本次作業我使用了 python 來完成，此中還有 import numpy 這個套件。

How to run the code

先將要 input 的 page 放在 web-search-files2 這個資料夾中，即 web-search-files2 中含有 page0、page1、…、page499，最後將 web-search-files2/、list.txt 和 main.py(主程式)放在同一層的資料夾中，直接執行 main.py 即可產生所需要 output 的 25 個檔案，並會將所有檔案放在新增的 output 資料夾中。

Page Rank

一開始我建立一個 dictionary 叫做 page，page0~page499 為 key 值;每個 page 可連結出去的 page 為 value(value 為一 list)，接著呼叫 PageRank(page, d, DIFF)這個 function，此 function 的寫法與老師提供的 pesudocode 一致，細節不再此贅述，其中 PR 我是用 numpy.array 去儲存。

Time complexity:若在給定的 DIFF 值下可以收斂，那麼 PageRank 中有 3 個迴圈，所以 Time complexity = $O(n^3)$
Space complexity:因為 dictionary 的 space complexity 為 O(n)，而 PR 為一個大小為 n 的一為 array，所以 Space complexity = O(n)

Reverse index

一開始一樣建立一個 dictionary 叫做 pageStr，page0~page499 為 key 值;每個 page 儲存的 key words 為 value(value 為一 list)，接著呼叫 ReverseIndex(pageStr)這個 function，對 pageStr 做 reverse dictionary 並存成 pageStrReverse，也就是 key 變 value、value 變 key，達到 Reverse index 的效果。另外在同一 page 下可能會出現重複的字，因此我先對 pageStr 的 value 這個 list 先轉為 set 再轉回 list，已去掉重複的部分。

Time complexity:做 reverse dictionary 需用 2 層迴圈來完成，故 Time complexity = $O(n^2)$
Space complexity:因為需要另一個新的 dictionary 去存 reverse dictionary 的結果，但使用 2 個 reverse dictionary 的 Space complexity = O(n)

Search Engine

一開始先用 2 維的 list 把 list.txt 的字存起來，並呼叫 SearchEngine(words, PR, pageStrReverse)這個 function，新增一個 search 的 dictionary，接著檢查 list.txt 每一行的字是否存在在 pageStrReverse 的 key 裡面，若有，則將 pageStrReverse value 中的 page 依照 page rank 由大到小排序，並取前 10 大的 page 存在 search 的 value；若無則存 none。另外，在多個單字中要取 AND 跟 OR 的結果中，我使用 set 中 and、or 來做計算，即我先將每一個字在 pageStrReverse 中的 value 轉為 set 做 and、or 再轉回 list，最後再依據 page rank 由大排到小取前 10 個。

Time complexity:如果 list.txt 有 W 個字要 input，因為每個字要 traverse pageStrReverse 的 key，traverse 完還要對 page rank 進行 sort，因此 Time complexity = $O(Wn^2 \log n)$
Space complexity:因為要新增一個 dictionary 為 O(n)，中間新增來幫忙運算的 set 也 O(n)，整個 Space complexity 還是 O(n)