به‌نام خداوند بخشنده و مهربان

تمرین اول: مقدمه‌ای بر اسپارک
درس: پایگاه‌داده پیشرفته

استاد: محمدعلی نعمت‌بخش
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 حلت 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.

3.

a. **Narrow Transformation**: map و filter و pipe و flatMap

b. **Wide Transformation**: groupByKey و reduceByKey و aggregateByKey و sortByKey

c. **چهار Action**: reduce و collect و foreach و saveAsTextFile

4. (لیســت شــامل نام ۵۰ دانشــگاه 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.

```
[1]  !pip install pyspark  # Requirements
     from pyspark import SparkContext, SparkConf

     Collecting pyspark
       Downloading pyspark-3.2.1.tar.gz (281.4 MB)
          |████████████████████████████████| 281.4 MB 33 kB/s
     Collecting py4j==0.10.9.3
       Downloading py4j-0.10.9.3-py2.py3-none-any.whl (198 kB)
          |████████████████████████████████| 198 kB 39.2 MB/s
     Building wheels for collected packages: pyspark
       Building wheel for pyspark (setup.py) ... done
       Created wheel for pyspark: filename=pyspark-3.2.1-py2.py3-none-any.whl
       Stored in directory: /root/.cache/pip/wheels/9f/f5/07/7cd8017084dce4e93
     Successfully built pyspark
     Installing collected packages: py4j, pyspark
     Successfully installed py4j-0.10.9.3 pyspark-3.2.1

[2]  conf = SparkConf().setAppName("homework").set('spark.ui.port', '4050')
     spark = SparkContext(conf=conf)
```

- لیست خود را به RDD تبدیل کنید.

```
[3]  rdd = spark.textFile("/content/data.csv")
     len(rdd.collect())

     50
```

- با کمک دستور filter بر روی RDD، از آن برای بازیابی عنصر 20ام لیست خود استفاده کنید. (برابر با عنصر 20ام باشد)

```
[4]  rdd.filter(lambda item: item == 'Princeton University').collect()

     ['Princeton University']
```

- با کمک map تمامی عناصر لیست خود را به حروف بزرگ تبدیل و آن را بازیابی کنید.

```
[5]  rdd.map(lambda item: item.upper()).collect()

     ['MASSACHUSETTS INSTITUTE OF TECHNOLOGY',
      'UNIVERSITY OF OXFORD',
      'UNIVERSITY OF CAMBRIDGE',
      'STANFORD UNIVERSITY',
      'HARVARD UNIVERSITY',
      'CALIFORNIA INSTITUTE OF TECHNOLOGY',
      'IMPERIAL COLLEGE LONDON',
      'SWISS FEDERAL INSTITUTE OF TECHNOLOGY IN ZURICH',
      'UNIVERSITY COLLEGE LONDON',
      'UNIVERSITY OF CHICAGO',
      'NATIONAL UNIVERSITY OF SINGAPORE',
      'NANYANG TECHNOLOGICAL UNIVERSITY',
      'UNIVERSITY OF PENNSYLVANIA',
      'SWISS FEDERAL INSTITUTE OF TECHNOLOGY IN LAUSANNE',
      'YALE UNIVERSITY',
      'UNIVERSITY OF EDINBURGH',
      'TSINGHUA UNIVERSITY',
      'PEKING UNIVERSITY',
      'COLUMBIA UNIVERSITY',
      'PRINCETON UNIVERSITY',
      'CORNELL UNIVERSITY',
      'UNIVERSITY OF HONG KONG',
      'UNIVERSITY OF TOKYO',
      'UNIVERSITY OF MICHIGAN',
      'JOHNS HOPKINS UNIVERSITY',
      'UNIVERSITY OF TORONTO',
      'MCGILL UNIVERSITY',
      'AUSTRALIAN NATIONAL UNIVERSITY',
      'UNIVERSITY OF MANCHESTER',
      'NORTHWESTERN UNIVERSITY',
      'FUDAN UNIVERSITY',
      'UNIVERSITY OF CALIFORNIA, BERKELEY',
      'KYOTO UNIVERSITY',
      'HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY',
      "KING'S COLLEGE LONDON",
      'SEOUL NATIONAL UNIVERSITY',
      'UNIVERSITY OF MELBOURNE',
      'UNIVERSITY OF SYDNEY',
      'CHINESE UNIVERSITY OF HONG KONG',
      'UNIVERSITY OF CALIFORNIA, LOS ANGELES',
      'KAIST',
      'NEW YORK UNIVERSITY',
      'UNIVERSITY OF NEW SOUTH WALES',
      'PARIS SCIENCES ET LETTRES UNIVERSITY',
      'ZHEJIANG UNIVERSITY',
      'UNIVERSITY OF BRITISH COLUMBIA',
      'UNIVERSITY OF QUEENSLAND',
      'UNIVERSITY OF CALIFORNIA, SAN DIEGO',
      'POLYTECHNIC INSTITUTE OF PARIS',
      'LONDON SCHOOL OF ECONOMICS']
```

- با کمک دســـتور groupby و map، لیســـت خود را بر اســـاس اولین کاراکتر آن دســـته بندی کنید.

```
[6]  rdd.groupBy(lambda item: item[0]).map(lambda item: (item[0], list(item[1]))).collect()
```

```
[('S',
  ['Stanford University',
   'Swiss Federal Institute of Technology in Zurich',
   'Swiss Federal Institute of Technology in Lausanne',
   'Seoul National University']),
 ('C',
  ['California Institute of Technology',
   'Columbia University',
   'Cornell University',
   'Chinese University of Hong Kong']),
 ('N',
  ['National University of Singapore',
   'Nanyang Technological University',
   'Northwestern University',
   'New York University']),
 ('J', ['Johns Hopkins University']),
 ('K', ['Kyoto University', "King's College London", 'KAIST']),
 ('L', ['London School of Economics']),
 ('M', ['Massachusetts Institute of Technology', 'McGill University']),
 ('U',
  ['University of Oxford',
   'University of Cambridge',
   'University College London',
   'University of Chicago',
   'University of Pennsylvania',
   'University of Edinburgh',
   'University of Hong Kong',
   'University of Tokyo',
   'University of Michigan',
   'University of Toronto',
   'University of Manchester',
   'University of California, Berkeley',
   'University of Melbourne',
   'University of Sydney',
   'University of California, Los Angeles',
   'University of New South Wales',
   'University of British Columbia',
   'University of Queensland',
   'University of California, San Diego']),
 ('H',
  ['Harvard University', 'Hong Kong University of Science and Technology']),
 ('I', ['Imperial College London']),
 ('Y', ['Yale University']),
 ('T', ['Tsinghua University']),
 ('P',
  ['Peking University',
   'Princeton University',
   'Paris Sciences et Lettres University',
   'Polytechnic Institute of Paris']),
 ('A', ['Australian National University']),
 ('F', ['Fudan University']),
 ('Z', ['Zhejiang University'])]
```

- عملیات map و reduce را بر روی یک متن نسبتا بلند پس از تبدیل توکن‌های آن به rdd، انجام دهید.

```
[7]  text_file = open("/content/text.txt", 'r')
     text_tokenized = []

     for line in text_file:
       for word in line.split():
         token = word.replace('.', '').strip().lower()
         if token:
           text_tokenized.append(token)

     text_rdd = spark.parallelize(text_tokenized)
```

```
[8]  text_rdd.map(lambda item: (item, 1)).reduceByKey(lambda count1, count2: count1 + count2).collect()
```

```
[('an', 4),
 ('of', 3),
 ('needed', 1),
 ('tell', 1),
 ('use', 1),
 ('paid', 1),
 ('law', 2),
 ('ever', 1),
 ('yet', 1),
 ('new', 2),
 ('style', 1),
 ('allow', 1),
 ('he', 2),
 ('there', 3),
 ('stand', 1),
 ('tears', 1),
 ('ten', 2),
 ('widen', 1),
 ('procuring', 1),
 ('continued', 1),
 ('pursuit', 1),
 ('brother', 1),
 ('are', 3),
 ('fifteen', 1),
 ('distant', 1),
 ('equal', 1),
 ('quiet', 1),
 ('visit', 1),
 ('appear', 1),
 ('as', 2),
 ('no', 3),
 ('praise', 1),
 ('in', 4),
 ('is', 1),
 ('amiable', 1),
 ('farther', 1),
 ('middletons', 1),
 ('boy', 1),
 ('moonlight', 1),
 ('interested', 1),
 ('difficulty', 1),
 ('assistance', 1),
 ('unaffected', 1),
 ('at', 2),
 ('ye', 1),
 ('compliment', 1),
 ('alteration', 1),
 ('arise', 1),
 ('parlors', 1),
 ('waiting', 1),
 ('against', 1),
 ('warrant', 1),
 ('settled', 1),
 ('was', 1),
```

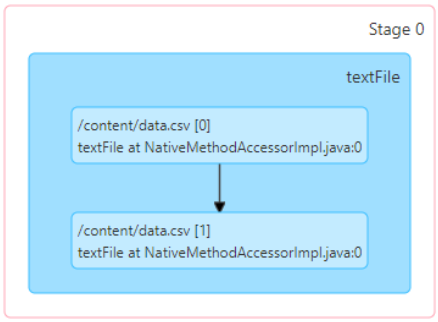
- چه تفاوتی بین Actionهای take و collect وجود دارد؟
  - عمل take(n) تعداد n عنصر اول از مجموعه داده‌ها (dataset) را در قالب یک آرایه برمی‌گرداند اما عمل collect تمام عناصر موجود در مجموعه داده‌ها (dataset) را برمی‌گرداند که بهتر است در مواقعی که داده‌ها تعداد کمی دارند (مثلا پس از filter) استفاده شود.

- 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.

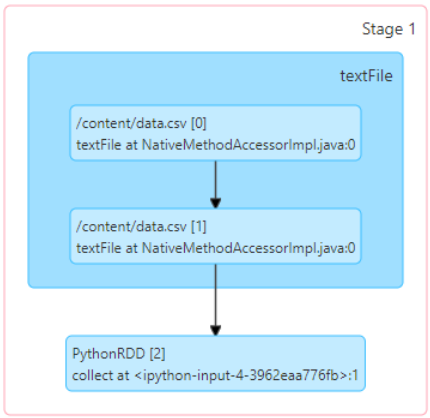
```
[10] !wget -qnc https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-linux-amd64.zip
     !unzip -n -q ngrok-stable-linux-amd64.zip
     !./ngrok authtoken 25hdZ065Vc4gRzz0PrT0hbdmJlL_7ie6Gaf3DnXNsonr42Wyi
     get_ipython().system_raw('./ngrok http 4050 &')
     !curl -s http://localhost:4040/api/tunnels | grep -Po 'public_url":"(?=https)\K[^"]*'

     Authtoken saved to configuration file: /root/.ngrok2/ngrok.yml
     https://b6ed-35-229-86-112.ngrok.io
```

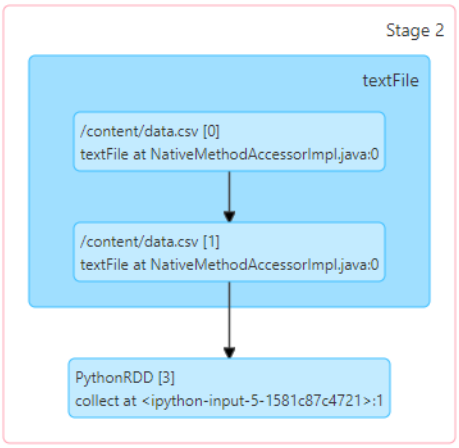
  - تبدیل لیست به RDD و فراخوانی collect برای محاسبه تعداد داده‌ها (len)



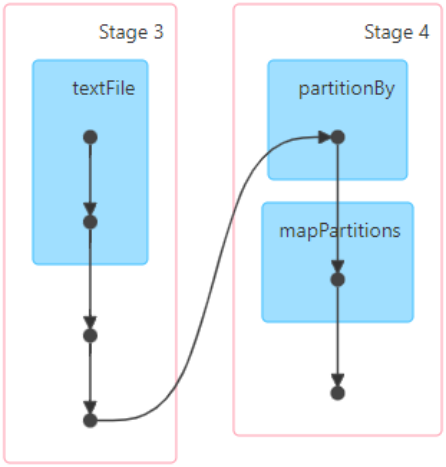
  - استخراج عنصر بیستم لیست از RDD و بازیابی با collect



- 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