به نام خدا

تمرین سوم بازیابی اطلاعات

محمد ناصری

۸۱۰۱۰۰۴۸۶

آذر ۱۴۰۰

بخش اول

در این بخش قصددد داریم با اسددتفاده از مدل مبتنی بر map/reduceاقدام به پیدا کردن لیسددتی از 100 جفت کلمه با بالاترین میزان فراوانی نسبی در دادگان موجود در فایل wikitext.txtنماییم

سوال اول

با استفاده از رویکرد ،Stripesروش map/reduceرا برای محاسبهی یک لیست نزولی از 100 جفت کلمه با بالاترین میزان فراوانی نسبی پیاده سازی کنید و خروجی را در فایل stripes.csvذخیره نمایید. نحوهی عملکرد این رویکرد را در فایل گزارش خود شرح دهید.

پاسخ:

هر Mapper

- برای هر کلمه خاص، اطلاعات کلمات همزمان دیده شده را در یک آرایه انجمنی ذخیره می کند
- جفت های کلید-مقدار را با کلمات به عنوان کلید و آرایه های ارتباطی مربوطه به عنوان مقادیر منتشر می کند.

هر Reducer

- تمام شمارش ها را در آرایه های انجمنی جمع می کند
- چارچوب اجرای MapReduce تضمین می کند که تمام آرایه های انجمنی با یک کلید در یک کاهنده گرد هم آمده اند.

- Example:

```
(a, b) -> 1
(a, c) -> 2
(a, d) -> 5
(a, e) -> 3
(a, f) -> 2
```

```
a -> {b: 1, c: 2, d: 5, e: 3, f: 2}
```

- Each mapper emits
  - `a -> {b: count(b), c: count(c), d: count(d) …}`
- Reducers perform element-wise sum of associative arrays

```
    a -> {b: 1,      , d: 5, e: 3      }
+   a -> {b: 1, c: 2, d: 2,       f: 2}
-------------------------------------------
    a -> {b: 2, c: 2, d: 7, e: 3, f: 2}
```

Stripes

```
1: class Mapper
2:     method Map(docid a, doc d)
3:         for all term w ∈ doc d do
4:             H ← new AssociativeArray
5:             for all term u ∈ Neighbors(w) do
6:                 H{u} ← H{u} + 1                ▷ Tally words co-occurring with w
7:             Emit(Term w, Stripe H)

1: class Reducer
2:     method Reduce(term w, stripes [H_1, H_2, H_3, ...])
3:         H_f ← new AssociativeArray
4:         for all stripe H ∈ stripes [H_1, H_2, H_3, ...] do
5:             Sum(H_f, H)                        ▷ Element-wise sum
6:         Emit(term w, stripe H_f)
```

با توجه به اینکه مرتبه این الگوریتم از $O(n^2)$ میباشد، بنده علی رغم تلاش زیاد و آزمایش روش های متفاوت خواندن ورودی و اجرای الگوریتم موفق به اجرای آن بر روی کل داده مورد نظر نشدم و هر بار با Memory Error مواجه شدم. فلذا آزمایش را بر روی ۲۰ خط اول داده بررسی و انجام داده‌ام.

سوال دوم

با استفاده از رویکرد ،Pairsروش map/reduceرا برای محاسبهی یک لیست نزولی از 100 جفت کلمه با بالاترین میزان فراوانی نسبی پیاده سازی کنید و خروجی را در فایل pairs.csvذخیره نمایید. نحوه عملکرد این رویکرد را در فایل گزارش خود شرح دهید

پاسخ:

هر Mapper :

- جفت های کلید-مقدار را با هر جفت کلمه همزمان مشاهده شده و عدد صحیح 1 منتشر می کند.

هر Reducer

- تمام مقادیر مرتبط با یک جفت کلمه مشابه را خلاصه می کند
- چارچوب اجرای MapReduce تضمین می کند که تمام مقادیر مرتبط با یک کلید در یک کاهنده گرد هم می آیند

Pairs

```
1: class Mapper
2:     method Map(docid a, doc d)
3:         for all term w ∈ doc d do
4:             for all term u ∈ Neighbors(w) do
5:                 Emit(pair (w, u), count 1)        ▷ Emit count for each co-occurrence

1: class Reducer
2:     method Reduce(pair p, counts [c1, c2, ...])
3:         s ← 0
4:         for all count c ∈ counts [c1, c2, ...] do
5:             s ← s + c                              ▷ Sum co-occurrence counts
6:         Emit(pair p, count s)
```

سوال سوم

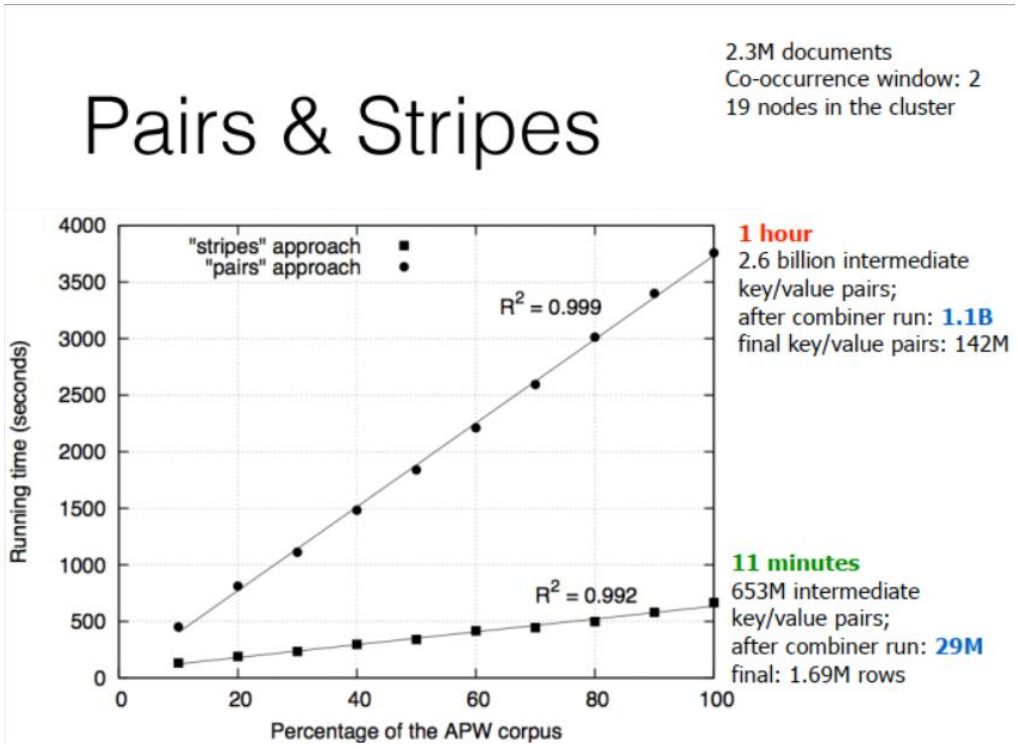
عملکرد دو رویکرد را از نظر حافظه مصرفی و زمان اجرا مقایسه کنید و جمع بندی خود را از این دو رویکرد با ذکر دلیل بیان کنید

پاسخ:

مقایسه دو روش Pairs, Stripes

- Pairs
  - مزیت: سادگی الگوریتم برای هم و پیاده سازی
  - عیب: تولید تعداد بسیار زیادی جفت کلید-مقدار
- Stripes
  - مزیت: تولید تعداد کمتر جفت کلید-مقدار
  - عیب: اندازه حافظه آرایه‌ها ممکن است بسیار زیاد باشد

طبق آزمایش‌های انجام شده زمان انجام عملیات Pairs برای داده تست ۲۰ سندی انتخاب شده برابر مقدار تقریبی ۴۵۵ ثانیه است در حالیکه این مقدار برای Stripes برابر مقدار تقریبی ۳۷ ثانیه است. این نتایج مزیت و معایب ذکر شده در بالا را تایید میکند. همچنین این امر بطور کلی توسط آزمایشگاه‌ها بررسی شده که در نمودار زیر قابل مشاهده است:



بخش دوم

پیدا کردن همه بنبست ها: یک نود را بن بست می گوییم اگر یا هیچ یال خروجی نداشته باشد و یا همه ی یالهای خروجی آن به یک نود بن بست متصل باشند

سوال اول

الگوی استفاده شده برای پیدا کردن بن بست ها در این تمرین بدین صورت است:

1 - پیدا کردن لیست Adjacency برای هر گره بدین صورت که هر A_i نشان دهنده گره هایی است که به i وارد شده اند.

2 - پیدا کردن لیست Degree برای هر گره بدین صورت که هر D_i نشان دهنده درجه خارج شده از گره است.

3 - یافتن گره های با درجه خرجی صفر و اضافه کردن آنها به لیست گره های End

4 - با شروع از پایین به بالا، به ازای هریک از گره های End و طبق لیست Adjacency یالهای متصل به این گره ها را قطع میکنیم. بعبارتی از گره های متصل به آنها یک درجه در لیست درجات کم میکنیم. حال اگر گره‌ای دارای درجه صفر شد یعنی فقط به گره‌های End متصل بوده پس آنرا به لیست جدید اضافه میکنیم و این کار را در هر مرحله با لیست گره‌های End جدید ادامه میدهیم.

سوال دوم

انجام الگوریتم PageRank که نتیجه های آن در فایل مربوطه آورده شده(با نام PageRank)

سوال سوم

بن بست یک صفحه وب بدون لینک است. وجود بن بست باعث می شود که رتبه صفحه برخی یا همه صفحات در محاسبات تکراری به 0 برسد، از جمله صفحاتی که بن بست نیستند.

بن‌بست‌ها را می‌توان قبل از انجام محاسبه رتبه صفحه با حذف بازگشتی گره‌ها بدون یال از بین برد. توجه داشته باشید که حذف کردن یک گره می تواند باعث شود که گره دیگری که فقط به آن متصل است تبدیل به بن بست شود، بنابراین فرآیند باید بازگشتی باشد.

رویکرد برای مقابله با بن بست ها:

می‌توانیم بن‌بست‌ها را از گراف حذف کنیم، و همچنین یالهای ورودی آن‌ها را حذف کنیم. انجام این کار ممکن است بن بست های بیشتری ایجاد کند که همچنین باید به صورت بازگشتی حذف شوند. با این حال، در نهایت، ما با یک مؤلفه قوی متصل که هیچ یک از گره‌های آن بن‌بست نیستند، پایان می‌دهیم. حذف بازگشتی بن‌بست‌ها، بخش‌هایی از اجزای بیرونی، حلقه ها و... را حذف می‌کند، اما درون مولفه و همچنین بخش‌هایی از اجزای کوچک جدا شده را نیز حذف می‌کند.

با انجام این عمل امتیازهای دقیق تری به صفحات انتساب پیدا بکند. همچنین زمان اجرا بدلیل بررسی گره های کمتر و همینطور نبودن برخی حلقه‌های dead end کمتر از حالت قبل است.