دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

نام و نام خانوادگی: پیمان هاشمی

شماره دانشجویی: 400131032

درس: تحلیل شبکه های پیچیده

تمرین: تمرین شماره 1

سوال 1)

الف)

برای حل این سوال، با استفاده از کتابخانه networkx یک گراف خالی ساخته و 7624 گره به آن اضافه می‌کنیم. با توجه به این که در مدل erdos-Renyi یال ها با احتمال p وجود دارند، با در نظر گرفتن p= 1/4 به تعداد 27806 نود پیداختیم.

نتایج حاصل از ساختن گراف به شرح زیر است:

'Graph with 7624 nodes and 27806 edges'

ب)

در مدل small world هر نود به همسایه های قبل و بعد از خودش متصل است. در ابتدا یک گراف خالی با 7624 نود می‌سازیم و با توجه به این که وصل کردن هر نود به همسایه هایش باعث بالا رفتن ضریب خوشه بندی برای گراف می‌شود. برای بدست آوردن ویژگی میانگین طول مسیر، نیاز داریم تا یال های تصادفی ایجاد کنیم. برای همین تا زمانی که تعداد یال ها مقدار خواسته شد (27806) نرسیده است، با احتمال 1/4 به صورت تصادفی به نود ها یال اضافه می‌کنیم.

```python
smal_world = nx.Graph()
for i in range(1,7625):
    smal_world.add_node(i)

for i in range(1,7625):
    smal_world.add_edge(i, i+1)
    smal_world.add_edge(i, 7625-(i+1))

    if  smal_world.number_of_edges() < 27806:
       for i in range(1,7624):
          smal_world.add_edge(i, i+2)
          smal_world.add_edge(i, 7625-(i+2))
    if  smal_world.number_of_edges() >= 27807:
      break

while smal_world.number_of_edges() < 27806:
    for i in range(1,7625):
        for j in range(1,7625):
            if smal_world.number_of_edges() < 27806:
                choice = random.randint(1,4)
                if smal_world.add_edge(i,j) == -1 and choice == 2:
                    smal_world.add_edge(i,j)
            else:
               break
nx.info(smal_world)
```

نتایج به شرح زیر است:

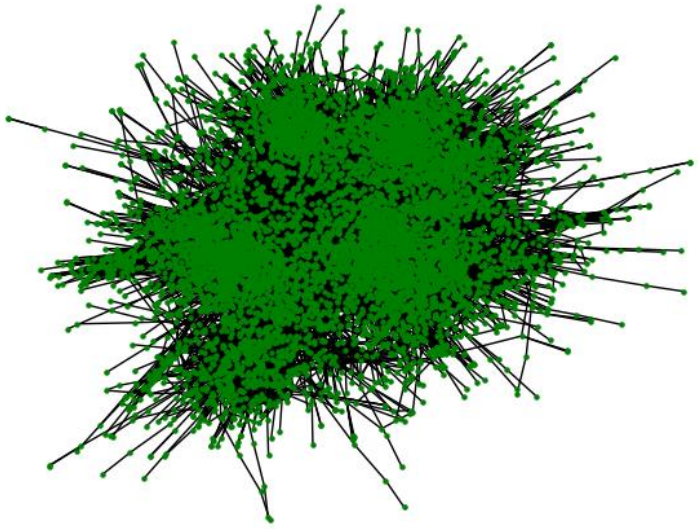
'Graph with 7626 nodes and 27806 edges'

ج)

دیتاست مورد نظر را خوانده و نمایش میدهیم.

نتایج به شرح زیر است:

	node_1	node_2
0	0	747
1	1	4257
2	1	2194
3	1	580
4	1	6478
...	...	...
27801	7488	7564
27802	7505	7579
27803	7533	7536
27804	7569	7587
27805	7580	7595

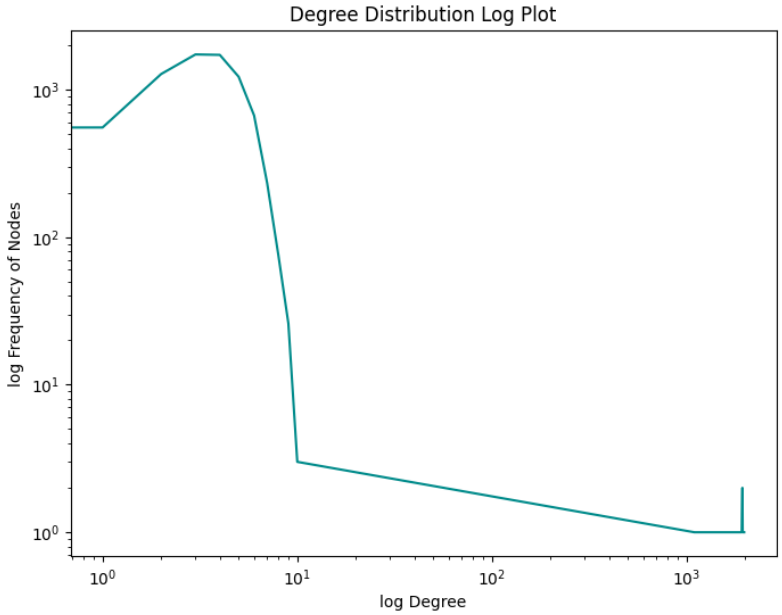


'Graph with 7626 nodes and 27806 edges'

د)

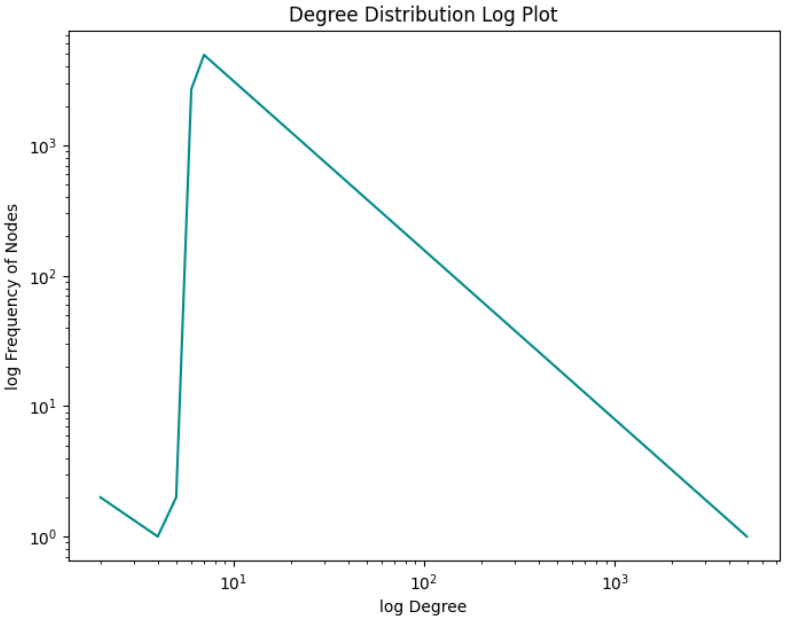
نمودار توزیع درجه اول (endros-renyi):

با توجه به این که گراف مورد بررسی یک گراف تصادفی بوده، دارای درجه نود های 0 تا بیشتر از 100 می‌باشد. نمومدار بدست آمده برابر زیر است:



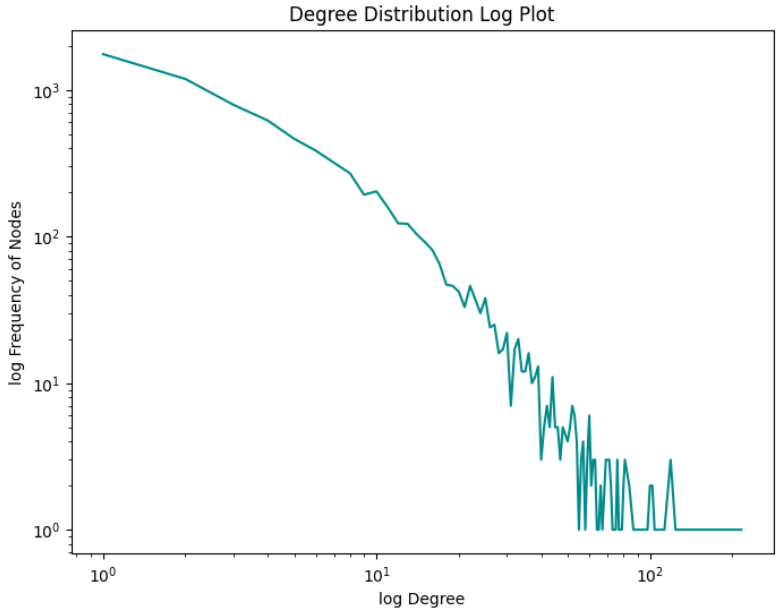
نمودار توزیع درجه اول (small_world):

گراف small_world ، گرافی منظم است (نود با درجه صفر وجود ندارد) اما چون در این قسمت یال ها با احتمال اضافه شده اند، نمودار توزیع درجه از توزیع نرمال پیروی می‌کند. همینطور با توجه به نمودار میتوان گفت نود های با درجه بالا بسیار کم هستند.



نمودار توزیع درجه اول (lastfm):

در این نمودار به دلیل اینکه نود های بالا و پایین وجود دارند، نمودار از توزیع نرمال پیروی نمی‌کند. نمودار، نموداری یک دم کلفت می‌باشد که به این معنی است که نود های با درجه بالا، بیشتر هستند.



ه)

در ابتدا با بدست آوردن نود های همسایه یک نود، آن را به لیست اضافه کرده و دوباره این کار را برای لیست اول تکرار می‌کنیم و نتیجه به لیست دوم اضافه می‌کنیم. سپس به اشتراک این دو لیست می‌پردازیم.

نتایج میانگین خوشه بندی برای هر شبکه به صورت زیر است :

result of endros_renyi Network : 0.12598837385480655

result of small world Network : 0.12648583170545874

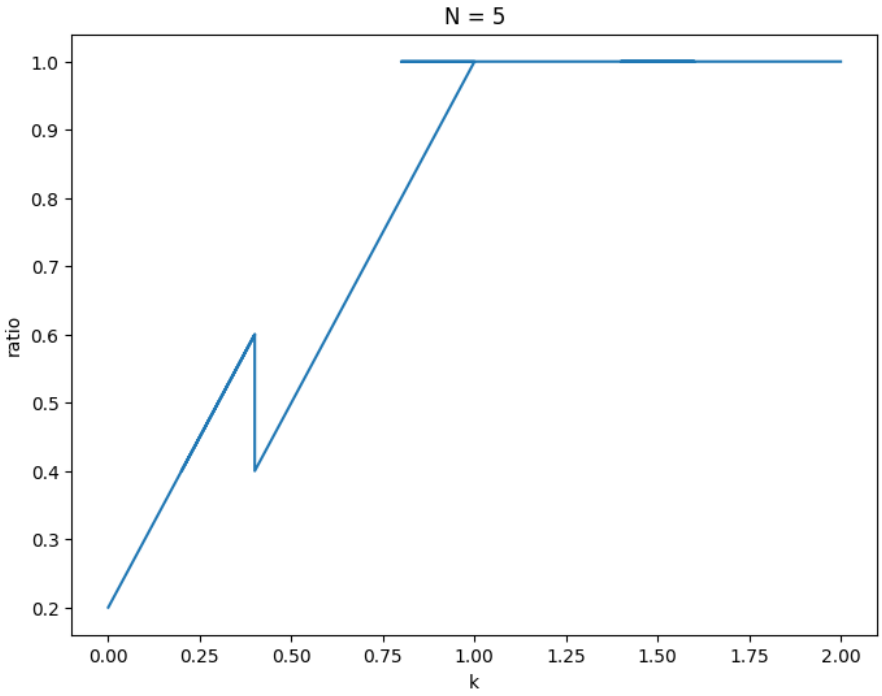
result of lastfm Network : 0.05364706035908317

small world > endros_renyi > lastfm

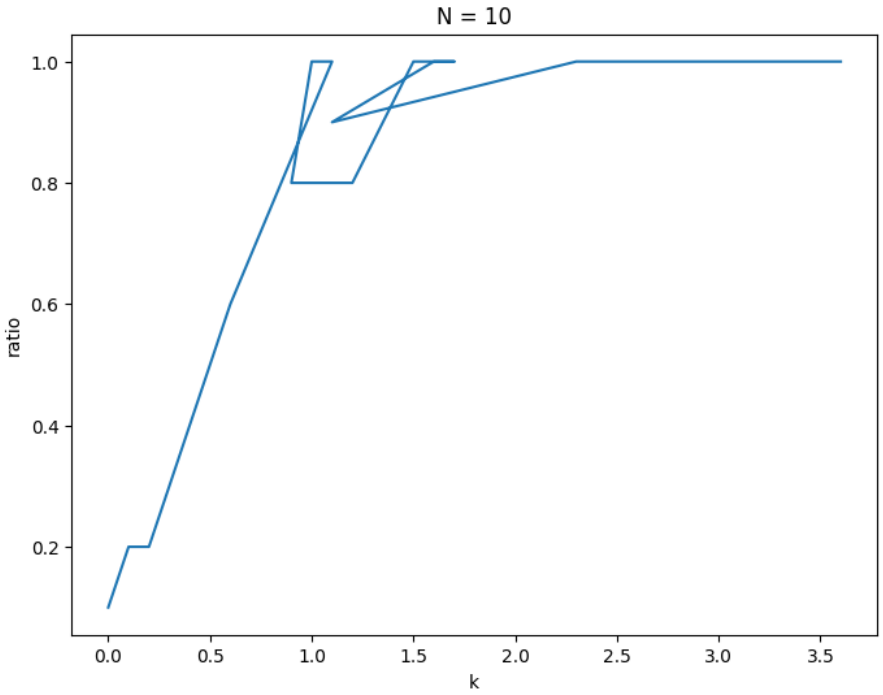
سوال 2)

با استفاده از رابطه p= k/nodes و مقادیر مختلف برای k، گراف های متفاوتی تولید کردیم. و (اندازه کل نود های گراف)/(بزرگترین قسمت متصل) برای هر k محسابه کردیم. همچنین تعداد نود های متفاوت نیز به کار گرفتیم.

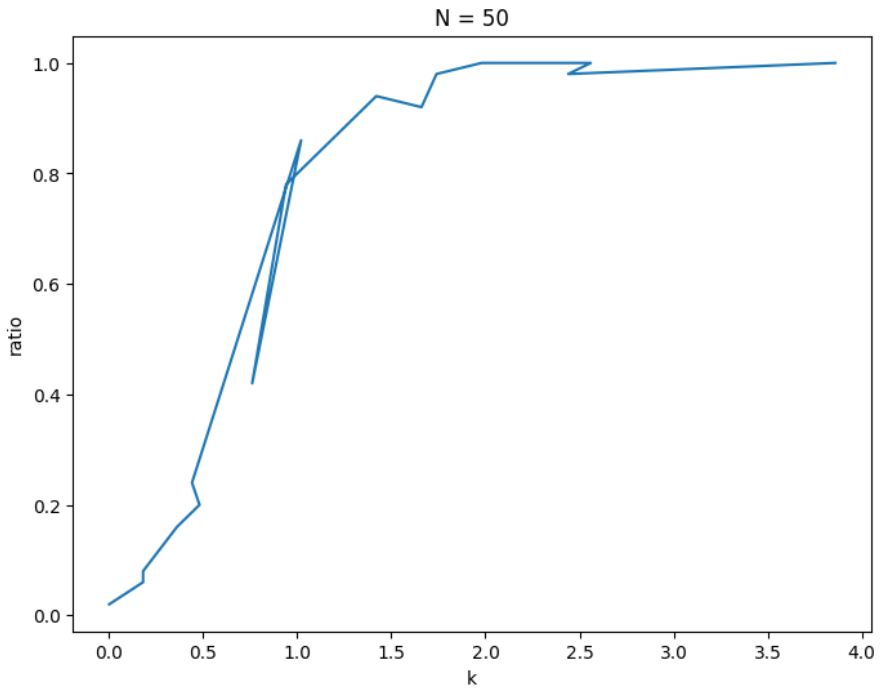
نتایج به شرح زیر است:



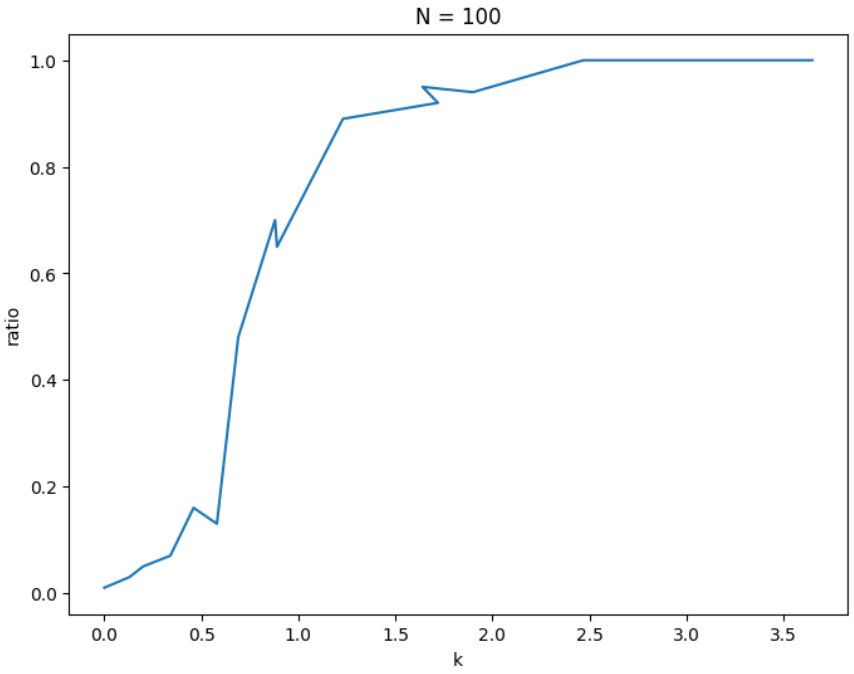
P = [0.0, 0.0625, 0.125, 0.1875, 0.25, 0.3125, 0.375, 0.4375, 0.5, 0.625, 0.75, 0.875, 1.0, 1.25, 1.5, 1.75]



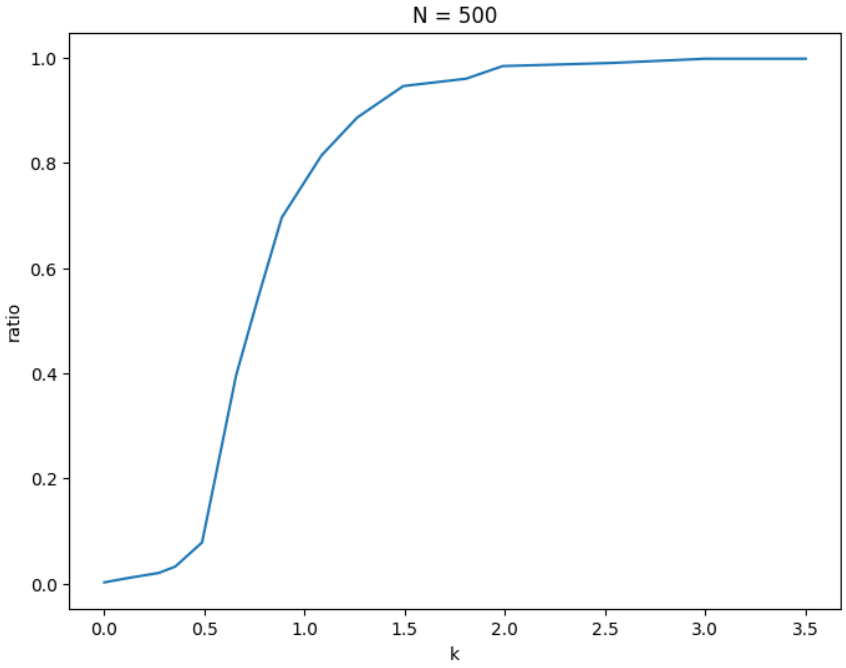
P = [0.0, 0.027777777777777776, 0.05555555555555555, 0.08333333333333333, 0.1111111111111111, 0.1388888888888889, 0.16666666666666666, 0.19444444444444445, 0.2222222222222222, 0.2777777777777778, 0.3333333333333333, 0.3888888888888889, 0.4444444444444444, 0.5555555555555556, 0.6666666666666666, 0.7777777777777778]



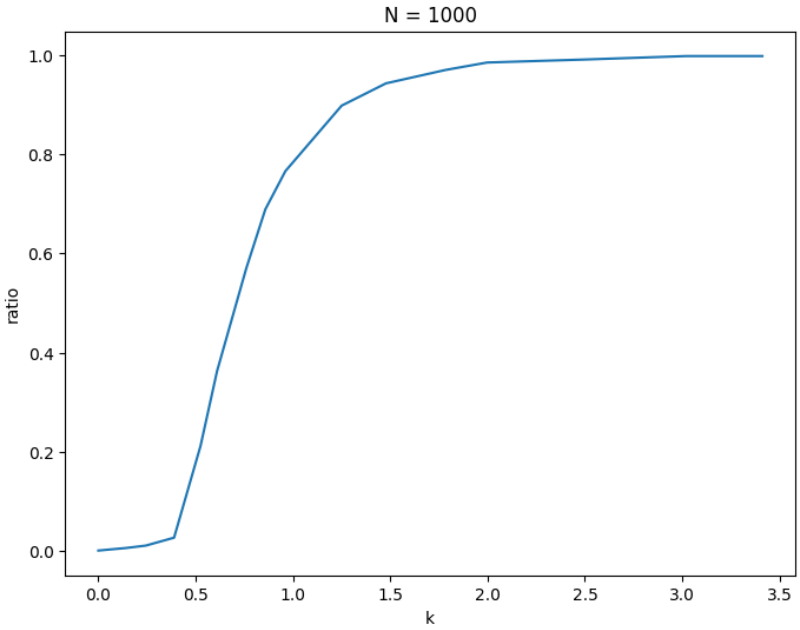
P = [0.0, 0.00510204081632653, 0.01020408163265306, 0.015306122448979591,
0.02040816326530612, 0.025510204081632654, 0.030612244897959183,
0.03571428571428571, 0.04081632653061224, 0.05102040816326531,
0.06122448979591836, 0.07142857142857142, 0.08163265306122448,
0.10204081632653061, 0.12244897959183673, 0.14285714285714285]



P = [0.0, 0.0025252525252525255, 0.005050505050505051, 0.007575757575757576,
0.010101010101010102, 0.012626262626262626, 0.015151515151515152,
0.017676767676767676, 0.020202020202020204, 0.025252525252525252,
0.030303030303030304, 0.03535353535353535, 0.04040404040404041,
0.050505050505050504, 0.06060606060606061, 0.0707070707070707]

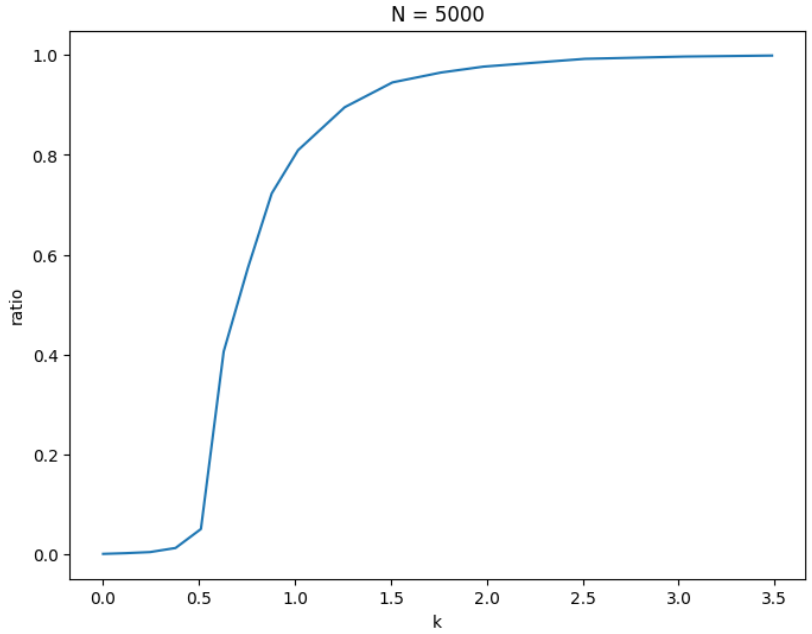


P = [0.0, 0.000501002004008016, 0.001002004008016032, 0.001503006012024048, 0.002004008016032064, 0.0025050100200400 8, 0.003006012024048096, 0.003507014028056112 3, 0.004008016032064128, 0.00501002004008016, 0.006012024048096192, 0.007014028056112224 5, 0.008016032064128256, 0.01002004008016032, 0.012024048096192385, 0.014028056112224449]



P = [0.0, 0.0002502502502502502 5, 0.000500500500500500 5, 0.000750750750750750 7, 0.001001001001001001, 0.001251251251251251 2, 0.001501501501501501 5, 0.001751751751751751 7, 0.002002002002002002, 0.002502502502502502 5,

0.003003003003003003, 0.0035035035035035035, 0.004004004004004004,
0.00500500500500500 5, 0.006006006006006006, 0.007007007007007007]



P = [0.0, 5.001000200040008e-05, 0.00010002000400080016, 0.00015003000600120024,
0.00020004000800160032, 0.00025005001000200 04, 0.00030006001200240 05,
0.00035007001400280056, 0.00040008001600320064, 0.00045001000200040008,
0.00060012002400480 01, 0.00070014002800560 11, 0.00080016003200640 13,
0.00100020004000800 16, 0.00120024004800960 2, 0.00140028005601120 22]

بر اسا نتایج بدست آمده دمیتوان نتیجه گرفت با افزایش تعداد گره ها، نتیجه به نمودار خواسته شده در صورت سوال نزدیک می‌شود و همگرا شدن تعداد گره ها به سمت بینهایت، خطا نیز به سمت صفر همگرا می‌شود.

همچنین نتایج بدست آمده میتوان نتیجه گرفت که در p=1/500 انتقال فاز صورت می‌گیرد چرا که اولین giant component در این احتمال شکل می‌گیرد. قبل از این احتمال نود های تک بسیار زیاد هستند و بعد از این احتمال نیز giant component شکل می‌گیرد.

مقدار k برابر زیر است:

k = [0,0.25,0.5,0.75,1,1.25,1.5,1.75,2,2.5,3,3.5,4,5,6,7]

سوال 3)

الف)

در این سوال، گراف به نوعی است که از دو قسمت تشکیل شده که نود هر قسمت می‌تواند به نودی از قسمت دیگر متصپصل باشد و هیچ یالی بین نود های یک قسمت وجود ندارد. در این حالت، تعداد نود های هر بخش برابر N1 و N2 است. حداکثر تعداد نود های متصل میتواند در حالتی رخ دهد که یک نود از N1 یا N2 به تمام نود های بخش مقابل وصل شود. پس برای همه نود ها اگر این حالت را در نظر بگیریم، بیشترین تعداد لینک در شبکه برابر N2 * N1 می‌باشد

ب)

اگر N1 و N2 دو بخشی نباشند، تعداد یال ها برابر جمع این دو قسمت می‌باشد

حداکثر تعداد لینک های که میتواند اتفاق بیافتد برابر زیر است:

$$\frac{N1^2+N2^2+2N1N2-N1-N2}{2} = \frac{(N1+N2)*(N1+N2-1)}{2}$$

تعداد لینکی که نمیتواند اتفاق بیافتد برابر زیر است:

$$\frac{N1^2+N2^2-N1-N2}{2}$$

ج)

با توجه به اینکه چگالی شبکه برابر (بیشترین تعداد لینک) / (تعداد لینک ها) است و N1<<N2 است یعنی N1 در مقابل N2 بسیار کوچک است. پس با ساده سازی داریم (N1*N2) / N2. چرا که تعداد لینک های N1 بسیار کمتر از N2 است پس لینک با N2 برابر است و با ساده سازی رابطه بالا برابر N1 / 1 می‌شود. چون مقدار N1 بسیار کم است پس چگالی به 1 تقریب زده می‌شود و چگالی شبکه تقریبا برابر 1 می‌شود.

د)

با توجه به اینکه میانگین درجه برابر (تعداد نود ها) / (تعداد لینک ها) می‌باشد، برای بخش اول میانگین درجه برابر K1 = m1/N1 و برای بخش دوم برابر K2 = m2/N2 می‌باشد. با توجه به اینکه در گراف دو بخشی است و داخل هر بخش لینکی وجود ندارد پس m1 = m2 است و داریم K1 * N1 = K2 * N2.

سوال 4)

توضیحات تئوری:

الگوریتم CELF

از این الگوریتم به منظور افزایش سرعت بهینه سازی که قابل اجرا به هر مسئله مشابهی که تابع هدف submodular است، استفاده می‌کنیم. Submodular به این صورت است که marginal gain نود با تکرار کاهش پیدا می‌کند. برای مشخص کردن marginal gain در تکرار i ام فرض میشود marginal gain های قبلی را در یک لیست مرتب شده و نزولی داریم. در این روش برای بدست آوردن i امین marginal gain، marginal gain نودی که در بالای لیست قرار دارد را محاسبه کرده در صورتی که از بقیه نود ها بزرگتر باشد، لازم به محاسبه بقیه نود ها نیست ولی در صورتی که بزرگتر از بقیه ها نباشد، باید marginal gain آن ها حساب کرد و در لیست مرتب شده در جای مناسب گذاشت. و این روند را تکرار کرده تا به بیشترین marginal gain برسیم. در صورتی که مقدار مناسب در انتخا لیست باشد و همه مقادیر را حساب کنیم، مثل روش حریصانه می‌شود که معمولا در دنیای واقعی به این مورد نمیرسیم.

الگوریتم حریصانه

در این الگوریتم یک مجموعه خالی در نظر میگیریم و سپس در دور اول نودی که بیشترین اندازه out را دارد انتخاب می‌کنیم و آن را به مجموعه S اضافه می‌کنیم. حال نود 1 را بدست اوردیم. برای انتخاب نود i-1، در بین نود های انتخاب نشده، نودی انتخاب می‌شود که marginal gain بالاتری را نسبت به نود های دیگر داشته باشد. اگر آن را u بنامیم داریم:

$$\max_{u} f(S_{i-1} \cup \{u\})$$

توضیحات برنامه نویسی:

دیتا ست را لود کرده و گراف را توسط آن تشکیل میدهیم.

DiGraph with 18772 nodes and 396160 edges

با توجه به صورت سوال 10 گره را توسط آن ها گره ها میزان تاثیر بیشینه خواهد داشت را پیاده سازی کردیم که نتایج آن به شرح زیر است:

DiGraph with 18772 nodes and 3851 edges
DiGraph with 18772 nodes and 3997 edges
DiGraph with 18772 nodes and 3952 edges
DiGraph with 18772 nodes and 3908 edges
DiGraph with 18772 nodes and 4022 edges
DiGraph with 18772 nodes and 4032 edges
DiGraph with 18772 nodes and 4090 edges

DiGraph with 18772 nodes and 4066 edges
DiGraph with 18772 nodes and 3890 edges
DiGraph with 18772 nodes and 4087 edges

با توجه به توضیحات داده شده در قسمت الگوریتم حریصانه، آن را پیاده سازی کرده ونتایج آن به شرح زیر است:

number 1 : 93504 spread = 64.0

number 2 : 35290 spread = 105.9

number 3 : 54681 spread = 138.3

number 4 : 46847 spread = 159.1

number 5 : 85740 spread = 173.5

number 6 : 8273 spread = 187.7

number 7 : 38109 spread = 201.1

number 8 : 44785 spread = 213.0

number 9 : 46572 spread = 223.7

number 10 : 77374 spread = 234.4

greedy output: [93504, 35290, 54681, 46847, 85740, 8273, 38109, 44785, 46572, 77374]

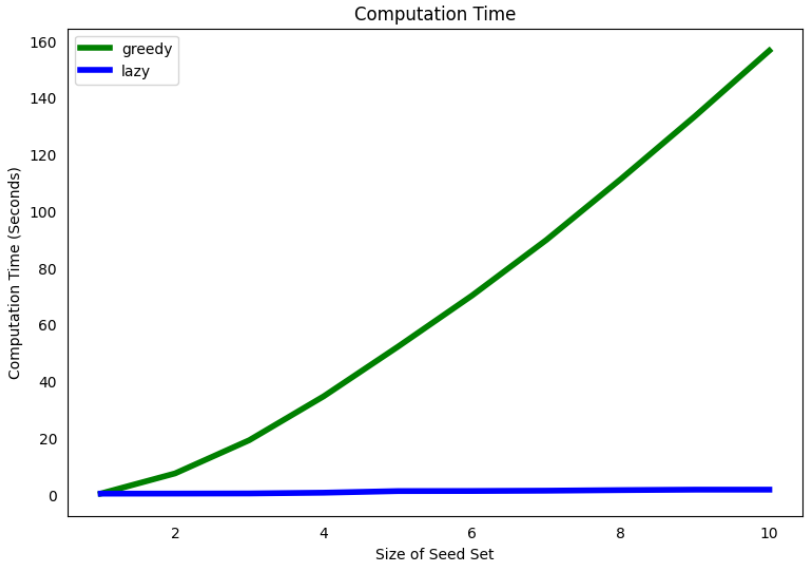
برای پیاده سازی قسمت CELF نیز در ابتدا لیست مرتب تکرار اول را محاسبه کرده و سپس لیستی از نود ها marginal gain آن ها میسازیم. سپس k-1 نود بعدی با استفاده از list sorting محاسبه می‌کنیم. در این روند spread مربوط به بالاترین نود در لیست را محاسبه کرده و نتیجه را با marginal gain ذخیره شده مقایسه می‌کنیم. در نهایت دوباره لیست را sort کرده و چک می‌کنیم که ایا نود قبلی در بالاترین جایگاه مانده است یا خیر. این روند را تکرار میکنیم تا نود مربوطه یافت شود.

نتیاج حاصل از این الگوریتم به شرح زیر است:

lazy_hill_climbing output: [93504, 35290, 54681, 46847, 85740, 8273, 38109, 44785, 67890, 77374]

خروجی برای هر دو الگوریتم یکسان است.

برای بررسی زمان نیز، زمان هر الگوریتم را محاسبه کرده و بر طبق نتایج بدست آمده داریم:



مشاهده می‌شود که پیاده سازی مربوط به الگوریتم حریصانه با سایز 10 نسبت به الگوریتم CELF زمان بیشتری را میگیرد. با توجه به جدول زمان مورد نیاز برای بدست آوردن نتایج لازم برایالگوریتم حریصانه برابر 350 ثانیه است در حالی که الگوریتم CELF کمتز از 5 ثانیه طول می‌کشد.

سوال 5)

توضیحات تئوری:

الگوریتم CELF

در مرحله اول این سوال، از الگوریتمی استفاده می‌شود که به جای در نظر گرفتن هزینه نود از marginal gain آن ها به عنوان معیار انتخاب استفاده می‌کند. سپس برای بدست آوردن نود های اولیه، در انتهای هر تکرار هزینه مصرفی (cost) را از کل کم می‌کنیم که در این سوال 10 در نظر گرفته شده است. سپس با استفاده از نسبت marginal gain و هزینه، نود را انتخاب کرده و هزینه آن را از بودجه کم می‌کنیم و آن ها رد مجموعه دیگر قرار میدهیم. حال از بین مجموعه اول و دوم، تابع ماکسیسمم گیری را اجرا می‌کنیم و داریم

$$S = \text{armax}(f(s'), f(s''))$$

همچنین در این سوال نیز می‌توان از خاصیت Submodular نیز استفاده کرد که در سوال قبل به طور کامل توضیح داده شد.

الگوریتم حریصانه

این سوال نیز مانند سوال قبل است و کشف شیوع مانند بیشینه سازی تاثیر است. در کشف شیوع تفاوتی که وجود دارد این است که به جای در نظر گرفتن هزینه نود با یک، هزینه هر نود برابر با 0.8 که در صورت سوال آورده شده است در نظر گرفته می‌شود. سپس مانند سوال قبل marginal gain بدست آمده از هر نود را در یک لیست خالی قرار میدهیم و بالاترین marginal gain را محاسبه کرده و نود آن را به لیست می‌افزاییم. هزینه هر نود را نیز محاسبه کرده و بررسی میکنیم که از 10 بیشتر نشود.

توضیحات برنامه نویسی:

الگوریتم CELF

در این سوال نیز مانند سوال 4، در ابتدا 10 گره تاثیر گذار (realization) از گراف با احتمال p=0.01 میسازیم.

با توجه به توضیحات داده شده در قسمت تئوری به پیاده سازی آن پرداختیم و نتایج بدست آمده به شرح زیر است:

خروجی با استفاده از marginal gain:

```
unit cost :

62821:
Spread = 10.325
cost = 6.960000000000001

not add 53213 :
spread = 18.775
cost = 14.559999999999999
time unit cost =  1.029052972793579
result = 62821
```

خروجی با استفاده از نسبت marginal gain به هزینه:

```
benefit Ratio :
118174:
Spread = 7.425
cost = 0.9600000000000002

number 2 : 76391 :
spread = 12.55 ,
cost = 1.7600000000000002

number 3 : 101558 :
spread = 18.6 ,
cost = 2.72

number 4 : 133109 :
spread = 23.05 ,
```

cost = 3.5200000000000005

number 5 : 85755 :
spread = 25.7 ,
cost = 4.48

number 6 : 59103 :
spread = 26.1 ,
cost = 5.28

number 7 : 46963 :
spread = 27.449999999999996 ,
cost = 6.08

number 8 : 95952 :
spread = 28.075 ,
cost = 7.279999999999999

number 9 : 78060 :
spread = 29.025 ,
cost = 8.24

not add 84886 :
spread = 30.125
cost = 9.280000000000001
result of benefit Ratio = [118174, 76391, 101558, 133109, 85755, 59103, 46963, 95952, 78060]

خروجی الگوریتم CELF برابر زیر است:

result of celf = [118174, 76391, 101558, 133109, 85755, 59103, 46963, 95952, 78060]
mean spread value = 10.325

الگوریتم حریصانه

62821 , cost = 6.960000000000001
finished
result of greedy algorithm = 62821