Mapreduce – Duplikate finden und auflisten

```
from mrsim import mr s imu l a t o r
d e f map ( key , v a l ) :
# r e s  List
r e s = [ ]
# ( key , v a l )  Tupel , d e r d e r L i s t e h i n z u g e f u e g t w i r d
r e s . append ( ( key , v a l ) )
r e t u r n r e s
d e f r e d u c e ( key , v a l ) :
r e s = [ ]
i f ( l e n ( v a l ) > 1 ) :
# remove d u p l i c a t e s ( o r i g i n a l o r d e r i s n o t p r e s e r v e d )
v a l = l i s t ( s e t ( v a l ) )
# r e s . append ( ( key , v a l ) )
r e s . append ( ( s t r ( key ) + ' : ' + s t r ( v a l ) ) )
r e t u r n r e s
i n p u t s = [ ( 1 2 3 , 'Name1 ' ) , ( 1 2 3 , 'Name2 ' ) , ( 4 5 6 , 'Name1 ' ) ,
( 3 4 5 , 'Name3 ' ) , ( 4 5 6 , 'Name2 ' ) , ( 1 2 3 , 'Name1 ' ) ]
r e s = mr s imu l a t o r ( i n p u t s , map , r e d u c e )
p r i n t ( r e s )
```

Python Suchen

a) Merge Sort:

```
#merge f u n c t i o n
d e f merge ( a r r 1 , a r r 2 ) :
r e s u l t = [ ]
whi l e l e n ( a r r 1 ) != 0 and l e n ( a r r 2 ) != 0 :
i f a r r 1 [ 0 ] < a r r 2 [ 0 ] :
r e s u l t . append ( a r r 1 [ 0 ] )
a r r 1 . remove ( a r r 1 [ 0 ] )
e l s e :
r e s u l t . append ( a r r 2 [ 0 ] )
a r r 2 . remove ( a r r 2 [ 0 ] )
i f l e n ( a r r 1 ) == 0 :
r e s u l t += a r r 2
e l s e :
r e s u l t += a r r 1
r e t u r n r e s u l t
#Code f o r merge s o r t
d e f me r g e s o r t ( a r r ) :
i f l e n ( a r r ) <= 1 :
r e t u r n a r r
middl e = l e n ( a r r ) // 2
l e f t = me r g e s o r t ( a r r [ : middl e ] )
r i g h t = me r g e s o r t ( a r r [ middl e : ] )
r e t u r n merge ( l e f t , r i g h t )
# t e s t
p r i n t ( me r g e s o r t ( [ 3 , 6 , 8 , 1 0 , 1 , 2 , 1 ] ) )
1
```

b) Depth-first Search:

```
d e f d e e p Se a r c h ( graph , s t a r t , d e s t i n a t i o n , v i s i t e d No d e s = [ ] ) :
i f s t a r t == d e s t i n a t i o n :
r e t u r n True
i f gr aph [ s t a r t ] :
f o r node in f i l t e r ( lambda x : x n o t in v i s i t e d No d e s ,
gr aph [ s t a r t ] ) :
v i s i t e d No d e s . append ( node )
i f d e e p Se a r c h ( graph , node , d e s t i n a t i o n , v i s i t e d No d e s ) :
r e t u r n True
```

```
else:
visitedNodes.append('Node'+str(node)+
'isfinished,nopathfrom'+str(node)+
'to'+str(destination))
returnFalse
adjList={0:[1,2],1:[2,3],2:[4],3:[4,5],4:[],5:[]}
visitedNodes=[]
print(deepSearch(adjList,0,5,visitedNodes))
print(visitedNodes)
```

c) Breadth-first Search:

```
import queue as q
defbreitensuche(adj,start,destination,visitedNodes=[]):
queue = q.Queue()
queue.put(start)
whilequeue.qsize()>0:
currentNode=queue.get()
visitedNodes.append(currentNode)
forsuccessorinadj[currentNode]:
ifsuccessorinvisitedNodes:
visitedNodes.append('Node'+str(successor)+
'wasalreadyvisitedearlier.')
continue
elifsuccessor==destination:
visitedNodes.append(successor)
returnTrue
else:
queue.put(successor)
returnFalse
adjList={0:[1,2],1:[2,3],2:[4],3:[4,5],4:[],5:[]}
visitedNodes=[]
print(breitensuche(adjList,0,5,visitedNodes))
print(visitedNodes)
```

Spark

```
fromoperatorimportadd
frompysparklingimportContext
sc=Context()
```

a) Find 25 suppliers with the lowest account balance.

```
top25=(
sc.textFile('supplier.tbl')
.map(lambdaline:line.split('j'))
.map(lambdarow:(row[1],float(row[5])))
.sortBy(lambdarow:row[1])
.take(25)
)
# moreefficientonPySpark(withoutglobalsorting,whichcausessubstantial
#shuffle/repartitioning),exactlythesameasabovesolutiononPysparkling,
# which implementstop(num,key)assortBy(key,ascending=False).take(num)
top25=(
sc.textFile('supplier.tbl')
.map(lambdaline:line.split('j'))
.map(lambdarow:(row[1],float(row[5])))
.top(25,key=lambdarow:row[1])
# or.takeOrdered(25,key=lambdarow:row[1])#onlyavailableonPySpark
)
print(top25)
```

b) How many suppliers have a positive account balance?

```
numposbal=(
sc.textFile('supplier.tbl')
.map(lambdaline:float(line.split('j')[5]))
.filter(lambdaacctbal:acctbal>0)
```

```
.count()
)
print(numposbal)
```

c) Find out all brands produced by the same manufacturer and calculate the items number and the
total sales price for each brand of each manufacturer.

```
brandmfgrcount=(
sc.textFile('part.tbl')
.map(lambdaline:line.split('j'))
.map(lambda row:((row[2],row[3]),1))
.reduceByKey(add)
)
brandmfgrsum=(
sc.textFile('part.tbl')
.map(lambdaline:line.split('j'))
.map(lambda row:((row[2],row[3]),float(row[7])))
.reduceByKey(add)
)
1
result=brandmfgrcount.join(brandmfgrsum).sortBy(lambdax:x[0]).collect()
print(result)
#shortersolution
defaddtupleselementwise(_args):
returntuple(sum(x)forxinzip(_args))
result=(
sc.textFile('part.tbl')
.map(lambdaline:line.split('j'))
.map(lambda row:((row[2],row[3]),(1,float(row[7]))))
.reduceByKey(addtupleselementwise)
)
print(result.sortBy(lambdax:x[0]).collect())
```

d) How many items have 3 words in their name?

```
numnamelength3=(
sc.textFile('part.tbl')
.map(lambdaline:line.split('j'))
.map(lambda row:len(row[1].split()))
.filter(lambdalength:length==3)
.count()
)
print(numnamelength3)
```

e) How many different items does each supplier have?

```
supplier=(
sc.textFile('supplier.tbl')
.map(lambdaline:line.split('j'))
.map(lambda row:(row[0],(row[1],row[2],row[3],row[4],row[5],row[6])))
)
partsuppliers=(
sc.textFile('partsupp.tbl')
.map(lambdaline:line.split('j'))
.map(lambda row:(row[1],(row[0],row[2],row[3],row[4])))
)
supplierpsright=(
supplier.rightOuterJoin(partsuppliers)
.map(lambda x:(x[1][1][0],(x[1][0],x[0],x[1][1][1:])))
)
part=(
sc.textFile('part.tbl')
.map(lambdaline:line.split('j'))
.map(lambda row:(row[0],(row[1],row[2],row[3],row[4],row[5],row[6],row[7],row[8])))
)
supplierpart=(
supplierpsright.rightOuterJoin(part)
.map(lambda x:(x[1][0][0][0],1))
```

```
. reduceByKey ( add )
)
print(supplierpart.collect())
```
2

Note: The is .join on Pysparkling, but it works incorrectly. The correct result for our query e) can be obtained using .rightOuterJoin, but in the general case that won't work.

It is possible to express and answer all 5 queries with Spark's RDDs. For the queries c) and e), however, the result is very elaborate and unattractive. DataFrame concept from Spark offers a

better approach.

Exercise 5: PageRank with MapReduce (homework)

```
from mrsim import mr s imu l a t o r
node s = [(('a',0.2),['a','c']),
(('b',0.2),['a','d']),
(('c',0.2),['b','c','d']),
(('d',0.2),['c','e']),
(('e',0.2),[])]
# t h i s can be done i n a n o t h e r map r e d u c e s t e p s
# c o l l e c t a l l v e r t e x l a b l e s
# add a l l gr aph v e r t i c e s t o a d j a c e n c y l i s t of dead ends
d e f r emove de adends ( node s ) :
v e r t e x i d s = [v[0][0] f o r v i n node s ]
f o r v i n node s :
a d j a c e n c y l i s t = v[1]
i f n o t a d j a c e n c y l i s t : # a d j e c e n c y l i s t i s empty
a d j a c e n c y l i s t . e x t e n d ( v e r t e x i d s )
r e turn node s
d e f map ( key , v a l ) :
r e s = []
f o r l i n k i n v a l :
r e s . append ( ( l i n k , ( ' v a l ' , key [ 1 ] / l e n ( v a l ) ) ) )
r e s . append ( ( key [ 0 ] , ( ' l i n k ' , v a l ) ) )
r e turn r e s
d e f r e d u c e ( key , v a l ) :
r e s = []
p r v a l u e = 0 . 0
l i n k s = ()
f o r v i n v a l :
i f ( v [ 0 ] == ' v a l ' ) :
p r v a l u e = p r v a l u e + v [ 1 ]
e l s e :
l i n k s = v [ 1 ]
p r v a l u e = 0 . 8 _ p r v a l u e + ( 0 . 2 ) / l e n ( node s )
r e s . append ( ( ( key , p r v a l u e ) , l i n k s ) )
r e turn r e s
i n t e r me d i a t e R e s u l t = r emove de adends ( node s )
# f i r s t 10 i t e r a t i o n
f o r i i n r a n g e ( 1 , 1 1 ) :
p r i n t ( ' I t e r a t i o n ' + s t r ( i ) + ' : ' )
i n t e r me d i a t e R e s u l t = mr s imu l a t o r ( i n t e r me d i a t e Re s u l t , map , r e d u c e )
p r i n t ( i n t e r me d i a t e R e s u l t )
```

Counting Triangles

```
importitertools
#tests,whethervertexxislessthanvertexy(x<y)ingraph;
#<definesstricttotalorderonvertices
defvertexlt(x,y,graph):
degx=len(graph[x])#becausealledgesarebidirectional
degy=len(graph[y])
ifdegx<degyordegx==degyandx<y:
returnTrue
returnFalse
defcounttriangles(graph):
numtriangles=0
forv,adjacencylistingraph.items():#visvertexid
#generatorexpressioninsteadoflistcomprehension
neighborsofgreaterdeg=(nforninadjacencylistifvertexlt(v,n,graph))
foru,winitertools.combinations(neighborsofgreaterdeg,2):
ifwingraph[u]:#becausealledgesarebidirectional
print('Foundtriangle:'+v+u+w)
numtriangles+=1
returnnumtriangles
graph=f'a':['b','c','d','f'],
'b':['a','c','e'],
'c':['a','b','d','f','g'],
'd':['a','c','g'],
'e':['b','f'],
'f':['a','c','e'],
'g':['c','d']g
print(counttriangles(graph))
3
```

Pandas (Blatt 1 aufgabe 3)

a)

```
import pandas as pd
print('Find 25 Suppliers with lowest account balance.')

# csv laden und Spaltennamen vergeben:
supplier =
pd.read_csv('C:/Dateien/Master/Skripte/2_Semester/Analyzing_Massive_Data_sets/
Exercises/Sheet01/supplier.tbl', sep='|',

names=['s_Id','s_Name','Address','Nationkey','Phone','Acctbal','s_Comment','Du
mmy'])

supplier.drop(supplier.columns[-1], axis=1, inplace=True)

top_25 = supplier.nsmallest(25, 'Acctbal')[['s_Name','Acctbal']]
#top_25 = (supplier.sort_values(by = 'Acctbal')[['s_Name', 'Acctbal']]).head

print(top_25)
```

b)

```
import pandas as pd
print('Wieviele Supps haben positive Balance?')

# csv laden und Spaltennamen vergeben:
supplier =
pd.read_csv('C:/Dateien/Master/Skripte/2_Semester/Analyzing_Massive_Data_sets/
Exercises/Sheet01/supplier.tbl', sep='|',

names=['s_Id','s_Name','Address','Nationkey','Phone','Acctbal','s_Comment','Du
mmy'])

supplier.drop(supplier.columns[-1], axis=1, inplace=True)

supplier_pos_bal = supplier[supplier['Acctbal'] > 0.0][['s_Name']].count()

print(supplier_pos_bal)

supplier_pos_bal = supplier[supplier['Acctbal'] > 0.0]

print(supplier_pos_bal.shape[0])
```

c)

```
import pandas as pd
print('Find out all brands produced by the same manufacturer and calculate the
items number and thte total sales price for each')

# csv laden und Spaltennamen vergeben:
part =
pd.read_csv('C:/Dateien/Master/Skripte/2_Semester/Analyzing_Massive_Data_sets/
Exercises/Sheet01/part.tbl', sep='|',

names=['p_Id','p_Name','Mfgr','Brand','Type','Size','Container','Retailprice',
'p_Comment','Dummy'])


part.drop(part.columns[-1], axis=1, inplace=True)

brand_manufacturer = part.groupby(['Mfgr','Brand']).agg({'p_Name':
'count','Retailprice': 'sum'})

print(brand_manufacturer)
```

d)

```
import numpy as np
import pandas as pd
print('How many items have 3 words in their name?')
```

```
# csv laden und Spaltennamen vergeben:
part =
pd.read_csv('C:/Dateien/Master/Skripte/2_Semester/Analyzing_Massive_Data_sets/
Exercises/Sheet01/part.tbl', sep='|',

names=['p_Id','p_Name','Mfgr','Brand','Type','Size','Container','Retailprice',
'p_Comment','Dummy'])


part.drop(part.columns[-1], axis=1, inplace=True)

partNameSeries = part['p_Name']

print(partNameSeries[(partNameSeries.str.split()).map(len) ==
5].aggregate('count'))

#print(np.sum(partNameSeries.str.split().map(len) == 3))
```

e)

```
import pandas as pd
print('How many items have 3 words in their name?')

# csv laden und Spaltennamen vergeben:
supplier =
pd.read_csv('C:/Dateien/Master/Skripte/2_Semester/Analyzing_Massive_Data_sets/
Exercises/Sheet01/supplier.tbl', sep='|',

names=['s_Id','s_Name','Address','Nationkey','Phone','Acctbal','s_Comment','Du
mmy'])

supplier.drop(supplier.columns[-1], axis=1, inplace=True)

part =
pd.read_csv('C:/Dateien/Master/Skripte/2_Semester/Analyzing_Massive_Data_sets/
Exercises/Sheet01/part.tbl', sep='|',

names=['p_Id','p_Name','Mfgr','Brand','Type','Size','Container','Retailprice',
'p_Comment','Dummy'])

part.drop(part.columns[-1], axis=1, inplace=True)

partsupp =
pd.read_csv('C:/Dateien/Master/Skripte/2_Semester/Analyzing_Massive_Data_sets/
Exercises/Sheet01/partsupp.tbl', sep='|',

names=['p_FK','s_FK','Availqty','Supplycost','Comment_partsupp','Dummy'])

partsupp.drop(supplier.columns[-1], axis=1, inplace=True)
```

```
supplierPS = pd.merge(supplier, partsupp, left_on = 's_Id', right_on = 's_FK')
supplierPart = pd.merge(supplierPS, part, left_on = 'p_FK', right_on = 'p_ID')

print(supplierPart.groupby('s_Name')['p_Name'].count())
```