



دانشگاه صنعتی امیر کبیر
(پلی تکنیک تهران)

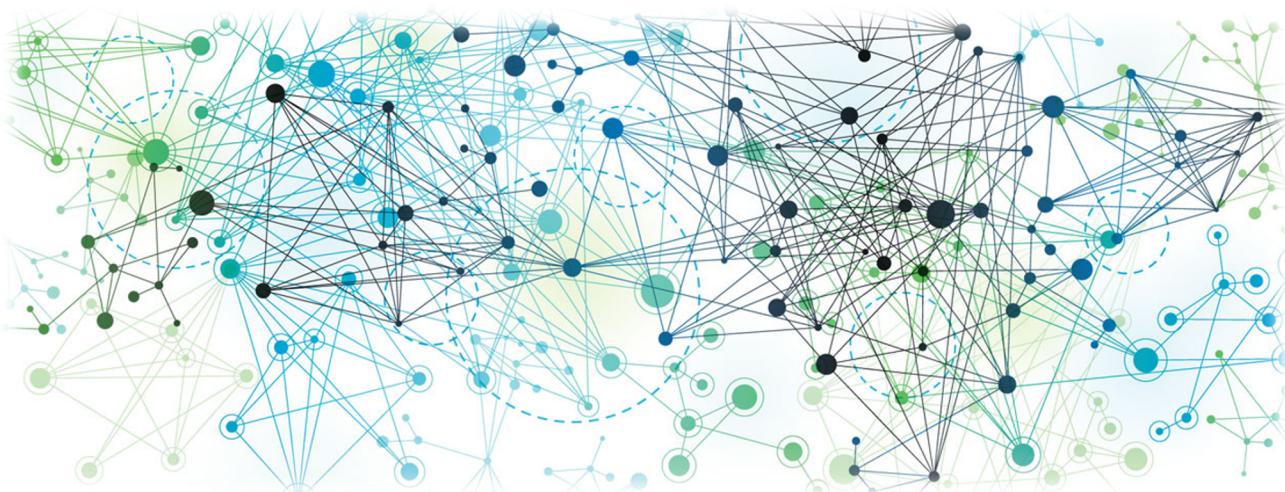
درس نحلپل شبکه های پیچیده

استاد درس جناب آقای دکتر چهرقانی

(تمرین سری اول)

محسن عبادپور | ۴۰۰۱۳۱۰۸۰ | m.ebadpour@aut.ac.ir

نیمسال اول سال تحصیلی ۱۴۰۱-۱۴۰۲



فهرست پاسخ‌ها

مسئله اول.....	3
مسئله اول: الف) تولید گراف Erdos-Renyi	3
مسئله اول: ب) تولید گراف Small-World	4
مسئله اول: ج) دانلود و لود کردن گراف LastFM	4
مسئله اول: د) نمودارهای توزیع درجه	5
مسئله اول: ه) ضریب خوشه‌بندی	7
مسئله اول: فعالیت بیشتر / تاثیر احتمال تعویض یال در ضریب خوشه‌بندی Small-World	8
مسئله دوم.....	9
مسئله دوم: فعالیت بیشتر / مصورسازی تشکیل بزرگترین مولفه به ازای تغییرات C	10
مسئله سوم: الف.....	12
مسئله سوم: ب.....	12
مسئله سوم: ج	12
مسئله سوم: د	13
مسئله چهارم.....	14
مسئله چهارم: حرصانه.....	14
مسئله چهارم: CELF	15
مسئله چهارم: Lazy Evaluation (سرعت بخشیدن به الگوریتم موجود)	15
مسئله چهارم: نتایج	16
مسئله چهارم: فعالیت بیشتر / بررسی تاثیر تعداد realization‌ها در زمان اجرا.....	18
مسئله پنجم.....	19

مسئله اول

مسئله اول

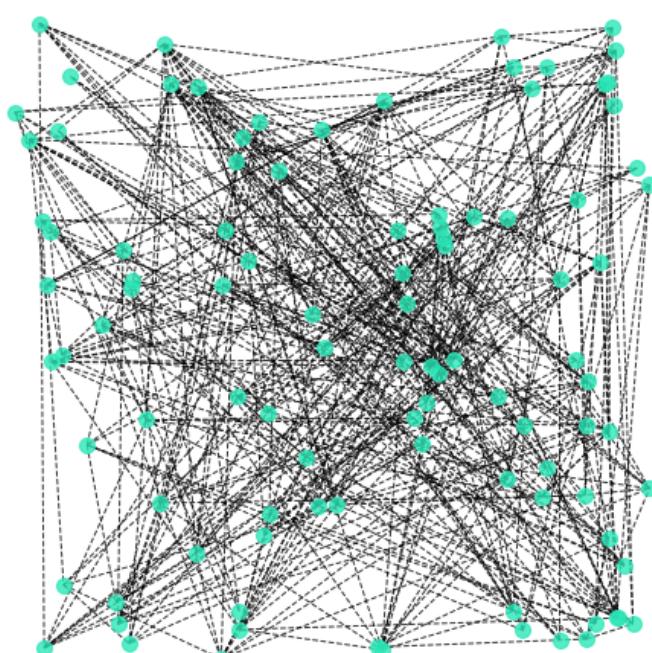
برای پاسخ گویی به این سوال و زیر بخش های آن از هیچ کتابخانه یا تابع آماده‌ای استفاده نشده و تماماً پیاده سازی شده است.

برای این منظور کلاسی با امضای Graph تعریف شده است که در آن همه فرآیند های افزودن و حذف یال، محاسبه درجه و ضربی خوشبندی پیاده سازی شده است و برای ساخت گراف و پردازش های مورد نظر در این سوال از این کلاس گراف استفاده شده است. در این کلاس شماره گذاری نود ها از 0 در نظر گرفته شده است.

مسئله اول: الف) تولید گراف Erdos-Renyi

برای تولید گراف تصادفی Erdos-Renyi یک شی از کلاس Graph با تعداد نود 7624 ایجاد می‌کنیم. سپس توسط تابع combinations از کتابخانه itertools که بصورت پیشفرض در زبان پایتون قرار دارد استفاده کرده و تمام جفت یال های ممکن که گراف میتواند داشته باشد را تولید می‌کنیم. سپس توسط تابع sample از کتابخانه random به تعداد 27806 یال از یال های ممکن را بصورت تصادفی انتخاب و سپس آن ها را به گراف اضافه می‌کنیم تا گراف تصادفی مورد نظر حاصل شود. به عنوان یک فعالیت اضافی برای این سوال و برای نشان دادن صحت پیاده سازی یک گراف 100 نودی با 500 یال ایجاد و آن را مصور می‌کنیم که می‌بینیم کاملاً تصادفی بوده و شرایط گراف Erdos-Renyi را دارد است. (صرفاً برای مصور سازی از networkx استفاده شده است)

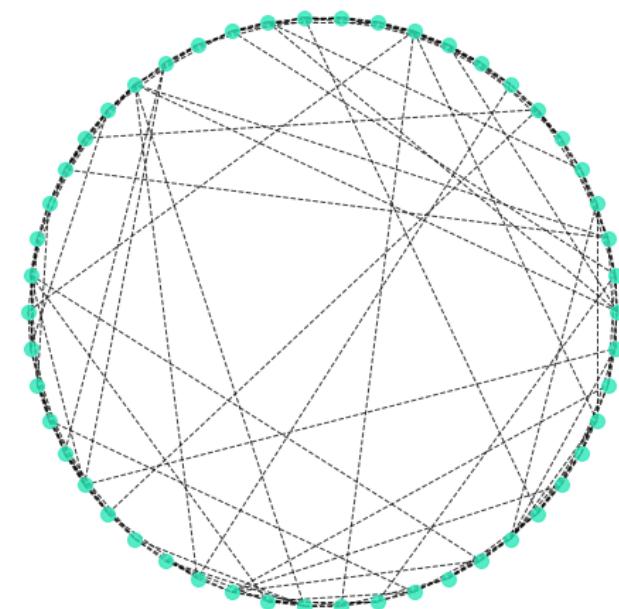
Sample Erdos-Renyi graph generated by 100 nodes and 400 edges



مسئله اول: ب) تولید گراف Small-World

برای تولید گراف دنیای کوچک (small-world) نیز از کلاس تعریفی خود با نام Graph استفاده می‌کنیم که کاملاً از پایه پیاده‌سازی شده است. ابتدا یک گراف منظم (regular) دایره‌ای شکل می‌سازیم که در آن نود‌ها روی محیط دایره قرار گرفته‌اند و هر نود با ۳ همسایه قبل و ۳ همسایه بعد یال دارد. در اینصورت تعداد یال‌های ایجاد شده کمتر از تعداد یال‌های گفته شده می‌باشد که برای حل این موضوع، بصورت منظم یال‌های باقی مانده را به گراف اضافه می‌کنیم تا نظم بهم نخورد. حال که گراف منظم ما ایجاد شد، بایستی با در نظر گرفتن یک احتمال (۰.۴) تصمیم بگیریم که آیا مقصد یال‌ها ثابت باقی بمانند یا به یک نود مقصد تصادفی دیگر متصل شوند. به عنوان یک فعالیت اضافی برای این سوال و برای نشان دادن صحت پیاده‌سازی یک گراف ۵۰ نودی با ۱۵۰ یال ایجاد و آن را مصور می‌کنیم که می‌بینیم شرایط گراف دنیای کوچک را داراست. (صرفاً برای مصور سازی از networkx استفاده شده است)

Sample Small-World graph generated by 50 nodes and 150 edges



مسئله اول: ج) دانلود و نود کردن گراف LastFM

از لینک درج شده در صورت سوال مجموعه داده‌ی مورد نظر دانلود کرده و سپس فایل گراف را خوانده و در کلاس تعریفی خود با نام Graph آن را وارد می‌کنیم.

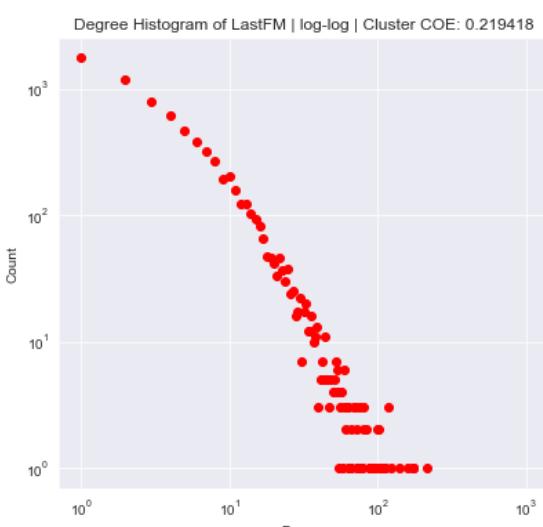
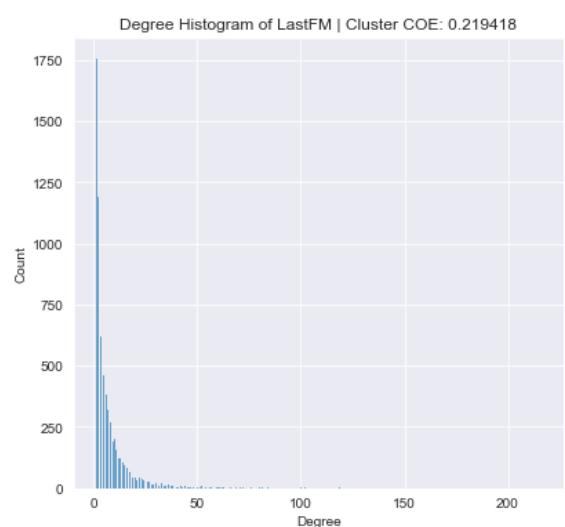
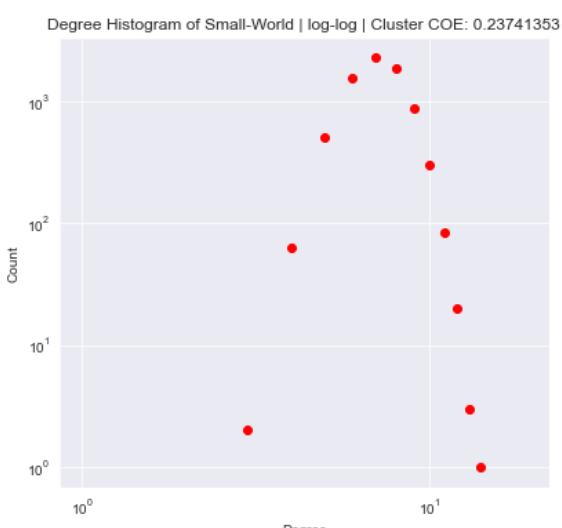
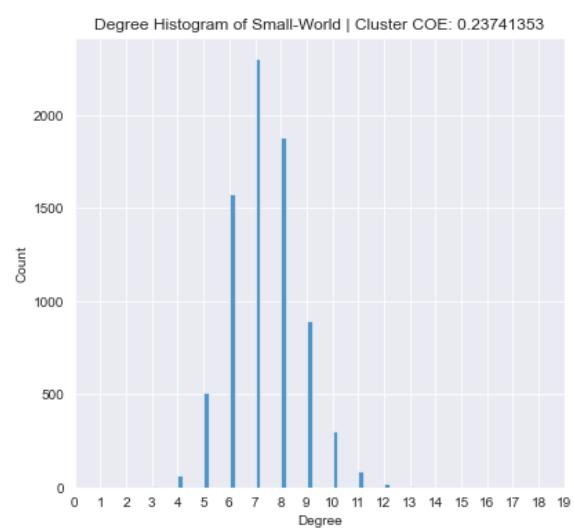
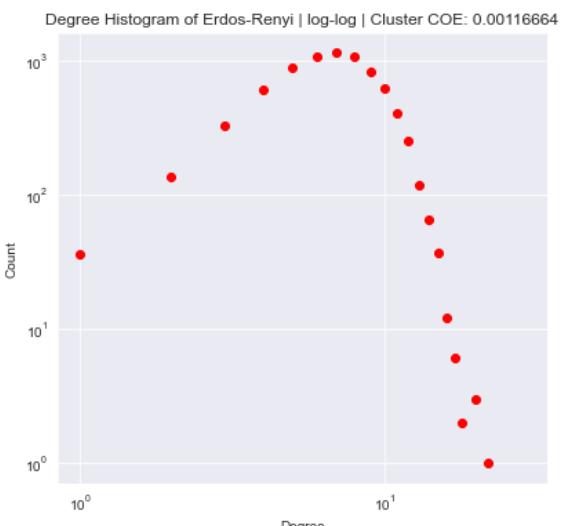
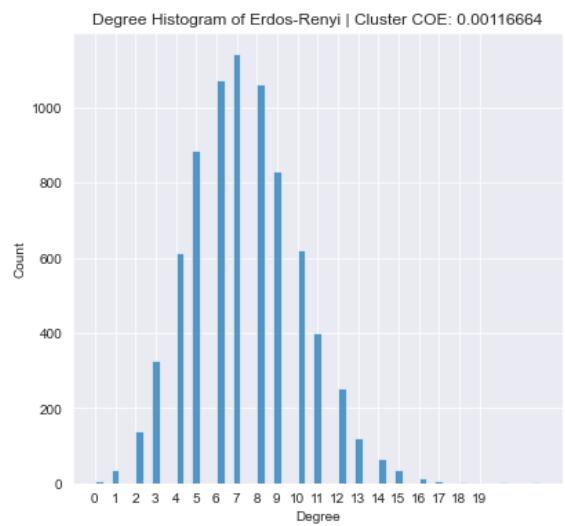
مسئله اول: د) نمودار های توزیع درجه

برای گراف های سه بخش قبل نمودار توزیع درجهها را رسم می کنیم؛ برای این منظور و برای هر کدام درجهی تمام نود ها را حساب کرده و سپس توسط تابع histplot از کتابخانه مصورسازی seaborn نمودار توزیع درجهها را رسم می کنیم. برای رسم در مقیاس log-log نیز از تابع xscale و yscale استفاده کرده و پارامتر "log" را به آنها ارسال می کنیم. نمودار های زیر حاصل می شود.

طبق نمودار های بدست آمده و انتظار قبلی ملاحظه می کنیم که توزیع درجهی گراف تصادفی Erdos-Renyi منطبق با یک شکل توزیع دو جمله‌ای می باشد که در آن احتمال وقوع نود هایی با درجهی بالا وجود نداشته و برابر با صفر است. مشابه با همین نیز Small-world واقع شده است که احتمال وقوع نود هایی با درجه بالا (و حتی بسیار پایین) صفر است؛ در گراف دنیای کوچک می بینیم که واریانس تغییرات درجه نود ها کمتر از واریانس تغییرات درجه نود ها در گراف Erdos-Renyi است (فاصله از میانگین کم است) چرا که از یک گراف منظم ساخته شده است و این منظم بودن باعث شده است که تغییرات بسیار زیادی در درجه نود ها اتفاق نیافتد.

در مقایسه نمودار توزیع درجه گراف های تصادفی با گراف دنیای واقعی (LastFM) در مقیاس log-log می بینیم که یک نمودار گراف دنیای واقعی یک خط نزولی بوده و احتمال وجود و رخداد نود هایی با درجهی بسیار بالاتر نیز وجود دارد در صورتی که در گراف های تصادفی این مورد وجود ندارد. مثلا احتمال رخداد نودی با درجهی ۵۰ در گراف تصادفی کاملاً صفر بوده و اصلاً رخداده است در صورتی که میدانیم در گراف های دنیای واقعی وجود چنین نود هایی محتمل است که این مورد در نمودار توزیع درجه گراف دنیای واقعی نیز مشهود بوده و می بینیم احتمال اینکه نود هایی تا درجه های ۳۰۰-۲۰۰ نیز به وجود آید قابل رویت است. (هر چند این احتمال بسیار پایین باشد).

به عبارتی دیگر تفاوت بین گراف های تصادفی و دنیای واقعی را میتوان این گونه گفت که احتمال و رخداد نود هایی با انواع درجه های پایین و بالا در نمودار گراف دنیای واقعی قابل رویت بوده و مشهود است چرا که در حقیقت نیز این مورد اتفاق می افتد اما این در صورتی است که در گراف های تصادفی این انعطاف پذیری در تخصیص درجه های بسیار متفاوت به نود ها وجود ندارد.



9

مسئله اول: ۵) ضریب خوشبندی

برای محاسبه ضریب خوشبندی، طبق فرمول مذکور در متن سوال اقدام شده است که برای این منظور یک تابع با عنوان Cluster_coe پیاده سازی شده است که به ازای یک نود خاص، ضریب خوشبندی آن را محاسبه و برمیگرداند که برای هر کدام از سه گراف خواسته شده و به ازای هر کدام از نود هایشان این تابع فراخوانی شده و از مقادیر حاصل برای نود های هر گراف میانگین گرفته می شود. برای هر کدام از سه گراف مفروض، در قسمت title نمودار توزیع درجه ها در بخش قبل میانگین ضریب خوشبندی درج شده است که مجددا در این قسمت آورده می شود:

ضریب خوشبندی گراف Erdos-Renyi : ۰.۰۰۱۱

ضریب خوشبندی گراف Small-World : ۰.۲۳۷۴

ضریب خوشبندی گراف LastFM : ۰.۲۱۹۴

همانطور که ملاحظه می کنیم طبق انتظار ضریب خوشبندی گراف تصادفی Erdos-Renyi خیلی کوچکتر از گراف دنیای واقعی می باشد چرا که تمامی یال ها گراف تصادفی ایجاد شده و هیچ معیاری خوشبندی یا شباهت بین آن ها در نظر نگرفته شده است. معیار خوشبندی گراف Small-World نیز بالا بوده و نزدیک به دنیای واقعی است چرا که گراف مذکور از یک گراف منظم ساخته شده و نود های همسایه‌ی هر نود بروی دایره‌ی فرضی با یکدیگر اتصال همسایگی دارند و این باعث می شود که ضریب خوشبندی بالاتر از گراف های کاملاً تصادفی برود.

مسئله اول: فعالیت بیشتر | تأثیر احتمال تمویض یال در ضریب خوشبندی Small-World

در مدل Small-World دیدیم که با یک احتمال (۰.۴) یال های گراف منظم را تغییر میدهیم تا گراف مدنظر حاصل شود. حال در این بخش به عنوان فعالیت بیشتر میخواهیم بررسی کنیم که تغییرات این احتمال چه تأثیری در ضریب خوشبندی گراف دارد. برای این منظور مدل Small-World را به ازای احتمال های بازه‌ی ۰.۹-۰.۱ ایجاد و ضریب خوشبندی آن را محاسبه می‌کنیم که نتایج بصورت زیر حاصل می‌شود.

ضریب خوشبندی	احتمال تغییر یال در گراف Small-World
۰.۴۹۱۱	۰.۱
۰.۳۷۸۱	۰.۲
۰.۲۹۷۷	۰.۳
۰.۲۳۸۹	۰.۴
۰.۱۹۳۰	۰.۵
۰.۱۵۸۴	۰.۶
۰.۱۳۵۰	۰.۷
۰.۱۱۳۵	۰.۸
۰.۰۹۵۳	۰.۹

از جدول فوق میتوان دریافت که در زمان ساخت گراف Small-World هرچقدر با احتمال بالاتری یال های گراف منظم مدنظر را جایه‌جا کنیم، گراف حاصل دارای ضریب خوشبندی کمتری خواهد بود.

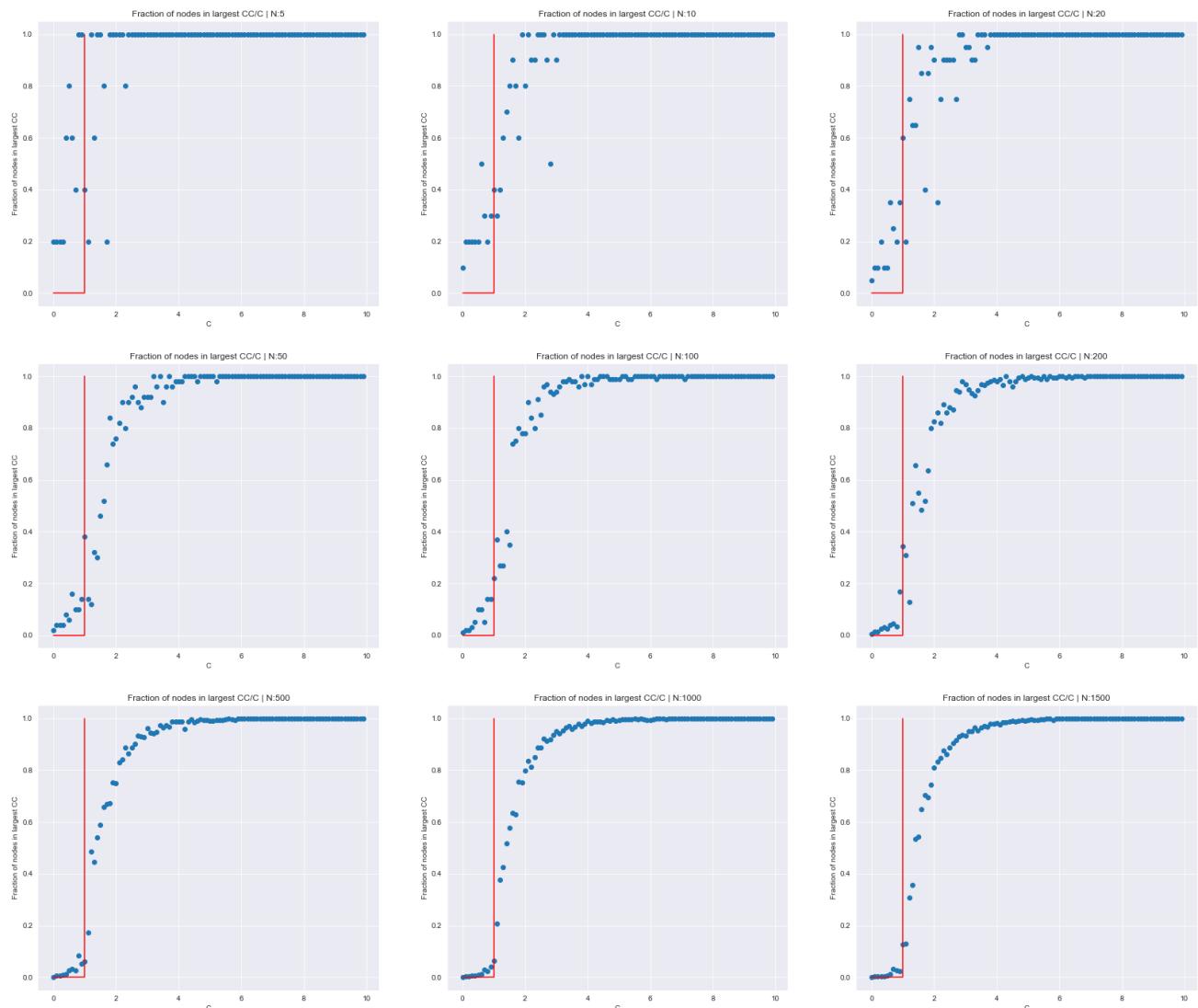
مسئله دوم

مسئله دوم

طبق پاورقی موجود در متن سوال برای پاسخ گویی به این مسئله از کتابخانه networkx استفاده شده است. برای مصورسازی ادعای خواسته شده در متن سوال، به ازای تعداد نود های $N=1500-1000-500-200-100-50-10$ گراف تصادفی Erdos-Renyi را مد نظر گرفته و هر یک را به ازای احتمال های $p=C/(N-1)$ ایجاد می کنیم که C در بازه‌ی 0.1 الی 10 تغییر پیدا میکند. در نهایت به ازای هر یک از N ها، نمودار تغییرات C به ازای اندازه نسبی بزرگترین مولفه (Largest Components) را رسم میکنیم تا نتایج بصورت صفحه بعد حاصل شود. (notation مربوط به پارامتر C از صفحه ۸۲ اسلاید درسی انتخاب شده است)

در گام اول از قالب کلی هر یک از نمودارها میتوان ادعای مطرح شده مبنی بر شکل تغییرات اندازه نسبی بزرگترین مولفه (LCC) را اثبات نمود که به ازای C از صفر تا یک LCC تشکیل نشده و با گذر از یک LCC شروع به تشکیل شده و اندازه آن بصورت نمایی بالا می‌رود. در گام بعد میتوان ملاحظه نمود که هر چقدر N افزایش پیدا می‌کند، شکل تغییرات همگرایتر شده و دقت گزاره های نمودار بیشتر و به هم نزدیکتر می‌شود. به ازای N های بزرگتر-مساوی ۵۰۰ میتوان گفت که خطای گزاره‌ها بسیار کم شده و شکل تغییرات مد نظر با دقت بیشتری نمایان می‌شود. اگر گراف تصادفی با تعداد نود های ۵۰۰ را به عنوان یک گراف قابل قبول در نظر بگیریم، به ازای $C \geq 6$ ($p=C/(N-1)=0.012$) بزرگترین مولفه گراف دربرگیرنده‌ی کل نود های گراف خواهد بود و نود ایزوله‌ای نخواهیم داشت.

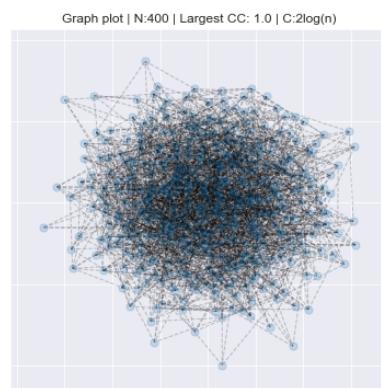
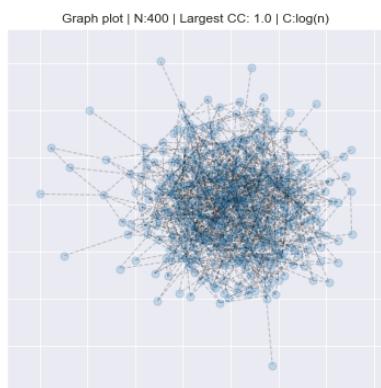
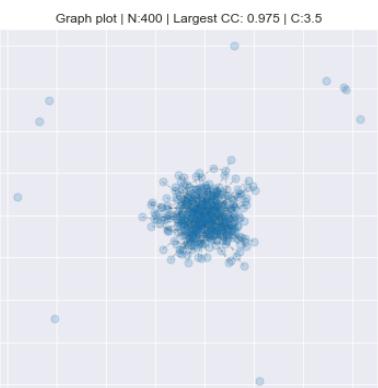
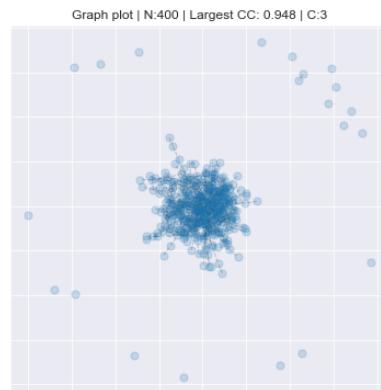
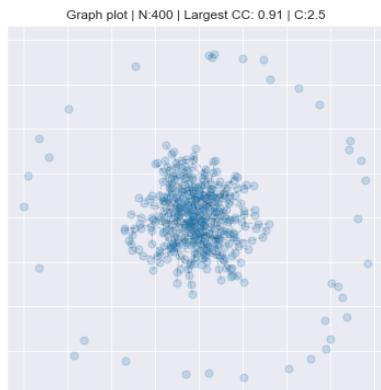
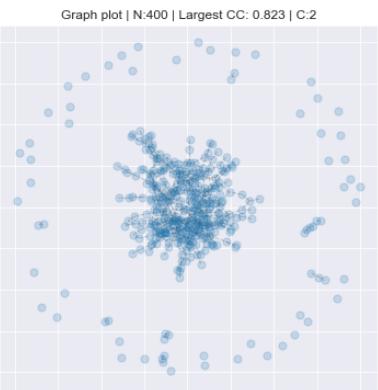
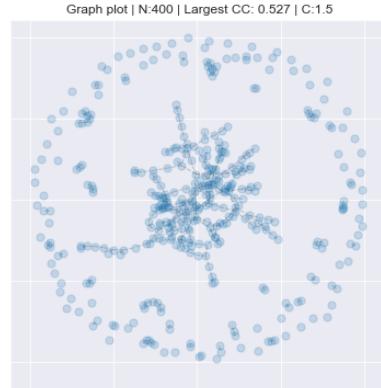
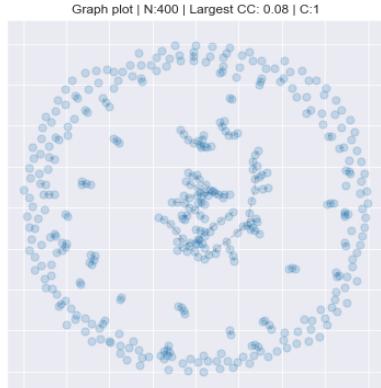
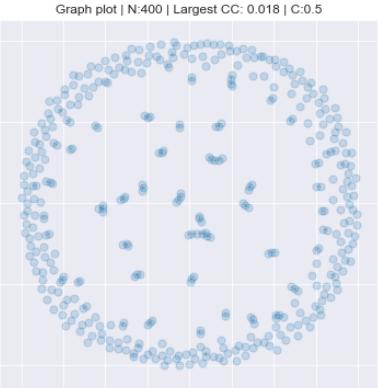
حال از طرفی اگر $C = \ln(N) = 8.9$ را نظر بگیریم، میتوان نتیجه گرفت که به ازای $C = \ln(N)$ قطعاً بزرگترین مولفه در گراف تشکیل شده و همه‌ی نود های گراف زیر مجموعه‌ای از آن قرار می‌گیرند؛ هر چند قبل تر در کلاس درس (صفحه ۸۳ اسلاید) این نکته را مشاهده کرده بودیم که وقتی N به سمت بی نهایت میل می‌کند، به ازای $p = C/(N-1)$ که در آن $C = \ln(N)$ می‌باشد، در گراف حاصل بزرگترین مولفه با احتمال ۱ تشکیل شده و تعداد نود های با درجه صفر وجود نخواهد داشت که این موضوع در این تمرین و در مثال گفته شده کاملاً اثبات گردید.



مسئله دوم: فعالیت بیشتر | مصوّر سازی تشکیل بزرگترین مولفه به ازای تغییرات C

در این قسمت بصورت بصری و با رسم نود های گراف و به ازای تغییرات C میخواهیم تشکیل بزرگترین مولفه گراف را ترسیم کنیم. برای این منظور تعداد نود ۴۰۰ را در نظر گرفته و به ازای ۹ مقدار برای C اقدام به ایجاد گراف کرده و نود های گراف و پیوند های بین شان را رسم می کنیم. نتایج بصورت صفحه بعد حاصل می شود و ملاحظه می کنیم که به ازای در نظر گرفتن

$$C = \ln(N)$$



مسئله سوم

مسئله سوم: الف

در گراف های دو بخشی میدانیم که وجود یال میتواند صرفا از یک نود در یک بخش به یک نود در بخش دیگر باشد و وجود یال در داخل هر بخش بین نود هایش مجاز نیست. حال اگر بخواهیم همه یال های ممکن در گراف را ایجاد نماییم، هر نود مانند A از بخش (۱) میتواند به همه نود های داخل بخش (۲) یال داشته باشد که تعداد آن با تعداد نود های موجود در بخش (۲) برابر است($N_2 = N_1$). حال ما در بخش (۱) به تعداد N_1 تا نود داریم که هر یک میتواند N_2 لینک را پذیرد لذا تعداد لینکی که کلا میتواند ایجاد شود برابر خواهد بود با : $N_1 * N_2$

مسئله سوم: ب

برای اینکه محاسبه کنیم چه تعداد لینک نمیتواند نسبت به حالت گراف عادی $N = N_1 + N_2$ رخ دهد، ابتدا تعداد کل لینک های ممکن را با فرمول $Max\ Edges = \frac{n*(n-1)}{2}$ محاسبه میکنیم و سپس از ان تعداد لینک های ممکن را کسر می کنیم که در بخش قبل برابر با $N_1 * N_2$ بدست آورده ایم. فرمول فوق نشان میدهد که اگر گراف ما کامل باشد چقدر یال خواهد داشت:

$$Max\ edges = \frac{n*(n-1)}{2} = \frac{(N_1+N_2)*(N_1+N_2-1)}{2} = \frac{N_1^2+N_2^2+2*N_1*N_2-N_1-N_2}{2}$$

$$Non-Possible\ Edges = Max\ edges - N_1 * N_2 = \frac{N_1^2+N_2^2+2*N_1*N_2-N_1-N_2}{2} - N_1 * N_2$$

مسئله سوم: ج

چگالی یک شبکه گراف برابر با نسبت تعداد یال های موجود به حداکثر یال های ممکن سنجیده می شود که اگر تعداد یال های موجود را برابر با L فرض کنیم، چگالی شبکه برابر با $\frac{L}{N_1 * N_2}$ خواهد بود. حال اگر تعداد نود های N_1 بسیار کم باشد، تعداد یال های موجود در گراف (L) از مرتبه N_2 خواهد بود چرا که هر یک از نود های مجموعه (۲) در صورت دارا بودن لینک، صرفا میتواند با مجموعه (۱) ارتباط داشته باشد و چون تعداد نود های آن بسیار بسیار کمتر از مجموعه (۲) است، لذا تعداد یال های گراف را میتوان با N_2 تخمین زد. حال با جایگذاری N_2 به جای L در رابطه ای فوق میتوان چگالی شبکه را بدین گونه بدست آورد:

$$\text{Density of network} = \frac{\#edges}{N1 * N2} = \frac{L}{N1 * N2} = \frac{N2}{N1 * N2}, N1 \ll N2 \rightarrow \text{Density} \approx \frac{N2}{N2} = 1$$

لذا میتوان نتیجه گرفت در صورتی که $N1 < N2$ برقرار باشد، گراف دو بخشی حاصل چگال تر حاصل خواهد شد.

مسئله سوم: a

همانطور که در بخش اول سوال اشاره شد، میدانیم که بین نود های هر بخش یال و اتصال برقرار نیست و هر یالی در گراف از یک بخش به یک بخش هست. حال فرض میکنیم میخواهیم میانگین درجه نود ها بخش یک را محاسبه کنیم. اگر تعداد یال

$$K1 = \frac{M}{N1}$$

حال در گام بعد میخواهیم میانگین درجه نود های بخش دوم را محاسبه کنیم، در مرحله قبل فرض کردیم تعداد یال خروجی از بخش اول برابر است با M لذا تعداد یال ورودی در بخش دوم نیز برابر خواهد بود با M و میتوانیم میانگین درجه نود های بخش

$$K2 = \frac{M}{N2}$$

هم اکنون اگر از هر یک از رابطه های فوق مقدار M را محاسبه و برابر گذاریم، خواهیم داشت:

$$N1 * K1 = M = N2 * K2 \rightarrow N1 * K1 = N2 * K2 \rightarrow \frac{K2}{K1} = \frac{N1}{N2}$$

مسئله چهارم

مسئله چهارم

برای حل این سوال یک کلاس پیاده سازی شده است که توسط توابع کدنویسی شده‌ی آن بتوان الگوریتم‌های حریصانه و CELF را اجرا نمود. کلاس مذکور با عنوان GRAPH پیاده سازی شده است که دارای چهار فیلد و چهار متدهای باشد که در زیر به جزئیات آن پرداخته می‌شود:

کلاس GRAPH:

- فیلد covers: این فیلد لیست نودهایی از گراف را در خود نگه میدارد که پوشش داده می‌شوند.
- فیلد influence: این فیلد لیست نودهای انتخابی را ذخیره و نگهداری می‌کند.
- فیلد graph: این فیلد گراف مد نظر در خود نگه میدارد.
- فیلد is_OutBreak: یک فیلد می‌باشد که نشان میدهد ما در حال حل سوال بیشینه کردن تاثیر هستیم(سؤال ۴) یا حل سوال کشف شیوع(سؤال ۵)

- متدهای get_cost: هزینه‌ی انتخاب یک نود بر اساس روش حل را بر می‌گرداند(برای حریصانه مقدار ۱ را بر می‌گرداند و...)
- متدهای get_gain: تعداد نودهایی را بر می‌گرداند که با انتخاب یک نود مفروض می‌توان از طریق آن به اینان رسید.(تعداد out هایی که قبلاً ملاقات نشده‌اند)
- متدهای get_outs: این فیلد out‌هایی را بر می‌گرداند. این متدهای بصورت دستی و با الگوریتم BFS پیاده سازی شده است.
- متدهای add_to_set: این متدهای نود دریافتی را به عنوان نود انتخابی برای مجموعه influence در نظر گرفته و مقدار covers در نظر گرفته و مقدار influence را بروز می‌کند.

مسئله چهارم: حریصانه

به جهت تئوری الگوریتم حریصانه بدین گونه عمل می‌کند که در هر مرحله gain (تعداد out‌های جدید) قابل افزایش که حاصل از اضافه کردن هر نود به مجموع جواب است را محاسبه کرده و سپس نودی را انتخاب و به مجموعه influence اضافه می‌کند

که بیشترین تعداد out جدید را به همراه دارد و این فرآیند به تعداد نود های انتخابی مدد نظر تکرار می شود. همانطور که مشخص است، مرتبه زمانی این الگوریتم بسیار بالاست چرا که به ازای انتخاب هر نود جدید برای مجموعه جواب بايستی N بار out محاسبه شود که هزینه محاسبه out هر یک نیز برابر با جست و جوی BFS برابر است؛ به صورت خلاصه میتوان بار محاسباتی و مرتبه اجرایی این الگوریتم را بصورت $O(k \cdot n \cdot R)$ در نظر گرفت که بترتیب از چپ به راست نشان دهنده تعداد نود های مد نظر برای انتخاب، تعداد گره ها، تعداد تکرار و تعداد یال های گراف می باشد(صفحه ۴۱ اسلاید درسی).

مسئله چهارم: CELF

ایده‌ای که در تئوری الگوریتم CELF مطرح شده است این است که هزینه و پاداش حاصل از انتخاب ها را در تصمیم گیری لحاظ کند بدین نحو که دو الگوریتم حریصانه مجزا را اجرا کند که یکی صرفاً مبتنی gain و دیگری مبتنی بر نسبت gain/cost در هر نود است؛ هر کدام از این دو اجرا مجموعه پاسخ خود را ارائه می کند که الگوریتم CELF بیشینه‌ی آن دو را به عنوان پاسخ نهایی خود انتخاب می‌کند. از جایی که در این سوال هزینه های انتخاب هر نود به عنوان influencer کلا ثابت و برابر با یک در نظر گرفته می شود، لذا اجرای الگوریتم حریصانه اولی پاسخ CELF خواهد بود. به جهت پاسخ های حاصل می‌توان این انتظار را داشت که پاسخ CELF در مجموع میتواند نتایجی همسطح یا بهتر از الگوریتم حریصانه را ارائه کند چرا که در دل خود حریصانه ساده را دارد. به جهت زمانی نیز CELF زمان بیشتری برای پاسخگویی نیاز دارد چرا که بايستی دوبار کامل الگوریتم حریصانه اجرا شود. برای سرعت بخشیدن به الگوریتم CELF نسخه های بهینه ای از حریصانه ارائه شده است که به جهت زمانی بسیار سریع تر میتواند به پاسخ برسد که در ادامه مورد بررسی قرار می‌گیرد.

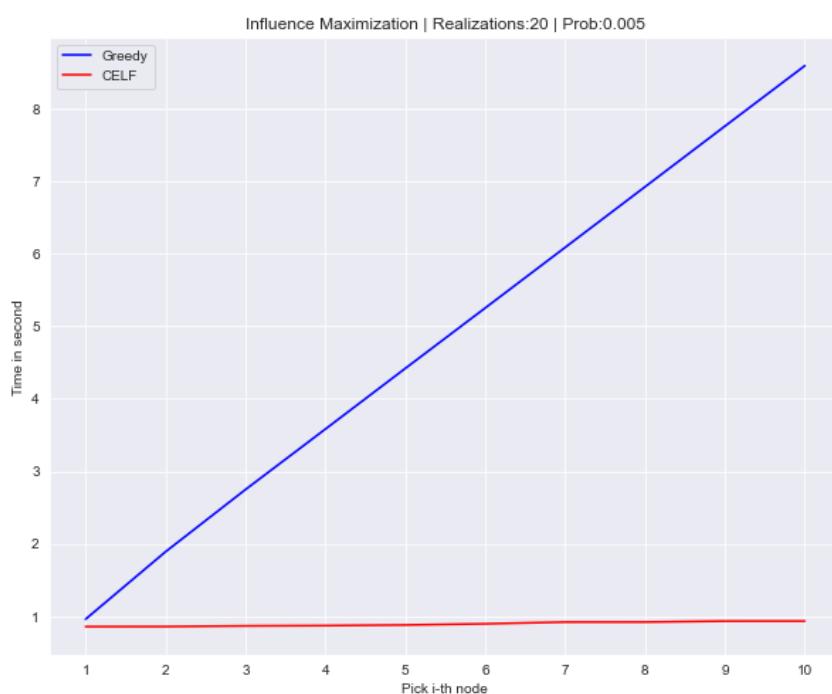
مسئله چهارم: Lazy Evaluation (سرعت بخشیدن به الگوریتم موجود)

ایده‌ای که در ادامه‌ی الگوریتم CELF برای بهبود سرعت محاسبات اجرا های حریصانه و رسیدن به پاسخ مطرح شده است مبتنی بر این اصل استوار است که gain حاصل از افزودن یک نودی مانند X به مجموعه انتخابی طی تکرار ثابت مانده و یا کاهش پیدا می‌کند؛ برای مثال اگر تعداد out های نود x (gain) در تکرار اول ۲۰ باشد، در تکرار های بعدی تعداد out های آن نمیتواند بیشتر از ۲۰ باشد چه بسا ممکن است تعدادی از out های آن توسط انتخاب قبلی پوشش داده شود و gain کنونی کمتر از ۲۰ حاصل شود. با توجه به نکته مذکور، نسخه Lazy Evaluation از الگوریتم حریصانه بدین صورت عمل می‌کند که در گام اول gain

(out) همه‌ی نود‌ها را حساب کرده و آنان را بر اساس gain نزولی مرتب می‌کند و اولین نود را به عنوان پاسخ انتخاب می‌کند؛ در گام بعدی سراغ نود دوم رفته و فقط gain آن را محاسبه می‌کند؛ اگر gain حاصل همچنان از gain نود بعدی بیشتر یا برابر باشد، با توجه به اینکه gain نود بعدی نمیتواند افزایش پیدا کرده باشد، همان نود دوم را انتخاب و به مجموعه جواب انتخاب اضافه میکند. حال اگر gain نود دوم کمتر از نود سوم باشد، یعنی لیست نود‌ها دیگر مرتب شده نمی‌باشد لذا نود دوم را بر اساس gain در جایگاه صحیح خود قرار داده و این فرآیند را برای نود جدیدی که در جایگاه دوم قرار میگیرد تکرار میکند و این توالی تا زمانی ادامه پیدا میکند که gain کنونی نود دوم بیشتر از نود سوم باشد. بدین صورت تعداد محاسبات gain برای نود‌های مختلف بشدت کاهش یافته و سرعت الگوریتم بشدت افزایش پیدا می‌کند.

مسئله چهارم: نتایج

در این سوال(۴) الگوریتم ساده حریصانه به همراه الگوریتم حریصانه CELF (نسخه Lazy Evaluation) برای پاسخ گویی مد نظر قرار گرفته است؛ در گام اول با احتمال ۰.۵ درصد برای فعال شدن یال‌ها realization مختلف با تابع get_realizations ایجاد میکنیم و در اجرای الگوریتم‌ها بیشینه‌سازی مجموع Gain‌ها در realization‌ها را هدف قرار میدهیم. برای الگوریتم حریصانه و گرفتن نتایج از تابع max_influence_lazy و برای دیگری از تابع max_influence_greedy استفاده میکنیم. هر دو الگوریتم را اجرا کرده و نمودار زمانی انتخاب نود‌ها را رسم می‌کنیم که بصورت زیر حاصل می‌شود تا بتوان بصورت عملی آنها را مقایسه نمود(مرجع زمانی هر یک از زمان شروع الگوریتم می‌باشد؛ به عبارتی دیگر تجمعی است):



همانطور که قابل مشاهده است در عمل و به جهت زمانی الگوریتم حریصانه عادی و CELF (نسخه Lazy Evaluation) باهم تفاوت بسیار زیادی دارند طوریکه برای انتخاب ۱۰ نود در الگوریتم حریصانه عادی ۱۰ برابر زمان بیشتری نسبت به الگوریتم CELF (نسخه Lazy Evaluation) نیاز است که نشان میدهد همان یکبار محاسبه Gain تمامی نود ها تقریبا برای شناسایی نود های مناسب کافی است(در هر تکرار صرفا محاسبه gain محدودی نیاز است) و نیاز نیست این فرآیند برای در هر تکرار برای همهی نود ها مجدد اجرا شود. نتایج دو الگوریتم فوق برای نود های انتخابی نیز بصورت زیر حاصل شده است:

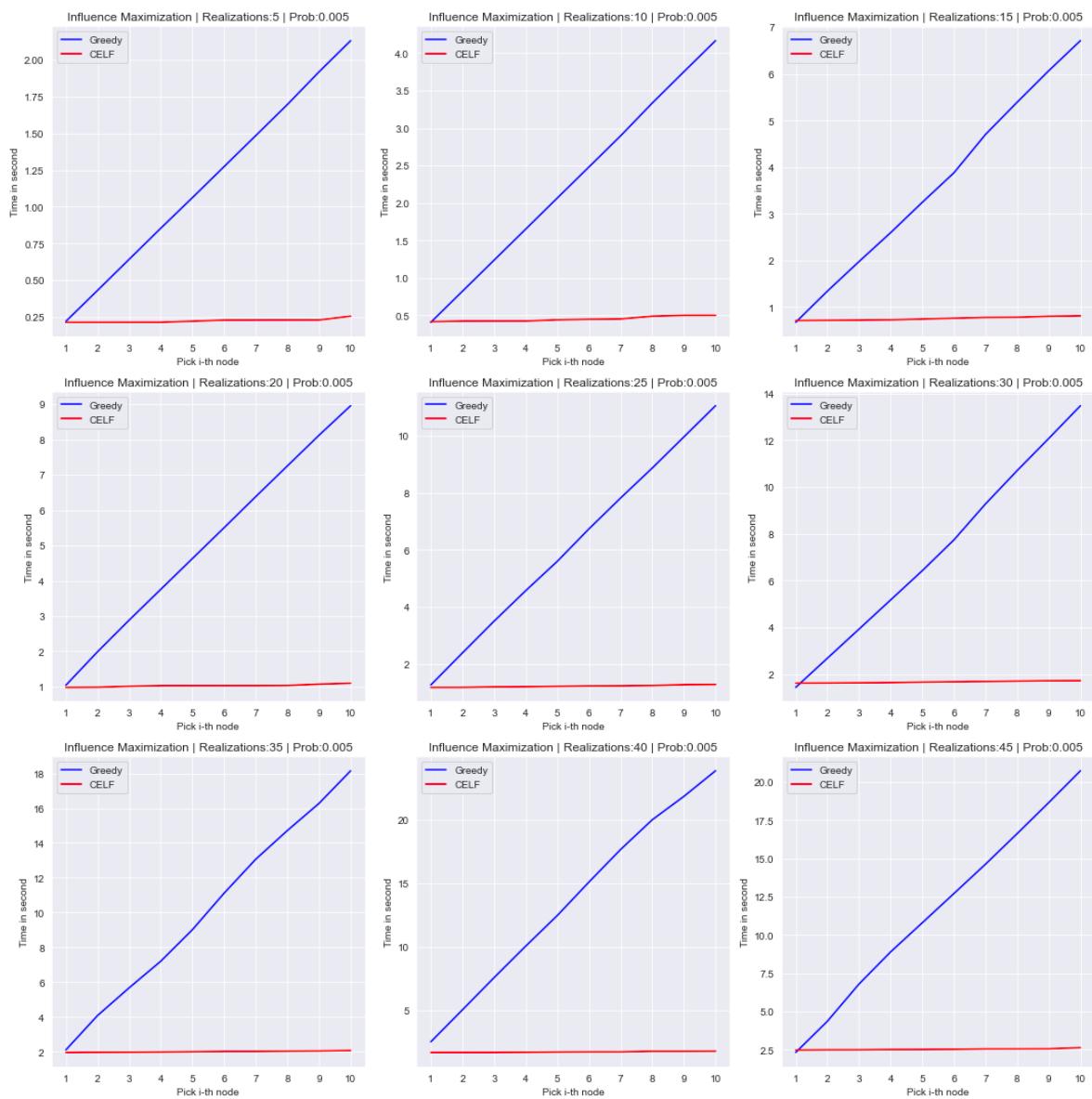
```
*****
Reporting for Greedy | Mean Spread in realizations(20):45.4 | Prob. of active edge:0.005
Time to select node #1(ID:53213) is 0.96 second and cost of it is equal to 1.0
Time to select node #2(ID:1086) is 1.89 second and cost of it is equal to 1.0
Time to select node #3(ID:62821) is 2.75 second and cost of it is equal to 1.0
Time to select node #4(ID:38109) is 3.58 second and cost of it is equal to 1.0
Time to select node #5(ID:92790) is 4.42 second and cost of it is equal to 1.0
Time to select node #6(ID:111161) is 5.25 second and cost of it is equal to 1.0
Time to select node #7(ID:106497) is 6.09 second and cost of it is equal to 1.0
Time to select node #8(ID:48189) is 6.92 second and cost of it is equal to 1.0
Time to select node #9(ID:36907) is 7.76 second and cost of it is equal to 1.0
Time to select node #10(ID:84122) is 8.59 second and cost of it is equal to 1.0
*****
Reporting for CELF | Mean Spread in realizations(20):45.4 | Prob. of active edge:0.005
Time to select node #1(ID:53213) is 0.86 second and cost of it is equal to 1.0
Time to select node #2(ID:1086) is 0.86 second and cost of it is equal to 1.0
Time to select node #3(ID:62821) is 0.87 second and cost of it is equal to 1.0
Time to select node #4(ID:38109) is 0.87 second and cost of it is equal to 1.0
Time to select node #5(ID:92790) is 0.88 second and cost of it is equal to 1.0
Time to select node #6(ID:111161) is 0.9 second and cost of it is equal to 1.0
Time to select node #7(ID:106497) is 0.92 second and cost of it is equal to 1.0
Time to select node #8(ID:48189) is 0.92 second and cost of it is equal to 1.0
Time to select node #9(ID:36907) is 0.93 second and cost of it is equal to 1.0
Time to select node #10(ID:84122) is 0.93 second and cost of it is equal to 1.0
```

به جهت نتیجه نهایی نود های انتخابی نیز همانطور که قابل انتظار است، نتایج هر دو الگوریتم یکسان حاصل شده است چرا که رویکرد انتخاب یک نود با هم یکی بوده است(انتخاب بر اساس out های جدید) و صرفا CELF (نسخه Lazy Evaluation) توانسته است با محدود کردن تعداد محاسبات در زمان سریع تری به جواب برسد. (مقدار عددی میانگین حداقل تاثیر که ۴۵.۴ است با احتمال ایجاد یال در realization ارتباط دارد و در صورت افزایش احتمال ایجاد یال، این مقدار نیز افزایش پیدا خواهد

(کرد)

مسئله چهارم: فعالیت بیشتر | بررسی تأثیر تعداد realization ها در زمان اجرا

در این بخش به ازای تعداد realization های مختلف میخواهیم بررسی کنیم آیا رفتار و عملکرد دو الگوریتم یاد شده به جهت realization زمانی تفاوتی میکنند یا خیر و اگر تفاوتی میکنند چه قدر است. برای این هدف ۹ بار فرآیند بخش قبل را به ازای تعداد realization های ۵ الی ۴۵ تکرار می کنیم که نتایج بصورت زیر حاصل می شود:



از مقایسه نمودار های فوق میتوانیم نتیجه بگیریم که تعداد realization های مورد استناد در اجرای الگوریتم یک تاثیر خطی در زمان اجرای هر دو الگوریتم داشته و در صورت افزایش تعداد realization ها به هر نسبت، زمان اجرای الگوریتم نیز تقریبا به همان نسبت افزایش پیدا می کند.

مسئله پنجم

مسئله پنجم

همانطور که در کلاس درس نیز مورد بحث قرار گرفته است، میتوان چندین روش و تابع هدف(objective function) برای کشف شیوه و شناسایی نود های مناسب برای قرار دادن سنسور تعريف و از آن استفاده کرد. یکی از این توابع هدف، کمینه کردن فاصله‌ی نود های انتخابی با نود های آلوده در گراف و حصول اطمینان از حضور نود های انتخابی در مسیر شیوه است. در این سوال من نیز با استناد به این تابع هدف پیاده ساز خود را انجام داده ام. برای این منظور میتوان معکوس مسئله بیشینه‌سازی تاثیر را در نظر گرفت بدین صورت که نود هایی مناسب قرار دادن سنسور برای کشف شیوه هستند که در out نود هایی بیشتری حضور داشته باشند(نه اینکه خود out بیشتری داشته باشند). برای حل کشف شیوه با روش مذکور میتوان جهت يال های گراف را معکوس کرده و مسئله‌ی بیشینه‌سازی تاثیر را روی آن اعمال نمود.(با این تفاوت که در اینجا هزینه و پاداش متفاوت است و نیاز است هر دو بخش الگوریتم CELF اجرا شده و بین آن ماکسیمم گرفته شود)

در سوال قبلی بصورت کامل الگوریتم های حریصانه، CELF و نسخه‌ی Lazy Evaluation آن بصورت کامل شرح داده شده است و برای جلوگیری از تکرار در اینجا مجدد بیان نمی شود. صرفا لازم است اشاره شود که در [این توضیحات](#) دیدیم که با استناد به این که Gain نود ها در طی تکرار افزایش پیدا نمی کند چگونه می‌توان به الگوریتم CELF که در دل خود الگوریتم حریصانه Lazy Evaluation را دارد سرعت بخشدید و نسخه‌ی Lazy Evaluation آن را مد نظر گرفت(روش افزایش سرعت استفاده از نسخه Lazy Evaluation میباشد).

در این سوال realization ۱۰ با احتمال ۱ درصد از گراف ایجاد کرده الگوریتم حریصانه و CELF را روی آن اعمال می‌کنیم که نتایج بصورت صفحه بعد حاصل می‌شود. همانطور که ملاحظه می‌شود الگوریتم حریصانه عادی با انتخاب ۲ نود و صرف ۹.۷۶ از بودجه ۱۰ تایی در طی ۰.۶۲ ثانیه به پایان رسیده است(بصورت حریصانه بودجه خود را سریع تمام کرده است) و بطور میانگین از شیوه بیماری به ۷۸.۴ نفر جلوگیری توانسته جلوگیری کند این در حالی است که الگوریتم CELF با اجرای دو الگوریتم حریصانه unit-cost lazy evaluation مجزا $1.72(1.33+0.39)$ ثانیه طول کشیده است. الگوریتم حریصانه اول CELF نسخه greedy evaluation با این تفاوت که در مقایسه مستقیم سریع تر از حریصانه عادی بوده است.

الگوریتم حریصانه دوم CELF (benefit-cost greedy) نسخه‌ی lazy evaluation توانسته است با صرف ۹.۶ از بودجه ۱۰ تایی و انتخاب ۱۲ نود از شیوع بیماری به ۱۸۱۶ نفر بصورت میانگین در بین realization های مختلف جلوگیری کند. الگوریتم CELF بیشینه‌ی این دو مورد مذکور را به عنوان پاسخ برミگرداند که بر اساس میانگین جلوگیری از شیوع، دومین نتیجه به عنوان پاسخ الگوریتم CELF انتخاب می‌شود.

در مجموع میتوان گفت که الگوریتم CELF با وجود صرف زمان اندکی بیشتر از زمان حریصانه عادی که قابل انتظار است (بواسطه اجرای دو الگوریتم حریصانه حتی نسخه‌ی lazy evaluation توانسته است به نتیجه‌ای تقریباً ۲.۵ برابر بهتر دست یابد که این نشان میدهد اعمال cost در زمان محاسبات هر نود میتواند نتیجه بهتری را رقم بزند.

```
*****
Reporting for Greedy
Mean Spread in realizations(10):78.4
Prob. of active edge:0.01
Sum Cost:9.76
Time to select node #1(ID:89732) is 0.31 second and cost of it is equal to 6.0
Time to select node #2(ID:38155) is 0.62 second and cost of it is equal to 3.76
*****
Reporting for CELF: unit-cost greedy
Minimized number of infected people in realizations(10):78.4
Prob. of active edge:0.01
Sum Cost:9.76
Time to select node #1(ID:89732) is 0.31 second and cost of it is equal to 6.0
Time to select node #2(ID:38155) is 0.39 second and cost of it is equal to 3.76
*****
Reporting for CELF: benefit-cost greedy
Minimized number of infected people in realizations(10):181.6
Prob. of active edge:0.01
Sum Cost:9.6
Time to select node #1(ID:104365) is 0.39 second and cost of it is equal to 0.56
Time to select node #2(ID:93176) is 0.39 second and cost of it is equal to 0.48
Time to select node #3(ID:47765) is 0.39 second and cost of it is equal to 0.64
Time to select node #4(ID:542) is 0.46 second and cost of it is equal to 0.72
Time to select node #5(ID:13109) is 0.53 second and cost of it is equal to 0.4
Time to select node #6(ID:14302) is 0.69 second and cost of it is equal to 0.96
Time to select node #7(ID:72658) is 0.71 second and cost of it is equal to 0.64
Time to select node #8(ID:19869) is 0.74 second and cost of it is equal to 1.92
Time to select node #9(ID:70339) is 0.77 second and cost of it is equal to 0.96
Time to select node #10(ID:123233) is 0.88 second and cost of it is equal to 1.52
Time to select node #11(ID:43025) is 1.3 second and cost of it is equal to 0.48
Time to select node #12(ID:102954) is 1.33 second and cost of it is equal to 0.32
```