HW 5 Report - ADT Comparison

B02901093 吳岳

一、實作

- Dlist

在Dlist 的實作上，我只有在Dlist的Class裡面額外加一個deleteNode的Function，額外去處理Clear跟Erase時需要刪掉Node的狀況。

Sort的部分是用iterated sort，time complexity 是O(n^2)

整體的實作並沒有太多的困難，很多做法在老師的講義跟作業pdf檔裡就有指示，唯二要注意的是empty 跟 node指向_root的條件。這些在進入function時就要先加以處理。(p.s : code裡面也沒有特別用到_isSorted)

- Array

裡面比較困難的function就屬push_back了，要特別考慮到_data = 0 (空)、_size = _capacity兩種例外。前者需要把初始值設定好(_capacity = 1, _data = 1); 後者我會用兩倍的capacity額外**new** 一個新的殼，把原本的資料放到那個殼裡面，再將原_data **delete**，將殼指向_data。

其他function相對簡單，我也沒有額外新增function。erase跟clear 的部分也不需要額外刪data，只要把_size減1，並注意別超出_capacity跟_size。
(p.s : code裡面也沒有特別用到_isSorted)

- BST

這做的時間是前兩個class加起來的兩倍吧，最後也沒有完全做出來，在**random delete時會segmentation fault**。

在class iterator 裡面，我用到_**trace**跟_**dir**(left = 0, right = 1)兩個額外的element，用**addTrace**去push_back _trace&_dir。**min**、**max**、**succ**、**pred**

在這個class面也有。size()去得到_trace的size(深度)。clear()用網路上找的swap的方法更新_trace & _dir 。

以operator overloading 來說，比較困難的是++ 跟 - - 的情形，需要額外處理沒有succ或pred的狀況，需用_trace 跟 _dir 反推，並將_trace & _dir pop_back。

在class BSTree裡，我有額外用到**getParent**、**findNode**、**nodeInsert**、**clearNode**，以下特別解釋：

BSTreeNode<T>* **getParents**(BSTreeNode<T>* n)：
顧名思義，從_root比對_data的大小找到對應的_node，再取其parent的node回傳，在pop_front、pop_back、erase會用到。

BSTreeNode<T>* **findNode**(const T& x)：
顧名思義，做法跟getParents很像，(其實可以跟getParents合併，但為了辨別function功能並方便debug還是作罷)，在erase時用到。

BSTreeNode<T>* **nodeInsert**(const T& x, BSTreeNode<T>* n = 0)：
recursive function, 從insert接出來的子function，有額外考慮equal value跟empty 的條件。

void **clearNode**(BSTreeNode<T>* n):
recursive function, 從clear接出來的子function。

二、實驗預期

根據reference code跑4 個test還有老師講義所述，預期Array所需時間跟記憶體都比Dlist少。(因為BST)

粗估BST在adtAdd時較慢，delete跟Dlist差不多。

三、比較及討論

為了區分差異，最後我並未使用O3提高sorting的速度。在跑高數量級的指令(處理10^4級以上數據)時，Array跟Dlist的差異有被凸顯出來。

BST是三個方法裡面操作最複雜的，然而他最大的優勢就是本身已將data sort完畢，insert跟delete都可以維持既有的規則(雖說會有inbalanced BST的問題)。

而Array之所以可在那麼多層面完勝Dlist，是因為delete跟insert之後並未要求排列順序，否則array 在 sort時需要copy 過多記憶體的問題就會被凸顯出來。另外，array的_data在被delete之後從最後面補資料的做法，也大大減少array資料平移的時間跟memory cost (copy)。

四、心得

最後還是沒有將BST完全做出來，下次真的要痛定思痛開始用Debugger而非Cout 大法了...