**作業三**

**110753201 資科碩一 曹昱維**

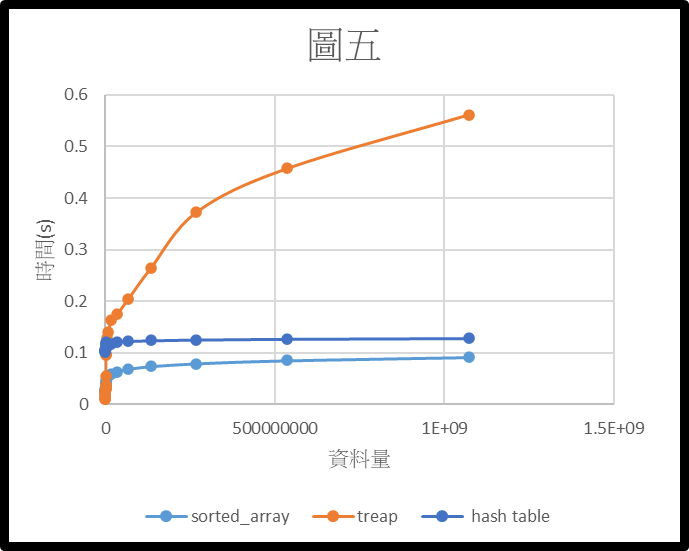
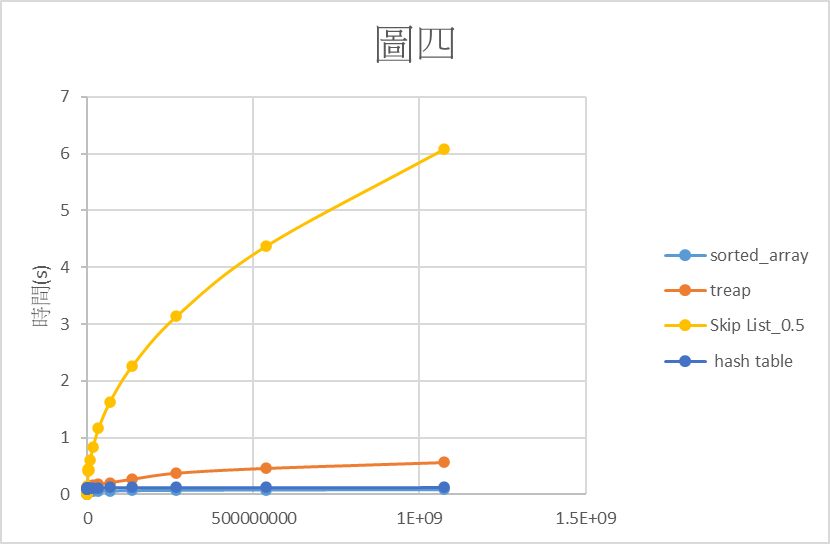
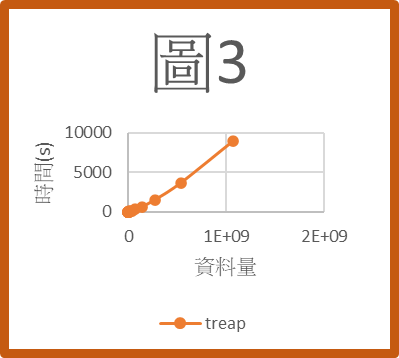
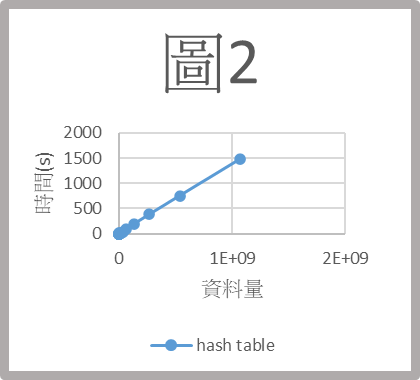
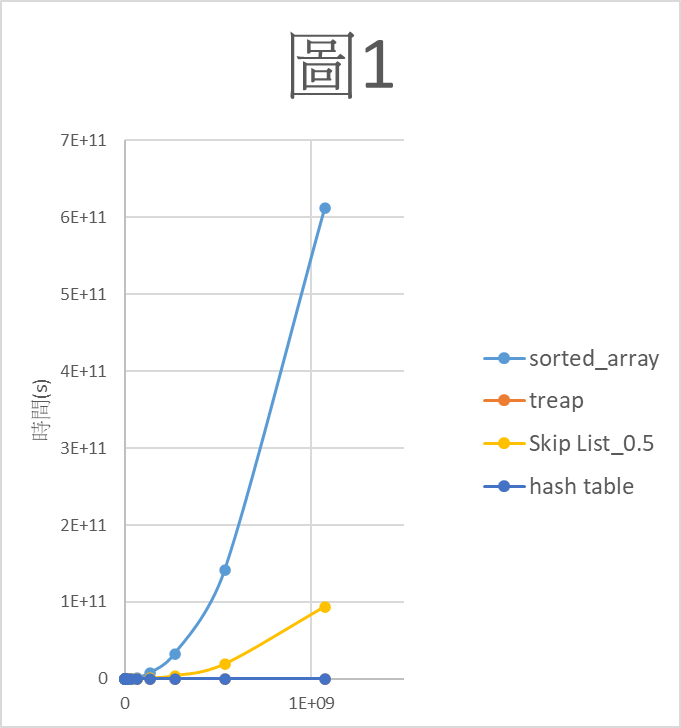
* ***作業目的：比較treap、skip list、sorted array與hash table。***
  1. *需自行實作上課提到的hash table（不須保證perfect hashing）。*
  2. *可假設一開始就知道n值，所以array初始化大小即可為。*
  3. *若額外實作可以如同dynamic array reallocation的hash table，並與其他資料結構比較，最多加10分*。
* ***（30%）在報告中請畫出每種資料結構新增資料所需時間：***

1. 圖一為四種資料結構在不同的資料量下所需的插入時間紀錄圖，圖二與圖三為圖一的局部放大圖
2. 原始數據(表一)為五種資料結構分別在不同的數據量時新增資料所需時長紀錄，當時長超過1個小時，或是記憶體不足的情況下，就按其在EXCEL的趨勢圖公式來推估後續數據(表格中的藍底白字)
   1. Sorted array insertion 估算公式 : y = 5E-08\*x^2.1136
   2. Treap insertion 估算公式 : y = 2E-08\*x^1.2901
   3. Skip list\_0.5 insertion 估算公式 : y = 2E-10\*x^2.289
   4. Hash table insertion 估算公式 : y = 2E-06\*x^0.9824
3. 從圖一，圖二，圖三中的結果可以看出:
   1. Hash table 所需的時間最少
   2. 理論上 Hash插入時間複雜度為O(1)，但是隨著資料量的成長，Hash的插入時間卻也跟著提升
   3. Sorted Array所需的時間遠大於Hash table 與Treap，因為在這裡採用的排序方法的時間複雜度是O(nlogn)，其遠大於Hash table的O(1) 與 Treap的O(logn)

* ***（30%）在報告中請畫出每個資料結構搜尋資料所需時間：***
  1. 圖四為四種資料結構在不同的資料量下所需的搜尋時間紀錄圖，圖五為圖四的局部放大圖
  2. 原始數據(表二)為四種資料結構分別在不同的數據量時搜索資料所需時長紀錄，但是在搜索之前筆需要先插入資料，當插入時長超過1個小時，或是記憶體不足的情況下，搜索時長就按其在EXCEL的趨勢圖公式來推估後續數據(表格中的藍底白字)

1. Sorted array search估算公式 : y = 0.0102\*x^0.1056
2. Treap search估算公式 : y = 0.0012\*x^0.2957
3. Skip list\_0.5 search估算公式 : y = 0.0003\*x^0.4769
4. Hash table search 估算公式 : y = 0.0901\*x^0.0171
   1. 可以從圖四觀察出，除了hash table以外的三種資料結構的搜索時間是符合O(logn)的趨勢，而hash table的搜索時間是符合O(1)的趨勢

* ***（30%）錄製影片（最長10分鐘）解釋你實作hash table的程式碼。報告請附影片連結。***
  1. 影片連結:
* ***（10%）實驗程式碼（含新增與搜尋的程式碼範例）與使用說明。***
  1. 使用說明:
     1. 執行test() 就會在工作目錄產生各種資料結構的 insert time 記錄，search time 記錄
        + 這些紀錄都會以csv檔儲存
        + 使用test () 需要輸入:
          - string file\_name\_it : insert time 記錄檔名
          - string file\_name\_st : search time 記錄檔名
          - string type\_record : 記錄檔中的註記文字，以及執行哪一種資料結構的運算
* ***（10%）心得、疑問、與遇到的困難***
  1. 疑問1 : Hash table 在插入資料的時間複雜度應該是O(1)，但是隨著資料量的成長，Hash的插入時間卻也跟著提升
  2. 遇到的困難 : 一開始是對每一個資料結構都寫一個測試函數，但因為對C++不熟，所以一直找不到方法把所有測試函數整合再一起，後來在stackoverflow 上發問，才知道可以用struct 與 template 來做成我想像中的功能
* ***報告請另外註明：***
  1. 作業程式碼 : https://github.com/theabc50111/nccu\_cs\_hw/blob/main/DataStructure\_HW/HW2/hw2.cpp
  2. 資料結構程式碼來源 :
     1. Skip list: <https://www.twblogs.net/a/5d09490abd9eee1e5c813476>
     2. Treap : <https://www.geeksforgeeks.org/treap-set-2-implementation-of-search-insert-and-delete/>
     3. Sorted array :
        + <https://www.cplusplus.com/reference/algorithm/sort/>
        + https://www.cplusplus.com/reference/algorithm/binary\_search/



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 表一 | | | | |
| 資料量 | sorted\_array | treap | Skip List\_0.5 | hash table |
| 1024 | 0.118991 | 0.000157 | 0.003101 | 0.002425 |
| 2048 | 0.513746 | 0.00037 | 0.009086 | 0.002258 |
| 4096 | 2.35717 | 0.001011 | 0.028855 | 0.004337 |
| 8192 | 10.0068 | 0.00236 | 0.125362 | 0.009066 |
| 16384 | 43.0796 | 0.005543 | 0.528231 | 0.019428 |
| 32768 | 185.266 | 0.018564 | 2.19997 | 0.039023 |
| 65536 | 793.683 | 0.040095 | 11.7064 | 0.079411 |
| 131072 | 3415.67 | 0.079898 | 65.0538 | 0.172225 |
| 262144 | 14855.6 | 0.21578 | 586.127 | 0.301063 |
| 524288 | 61355.4269 | 0.459132 | 3221.26 | 0.606563 |
| 1048576 | 265527.7901 | 1.05517 | 13548.5 | 1.22402 |
| 2097152 | 1149124.224 | 2.56158 | 82553.2 | 2.42018 |
| 4194304 | 4973063.203 | 6.3147 | 288601.4557 | 5.3753 |
| 8388608 | 21521918.26 | 18.3904 | 1410445.071 | 10.3725 |
| 16777216 | 93140373.76 | 47.6611 | 6893088.231 | 18.8713 |
| 33554432 | 403083457.6 | 123.794 | 33687710.6 | 38.1039 |
| 67108864 | 1744423682 | 282.33 | 164637649.7 | 97.7368993 |
| 134217728 | 7549339787 | 611.9522355 | 804612578.6 | 193.1036241 |
| 268435456 | 32671266614 | 1496.498628 | 3932280392 | 381.5243775 |
| 536870912 | 1.41391E+11 | 3659.612653 | 19217732229 | 753.7965755 |
| 1073741824 | 6.11899E+11 | 8949.399963 | 93920370670 | 1489.313163 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **表二** | | | | |
| 資料量 | sorted\_array | treap | Skip List\_0.5 | hash table |
| 1024 | 0.022243 | 0.010066 | 0.011863 | 0.102385 |
| 2048 | 0.022263 | 0.01118 | 0.015159 | 0.104716 |
| 4096 | 0.025243 | 0.012922 | 0.017902 | 0.102846 |
| 8192 | 0.025643 | 0.016056 | 0.021901 | 0.10311 |
| 16384 | 0.027887 | 0.021385 | 0.027376 | 0.10347 |
| 32768 | 0.029286 | 0.024943 | 0.031778 | 0.105057 |
| 65536 | 0.032101 | 0.028871 | 0.055465 | 0.109526 |
| 131072 | 0.036505 | 0.031257 | 0.053954 | 0.109614 |
| 262144 | 0.039491 | 0.038187 | 0.152943 | 0.111212 |
| 524288 | 0.040981397 | 0.055122 | 0.107731 | 0.120776 |
| 1048576 | 0.044093596 | 0.098269 | 0.418019 | 0.117863 |
| 2097152 | 0.047442142 | 0.094732 | 0.442897 | 0.117517 |
| 4194304 | 0.051044982 | 0.129325 | 0.431984101 | 0.116156 |
| 8388608 | 0.054921429 | 0.139935 | 0.601213838 | 0.115837 |
| 16777216 | 0.059092259 | 0.163874 | 0.836739311 | 0.116533 |
| 33554432 | 0.06357983 | 0.174807 | 1.164531869 | 0.12118 |
| 67108864 | 0.068408194 | 0.204799 | 1.620737135 | 0.1226204 |
| 134217728 | 0.073603234 | 0.264924 | 2.255660777 | 0.124082444 |
| 268435456 | 0.079192794 | 0.372897265 | 3.139315706 | 0.125561921 |
| 536870912 | 0.085206834 | 0.457724088 | 4.369142382 | 0.127059039 |
| 1073741824 | 0.091677592 | 0.561847351 | 6.080753559 | 0.128574007 |