資料結構與演算法

期末專題

˙組名：Exia

˙組員：B97902037 鐘明穎、B98902134 陳秀昌、B99902028 陳珮珮

˙工作分配

|  |  |
| --- | --- |
| 組員姓名 | 工作 |
| 鐘明穎 |  |
| 陳秀昌 |  |
| 陳珮珮 |  |

˙資料結構的比較

我們測試了三種資料結構來儲存字典，分別是sorted index array、minimum perfect hash table以及Trie。總括來說，在這三種資料結構當中，以Trie所生成的字典檔案大小最小，而sorted index array與minimum perfect hash table則差不多大，但都比Trie大很多。而對於這三種資料結構本身而言，表現方式及壓縮方法的不同也會產生出檔案大小不一樣的字典檔。

在這次的期末專題中，我們採用的是bzip2的壓縮格式，該格式使用了Burrows-Wheeler transform、move-to-front transform以及霍夫曼編碼來達成壓縮檔案的目的。

以下是上述三種資料結構在judge網站上所測出來的結果：

|  |  |  |  |
| --- | --- | --- | --- |
| 資料結構 | 壓縮與否 | 檔案大小 | 執行時間 |
| Sorted index array | X | 847143.0 | 0.05 |
| Sorted index array | O | 280975.0 | 0.22 |
| Minimum perfect hash table | X | 877940.0 | 0.03 |
| Minimum perfect hash table | O | 342098.0 | 0.25 |
| Trie | X | 622750.0 | 0.21 |
| Trie | O | 112876.0 | 0.36 |

˙建議的資料結構

在根據上表的比較之後，可以發現trie雖然在速度上比較慢，但是在生成的字典大小上，卻遠勝過其他兩種方法。因此我們認為trie是最適合用在儲存字典的資料結構。

˙Trie的優點

Trie在儲存字典上類似26元樹一般，每個連接小孩的邊都代表一個英文字母，因此其在搜尋上非常有效率。搜尋一長度為m的單字，只需要O(m)的時間。而且其在空間上的利用也很有效率，因為其會將所有單字中，在同一個位置出現的同一個字元合併為一個，這與dense array或是hash table相比皆是很省空間。

˙Trie的缺點

Trie的缺點其實我想不太到，我個人認為它是我心目中理想的資料結構，可是在測試時卻發現其執行速度總是比較慢。推測可能是cache missing所造成的瓶頸。

˙如何編譯與執行

將檔案解壓縮後輸入make即可編譯，而使用的方法則與網站上所定的格式一樣。

˙Reference

Trie：<http://en.wikipedia.org/wiki/Trie>

Minimum Perfect Hashing：<http://cmph.sourceforge.net/>

Bzip2：<http://www.bzip.org/index.html>