سوال اول:

۱. ضریب خوشه‌بندی:

ابتدا فرمول کلی را می‌نویسیم و سپس از روی آن اجزا را توضیح می‌دهیم:

$$C = \frac{\#Triangles \times 3}{\#ConnectedTriples}$$

$$-> C = \frac{\boxed{\frac{1}{4}nc(\frac{1}{2}c-1)} \times 3}{\frac{1}{2}nc(c-1) + nc^2p + \frac{1}{2}nc^2p^2}$$

---

1. $\frac{1}{4}nc(\frac{1}{2}c-1)$: تعداد مثلث‌های موجود در دایره‌ی اولیه‌ی نیومن-واتس

برای شمارش تعداد مثلث‌ها ابتدا باید مثلث‌های موجود در مدل نیومن-واتس را پیدا کرد. مثلث‌ها یا با استفاده از میانبرها[1] ساخته می‌شوند یا مثلث‌هایی بوندند که در دایره‌ی اولیه‌ی نیومن واتس وجود داشتند. تعداد میانبرها معادل $\frac{1}{2}ncp$ است و انتخاب ۲ گره از تمام گره‌ها، یعنی انتخاب $\binom{n}{2}$ تمامی جفت گره‌ها را به ما می‌دهد. حال برای محاسبه‌ی احتمال وجود میانبر، به مقدار زیر می‌رسیم:

$$\frac{\frac{1}{2}ncp}{\binom{n}{2}} = \frac{cp}{n-1} \approx \frac{cp}{n}$$

این تنها یک ضلع از مثلث است. دو ضلع دیگر باید از روی دایره‌ی اولیه گرفته شود که چون در این دایره مسیر به طول ۲، با n متناسب است، پس تعداد مثلث‌های با تنها یک میانبر مقدار زیر است:

$$n \times \frac{cp}{n} = cp$$

که همانگونه که از فرمول پیداست، با n متناسب نیست و عددی ثابت است. پس به ازای n های بزرگ می‌توان از این مقدار چشم‌پوشی کرد. تعداد مثلث‌هایی که دو یا سه ضلع میانبر ساخته می‌شوند نیز قابل چشم‌پوشی هستند چرا که تعداد آن‌ها بسیار کم است. در نتیجه تنها مثلث‌هایی

---

1 . Shortcut

که در دایره‌ی اولیه وجود داشتند و اصلا میانبری ندارند، برایمان حائز اهمیت هستند که با فرمول روبرو قابل محاسبه است:[2] $\frac{1}{4}nc(\frac{1}{2}c-1)$

---

2. nc^2p: با یک میانبر

تعداد میانبرها را در قسمت قبل گفتیم: $\frac{1}{2}ncp$

هر میانبر به دو گره متصل می‌شود که هر گره c یال غیر میانبر دارد. پس تعداد کل ۳تایی‌ها می‌شود:

$$2\times\frac{1}{2}ncp\times c = nc^2p$$

به این علت ضربدر ۲ می‌شوند که دو گره در انتهای هر یال است.

---

3. $\frac{1}{2}nc^2p^2$: با دو میانبر

توزیع پوآسون بر روی میانبرها با میانگین cp، و با درنظر گرفتن گره‌ای ثابت به عنوان مرکز برای متصل کردن دو میانبر دیگر، ما را به رابطه‌ی $\frac{1}{2}\times cp\times cp = \frac{1}{2}c^2p^2$ می‌رساند چرا که برای یک میانبر دیگر نیز میانگین cp را داریم اما چون یال‌ها تکراری شمردیم، تقسیم بر ۲ نیز باید بکنیم.

---

4. $\frac{1}{2}nc(c-1)$: بدون میانبر

تمام ۳تایی‌هایی که بر روی دایره‌ی ابتدایی هستند، با رابطه‌ی $\frac{1}{2}nc(c-1)$ قابل محاسبه هستند.[3] چرا که با انتخاب ۲ مسیر از c مسیر برای به جلو رفتن، با تعداد n گره در مجموع به $\binom{c}{2}\times n$ خواهیم رسید.

۲. میانگین طول مسیر:

اگر s تا میانبر داشته باشیم، پس $2s$ تا مسیر احتمالی داریم، چرا که یک میانبر هم می‌تواند از یک سمت به سمت دیگر باشد و هم می‌تواند از سمت دیگر به این سمت باشد، پس هر میانبر، ۲ تا مسیر را می‌تواند مشخص کند. از این رو میانگین فاصله‌ی بین گره‌های میانبرها با رابطه‌ی $\frac{n}{2s}$

---

[2] . خط ۵۵۸ کتاب نیومن، خط ۶ قسمت ۱۵.۱.۲

[3] . صفحه‌ی ۵۵۴ کتاب نیومن، خط ۱۰، قسمت ۱۵.۱

بدست خواهد آمد. پس برای بدست آوردن میانگین طول مسیر می‌توان فهمید که با $2s$ ارتباط دارد، از این روی رابطه‌ی میانگین طول مسیر را می‌توان به صورت تابعی از $2s$ به شکل زیر نوشت چرا که برای پیدا کردن کوتاه‌ترین مسیر باید از روی میانبرها نیز رد شد:

$$\ell = \frac{2n}{c} F(2s)$$

علت پدیدار شدن $\frac{2}{c}$ نیز این است که به ازای دو برابر کردن c، مسیر ما نصف خواهد شد چرا که اگر میانبری وجود داشته باشد، عملا نصف گراف را نیاز نیست عبور کنیم و با استفاده از میانبر، سریع‌تر به گره بعدی خواهیم رسید و عملا سرعتمان $\frac{c}{2}$ برابر می‌شود. برای محاسبه‌ی تابع $F(2s)$ با استفاده از ؟تقریب سری‌ها؟ به عبارت $F(x) = \frac{lnx}{x}$ می‌رسیم.[4] از آنجایی هم که می‌دانستیم $s = \frac{1}{2}ncp$ است، پس برای میانگین طول مسیر به فرمول زیر می‌رسیم:

$$\ell = \frac{n}{c} \times F(2s) = \frac{n}{c} \times F(ncp) = \frac{n}{c} \times \frac{\ln(ncp)}{ncp} = \frac{\ln(ncp)}{c^2 p}$$

$$-> \ell = \frac{\ln(ncp)}{c^2 p}$$

سوال دوم:

$$L = \frac{1}{c^2 \times p} + 1 + 1 + 1 + \frac{1}{c^2 \times p}$$

Then: $L = \frac{2 \times (c^2 \times p + 1)}{c^2 \times p}$

سوال سوم:

قسمت اول:

احتمال اینکه تمام یال‌های متصل به گره دلخواهمان حذف شود: p^c

مقدار بالا احتمال این است که یال متصل به یک گره rewire شود. این مقدار از آنجایی حساب شد که چون در ابتدا با احتمال $\frac{1}{n}$ متصل می‌شوند، پس هر گره با احتمال $1 - \frac{1}{n}$ متصل نمی‌شوند. در نتیجه برای کل شبکه خواهیم داشت: $(1 - \frac{1}{n})^{ncp}$ که این مقدار را می‌توان با e^{-cp} تقریب زد.

[4] . صفحات ۵۶۳ و ۵۶۴ کتاب نیومن

$$\text{تعداد مسیرکوتاه‌ها}: p = e^{-cp}\frac{cp^s}{s!}$$
$$if\ s = 0 \rightarrow p = e^{-cp}$$
$$Then: p^c \times e^{-cp} = (pe^-p)^c$$

قسمت دوم:

به ازای مقادیر داده شده، آن مقادیر را در فرمول قسمت قبل گذاشته و محاسبات را انجام می‌دهیم:

$$p^c \times e^{-cp} = (pe^{-p})^c = 9.4 \times 10^{-13}$$
$$\rightarrow N = \frac{1}{9.4 \times 10^{-13}} = 10^{12}$$

سوال چهارم:

مجموعه داده‌ای که من انتخاب کردم، Wiki-Vote است که اطلاعات مربوط به آن در شکل زیر (شکل ۱) قابل مشاهده است:

شکل ۱

Name	Type	Nodes	Edges	Description
wiki-Vote	Directed	7,115	103,689	Wikipedia who-votes-on-whom network

کد مربوط به این سوال را در شکل زیر (شکل ۲) مشاهده می‌کنیم. در خط ۵ ابتدا گراف را با توجه به لیست یالی‌ای که از آدرس داده شده دانلود کردم، می‌سازیم.

در خط ۷ حل قسمت A را مشاهده می‌کنیم که صرفا از ما تعداد نودها را خواسته بود اما تعداد یال‌ها را نیز برای اطمینان بیشتر پرینت گرفتم که در شکل ۳ قابل مشاهده است.

در خطوط ۱۰ و ۱۱ حل قسمت B را مشاهده می‌کنیم که از ما تعداد یال‌های جهتدار و غیر جهتدار خواسته شده بود که خروجی در شکل ۳ قابل مشاهده است. البته با توجه به اینکه داده‌ای که من داشتم تنها از نوع جهتدار است، پس مقدار یال‌های جهتدار برای ما حائز اهمیت است.

در خطوط ۱۴ و ۱۵ حل قسمت C را مشاهده می‌کنیم که تعداد نودها با درجه ورودی و درجه خروجی صفر خواسته شده بود که با تابع CntInDegNodes(0) برای درجه ورودی و با تابع CntOutDegNodes(0) برای درجه خروجی در شکل ۳ مقدار آنها قابل مشاهده است.

در خط ۱۸ حل قسمت D را مشاهده می‌کنیم که میانگین ضریب خوشگی را از ما خواسته بود و با تابع GetClustCf() قابل مشاهده است.

در خط ۲۱ حل قسمت E را مشاهده می‌کنیم که قطر شبکه از ما خواسته شده بود که با تابع GetBfsFullDiam(100) قابل مشاهده است. عدد ۱۰۰ در اینجا به معنی انتخاب ۱۰۰ گره برای حل بی‌فااس در کل گراف است که البته می‌توان تمام ۷۰۰۰ گره را برای محاسبه‌ی دقیق داد اما برای محاسبه‌ی بهینه ۱۰۰ گره کافیست (تقریبا ۱٪ از کل گره‌ها) به علت لود عملیاتی.

شکل ۲

```
1  import os
2  import snap
3
4
5  graph = snap.LoadEdgeList(snap.TNGraph, "Wiki-Vote.txt", 0, 1)
6  # A:
7  print('#Nodes: {NODES}\n#Edges: {EDGES}'.format(NODES=graph.GetNodes(), EDGES=graph.GetEdges()))
8
9  # B:
10 print('#Directed Edges: {}'.format(graph.CntUniqDirEdges()))
11 print('#UnDirected Edges: {}'.format(graph.CntUniqUndirEdges()))
12
13 # C:
14 print('#In-Degree Nodes: {}'.format(graph.CntInDegNodes(0)))
15 print('#Out-Degree Nodes: {}'.format(graph.CntOutDegNodes(0)))
16
17 # D:
18 print('#Cluster Coefficient: {:.5f}'.format(graph.GetClustCf()))
19
20 # E:
21 print('Diameter of Network: {}'.format(graph.GetBfsFullDiam(100)))
22
23 # F:
24 print('#Triads: {}'.format(graph.GetTriads()))
25
26 # G:
27 ComponentDist = graph.GetWccs()
28 print('#Weak Connected Components: {}'.format(len(ComponentDist)))
29 # for comp in ComponentDist:
30 #     print('Size: {}'.format(comp.Len()))
31
32 # H:
33 MxScc = graph.GetMxScc()
34 counter = 0
35 for node in MxScc.Nodes():
36     counter += 1
37 print('#Nodes in MxScc: {}'.format(counter))
38
39 # I:
40 if os.path.exists("InDegreeDistribution.plt"):
41     os.remove('InDegreeDistribution*')
42 if os.path.exists("OutDegreeDistribution.plt"):
43     os.remove('OutDegreeDistribution*')
44 graph.PlotInDegDistr('InDegreeDistribution', 'Undirected graph - in-degree Distribution')
45 graph.PlotOutDegDistr('OutDegreeDistribution', 'Undirected graph - out-degree Distribution')
46
```

کدهای مربوطه به سوال چهارم در یک فایل به نام q4.py

در خط ۲۴ حل قسمت F را مشاهده می‌کنیم که از ما تعداد مثلث‌ها را خواسته بود که با تابع ()GetTriads قابل مشاهده است.

در خطوط ۲۷ الی ۳۰ نیز حل قسمت G را مشاهده می‌کنیم که از ما تعداد مؤلفه‌های متصل ضعیف خواسته شده بود که با تابع ()GetWccs. ابتدا کل مؤلفه‌های متصل ضعیف به دست می‌اید و سپس با گرفتن طول خروجی، تعداد مؤلفه‌های متصل ضعیف بدست می‌آید.

در خطوط ۳۳ الی ۳۷ حل قسمت H را مشاهده می‌کنیم که از ما تعداد گره‌ها در بزرگترین مؤلفه‌ی همبندی قوی خواسته شده بود که ابتدا با تابع ()GetMxScc. بزرگترین مؤلفه‌ی همبندی قوی را میگیریم و سپس تعداد گره‌های آن را می‌شماریم و پرینت می‌کنیم.

در خطوط ۴۰ الی ۴۵ نیز حل قسمت I را مشاهده می‌کنیم که از ما توزیع درجات ورودی و خروجی خواسته شده بود که نتایج این قسمت در شکل‌های ۴ و ۵ قابل مشاهده است. ابتدا برای اینکه هر بار اجرا کردن این قسمت از سوال فایل‌های متعددی تولید می‌کرد، فایل‌های گذشته را پاک می‌کنم و سپس با استفاده از توابع PlotInDegDistr(<Name of file>, <Description of graph>).

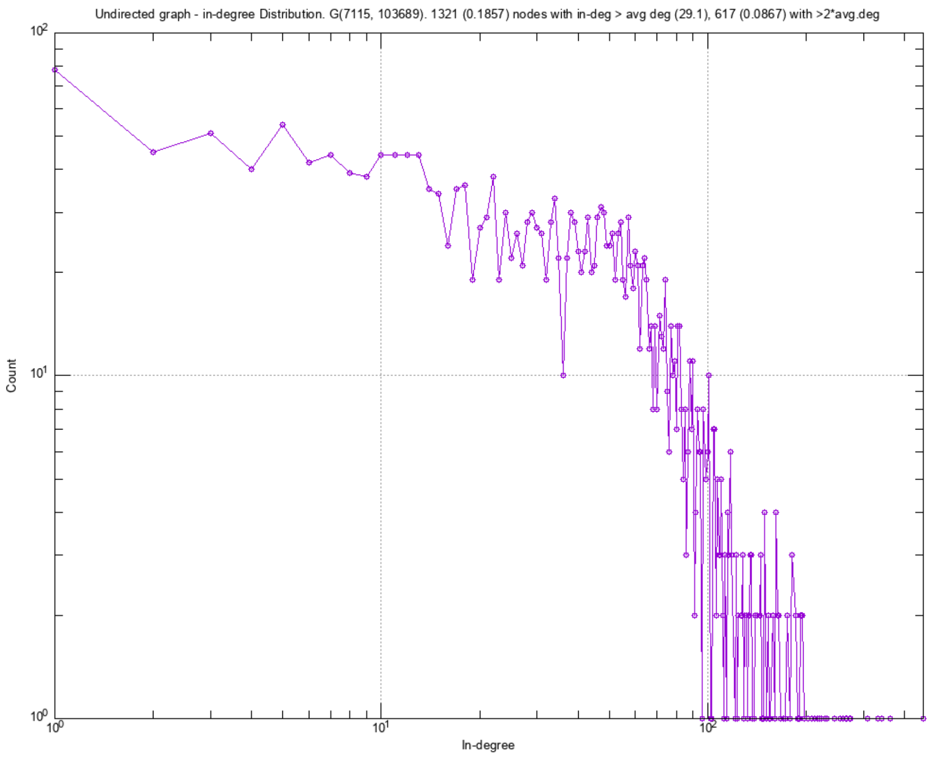
شکل ۳

```
meshkat●Semester 4/Complex Networks/HW3» python3 q4.py
#Nodes: 7115
#Edges: 103689
#Directed Edges: 103689
#UnDirected Edges: 100762
#In-Degree Nodes: 4734
#Out-Degree Nodes: 1005
#Cluster Coefficient: 0.14090
Diameter of Network: 7
#Triads: 608389
#Weak Connected Components: 24
#Nodes in MxScc: 1300
```

خروجی اجرای سوال چهارم با کد شکل ۲

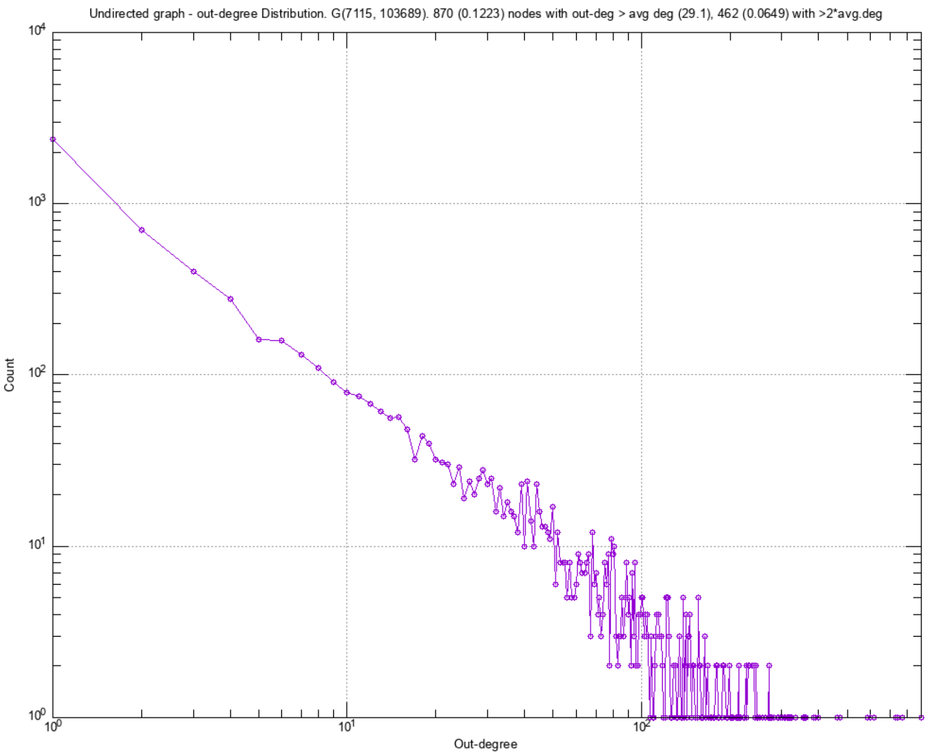
و PlotOutDegDistr(<name of file>, <Description of graph>). فایل های inDeg.InDegreeDistribution.png و outDeg.OutDegreeDistribution.png ساخته می‌شود. البته با فرمت‌های plt. و tab. نیز ساخته می‌شوند.

شکل ۴



خروجی توزیع درجات ورودی به فرم log-log

شکل ۵



خروجی توزیع درجات خروجی به فرم log-log