DS 作業2-實驗報告

1. 使用的資料結構

std:map(似乎是紅黑樹)

Source code: C++ STL

1. 初始化



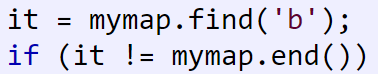
**B.**  Insert

Insert的operator overloading



**C.** Search

Find函式，如果未找到會傳最後一個iterator end()



Btree

Source code: <https://www.geeksforgeeks.org/insert-operation-in-b-tree/>

1. 初始化:



\*3表示每一個node的array大小

**B.** 插入



使用member function

C.搜尋



如果找不到值會回傳NULL

Skip list

Source code: <https://www.geeksforgeeks.org/skip-list-set-3-searching-deletion/>

A.初始化:



第一個參數是給最深開到幾層

第二個參數是給開下一層的機率

B.插入



C.搜尋



類似於Btree的用法

1. 實驗結果

A.插入時間比較

縱軸單位: 秒 ， 橫軸單位: 第i組1000筆資料

註1: 為了更好的看出趨勢，將每100筆資料再平均一次，得到以下結果

大概只有Btree有上升的趨勢

B.搜尋10^5筆資料時間比較

Std::map 搜尋時間: 0.043 秒

Btree 搜尋時間: 0.04 秒

Skip list 搜尋時間:0.042 秒

================================補充筆記=============================

1. Skiplist層數

使用的source code有限制最大層數

剛開始沒有注意直接使用預設的3，導致效能會在100筆之後就上升

(約0.04秒/一組資料)

搜尋時間會暴增到4.354秒

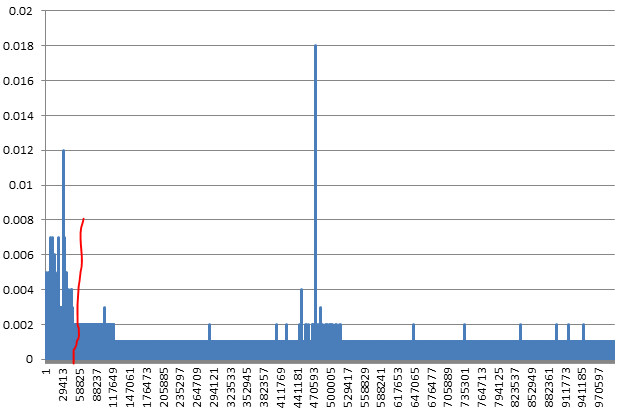
1. skiplist 和 Map 的插入時間大概是O(1)?

假設一:資料量不夠大，看不出O(logn)的趨勢

實驗:試試10^9情形下有無改善

10^9情況下還是沒有看出O(logn)趨勢

skiplist可能是塞不下，後面會出現很多0秒就完成的



額外嘗試AVL tree 與 treap (一樣做10^7 插入資料然後 每100筆均)

發現AVL tree有這樣的特徵(logN)，而AVL tree 插入(應該)是比較慢的

所以應該可以得出是其他資料結構太快 ?)