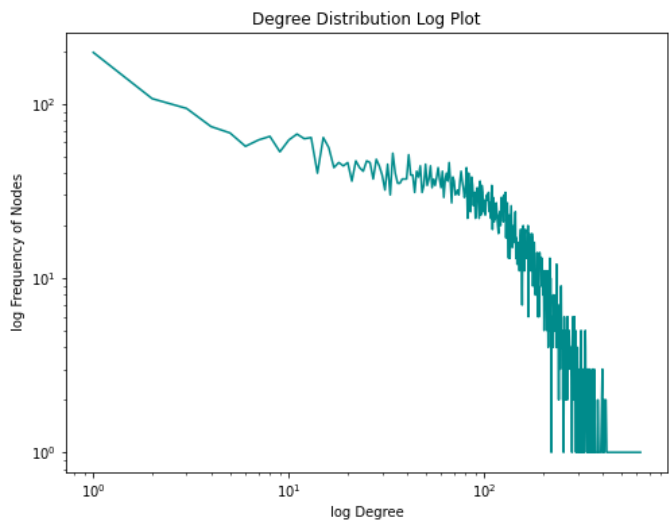
گزارش تمرین اول درس شبکه‌های پیچیده

فاطمه غلام‌زاده

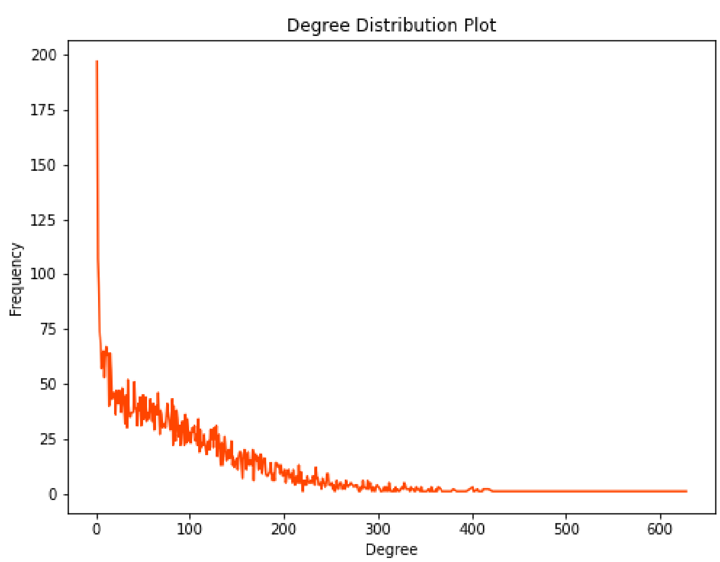
۹۹۱۳۱۰۰۳

سوال۱)

الف) نمودار توزیع درجه در مقیاس log-log در شکل زیر رسم شده است.



نمودار توزیع درجه در مقیاس عادی:



تحلیل نمودار: نمودار توزیع درجه این گراف به فرم نمودار توزیع درجه‌ی گراف‌های دنیای واقعی است. اصطلاحا گفته می‌شود که گراف‌های دنیای واقعی در نمودار توزیع درجه‌شان یک دم کلفت دارند به این معنی که نودهای با درجه بالا تعدادشان زیاد است. همانطور که مشاهده می‌شود نمودار توزیع درجه این گراف نیز این ویژگی را دارد بنابراین گراف پیچیده واقعی است.

ب) برای یافتن تعداد مسیرهای به طول ۷، ماتریس مجاورت را به توان ۷ می‌رسانیم و مجموع درایه‌های آن را به دست می‌آوریم و چون گراف غیرجهتدار است تقسیم بر ۲ می‌کنیم. تعداد مسیرهای به طول ۷:

6.128511995623708e+18

ج) چون این گراف متصل نیست یک قطر کلی نمی‌توان برای آن گزارش کرد بنابراین طول قطر هر یک از مولفه‌های همبندی آن را گزارش می‌کنیم:

9,1,2,1,1,2,1,1,1,1

قطر بزرگترین مولفه همبندی ۹ است. هم چنین طول طولانی‌ترین مسیری که در بزرگترین مولفه همبندی وجود دارد برابر است با : 5896

د) ضریب خوشه‌بندی عمومی:

0.3920165748018862

نتیجه: همانطور که مشاهده می‌شود ضریب خوشه بندی عمومی عدد ۰٫۳۹ است که عدد بزرگی است. یکی از ویژگی‌های مهم گراف‌های دنیای واقعی این است که ضریب خوشه‌بندی بالایی دارند و جوامع کوچک dense در آن‌ها وجود دارد. بنابراین از روی این ضریب خوشه‌بندی بالا می‌توان مشاهده کرد که مربوط به یک گراف پیچیده واقعی است.

ه)

تعداد اجزای متصل: 10

اندازه هر یک از اجزای متصل به صورت مرتب شده:

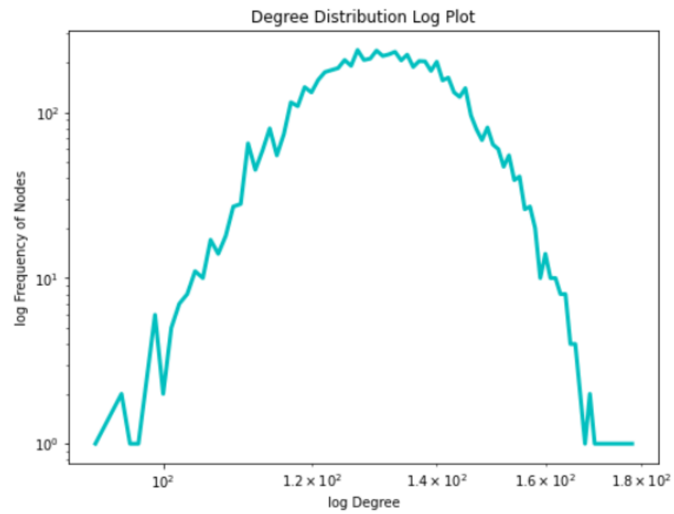
```
size of connected components:  [6575, 3, 3, 3, 2, 2, 2, 2, 2, 2]
```

بنابراین اندازه ۵ جزء بزرگ شبکه : 6575, 3, 3, 3, 2

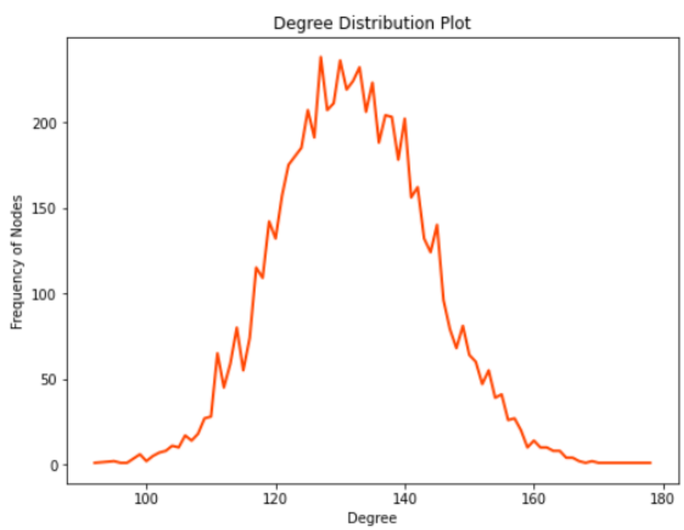
نتیجه: با توجه به اندازه‌های connected component ها می‌توان نتیجه گرفت این گراف دنیای واقعی است زیرا گراف‌های دنیای واقعی به این شکل هستند که یک giant connected compont دارند که اندازه‌اش با سایر connected component ها اختلاف زیادی دارد.

سوال دوم

الف) نمودار توزیع درجه در مقیاس log-log:



نمودار توزیع درجه در مقیاس عادی:



تحلیل: با توجه به نمودار این گراف پیچیده واقعی نیست زیرا گراف‌های دنیای واقعی در نمودار توزیع درجه‌شان یک دم کلفت دارند به این معنی که نودهای با درجه بالا تعدادشان زیاد است اما در این نمودار این مورد مشاهده نمی‌شود و نمودار توزیع درجه اش شبیه توزیع نرمال است.

ب) تعداد مسیرهای به طول ۷ مشابه سوال ۱ به دست می‌آید:

2.3494268585078287e+18

ج) اندازه قطر گراف: ۳

اندازه طولانی ترین مسیر در گراف: 6548

د) ضریب خوشه بندی: 0.040837859718073 09

نتیجه : همانطور که مشاهده می‌شود ضریب خوشه‌بندی عدد کوچکی است، در گراف‌های پیچیده واقعی ضریب خوشه‌بندی عدد بزرگی است که نشان‌دهنده چگال بودن اجتماع‌های محلی است. بنابراین چون در این گراف ضریب خوشه‌بندی کوچک است ویژگی گراف‌های پیچیده واقعی را ندارد.

ه) تعداد اجزای متصل: این گراف متصل است یعنی تعداد اجزای متصل آن ۱ است که شامل همه‌ی رئوس است:

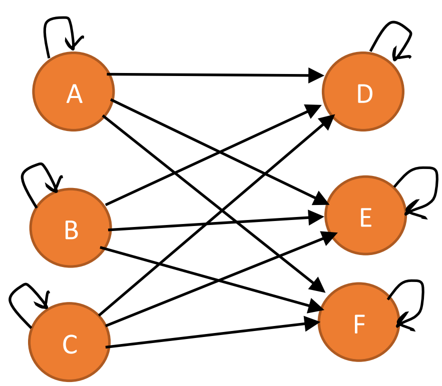
```
number of connected components is:  1
size of connected components:  [6596]
```

نتیجه: گراف‌های دنیای واقعی اینگونه هستند که یک giant connected component دارند، این گراف از این نظر که دارای یک مولفه همبندی بزرگ است شبیه گراف‌های دنیای واقعی است ولی معمولا گراف‌های دنیای واقعی connected نیستند اما این گراف متصل است، پس از این نظر با گراف پیچیده واقعی تفاوت دارد.

و) مدل GNP به عنوان یک مدل مرجع مورد استفاده قرار می‌گیرد که در محاسبه‌ی بسیاری از کمیت‌های گراف به ما کمک می‌کند. این کمیت‌ها را می‌توانیم با گراف‌های دنیای واقعی مورد مقایسه قرار بدهیم (مشابه عملی که در سوال ۱ و ۲ این تمرین انجام گرفت) همچنین این مدل به ما کمک می کند تا بفهمیم یک ویژگی خاص که در گراف وجود دارد تا چه میزان نتیجه یک فرآیند تصادفی است.

سوال سوم:

الف) گراف این سوال به این شکل است:



ماتریس مجاورت این گراف به این شکل است:

	A	B	C	D	E	F
A	1	0	0	1	1	1
B	0	1	0	1	1	1
C	0	0	1	1	1	1
D	0	0	0	1	0	0
E	0	0	0	0	1	0
F	0	0	0	0	0	1

ب) با انجام عملیات pagerank به صورت دستی تا 6 تا iteration نتایج زیر حاصل می‌شود:

	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5	Iteration 6
A	1/6	0.04166667	0.01041667	0.00260417	0.00065104	0.00016276
B	1/6	0.04166667	0.01041667	0.00260417	0.00065104	0.00016276
C	1/6	0.04166667	0.01041667	0.00260417	0.00065104	0.00016276
D	1/6	0.29166667	0.32291667	0.33072917	0.33268229	0.33317057
E	1/6	0.29166667	0.32291667	0.33072917	0.33268229	0.33317057
F	1/6	0.29166667	0.32291667	0.33072917	0.33268229	0.33317057
sum	1	1	1	1	1	1

همانطور که مشاهده می‌شود به نقطه تعادل قطعی نمی‌رسیم ولی هر چقدر جلو می‌رویم تغییرات کاهش پیدا می‌کنند به طوری که از یک جایی به بعد می‌توان از این تغییرات صرف نظر کرد. آن چه از این عملیات نتیجه می‌شود این است که مقدار امتیاز نودهای A و B و C که در دسته X قرار دارند به سمت صفر میل می‌کند و مقدار امتیاز نودهای D و E و F که در دسته Y قرار دارند به سمت 0.33 = 1/3 میل می‌کند.

در الگوریتم pagerank در واقع لینک‌های ورودی مهم هستند. اگر به ساختار گراف نگاه کنیم نودهای دسته‌ی Y به نظر نودهای مهمی هستند که همه صفحات دسته‌ی X به آن‌ها لینک می‌دهند ولی صفحاتی که در دسته X هستند به نظر می‌رسد که خودشان دارای اطلاعات مهمی نیستند و فقط به نودهای دسته Y لینک می‌دهند. بنابراین با مشاهده‌ی گراف هم می‌توان به این نتیجه رسید که بعد از مدتی نودهای دسته X تقریبا تمام امتیاز خود را به نودهای دسته Y می‌دهند به طوری که امتیاز خودشان به صفر میل خواهد کرد و همه‌ی امتیاز با توجه به ساختار متقارنی که گراف دارد، به طور مساوی بین نودهای دسته Y تقسیم خواهد شد و نودهای این دسته دارای rank بالاتری خواهند شد.

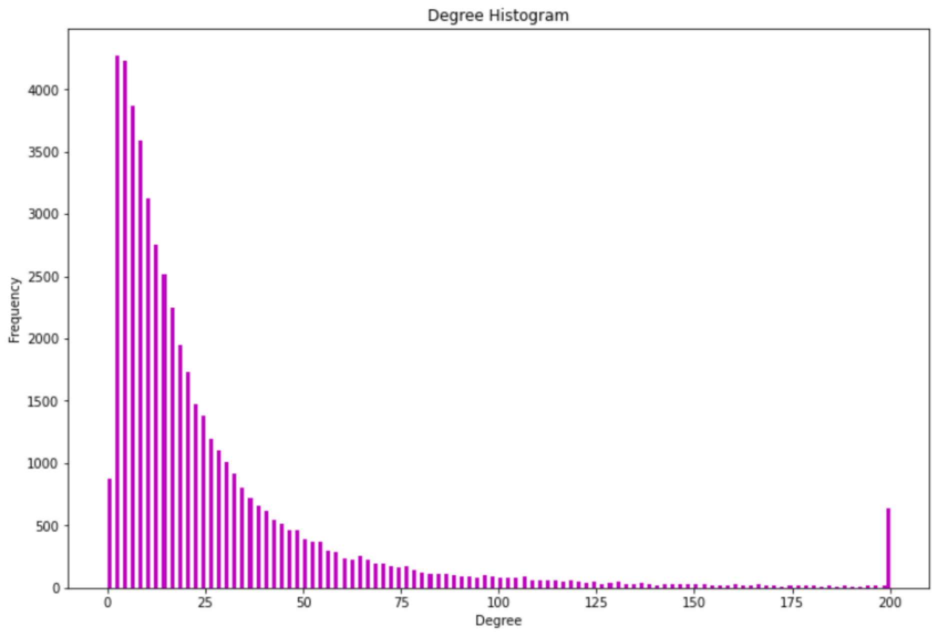
سوال چهارم:

الف) تعداد رئوس: 49995 تعداد یال‌ها: 661596

ب) میانگین درجه در گراف‌های جهتدار از رابطه‌ی E/N به دست می‌آید که در آن E تعداد یال‌ها و N تعداد رئوس است. بنابراین میانگین درجه این گراف برابر است با:

```
Average Degree is :  13.233243324332433
```

ج) هیستوگرام درجه‌های رئوس گراف:



تحلیل: همانطور که مشاهده می‌شود تعداد رئوس با درجه‌های کم خیلی زیاد است و در ادامه با افزایش درجه تعداد رئوس با درجه‌های بالاتر کمتر می‌شوند. با وجود اینکه رئوس با درجه بالا تعدادشان کم است، اما به صفر نمی‌رسد و به عنوان مثال حدود ۷۰۰ نود با درجه ۲۰۰ داریم که این یکی از ویژگی‌های گراف‌های پیچیده واقعی است که تعداد نودهای با درجات خیلی بالا در آن‌ها به صفر نمی‌رسد. بنابراین می‌توان نتیجه گرفت این گراف یک گراف پیچیده واقعی است.

ج) ۱۰ راسی که بیشترین pagerank را دارند به همراه مقدار pagerank شان نشان داده شده است:

```
{10164: 0.00047178462464182096,
 15496: 0.0003832072391779698,
 14689: 0.0003161636079585498,
 24966: 0.0002960344523447491,
 5148: 0.00028657154343845705,
 7884: 0.0002815570829444765,
 38123: 0.00027899004577850215,
 934: 0.00027506929328491607,
 45870: 0.0002604994958564457,
 20283: 0.0002558372953472026,
```

سوال پنجم

برای این سوال ۱۰ عدد realization از گراف تولید شده است و الگوریتم روی آن‌ها اجرا شده و پاسخ ها میانگین گرفته شده است. مجموعه‌های S یافت شده توسط دو الگوریتم در ادامه آورده شده است.

مجموعه S با استفاده از الگوریتم اول(Greedy):

```
greedy_hill_climbing output:
[412, 41, 4894, 6068, 2259, 1195, 1232, 5107, 957, 1286]
```

مجموعه S با استفاده از الگوریتم دوم (Lazy Greedy):

```
lazy_hill_climbing output:
[412, 41, 4894, 6068, 2259, 1195, 1232, 5107, 957, 1286]
```

تفاوت این دو روش:

۱) الگوریتم greedy به این صورت است: از یک مجموعه تهی برای S شروع می‌کنیم، در تکرار اول نودی را انتخاب می-کنیم که بیشترین اندازه out را دارد. سپس در هر تکرار فرض کنید که اعضای اول تا i-1 ام مجموعه S را پیدا کرده‌ایم. برای انتخاب نود i ام نودی را انتخاب می‌کنیم که در بین همه‌ی نودهای اضافه نشده به S بیشترین marginal gain را داشته باشد. یعنی نود u ای را انتخاب می‌کنیم که:

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

یعنی همه نودهای باقی‌مانده را به S اضافه می‌کنیم و f را حساب می‌کنیم، در نهایت مقدار f(S) برای هر u ای بیشتر شد آن نود u را انتخاب می‌کنیم و به S اضافه می‌کنیم.

شبه کد الگوریتم Greedy به این صورت است:

Algorithm:
- Start with $S_0 = \{\,\}$
- For $i = 1 \dots k$
  - Activate node u that max $f(S_{i-1} \cup \{u\})$
  - Let $S_i = S_{i-1} \cup \{u\}$

۲)الگوریتم **celf** برای کشف شیوع پیشنهاد شده است، این الگوریتم ۲ بخش دارد:

- بخش مربوط به افزایش سرعت، که قابل اعمال به هر مسئله مشابهی که تابع هدف submodular است، می باشد
- بخش مربوط به حالتی که هزینه داریم، که قابل اعمال به هر مسئله مشابهی که تابع هدف submodular است و هر گره هزینه ای دارد، می‌باشد.

در این مسئله چون هزینه نداریم فقط از بخش اول الگوریتم celf که مربوط به افزایش سرعت است استفاده می‌کنیم یعنی از ایده‌ی **lazy hill climbing** بهره می‌گیریم.

از خاصیت submodularity نتیجه می‌شود که :

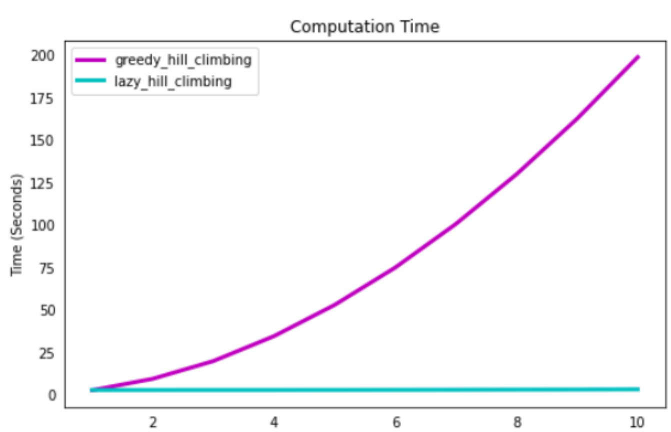
$$f(S_i \cup \{u\}) - f(S_i) \geq f(S_j \cup \{u\}) - f(S_j) \text{ for } i < j$$

یعنی هر چقدر iteration ها جلوتر می‌رود marginal gain نودها بیشتر نمی‌شود بلکه کاهش می‌یابد یا ثابت می‌ماند. از این نتیجه‌گیری می‌توانیم به این صورت استفاده کنیم: اگر نودی مثل v داشته باشیم و marginal gain آن را در iteration جدید حساب کنیم و مشاهده کنیم که مقدار آن از marginal gain نودهای دیگر در iteration قبلی بیشتر است در اینصورت می‌توانیم نتیجه بگیریم که marginal gain نود v در iteration جدید نیز مقدارش از بقیه بیشتر است چون هر چه iteration جلو می‌رود مقدار marginal gain نودها کاهش پیدا می‌کند یا ثابت می‌ماند. بنابراین بدون اینکه برای نودهای دیگر (غیر از v) در iteration جدید مقدار marginal gain را حساب کنیم مستقیما نود v را به خروجی می‌فرستیم.

در عمل چه اتفاقی می‌افتد؟

فرض کنید marginal gain نودها را از iteration = i-1 داریم و می‌خواهیم در iteration=i نودی را پیدا کنیم که marginal gain بیشتری دارد. این marginal gain نودها که از iteration های قبلی داریم به صورت مرتب شده و نزولی در یک لیست نگه می‌داریم. درا امین تکرار i فقط نودی که بالای این لیست قرار دارد را انتخاب می‌کنیم و برای این نود marginal gain را در تکرار جدید حساب می‌کنیم. اگر marginal gain این نود از marginal gain بقیه نودها بزرگتر بود نیازی به محاسبه marginal gain بقیه نودها نیست و همین نود انتخاب می‌شود. اگر این اتفاق نیفتاد و نودی وجود داشت که marginal gain اش بزرگتر از marginal gain این نود در تکرار جدید بود، این نود را در لیست مرتب در جای مناسب خودش قرار می‌دهیم به گونه‌ای که لیست مرتب بماند و این کار را برای عنصر دوم لیست که حالا در جایگاه عنصر اول قرار گرفته تکرار می‌کنیم. در بدترین حالت یک دور تا انتهای لیست را می‌رویم و برای تک تک نودها مقدار marginal gain را حساب می‌کنیم که هزینه‌اش مانند الگوریتم greedy است ولی در عمل این اتفاق نمی‌افتد. در عمل فقط کافی چند نود اول را چک کنیم. در اینجا داریم یک lazy evaluation انجام می‌دهیم یعنی در محاسبه تنبلانه عمل می‌کنیم و محاسبه‌ی marginal gain ها را تا جایی که امکان دارد عقب می‌اندازیم.

نمودار زیر زمان مورد نیاز برای الگوریتم Greedy و Lazy Greedy را برای مجموعه داده facebook 101 با اندازه k=10 نشان می‌دهد. همانطور که انتظار می‌رفت زمان الگوریتم Lazy در مقایسه با Greedy خیلی کمتر است.



سوال ششم

مجموعه یافته شده به منظور کشف شیوع با استفاده از ۲۰ عدد realization روی گراف به این صورت است:

```
celf output =  [556, 369, 390, 504, 664, 755, 1036, 1231, 1754, 2030, 2184, 2517, 2693, 2750, 2786, 3130, 3366, 3472, 3902, 4378, 4620, 4629, 4700, 5730, 6164, 6343, 6356, 2378, 4303, 4481, 675, 3733, 4853, 4668, 5416, 624, 5720, 6360, 2478, 2367, 3888, 5251, 4089, 2689, 3450, 4909, 2885, 3493, 1385, 3109, 3253, 5808, 1241, 1557, 5810, 2799, 126, 357, 1871, 2859, 5588, 6451, 4082, 4581, 3639, 1269, 980, 1540, 1116, 4767, 5411, 2443, 5675, 2476, 2624, 5061, 5700, 4749, 1027, 3217, 5811, 3955, 2673, 5850, 2710, 6580, 2475, 4021, 4468, 3663, 3302, 5419, 2645, 4901, 336, 478, 890, 3752, 3967, 3018, 3795, 1123, 5332, 4227, 6278, 5614, 4096, 231, 4852, 3195, 6063, 1040, 4705, 1789, 3897, 6255, 6036, 2171, 3058, 2943, 3290, 2520, 3395, 4595, 1046, 6055, 5781, 1316, 4314, 661, 5230, 3163, 5399, 855, 4878, 5120, 2402, 1396, 6005, 1197, 2186, 629, 4320, 4582, 850, 2331, 1328, 2715, 4075, 3406, 5337, 3093, 2523, 109, 754, 5098, 2471, 2977, 2017, 4421, 6279, 5099, 577, 2834, 4389, 2902, 2274, 3677, 3438, 5897, 2822, 593, 4011, 923, 2871, 3798, 1371, 200, 4961, 2390, 3698, 6257, 2753, 2768, 1677, 5290, 5816, 5520, 4835, 1346, 5087, 1610, 6075, 1808, 1875, 3244, 4883, 1104, 4744, 1264, 4428, 1765, 358, 1786, 780, 3783, 2537, 5024, 904, 3038, 170, 2935, 1317, 764, 2867, 5494, 6241, 1758, 5725, 1012, 549, 2502, 6350, 5874, 6071, 758, 4429, 22, 2007, 2538, 6115, 720, 1367, 3528, 5577, 5301, 2707, 4217, 2775, 3435, 1823, 5859, 4662, 1704, 3090, 2987, 3876, 1016, 5579, 873, 3600, 4994, 1930, 335, 3785, 2514, 2229, 3646, 2679, 4129, 5135, 6465, 2709, 728, 4522, 5848, 2708, 161, 1809, 4698, 2085, 3918, 1402, 1835, 4990, 1845, 2928, 3717, 4634, 4832, 3804, 5005, 5744, 4239, 1826, 6363, 1443, 584, 3801, 2620, 773, 6332, 4133, 4128, 445, 5717, 5309, 4993, 3183, 5179, 363, 646, 3786, 5707, 1229, 414, 3747, 6374, 2418, 2112, 2508, 6493, 4039, 5205, 6294, 2889, 4238, 5985, 5555, 6240, 3948, 2580, 3232, 2730, 3191, 4619, 4019, 2389, 5915, 6089, 4414, 879, 389, 1173, 3572]
```

الگوریتمی که از آن استفاده شده است الگوریتم CELF است.

توضیح الگوریتم CELF:

در این الگوریتم از هر دو معیار unit cost و benefit ratio استفاده می‌شود. یعنی یک بار الگوریتم greedy اجرا می‌شود به گونه‌ای که معیار انتخاب فقط marginal gain ای است که هر نود دارد و هزینه (cost) را در نظر نمی‌گیریم و فقط در انتهای در iteration هزینه را از بودجه کم می‌کنیم تا بیشتر از بودجه نشود. در اثر اجرای این الگوریتم یک مجموعه ’S به دست می‌آید.

سپس از معیار دوم استفاده می‌کنیم: مجددا الگوریتم Greedy را اجرا می‌کنیم اما این بار معیار نسبت (marginal reward(gain به هزینه(cost) را در نظر می‌گیریم. یعنی علاوه بر اینکه هر گاه یک نود انتخاب شد هزینه‌اش را از بودجه کم می‌کنیم، در انتخاب نود هم هزینه را دخالت می‌دهیم. در نتیجه‌ی اجرای این روش مجموعه ’’S به دست می‌آید.

در نهایت برای ’S و ’’S مقدار f شان را مقایسه می‌کنیم. هر کدام f بزرگتری داشت آن را انتخاب می‌کنیم:

S = argmax (f(S’) , f(S’’))

خطای الگوریتم CELF دارای یک باند است و اینطور نیست که مانند دو الگوریتم دیگر تا حد دلخواهی بد عمل کند. کیفیت آن حد پایینی دارد که برابر $\frac{1}{2}\left(1-\frac{1}{e}\right)$ است و از این بدتر عمل نمی‌کند یعنی مجموعه S ای که از این الگوریتم به دست می‌آید f(S) آن نسبت به f(OPT) تا حد دلخواهی پایین نیست.

راه حل برای افزایش سرعت الگوریتم: استفاده از روش lazy hill-climbing

همان طور که در توضیحات سوال ۵ هم گفته شد، در الگوریتم CELF هر چقدر iterationها را جلو می‌رویم مقدار marginal gain نودها نسبت به S کوچکتر می‌شود (از خاصیت submodularity این نتیجه حاصل می‌شود). از این نتیجه‌گیری می‌توانیم به این صورت استفاده کنیم:

اگر نودی مثل v داشته باشیم و marginal gain آن را در iteration جدید حساب کنیم و مشاهده کنیم که مقدار آن از marginal gain نودهای دیگر در iteration قبلی بیشتر است در اینصورت می‌توانیم نتیجه بگیریم که marginal gain نود v در iteration جدید نیز مقدارش از بقیه بیشتر است چون هر چه iteration جلو می‌رود مقدار marginal gain نودها کاهش پیدا می‌کند یا ثابت می‌ماند. بنابراین بدون اینکه برای نودهای دیگر (غیر از v) در iteration جدید مقدار marginal gain را حساب کنیم مستقیما نود v را به خروجی می‌فرستیم.

منابع و مراجع:

[1] https://networkx.org/

[2] https://stackoverflow.com/

[3] https://math.stackexchange.com/questions/3695601/calculate-the-number-of-triplets-in-a-graph

[4] https://hautahi.com/im_greedycelf

[5] https://www.geeksforgeeks.org/number-of-triangles-in-a-undirected-graph/

[6] https://notebook.community/SubhankarGhosh/NetworkX/4.%20Cliques,%20Triangles%20and%20Graph%20Structures%20(Instructor)