資料結構與程式設計 project #1

一、說明

此程式需輸入 2 項 argument：第一個是確認 Page Rank 值達到平衡的 stop difference（stop_diff），第二個是 dumping factor（d）。程式執行後，將輸出 Reverse index，以及對應至 d 和 stop_diff 的結果。

二、使用之方法

此次主要以 C++ STL 提供的 vector 和 map 做資料儲存。首先，設計一個命名為 Page 的 struct，之後建立存放所有 Page 的 vector。各個 Page 裡面包含以下五種資料：

- 該 page 的檔名（string page_name）
- 該 page 的 Page Rank 值（double pr_rank）
- 該 page 的連結到其他頁面的 outbranching 數（int out_links）
- 連結至該 page 的頁面清單（vector<string> links）
- 該頁面所含有的不重複字詞（vector<string> words）

1. Page Rank list

先將所有 page 的 Page Rank 值初始化為 1 / N（N 為 page 總數），並計算出 outbranching 數。依照 Page Rank 公式反覆計算，算到平衡（diff < stop_diff）為止。

接著使用 insertion sort 依 pr_score 放入另一個容器排序，生成排序好的 Page 的 vector。計算並排序完成後，輸出 page_name、out_links 與 pe_score。

2. Reverse index

利用前述排序過的 vector，以取出頁面所含的字詞 words 為 key，page_name 為 value，填入 STL 的 map 容器。

由於 map 本身為 red-black tree 的架構，因此做 traversal 便可得到依照 ASCLL 編碼順序的 key；由於是利用已排序的 vector，對應之 value 也會是依照 Page Rank 排序過的頁面。

3. Search engine

用空白切字，依搜尋字詞數量分為 1 字及多字進行處理。若是 1 字，就直接查詢 Reverse index；若是多字，分別作交集及聯集處理。

交集方面，先用關鍵字當 key 查詢 Reverse index 取得哪些頁面含有關鍵字。每個字都查完後，找出對應至最少頁面數的關鍵字，將該關鍵字對應之頁面先預設為交集頁面，再拿去逐一和其他關鍵字對應的頁面比較，一旦發現有任一關鍵字的對應頁面沒有就剔除，比較完成後即可得到交集。又，Reverse index 的 value 本來就是 Page Rank 由高排到低，所以不需另作排序。

聯集方面，同樣先用關鍵字當 key 查詢 Reverse index 取得哪些頁面含有關鍵字。每個字都查完後，找出對應至最多頁面數的關鍵字，將該關鍵字對應之頁面先預設為聯集頁面，再拿其他關鍵字對應的頁面來比較，一旦發現有任一關鍵字的對應頁面沒有就加入，比較完成後即可得到聯集。由於這次有新加入的頁面，因此必須重新進行排序，在此使用 heap sort。

三、分析

Reverse index 使用 red-black tree 架構的 map 產生，因此生成的時間複雜度為 O(n * log n)。而這次作業中，分別使用到 2 種排序法：insertion sort 及 heap sort。insertion sort 的時間複雜度為 O(n^2)，heap sort 為 O(n * log n)。

stop difference 的不同，並不會對 Page Rank 造成太大的影響，且這次所使用的資料。在 stop_diff = 0.01 和 stop_diff = 0.1 的情況下，除了 0.85 的 dumping factor 值都只用 2 輪便達到平衡；而 stop_diff = 0.001 則會多出幾輪，最多到 6 輪，然結果還是沒有差異很大。

相較於 stop difference，dumping factor 的不同對 Page Rank 影響甚大。大體上排序的改變不大，但是當 dumping factor 越大，高低分的落差越大。從公式中可看出 dumping factor 的作用是讓 rare event（沒有任何頁面連到該頁面）的分數不至於會是 0，同時也是連結到該頁面的其他頁面影響之比例。

最後，可以發現該頁面連結至其他頁面的數量在 Page Rank 的算法下顯得不是很重要，而該頁面被其他頁面連結的數量就顯得很重要。

四、其他

這次的作業大致上都有順利完成，唯一在用 list.txt 餵入 Search engine 時出了狀況。直接輸入的情況所使用的方法，和讀 list.txt 後轉換成輸入的情況所使用的方法幾乎相同，然而前者可順利執行，而後者卻出現大量 none。

雖然推測可能是讀入字串或讀入字串轉換的錯誤，但花了不少時間用了數種不同的寫法皆出現一樣的錯誤結果。這部分拖到最後還是沒有完成，算是比較遺憾的部分。