S.I.W.S

N.R SWAMY COLLEGE OF COMMERCE AND ECONOMICS AND SMT.THIRUMALAI COLLEGE OF SCIENCE.

INFORMATION RETRIEVAL

JAISWAR ROHIT MANOJ
T.Y.Bsc.C.S.
ROLL NO:39041

Year 2022-2023

S.I.W.S
N.R SWAMY COLLEGE OF COMMERCE AND ECONOMICS
AND
SMT. THIRUMALAI COLLEGE OF SCIENCE
Plot No.337, Major R. Parmeshwaran Marg, Sewree Wadala Estate,
Wadala, Mumbai-400 031.

T.Y.B.Sc.(Computer Science)
Semester V

CERTIFICATE

Class:_____ University Seat No.:_____

Roll No.:_____

This is to certify that the experiments entered in this journal is the work of

Mr./Ms._____ in the Computer Science Department of S.I.W.S

Degree College during the year 2022 – 2023.

_____ _____
Teacher-In-Charge Co-Ordinator

_____ _____
Internal Examiner External Examiner

Date: _____

College Stamp

INDEX

Output of the code:

```
IDLE Shell 3.11.1
File  Edit  Shell  Debug  Options  Window  Help
Python 3.11.1 (tags/v3.11.1:a7a450f, Dec  6 2022, 19:58:39) [MSC v.1934 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
====== RESTART: E:\INFORMATION RETRIEVAL\practical 1(bitwise operation).py =====
And operator overloaded
8
Or operator overloaded
14
Xor operator overloaded
6
lshift operator overloaded
40960
rshift operator overloaded
0
Invert operator overloaded
-11
```

PRACTICAL NO 1

AIM: Write a program to demonstrate bitwise operation.

Input of the code:

```
class Geek():
    def __init__(self, value):
        self.value = value
   def __and__(self, obj):
        print("And operator overloaded")
        if isinstance(obj, Geek):
            return self.value & obj.value
        else:
            raise ValueError("Must be a object of class Geek")
   def __or__(self, obj):
        print("Or operator overloaded")
        if isinstance(obj, Geek):
            return self.value | obj.value
        else:
            raise ValueError("Must be a object of class Geek")
    def __xor__(self, obj):
        print("Xor operator overloaded")
        if isinstance(obj, Geek):
            return self.value ^ obj.value
        else:
            raise ValueError("Must be a object of class Geek")
    def __lshift__(self, obj):
        print("lshift operator overloaded")
        if isinstance(obj, Geek):
            return self.value << obj.value
        else:
            raise ValueError("Must be a object of class Geek")
 def __rshift__(self, obj):
        print("rshift operator overloaded")
```

```
        if isinstance(obj, Geek):
            return self.value >> obj.value
        else:
            raise ValueError("Must be a object of class Geek")
  def __invert__(self):
        print("Invert operator overloaded")
        return ~self.value
if __name__ == "__main__":
    a = Geek(10)
    b = Geek(12)
    print(a & b)
    print(a | b)
    print(a ^ b)
    print(a << b)
    print(a >> b)
    print(~a)
```

Output of the code:

```
IDLE Shell 3.11.1
File  Edit  Shell  Debug  Options  Window  Help
Python 3.11.1 (tags/v3.11.1:a7a450f, Dec  6 2022, 19:58:39) [MSC v.1934 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
======== RESTART: E:\INFORMATION RETRIEVAL\practical 2(edit Distance).py =======
5
```

PRACTICAL NO 2

AIM: Implement Dynamic programming algorithm for computing the edit distance between strings s1 and s2. (Hint. Levenshtein Distance)

Input of the code:

```
def editDistance(str1, str2, m, n):
    if m == 0:
        return n
    if n == 0:
        return m
    if str1[m-1] == str2[n-1]:
        return editDistance(str1, str2, m-1, n-1)
    return 1 + min(editDistance(str1, str2, m, n-1),
                   editDistance(str1, str2, m-1, n),
                   editDistance(str1, str2, m-1, n-1)
                   )
str1 = "monday"
str2 = "Saturday"
print (editDistance(str1, str2, len(str1), len(str2)))
```

Output of the code:

```
Python 3.11.1 (tags/v3.11.1:a7a450f, Dec  6 2022, 19:50:39) [MSC v.1934 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

======== RESTART: E:\INFORMATION RETRIEVAL\practical 3(page ranking).py ========
[[0.333+0.j]
 [0.217+0.j]
 [0.45 +0.j]]
[[0.415+0.j]
 [0.217+0.j]
 [0.368+0.j]]
[[0.350+0.j]
 [0.245+0.j]
 [0.397+0.j]]
[[0.378+0.j]
 [0.225+0.j]
 [0.397+0.j]]
[[0.378+0.j]
 [0.232+0.j]
 [0.39  +0.j]]
[[0.373+0.j]
 [0.232+0.j]
 [0.395+0.j]]
[[0.376+0.j]
 [0.231+0.j]
 [0.393+0.j]]
[[0.375+0.j]
 [0.232+0.j]
 [0.393+0.j]]
[[0.375+0.j]
 [0.231+0.j]
 [0.394+0.j]]
[[0.375+0.j]
 [0.231+0.j]
 [0.393+0.j]]
[[0.375+0.j]
 [0.231+0.j]
 [0.393+0.j]]
[[0.375+0.j]
 [0.231+0.j]
```

```
[[0.375+0.j]
 [0.231+0.j]
 [0.393+0.j]]
[[0.375+0.j]
 [0.231+0.j]
 [0.393+0.j]]
[[0.375+0.j]
 [0.231+0.j]
 [0.393+0.j]]
[[0.375+0.j]
 [0.231+0.j]
 [0.393+0.j]]
[[0.375+0.j]
 [0.231+0.j]
 [0.393+0.j]]
[[0.375+0.j]
 [0.231+0.j]
 [0.393+0.j]]
[[0.375+0.j]
 [0.231+0.j]
 [0.393+0.j]]
[[0.375+0.j]
 [0.231+0.j]
 [0.393+0.j]]
[[0.375+0.j]
 [0.231+0.j]
 [0.393+0.j]]
[[0.375+0.j]
 [0.231+0.j]
 [0.393+0.j]]
[[0.375+0.j]
 [0.231+0.j]
 [0.393+0.j]]
Final:
 [[0.375+0.j]
 [0.231+0.j]
 [0.393+0.j]]
sum 0.9999999999999951
```

PRACTICAL NO 3

AIM: Write a Program to Implement Page Rank Algorithm.

Input of the code:

```
import numpy as np
import scipy as sc
import pandas as pd
from fractions import Fraction
def display_format(my_vector, my_decimal):
    return np.round((my_vector).astype(np.cfloat), decimals=my_decimal)
my_dp = Fraction(1,3)
Mat = np.matrix([[0,0,1],
[Fraction(1,2),0,0],
[Fraction(1,2),1,0]])
Ex = np.zeros((3,3))
Ex[:] = my_dp
beta = 0.7
Al = beta * Mat + ((1-beta) * Ex)
r = np.matrix([my_dp, my_dp, my_dp])
r = np.transpose(r)
previous_r = r
for i in range(1,100):
    r = Al * r
    print (display_format(r,3))
    if (previous_r==r).all():
        break
previous_r = r
print ("Final:\n", display_format(r,3))
print ("sum", np.sum(r))
```

Output of the code:

```
IDLE Shell 3.11.1
File Edit Shell Debug Options Window Help
Python 3.11.1 (tags/v3.11.1:a7a450f, Dec  6 2022, 19:58:39) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: E:\INFORMATION RETRIEVAL\practical no 4(remove stopword).py =====
['This', 'is', 'a', 'sample', 'sentence', ',', 'showing', 'off', 'the', 'stop', 'words', 'filtration', '.']
['This', 'sample', 'sentence', ',', 'showing', 'stop', 'words', 'filtration', '.']
```

PRACTICAL NO 4

AIM: Write a program for Pre-processing of a Text Document: stop word removal.

Input of the code:

```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
example_sent = """This is a sample sentence,
                showing off the stop words filtration."""
stop_words = set(stopwords.words('english'))
word_tokens = word_tokenize(example_sent)
filtered_sentence = [w for w in word_tokens if not w.lower() in stop_words]
filtered_sentence = []
for w in word_tokens:
    if w not in stop_words:
        filtered_sentence.append(w)
print(word_tokens)
print(filtered_sentence)
```

Output of the code:

```
IDLE Shell 3.11.1
File  Edit  Shell  Debug  Options  Window  Help
Python 3.11.1 (tags/v3.11.1:a7a450f, Dec  6 2022, 19:58:39) [MSC v.1934 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: E:\INFORMATION RETRIEVAL\practical 5(similarly between two text).py =
File GFG.txt :
9 lines,
3 words,
3 distinct words
File file.txt :
11 lines,
4 words,
4 distinct words
The distance between the documents is:  1.570796 (radians)
```

PRACTICAL NO 5

AIM: Write a program to Compute Similarity between two text documents

Input of the code:

```
import math
import string
import sys
def read_file(filename):
        try:
            with open(filename, 'r') as f:
                data = f.read()
            return data
        except IOError:
            print("Error opening or reading input file: ", filename)
            sys.exit()
translation_table = str.maketrans(string.punctuation+string.ascii_uppercase,
                                                                                    "
"*len(string.punctuation)+string.ascii_lowercase)
def get_words_from_line_list(text):
        text = text.translate(translation_table)
        word_list = text.split()
        return word_list
def count_frequency(word_list):
        D = {}
        for new_word in word_list:
            if new_word in D:
                D[new_word] = D[new_word] + 1
                else:
                D[new_word] = 1
        return D
def word_frequencies_for_file(filename):
        line_list = read_file(filename)
```

```
        word_list = get_words_from_line_list(line_list)
        freq_mapping = count_frequency(word_list)
        print("File", filename, ":", )
        print(len(line_list), "lines, ", )
        print(len(word_list), "words, ", )
        print(len(freq_mapping), "distinct words")
        return freq_mapping
def dotProduct(D1, D2):
        Sum = 0.0
        for key in D1:
                if key in D2:
                        Sum += (D1[key] * D2[key])
                        return Sum
def vector_angle(D1, D2):
        numerator = dotProduct(D1, D2)
        denominator = math.sqrt(dotProduct(D1, D1)*dotProduct(D2, D2))
        return math.acos(numerator / denominator)
def documentSimilarity(filename_1, filename_2):
        sorted_word_list_1 = word_frequencies_for_file(filename_1)
        sorted_word_list_2 = word_frequencies_for_file(filename_2)
        distance = vector_angle(sorted_word_list_1, sorted_word_list_2)
        print("The distance between the documents is: % 0.6f (radians)"% distance)
documentSimilarity('GFG.txt', 'file.txt')
```

Output of the code:

```
IDLE Shell 3.11.1
File Edit Shell Debug Options Window Help
Python 3.11.1 (tags/v3.11.1:a7a450f, Dec  6 2022, 19:59:39) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
======== RESTART: E:\INFORMATION RETRIEVAL\practical no 6(map reduce).py =======
[(3, 'mapreduce')]
[(8, 'is'), (3, 'mapreduce')]
[(8, 'is'), (4, 'a'), (3, 'mapreduce')]
[(8, 'is'), (4, 'a'), (3, 'mapreduce'), (2, 'processing')]
[(8, 'is'), (4, 'a'), (3, 'mapreduce'), (2, 'processing'), (1, 'technique')]
[(8, 'is'), (6, 'and'), (4, 'a'), (3, 'mapreduce'), (2, 'processing'), (1, 'technique')]
[(8, 'is'), (6, 'and'), (4, 'a'), (3, 'mapreduce'), (2, 'processing'), (1, 'technique'), (1, 'program')]
[(8, 'is'), (6, 'and'), (4, 'a'), (3, 'mapreduce'), (2, 'processing'), (1, 'technique'), (1, 'program'), (1, 'model')]
[(8, 'is'), (6, 'and'), (4, 'a'), (3, 'mapreduce'), (2, 'processing'), (1, 'technique'), (1, 'program'), (1, 'model'), (1, 'for')]
[(8, 'is'), (6, 'and'), (4, 'a'), (3, 'mapreduce'), (2, 'processing'), (1, 'technique'), (1, 'program'), (1, 'model'), (1, 'for'), (1, 'distributed')]
[(8, 'is'), (6, 'and'), (4, 'a'), (3, 'mapreduce'), (2, 'processing'), (1, 'technique'), (1, 'program'), (1, 'model'), (1, 'for'), (1, 'distributed'), (1
, 'computing')]
[(8, 'is'), (6, 'and'), (4, 'a'), (3, 'mapreduce'), (2, 'processing'), (1, 'technique'), (1, 'program'), (1, 'model'), (1, 'for'), (1, 'distributed'), (1
, 'computing'), (1, 'based')]
[(8, 'is'), (6, 'and'), (4, 'a'), (3, 'mapreduce'), (2, 'processing'), (1, 'technique'), (1, 'program'), (1, 'on'), (1, 'model'), (1, 'for'), (1, 'distri
buted'), (1, 'computing'), (1, 'based')]
[(8, 'is'), (6, 'and'), (4, 'a'), (3, 'mapreduce'), (2, 'processing'), (1, 'technique'), (1, 'program'), (1, 'on'), (1, 'model'), (1, 'java'), (1, 'for')
, (1, 'distributed'), (1, 'computing'), (1, 'based')]
[(21, 'the'), (8, 'is'), (6, 'and'), (4, 'a'), (3, 'mapreduce'), (2, 'processing'), (1, 'technique'), (1, 'program'), (1, 'on'), (1, 'model'), (1, 'java'
), (1, 'for'), (1, 'distributed'), (1, 'computing'), (1, based')]
[(21, 'the'), (8, 'is'), (6, 'and'), (4, 'a'), (3, 'mapreduce'), (2, 'processing'), (1, 'technique'), (1, 'program'), (1, 'on'), (1, 'model'), (1, 'java'
), (1, 'for'), (1, 'distributed'), (1, 'computing'), (1, based'), (1, 'algorithm')]
[(21, 'the'), (8, 'is'), (6, 'and'), (4, 'a'), (3, 'mapreduce'), (2, 'processing'), (1, 'technique'), (1, 'program'), (1, 'on'), (1, 'model'), (1, 'java'
), (1, 'for'), (1, 'distributed'), (1, 'containn'), (1, 'computing'), (1, based'), (1, 'algorithm')]
[(21, 'the'), (8, 'is'), (6, 'and'), (4, 'a'), (3, 'mapreduce'), (2, 'processing'), (1, 'two'), (1, 'technique'), (1, 'program'), (1, 'on'), (1, 'model')
, (1, 'java'), (1, 'for'), (1, 'distributed'), (1, 'contains'), (1, 'computing'), (1, based'), (1, 'algorithm')]
[(21, 'the'), (8, 'is'), (6, 'and'), (4, 'a'), (3, 'mapreduce'), (2, 'processing'), (1, 'two'), (1, 'technique'), (1, 'program'), (1, 'on'), (1, 'model')
, (1, 'java'), (1, 'important'), (1, 'for'), (1, 'distributed'), (1, 'contains'), (1, 'computing'), (1, based'), (1, 'algorithm')]
[(21, 'the'), (8, 'is'), (6, 'and'), (4, 'a'), (3, 'mapreduce'), (2, 'processing'), (1, 'two'), (1, 'technique'), (1, 'tasks'), (1, 'program'), (1, 'on')
, (1, 'model'), (1, 'java'), (1, 'important'), (1, 'for'), (1, 'distributed'), (1, 'contains'), (1, 'computing'), (1, based'), (1, 'algorithm')]
[(21, 'the'), (8, 'is'), (6, 'and'), (4, 'a'), (3, 'mapreduce'), (2, 'processing'), (1, 'two'), (1, 'technique'), (1, 'tasks'), (1, 'program'), (1, 'on')
, (1, 'namely'), (1, 'model'), (1, 'java'), (1, 'important'), (1, 'for'), (1, 'distributed'), (1, 'contains'), (1, 'computing'), (1, based'), (1, 'algor
ithm')]
[(21, 'the'), (8, 'is'), (6, 'and'), (5, 'map'), (4, 'a'), (3, 'mapreduce'), (2, 'processing'), (1, 'two'), (1, 'technique'), (1, 'tasks'), (1, 'program'
), (1, 'on'), (1, 'namely'), (1, 'model'), (1, 'java'), (1, 'important'), (1, 'for'), (1, 'distributed'), (1, 'contains'), (1, 'computing'), (1, based')
, (1, 'algorithm')]
[(21, 'the'), (8, 'is'), (6, 'and'), (5, 'reduce'), (5, 'map'), (4, 'a'), (3, 'mapreduce'), (2, 'processing'), (1, 'two'), (1, 'technique'), (1, 'tasks')
, (1, 'program'), (1, 'on'), (1, 'namely'), (1, 'model'), (1, 'java'), (1, 'important'), (1, 'for'), (1, 'distributed'), (1, 'contains'), (1, 'computing'
```

PRACTICAL NO 6

AIM: Write a map-reduce program to count the number of occurrences of each alphabetic character in the given dataset. The count for each letter should be case-insensitive (i.e., include both upper-case and lower-case versions of the letter; Ignore non-alphabetic characters).

Input of the code:

```
Text = """MapReduce is a processing technique and a program model for distributed
computing based on java.The MapReduce algorithm contains two important tasks, namely
Map and Reduce.Map takes a set of data and converts it into another set of data, where
individual elements are broken down into tuples (key/values pairs).Secondly, reduce task,
smaller set of tuples. As the sequence of the name MapReduce implies, the reduce task is
always performed after the map job.Map stage - The map or mapper's job is to process the
input data. Generally the input data is in the form of (file or directory and is stored in the
Hadoop file system (HDFS).The input file is passed to the mapper function line by line. The
mapper processes the data and creates several small chunks of data.Reduce stage - This is
the combination of the Shuffle stage and the Reduce stage.  The Reducer's job is to process
the data that comes from the mapper.After processing, it produces a new set of ouput, which
will be stored in the HDFS.
"""

for char in '-.,\n':
    Text = Text.replace(char, ' ')
    Text = Text.lower()
    word_list = Text.split()
    from collections import Counter
    Counter(word_list).most_common()
    d = {}
    for word in word_list:
        d[word] = d.get(word, 0) + 1
        word_freq = []
        for key, value in d.items():
            word_freq.append((value, key))
            word_freq.sort(reverse=True)
            print(word_freq)
```

Ouput of the code:

```
IDLE Shell 3.11.1
File  Edit  Shell  Debug  Options  Window  Help
Python 3.11.1 (tags/v3.11.1:a7a450f, Dec  6 2022, 19:58:39) [MSC v.1934 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
======= RESTART: E:\INFORMATION RETRIEVAL\practical no 7(web crawler).py =======
Andhra Pradesh

['Andhra Pradesh\n']
Arunachal Pradesh

['Andhra Pradesh\n', 'Arunachal Pradesh\n']
Assam

['Andhra Pradesh\n', 'Arunachal Pradesh\n', 'Assam\n']
Bihar

['Andhra Pradesh\n', 'Arunachal Pradesh\n', 'Assam\n', 'Bihar\n']
Chhattisgarh

['Andhra Pradesh\n', 'Arunachal Pradesh\n', 'Assam\n', 'Bihar\n', 'Chhattisgarh\
n']
Goa

['Andhra Pradesh\n', 'Arunachal Pradesh\n', 'Assam\n', 'Bihar\n', 'Chhattisgarh\
n', 'Goa\n']
Gujarat

['Andhra Pradesh\n', 'Arunachal Pradesh\n', 'Assam\n', 'Bihar\n', 'Chhattisgarh\
n', 'Goa\n', 'Gujarat\n']
Haryana

['Andhra Pradesh\n', 'Arunachal Pradesh\n', 'Assam\n', 'Bihar\n', 'Chhattisgarh\
n', 'Goa\n', 'Gujarat\n', 'Haryana\n']
Himachal Pradesh

['Andhra Pradesh\n', 'Arunachal Pradesh\n', 'Assam\n', 'Bihar\n', 'Chhattisgarh\
n', 'Goa\n', 'Gujarat\n', 'Haryana\n', 'Himachal Pradesh\n']
Jharkhand
```

PRACTICAL NO 7

AIM: Write a program to implement simple web crawler.

Input of the code:

```
import requests
from bs4 import BeautifulSoup
URL="https://en.wikipedia.org/wiki/States_and_union_territories_of_India"
res=requests.get(URL).text
soup=BeautifulSoup(res,'lxml')
states=[]
for items in soup.find('table', class_='wikitable').find_all('tr')[1::1]:
    data=items.find_all(['th','td'])
    print(data[0].text)
    states.append(data[0].text)
    print(states)
```

Output of the code:

```
IDLE Shell 3.11.1
File  Edit  Shell  Debug  Options  Window  Help
Python 3.11.1 (tags/v3.11.1:a7a450f, Dec  6 2022, 19:58:39) [MSC v.1934 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
=============== RESTART: C:/Users/cs/Desktop/practical 8/item.py ===============
Expertise Data:
First
Second

Third

Fourth
```

PRACTICAL NO 8

AIM: Write a program to parse XML text, generate Web graph and compute topic specific page rank.

Input of the code:

XML File

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<root testAttr="testValue">
The Tree
<children>
<child name="Jack">First</child>
<child name="Rose">Second</child>
<child name="Blue Ivy">
Third
<grandchildren>
<data>One</data>
<data>Two</data>
<unique>Twins</unique>
</grandchildren>
</child>
<child name="Jane">Fourth</child>
</children>
</root>
```

Py.File

```
import xml.etree.ElementTree as ET
tree = ET.parse('item.xml')
root = tree.getroot()
print('Expertise Data:')
for elem in root:
  for subelem in elem:
 print(subelem.text)
```