Task2 报告

题目:用PySpark实现PAT排名(参考地址:https://pintia.cn/problem-sets/16/problems/677)

我用到的RDD算子如下:

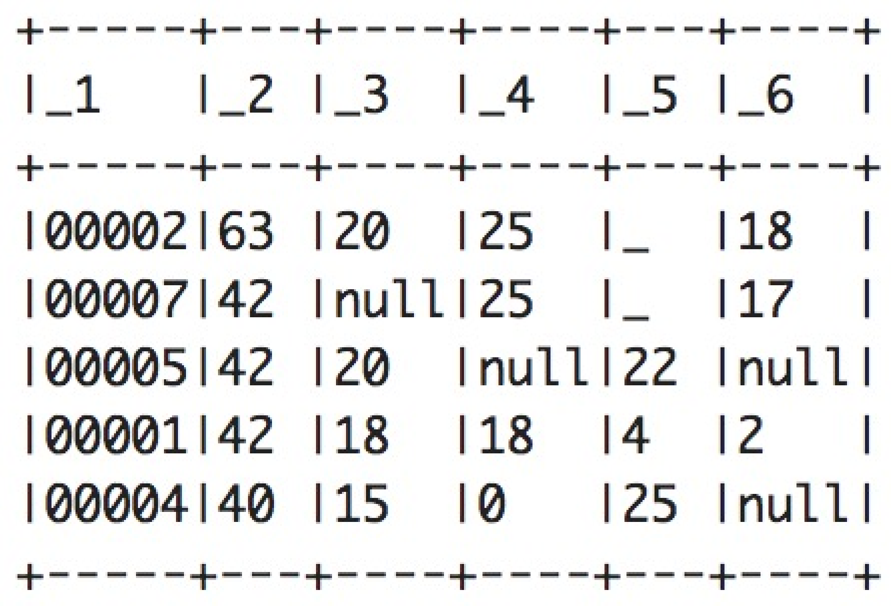
1. map
2. reduceByKey
3. aggregateByKey(zeroValue, mergeVal, mergeComb)
4. mapValues
5. sortBy
6. filter

用到的DataFrame基础函数如下:

1. createDataFrame
2. withColumn
3. 窗口函数rand().over

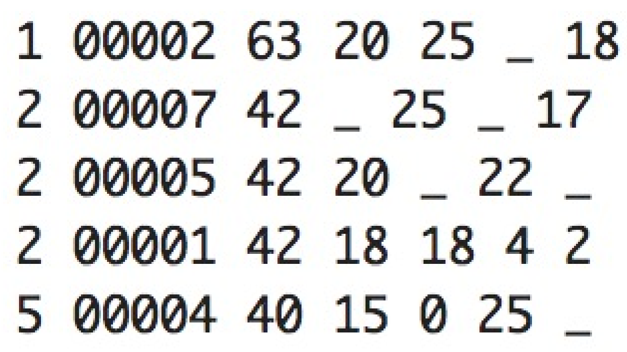
遇到的疑惑:

在单用PySpark得到如下图后(离目标还差排序名次):



```
+-----+---+----+----+---+----+
|_1   |_2 |_3  |_4  |_5 |_6  |
+-----+---+----+----+---+----+
|00002|63 |20  |25  |_  |18  |
|00007|42 |null|25  |_  |17  |
|00005|42 |20  |null|22 |null|
|00001|42 |18  |18  |4  |2   |
|00004|40 |15  |0   |25 |null|
+-----+---+----+----+---+----+
```

我尝试使用了将RDD变成iterable的List，然后一行一行打印，边打印边添加序号，但是觉得这种做法在行数超大的时候应该不可取。于是想到了用Spark的DataFrame APIs，用里面的窗口函数rank()，最终得到了想要的结果:

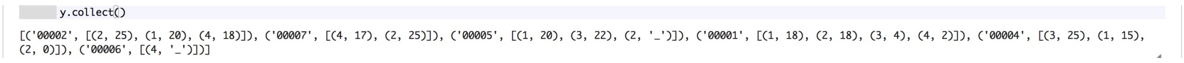


```
1 00002 63 20 25 _ 18
2 00007 42 _ 25 _ 17
2 00005 42 20 _ 22 _
2 00001 42 18 18 4 2
5 00004 40 15 0 25 _
```

代码如下:

```
<!--输入数据-->
info = [("00002", 2, 12), ("00007", 4, 17), ("00005", 1, 19), ("00007", 2,  25),
("00005", 1, 20), ("00002", 2, 2), ("00005", 1, 15), ("00001", 1, 18), ("00004",
3, 25), ("00002", 2, 25), ("00005", 3, 22), ("00006", 4, -1), ("00001", 2, 18),
("00002", 1, 20), ("00004", 1, 15), ("00002", 4, 18), ("00001", 3, 4), ("00001",
4, 2), ("00005", 2, -1), ("00004", 2, 0)]

<!--当ID和题号相同时，取出得分最高的。例如，“0001”号学生对于第一题提交了两次，分别得分15和
20，那么应该取20作为“0001”号对于第一题的最终得分-->
high_marks = sc.parallelize(info).map(lambda line : ((line[0], line[1]),
line[2])).reduceByKey(max)

<!--将学号作为键，题号和该题得分作为值(该步用map算子实现)-->
marks = sc.parallelize(high_marks.collect()).map(lambda line : (line[0][0],
(line[0][1], line[1])))

<!--将同一学生的不同题目的最高得分汇总在一起，这里有一个特殊情况就是:当得分为-1时说明该题压根编
译不通过(要和题目得分为0区别开)，所以我们给该题得分赋为‘_’(该步用aggregateByKey算子实现)-->
zeroValue = []
mergeVal = (lambda aggregated, el: aggregated + [el] if el[1]!=-1 else
aggregated + [(el[0], '_')])
mergeComb = (lambda agg1,agg2: agg1 + agg2)

y = marks.aggregateByKey(zeroValue, mergeVal, mergeComb)
```

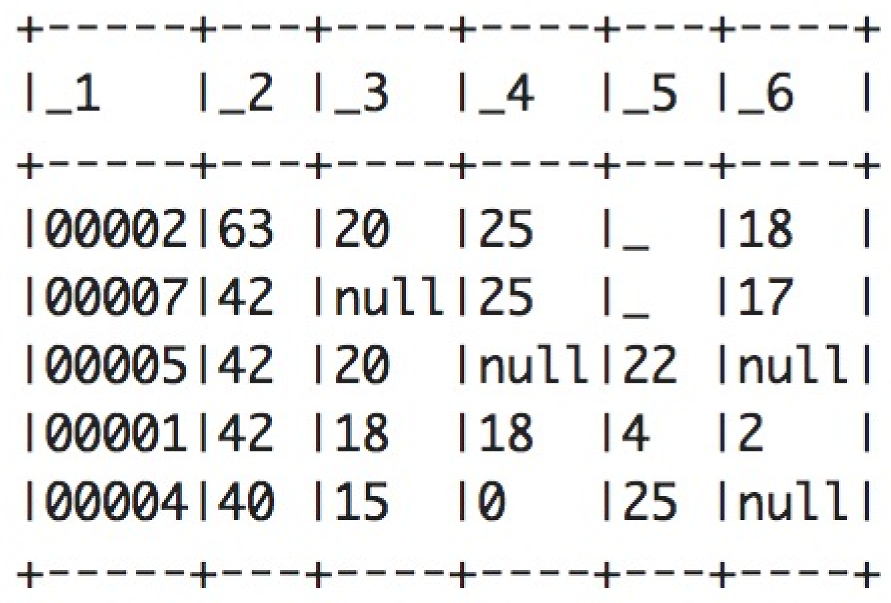
现得到效果如下



```
y.collect()

[('00002', [(2, 25), (1, 20), (4, 18)]), ('00007', [(4, 17), (2, 25)]), ('00005', [(1, 20), (3, 22), (2, '_')]), ('00001', [(1, 18), (2, 18), (3, 4), (4, 2)]), ('00004', [(3, 25), (1, 15),
(2, 0)]), ('00006', [(4, '_')])]
```

```
<!--定义UDF，该UDF意思是:对于某一学生的从未提交过的题目，该题目得分赋为‘_’，同时计算每个学生各
个题目的总分和-->
def add_(x):
    sum = 0
    full_set = set([1, 2, 3, 4])
    s = set()
    for i in range(0, len(x)):
        s.add(x[i][0])
        if(x[i][1]!='_'):
            sum += x[i][1]
    x.insert(0, (0, sum))
    missing = full_set-s
    for e in missing:
        x.append((e, '_'))
    return x

<!--按照总分和对RDD进行排序-->
def sort_(x):
    x = sorted(x, key=lambda a : a[0])
    return x

<!--用到mapValues和sortBy算子-->
y =  y.mapValues(add_).mapValues(sort_).sortBy(lambda x : -x[1][0][1])
result = y.filter(lambda line : line[1][0][1]!=0)

<!--去掉题目编号，只显示总分和各题分数-->
def f(x):
    for i in range(0, 5):
        x[i] = x[i][1]
    return x

result = result.mapValues(f).map(lambda line : (line[0], line[1][0], line[1][1],
line[1][2], line[1][3], line[1][4]))
```

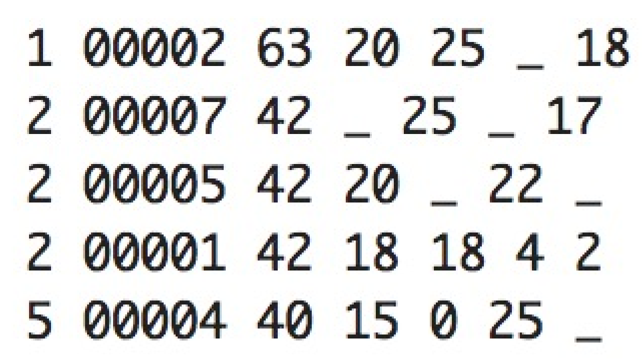
现得到结果如下:



```
+-----+---+----+----+---+----+
|_1   |_2 |_3  |_4  |_5 |_6  |
+-----+---+----+----+---+----+
|00002|63 |20  |25  |_  |18  |
|00007|42 |null|25  |_  |17  |
|00005|42 |20  |null|22 |null|
|00001|42 |18  |18  |4  |2   |
|00004|40 |15  |0   |25 |null|
+-----+---+----+----+---+----+
```

```
<!--将RDD转换为DataFrame以便利用DataFrame的窗口函数-->
from pyspark.sql.window import Window
from pyspark.sql.types import  StructField, StructType, StringType
from pyspark.sql.functions import col, rank, desc

df = spark.createDataFrame(result, StructType([StructField("ID", StringType(),
False), StructField("sum", StringType(), False), StructField("course1",
StringType(), True),  StructField("course2", StringType(), True),
StructField("course3", StringType(), True),  StructField("course4",
StringType(), True)]))

<!--添加排序序号-->
df = df.withColumn("rank", rank().over(Window.orderBy(desc("sum"))))
df = df.select(["rank", "ID", "sum", "course1", "course2", "course3",
"course4"])

<!--再将DataFrame转换回RDD-->
rdd = df.rdd
def p(a):
    values = [str(a[i]) for i in range(0, len(a))]
    print(' '.join(values))

<!--按照特定打印格式对RDD iterable后的结果进行打印-->
[p(a) for a in rdd.collect()]
```

现得到最终结果如下:



```
1 00002 63 20 25 _ 18
2 00007 42 _ 25 _ 17
2 00005 42 20 _ 22 _
2 00001 42 18 18 4 2
5 00004 40 15 0 25 _
```