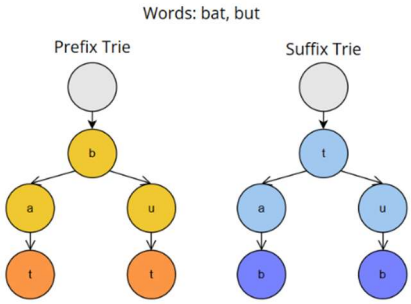
資料結構 Final Project – 專案報告

A. 程式實作

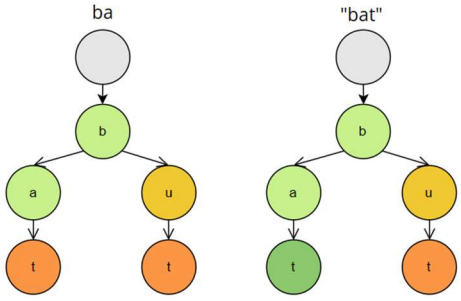
程式的內核其實十分簡單，主要分成 Trie 建構、Query 搜尋以及輸出三個步驟執行。

首先，程式將依序讀取每個文件的每一行，並透過 word_parse 與 split 函數將文件分成一串單字，再分別插入兩個有不同作用的 Trie—Prefix/Suffix Tree 裡，且由於 Suffix Tree 主要用作後綴搜尋，因此須將單字反向讀入，如右圖所示。



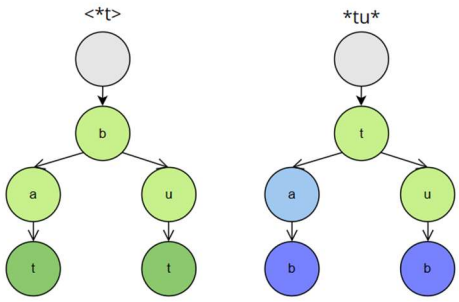
接著，程式會先讀取 query 檔案的每一行並存入包含搜尋關鍵字與運算字符（ +, -, / ）的列表中，再一行行輸入專門搜尋的函數 run_query 裡，該函數根據搜尋方式的不同有不同的流程如下：

- Prefix: 從 Prefix Trie 的 root 根據關鍵字的每個字元往對應方向走，若：
  - Prefix Search: 找不到路線→false，反之
  - Exact Search：找不到路線或終點非任何單字的結尾→false，反之



(綠色節點為 run_query 選擇路線)

- Suffix: 從 Suffix Trie 的 root 依序往關鍵字的每個字元走，若：
  - Suffix Search: 找不到路線→false，反之



- Wildcard: 根據字元的不同從 Prefix Trie 的 root 向下做 dfs 搜尋，若：
  - 字元為 a-z: 往該對應方向走，並推進至下個字元（若跑完所有字元→true）
  - 字元為*: 先推進至下個字元看在此字元是否可通，若否則朝所有路線行進如此遞迴，若找不到任何可跑完所有字元的路線則→false

一開始函數會將列表首個關鍵字的搜尋結果存入變數 matched 中，再依據運算字符做不同的處理，若運算字為 AND(+)或 SUB(-)則只會在當 matched=true 時執行下個關鍵字，反之或運算字為 OR(/)則只會當 matched=false 時執行，再將該關鍵字的搜尋結果存入變數 matched 中，若運算字為 SUB 則會反轉搜尋結果的值。

當 query file 的所有 query 都完成 run_query 的搜尋且皆將結果存入 matched_essays 列表中時，即可開始建構下一個 essay 的 Trie。為了節省空間，我還額外在每個 Trie node 中儲存 essay 標籤以確認此 node 為最新產生的而非之前的路線，這樣便避免了產生 node 而拖累執行時間的問題。

最後，當所有 essay 都完成搜尋過後，程式會依據 matched_essays 的值輸出對應的 title 進指定的輸出檔案中，並結束此程式。

B. 程式優化

由於 Linux 環境對空間與執行緒的高度限制，透過 multithreading 建構多個 Trie 以加快執行進度的方式經測試能在 windows 上能壓到近 100ms/1000 筆測資的優秀成績，但在 ubuntu 卻會因大量分配記憶體使速度大減至 1000ms/1000 筆測資以上，因此在多次測試與修改後我決定只使用兩個 Trie，並透過在 node 標籤做少量修改以達成建構 Trie 但不新增 node 的效果，以換取更快的建構速度。

除此之外，透過 omp.h 的 parallel programming 功能（編譯時須加入-fopenmp 標籤以引入該套件），程式能平行讀檔一次性讀入數百個檔案並存入序列中，再依序導入 Trie 的建構過程中以加快速度，而在 query 搜尋的部分我也使用了相同的技巧，利用多執行緒各自執行獨立的搜尋工作，經測試能夠使程式平均所需時間壓低至原先的 50-60%左右。

C. 結尾

對於不熟悉競程也不常接觸演算法的我而言，這次的期末專題對我來說是個不容小覷的挑戰，而在實作期間我也確實踩了不少坑。其中最印象深刻的一次就是作業系統的差別了，一開始我打算透過 Windows 實作再移轉到 Linux 跑分，然而卻因為方便而使用了大量的執行緒建構 Trie，也不曾想過大量的 new 以及 delete 會嚴重拖累在 Linux 上的表現，這才導致了我一直到截止前幾天才發現並嘗試重新建構程式碼，並以最大化降低 Trie 的空間占有率以及 new/delete 使用率為目標設計了另一套可在 Linux 達到 140-165%表現的程式（個人估計部分受到 VMware 的影響），這才解決了眼前的危機。

D. 參考資料

- https://www.geeksforgeeks.org/trie-insert-and-search
- https://stackoverflow.com/questions/11773115/parallel-for-loop-in-openmp
- TeodorDyakov/wildcard-trie: String trie that supports wildcard search (github.com)