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: محمد حمیدی اصفهانی
آدرس گیت: https://github.com/hajmamed/BigDataHws
آدرس فایل تمرین شماره یک :
https://github.com/hajmamed/BigDataHws/blob/main/BigData_HW1_Hamidi.ipynb

- **لطفا پاسخ تمارین حتما در سامانه‌ی کوئرا ارسال شود.**
- **لطفا پاسخ‌های خود را در خود سند سوال نوشته و در قالب یک فایل PDF ارسال کنید.**
- **نام سند ارسالی HW-{homework number}-{Name Family}-{student number}**
- **تمامی فایل‌های مورد نیاز این تمرین در این لینک قابل دسترس است.**
- **خروجی از هر مرحله‌ی تمرین را در سند خود بارگذاری کنید.**
- **کد + سند را در گیت بارگذاری کرده و لینک آن را در سند قرار دهید.**

در این تمرین، هدف ما آشنایی Action و Transformation در موتور تحلیلی Spark است.

۱. *منظور از Lazy Evaluation در Spark چیست؟ این مفهوم را همراه با یک مثال توضیح دهید.*

قاعدتا منظور از lazy Evaluation این مفهوم است که عملیات‌ها در spark در لحظه انجام نمی‌شوند. در حقیقت عملیات‌های مورد نیاز در یک Directed Acyclic Graph یا اصطلاحا DAG ذخیره می‌شود و زمانی که یک فرمان واکشی داده فراخوانی می‌شود تمامی عملیات‌ها اجرا می‌شود. در حقیقت اگر این عملیات‌های میانی به ترتیب اجرا شود حجم زیادی حافظه برای نگهداری اطلاعات میانی نیاز است که در واقع نتیجه مورد انتظار نیست.

در spark تمام ترنسفورم‌ها Laze Evaluated هستند؛ یعنی نتایج همان لحظه محاسبه نمی‌شوند بلکه به‌عنوان یک Lineage ذخیره می‌شوند. یک Lineage به اسپارک اجازه می‌دهد که بعدا در برنامه‌ی اجرا، یک سری ترنسفورم‌ها را دوباره مرتب‌سازی کند، آن‌ها را یکی کند یا آن‌ها را تبدیل به Stageهایی بکند تا عملیات بهینه انجام شود. Lazy Evaluation استراتژی اسپارک برای به تاخیر انداختن عملیات اجرا تا زمانی است که یک Action فراخوانی شود. یک Action باعث به حرکت افتادن Lazy Evalutaion تمام ترنسفورم‌های ذخیره شده می‌شود.

برای مثال می‌توان خروجی توابع range و xrange را مقایسه کرد. مزیت xrange در تولید اعداد این است که مقادیر را زمانی تولید می کند که به آنها نیاز داریم. تفاوت این تابع با تابع ()range این اســـت که تابع ()range، کل لیســت حاوی بازه اعداد داده شـده را یکجا تولید می‌کند که این امر برای اعداد بسـیار بزرگ، مشـکل کمبود حافظه را در برخواهد داشـت در حالی که تابع ()xrange هر عنصـر لیسـت را وقتی که به آن نیاز داریم، تولید می‌کند یعنی وقتی یک عنصر تولید شد و داخل حلقه برنامه، از آن استفاده کردیم، هنگام درخواست عدد بعدی توسط برنامه، در همان لحظه تولید و تحویل برنامه خواهد شد. به همین دلیل هنگام کار با بازه های بزرگ، استفاده از تابع xrange باعث استفاده ی موثرتر از حافظه خواهد شد.

۲. *منظور از Narrow Transmitaion (NT) و Wide Transmitation (WT) را در Spark همراه با یک مثال بیان کنید. تفاوت اصلی این دو مفهوم چیست؟*

هنگامی که هر پارتیشـن خروجی یک ترنسـفورم تنها نتیجه یک پارتیشـن ورودی است به آن NT میگوئیم. محاسـبات این نوع ترنسـفورم‌ها نسـبتا سـریع اسـت زیرا نیازی به تغییر داده نیسـت. همچنین در این نوع ترنسـفورم می‌توان از بهینه‌سـازی هایی نظیر pipelining بهره برد. زمانی که هر پارتیشـن خروجی نتیجه حاصل از تعدادی پارتیشن ورودی باشد به آن WT گفته می‌شود که قطعا بر روی سرعت اجرا تاثیر خواهد گذاشت.

مثال هایی از NT	مثال هایی از WT
map	groupByKey
mapPartition	join
filter	aggregateBykey
flatMap	aggregate
union	repartition

۳. با توجه به سوال پیشین، ۴ مورد از NT، WT و Action هایی که در اسپارک وجود دارند نام ببرید.

- NT
  - ()filter : با توجه به آرگومان‌های ورودی لیسـت RDD اصـلی را فیلتر کرده و یک لیسـت RDD برمی‌گرداند که حاوی مقادیر فیلتر شده است.
  - ()map: یک ترنسـفورم اسـت که ترسـفورم های مورد نیاز را بر روی تک تک عناصـر RDD یا Dataset اعمال می‌کند.
- WT
  - ()join : با توجه به عبارات وارد شده دو dataframe را join می‌کند

- aggregate() : ابتدا عناصـــر داخل یک پارتیشـــن را aggregate و ســـپس نتایج تمامی پارتیشن‌ها را ترکیب می‌کند.

- Action
  - collect() : تمامی عناصر یک لیست RDD را برمی‌گرداند
  - take() : تعداد K عنصر ابتدای لیست RDD را برمی‌گرداند.
  - distinct() : با حذف عناصر تکراری dataset را برمی‌گرداند.

۴. *برای آشنایی بیشتر با مفاهیم بیان شده و مقدمه‌ای بر توابع عملیات‌های زیر را انجام داده و خروجی هریک به همراه بلاک کد آن را گزارش دهید. مثالی از خروجی برای هر بخش نمایش داده شده است.*

- برای کار با اسپارک، کتاب‌خانه‌ای با نام pyspark وجود دارد.
- نوت‌بوکی بر روی گوگل کولب ایجاد کرده و این کتاب‌خانه را فراخوانی کنید.
- سپس یک لیست ۵۰ تایی از یک موضوع را برای خود درست کنید. برای مثال لیستی از (کتاب‌ها، نرم‌افزارها و ...)
- لیست خود را به RDD تبدیل کنید.

```
Top_Steel_Co  #List of Top 50 Steel Companies in China

"""# Make RDD"""

from pyspark.sql import SparkSession

spark = SparkSession.builder.appName('BigDataHw1').getOrCreate()
sc=spark.sparkContext

rdd = sc.parallelize(Top_Steel_Co)
```

در نهایت برای نمایش لیست RDD از دستور زیر استفاده می کنیم

```
rdd.collect()
```

```
['China Baowu Group (1)',
 'ArcelorMittal (2)',
 'HBIS Group (3)',
 'Shagang Group',
 'Nippon Steel Corporation (4)',
 'POSCO',
 'Ansteel Group',
 'Jianlong Group',
 'Shougang Group',
 'Shandong Steel Group',
 'Delong Steel Group',
 'Tata Steel Group',
 'Valin Group',
 'JFE Steel',
 'Nucor Corporation',
 'Hyundai Steel',
 'Fangda Steel',
 'IMIDRO (5) (e)',
 'Benxi Steel',
 'Liuzhou Steel',
 'Jingye Steel',
 'Novolipetsk Steel (NLMK)',
 'Baotou Steel',
 'Steel Authority of India Ltd. (SAIL)',
 'JSW Steel',
 'Rizhao Steel',
 'Sinogiant Group',
 'China Steel Corporation',
 'CITIC Pacific',
 'EVRAZ',
 'Shaanxi Steel',
 'Gerdau',
 'Zenith Steel',
 'Techint Group',
 'Shenglong Metallurgical',
 'Nanjing Steel',
 'Magnitogorsk Iron & Steel Works (MMK)',
 'United States Steel Corporation',
 'Sanming Steel',
 'Severstal',
 'Anyang Steel',
 'Donghai Special Steel',
 'Tsingshan Holding',
 'thyssenkrupp',
 'Metinvest Holding',
 'Xinyu Steel',
 'Jiuquan Steel',
 'Erdemir Group',
 'Steel Dynamics, Inc.',
 'Jinxi Steel']
```

- با کمک دستور filter بر روی RDD، از آن برای بازیابی عنصر ۲۰ام لیست خود استفاده کنید. (برابر با عنصر ۲۰ام باشد)

```
"""Retrive 20th item"""
```

```
rdd.filter(lambda x:(x)).collect()[19]
```

```
'Liuzhou Steel'
```

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

```
"""# Make All items in list to Uppercase"""

from pyspark.sql.functions import upper
rdd.map(lambda x:x.upper()).collect()
```

```
['CHINA BAOWU GROUP (1)',
 'ARCELORMITTAL (2)',
 'HBIS GROUP (3)',
 'SHAGANG GROUP',
 'NIPPON STEEL CORPORATION (4)',
 'POSCO',
 'ANSTEEL GROUP',
 'JIANLONG GROUP',
 'SHOUGANG GROUP',
 'SHANDONG STEEL GROUP',
 'DELONG STEEL GROUP',
 'TATA STEEL GROUP',
 'VALIN GROUP',
 'JFE STEEL',
 'NUCOR CORPORATION',
 'HYUNDAI STEEL',
 'FANGDA STEEL',
 'IMIDRO (5) (E)',
 'BENXI STEEL',
 'LIUZHOU STEEL',
 'JINGYE STEEL',
 'NOVOLIPETSK STEEL (NLMK)',
 'BAOTOU STEEL',
 'STEEL AUTHORITY OF INDIA LTD. (SAIL)',
 'JSW STEEL',
 'RIZHAO STEEL',
 'SINOGIANT GROUP',
 'CHINA STEEL CORPORATION',
 'CITIC PACIFIC',
```

- با کمک دستور groupby و map، لیست خود را بر اساس اولین کاراکتر آن دسته بندی کنید.

```
"""# Categorize based on First Character"""

rdd.map(lambda x:x.upper()).groupBy(lambda x: x[0]).mapValues(list).col-
lect()
```

```
[('C', ['CHINA BAOWU GROUP (1)', 'CHINA STEEL CORPORATION', 'CITIC PACIFIC']),
 ('S',
  ['SHAGANG GROUP',
   'SHOUGANG GROUP',
   'SHANDONG STEEL GROUP',
   'STEEL AUTHORITY OF INDIA LTD. (SAIL)',
   'SINOGIANT GROUP',
   'SHAANXI STEEL',
   'SHENGLONG METALLURGICAL',
   'SANMING STEEL',
   'SEVERSTAL',
   'STEEL DYNAMICS, INC.']),
 ('N',
  ['NIPPON STEEL CORPORATION (4)',
   'NUCOR CORPORATION',
   'NOVOLIPETSK STEEL (NLMK)',
   'NANJING STEEL']),
 ('J',
  ['JIANLONG GROUP',
   'JFE STEEL',
   'JINGYE STEEL',
   'JSW STEEL',
   'JIUQUAN STEEL',
   'JINXI STEEL']),
 ('L', ['LIUZHOU STEEL']),
 ('R', ['RIZHAO STEEL']),
 ('A', ['ARCELORMITTAL (2)', 'ANSTEEL GROUP', 'ANYANG STEEL']),
 ('H', ['HBIS GROUP (3)', 'HYUNDAI STEEL']),
 ('P', ['POSCO']),
 ('D', ['DELONG STEEL GROUP', 'DONGHAI SPECIAL STEEL']),
 ('T',
  ['TATA STEEL GROUP', 'TECHINT GROUP', 'TSINGSHAN HOLDING', 'THYSSENKRUPP']),
 ('V', ['VALIN GROUP']),
 ('F', ['FANGDA STEEL']),
 ('I', ['IMIDRO (5) (E)']),
 ('B', ['BENXI STEEL', 'BAOTOU STEEL']),
 ('E', ['EVRAZ', 'ERDEMIR GROUP']),
 ('G', ['GERDAU']),
 ('Z', ['ZENITH STEEL']),
 ('M', ['MAGNITOGORSK IRON & STEEL WORKS (MMK)', 'METINVEST HOLDING']),
 ('U', ['UNITED STATES STEEL CORPORATION']),
 ('X', ['XINYU STEEL'])]
```

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

```
"""# read data from text file and split each line into words

"""

text = sc.textFile("/content/apache_spark_text").flatMap(lambda line:
line.split(" "))
text.collect()

"""# count the occurrence of each word
```

```
"""

wordCounts = text.map(lambda word: (word, 1)).reduceByKey(lambda a,b:a +b)
wordCounts.collect()
```

```
[('Apache', 8),
 ('Spark', 17),
 ('is', 12),
 ('an', 6),
 ('open-source', 1),
 ('unified', 1),
 ('analytics', 1),
 ('engine', 1),
 ('provides', 3),
 ('programming', 3),
 ('clusters', 1),
 ('implicit', 1),
 ('fault', 1),
 ('developed', 2),
 ('at', 1),
 ('University', 1),
 ('of', 22),
 ("Berkeley's", 1),
 ('AMPLab,', 1),
 ('codebase', 1),
```

تصویر بالا قسمتی از خروجی اصلی است به دلیل طولانی بودن متن فقط قسمتی از خروجی در تصویر بالا گرفته شده است.

- چه تفاوتی بین Actionهای take و collect وجود دارد؟

دستور collect تمامی عناصر موجود در RDD را نمایش می‌دهد ولی دستور take می‌تواند یک ورودی دریافت کند و k عنصر ابتدی لیست را برگرداند. این کار بدین صورت است که ابتدا یک پارتیشن را اسکن می کند و از نتایج آن پارتیشن برای تخمین تعداد پارتیشن های اضافی مورد نیاز برای برآوردن محدودیت استفاده می کند.

هر دوی دستورات بالا در صورتی می‌تواند استفاده شود که اندازه لیست کوچک باشد در غیراینصورت ممکن است به مشکل حافظه بربخوریم.

- 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)