Complex Networks: Quiz #4

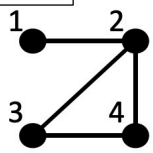
Due on Dec 16, 2018

RUOPENG XU
18M38179

Problem 1

What happens when a walker keeps walking for a long time on a graph?
pi(t): probability that the walk is at vertex i at time t
1. graph 1: find AD^{-1}

graph 1



$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

$$p_i(t) = \sum_j \frac{A_{ij}}{k_j} p_j(t-1)$$

$$\mathbf{p}(t) = \mathbf{A}\mathbf{D}^{-1}\mathbf{p}(t-1)$$

$$\mathbf{D}^{-1} = \begin{pmatrix} 1/k_1 & 0 & 0 & \cdots \\ 0 & 1/k_2 & 0 & \cdots \\ 0 & 0 & 1/k_3 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

Answer 1

According to the defination of D $D^{-1} =$
$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{3} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & \frac{1}{2} \end{bmatrix}$

Thus, $AD^{-1} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{3} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & \frac{1}{2} \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{3} & 0 & 0 \\ 1 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & \frac{1}{3} & 0 & \frac{1}{2} \\ 0 & \frac{1}{3} & \frac{1}{2} & 0 \end{bmatrix}$

Problem 2

2. graph 1: find p0(∞),p1(∞),p2(∞), and p3(∞)

Answer2

When p -> ∞, the $p_i(\infty)$ is the probability of each node in the network. According to the slides: **probability is proportional to the degree of the vertex** and the sum of $p_i = 1$
Thus,

p0(∞)+p1(∞)+p2(∞)+p3(∞)=1,

make p0(∞)=x,p1(∞)=3x, p2(∞) = 2x,p3(∞) = 2x

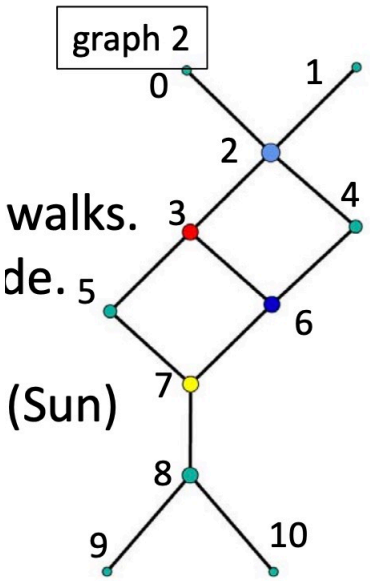
x + 3x + 2x + 2x = 1

$x = \frac{1}{8}$

As a result, $p0(\infty) = \frac{1}{8}, p1(\infty) = \frac{3}{8}, p2(\infty) = \frac{1}{4}, and, p3(\infty) = \frac{1}{4}$

Problem 3

3. graph 2: make a program of simulating random walks. show its results and answer the most visited node




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

G = nx.Graph()
G.add_nodes_from(range(0,10))
G.add_edges_from([(0,2),(1,2),(2,3),(2,4),(3,5),(3,6),(4,6),(5,7),(6,7),(7,8),(8,9),(8,10)])
plt.figure(figsize=(5, 5))
nx.draw_spring(G, node_size=400, node_color='red', with_labels=True, font_weight='bold')
A = nx.adjacency_matrix(G).todense()
A = np.array(A, dtype = np.float64)
c = np.array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]) # counter
walkLength = 100000 # random walk length
id = 0 # starting node (0-th)
for k in range(walkLength):
  p = A[id,:]
  id = np.random.choice(np.flatnonzero(p == p.max())) # random choice
  visited.append(id)
  c[id] = c[id] + 1
print(c)
```

Simulate we travel 10000 steps and try three times, according to the result:

```
[ 4201  4289 16858 12600  8252  8285 12502 12453 12355  4045  4160]

[ 4294  4243 16836 12526  8287  8275 12580 12508 12309  4046  4096]

[ 4006  4100 16334 12456  8280  8425 12401 12572 12840  4282  4304]
```

Thus, the most visited node is node 2