به نام خدا

دانشگاه صنعتی شاهرود

Shahrood University of Technology

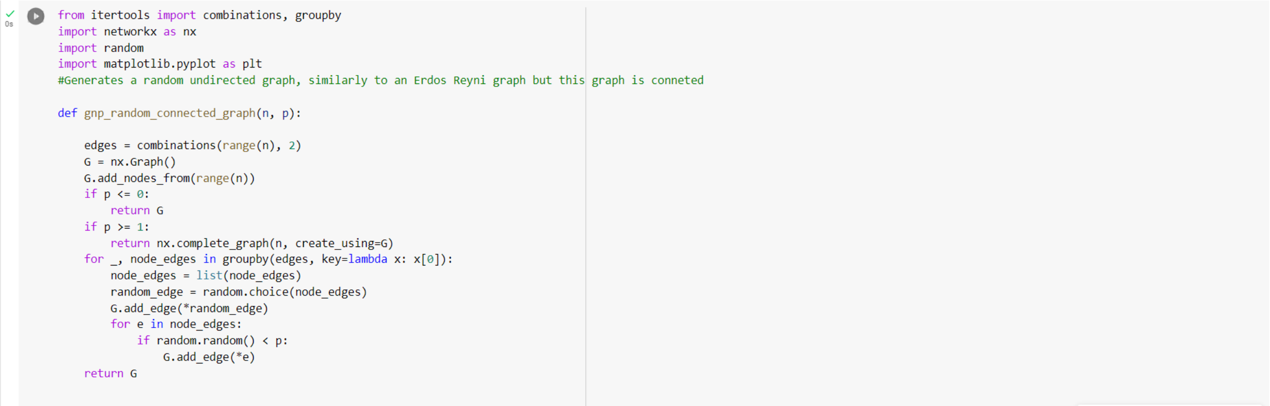
دانشکده مهندسی کامپیوتر و فناوری اطلاعات

عنوان تمرین : توضیح پیاده سازی گراف تصادفی با تعداد نودهای مختلف

استاد: دکتر محسن رضوانی

نویسنده: مریم درویشیان

تاریخ:1402/01/05

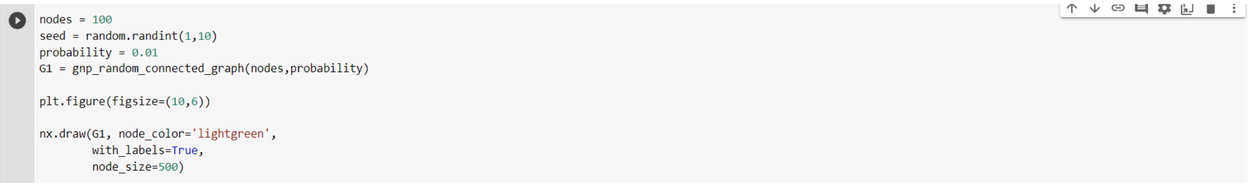


```python
from itertools import combinations, groupby
import networkx as nx
import random
import matplotlib.pyplot as plt
#Generates a random undirected graph, similarly to an Erdos Reyni graph but this graph is conneted

def gnp_random_connected_graph(n, p):

    edges = combinations(range(n), 2)
    G = nx.Graph()
    G.add_nodes_from(range(n))
    if p <= 0:
        return G
    if p >= 1:
        return nx.complete_graph(n, create_using=G)
    for _, node_edges in groupby(edges, key=lambda x: x[0]):
        node_edges = list(node_edges)
        random_edge = random.choice(node_edges)
        G.add_edge(*random_edge)
        for e in node_edges:
            if random.random() < p:
                G.add_edge(*e)
    return G
```

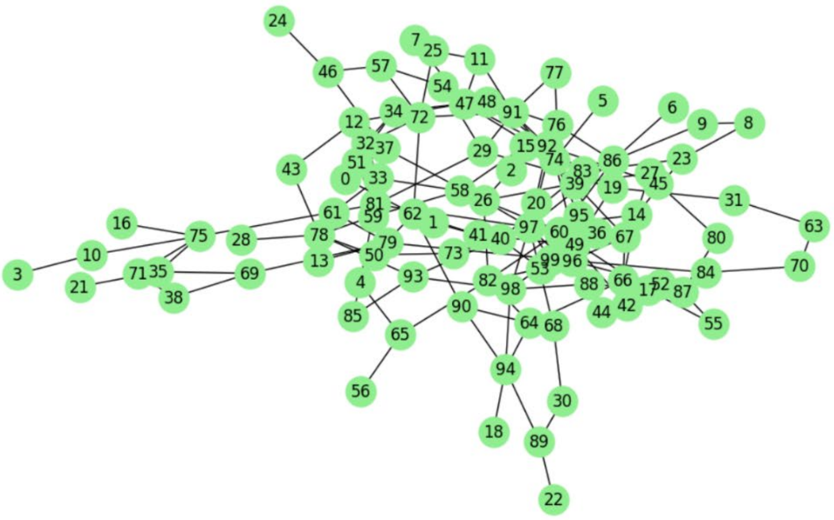
In this section, I write code to generate a random connected graph with a minimum of one degree between vertices. Similar to an Erdos Reyni graph which is connected. I use import combination and group by from itertools and other calls of python's libraries. Then, a function wrote as a gnp_random_connected_graph. I set at least one edge for each node using the result from combinations(range(n_nodes), 2). This line causes to create a connected random graph.

The function input gets the number of nodes and the existence probability of each node. In the next step, I call the function with the indicated number of vertices and stable probability. To display a random graph I utilized the matplot library with colour, size and identified labels.

I called a function of gnp_random_connected_graph with distinct and defined probability in this cell. Then by using matplotlib library functions I plot each of the graphs. For example:



```python
nodes = 100
seed = random.randint(1,10)
probability = 0.01
G1 = gnp_random_connected_graph(nodes,probability)

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

nx.draw(G1, node_color='lightgreen',
        with_labels=True,
        node_size=500)
```


24
7
25
11
57
46
77
54
5
47
48
91
6
9
8
34
72
12
76
15
92
29
32
37
86
23
74
83
27
43
51
2
0
33
45
39
19
58
31
26
20
81
16
61
62
14
95
63
59
1
75
97
36
28
78
41
40
60
67
80
49
10
79
73
99
96
50
13
53
3
71
35
69
84
70
93
4
82
88
66
52
21
98
17
87
38
90
42
44
85
64
68
55
65
94
56
30
18
89
22