Week 6: Homework
Project: PageRank on GCP

Navya Kandimalla
19644

PageRank.py

```
import re
import sys
from operator import add
from pyspark.sql import SparkSession
def computeContribs(urls, rank):
    """Calculates URL contributions to the rank of other URLs."""
    num_urls = len(urls)
    for url in urls:
        yield (url, rank / num_urls)



def parseNeighbors(urls):
    """Parses a urls pair string into urls pair."""
    parts = re.split(r'\s+', urls)
    return parts[0], parts[1]



if __name__ == "__main__":
    if len(sys.argv) != 3:
        print("Usage: pagerank <file> <iterations>", file=sys.stderr)
        sys.exit(-1)

    print("WARN: This is a naive implementation of PageRank and is given as an example!\n" +
          "Please refer to PageRank implementation provided by graphx",
          file=sys.stderr)

    # Initialize the spark context.
    spark = SparkSession\
        .builder\
                    .appName("PythonPageRank")\
                    .getOrCreate()
lines = spark.read.text(sys.argv[1]).rdd.map(lambda r: r[0])

links = lines.map(lambda urls: parseNeighbors(urls)).distinct().groupByKey().cache()

ranks = links.map(lambda url_neighbors: (url_neighbors[0], 1.0))

for iteration in range(int(sys.argv[2])):
        contribs = links.join(ranks).flatMap(lambda url_urls_rank: computeContribs(url_urls_rank[1][0],
        url_urls_rank[1][1]))

        ranks = contribs.reduceByKey(add).mapValues(lambda rank: rank * 0.85 + 0.15)

for (link, rank) in ranks.collect():
        print("%s has rank: %s." % (link, rank))

spark.stop()
```

PageRank_Scala

```
val lines = sc.textFile("hdfs:///mydata/pagerank_input.txt")

val links = lines.map{ s =>val parts = s.split("\\s+")
        (parts(0), parts(1)) }.distinct().groupByKey().cache()

var ranks = links.mapValues(v => 1.0)

for (i <- 1 to 10)
{
        val contribs = links.join(ranks).values.flatMap{ case (urls, rank) =>
        val size = urls.size urls.map(url => (url, rank / size)}
        ranks = contribs.reduceByKey(_ + _).mapValues(0.15 + 0.85 * _)
}
```

```
val output = ranks.collect()
output.foreach(tup => println(tup._1 + " has rank: " + tup._2 + "."))

ctx.stop()
```