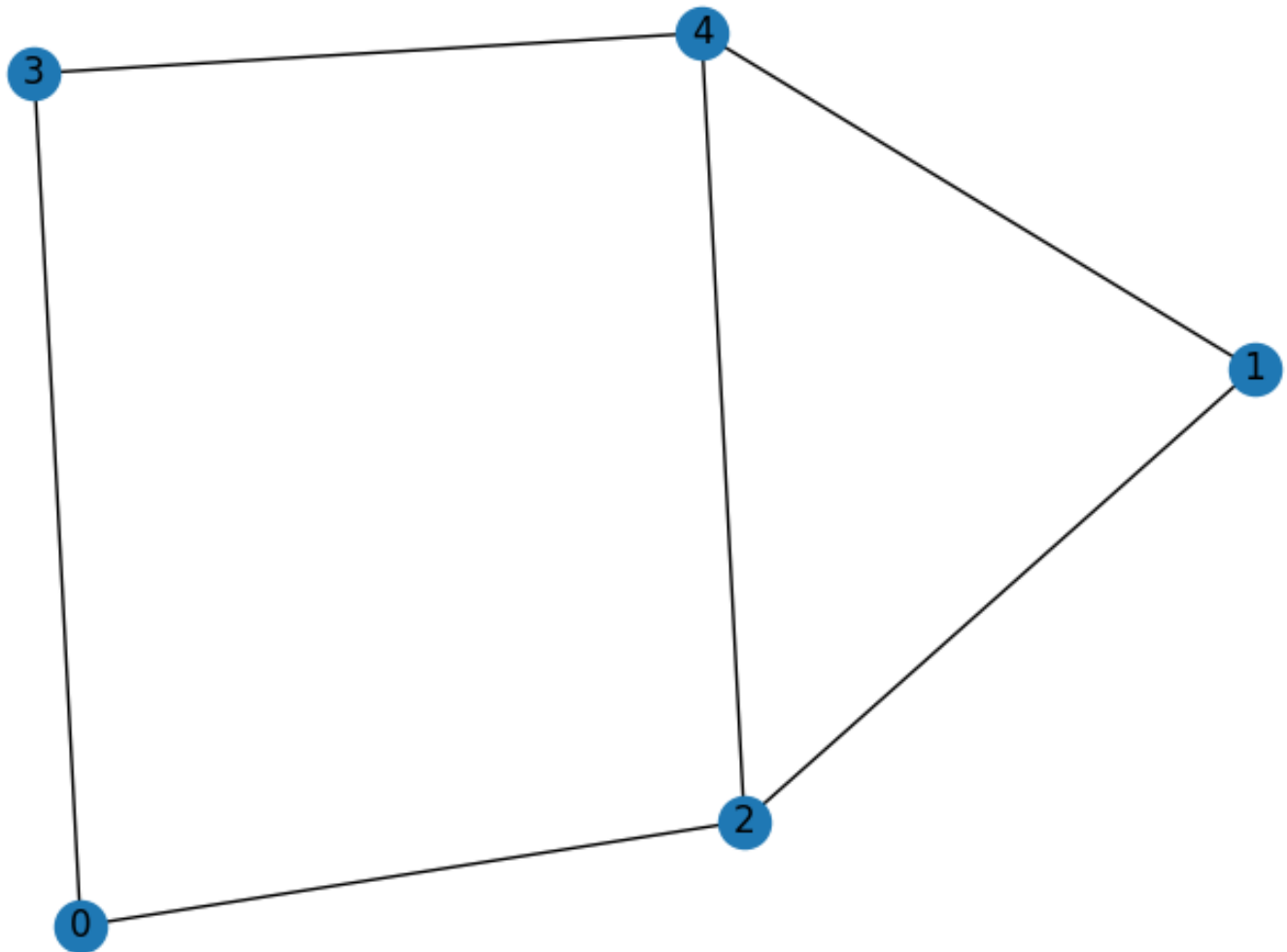


Problema Simples

```
lista_adj = {  
    0:[2, 3],  
    1:[2, 4],  
    2:[],  
    3:[4],  
    4:[2]  
}
```

```
qtd_entradas = 0  
vertice_buscado = int(input())  
for vertice in lista_adj:  
    if vertice_buscado in vertice:  
        qtd_entradas += 1  
print(qtd_entradas)
```

BFS



BFS

```
from collections import deque
```

```
def initialize(dist, graph):  
    for i in range(vertices):  
        dist[i] = -1
```

```
    for i in range(vertices):  
        graph[i] = set()
```


BFS

```
def populate(graph, arestas, bi=False):  
    for _ in range(arestas):  
        v, w = map(int, input().split())  
        graph[v].add(w)  
        if bi:  
            graph[w].add(v)
```

BFS

```
def bfs(start, graph, dist):  
    fila = deque()  
    dist[start] = 0  
    fila.append(start)  
    while fila:  
        v = fila.popleft()  
        for w in graph[v]:  
            if dist[w] == -1:  
                dist[w] = dist[v] + 1  
                fila.append(w)
```

BFS

```
graph, dist = dict(), dict()
initialize(dist, graph)
vertices, arestas = map(int, input().split())
populate(graph, arestas)

bfs(0)
print(dist)
```

Exercicios

A - Love Triangle

```
n = int(input())
linha = [1]*n
for _ in range(n-1):
    nova_linha = list()
    for i in range(1, n+1):
        nova_linha.append(sum(linha[:i]))
    linha = nova_linha
print(linha[-1])
```

Exercicios

C - Love Triangle

```
vertices = int(input())
avioes = list(map(lambda x: int(x)-1, input().split()))
for vertice in range(vertices):
    if avioes[avioes[avioes[vertice]]] == vertice:
        print('YES')
        exit(0)
print('NO')
```

Exercicios

C - Metro

```
from collections import deque

def initialize(dist, graph, vertices):
    for i in range(vertices):
        dist[i] = -1

    for i in range(vertices):
        graph[i] = set()
```

Exercicios

C - Metro

```
def bfs(start, graph, dist):  
    fila = deque()  
    dist[start] = 0  
    fila.append(start)  
    while fila:  
        v = fila.popleft()  
        for w in graph[v]:  
            if dist[w] == -1:  
                dist[w] = dist[v] + 1  
                fila.append(w)
```

Exercícios

C - Metro

```
graph, dist = dict(), dict()
vertices, final = map(int, input().split())
initialize(dist, graph, vertices)

# ida
estacoes = list(map(int, input().split()))
for i in range(vertices):
    for j in range(i+1, vertices):
        if estacoes[i] == 1 and estacoes[j] == 1:
            graph[i].add(j)
```


Exercícios

C - Metro

```
# volta
estacoes = list(map(int, input().split()))
for i in range(vertices-1, -1, -1):
    for j in range(i-1, -1, -1):
        if estacoes[i] == 1 and estacoes[j] == 1:
            graph[i].add(j)

bfs(0, graph, dist)
print('NO' if dist[final-1] == -1 else 'YES')
```