Sami Emre Erdogan
Student ID: 2019280513

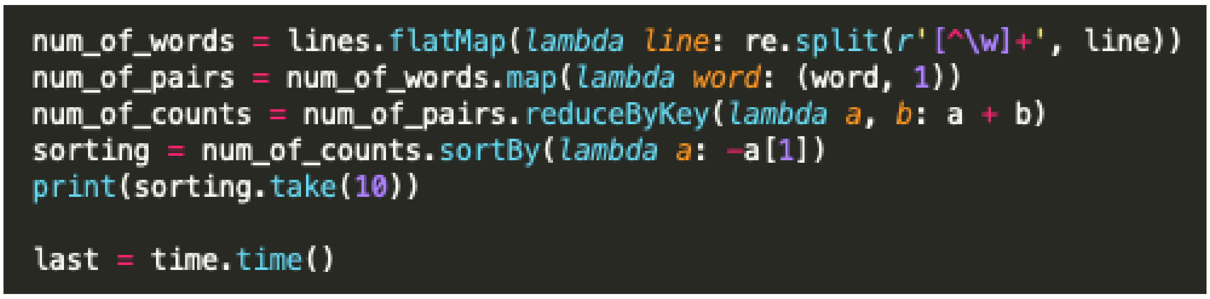
Homework 7: Spark Programming

Word Count

Output of wc.txt

```
took 0.203317 s
[('', 197060), ('the', 23455), ('I', 22225), ('and', 18715), ('to',
16433), ('of', 15830), ('a', 12851), ('you', 12236), ('my', 10840),
('in', 10074)]
Total program time: 4.02 seconds
```

The code below was researched and looked from different resources and slightly modified according to our implementation.
The official https://spark.apache.org/examples.html was a big help I just wanted to break down the code into variables so it was easier to follow other than that it was pretty much really similar apart from maybe changing **line.split(" "))-> re.split(r'[^\w]+', line))** sorting the number of counts and taking the top 10.



```
num_of_words = lines.flatMap(lambda line: re.split(r'[^\w]+', line))
num_of_pairs = num_of_words.map(lambda word: (word, 1))
num_of_counts = num_of_pairs.reduceByKey(lambda a, b: a + b)
sorting = num_of_counts.sortBy(lambda a: -a[1])
print(sorting.take(10))

last = time.time()
```

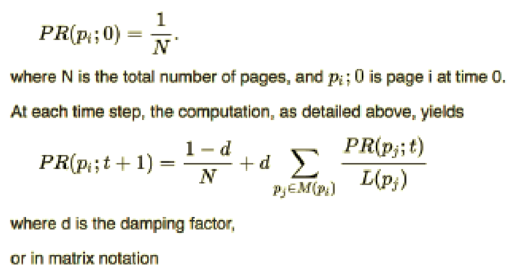
Sami Emre Erdogan
Student ID: 2019280513

Page Rank

Output of full.txt

```
took 0.148354 s
[(263, 0.0020202911815182184), (537, 0.0019433415714531497), (965,
0.0019254478071662631), (243, 0.0018526340162417 31), (285,
0.0018273721700645144)]
Total program time: 23.11 seconds
```

In this PagePank implementation I have also looked at several links and took pieces from from different sources to implement this code. The main difference was that I just used the formula given within the assignment to make it more relevant to the assignment.



$$PR(p_i;0) = \frac{1}{N}.$$

where N is the total number of pages, and $p_i;0$ is page i at time 0.

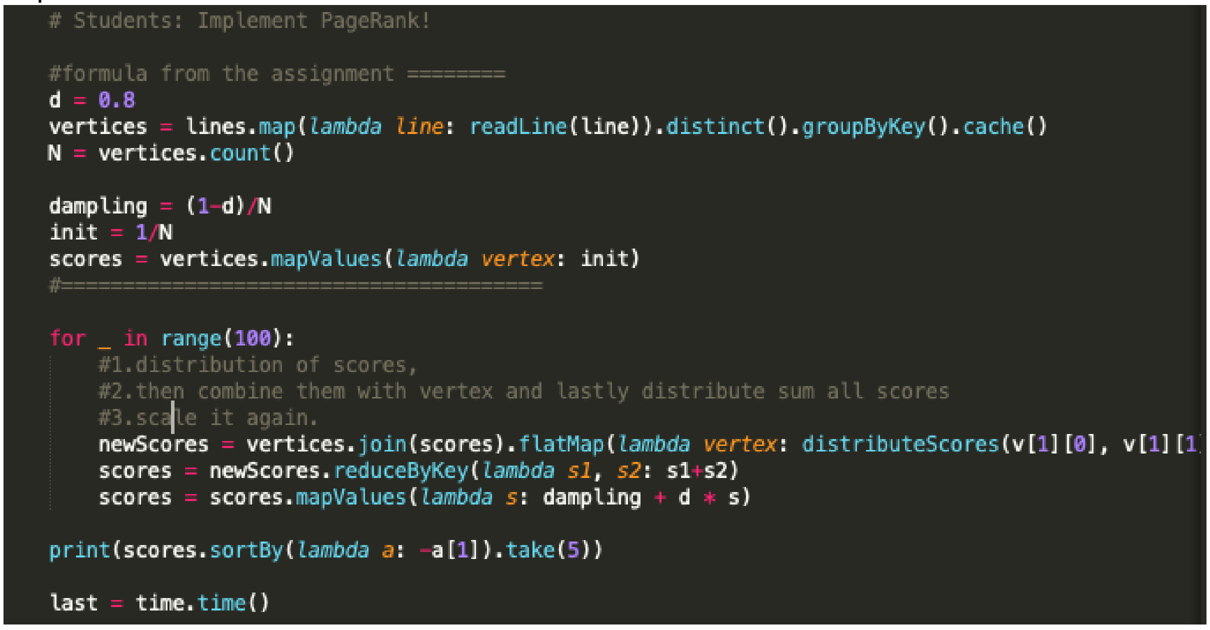
At each time step, the computation, as detailed above, yields

$$PR(p_i;t+1) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j;t)}{L(p_j)}$$

where d is the damping factor,

or in matrix notation

The implementation screenshot is shown below and it has been commented out for relevant explanation: Last part is sort and take the first 5. Similar idea to word count implementation.



```
# Students: Implement PageRank!

#formula from the assignment ========
d = 0.8
vertices = lines.map(lambda line: readLine(line)).distinct().groupByKey().cache()
N = vertices.count()

dampling = (1-d)/N
init = 1/N
scores = vertices.mapValues(lambda vertex: init)
#======================================

for _ in range(100):
    #1.distribution of scores,
    #2.then combine them with vertex and lastly distribute sum all scores
    #3.scale it again.
    newScores = vertices.join(scores).flatMap(lambda vertex: distributeScores(v[1][0], v[1][1
    scores = newScores.reduceByKey(lambda s1, s2: s1+s2)
    scores = scores.mapValues(lambda s: dampling + d * s)

print(scores.sortBy(lambda a: -a[1]).take(5))

last = time.time()
```