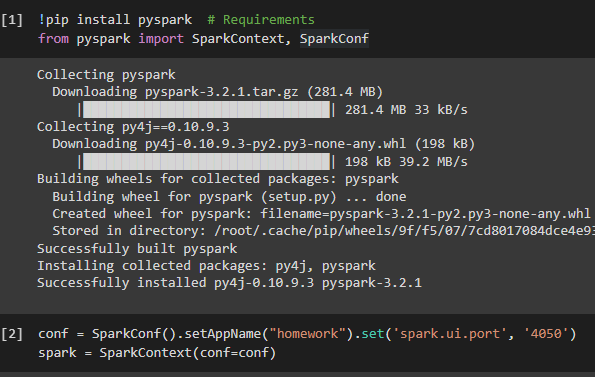
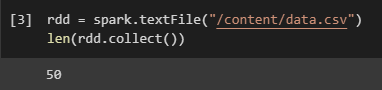
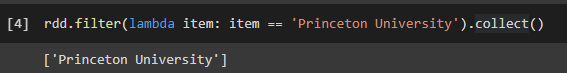
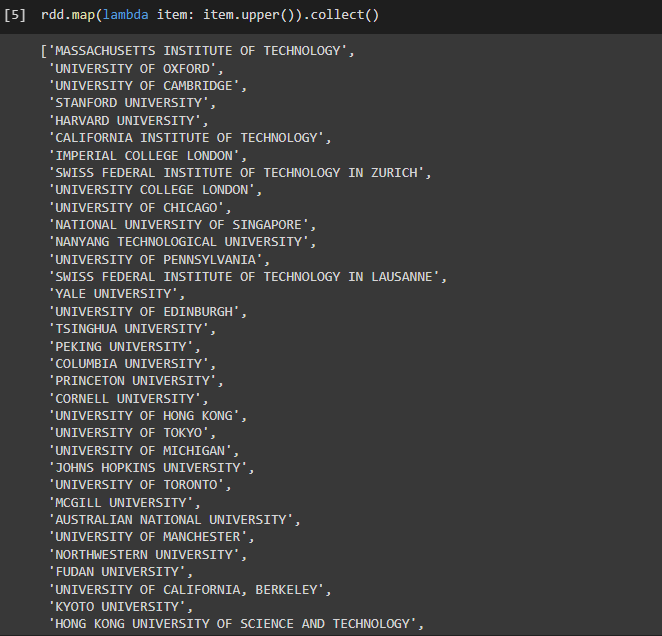
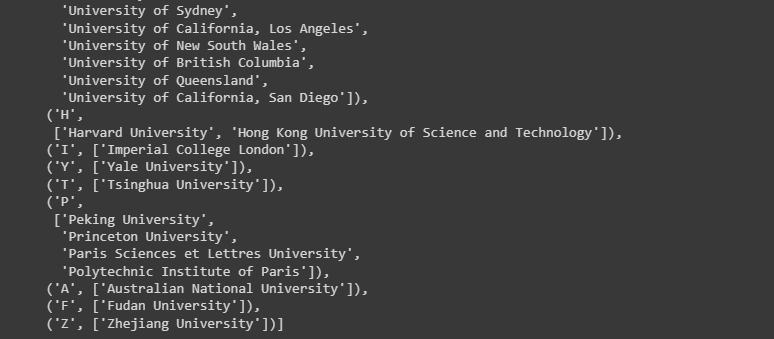
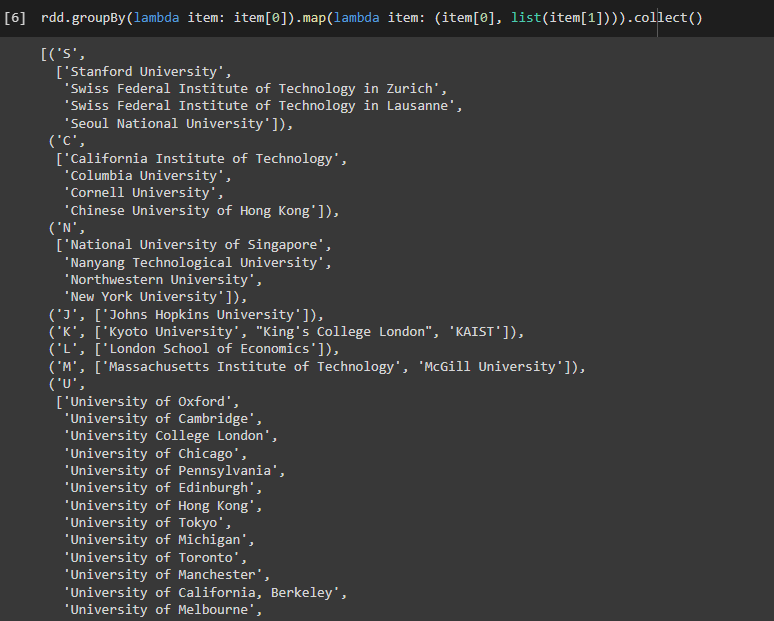
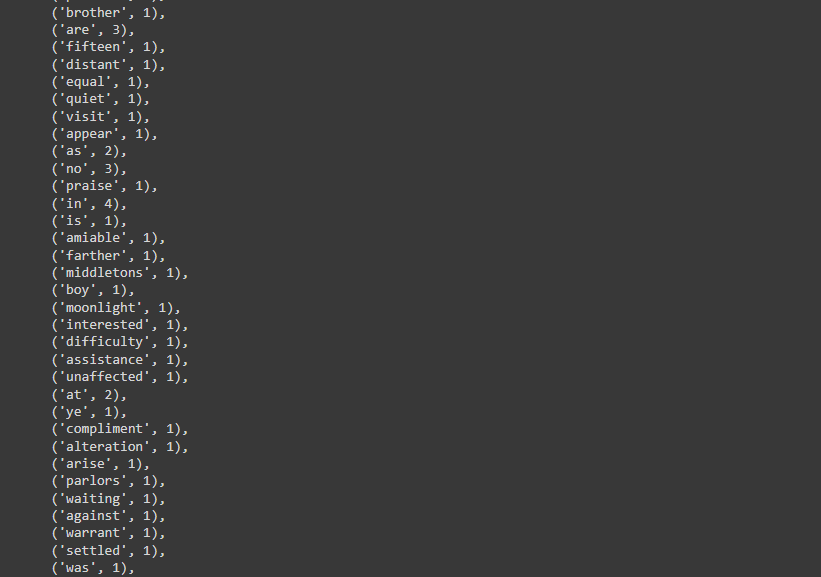
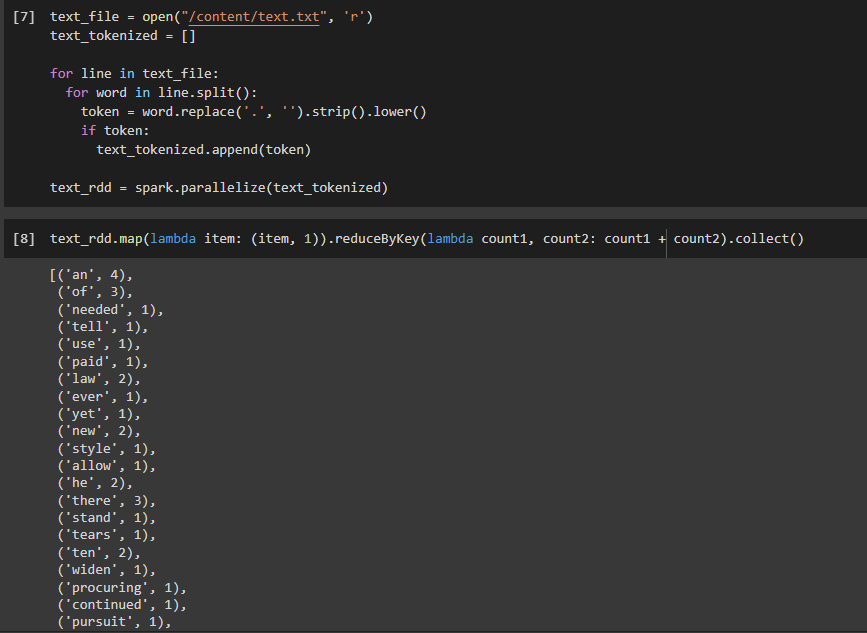
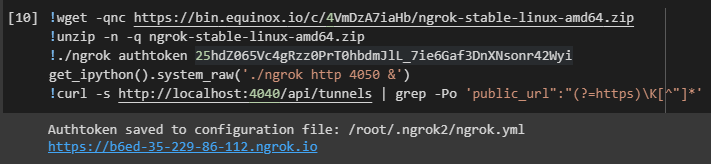
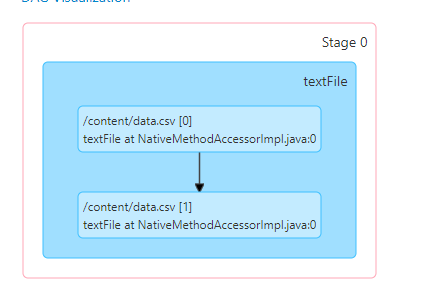
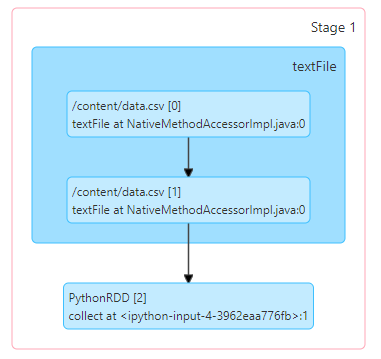
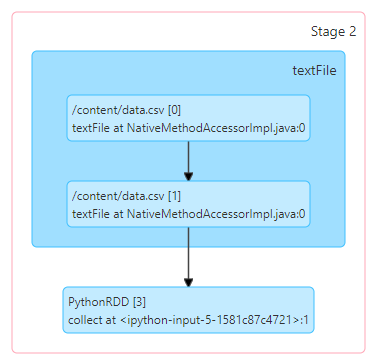
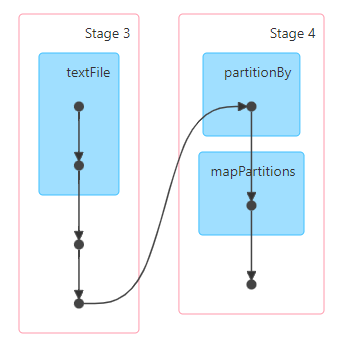
|  |
| --- |
| به‌نام خداوند بخشنده و مهربان  **تمرین اول:** مقدمه‌ای بر اسپارک **استاد:** محمد‌علی نعمت‌بخش  **درس:** پایگاه‌داده پیشرفته دستیاران: فاطمه‌ ابراهیمی، پریسا لطیفی، امیر سرتیپی  **نام و نام‌خانوادگی:** سید عمید اسدالهی مجد  **آدرس گیت:**<https://github.com/amidmajd/HW-1-spark-intro> |

1. Lazy Evaluation در اسپارک به این معناست که عمل اجرا شروع نمی‌شود تا هنگامی که یک Action فراخوانی شود. بنابراین هر تعداد Transformation می‌توان روی داده داشت اما اجرای آن‌ها به هنگام فراخوانی یک Action مانند collect خواهد بود. به طور مثال فرض کنیم یک RDD شامل نام محصول و قیمت آن‌ها داریم و می‌خواهیم ۱۰ محصول که از همه‌ی محصولات دیگر گران‌تر هستند پیدا کنیم. برای انجام اینکار ابتدا با فراخوانی sort به صورت نزولی با کلید قیمت روی RDD یک Transformation انجام می‌دهیم، سپس با فراخوانی عمل take(10) ده محصول گران‌تر از بقیه محصولات را استخراج می‌کنیم. در این مثال تا قبل از فراخوانی عمل take هیچ اجرایی صورت نگرفته است.
2. در حالت Narrow Transformation هر Partition که شامل بخشی از داده‌ها‌ است فقط در تولید نتیجه یک Partition خروجی تاثیر می‌گذارد، اما در حالت Wide Transformation ممکن است هر Partition در تولید نتیجه تعداد مختلفی از Partition های خروجی تاثیر بگزارد. تفاوت اصلی این دو حالت نیاز به جابه‌جایی داده‌ها میان Partition ها برای حالت Wide است که هزینه بیشتری دارد. در واقع ورودی‌های مورد نیاز برای تولید هر Partition خروجی در حالت Wide ممکن است در Partition های متفاوت باشند اما در حالت Narrow ورودی‌های مورد نیاز فقط در یک Partition هستند.

به طور مثال برای انجام groupByKey ممکن است داده‌ها به ازای کلید مورد نظر مرتب نباشند و داده‌های مربوط به هر کلید در Partition های مختلف قرار داشته باشند، بنابراین ابتدا باید داده‌های مربوط به هر کلید از Partition های مختلف به یک Partition منتقل شوند و سپس اجرا صورت گیرد.

* 1. **Narrow Transformation:** map و filter و pipe و flatMap
  2. **Wide Transformation:** groupByKey و reduceByKey و aggregateByKey و sortByKey
  3. **چهار Action:** reduce و collect و foreach و saveAsTextFile

1. (لیست شامل نام ۵۰ دانشگاه برتر دنیا که در فایل data.csv قرار دارد، و متن برای بخش map-reduce در فایل text.txt قرار دارد که هردو فایل قبل از اجرا در گوگل کولب بارگزاری شدند)

* نوت‌بوکی بر روی گوگل کولب ایجاد کرده و این کتابخانه را فراخوانی کنید.  
  
* لیست خود را به RDD تبدیل کنید. 
* با کمک دستور filter بر روی RDD، از آن برای بازیابی عنصر 20ام لیست خود استفاده کنید. (برابر با عنصر 20ام باشد)
* با کمک map تمامی عناصر لیست خود را به حروف بزرگ تبدیل و آن را بازیابی کنید. 
* با کمک دستور groupby و map، لیست خود را بر اساس اولین کاراکتر آن دسته بندی کنید.  
  
* عملیات map و reduce را بر روی یک متن نسبتا بلند پس از تبدیل توکن‌های آن به rdd، انجام دهید.   
    
  
* چه تفاوتی بین Actionهای take و collect وجود دارد؟
  + عمل take(n) تعداد n عنصر اول از مجموعه داده‌ها (dataset) را در قالب یک آرایه برمی‌گرداند اما عمل collect تمام عناصر موجود در مجموعه داده‌ها (dataset) را برمی‌گرداند که بهتر است در مواقعی که داده‌ها تعداد کمی دارند (مثلا پس از filter) استفاده شود.
* در صورتی که بتوانید توالی انجام هریک از عملیات‌ها در اسپارک که برای هر دستور انجام می‌دهد را برای هریک از دستورات بالا نمایش دهید و باتوجه به مفاهیم سوالات قبل آن را تصویر سازی کنید، نمره اضافه‌ای دریافت خواهید کرد. (به کمک ngrok و UI Spark)
  + پس از اجرای دستورات زیر یک آدرس نشان داده می‌شود که پنل کاربری اسپارک است.  
    - تبدیل لیست به RDD و فراخوانی collect برای محاسبه تعداد داده‌ها (len) 
    - استخراج عنصر بیستم لیست از RDD و بازیابی با collect 
    - تبدیل تمام عناصر لیست به حروف بزرگ و بازیابی با collect 
    - گروه‌بندی لیست بر اساس حروف اول عناصر آن و بازیابی با collect
    - عمل map و reduce روی یک متن و بازیابی با collect 