Readme

B07901112 劉聿珉

ReadFile.py:

```
def parsefilecontent(filename):
    content = open(filename).read().split('\n')
    for i in range(0,len(content)-2):
        content[i] = content[i][4:]
    vote = content[0:-3]
    pagecontent = content[-2].split(' ')[:-1]
    return vote, pagecontent  #list, list
def readsearchlist(filename):
    content = open(filename).read().split('\n')
    # print(content)
```

將 file 中的各個資訊分開，並傳 vote 跟 content

Matrix.py:

```
import ReadFile as rd
import numpy as np
def getpointmatrix():
    a = np.array([[0 for i in range(501)]],dtype=float).T
    for i in range(500):
        vote ,content = rd.parsefilecontent("./web-search-files2/page%s"%(i))
        if len(vote) == 0:
            value = 0
        else: value = 1/len(vote)
        column = np.array([[0 for i in range(501)]],dtype=float)
        for i in vote:
            column[0][int(i)] = value
        a = np.concatenate((a,column.T),axis=1)
    a = np.delete(a,0,axis=1)
    a = np.concatenate((a,np.array([[0 for i in range(501)]]).T),axis=1)
    return a
```

利用 numpy 建立矩陣，以利後續運算

Main.py:

```
def find_v_matrix(d,DIFF):
    N = 501
    v = np.array([[1/501 for i in range(501)]]).T
    matrix = Matrix.getpointmatrix()
    counter = 0
    diff = 0
    while True:
        v_before = v
        v = (1-d)/N + d*np.dot(matrix, v_before)
        if sum(abs(v_before-v)) < DIFF:
            break
    return v, matrix
```

給 d 和 DIFF，會利用矩陣乘法算到誤差夠小時就會停止，並且回傳矩陣。

Main.py 中還有三段程式碼會分別產出三組所求的檔案(page rank list*12, reserve index*1, Search engine*12)
page rank list 就是將 v 中的值整理大小後輸出
reserve index 會整理出這 500 個 page 所用過的所有單字，一個一個尋找，若單字有出現在裡面就 output 出來
Search engine 分別討論AND跟OR的情況在文件中尋找，並且依規定輸出。
若想要產出檔案必須按照main.py中的註解只是un註解code。
複雜度:
Time complexity: O(N*W), 其中N為page數, W為page連到的頁數
Space complexity: O(N*N), 在建立matrix看誰連到誰的矩陣耗費最大