Experiment-02

Name: Atharva Paliwal

Roll No: 40

Date: 11/08/2021

Aim: Write a program to:

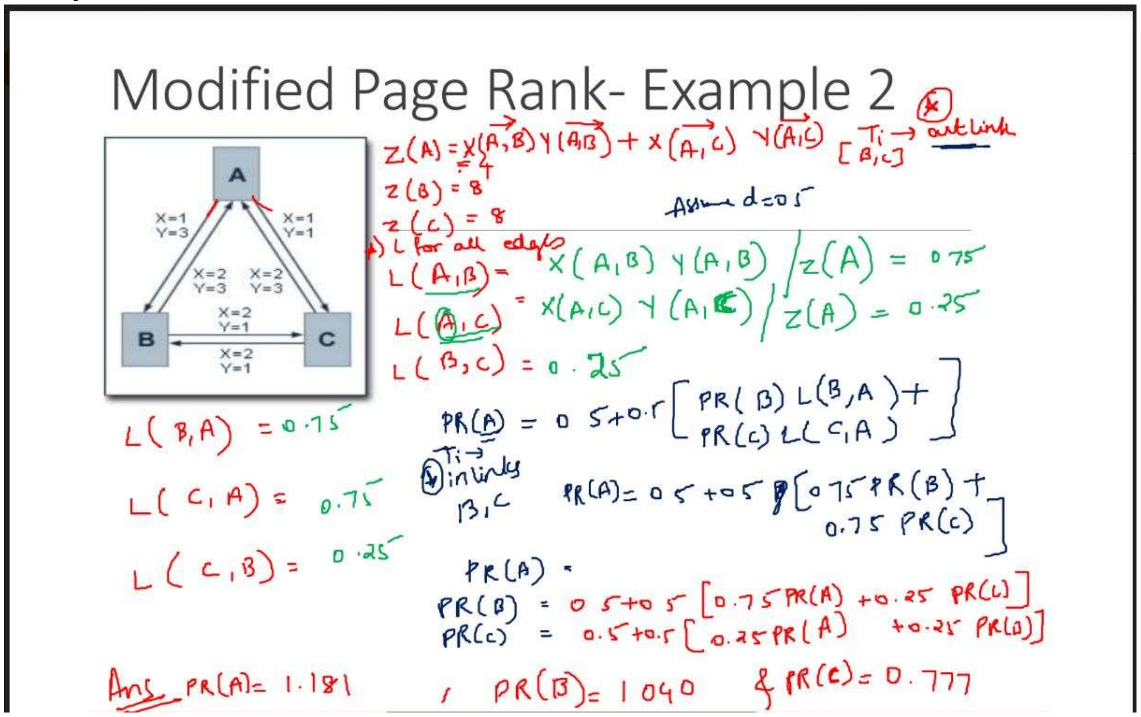
A. Compute the Page Rank of all the webpages in an undirected web graph using the modified page rank method.

B. Integrate the page rank code with inverted index (practical 1) and now the results must be ranked according to page rank instead of term frequency.

Details:

- Accept graph as input from the user. By accepting names of web pages and one by one entering the edges.
- Ask the user to enter the information about the link location (upper half / lower half) and if it is emphasized. Accordingly, find the value of X and Y (weights) for each edge.
- Write function to create a weighted graph and add edges in a graph.
- Decide the data structure to be used for implementing graph. Also display graph if possible. [networkx can be used in python]
- Using the formulas of modified page rank find the equation for each web page in the web graph.
- Compute the page rank using the iterative approach for page rank computation.
- Display the page ranks and web pages ranked according to the page rank.
- Integrate the page rank code with inverted index (practical 1) and now the results must be ranked according to page rank instead of term frequency.
- Consider only 4 to 5 web pages in the web graph and text files for inverted index creation.

Theory:



Modified Page Rank- Example 2

$\boxed{*}$

$Z(A) = X(\overrightarrow{A,B})\,Y(\overrightarrow{A,B}) + X(\overrightarrow{A,C})\,Y(\overrightarrow{A,C})$ $T_i \rightarrow$ outlink $[B,C]$

$= 4$

$Z(B) = 8$

$Z(C) = 8$

Assume $d = 0.5$

1) L for all edges

$L(A,B) = X(A,B)\,Y(A,B) / Z(A) = 0.75$

$L(A,C) = X(A,C)\,Y(A,C) / Z(A) = 0.25$

$L(B,C) = 0.25$

$L(B,A) = 0.75$

$L(C,A) = 0.75$

$L(C,B) = 0.25$

$PR(A) = 0.5 + 0.5 \left[PR(B)\,L(B,A) + PR(C)\,L(C,A) \right]$

$T_i \rightarrow$ inlinks B, C

$PR(A) = 0.5 + 0.5 \left[0.75\,PR(B) + 0.75\,PR(C) \right]$

$PR(A) =$

$PR(B) = 0.5 + 0.5 \left[0.75\,PR(A) + 0.25\,PR(C) \right]$

$PR(C) = 0.5 + 0.5 \left[0.25\,PR(A) + 0.25\,PR(B) \right]$

Ans $PR(A) = 1.181$, $PR(B) = 1.040$ & $PR(C) = 0.777$

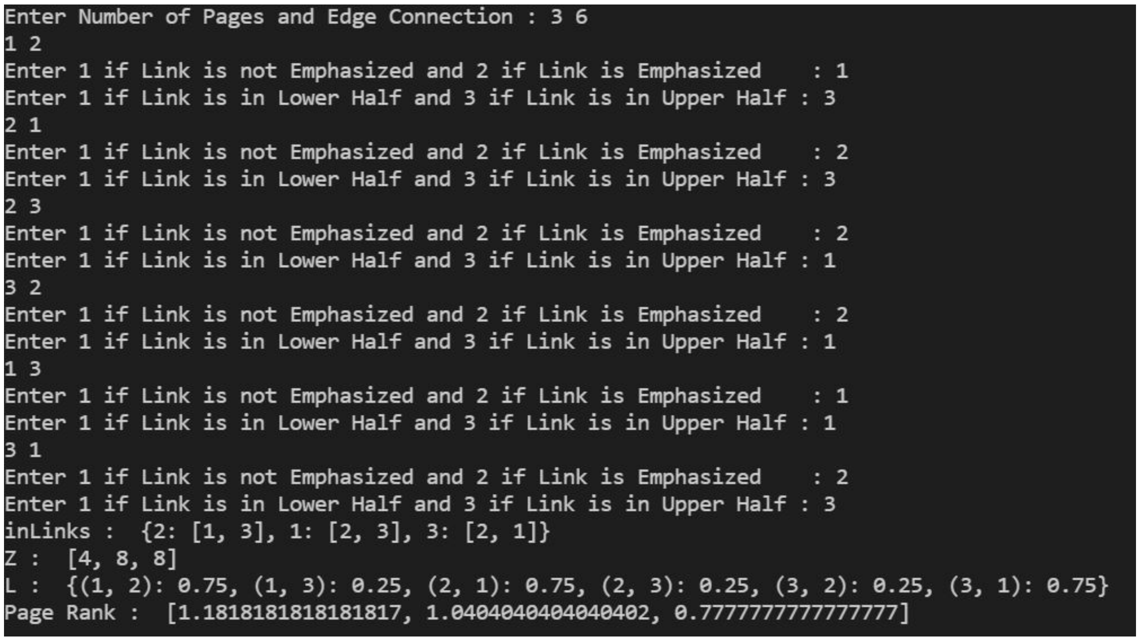
Program:

```
#PART-1

#Part 1.0 - GRAPH INPUT

n,e=map(int,input("Enter Number of Pages and Edge Connection : ").split())

adj=[[] for i in range(n+1)]

inLink = {}

for _ in range(e):

    u,v=map(int,input().split())

    if (v in inLink):

        inLink[v].append(u)

    else:

        inLink[v] = [u]

    x=int(input("Enter 1 if Link is not Emphasized and 2 if Link is Emphasized : "))

    y=int(input("Enter 1 if Link is in Lower Half and 3 if Link is in Upper Half : "))

    w=[x,y]

    adj[u].append([v,w])


# print(adj[1])
```

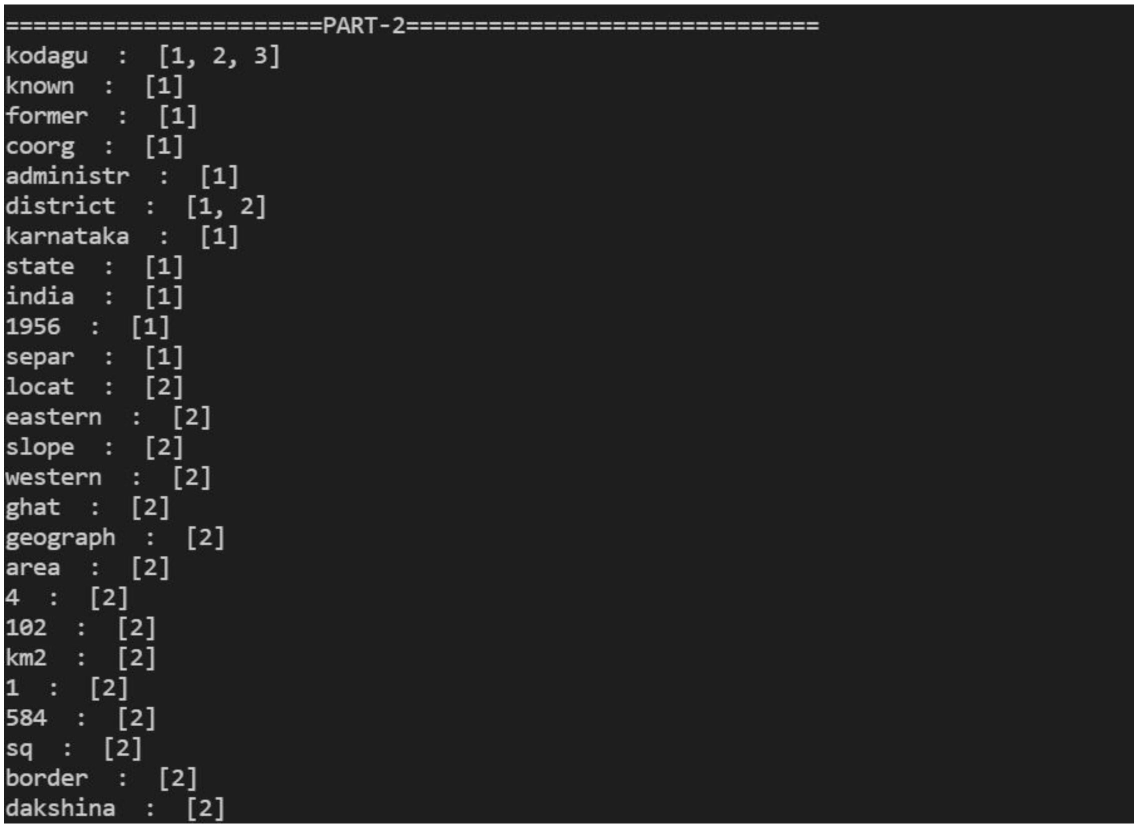
```
#Displaying incomming Links of Nodes
print('inLinks : ',inLink)

#Part 1.1 - VALUE OF Z
# Z = Sum(X(A,outLinks)*Y(A,outLinks))
z=[]
for i in range(1,n+1):
    curr_z=0
    for j in adj[i]:
        curr_z+=j[1][0]*j[1][1]
    z.append(curr_z)
#Displaying Z values for all nodes
print("Z : ", z)

#Part1.2 - VALUE OF L
# L(A,B) = X(A,B)*Y(A,B) / Z(A)
L={}
for i in range(1,n+1):
    for j in adj[i]:
        L[(i,j[0])]=j[1][0]*j[1][1]/z[i-1]
#Displaying L values for all edges
print("L : ", L)

#Part 1.3 - VALUE OF PAGE RANK
#d=0.5
# PR(A) = 0.5  + 0.5*(PR(inLinks)*L(inLinks, A))

pageRank = [0]*(n+1)
tempRank = [1]*(n+1)
while pageRank != tempRank:
    tempRank = pageRank[:]
    for i in range(1, n+1):
        temp = 0
        for j in inLink[i]:
            temp = temp + pageRank[j]*L[(j,i)]
        pageRank[i] = 0.5 + 0.5*temp
print("Page Rank : ", pageRank[1:])
print()
print()

###################################################################

print("======================PART-2==============================")
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer

ps = PorterStemmer()

def readfile(filename,dict,cnt):
    #reading filename
    file = open(filename, encoding='utf8')
    read = file.read() #extracting the file contents
    file.seek(0)  #pointer from where the data has to be written/read
    line = 1 #line number
    for word in read: #extracting each word from the file contents
        if word == '\n': #if new line occurs increment the line
            line += 1
    array = []
    for i in range(line):
        array.append(file.readline()) #appending every line in an array

    #1.REMOVE PUNCTUATIONS
    punc = '''!()-[]{};:'"\, <>./?@#$%^&*_~''' #string of punctuations
    #checking whether there is any punctuation present,if any then replacing it
with a space
    for ele in read:
        if ele in punc:
            read = read.replace(ele, " ")
    read=read.lower() #doing all elements lowercase


    #2.REMOVING STOPWORDS
    for i in range(1):
        text_tokens = word_tokenize(read)  #tokenizing every word ex -> form
becomes 'form'


    tokens_without_sw = [word for word in text_tokens if not word in
stopwords.words()] #list of tokenized words without stopwords


    #3. STORING IN DICTIONARY
    for i in range(line):
        check = array[i].lower()
        for w in tokens_without_sw:
            # print(dict_for_cnt)
            if w in check:
                item=ps.stem(w)     #This line gives us the root word for the
current word
                if item not in dict:
                    dict[item] = [] #creating a list for storing count if the word
is not present in dict


                if item in dict:
                    dict[item].append(cnt) #appending the file number where the
word is present
                    dict[item]=list(set(dict[item]))#using set so that if any word
is present multiple times in a file
                                                    #then it is not written
multiple times


#DRIVER CODE
dict={} #dictionary for storing the file numbers
for i in range(1,4):
    readfile('text'+str(i)+'.txt',dict,i) #reading each file once
```

```
#Displaying all the words along with there occurence in which file
for word,list_of_files in dict.items():
    print(word,' : ',list_of_files)



query = input("Enter String to be searched : ")
qword = query.split()
qwords=[]
for q in qword:
    qroot=ps.stem(q) #getting the root word for the current word
    if qroot not in qwords:
        qwords.append(qroot)  #appending the root words



#Displaying in which file the string is present
word_list=[]  #list for displaying part 3
for q in qword:
    word=ps.stem(q) #taking out the root word for the current word
    for doc_id in dict[word]:
        word_list.append((pageRank[doc_id],doc_id,q)) #appending for every doc_id
the count the word appears in the doc

word_list.sort(reverse=True)
for word in word_list:
    print(word[2],'appears in Document ',word[1],'with Page Rank =>',word[0])
```

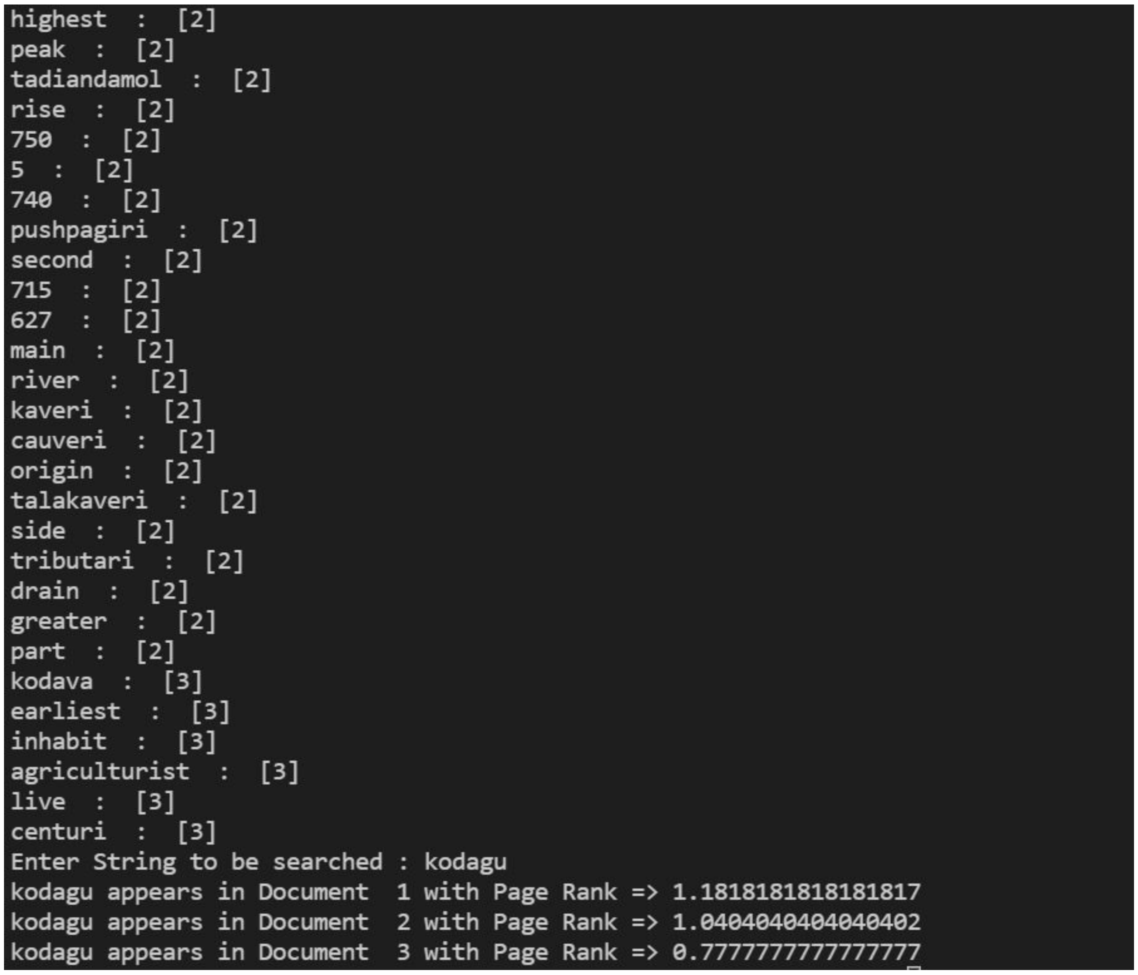
Output:



```
Enter Number of Pages and Edge Connection : 3 6
1 2
Enter 1 if Link is not Emphasized and 2 if Link is Emphasized    : 1
Enter 1 if Link is in Lower Half and 3 if Link is in Upper Half : 3
2 1
Enter 1 if Link is not Emphasized and 2 if Link is Emphasized    : 2
Enter 1 if Link is in Lower Half and 3 if Link is in Upper Half : 3
2 3
Enter 1 if Link is not Emphasized and 2 if Link is Emphasized    : 2
Enter 1 if Link is in Lower Half and 3 if Link is in Upper Half : 1
3 2
Enter 1 if Link is not Emphasized and 2 if Link is Emphasized    : 2
Enter 1 if Link is in Lower Half and 3 if Link is in Upper Half : 1
1 3
Enter 1 if Link is not Emphasized and 2 if Link is Emphasized    : 1
Enter 1 if Link is in Lower Half and 3 if Link is in Upper Half : 1
3 1
Enter 1 if Link is not Emphasized and 2 if Link is Emphasized    : 2
Enter 1 if Link is in Lower Half and 3 if Link is in Upper Half : 3
inLinks :  {2: [1, 3], 1: [2, 3], 3: [2, 1]}
Z :  [4, 8, 8]
L :  {(1, 2): 0.75, (1, 3): 0.25, (2, 1): 0.75, (2, 3): 0.25, (3, 2): 0.25, (3, 1): 0.75}
Page Rank :  [1.1818181818181817, 1.0404040404040402, 0.7777777777777777]
```



```
=======================PART-2===============================
kodagu  :  [1, 2, 3]
known  :  [1]
former  :  [1]
coorg  :  [1]
administr  :  [1]
district  :  [1, 2]
karnataka  :  [1]
state  :  [1]
india  :  [1]
1956  :  [1]
separ  :  [1]
locat  :  [2]
eastern  :  [2]
slope  :  [2]
western  :  [2]
ghat  :  [2]
geograph  :  [2]
area  :  [2]
4  :  [2]
102  :  [2]
km2  :  [2]
1  :  [2]
584  :  [2]
sq  :  [2]
border  :  [2]
dakshina  :  [2]
```



```
highest  :  [2]
peak  :  [2]
tadiandamol  :  [2]
rise  :  [2]
750  :  [2]
5  :  [2]
740  :  [2]
pushpagiri  :  [2]
second  :  [2]
715  :  [2]
627  :  [2]
main  :  [2]
river  :  [2]
kaveri  :  [2]
cauveri  :  [2]
origin  :  [2]
talakaveri  :  [2]
side  :  [2]
tributari  :  [2]
drain  :  [2]
greater  :  [2]
part  :  [2]
kodava  :  [3]
earliest  :  [3]
inhabit  :  [3]
agriculturist  :  [3]
live  :  [3]
centuri  :  [3]
Enter String to be searched : kodagu
kodagu appears in Document  1 with Page Rank => 1.1818181818181817
kodagu appears in Document  2 with Page Rank => 1.0404040404040402
kodagu appears in Document  3 with Page Rank => 0.7777777777777777
```

END