



به نام خدا
سیستم‌های توزیع شده
۲- ۱۳۹۹

تمرین چهارم
مدرس: صابر صالح

نکات مهم

لطفا ابتدا به نکات زیر توجه کنید:

- برای پیاده‌سازی این تمرین از Spark و زبان Python3 استفاده نمایید.
 - پاسخ تمرین را به صورت یک فایل zip با نام شماره دانشجویی خود در Quera آپلود کنید.
 - به شیوه‌ی اجرای برنامه دقت کنید.
 - مهلت ارسال تمرین تا پایان روز ۱۴۰۰/۳/۱۱ می‌باشد.
- موفق باشید.

مسئله رتبه‌بندی

شبکه‌ای از وب‌سایت‌ها را می‌توان به صورت گرافی جهت‌دار نمایش داد. هر راس در این گراف نشان‌دهنده یک صفحه از یک وب‌سایت و هر یال جهت‌دار نشان‌دهنده وجود پیوند از صفحه متناظر راس مبدا به صفحه متناظر راس مقصد خواهد بود. هدف ما یافتن یک متریک خوب برای اندازه‌گیری اهمیت هر راس در گراف است که با استفاده از آن بتوانیم رتبه‌بندی‌ای از وب‌سایت‌ها به دست آوریم. در مقالات آکادمیک، یکی از متریک‌هایی که برای محاسبه اهمیت یک مقاله به کار می‌برند، تعداد ارجاعاتی است که دریافت کرده‌است. ما نیز می‌توانیم از یک روش مشابه در رتبه‌بندی وب‌سایت‌ها استفاده کنیم و اهمیت یک وب‌سایت را با تعداد وب‌سایت‌هایی که به آن پیوند دارند، اندازه‌گیری کنیم. البته در حالت ایده‌آل ترجیح می‌دهیم اهمیتی که یک وب‌سایت از پیوند شدن توسط وب‌سایت دیگر دریافت می‌کند، با اهمیت وب‌سایت پیوند دهنده و کیفیت پیوند (بزرگی فونت یا محل قرارگیری) نیز ارتباط داشته‌باشد. برای سادگی فرض می‌کنیم کیفیت پیوند را می‌توانیم با یک عدد اعشاری مثبت نمایش دهیم و این عدد را وزن یال متناظر با پیوند در نظر می‌گیریم.

در ابتدا برای این که وب‌سایت‌ها نتوانند با پیوند دادن به خود، اهمیت خود را بالا ببرند تمام طوقه‌ها را حذف می‌کنیم.

حال می‌توانیم برای دقیق‌تر کردن شهود مسئله، از گشت تصادفی^۱ استفاده کنیم. کاربری را در نظر می‌گیریم که در حال مرور وب‌سایت‌ها است و با توجه به کیفیت پیوندها بر روی آن‌ها کلیک می‌کند. برای مثال اگر کاربر در یک وب‌سایت قرار دارد که سه پیوند به وب‌سایت‌های دیگر با کیفیت‌های 1.4، 0.7 و 0.4 دارد، احتمال کلیک کردن او بر روی پیوندها به ترتیب برابر 0.4، 0.2 و 0.4 است. با توجه

¹Random Walk

به اینکه طوقه‌ها را از گراف حذف می‌کنیم، اگر صفحه‌ای به خودش پیوند داده باشد احتمال کلیک کردن بر روی آن را صفر در نظر گرفته‌ایم. هدف ما بررسی رفتار این کاربر در زمانی است که بی‌نهایت گام برمی‌دارد. برای بیان بهتر شرایط از ماتریس انتقال کاربر استفاده می‌کنیم که در آن $w(e_{i,j})$ نشان‌دهنده وزن یال از راس i به راس j است:

$$H_{i,j} = \begin{cases} \frac{w(e_{i,j})}{\sum_{k=1}^{k=n} w(e_{i,k})} & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$$

حال $H_{i,j}^k$ نشان‌دهنده احتمال رفتن از راس i به راس j در دقیقه k گام در یک گشت تصادفی بر روی گراف است.

با داشتن این ماتریس انتقال و استفاده از قضایای موجود در زنجیره مارکوف می‌توانیم به توزیعی برسیم که نشان‌دهنده زمان نسبی است که به‌طور میانگین کاربر در هر وب‌سایت، در یک گشت نامتناهی می‌گذراند. اما در این روش مشکلاتی نیز ممکن است وجود داشته باشد. برای مثال که کاربر در یک وب‌سایت که به هیچ وب‌سایت دیگری پیوند ندارد، گیر می‌کند و همچنین هیچ وقت به یک وب‌سایت که از دیگر وب‌سایت‌ها پیوند ندارد وارد نمی‌شود. این دو مشکل در استفاده از توزیع به دست آمده برای رتبه‌بندی وب‌سایت‌ها اختلال ایجاد می‌کنند.

برای حل این مشکلات ابتدا از هرکدام از راس‌های بدون یال خروجی، به تمامی راس‌های دیگر یال با کیفیت برابر ایجاد می‌کنیم. حال زمانی که کاربر در وب‌سایتی باشد که به هیچ وب‌سایت دیگری پیوند ندارد، در گام بعدی به احتمال برابر می‌تواند به هر یک از وب‌سایت‌های دیگر برود. این شرایط را نیز می‌توانیم با ماتریس $S = H + A$ نمایش می‌دهیم که در آن A را به صورت زیر تعریف می‌کنیم:

$$A_{i,j} = \begin{cases} \frac{1}{n-1} & \text{if } i \neq j \text{ and } i \text{ has no outgoing edge} \\ 0 & \text{otherwise} \end{cases}$$

سپس به کاربر توانایی پرش احتمالی نیز اضافه می‌کنیم. برای شهود یافتن می‌توان اینگونه در نظر گرفت که در ابتدای هر گام کاربر با احتمال $1 - \alpha$ انتخاب می‌کند که از پیوندهای موجود در وب‌سایت فعلی استفاده کند یا به احتمال α به یک وب‌سایت دیگر برود. حالت دوم را در دنیای واقعی می‌توان وارد کردن آدرس وب‌سایت به صورت مستقیم توسط کاربر در نظر گرفت. این شرایط را نیز با ماتریس $G = (1 - \alpha)S + \alpha \frac{1}{n} ee^T$ مدل می‌کنیم که برداری n بعدی است که تمام خانه‌های آن 1 هستند و $0 < \alpha < 1$ است. واضح است که مجموع مقادیر موجود در هر سطر ماتریس G برابر 1 است.

در پیاده‌سازی بهتر است به جای استفاده از ماتریس با تمام خانه‌های برابر برای نمایش مقصد پرش کاربر $(\frac{1}{n} ee^T)$ ، از ماتریسی که بر اساس ترجیحات احتمالی کاربران برای وارد کردن آدرس مستقیم وب‌سایت ایجاد شده است، استفاده کرد. زیرا برای مثال احتمال وارد کردن آدرس وب‌سایت دانشگاه شریف با وارد کردن آدرس صفحه اصلی گوگل برای کاربر برابر نیست. ما در این تمرین برای حفظ سادگی، از این مورد صرف نظر می‌کنیم.

حال به حل کردن مسئله اصلی می‌پردازیم. ما به دنبال بردار سطری π از احتمالات می‌گردیم که:

$$\pi G = \pi$$

در این تمرین قصد حل این معادله به صورت حل تکراری^۲ و با استفاده از روش توانی^۳ را داریم.

^۲iterative

^۳power method

در ابتدا یک بردار توزیع اولیه یکنواخت

$$\pi^{(0)} = \left[\frac{1}{n} \quad \dots \quad \frac{1}{n} \right]$$

ایجاد می‌کنیم و سپس مسیر همگرا زیر را ادامه می‌دهیم:

$$\pi^{(t+1)} = \pi^{(t)} G$$

در این تمرین به اثبات همگرایی نمی‌پردازیم، اما این همگرایی به دلیل افزودن توانایی پرش و تغییر سطرهاى تمام صفر در ماتریس، به وجود آمده است.

$$\lim_{t \rightarrow \infty} \pi^{(t)} = \pi$$

این فرآیند را معمولاً به تعداد مشخصی بار تکرار می‌کنند اما در بعضی حالت‌ها، بردار π سریع‌تر از پایان تعداد تکرار مشخص شده الگوریتم، به دقت مورد نظر می‌رسد و الگوریتم را سریع‌تر متوقف می‌کنیم. در این تمرین زمانی می‌گوییم بردار همگرا شده است که نرم-۱ تفاضل بردار با بردار گام قبلی، کمتر از آستانه مشخص شده که با β نمایش می‌دهیم، باشد. یعنی:

$$\|\pi^{(t)} - \pi^{(t-1)}\|_1 < \beta$$

در این حالت بردار $\pi^{(t)}$ را به عنوان خروجی انتخاب می‌کنیم.

حتی با وجود رسیدن به راه‌حل مسئله، همچنان مشکل بسیار بزرگ بودن ماتریس G و سطر $\pi^{(k)}$ به اندازه‌ای که هیچ کدام از آن‌ها نمی‌توانند در حافظه اصلی یک سیستم قرار گیرند، وجود دارد. نکته‌ای که

در اینجا به ما کمک می‌کند این است که تعداد پیوندهای موجود در هر صفحه وب، نسبت به تعداد تمام وب‌سایت‌ها بسیار کم است و در نتیجه تعداد کمی از خانه‌های هر سطر H غیر صفر هستند. با توجه به اینکه روند ایجاد ماتریس G از روی ماتریس H را داریم، پس تنها با نگهداری شماره ستون خانه‌های غیر صفر در هر سطر و مقدار آن خانه‌ها، می‌توان سطر H را در حافظه اصلی قرار داد. همچنین برای نگهداری ماتریس G فضای ذخیره‌سازی بسیار بیشتری نسبت به نگهداری ماتریس H به شکلی که گفته‌شد، مورد نیاز است که این فضا در دسترس سیستم‌های ما نیست.

حال شما باید با فرض گفته‌شده محاسبات را با کمک پارادایم MapReduce به شکلی تغییر دهید که به صورت توزیع‌شده امکان‌پذیر شوند.

دقت کنید که $G = (1 - \alpha)(H + A) + \alpha \frac{1}{n} ee^T$ است و نیازی به تشکیل و نگهداری کل ماتریس A نیز ندارید و نباید این کار را انجام دهید. صرفاً می‌توان خانه مورد نظر از ماتریس A در هر لحظه را با توجه به اینکه راس متناظر با آن خانه، یال خروجی دارد یا خیر، به دست آورد. برای اطلاعاتی که در هر گام الگوریتم نیاز هستند و محاسبه آن‌ها زمان‌بر است (مانند بررسی اینکه یک راس یال خروجی دارد یا خیر) حتماً پیش‌پردازش و نگهداری کنید. دقت کنید که برخی از این اطلاعات نیز می‌توانند حجیم باشند و در حافظه اصلی یک سیستم قرار نگیرند و لازم باشد آن‌ها را به صورت توزیع‌شده (در RDD)، محاسبه و استفاده کنید.

راه‌حل ارائه‌شده به الگوریتم PageRank شباهت دارد و نحوه پیاده‌سازی توزیع‌شده این الگوریتم، می‌تواند به شما در حل این تمرین کمک کند. می‌توانید از این لینک و این لینک استفاده کنید.

شما در این تمرین باید راه‌حل ارائه شده را با توجه به محدودیت‌های گفته شده، در pySpark پیاده‌سازی کنید و برنامه شما در نهایت برای مقدار خواسته شده k ، بردار $\pi^{(k)}$ را و در صورتی که زودتر همگرایی اتفاق افتاد، بردار مورد نظر را خروجی دهد.

نحوه اجرا

برنامه شما با آرگومان‌های زیر اجرا می‌شود.

```
<file> <teleport-probability> <max-iterations> <convergence-threshold>
```

<file> آدرس فایل گراف ورودی به صورت کامل، <teleport-probability> احتمال پرش (α)، <max-iterations> حداکثر تعداد تکرار الگوریتم (k) است که در صورت نرسیدن به همگرایی تا قبل از آن باید $\pi^{(k)}$ را خروجی دهید و <convergence-threshold> آستانه همگرایی (β) است.

نمونه اجرا:

```
spark-submit ranking.py /home/sample-test.txt 0.15 10 0.01
```

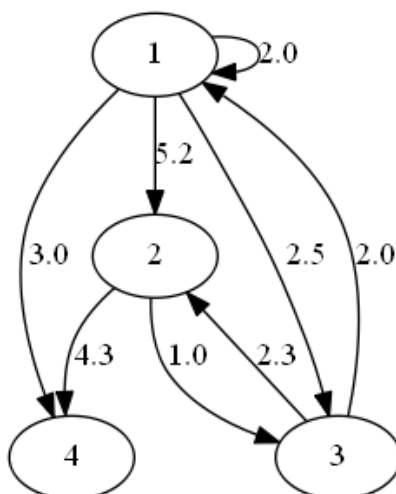
فایل ورودی با فرمت متن ذخیره شده‌است و در هر خط از آن مانند خط زیر، سه عدد که با فاصله از یکدیگر جدا شده‌اند قرار دارند که نشان‌دهنده یک یال از گراف هستند.

```
<source_id:int> <destination_id:int> <weight:float>
```

برای مثال یک فایل ورودی می‌تواند مانند زیر باشد.

```
1 2 5.2
1 4 3
1 3 2.5
1 1 2.0
2 3 1
2 4 4.3
3 1 2
3 2 2.3
```

شکل گراف متناظر ورودی نمونه، به صورت زیر خواهد بود.



شناسه راس‌ها معمولاً چکیده آدرس صفحات متناظرشان است. به همین دلیل فایل ورودی لزوماً ترتیب ندارد و شناسه راس‌ها نیز لزوماً از اعداد 1 تا n نیست. با توجه به این موارد، فرض می‌کنیم راسی وجود ندارد که هم یال خروجی و هم یال ورودی نداشته باشد.

برای ذخیره خروجی، آن را با ساختار زیر و با دستور `saveAsTextFile('./result')` ذخیره کنید.

```
(<vector_id:int>, <value:float>)
```

شما باید بتوانید در فایل‌های `part-*****` ایجاد شده در پوشه `result`، خروجی خود را به صورت متنی مشاهده کنید. برای مثال خروجی مطابق ورودی مثال بالا و دستور نمونه اجرا به شکل زیر می‌شود. تطابق جواب شما با مقادیر صحیح تا ۱۲ رقم اعشار کافی است. ترتیبی برای خروجی لازم نیست.

```
(1, 0.2023985764262125)
(2, 0.2976103516659765)
(3, 0.2084394700528872)
(4, 0.2915516018549236)
```

دقت کنید که در این مثال، الگوریتم بعد از ۵ گام به محدوده همگرایی مشخص شده رسیده و متوقف

شده است.

نکات

◀ برای نگه‌داری داده‌ها از داده‌ساختار RDD استفاده کنید. استفاده از `dataFrame` و `graphX` مجاز نیست. همچنین تنها مجاز به استفاده از کتابخانه‌های پیشفرض `python3` مانند `sys` هستید.

◀ داده‌های اولیه را حداقل به ۸ پارتیشن تقسیم کنید. سعی کنید این پارتیشن بندی را به صورتی انجام دهید که با کاهش میزان داده‌ای که قرار است در مراحل بعدی در فرایند `shuffle` جابه‌جا شوند، باعث کاهش زمان اجرای برنامه در مراحل بعدی شود. با توجه به اینکه تعداد مناسب پارتیشن‌ها برای بازده بهتر به شرایط کلاستر و داده وابسته است، با در نظر گرفتن محیط تست پیشنهاد می‌کنیم سعی کنید هر مجموعه داده را در طول برنامه نیز در ۸ پارتیشن نگه دارید.

◀ برنامه شما باید از نظر زمان اجرا، عملکرد خوبی داشته باشد و یکی از معیارهای اصلی نمره‌دهی، زمان اجرای برنامه است. یکی از راه‌های کاهش زمان اجرا استفاده از `caching` است. یکی از توابعی که می‌تواند در این زمینه به شما کمک کند، تابع `persist` است که می‌توانید توضیحات بیشتر در رابطه با آن و انواع سطح استفاده از آن را در این لینک مطالعه کنید. شما باید از این تابع یا توابع با کاربرد مشابه در برنامه خود استفاده کنید.

گزارش

در کنار فایل نهایی، باید گزارشی با فرمت PDF نیز با محتوای موارد زیر تحویل دهید.

◀ نحوه دقیق عملکرد برنامه خود را به صورت کلی و خط به خط شرح دهید.

◀ اگر از ایده خاصی برای کاهش میزان محاسبات استفاده کردید، شرح دهید.

◀ اگر ایده دیگری برای کاهش میزان محاسبات داشتید و نتوانستید پیاده‌سازی کنید، شرح دهید.