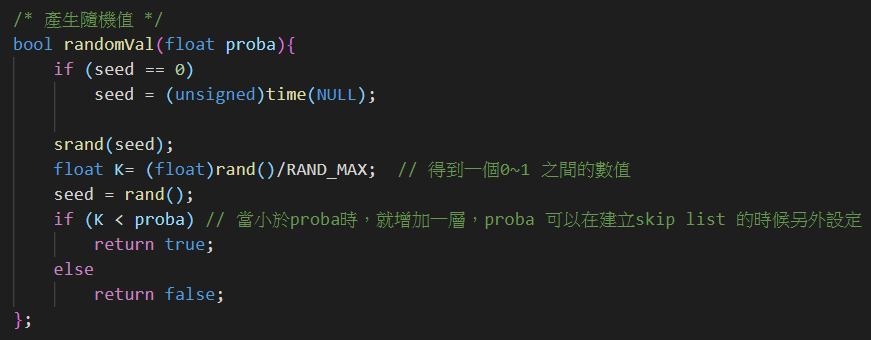
**作業三**

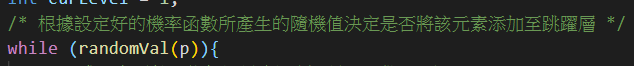
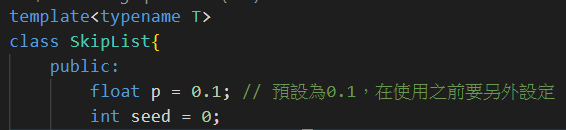
**110753201 資科碩一 曹昱維**

* ***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，理論上 Hach插入時間複雜度為O(1)
   2. 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)
   3. Skip list 隨著機率上升而插入資料所需的時間也大幅上升，推估的原因為Skip list在插入資料後，還需要向上疊加，而這個**向上疊加的行為主導了大部分的時間**，所以可以看到Skip list 的**插入時間隨著向上疊加機率上升而大幅提升**

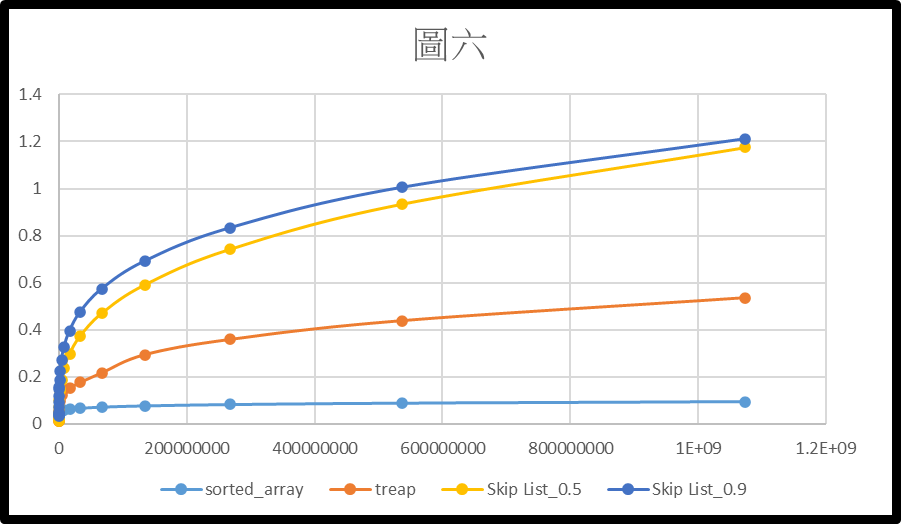
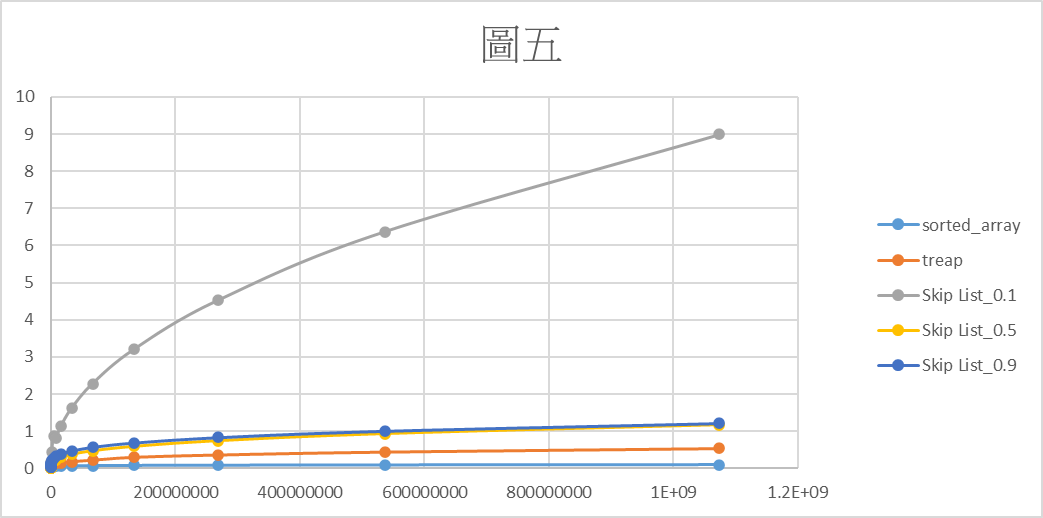
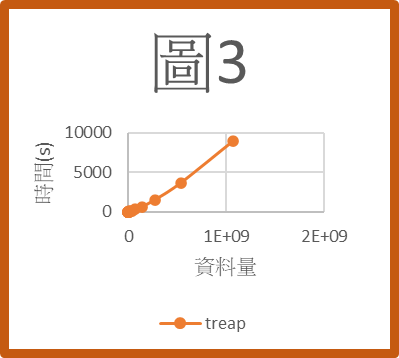
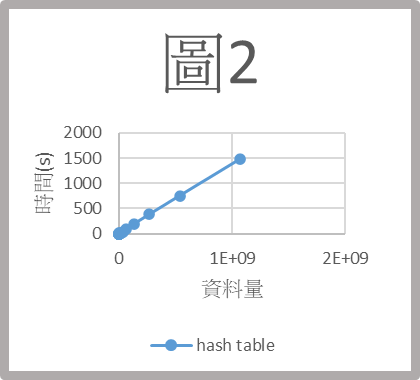
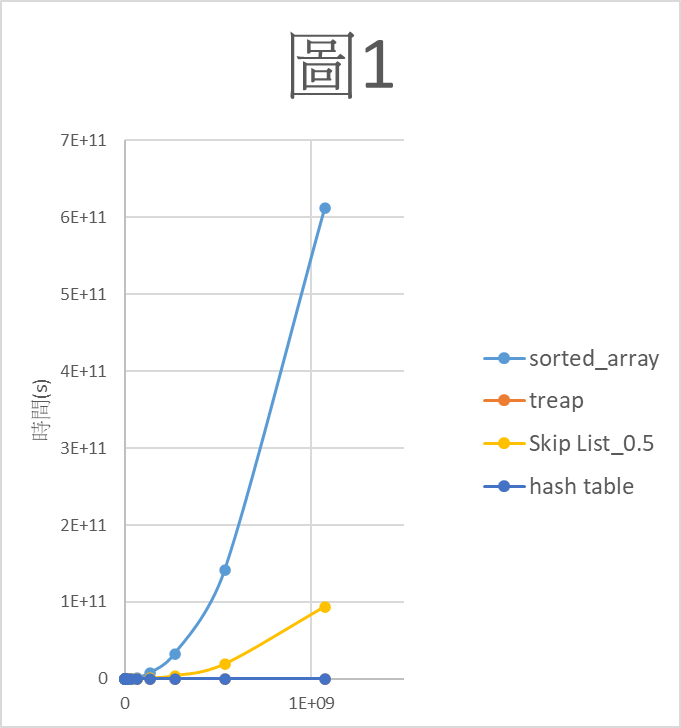
* ***（30%）在報告中請畫出每個資料結構搜尋資料所需時間：***
  1. 圖四為四種資料結構在不同的資料量下所需的搜尋時間紀錄圖，圖六為圖五的局部放大圖
  2. 原始數據(表三)為五種資料結構分別在不同的數據量時搜索資料所需時長紀錄，但是在搜索之前筆需要先插入資料，當插入時長超過1個小時的情況，搜索時長就按其在EXCEL的趨勢圖公式來推估後續數據(表格中的藍底白字)
  3. Sorted array insertion 估算公式 : y = 0.0112\*x^0.1023
  4. Treap insertion 估算公式 : y = 0.0013\*x^0.2895
  5. Skip list\_0.1 insertion 估算公式 : y = 0.0003\*x^0.4957
  6. Skip list\_0.5 insertion 估算公式 : y = 0.0012\*x^0.3312
  7. Skip list\_0.9 insertion 估算公式 : y = 0.0044\*x^0.2702
  8. 可以從圖五觀察出，五種資料結構的搜索時間是符合O(logn)的趨勢
  9. 從圖六可以看出**Skip list\_0.5 在所有skip list中所需的時間最少**，理由可以參考上述
* ***（5%）在報告中請畫出三種skip list在不同的n值時，平均每個data的additional copy個數。請解釋你得到的結果。n = 2^k (k = 10, 11, 12, …直到Tinsert(n)超過一小時或是k=30)***
  1. 平均每個data的additional copy個數也同時是skip list 的平均層數，而我們的平均層數又受到初始設定的機率(0.1, 0.5, 0.9)影響
  2. 圖七為三種Skip list 在不同資料量下平均每個data的additional copy個數的紀錄圖
  3. 原始資料(表三)是三種skip list 的平均每個data的additional copy個數紀錄
  4. 從圖七可以看出 : **平均每個data的additional copy個數並不會隨著資料量變化**
  5. 平均層數的期望值： 
  6. 我們的實驗結果也符合上述算式
* ***（10%）在報告中請畫出三種skip list在不同的n值時的list個數。請解釋你得到的結果。  
  n = 2^k (k = 10, 11, 12, …, 直到Tinsert(n)超過一小時或是k=30)***
  1. 圖八為三種Skip list在不同資料量下的list數量記錄圖，圖九是圖八的放大圖
  2. 原始資料(表四)為為三種Skip list在不同資料量下的list數量記錄
  3. 從圖八與圖九可以看出，三種Skip list的線條幾乎疊在一起，這代表**list 數量與上升機率沒什麼關聯**
  4. 從圖八也可以看出Skip list的**List 數量與資料量呈線性關係**
* ***（5%）解釋如何修改skip list程式碼（以註解呈現）***
  1. 決定是否要上升的隨機函數:
     1. 這個隨機函數被修改成可以設定小於一定數值才會回傳true

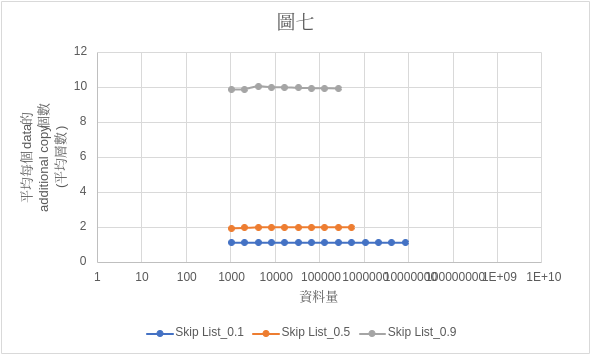
******

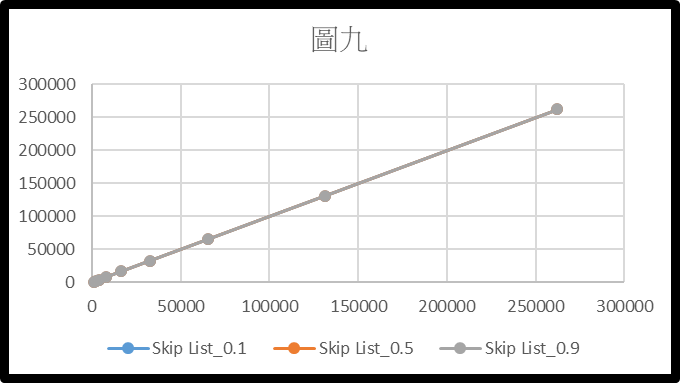
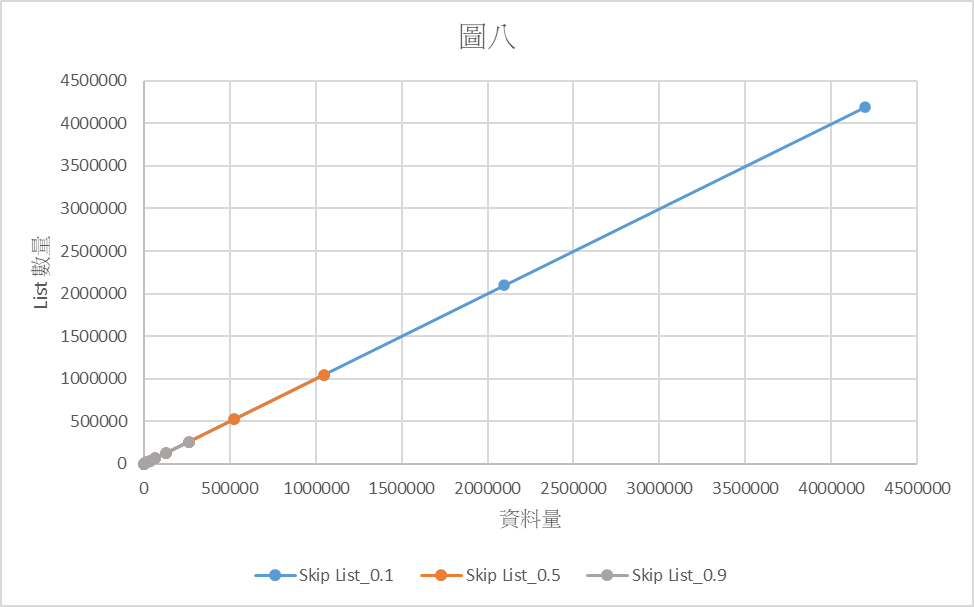
* 1. Insert method中的則是藉由執行上述的隨機函數來決定是否上升
     1. ******
  2. 另外新增Skip list 的上升閾值p，預設值為0.1，在使用前另外設定即可
     1. ******
* ***（10%）實驗程式碼（含新增與搜尋的程式碼範例）與使用說明。***
  1. 使用說明:
     1. 執行test\_skip\_list() 就會在工作目錄產生Skip List 的 insert time 記錄，search time 記錄，list number 記錄，average list layer 記錄
        + 這些紀錄都會以csv檔儲存
        + 使用test\_skip\_list() 需要輸入:
          - float prob : 上升閾值
          - string file\_name\_it : insert time 記錄檔名
          - string file\_name\_st : search time 記錄檔名
          - string file\_name\_l : list number記錄檔名
          - string file\_name\_al : average list layer 記錄檔名
          - string type\_record : 記錄檔中的註記文字
     2. 執行test\_treap()就會在工作目錄產生Treap 的insert time 記錄，search time 記錄
        + 這些紀錄都會以csv檔儲存
        + 使用test\_treap () 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
     3. 執行test\_sorted\_array ()就會在工作目錄產生Sorted Array的insert time 記錄，search time 記錄
        + 這些紀錄都會以csv檔儲存
        + 使用test\_sorted\_array () 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
* ***（10%）心得、疑問、與遇到的困難***
  1. 疑問1 : 圖四是只考慮在最底層插入資料(去除掉向上疊加資料)的情況下，三種skip list 的插入時間，雖然這三條線的趨勢符合0.5的機率應該是插入時間最少的，但是這三條線的趨勢卻沒有符合**O(logn)的趨勢(理論上來說應該要符合)**，這點我還不清楚為什麼，不太確定是程式碼的問題還是有其他的因素影響
  2. 疑問2 : 圖五為五種資料結構在不同的資料量下所需的搜尋時間紀錄圖，雖然五種資料結構的

搜索時間都符合O(logn)，但還是有一些常數倍的差距，我還不太確定原因是什麼

* 1. 遇到的困難1 : Skip list 程式碼來源並沒有發揮skip list 的優勢，原來的程式碼在插入資料時，是直接走到最底下，在往右搜索然後才插入，為了找出這個問題花費不少時間
* ***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\_0.1 | Skip List\_0.5 | Skip List\_0.9 |
| 1024 | 0.116163 | 0.000158 | 0.001347 | 0.00339 | 0.025146 |
| 2048 | 0.516636 | 0.000304 | 0.002895 | 0.007824 | 0.101677 |
| 4096 | 2.42442 | 0.000748 | 0.006278 | 0.02906 | 0.433132 |
| 8192 | 10.4029 | 0.001899 | 0.014581 | 0.127707 | 1.89453 |
| 16384 | 46.3294 | 0.003843 | 0.034601 | 0.563815 | 7.3325 |
| 32768 | 191.083 | 0.008394 | 0.113346 | 2.35801 | 38.1586 |
| 65536 | 819.387 | 0.023576 | 0.367627 | 9.1506 | 259.826 |
| 131072 | 3708.27 | 0.058011 | 1.41725 | 47.033 | 1716.52 |
| 262144 | 14847.9 | 0.108416 | 6.77385 | 472.308 | 9328.04 |
| 524288 | 67993.22293 | 0.300291 | 30.2674 | 2866.61 | 35004.36332 |
| 1048576 | 295849.3779 | 0.906438 | 251.04 | 13439.7 | 174231.6441 |
| 2097152 | 1287287.918 | 1.69457 | 1452.94 | 44366.45161 | 867225.1953 |
| 4194304 | 5601195.432 | 4.25569 | 6972.47 | 209803.8391 | 4316549.632 |
| 8388608 | 24371696.36 | 11.7082 | 7897.164632 | 992138.1879 | 21485308.34 |
| 16777216 | 106045145.3 | 23.9074 | 29003.09116 | 4691707.207 | 106941542.2 |
| 33554432 | 461419372.6 | 53.2171 | 106516.6216 | 22186542.94 | 532293661.5 |
| 67108864 | 2007709422 | 123.156 | 391192.4636 | 104917605.9 | 2649452554 |
| 134217728 | 8735864516 | 258.2411756 | 1436691.675 | 496143273 | 13187455240 |
| 268435456 | 38011142448 | 611.6553864 | 5276387.358 | 2346204387 | 65639588620 |
| 536870912 | 1.65393E+11 | 1448.732221 | 19378036.38 | 11094930290 | 3.26716E+11 |
| 1073741824 | 7.19649E+11 | 3431.384885 | 71167688.91 | 52466647318 | 1.62621E+12 |

|  |  |  |  |
| --- | --- | --- | --- |
| **表二** | | | |
| 資料量 | Skip List\_0.1\_no\_up | Skip List\_0.5\_no\_up | Skip List\_0.9\_no\_up |
| 1024 | 0.000806 | 0.000959 | 0.001551 |
| 2048 | 0.00205 | 0.002725 | 0.004476 |
| 4096 | 0.005328 | 0.00631 | 0.010452 |
| 8192 | 0.011735 | 0.013016 | 0.024366 |
| 16384 | 0.024824 | 0.028065 | 0.051083 |
| 32768 | 0.052085 | 0.061243 | 0.130241 |
| 65536 | 0.114321 | 0.129423 | 0.365587 |
| 131072 | 0.256397 | 0.300593 | 1.0285 |
| 262144 | 0.639474 | 0.876259 | 2.49592 |
| 524288 | 1.64184 | 2.52199 | 6.520141779 |
| 1048576 | 5.13809 | 3.801327529 | 16.20653016 |
| 2097152 | 16.5588 | 8.787807917 | 40.2831148 |
| 4194304 | 49.919 | 20.3154207 | 100.1281164 |
| 8388608 | 296.281 | 46.96464945 | 248.8794559 |
| 16777216 | 277.5757533 | 108.571628 | 618.6172855 |
| 33554432 | 701.6159532 | 250.9930029 | 1537.641363 |
| 67108864 | 1773.443609 | 580.2389511 | 3821.97688 |
| 134217728 | 4482.654962 | 1341.38098 | 9499.944277 |
| 268435456 | 11330.60866 | 3100.968887 | 23613.1573 |
| 536870912 | 28639.8783 | 7168.737428 | 58693.10192 |
| 1073741824 | 72391.75349 | 16572.49659 | 145888.166 |