MSDS 697
Distributed Data Systems

DIANE WOODBRIDGE, PH.D

Week 3 - Comments (What you liked/disliked so far? What should I do for you?)

Alumni Feedback

Monday, November 1st

Shruti Roy 9:00 AM

Hi professor Diane,

Happy Halloween 🎃 to you and your family!

I just wanted to say thank you for your distributed computing class! I am setting up a lot of pipelines while working at RDC and I was genuinely able to do that a lot because of what I learnt in your classes.

dwoodbridge 9:00 AM

Awww - this means so much to me! I will share what you said in my class 🙂 I am teaching it this module! 🙂

❤️ 1

Shruti Roy 9:01 AM

Absolutely Professor Diane! Super useful stuff

❤️ 1

9:01 I hope you and your family are in good health and doing well ❤️

Announcement

Group Assignment 1 - Due tomorrow midnight!

- Task 1 (Nov 5) : Data Selection

Quiz - 1st floor

- Nov 11, **9:00 - 9:55 AM :** Multiple Choice (Closed-book), and Programming (Open-book).
  - Study list and example quiz are available.

Office Hour : Tuesday 9:15-10 am

Professionalism

- **Class Attendance**
  - No cellphones, social media, slack, texting during the class .
  - Please only use laptops for class-related purposes.

- **Assignment**
  - No late submissions are allowed.
  - Do not share any homework and exam files - All the codes including the last 5 years will be tested by Moss (Measure Of Software Similarity).
  - Make sure that your code runs in Python 3.10 and Pyspark 3.3

Review

Hadoop MapReduce vs. Spark

Spark Overview

Spark Components

Spark Use Cases

Resilient Distributed Dataset (RDD)

RDD Operations

- Transformation
- Action (Week 3)

Contents

RDD Operations

- Transformation
- Action

RDD Operations - Transformation

Transformation

- Construct a new RDD from an existing RDD.

```
line_with_spark = lines.filter(lambda x : "spark" in x)
```

- Lazy Evaluation.
  - Computation doesn't take place until an action is triggered.
- Return new RDDs.
  - Doesn't change the original RDDs.

RDD Operations - Transformation

Transformation Operation Types

map(*func*)	Return a new distributed dataset formed by passing each element of the source through a function *func*.
filter(*func*)	Return a new dataset formed by selecting those elements of the source on which *func* returns true.
distinct()	Return a new dataset that contains the distinct elements of the source dataset.
union(*otherDataset*)	Return a new dataset that contains the union of the elements in the source dataset and the argument.
intersection(*otherDataset*)	Return a new RDD that contains the intersection of elements in the source dataset and the argument.
subtract(*otherDataset*)	Return each value in self that is not contained in oterDataset.
cartesian(*otherDataset*)	Return the Cartesian product of the RDD and otherDataset.
sortBy(*func*, *ascending=True*)	Sorts this RDD by the given *func*.

http://spark.apache.org/docs/latest/programming-guide.html

Contents

RDD Operations

- Transformation
- **Action**

RDD Operations - Action

- Compute a result based on an RDD and return the result to the driver program.

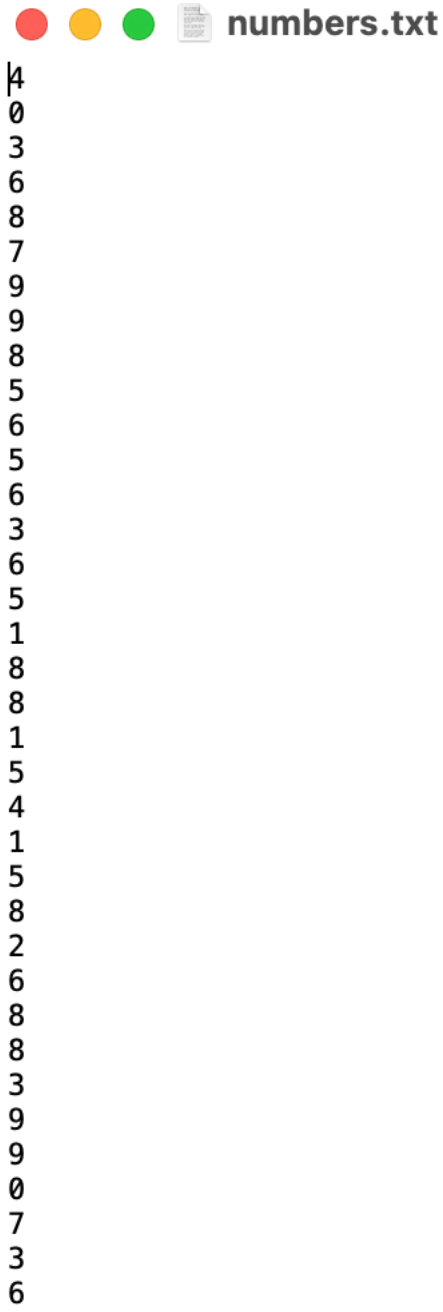
```
input_rdd.count()
input_rdd.collect()
```

- Return non-RDDs like a single number, string, array, etc.

Today's Example

For data/numbers.txt, calculate the (count, sum) of all odd numbers.

Expected output : (49, 233)



RDD Operations - Action

Action Operation Types

collect()	Return all the elements of the dataset as an array at the driver program. This is usually useful after a filter or other operation that returns a sufficiently small subset of the data.
first()	Return the first element of the dataset (similar to take(1)).
take(n)	Return an array with the first n elements of the dataset.
count()	Return the number of elements in the dataset.
countByValue()	Return the number of times each elemnt occurs in the RDD.
reduce(func)	Combine the elements of the RDD together in parallel. (eg. Sum)
fold(zero, func)	Same as reduce(), but with the provided zero value.
aggregate(zero, SeqOp, combOp)	Similar to reduce() but used to return a different type.
saveAsTextFile(path)	Write the elements of the dataset as a text file (or set of text files) in a given directory in the local filesystem, HDFS or any other Hadoop-supported file system. Spark will call toString on each element to convert it to a line of text in the file.
mean(), sum(), min(), max(), variance(), stdev()	Returns mean, sum, minimum, maximum, variance and standard deviation.

http://spark.apache.org/docs/latest/programming-guide.html

RDD Operation - Action

Numeric RDD Action Types

- mean()
  - Return the mean of the RDD's elements.
- sum()
  - Add up the elements in the RDD.
- max()
  - Return the maximum item in the RDD.
- min()
  - Return the minimum item in the RDD.
- variance()
  - Return the variance of the RDD's elements.
- stdev()
  - Return the standard deviation of the RDD's elements.

https://spark.apache.org/docs/latest/api/python/pyspark.html#pyspark.RDD

ex01

For data/numbers.txt, calculate the count, sum, mean, max, min, variance and standard deviation for the odd numbers.

```
odd_numbers.count()
```

```
49
```

```
odd_numbers.sum()
```

```
233
```

```
odd_numbers.mean()
```

```
4.755102040816326
```

```
odd_numbers.max()
```

```
9
```

```
odd_numbers.min()
```

```
1
```

```
odd_numbers.variance()
```

```
8.10329029571012
```

```
odd_numbers.stdev()
```

```
2.8466278814959503
```

RDD Operations - Action

collect()
- Return all the elements of the dataset as an array at the driver program.
- This is usually useful after a filter or other operation that returns a sufficiently small subset of the data.

first()
- Return the first element of the dataset (similar to take(1)).

take(n)
- Return an array with the first n elements of the dataset.

count()
- Return the number of elements in the dataset.

countByValue()
- Return the number of times each element occurs in the RDD.

https://spark.apache.org/docs/latest/api/python/pyspark.html#pyspark.RDD

ex02

For data/numbers.txt,

- What are the odd numbers in the file?
- What is the first odd number?
- What are the 5 odd numbers?
- What are the occurrences of each odd number?

https://spark.apache.org/docs/latest/api/python/pyspark.html#pyspark.RDD

ex02

For data/numbers.txt,

- What are the odd numbers in the file?
- What is the first odd number?
- What are the 5 odd numbers?
- What are the occurrences of each odd number?

```
odd_numbers.collect()
```

```
odd_numbers.first()
```

```
odd_numbers.take(5)
```

```
odd_numbers.count()
```

```
odd_numbers.countByValue()
```

```
odd_numbers.countByValue().items()
```

rdd.count().count() works without any errors, where rdd is an RDD.

True

False

rdd.count().count() works without any errors, where rdd is an RDD.

True

False

rdd.count().count() works without any errors, where rdd is an RDD.

True

False

RDD Operations - Action

reduce(func)

- Take a function that operates on **two** elements of the type in your RDD.
- Returns a new element of the same type.
- Ex. `sum = rdd.reduce(lambda x, y : x+y)`

https://spark.apache.org/docs/latest/api/python/pyspark.html#pyspark.RDD

ex03

For data/numbers.txt, calculate the sum of the odd numbers.

ex03

For data/numbers.txt, calculate the sum of the odd numbers.

```
numbers = sc.textFile("../data/numbers.txt")
```

```
odd_numbers = numbers.map(lambda x : int(x))\
                     .filter(lambda x : x % 2 == 1)
```

```
odd_numbers.reduce(lambda x,y : x + y)
```

RDD Operations - Action

fold(zeroValue, func)

- Take a function with the same signature as needed for reduce().
- Take a "zero value" to be used for the initial call on each partition.
- Returns a new element of the same type.
- fold() is useful when the given RDD is empty. cf. reduce()

https://spark.apache.org/docs/latest/api/python/pyspark.html#pyspark.RDD

ex04

For data/numbers.txt, calculate the sum of the odd numbers with different zero values using fold().

ex04

For data/numbers.txt, calculate the sum of the odd numbers with different zero values using fold().

```
numbers.getNumPartitions()
```

```
8
```

```
odd_numbers = numbers.map(lambda x : int(x))\
                     .filter(lambda x : x%2 == 1)
```

```
odd_numbers.fold(0, lambda x,y : x+y)
```

```
233
```

```
odd_numbers.fold(1, lambda x,y : x+y)
```

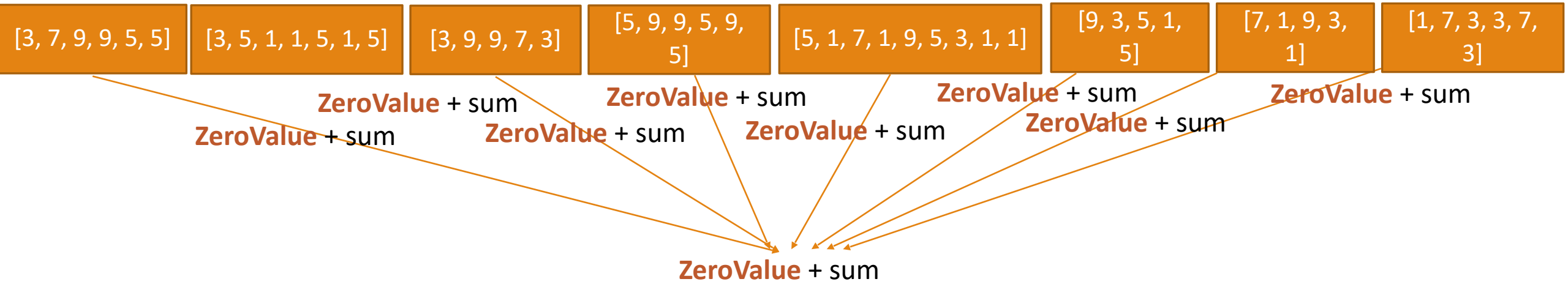
```
242
```

```
odd_numbers.fold(2, lambda x,y : x+y)
```

```
251
```

ex04

For data/numbers.txt, calculate the sum of the odd numbers with different zero values using fold().



ex04

For data/numbers.txt, calculate the sum of the odd numbers with different zero values using fold().

To do : Try it with different number partitions.

ex04

For data/numbers.txt, calculate the sum of the odd numbers with different zero values using fold().

fold() is useful when the given RDD is empty. cf. reduce()

```
numbers = sc.parallelize([])
```

```
numbers.reduce(lambda x,y : x+y)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-19-0efd3e2d4ec0> in <module>
----> 1 numbers.reduce(lambda x,y : x+y)

~/anaconda3/envs/MSDS694/lib/python3.6/site-packages/pyspark/rdd.py in reduce(self, f)
    863             if vals:
    864                 return reduce(f, vals)
--> 865             raise ValueError("Can not reduce() empty RDD")
    866
    867         def treeReduce(self, f, depth=2):

ValueError: Can not reduce() empty RDD
```

```
numbers.fold(0, lambda x,y : x+y)
```

```
0
```

RDD Operations - Action

aggregate(zeroValue, seqOp, combOp)

- Aggregate the elements of each partition using zeroValue and seqOp → Aggregate the results for all the partitions, using combOp and zeroValue.
  - zeroValue : should be in a format we want to return.
    - Can return an element of a different type from the original RDD.
  - seqOp : Function to combine the elements from the RDD. Runs within the partition.
  - combOp : Function to merge two accumulators resulted from seqOp and zeroValue, given that each nodes accumulates its own results locally.

https://spark.apache.org/docs/latest/api/python/_modules/pyspark/rdd.html#RDD.aggregate

ex05

For data/numbers.txt, calculate the number (count) and sum of the odd numbers as a pair using the aggregate function.

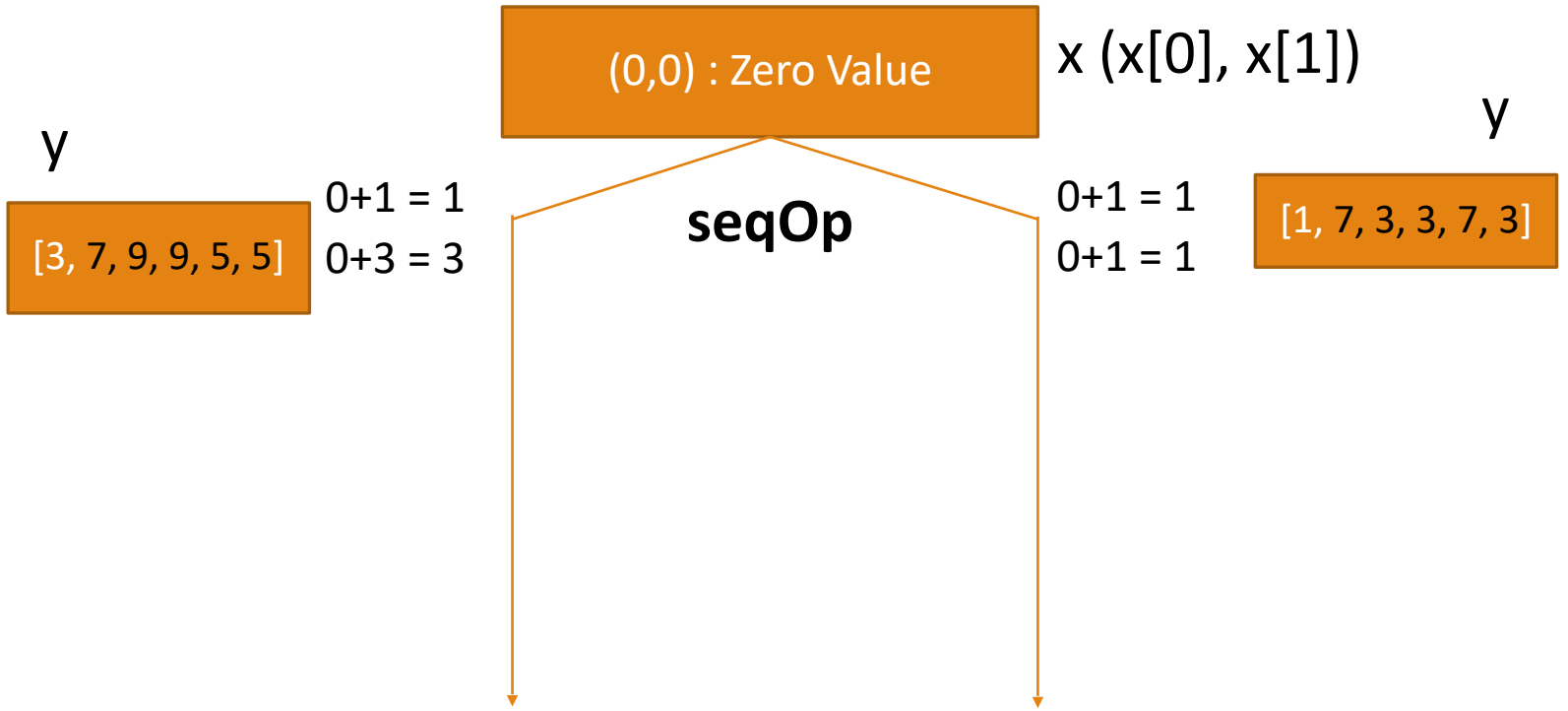
ex05

For data/numbers.txt, calculate the number (count) and sum of the odd numbers as a pair using the aggregate function.

- The type to return : (Count, Sum) ➔ (Int, Int) - the format of the zero value
- In the same partition (like Map)
  - Every time encountering a new number, increase the counter by 1
    - counter + 1
  - Every time encountering a new number, add the two numbers
    - accumulated sum + new number
- Merging results from partitions (like the last step in Reduce)
  - Add resulted counters together
  - Add resulted sums together

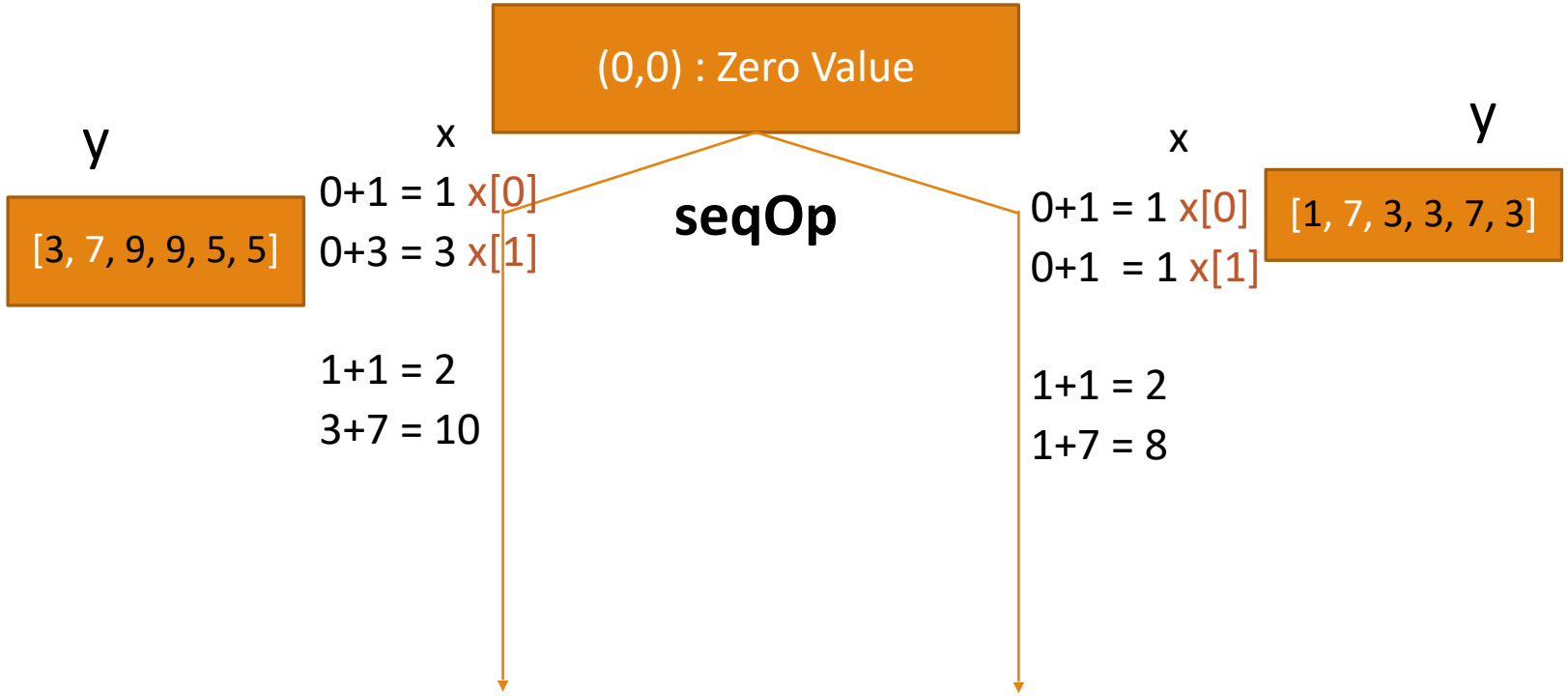
ex05

For data/numbers.txt, calculate the number and sum of the odd numbers as a pair using the aggregate function.



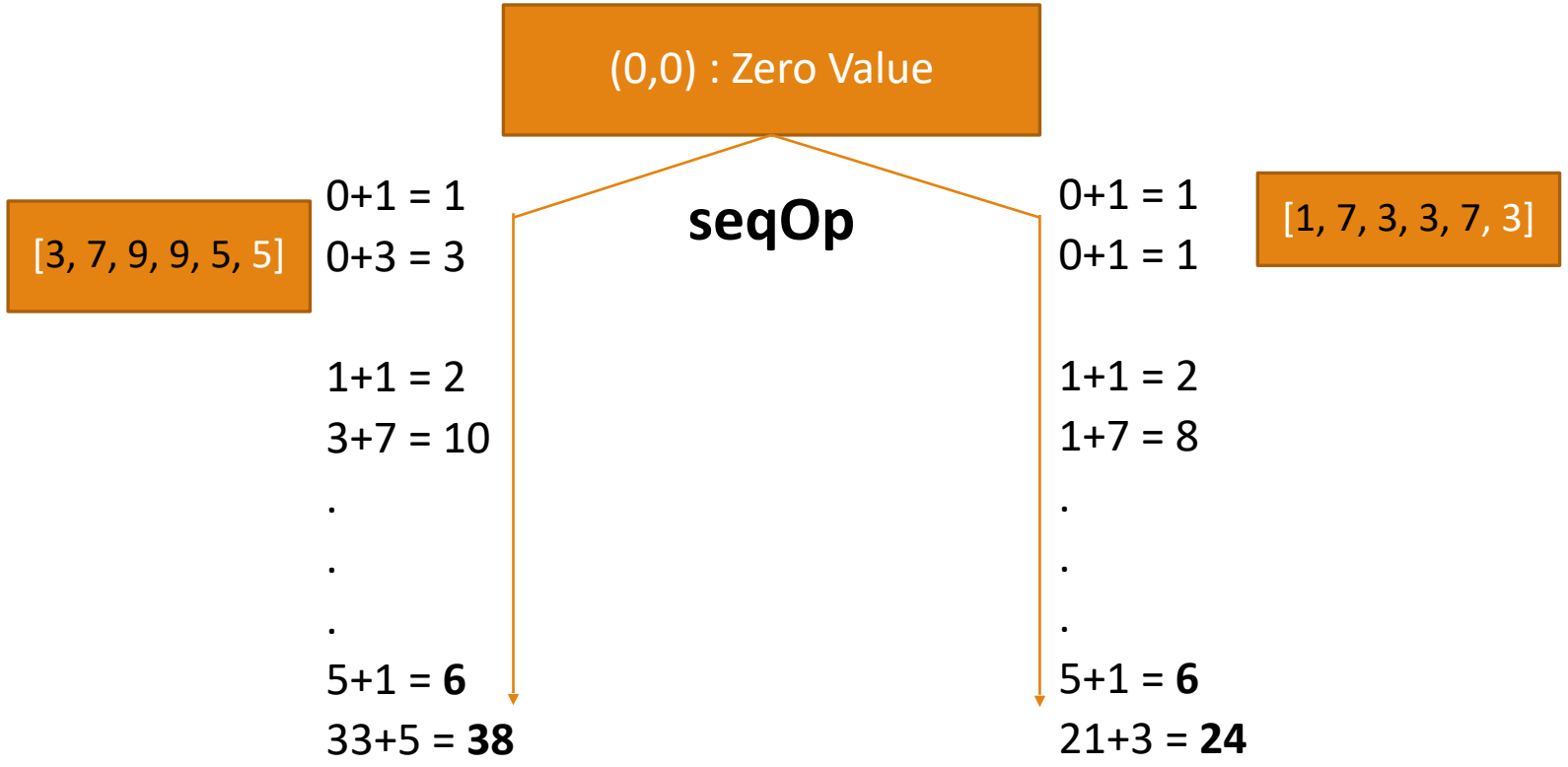
ex05

For data/numbers.txt, calculate the number and sum of the odd numbers as a pair using the aggregate function.



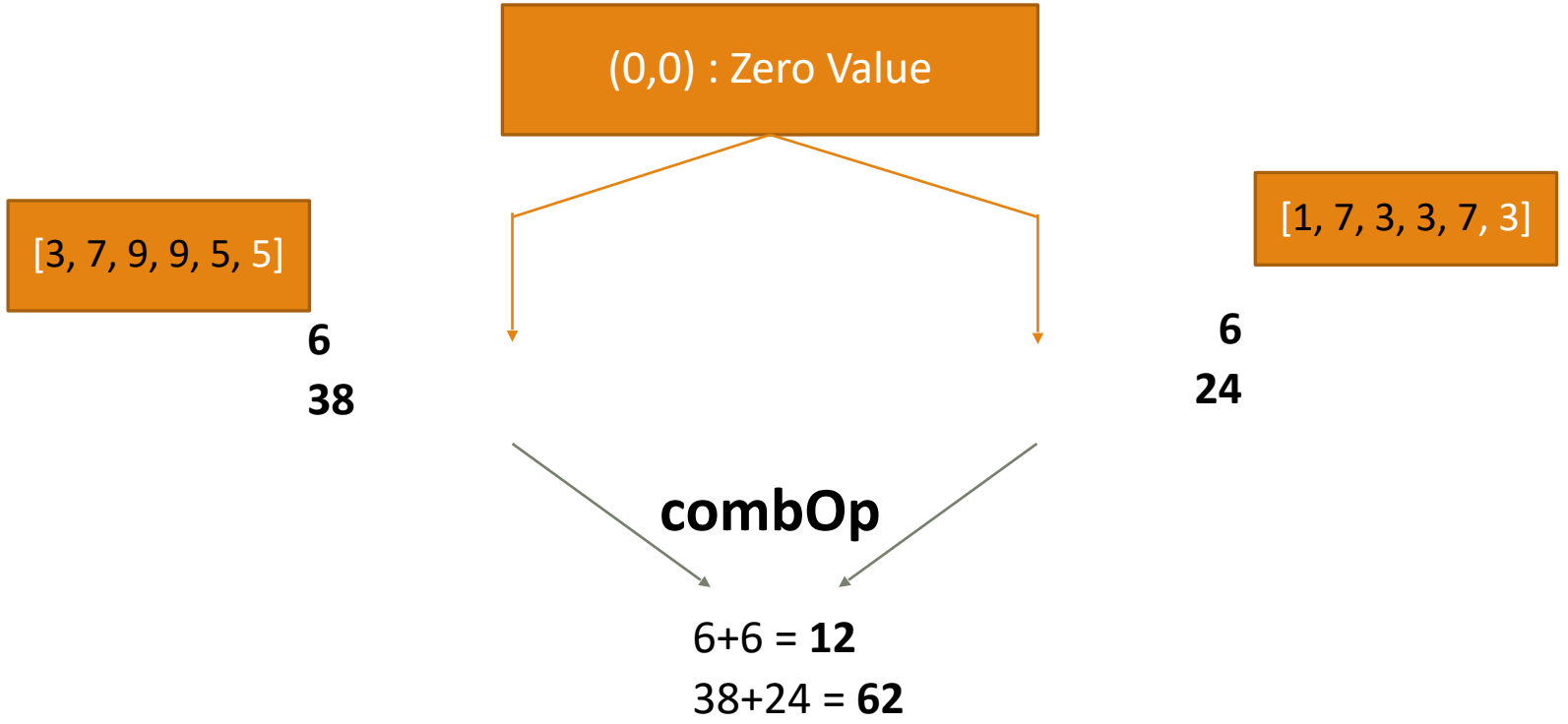
ex05

For data/numbers.txt, calculate the number and sum of the odd numbers as a pair using the aggregate function.



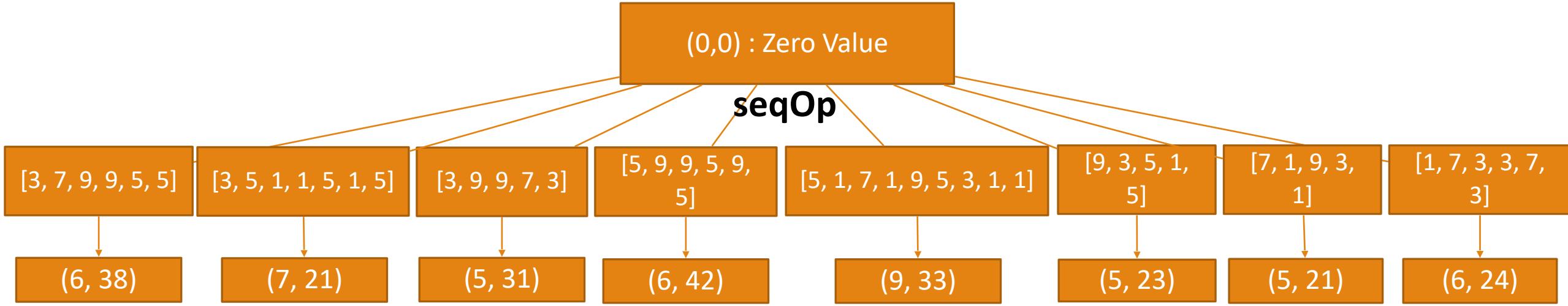
ex05

For data/numbers.txt, calculate the number and sum of the odd numbers as a pair using the aggregate function.



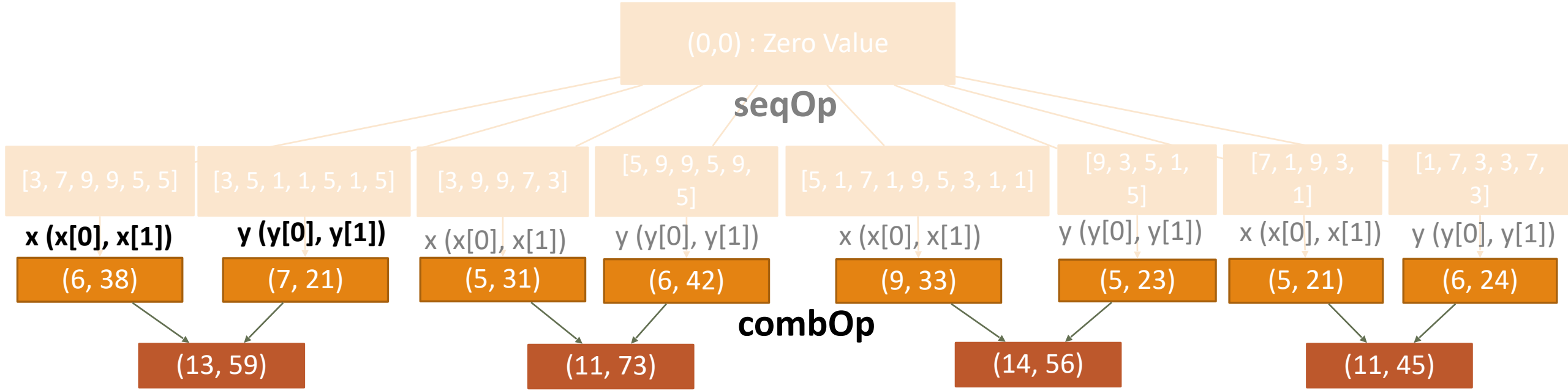
ex05

For data/numbers.txt, calculate the number and sum of the odd numbers as a pair using the aggregate function.



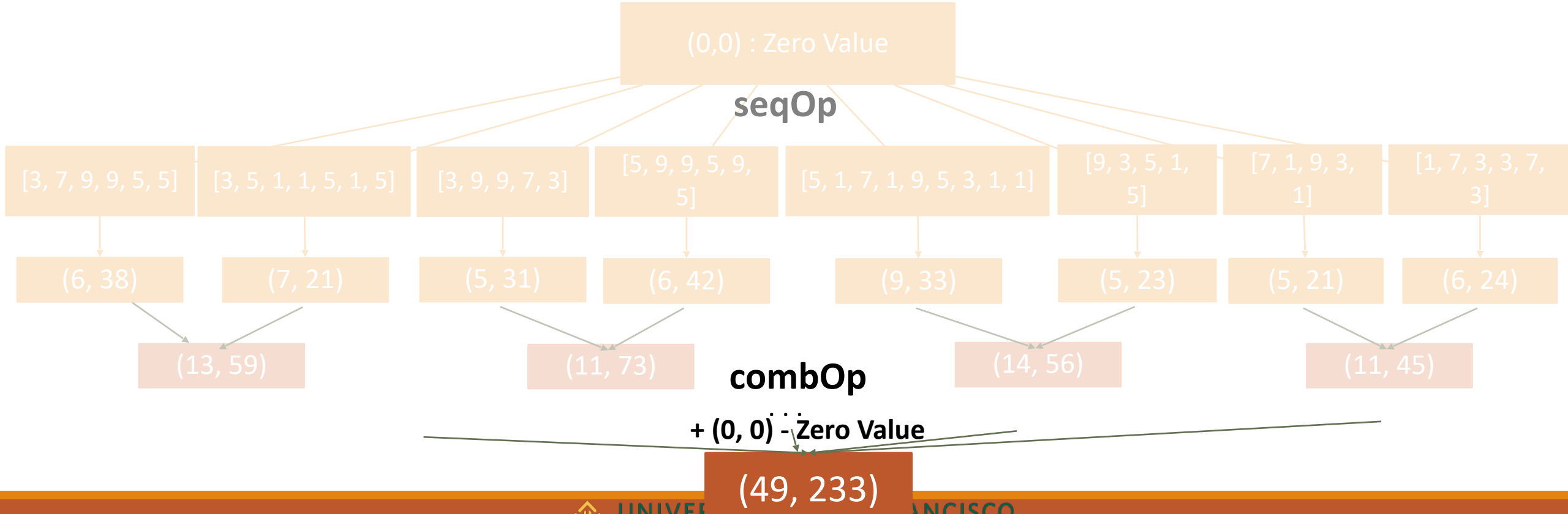
ex05

For data/numbers.txt, calculate the number and sum of the odd numbers as a pair using the aggregate function.



ex05

For data/numbers.txt, calculate the number and sum of the odd numbers as a pair using the aggregate function.



ex05

For data/numbers.txt, calculate the number and sum of the odd numbers as a pair using the aggregate function.

```
odd_numbers.aggregate((0, 0),
                      (lambda x, y: (x[0] + 1, x[1] + y)),
                      (lambda x, y: (x[0] + y[0], x[1] + y[1])))
```

```
(49, 233)
```

RDD Operations - Action

saveAsTextFile(new_subdir_name)

- Return multiple output files underneath the new_subdir_name, as Spark writes the output from multiple nodes.
  - The number of files = the number of partition + 1 (_SUCCESS)

https://spark.apache.org/docs/latest/api/python/pyspark.html#pyspark.RDD

ex06

Store odd_numbers, the RDD in the “ex06_output” directory.

Read data from the “ex06_output” directory.

ex06

Store odd_numbers, the RDD in the “ex06_output” directory.

Read data from the “ex06_output” directory.

Store odd_numbers, the RDD in the “ex06_output” directory.

```
odd_numbers.saveAsTextFile("ex06_output")
```

Read data from the “ex06_output” directory.

```
odd_numbers_2 = sc.textFile("ex06_output")
odd_numbers_2.collect()
```

Contents

RDD Operations

- Transformation
- Action

How many files are created in the output folder

1	**A**
8	**B**
9	**C**
None of the above	**D**

How many files are created in the output folder

1	**A**
8	**B**
9	**C**
None of the above	**D**

How many files are created in the output folder

1	A
8	B
9	C
None of the above	D

RDD Operations - Action

Action Operation Types

collect()	Return all the elements of the dataset as an array at the driver program. This is usually useful after a filter or other operation that returns a sufficiently small subset of the data.
first()	Return the first element of the dataset (similar to take(1)).
take(n)	Return an array with the first n elements of the dataset.
count()	Return the number of elements in the dataset.
countByValue()	Return the number of times each elemnt occurs in the RDD.
reduce(func)	Combine the elements of the RDD together in parallel. (eg. Sum)
fold(zero, func)	Same as reduce(), but with the provided zero value.
aggregate(zero, SeqOp, combOp)	Similar to reduce() but used to return a different type.
saveAsTextFile(path)	Write the elements of the dataset as a text file (or set of text files) in a given directory in the local filesystem, HDFS or any other Hadoop-supported file system. Spark will call toString on each element to convert it to a line of text in the file.
mean(), sum(), min(), max(), variance(), stdev()	Returns mean, sum, minimum, maximum, variance and standard deviation.

http://spark.apache.org/docs/latest/programming-guide.html

Reference

Spark Online Documentation, http://spark.apache.org/docs/latest/

Zecevic, Petar, et al. Spark in Action, Manning, 2016.

Aven, Jeffrey. Apache Spark in 24 Hours, Sams Publishing , 2016.

Karau, Holden, et al. Learning spark: lightning-fast big data analysis. O'Reilly Media, Inc., 2015.

AWS Online Documentation, https://aws.amazon.com/documentation/