به نام خدا

مرضیه علیدادی – 810101236 **تکلیف اول شبکه‌های اجتماعی**

- کد من در فایلی از نوع ipynb. قرار دارد. که شامل بخش‌های کلی: توابع و 4 بخش دیگر برای تولید گراف در 4 بازه‌ی مختلف alpha و مقایسه‌ی میانگین فاصله در آن‌هاست.
- در هر بخش، خروجی تولید شده نمایش داده شده است و یک سری توضیحات در قالب text block و comment درج شده است.

- توابع تعریف شده در بخش functions:
- تابع make_sequence(n, t) : یک دنباله از n عدد با توزیع power law تولید می‌کند. این کار را با استفاده از توابع موجود در کتابخانه‌ی networkx انجام می‌دهد. این دنباله را به عنوان درجه‌ی گره‌های گرافمان استفاده خواهیم کرد. سپس بررسی می‌کند که آیا این دنباله می‌تواند به عنوان درجه‌های گره‌های یک گراف معتبر درنظر گرفته شوند یا خیر. n تعداد گره‌های گراف را تعیین می‌کند و t هم پارامتر گاما در توزیع power law است.
- تابع eval_k(seq) : دنباله‌ی درجات seq را دریافت می‌کند و میانگین درجات گره‌های گراف را به عنوان خروجی تولید می‌کند. این کار را با یک میانگین‌گیری ساده انجام می‌دهد.
- تابع generate_adj_vector(degreeseq) : با دریافت دنباله‌ی درجات گراف، لیست همسایگی‌های گراف را تولید می‌کند. برای این کار، ابتدا باید دنباله را به ماتریس همسایگی تبدیل کنیم. نمی‌توان مستقیما به لیست مجاورت تبدیل کرد. اما به دلیل اسپارس بودن ماتریس مجاورت، نسبت به لیست مجاورت فضای بسیار بیشتری نیاز دارد. پس، بعد از تولید هر سطر از ماتریس مجاورت، لیست مجاورت آن گره را تولید می‌کنم و آن فضای ماتریس را آزاد می‌کنم. به این شکل، دنباله درجات را به لیست همسایگی‌ها تبدیل کردم.
- greatest_component(visited, graph, node) : از گره‌ی ورودی node شروع به پیمایش گراف می‌کند و هر گره‌ای را که ملاقات می‌کند، درایه‌ی مربوط به آن را در لیست visited برابر 1 می‌کند. و بزرگ ترین connected component گراف را به عنوان خروجی برمیگرداند.

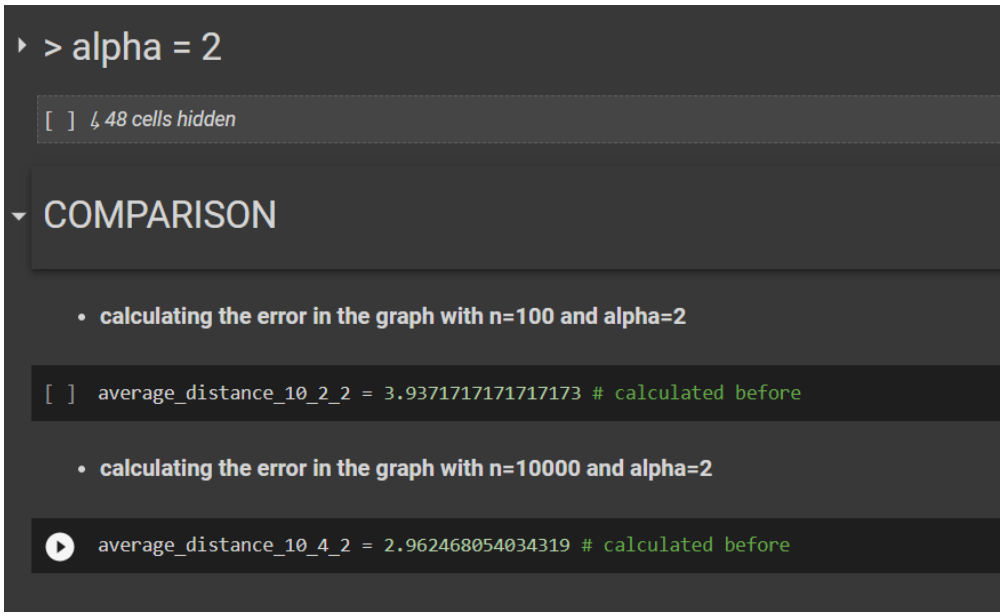
- BFS(adj, src, dest, v, pred, dist) و shortestDistance(adj, src, dest, v) : با صدا زدن تابع shortestDistance، با استفاده از الگوریتم BFS، برای گراف با لیست همسایگی adj، فاصله‌ی کوتاه‌ترین مسیر بین دو گره‌ی src و dest محاسبه می‌شود.

- برای هر یک از alpha های موردنظر، مراحل زیر را انجام دادم:
- ابتدا یک دنباله با توزیع power law با گاما و تعداد گره‌ی مورد نظر به عنوان دنباله درجات گراف تولید کردم. و میانگین درجاتش را محاسبه کردم. (گاهاً در این مرحله لازم شد که از درجه ی تمام گره ها به طور یکسان مقداری را کم کنم تا میانگین درجات گراف کاهش پیدا کند. سعی می‌شود که میانگین درجات در این مرحله به اندازه‌ی کافی کوچک باشد، تا بعداً با اضافه کردن یال جدید برای connected شدن گراف، میانگین درجات از 3 بیشتر نشود.)
- سپس آن را در scale عادی plot کردم، تا power law بودن آن را ببینم. سپس آن را به صورت log-log ، plot کردم تا ببینم در این نمایش، linear باشد.
- سپس لیست همسایگی‌های گراف را تولید کردم.
- بزرگ ترین connected component گراف را بدست آوردم.
- تمام connected component های گراف را نمایش دادم.
- سپس به ازای تمام connected component های کوچک موجود در گراف، یک یال بین یکی از گره های آن connected component و یکی از گره‌های بزرگترین connected component ایجاد کردم. حالا گراف کاملا connected شده است.
- بعد، میانگین درجات گراف را محاسبه کردم. اگر از 3 کمتر بود، به طور رندوم یال به گراف اضافه می‌کنم تا میانگین درجات برابر 3 شود. دنباله درجات و لیست همسایگی‌ها را هم برای گراف به روز می‌کنم.
- حالا بعد از این تغییرات دوباره دنباله درجات را در scale عادی plot کردم، تا power law بودن آن را ببینم. سپس آن را به صورت log-log ، plot کردم تا ببینم در این نمایش، linear باشد. و connected component های گراف را دوباره نمایش می‌دهم و چک می‌کنم که گراف کاملا connected شده باشد و فقط شامل یک connected component باشد.
- درنهایت، فاصله‌ی کوتاه‌ترین مسیر بین هر دو گره‌ی گراف را محاسبه می‌کنم. و از آن برای

کل گراف میانگین می‌گیرم. اگر تعداد گره‌های گراف زیاد باشـــد و نتوان برای هر دو گره‌ی گراف این محاسبات را انجام داد، این محاسبات را برای یک زیرمجموعه تصادفی از گراف انجام می‌دهم.

- در آخر، در بخش‌های comparison، مقدار فاصـــله میانگینی که با هر گاما و تعداد گره‌ی مشـــخص باید بدســـت آید را با اســـتفاده از فرمول متناظر حســـاب کردم. و برای تمام گراف‌هایی که با گامای یکســـان و تعداد گره‌های متفاوت ایجاد کردم، آن ها را با مقدار بدست آمده با فرمول مقایسه کردم و مقدار خطا را بدست آوردم. در نهایت مقدار خطای بدست آمده از گراف‌های با گامای یکسان و تعداد گره‌های متفاوت را مقایسه کردم تا این نتیجه بدست آید که با افزایش تعداد گره‌ها، خطا کمتر می‌شود و میانگین فاصله‌ی گراف دقیق‌تر می‌شود.

- نتایج مقایسه گراف‌های هر کدام از گاما ها:
- **گاما مساوی 2:**



دو گراف با تعداد گره‌های 100 و 10000 ایجاد کردم. مشـخص اسـت که روند زیاد شـدن میانگین فاصله در دو گراف به صورت خطی است و مطابق چیزی ست که برای گراف‌های با این گاما مدنظر است.

- **گاما بین 2 و 3 (اینجا برابر 2.8):**

سه گراف با تعداد گره‌های 100 و 10000 و 100000 ایجاد کردم.

- خطا در گراف با 100 گره مطابق شکل زیر برابر 224 درصد بود.

```
> 2 < alpha < 3

[ ] 69 cells hidden

COMPARISON

• calculating the error in the graph with n=100 and alpha=2.8

[35] from math import log
     expected_value_10_2_2plus = log(log(100)) / log(1.8)
     expected_value_10_2_2plus

     2.598186922226645

[36] average_distance_10_2_2plus = 8.436767676767676 # calculated before
     error_percent_10_2_2plus = (abs(average_distance_10_2_2plus - expected_value_10_2_2plus) / expected_value_10_2_2plus) * 100
     error_percent_10_2_2plus

     224.717502216406
```

- خطا در گراف با 10000 گره مطابق شکل زیر برابر 149 درصد بود.

```
• calculating the error in the graph with n=10000 and alpha=2.8

expected_value_10_4_2plus = log(log(10000)) / log(1.8)
expected_value_10_4_2plus

3.777436507066021

[54] average_distance_10_4_2plus = 9.412525252525253 # calculated before
     error_percent_10_4_2plus = (abs(average_distance_10_4_2plus - expected_value_10_4_2plus) / expected_value_10_4_2plus) * 100
     error_percent_10_4_2plus

     149.17759001159416
```

- خطا در گراف با 100000 گره مطابق شکل زیر برابر 79 درصد بود.

```
• calculating the error in the graph with n=100000 and alpha=2.8

[78] expected_value_10_6_2plus = log(log(100000)) / log(1.8)
     expected_value_10_6_2plus

     4.157070079310074

[79] average_distance_10_6_2plus = 7.448163265306122 # calculated before
     error_percent_10_6_2plus = (abs(average_distance_10_6_2plus - expected_value_10_6_2plus) / expected_value_10_6_2plus) * 100
     error_percent_10_6_2plus

     79.16857602126959
```

همانطور که مشخص است، با افزایش تعداد گره‌های گراف، میانگین فاصله دقیق تر شده و خطای آن نسبت به مقدار مورد نظر کاهش یافته است.

- **گاما مساوی 3:**

دو گراف با تعداد گره‌های 100 و 10000 و ایجاد کردم.

- خطا در گراف با 100 گره مطابق شکل زیر برابر 194 درصد بود.

```
> alpha = 3

[ ] 47 cells hidden

COMPARISON

• calculating the error in the graph with n=100 and alpha=3

[112] from math import log
      expected_value_10_2_3 = log(100) / log(log(100))
      expected_value_10_2_3

      3.0154738238809906

[113] average_distance_10_2_3 = 8.86909090909091 # calculated before
      error_percent_10_2_3 = (abs(average_distance_10_2_3 - expected_value_10_2_3) / expected_value_10_2_3) * 100
      error_percent_10_2_3

      194.11931348408012
```

- خطا در گراف با 10000 گره مطابق شکل زیر برابر 29 درصد بود.

```
• calculating the error in the graph with n=10000 and alpha=3

[184] from math import log
      expected_value_10_4_3 = log(10000) / log(log(10000))
      expected_value_10_4_3

      4.148191313801706

[185] average_distance_10_4_3 = 2.9396984924623117 # calculated before
      error_percent_10_4_3 = (abs(average_distance_10_4_3 - expected_value_10_4_3) / expected_value_10_4_3) * 100
      error_percent_10_4_3

      29.133005927632666
```

همانطور که مشخص است، با افزایش تعداد گره‌های گراف، میانگین فاصله دقیق تر شده و خطای آن نسبت به مقدار مورد نظر کاهش یافته است.

- **گاما بزرگ‌تر از 3:**

سه گراف با تعداد گره‌های 100 و 10000 و ایجاد کردم.

- خطا در گراف با 100 گره مطابق شکل زیر برابر 90 درصد بود.

```
> alpha > 3

[ ] 71 cells hidden

COMPARISON

• calculating the error in the graph with n=100 and alpha>3

[1021] from math import log
       expected_value_10_2_3plus = log(100)
       expected_value_10_2_3plus

       4.605170185988092

[1111] average_distance_10_2_3plus = 8.773737373737374 # calculated before
       error_percent_10_2_3plus = (abs(average_distance_10_2_3plus - expected_value_10_2_3plus) / expected_value_10_2_3plus) * 100
       error_percent_10_2_3plus

       90.5192863541235
```

- خطا در گراف با 10000 گره مطابق شکل زیر برابر 142 درصد بود.

```
• calculating the error in the graph with n=10000 and alpha>3

[1023] from math import log
       expected_value_10_4_3plus = log(10000)
       expected_value_10_4_3plus

       9.210340371976184

[1154] average_distance_10_4_3plus = 22.36686868686869 # calculated before
       error_percent_10_4_3plus = (abs(average_distance_10_4_3plus - expected_value_10_4_3plus) / expected_value_10_4_3plus) * 100
       error_percent_10_4_3plus

       142.84519120404258
```

- خطا در گراف با 100000 گره مطابق شکل زیر برابر 1196 درصد بود.

```
• calculating the error in the graph with n=100000 and alpha>3

[1155] from math import log
       expected_value_10_6_3plus = log(100000)
       expected_value_10_6_3plus

       11.512925464970229

       average_distance_10_6_3plus = 149.3030612244898 # calculated before
       error_percent_10_6_3plus = (abs(average_distance_10_6_3plus - expected_value_10_6_3plus) / expected_value_10_6_3plus) * 100
       error_percent_10_6_3plus

       1196.8299124211856
```

متاسفانه در این گاما، افزایش دقت با افزایش تعداد گره ها اثبات نشد.