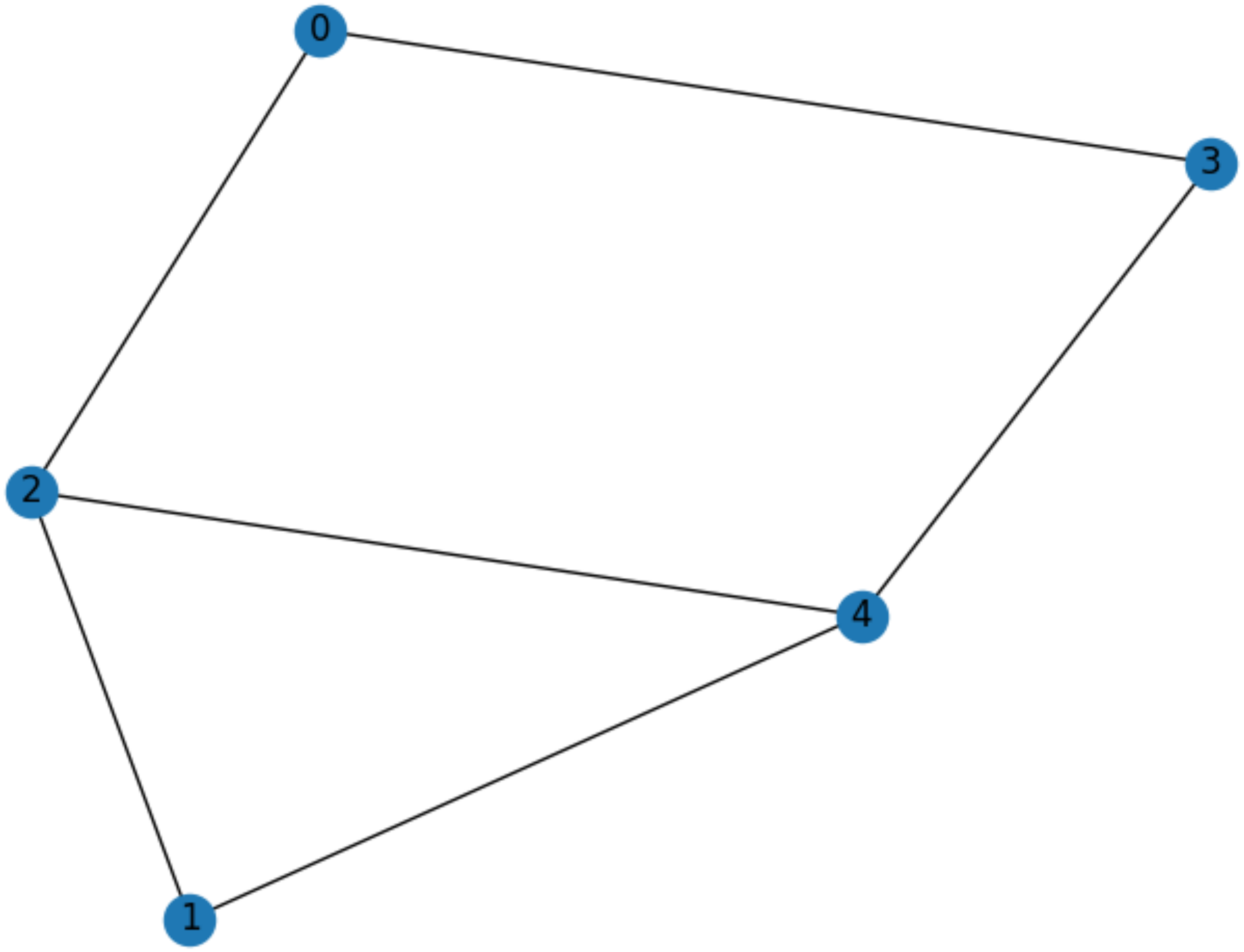
Aula 03

MARATONA IESB
PROGRAMAÇÃO

Rodrigo G. Araújo

26/07/2019

Introdução a Teoria dos Grafos



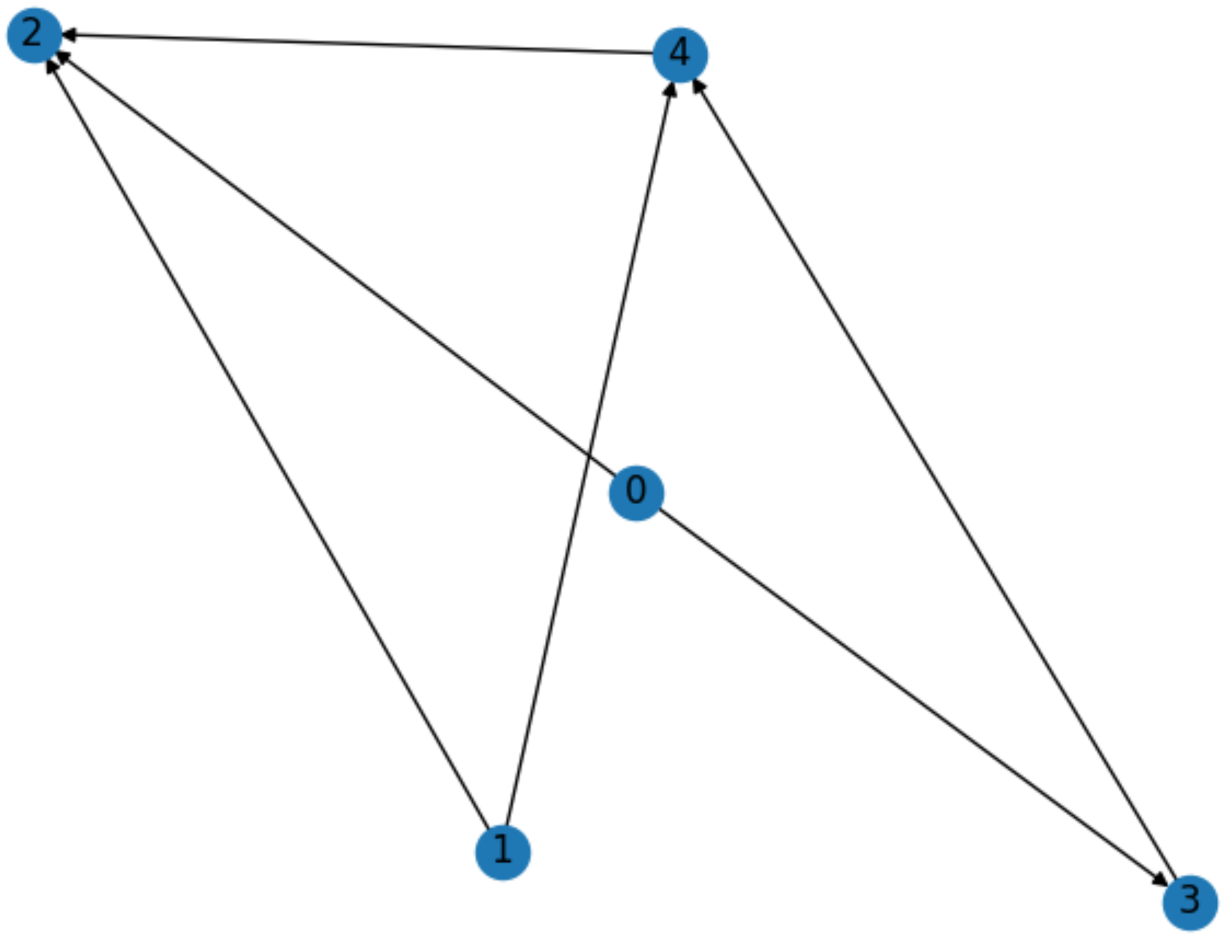
Lista de Adjacência

```
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

```python
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

```
from collections import deque

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

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

BFS

```python
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[i] == -1:
                            dist[w] = dist[v] + 1
                            fila.append(w)
```

BFS

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


bfs(0)
print(dist)
```