Project: PageRank on GCP

Jisen Fang (19673)

Table of contents

1. Key Technologies
2. About the Question
3. Create an input.txt
4. Create Bucket and Cluster on GCP
5. PySpark Code
6. PySpark Execution
7. PySpark Result
8. PageRank + Scala + GCP
9. Scala Code
10. Execution
11. Result

Key Technologies

GCP: Google Cloud Platform is one of the major Cloud Computing Platforms. It consists of a set of physical assets, such as computers and hard disk drives, and virtual resources, such as virtual machines (VMs), that are contained in Google's data centers around the globe.

PySpark: PySpark is the Python API for Apache Spark. Python is an interpreted, object-oriented, high-level programming language along with dynamic typing and dynamic binding.

Scala: Scala combines object-oriented and functional programming in one concise, high-level language.

PageRank: PageRank is a way of measuring the importance of website pages by counting the number and quality of links to the page, developed by Google in 1997.

About the Question

Base on the relations, A's PageRank is

$$PR(A) = (1-d) + d * (PR(C) / 1)$$

B’s PageRank:

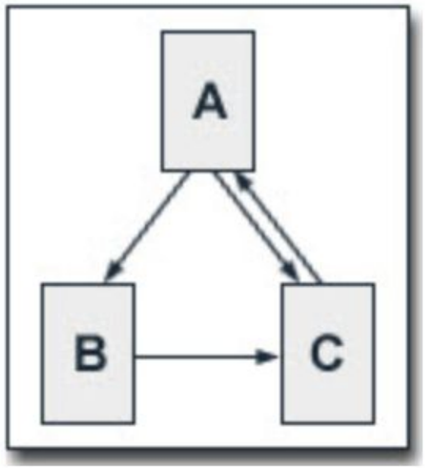
$$PR(B) = (1-d) + d * (PR(A) / 2)$$

C’s PageRank:

$$PR(C) = (1-d) + d * (PR(A) / 2 + PR(B)/1)$$

The initial PageRank value for each webpage is 1, and the damping factor is 0.85

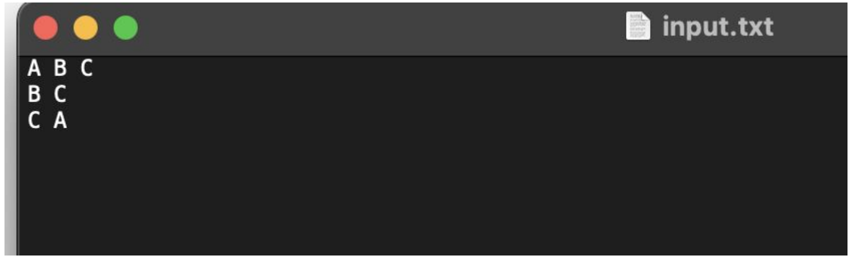
- the relation of the webpages is:



Create an input.txt

I use an adjacency list to represent the graph with web page connections.

In the question:
Web page A is connected to pages B and C.
Web page B is connected to page C.
Web page C is connected to pages A

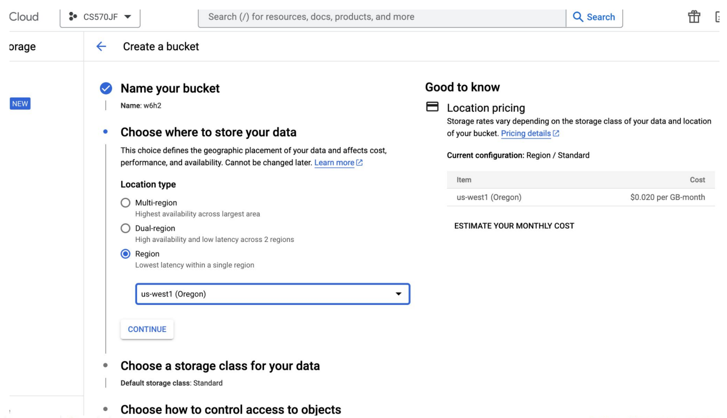


```
A B C
B C
C A
```

Create Bucket and Cluster on GCP

Create bucket and upload the input.txt.

Create Dataproc cluster in the same region as bucket.


Cloud
CS570JF
Search (/) for resources, docs, products, and more
Search
orage
NEW
ESTIMATE YOUR MONTHLY COST

Create a bucket

Name your bucket

Name: w6h2

Choose where to store your data

This choice defines the geographic placement of your data and affects cost, performance, and availability. Cannot be changed later. Learn more

Location type

- Multi-region – Highest availability across largest area
- Dual-region – High availability and low latency across 2 regions
- [✓ Region] – Lowest latency within a single region

us-west1 (Oregon)

CONTINUE

Choose a storage class for your data

Default storage class: Standard

Choose how to control access to objects

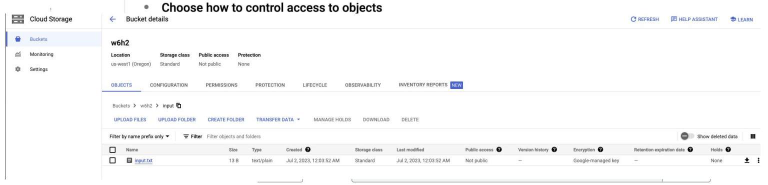
Good to know

Location pricing

Storage rates vary depending on the storage class of your data and location of your bucket. Pricing details

Current configuration: Region / Standard

Item	Cost
us-west1 (Oregon)	$0.020 per GB-month



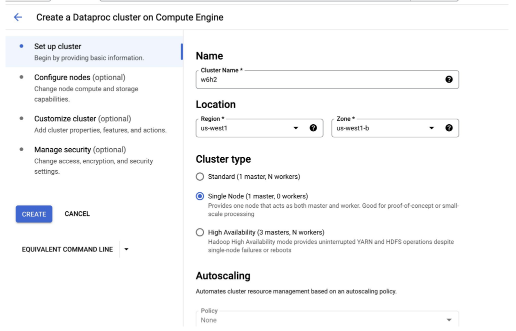
Bucket details

w6h2

Location	Storage class	Public access	Protection
us-west1 (Oregon)	Standard	Not public	None

Buckets > w6h2 > input

Name	Size	Type	Created	Storage class	Last modified	Public access	Version history	Encryption	Retention expiration date	Holds
input.txt	13 B	text/plain	Jul 2, 2023, 12:03:52 AM	Standard	Jul 2, 2023, 12:03:52 AM	Not public	–	Google-managed key	–	None



Create a Dataproc cluster on Compute Engine

- Set up cluster – Begin by providing basic information.
- Configure nodes (optional) – Change node compute and storage capabilities.
- Customize cluster (optional) – Add cluster properties, features, and actions.
- Manage security (optional) – Change access, encryption, and security settings.

Name

Cluster Name *: w6h2

Location

Region *: us-west1

Zone *: us-west1-b

Cluster type

- Standard (1 master, N workers)
- [✓ Single Node (1 master, 0 workers)] – Provides one node that acts as both master and worker. Good for proof-of-concept or small-scale processing
- High Availability (3 masters, N workers) – Hadoop High Availability mode provides uninterrupted YARN and HDFS operations despite single-node failures or reboots

Autoscaling

Automates cluster resource management based on an autoscaling policy.

Policy: None

PySpark Code

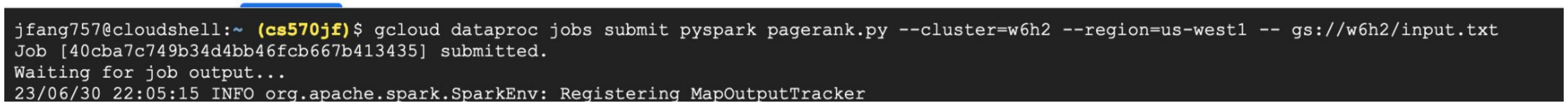
```
from pyspark.sql  import  SparkSession
import  sys

# Initialize SparkSession
spark = SparkSession.builder.appName(  "PageRank" ).getOrCreate()

# Read input data from GCS
if len(sys.argv)  !=   2:
  raise  Exception ("Exactly 1 arguments are required: <inputUri>"   )
inputUri = sys.argv[ 1]

lines = spark.read.text(inputUri).rdd.map(  lambda  r: r[0])
# Create an RDD of (destination page, list of links) pairs
def parse_links (link):
  links = link.split( " ")
  destination = links[ 0]
  source = links[ 1:]
  return  (destination, source)

links = lines.map(parse_links)
```

```
# Initialize the PageRank values 1.0 for each web page
ranks = links.map( lambda  page: (page[ 0],  1.0))

damping_factor =  0.85

# 2 iterations
for  i  in  range(2):
   contributions = links.join(ranks).flatMap(   lambda  x: [(page, x[ 1][1] /
len(x[1][0]))  for page  in  x[1][0]])
   ranks = contributions.reduceByKey(  lambda  x,  y: x + y).mapValues( lambda  rank:
(1-damping_factor) + damping_factor * rank)
   # Print each iteration result
   print(ranks.collect())

# Print
for  (page, rank)  in  ranks.collect():
   print("%s has rank:  %s." %  (page, rank))

# Stop the SparkSession
spark.stop()
```

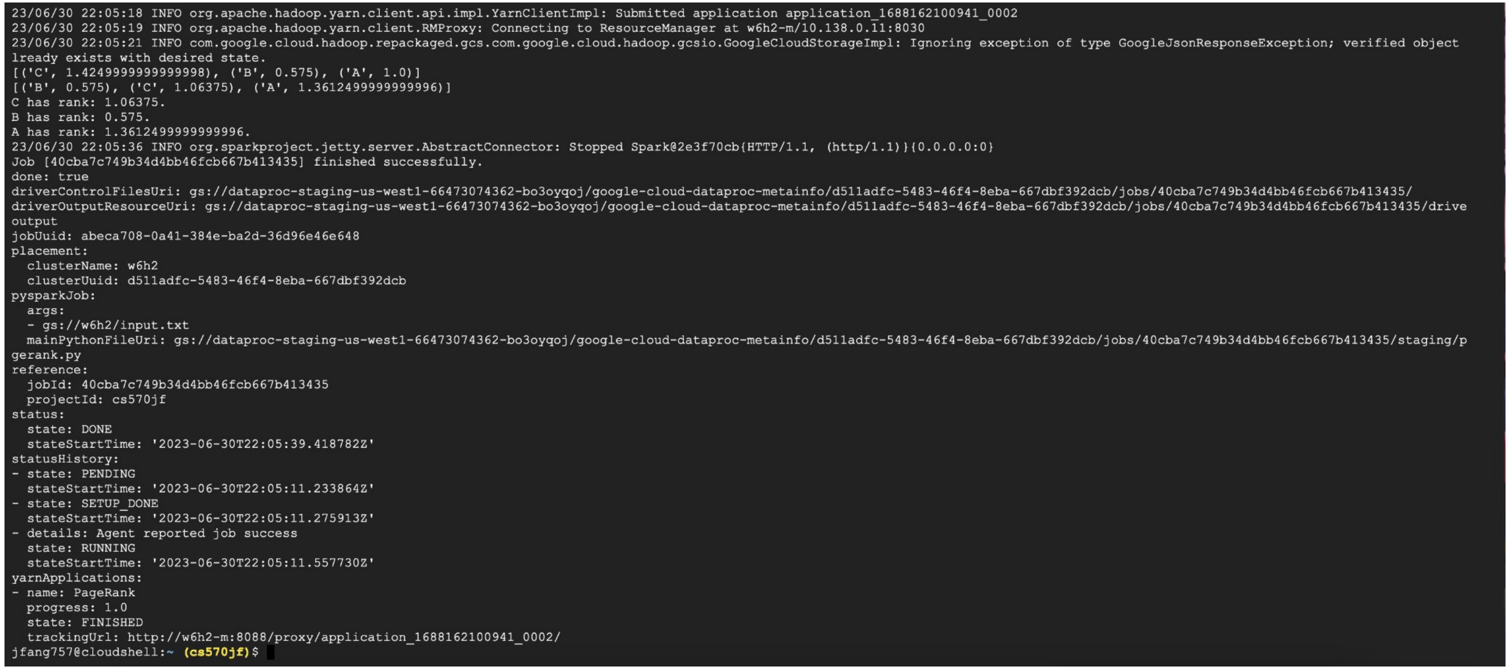
PySpark Execution

In Cloud Shell Terminal, type:

```
gcloud dataproc jobs submit pyspark pagerank.py --cluster=w6h2
--region=us-west1 -- gs://w6h2/input.txt
```



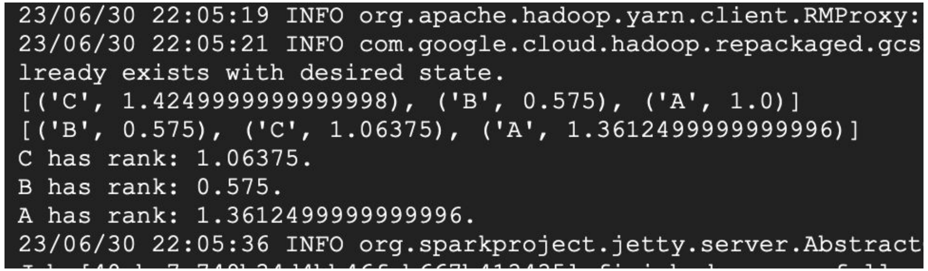
```
jfang757@cloudshell:~ (cs570jf)$ gcloud dataproc jobs submit pyspark pagerank.py --cluster=w6h2 --region=us-west1 -- gs://w6h2/input.txt
Job [40cba7c749b34d4bb46fcb667b413435] submitted.
Waiting for job output...
23/06/30 22:05:15 INFO org.apache.spark.SparkEnv: Registering MapOutputTracker
```



```
23/06/30 22:05:18 INFO org.apache.hadoop.yarn.client.api.impl.YarnClientImpl: Submitted application application_1688162100941_0002
23/06/30 22:05:19 INFO org.apache.hadoop.yarn.client.RMProxy: Connecting to ResourceManager at w6h2-m/10.138.0.11:8030
23/06/30 22:05:21 INFO com.google.cloud.hadoop.repackaged.gcs.com.google.cloud.hadoop.gcsio.GoogleCloudStorageImpl: Ignoring exception of type GoogleJsonResponseException; verified object
lready exists with desired state.
[('C', 1.4249999999999998), ('B', 0.575), ('A', 1.0)]
[('B', 0.575), ('C', 1.06375), ('A', 1.3612499999999996)]
C has rank: 1.06375.
B has rank: 0.575.
A has rank: 1.3612499999999996.
23/06/30 22:05:36 INFO org.sparkproject.jetty.server.AbstractConnector: Stopped Spark@2e3f70cb{HTTP/1.1, (http/1.1)}{0.0.0.0:0}
Job [40cba7c749b34d4bb46fcb667b413435] finished successfully.
done: true
driverControlFilesUri: gs://dataproc-staging-us-west1-66473074362-bo3oyqoj/google-cloud-dataproc-metainfo/d511adfc-5483-46f4-8eba-667dbf392dcb/jobs/40cba7c749b34d4bb46fcb667b413435/
driverOutputResourceUri: gs://dataproc-staging-us-west1-66473074362-bo3oyqoj/google-cloud-dataproc-metainfo/d511adfc-5483-46f4-8eba-667dbf392dcb/jobs/40cba7c749b34d4bb46fcb667b413435/drive
output
jobUuid: abeca708-0a41-384e-ba2d-36d96e46e648
placement:
  clusterName: w6h2
  clusterUuid: d511adfc-5483-46f4-8eba-667dbf392dcb
pysparkJob:
  args:
  - gs://w6h2/input.txt
  mainPythonFileUri: gs://dataproc-staging-us-west1-66473074362-bo3oyqoj/google-cloud-dataproc-metainfo/d511adfc-5483-46f4-8eba-667dbf392dcb/jobs/40cba7c749b34d4bb46fcb667b413435/staging/p
gerank.py
reference:
  jobId: 40cba7c749b34d4bb46fcb667b413435
  projectId: cs570jf
status:
  state: DONE
  stateStartTime: '2023-06-30T22:05:39.418782Z'
statusHistory:
- state: PENDING
  stateStartTime: '2023-06-30T22:05:11.233864Z'
- state: SETUP_DONE
  stateStartTime: '2023-06-30T22:05:11.275913Z'
- details: Agent reported job success
  state: RUNNING
  stateStartTime: '2023-06-30T22:05:11.557730Z'
yarnApplications:
- name: PageRank
  progress: 1.0
  state: FINISHED
  trackingUrl: http://w6h2-m:8088/proxy/application_1688162100941_0002/
jfang757@cloudshell:~ (cs570jf)$
```

PySpark Result

The reduceByKey() function uses double precision floating point numbers, which have a precision of 15 decimal places. The mapValues() function also uses double precision floating point numbers. So when the mapValues() function adds the contributions to A,B and C, it may round the values to 15 decimal places. In the first iteration, this rounding can cause the final PageRank value for C to be 1.42499999, slightly less than 1.425.

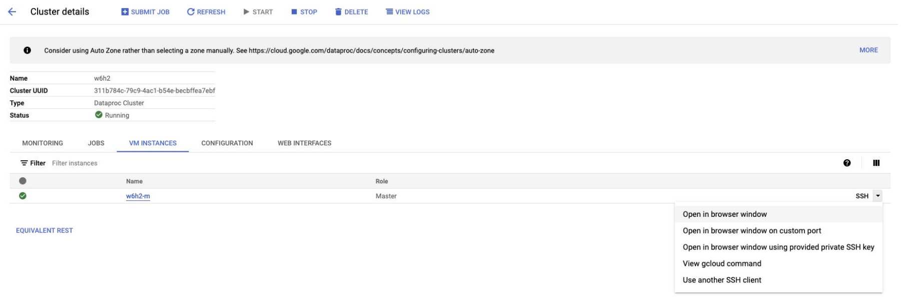


```
23/06/30 22:05:19 INFO org.apache.hadoop.yarn.client.RMProxy:
23/06/30 22:05:21 INFO com.google.cloud.hadoop.repackaged.gcs
lready exists with desired state.
[('C', 1.4249999999999998), ('B', 0.575), ('A', 1.0)]
[('B', 0.575), ('C', 1.06375), ('A', 1.3612499999999996)]
C has rank: 1.06375.
B has rank: 0.575.
A has rank: 1.3612499999999996.
23/06/30 22:05:36 INFO org.sparkproject.jetty.server.Abstract
```

PageRank + Scala + GCP

I use Spark-shell in the cluster to run Spark Scala code.

Open SSH-in-browser in the cluster and type: spark-shell



Consider using Auto Zone rather than selecting a zone manually. See https://cloud.google.com/dataproc/docs/concepts/configuring-clusters/auto-zone

Name	w6h2
Cluster UUID	311b784c-79c9-4ac1-b54e-becbffea7ebf
Type	Dataproc Cluster
Status	Running

	Name	Role	
✓	w6h2-m	Master	SSH

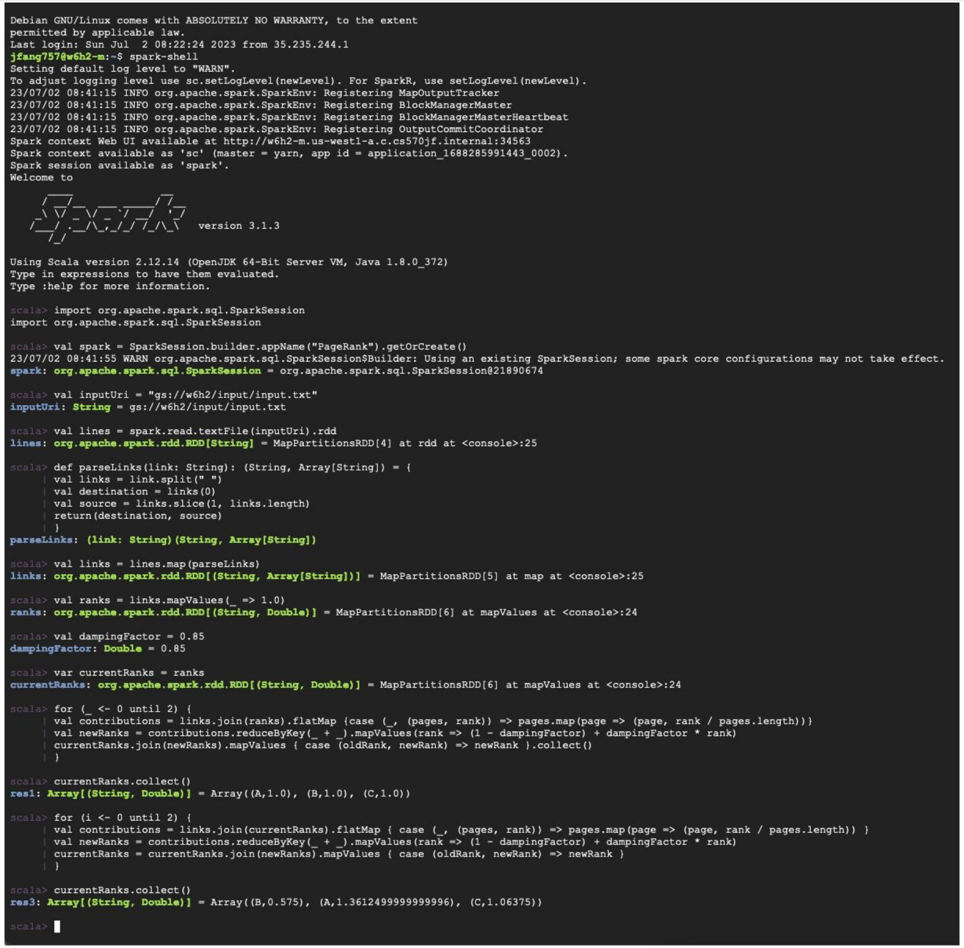
Open in browser window
Open in browser window on custom port
Open in browser window using provided private SSH key
View gcloud command
Use another SSH client
EQUIVALENT REST

Scala Code

```scala
import org.apache.spark.sql. SparkSession
// Initialize SparkSession
val spark = SparkSession .builder.appName( "PageRank" ).getOrCreate()

// Read input data from GCS
val inputUri = "gs://w6h2/input/input.txt"

val lines = spark.read.textFile(inputUri).rdd

// Create an RDD of (destination page, list of links) pairs
def parseLinks (link: String): (String, Array[String]) = {
   val links = link.split( " ")
   val destination = links(0)
   val source = links.slice(1, links.length)
   return (destination, source)
}

val links = lines.map(parseLinks)
```

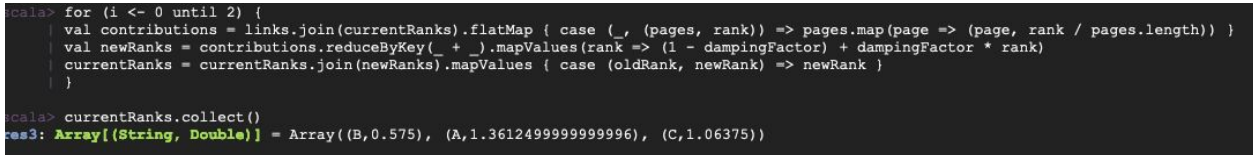
```scala
// Initialize the PageRank values 1.0 for each web page
val ranks = links.mapValues(_ => 1.0)
val dampingFactor = 0.85
// In Scala, val is immutable, and cannot reassign a new value to it.
To fix this, use a mutable variable (var) for ranks instead.
var currentRanks = ranks

for (i <- 0 until 2) {
 val contributions = links.join(currentRanks).flatMap {  case (_,
(pages, rank)) => pages.map(page => (page, rank / pages.length)) }
 val newRanks = contributions.reduceByKey(_ + _).mapValues(rank => ( 1 -
dampingFactor) + dampingFactor * rank)
 currentRanks = currentRanks.join(newRanks).mapValues {  case (oldRank,
newRank) => newRank }
}

currentRanks.collect()
```

Execution



```
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Jul  2 08:22:24 2023 from 35.235.244.1
jfang757@w6h2-m:~$ spark-shell
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
23/07/02 08:41:15 INFO org.apache.spark.SparkEnv: Registering MapOutputTracker
23/07/02 08:41:15 INFO org.apache.spark.SparkEnv: Registering BlockManagerMaster
23/07/02 08:41:15 INFO org.apache.spark.SparkEnv: Registering BlockManagerMasterHeartbeat
23/07/02 08:41:15 INFO org.apache.spark.SparkEnv: Registering OutputCommitCoordinator
Spark context Web UI available at http://w6h2-m.us-west1-a.c.cs570jf.internal:34563
Spark context available as 'sc' (master = yarn, app id = application_1688285991443_0002).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.1.3
      /_/

Using Scala version 2.12.14 (OpenJDK 64-Bit Server VM, Java 1.8.0_372)
Type in expressions to have them evaluated.
Type :help for more information.

scala> import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.SparkSession

scala> val spark = SparkSession.builder.appName("PageRank").getOrCreate()
23/07/02 08:41:55 WARN org.apache.spark.sql.SparkSession$Builder: Using an existing SparkSession; some spark core configurations may not take effect.
spark: org.apache.spark.sql.SparkSession = org.apache.spark.sql.SparkSession@21890674

scala> val inputUri = "gs://w6h2/input/input.txt"
inputUri: String = gs://w6h2/input/input.txt

scala> val lines = spark.read.textFile(inputUri).rdd
lines: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[4] at rdd at <console>:25

scala> def parseLinks(link: String): (String, Array[String]) = {
     | val links = link.split(" ")
     | val destination = links(0)
     | val source = links.slice(1, links.length)
     | return(destination, source)
     | }
parseLinks: (link: String)(String, Array[String])

scala> val links = lines.map(parseLinks)
links: org.apache.spark.rdd.RDD[(String, Array[String])] = MapPartitionsRDD[5] at map at <console>:25

scala> val ranks = links.mapValues(_ => 1.0)
ranks: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[6] at mapValues at <console>:24

scala> val dampingFactor = 0.85
dampingFactor: Double = 0.85

scala> var currentRanks = ranks
currentRanks: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[6] at mapValues at <console>:24

scala> for (_ <- 0 until 2) {
     | val contributions = links.join(ranks).flatMap {case (_, (pages, rank)) => pages.map(page => (page, rank / pages.length))}
     | val newRanks = contributions.reduceByKey(_ + _).mapValues(rank => (1 - dampingFactor) + dampingFactor * rank)
     | currentRanks.join(newRanks).mapValues { case (oldRank, newRank) => newRank }.collect()
     | }

scala> currentRanks.collect()
res1: Array[(String, Double)] = Array((A,1.0), (B,1.0), (C,1.0))

scala> for (i <- 0 until 2) {
     | val contributions = links.join(currentRanks).flatMap { case (_, (pages, rank)) => pages.map(page => (page, rank / pages.length)) }
     | val newRanks = contributions.reduceByKey(_ + _).mapValues(rank => (1 - dampingFactor) + dampingFactor * rank)
     | currentRanks = currentRanks.join(newRanks).mapValues { case (oldRank, newRank) => newRank }
     | }

scala> currentRanks.collect()
res3: Array[(String, Double)] = Array((B,0.575), (A,1.3612499999999996), (C,1.06375))

scala> 
```

Result



```
cala> for (i <- 0 until 2) {
     | val contributions = links.join(currentRanks).flatMap { case (_, (pages, rank)) => pages.map(page => (page, rank / pages.length)) }
     | val newRanks = contributions.reduceByKey(_ + _).mapValues(rank => (1 - dampingFactor) + dampingFactor * rank)
     | currentRanks = currentRanks.join(newRanks).mapValues { case (oldRank, newRank) => newRank }
     | }

cala> currentRanks.collect()
es3: Array[(String, Double)] = Array((B,0.575), (A,1.3612499999999996), (C,1.06375))
```