بازیابی هوشمند اطلاعات

تمرین سوم

نام و نام خانوادگی : حسین سیفی

شماره دانشجویی : ۸۱۰۱۰۰۳۸۶

آذر ۱۴۰۰

بخش اول – MapReduce

هدف بخش اول تمرین پیدا کردن ۱۰۰ زوج کلمه با بیشترین فراوانی نسبی با مدلی مبتنی بر MapReduce و به کمک زبان پایتون و کتابخانه Mrjob می‌باشد.

در ابتدا تلاش کردیم که فراوانی همزمان زوج کلمات را به دست آوریم اما متاسفانه حافظه ماشین بنده(8 GB) اجازه این محاسبه را نداد و قادر به ادامه دادن انجام بخش اول تمرین نیستم!

بخش دوم – Page Rank

در بخش دوم تمرین به بررسی الگوریتم Page-rank با کمک زبان پایتون کتابخانه Networkx روی گراف‌های جهت‌دار spider10k و spider800k می‌پردازیم.

سوال ۱

هدف این سوال ، پیدا کردن تمامی بن‌بست‌ها است. طبق تعریف گرهی بن‌بست است که یال خروجی نداشته باشد یا کل یال‌های خروجی آن به گره‌های بن‌بست ختم شوند. بنابراین می‌توان دو رویکرد را برای پیدا کردن گره‌هایی با این ویژگی دنبال کرد.

۱. در این روش ابتدا کل گره‌هایی که یال خروجی ندارند را مشخص می‌کنیم و در لیستی ذخیره می‌کنیم. سپس به ازای هر گره مشخص شده(n_1) ، گره‌هایی که یکی از یال‌های خروجی آن‌ها گره مورد نظر است(n_1) را بررسی می‌کنیم. اگر کل یال‌های این گره جدید ، به گره‌های بن‌بست ختم شود ، این گره نیز بن‌بست است. عمل ذکر شده به ازای تمام گره‌های موجود در لیست بن‌بست(به همراه گره‌هایی که پس از مقدار دهی اولیه اضافه می‌شوند) انجام می‌شود. این روش برای گراف جهت‌دار spider10k با زمان معقولی جواب را مشخص می‌کرد اما برای گراف جهت‌دار spider800k در مدت زمان طولانی هم نتیجه‌ای به ما بازنمی‌گرداند.

۲. در این روش ابتدا گره‌هایی با درجه خروجی برابر صفر را به لیست بن‌بست اضافه می‌کنیم و سپس این گره‌ها را از گراف حذف می‌کنیم. در نتیجه گره‌هایی که تنها به گره‌های بن‌بستی که تا کنون شناخته شده‌اند یال خروجی دارند ، اکنون درجه خروجی برابر صفر را به دست می‌آورند و در تکرار مجددِ حذف گره‌هایی با درجه‌ای برابر صفر این گره‌ها نیز حذف می‌شوند و به لیست بن بست اضافه می‌شوند. عمل فوق رو تا جایی تکرار می‌کنیم که گرهی با خروجی برابر صفر در گراف باقی نباشد.

روش ۲ در زمانی مطلوب(کمتر از ۱ دقیقه) گره‌های بن‌بست را در گراف جهت‌دار spider800k مشخص روش ۲ برای گراف جهت‌دار spider10k تعداد ۱۵۴۴ گره بن‌بست و برای گراف جهت‌دار spider800k تعداد ۱۸۱۰۵۷ گره بن‌بست را پیدا می‌کند. لیست این گره‌ها و کد روش ۲ به ضمیمه ارسال می‌شوند.

سوال ۲

در پاسخ سوال ۲ ، الگوریتم Page-rank را بر روی داده‌های فایل‌های spider10k و spider800k اجرا می‌کنیم. این امر به کمک کتابخانه Networkx و تابع زیر انجام می‌گیرد.

```
pr_10k = nx.pagerank(graph_10k, max_iter=10)
```

تابع pagerank از کتابخانه Networkx را برای اعمال این الگوریتم روی گراف ایجاد شده از داده‌های فایل‌های spider بدین صورت اعمال می‌کنیم که گراف را به عنوان پارامتر اول به تابع می‌دهیم و پارامتر Max_iter تعداد تکرارهای محاسبه‌ی بردار P را قبل از همگرا شدن(Converge) به مقدار نهایی نشان می‌دهد. اگرچه مقدار Page-rank این گراف‌ها با ۱۰ تکرار همگرا می‌شود ولی در صورتی که مقدار بردار p با این تعداد تکرار همگرا نشود ، محاسبه متوقف شده و نتایج محاسبه شده با این تکرار را بازمی‌گرداند.

نتایج اجرای این تابع روی گراف‌های spider10k و spider800k نیز در فایل‌هایی با فرمت csv ذخیره شده و به ضمیمه ارسال می‌شود.

سوال ۳

دراین سوال گره‌های بن‌بست به دست آمده برای هر گراف در سوال ۱ از بخش دوم تمرین را از گراف‌های spider10k و spider800k حذف می‌کنیم سپس محاسباتی مشابه با سوال ۲ از بخش دوم تمرین را بر روی گراف‌های جدید اعمال می‌کنیم. تفاوتی که در این سوال نسبت به سوال قبل دیده می‌شود این است که مقادیر بردار p در این سوال با تعداد تکرار ۱۰ بار برای هیچ یک از دو گراف همگرا نمی‌شود. مشخصا با افزایش تعداد تکرار محاسبه فرمول Page-rank به کمتر از ۱۰۰ تکرار ، زمان مورد نیاز برای اعمال الگوریتم نسبت به حالتی که گره‌های بن‌بست در گراف وجود دارند ، بیشتر می‌شود. علت می‌تواند این باشد که بین گره‌های باقی‌مانده دور(cycle) وجود دارد و به یکدیگر اشاره می‌کنند. این اشارات باعث می‌شود می‌شود که این گره‌ها امتیازات یکدیگر را بسیار بالا ببرند. از طرفی دیگر در صورت وجود دوری بدون خروج بین گره‌ها باعث می‌شود امتیازات آن‌ها بسیار بالا برود و تنها راه خروج از این دور ، پرش تصادفی به گرهی دیگر است. با توجه به اینکه در این گراف‌ها گره بن‌بستی نداریم ، امکان اینکه با پرش تصادفی در داخل دور بدون خروجی دیگری گرفتار شویم. توضیحات فوق می‌تواند دلیل هر دو اتفاق رخ داده یعنی افزایش زمان محاسبات و افزایش عجیب و بیش از اندازه مقدار Page-rank را شرح دهد.