

هو الغيور



دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

۹۸۱۳۱۰۵۹

زهرا دهقانیان

۹۷۱۳۱۰۱۷

علی اکبر بدری

پروژه پایانی درس الگوریتم‌های شبکه‌های پیچیده

پائیز ۹۹

کد این تمرین در مخزن گیت‌هاب زیر قرار داده شده است:

https://github.com/AliAkbarBadri/graph_centrality

سوال یک.

خواندیم و متوجه شدیم!

سوال دو.

کد این قسمت در لینک زیر قرار داده شده است:

<https://github.com/AliAkbarBadri/graph-centrality/blob/main/centrality.ipynb>

نتایج زمانی نیز در آدرس‌های زیر قرار داده شده‌اند:

https://github.com/AliAkbarBadri/graph-centrality/blob/main/results/graph_time.json

https://github.com/AliAkbarBadri/graph-centrality/blob/main/results/tree_time.json

نتایج نیز در دو لینک زیر موجود هستند:

https://github.com/AliAkbarBadri/graph-centrality/blob/main/results/graph_centrality.json

https://github.com/AliAkbarBadri/graph-centrality/blob/main/results/tree_centrality.json

دادگانی که روی آن‌ها تست شده‌اند:

- **bn-mouse_visual-cortex_1** 29 44 (<http://networkrepository.com/bn-mouse-visual-cortex-1.php>)
- **ca-sandi_auths** 86 124 (http://networkrepository.com/ca_sandi_auths.php)
- **reptilia-tortoise-network-lm** 45 106 (<http://networkrepository.com/reptilia-tortoise-network-lm.php>)
- **road-chesapeake** 39 170 (<http://networkrepository.com/road-chesapeake.php>)
- **rt-retweet** 96 117 (http://networkrepository.com/rt_retweet.php)

به وضوح اختلاف زیادی در زمان‌های اجرای دو الگوریتم گرافی و درختی وجود دارند و درختی بسیار کمتر طول می‌کشد. در گراف‌های با اندازه‌ی کوچکتر اختلاف در حد چند دقیقه و در گراف‌های با اندازه‌ی

بزرگتر اختلاف به چند ساعت نیز می‌رسد. اعداد زیر به ثانیه هستند. البته در دو گراف آخر با توجه به مولتی پراسس کردن کد، اکثر گره‌ها در حد چند دقیقه بدست آمدند و چند گره خیلی طول کشیدند.

```
{  
  "ca-sandi_auths": 0.2460329532623291,  
  "reptilia-tortoise-network-lm": 1.1568102836608887,  
  "bn-mouse_visual-cortex_1": 0.0065228939056396484,  
  "road-chesapeake": 0.46460986137390137,  
  "rt-retweet": 0.26991796493530273  
}
```

```
{  
  "ca-sandi_auths": 817.7409603595734,  
  "reptilia-tortoise-network-lm": 1614.1442775726318,  
  "bn-mouse_visual-cortex_1": 0.44628357887268066,  
  "road-chesapeake": 17560.795307652936302,  
  "rt-retweet": 13916.38764382746283,  
}
```

سوال سه.

کدهای این بخش در لینک زیر قرار داده شده‌اند:

<https://github.com/AliAkbarBadri/graph-centrality/blob/main/axiom.ipynb>

در ادامه به توضیح هر اکسیم و نحوه اثبات آن خواهیم پرداخت، فقط قابل توجه است که به دلیل زمان اجرای طولانی `count_subgraph`، محاسبه مرکزیت‌ها را بر روی گراف‌هایی با اندازه کمتر از ۵۰ اجرا میکنیم:

مفهوم اکسیم اول واضح است، منظور از این اکسیم این است که در گراف با اضافه شدن یال، مقدار معیار مرکزیت کاهش نیابد. به همین منظور، از هر گراف ۱۰ نود رندوم انتخاب می‌کنیم، سپس به ازای هر کدام مقدار مرکزیت را محاسبه میکنیم و سپس یک یال رندوم انتخاب کرده و به گراف اضافه میکنیم و بار دیگر معیار را حساب میکنیم. همانطور که در خروجی مشخص است، این اکسیم در تمامی اجراها درست است و به طور تجربی اثبات میشود.

```
Graph: bn-mouse_visual-cortex_1
-----
Success rate tree centrality = 100%
Success rate subgraph centrality = 100%
-----
Graph: reptilia-tortoise-network-lm
-----
Success rate tree centrality = 100%
Success rate subgraph centrality = 100%
-----
Graph: road-chesapeake
-----
Success rate tree centrality = 100%
Success rate subgraph centrality = 100%
```

اکسیم دوم می‌گوید با اضافه کردن یال به گراف اردر نودها بهم نمیریزد، یعنی اگر نودی در گراف میزان مرکزیت بیشتری از نود دیگر دارد، با افزودن یال نیز مقدار مرکزیت آن بیشتر باقی می‌ماند. این اکسیم را نیز به این صورت تست می‌کنیم که از هر گراف ۱۰ بار ۲ نود دلخواه و یک یال دلخواه انتخاب میکنیم و مقدار معیار

ه‌ا را یک بار با حذف یال دلخواه و یک بار با وجود یال دلخواه اندازه می‌گیریم و اگر ترتیب حفظ شده بود، در این صورت به عنوان یک اجرای تایید و اگر حفظ نشد بود، به عنوان یک اجرای رد در نظر می‌گیریم. خروجی این بخش به صورت زیر است:

```
Graph: bn-mouse_visual-cortex_1
-----
Success rate tree centrality = 100%
Success rate subgraph centrality = 100%
-----
Graph: reptilia-tortoise-network-lm
-----
Success rate tree centrality = 100%
Success rate subgraph centrality = 100%
-----
Graph: road-chesapeake
-----
Success rate tree centrality = 100%
Success rate subgraph centrality = 100%
```

اکسپیم سوم می‌گوید که به ازای هر نود دلخواه در گراف، میزان مرکزیت از نود ۰ در گراف هم ساین خطی بیشتر باشد. در واقع می‌گوید که کمترین مرکزیت همیشه متعلق به نود ۰ در گراف خطی باشد. برای اثبات این موضوع نیز در هر گراف برای ۱۰ نود رندوم (پس از حذف نود های isolated)، میزان مرکزیت در گراف را محاسبه کردیم و سپس میزان مرکزیت نود ۰ در گراف خطی معادل را نیز حساب کردیم. خروجی این بخش به صورت زیر است :

```
Graph: bn-mouse_visual-cortex_1
-----
Success rate tree centrality = 20%
Success rate subgraph centrality = 50%
-----
Graph: reptilia-tortoise-network-lm
-----
Success rate tree centrality = 40%
Success rate subgraph centrality = 30%
-----
Graph: road-chesapeake
-----
Success rate tree centrality = 80%
```

```
Success rate subgraph centrality = 60%
```

علت این رخداد این امر، این است که گراف متصل نیست و از چندین کامپوننت تشکیل شده. در توضیحات اکسیم فوق نیز آمده است که در گراف های **connected** صدق می کند.

برای اثبات اکسیم ۴، یک مشکل اساسی صدق نکردن شرط اولیه، یعنی یافتن نودی که در زیر گراف مقدار مرکزیت کمتری داشته باشد بود، در واقع برای این کار در دو حلقه تو در تو، به ازای تمام اندازه ها برای زیر گراف و برای تمام نودها باید چک کنیم که ابا شرط برقرار است یا نه و در صورتی که برقرار است باید تک به تک یال هایی که عضو گراف نیستند را به زیر گراف اضافه کنیم و ببینیم که آیا سبب افزایش میزان مرکزیت می شوند یا نه و این عملیات را تا رسیدن به اندازه مرکزیت در گراف اصلی ادامه دهیم. در اینجا دقیقاً به همین ترتیب عمل میکنیم و اگر تابع بتواند به **sequence** یال هایی که سبب افزایش معیار تا رسیدن به مرکزیت در گراف اصلی بشود، برسد، فرض میکنیم یک مثال + است و اگر نتواند دست یابد، به عنوان یک مثال منفی تلقی میشود. خروجی این بخش :

```
Graph: bn-mouse_visual-cortex_1
-----
Success rate tree centrality = 100%
-----
Graph: reptilia-tortoise-network-lm
-----
Success rate tree centrality = 44%
-----
Graph: road-chesapeake
-----
Success rate tree centrality = 100%
-----
```


برای اثبات اکسیم ۵ یک بحث اساسی که وجود دارد، دسترسی نداشتن به گراف **infinite** است. پس در اینجا مجبوریم به این صورت عمل کنیم که به ازای یک عدد دلخواه (در بازه ذکر شده در مقاله) بینیم می توان نودی با آن مقدار مرکزیت بدست آورد یا نه. در واقع چون هر گراف زیرگراف خودش هم محسوب می شود، در اینجا اگر بتوانیم نودی با مرکزیت بیشتر بیابیم، این اکسیم ثابت خواهد شد. خروجی این بخش به صورت زیر است :

```
Graph: bn-mouse_visual-cortex_1
-----
Success rate tree centrality = 100%
Success rate subgraph centrality = 100%
-----
Graph: reptilia-tortoise-network-lm
-----
Success rate tree centrality = 100%
Success rate subgraph centrality = 100%
-----
Graph: road-chesapeake
-----
Success rate tree centrality = 100%
Success rate subgraph centrality = 100%
```

سوال چهار.

کدهای این سوال در انتهای نوتبوک زیر (بخش‌های Eigenvector, Betweenness Centrality و Centrality و Degree Centrality) قرار داده شده‌اند:

<https://github.com/AliAkbarBadri/graph-centrality/blob/main/centrality.ipynb>

کدهای این بخش در لینک زیر قرار داده شده‌اند:

<https://github.com/AliAkbarBadri/graph-centrality/blob/main/correlation.ipynb>

نتایج محاسبه‌ی مرکزیت‌های دیگر نیز در لینک‌های زیر قرار داده شده‌اند:

https://github.com/AliAkbarBadri/graph-centrality/blob/main/results/betweenness_centrality.json

https://github.com/AliAkbarBadri/graph-centrality/blob/main/results/degree_centrality.json

https://github.com/AliAkbarBadri/graph-centrality/blob/main/results/eigenvector_centrality.json

نتایج همبستگی بین انواع مرکزیت‌ها در تصویر زیر آورده شده است. همانطور که مشخص است بین دو مرکزیت گراف‌ی و درختی مطرح شده در مقاله، دو همبستگی بالایی در سه گراف از پنج گرافی آزمایش شده وجود دارد.

همچنین این دو همبستگی قابل قبولی با degree centrality (دو الی سه تا از پنج تا گراف) دارند که این

موضوع در خود مقاله نیز اشاره شده بود؛ اما در هیچ کدام از گراف ها بین این دو و دو معیار مرکزیت

eigenvector centrality و betweenness centrality همبستگی خاصی مشاهده نمی شود.

```
bn-mouse_visual-cortex_1
  betweenness  degree  eigenvector  graph  tree
betweenness    1.000000  0.097626   -0.090741 -0.057167 -0.058548
degree         0.097626  1.000000   -0.178116  0.756884  0.762884
eigenvector    -0.090741 -0.178116    1.000000 -0.112443 -0.114604
graph         -0.057167  0.756884   -0.112443  1.000000  0.999516
tree          -0.058548  0.762884   -0.114604  0.999516  1.000000
```

```
ca-sandi_auths
  betweenness  degree  eigenvector  graph  tree
betweenness    1.000000  0.704289   -0.035054 -0.030455 -0.033335
degree         0.704289  1.000000    0.147261  0.072563  0.005460
eigenvector    -0.035054  0.147261    1.000000 -0.017255 -0.012530
graph         -0.030455  0.072563   -0.017255  1.000000  0.097924
tree          -0.033335  0.005460   -0.012530  0.097924  1.000000
```

```
reptilia-tortoise-network-lm
  betweenness  degree  eigenvector  graph  tree
betweenness    1.000000  0.751974   -0.109496 -0.066926  0.103166
degree         0.751974  1.000000    0.022976  0.027498  0.208281
eigenvector    -0.109496  0.022976    1.000000 -0.040686 -0.104943
graph         -0.066926  0.027498   -0.040686  1.000000  0.095375
tree          0.103166  0.208281   -0.104943  0.095375  1.000000
```

```
road-chesapeake
  betweenness  degree  eigenvector  graph  tree
betweenness    1.000000  0.131260   -0.080542 -0.140952 -0.143659
degree         0.131260  1.000000   -0.124686  0.829440  0.837439
eigenvector    -0.080542 -0.124686    1.000000 -0.049794 -0.054402
graph         -0.140952  0.829440   -0.049794  1.000000  0.988249
tree          -0.143659  0.837439   -0.054402  0.988249  1.000000
```

```
rt-retweet
  betweenness  degree  eigenvector  graph  tree
betweenness    1.000000  0.812015   -0.028507  0.062972  0.018868
degree         0.812015  1.000000    0.104836  0.443383  0.315248
eigenvector    -0.028507  0.104836    1.000000 -0.019925 -0.014476
graph         0.062972  0.443383   -0.019925  1.000000  0.765244
tree          0.018868  0.315248   -0.014476  0.765244  1.000000
```