DSnP HW5

Abstract Data Structures

姓名：馬健凱　　學號：b07901108

目錄

Array 實作 **1**

Doubly linked list 實作 **1**

Sort 1

Binary search tree 實作 **2**

Data members in BSTreeNode<T> 2

Data members in BSTree<T> 2

BSTree<T> constructor 2

Insert 2

Erase 3

How I implement BSTree<T>::iterator 4

實驗 **5**

General performance 5

Sort : Array vs DList 5

1. Array 實作

因為平常不需考慮資料排序，所以pop_front或erase刪除元素後，實作上用最後一個元素補它的空位。

2. Doubly linked list 實作

a. Sort

Sorting用到了QuickSort，用recursive的方式將DList分為兩邊(myPartition)。實作上以每個sublist的頭(稱為myHead)為pivot，比myHead->_data小就往前面移動。移動方式是與前面交換_data，直到與myHead->_data交換後結束。

3.Binary search tree 實作

a. Data members in BSTreeNode<T>

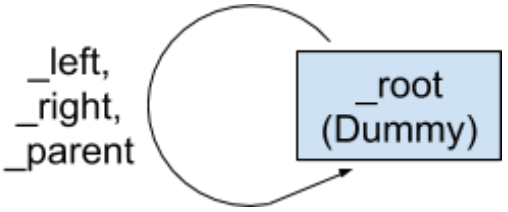
T _data存放資料。BSTreeNode<T>* _parent, _left, _right記錄相鄰的節點。

b. Data members in BSTree<T>

BSTreeNode<T>* _root記錄tree的root。

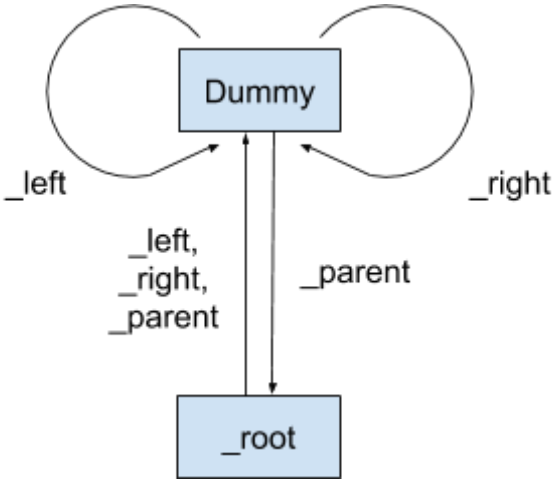
size_t _size記錄節點個數，每存入一筆資料就加1，每刪除一筆資料就減1。

c. BSTree<T> constructor

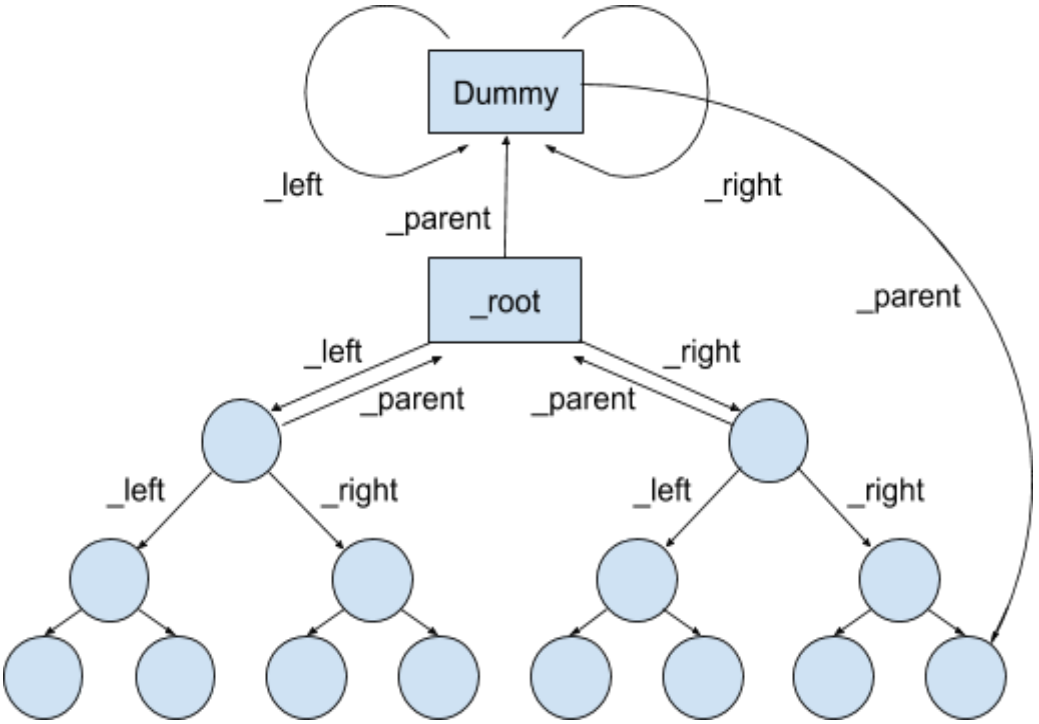


除了_root以外，一開始new一個node當作dummy node，將其_parent, _left和_right都指到自己。此時，將_root設為dummy。

d. Insert



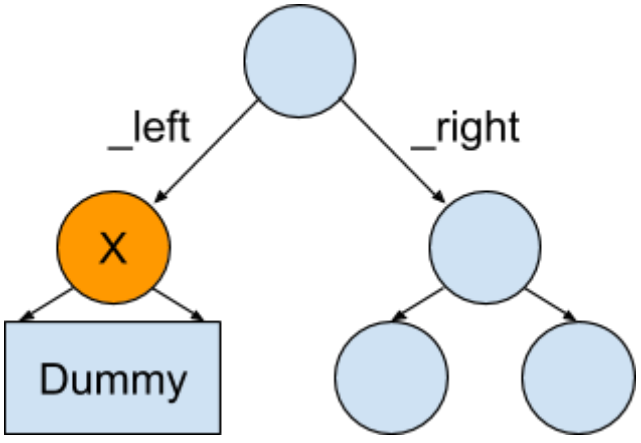
第一筆資料存入的時候，將其設為_root，將其_parent, _left和_right都指到Dummy。將Dummy的_parent指到最大資料的node，即_root。如上圖。



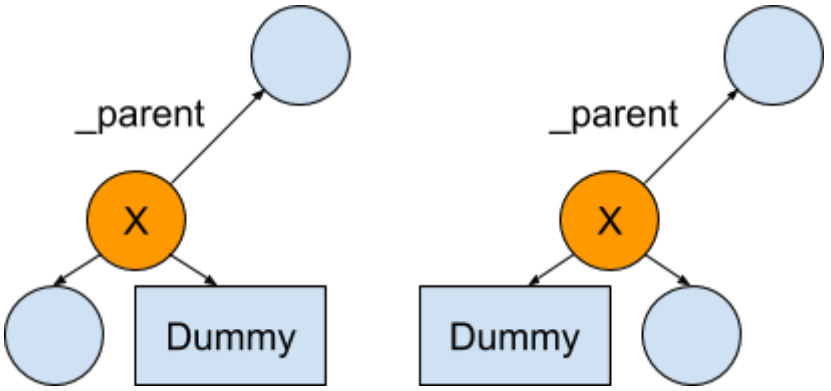
之後存入資料的時候，持續與走到的node的資料比大小，小的往左，大的往右，直到走到Dummy為止。新的node的_left和_right都指到Dummy，_parent指到前一個走到的node。確保Dummy指到資料最大的node。

最後BSTree<T>的結構大致如上圖，底下的箭頭代表_left和_right，圖底下12個node的_parent省略。

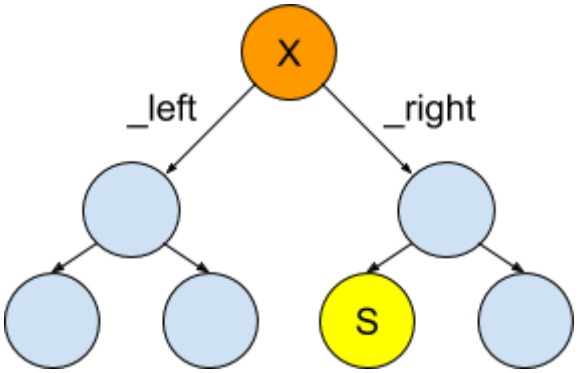
e. Erase



要刪掉的node標示為X。最簡單的情形是X左右指到的node都是Dummy。直接把X的_parent指到Dummy，再刪掉X就好了。假如X是_root，那就變回只剩Dummy的情況。



另一種情況是X左右指到的node其中一個是Dummy，那就把非Dummy那邊的node跟X的_parent連在一起，再刪掉X就好了。假如X是_root，那就把非Dummy那邊的node設成_root。



最後一種情況是X左右指到的node都不是Dummy。從X的_right下去找最小的node，即為X的Successor，標示為S。將S的資料移到X，再刪掉S就完成了。

pop_front, pop_back就是分別刪掉資料最小和最大的node，clear是不斷做pop_back。另外，要確保_root的_parent指到Dummy，Dummy的_parent指到最大的node。

f. How I implement BSTree<T>::iterator

寫了兩個private function : bool isDummy, bool isRoot來判斷node是不是dummy或是root，但是iterator不需要記得_root。

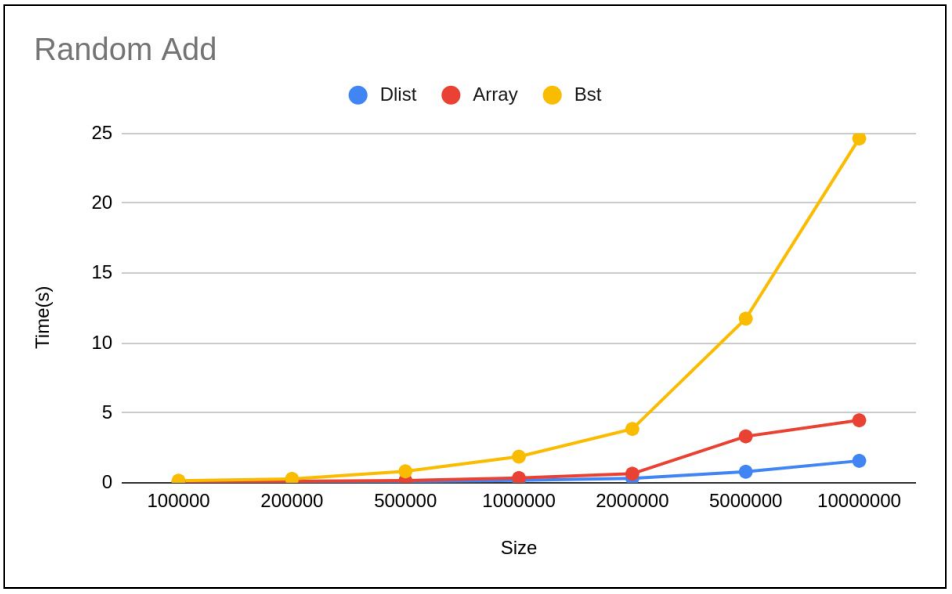
```
bool isDummy(BSTreeNode<T>* n) const
{ return (n == n->_left && n == n->_right); }

bool isRoot(BSTreeNode<T>* n) const
{ return isDummy(n->_parent); }
```

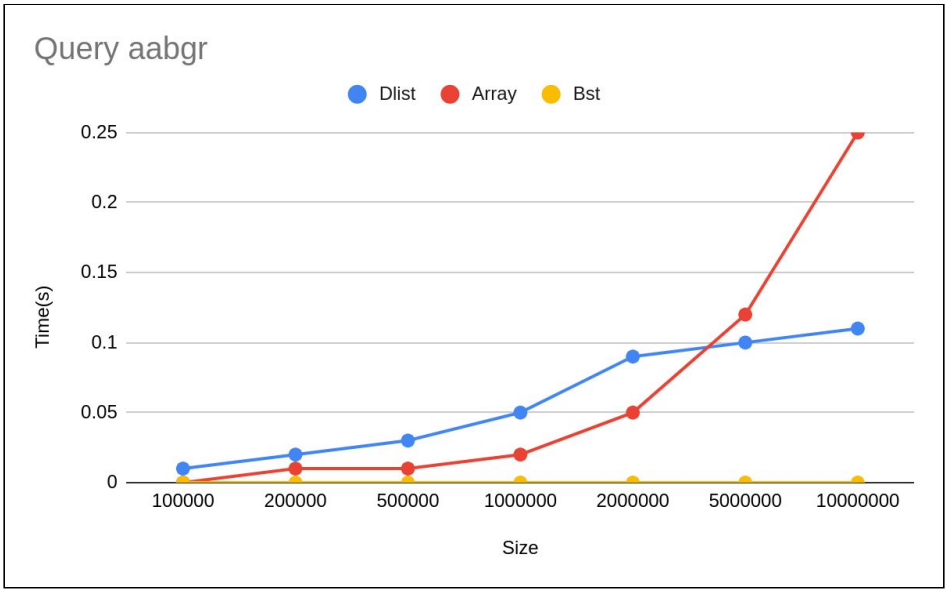
另外，在BSTree<T>和BSTree<T>::iterator都寫了BSTreeNode<T>* findMin和BSTreeNode<T>* findMax兩個public function，雖然程式碼不太一樣，但是都用來找某個node底下最小或最大的node。

4. 實驗

a. General performance



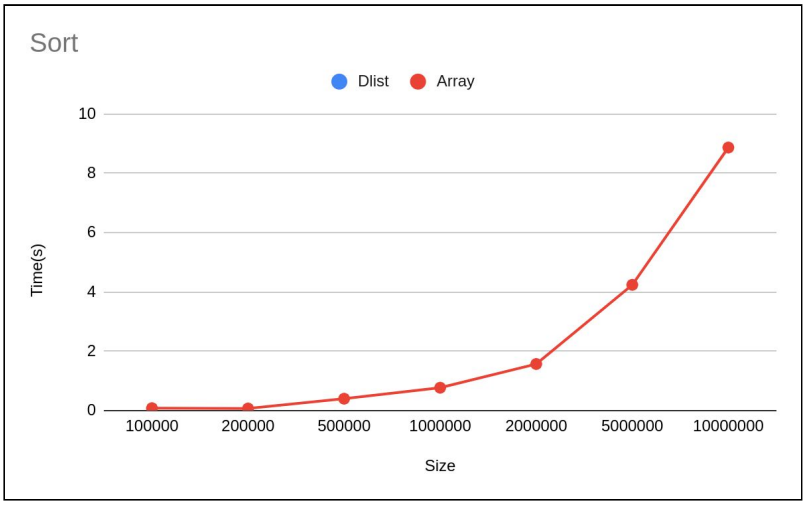
不知道為什麼，我的bst比其他兩個慢了很多。後來才發現我usage用錯了。後來在adta -r的下一行也加上usage，就正常了。



原始數據：[adtPerf]

b. Sort : Array vs DList

因為dlist的sort實在跑太久了(超過一分鐘)，所以沒有畫在表上。


Sort
Dlist
Array
10
8
6
4
2
0
Time(s)
100000
200000
500000
1000000
2000000
5000000
10000000
Size