Big Data Assignment 3

Topic: Multiple-Source Personalized Page Rank.

Section: BS (DS) M

Group Members:

Dawood Tanvir 21i-1665

Ume Hani 21i- 1715

Laiba Batool 21i-1781

PERSONALIZED PAGE RANK:

In this assignment we learn about personalized page rank. Personalized page rank is the extension of page rank algorithm that incorporates a personalized bias towards a specific set of nodes. In personalized page rank, the importance of a node is measured based on its proximity to a set of source nodes that are considered important or relevant. The algorithm assigns a higher probability of reaching the source nodes and uses this bias to calculate the importance scores of all nodes in the network. We learned that personalized page rank can provide more accurate result for specific use cases. Finally, we concluded that personalized page rank can be used for various applications, such as identifying the most relevant search results for a particular user.

In this assignment we write our code in three files i.e.; input.txt, mapper.py, reducer.py.

INPUT.txt:

Input file is same file which you provided in last assignment.

MAPPER.py:

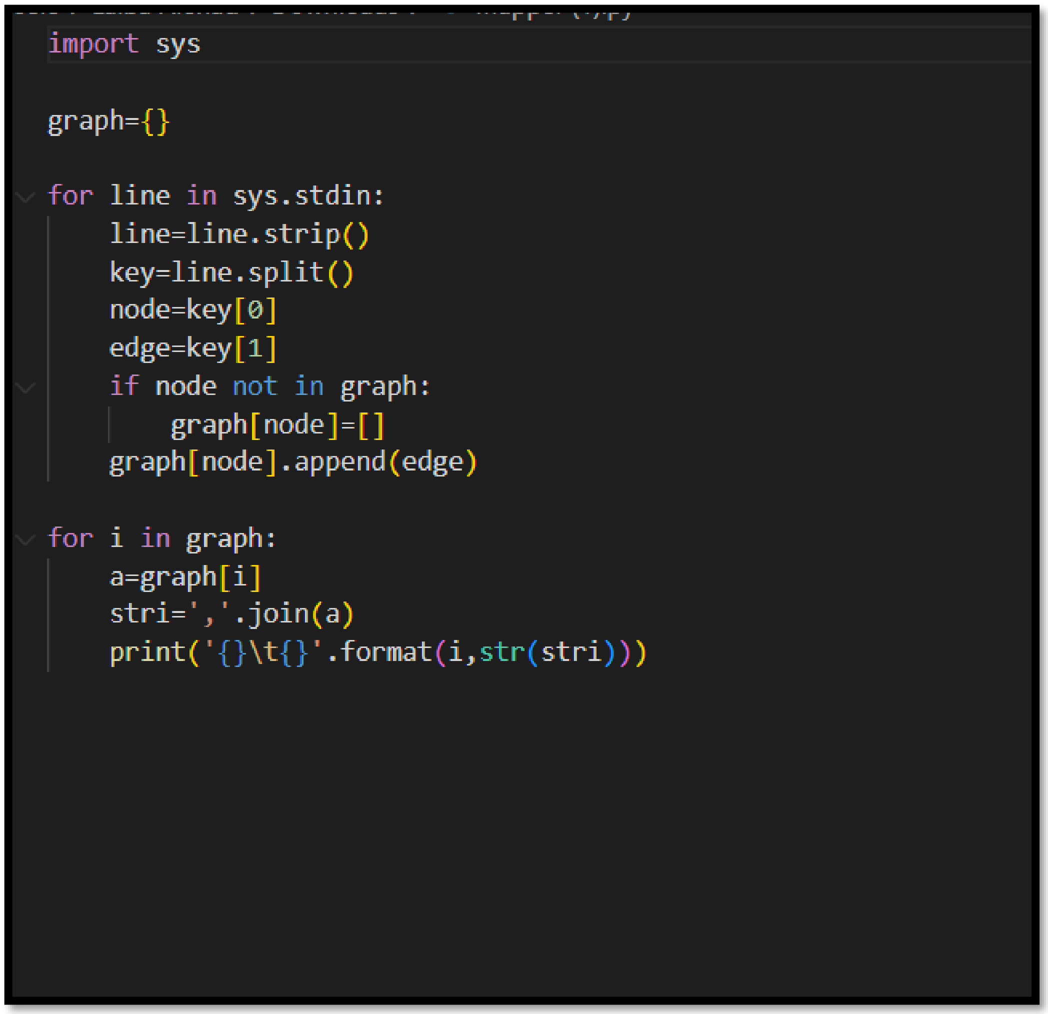
First, we make mapper file in which we write code to read input.txt file.

We make dictionary named as graph, in dictionary key is node and value are set of friend node.

Then we print all nodes with its friend node.

This output works as an input for reducer.

MAPPER CODE:


```
import sys

graph={}

for line in sys.stdin:
    line=line.strip()
    key=line.split()
    node=key[0]
    edge=key[1]
    if node not in graph:
        graph[node]=[]
    graph[node].append(edge)

for i in graph:
    a=graph[i]
    stri=','.join(a)
    print('{}\t{}'.format(i,str(stri)))
```

REDUCER.py:

Reducer is a crucial part of the MapReduce framework and is responsible for combining and aggregating the output of the mappers to produce the final output.

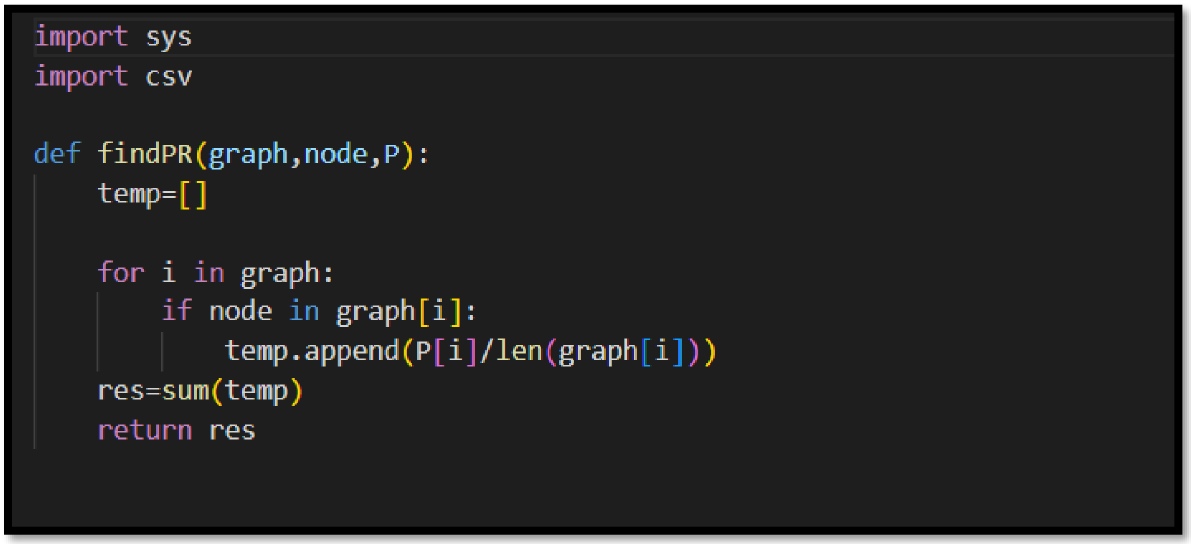
In reducer of personalized page rank we read the output of mapper for further processing.

Then we take three source nodes i.e. 1665, 1715, 1781.

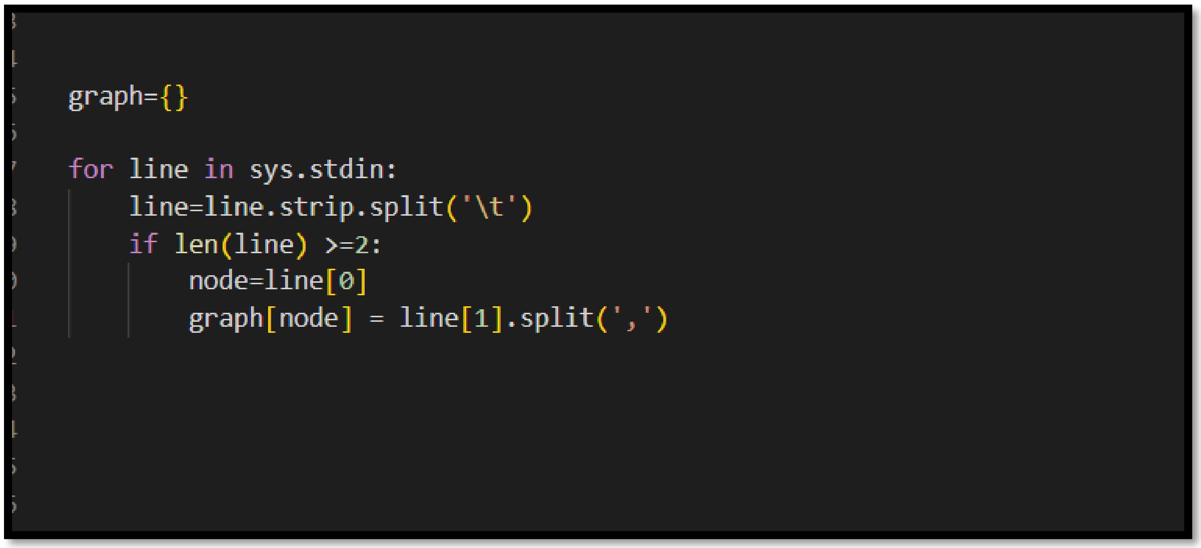
We make page rank dictionary, named as PR, in which we initialized the page rank of nodes. If node is a source node, then page rank is 1/3 else 0. This means that the source nodes are given a higher weight than other nodes.

Then we write function named as findPR with three parameters graph, node, and initialize page rank. This function finds the page rank of nodes and is biased towards the source nodes, resulting in a higher probability of ending up at a source node compared to traditional PageRank.

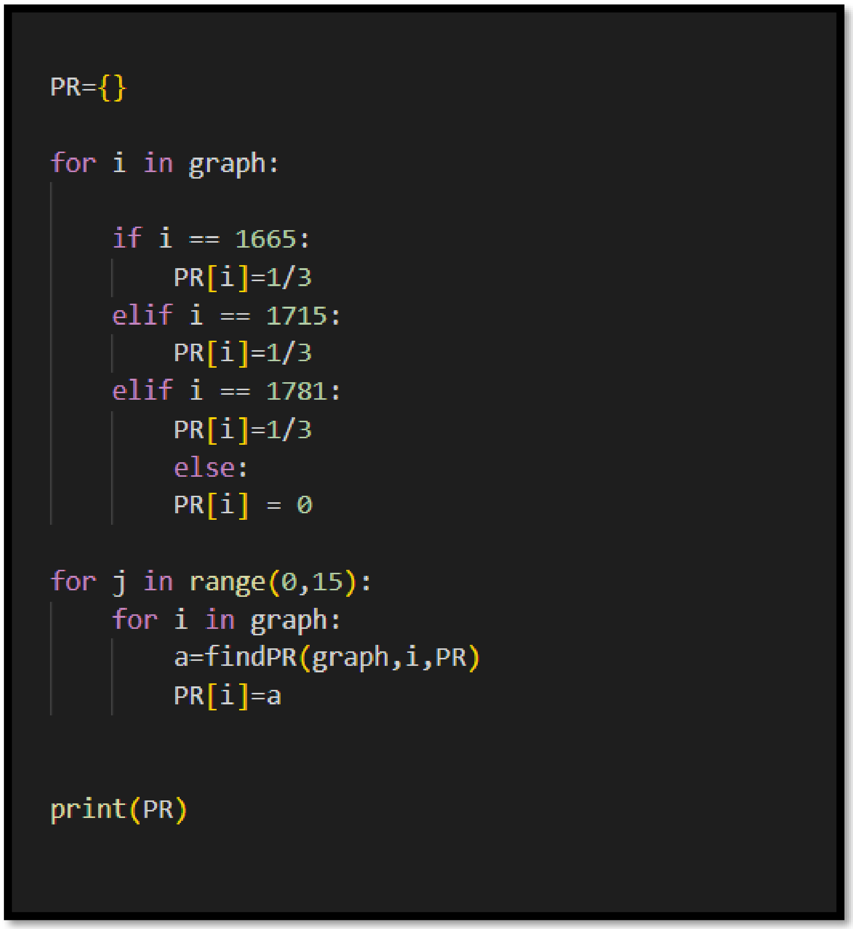
REDUCER CODE:



```
import sys
import csv

def findPR(graph,node,P):
    temp=[]

    for i in graph:
        if node in graph[i]:
            temp.append(P[i]/len(graph[i]))
    res=sum(temp)
    return res
```



```
graph={}

for line in sys.stdin:
    line=line.strip.split('\t')
    if len(line) >=2:
        node=line[0]
        graph[node] = line[1].split(',')
```



```
PR={}

for i in graph:

    if i == 1665:
        PR[i]=1/3
    elif i == 1715:
        PR[i]=1/3
    elif i == 1781:
        PR[i]=1/3
        else:
        PR[i] = 0

for j in range(0,15):
    for i in graph:
        a=findPR(graph,i,PR)
        PR[i]=a


print(PR)
```

Hadoop multi node setting Error:

We did our best to set up a multi-node configuration, but we encountered an authentication key error.

EACH GROUP MEMBER'S WORK:

CODE: Ume Hani and Dawood

REPORT: Laiba Batool