B06705048 資管二 王佩琳

一、ADT 實作

(1)Doubly Linked List

- 簡介：

  由 Node 串起所需要儲存的資料，每個 Node 內含：自己的資料，和分別指向前一個和後一個 Node 的指標，DList 本身只記住最前面的 Node，存在一 dummy node 為__head 的_prev 和最後一個有存資料的 node 的_next，操作性上為沒有大小限制，可以無限加入 node，但計算大小和尋找 node 的時候，都需要使用 iterator 整個搜尋一遍。

- 實作：

  分別由 3 個 function：begin()、tail()、end()記住資料的頭，資料的尾和 dummy node，overload 各 iterator 功能，在三者之間找到所需要處理的正確資料

- 實作優缺點：

  寫起來比較清楚，加上善用 for 迴圈遍歷資料，就能處理每個需要 implement 的 function

(2)Array

- 簡介：

  由一 dynamic array 儲存資料，每次剩餘容量不夠時，就開一個兩倍大的 array，並把先前資料複製過去，雖然比較耗時，但就能直接呼叫 STL 內的 sort，且可以直接取出資料，如 erase 能把最後的一個資料直接取代要移除的資料。

- 實作：

  資料由 T* _data 儲存，可視為一 array 使用，在 size 和 capacity 的限制下，使用 index 對資料做出處理

- 實作優缺點：

  無，沒有增加 function，直觀的 implement 老師給的 code 的 data member 和 function

(3)BST

- 簡介：

  由 node 組成，再由_root 於 BSTree 中紀錄樹的頂點，每個 node 除了自己本身的資料外，還存在指標指向他的 parent、left child、right child，永遠維持排序好的狀態：left child < itself < right child，因此，在 insert 時，需要正確使用 traversal，指到正確的位置。

- 實作：

  主要寫了四個 function 幫助尋找到想要的資料位置，分別為

leftmost(subtree 中最小值)(一路向左)、successor(下一個)(先向右，再一路向左)、rightmost(subtree 中最大值)(一路向右)、predecessor(上一個)(先向左，再一路向右)。

- 實作優缺點：

  寫到很混亂，沒有很明確的去設出 dummy node 忘了在哪裡沒有更新好後就放棄使用了，在 iterator –的時候會出現問題，且在 delete 的時候需要考慮到很多不同的 case，額外把 root 的 parent 更新成 NULL，也不確定在 delete 時，是直接交換(successor 和要 delete 的 node)，裡面的資料比較好維持，還是分別更改影響的 node，修修補補更新 parent 和 root，目前狀態是可以 pop_front 和 pop_back，但 delete 還有 bug 還沒找到。

二、實驗比較

//bst 還沒有寫好 比較是用老師給的 ref

- 實驗設計：將基本的每種 command 跑過一次

1. insert：(adta -r 100000)

   (1)dlist：0.01 秒

   一直 new node 在最後就好，會最快

   (2)array：0.06

   當超過 array 的 capacity，要重開新地並複製過去，會較慢

   (3)bst：0.09

   insert 的同時就要 sort 完，會花較久時間

2.delete：(adtd -r 50000) //原有 100000 比資料

   (1)dlist：12.18

   要從 begin() tarverse 到位置才能刪除，會較久

   (2)array：0.01

   直接把 size--，幾乎不用花時間

   (3)bst：66.92

   要從_root tarverse 到位置才能刪除，會較久

3. find：(adtq yyy)

   (1)dlist：0

   (2)array：0

   (3)bst：0.01

   皆是從 begin()到 end()去找，時間差異不大

3.sort：(adts) //此時有 50000 比資料

   (1)dlist：0.06

   使用 quick sort，跑起來出乎意料的快，原本用 bubble sort 寫，debug 的時

候就發現會花較久時間

(2)array：0.03

呼叫 STL 的 sort

(3)bst：0

一直維持在 sort 狀態

4.print：(adtp) //此時有 50000 比資料

(1)dlist：0.04

(2)array：0.03

(3)bst：0.02

印出來的時間皆差不多。