DSnP HW5 Report

B06705033 資管三 莊海因

一、 資料結構實作

資料結構 / Member function	**Dynamic array**	**Doubly linked list**	**Binary search tree**
empty()	檢查_size==0	檢查初始值_head==_head->next	檢查 size==0
size()	回傳_size	每次以 iterator traverse 計算	回傳_size
push_back() /insert()	push_back()：如果 _size==_capacity，用 expand()擴大 array，指派 _data[size]=x，且 _size++，_isSorted 狀態為 false	push_back()：若 empty()，新增 node 與 dummy node 為彼此的_next 與_prev，並將此 node 設為_head。 否則，新增一個 node，_prev 為最後一個 element，_next 為 dummy node	insert()：insert 從 root 開始，根據傳入的 data x 大於/小於目前 node 決定該往右/左走，直到找到 NULL 成為 position 為止。
pop_front()	若_size>=2，_data[0]=_data[_size-1] 讓第一個 element 被最後一個取代，且_size--，_isSorted 狀態為 false	將第二個 element 的_prev 設為 dummy node，dummy node 的 next 為第二個 element，delete _head 後新_head 為第二個 element	erase(begin())，begin()為 BST 中 Leftmost（整棵樹最左邊的 node）的 iterator
pop_back()	_size--	同上將最後一個 element 以倒數第二個與 dummy node 重新連結取代，並 delete	erase BST 中 Rightmost（整棵樹最右邊的 node）的 iterator
erase(iterator pos)	*pos=_data[_size-1]，使 pos 的_data 成為最後一個 element 的_data，且 _size--	將 pos 指向的 node 的前一個與下一個 element 互相連結後，刪除此 node	1.欲刪除之 node 沒有 child pointer； 2.欲刪除之 node 只有一個 child pointer(不論是 leftchild 或 rightchild)； 3.欲刪除之 node 有兩個 child pointer。 取得 iterator pos 指向的 node，分別判斷處理三種情況。
erase(const T& x)	使用 iterator 找到 x 的 pos i，傳入上一個 function: erase(i)	使用 iterator 找到 x 的 pos i，傳入上一個 function: erase(i)	使用 iterator 找到 x 的 pos i，傳入上一個 function: erase(i)

find()	尋找_data=x 的 iterator，並回傳找到的 iterator	尋找_data=x 的 iterator，並回傳找到的 iterator	將 current node 指向 root，根據搜尋目標 data x 小於/大於 current，決定 current 該往左/右繼續搜尋，找到時傳回 current 的 iterator
clear()	_size=0	iterator traverse 同時刪除每個 node，並將 list 還原為未初始化狀態	iterator traverse 同時刪除每個 node
sort()	若 isSorted==false，使用 STL 內建 sort()，並將 isSorted 設為 true	若 isSorted==false，使用 _quickSort()排序	Insert 時以同時排序，不需另行排序

二、 實作補充說明

1. Array

(1) expand()：當 capacity 不足時，增加 capacity 給 array 使用（每次幅度為兩倍）。

(2) size()：在 push_back()、pop_front()、pop_back()、erase(iterator pos)、clear()等需要更改 size 大小的 function 進行加減。

2. DList

(1) _quickSort()：協同 swap()、partition()來進行排序。swap()交換 element 位置，partition(DListNode<T> *l, DListNode<T> *h)以 h 為 pivot element，將 l 到 h 的 list 之中小於 pivot 的放到 pivot 左邊，大於 pivot 的放到右邊，形成排序完成的 list。最後_quickSort()對 list 作 recusive implementation。

Quick Sort 的平均 time complexity 為 O(nlogn)，但當 list 已經排序完成有 worst case O(n^2)，但比起 Bubble Sort(O(n^2))已快很多。

(2) size()：每次使用必須重新 traverse 過整個 list（time complexity:O(n)）。

3. BST

Node：

BST 的 node 為 class BSTreeNode<T>，包含指向兩 child 的_left、_right，指向 node 之 parent 的 _parent 與含有資料本身的_data。Constructor 傳入四參數_data、_parent、_right、_left，其中後三者的 default 值為 0。

Tree：

BSTree 包含指向此樹之 root 的_root、指向 dummy node 的_dummy 以及計數的_size。dummy node 將永遠為整棵樹中最右邊的 node 之右邊的 child。

Iterator：

對 BSTree 進行 inorder Traversel 的指標。

Functions：

Rightmost、Leftmost 分別會回傳以傳入之 node 為 root 的 BSTree 之最右邊、最左邊的 node。

Predecessor、Successor 分別回傳傳入之 current node 以 inorder traverse 尋訪的上一個、下一個 node。

erase(iterator pos)會遇見三種情況：

1.欲刪除之 node 沒有 child pointer；

2.欲刪除之 node 只有一個 child pointer(不論是 leftchild 或 rightchild)；

3.欲刪除之 node 有兩個 child pointer。

先把要刪除之node的pointer調整成「至多只有一個child」的node（找到它的Successor當替身，由情況3變成情況1/2），接著將欲刪除之node的child指向新的parent，欲刪除之node的parent指向新的child；若記憶體是「替身」(情況3)，則再把替身的資料放回BST中。

三、 實驗

1. **實驗設計：**建立 dofile 執行以下指令以測試效能
   (1) adta –r 100000, usage　顯現處理 push_back()/insert() 的效能
   (2) adtp, usage　顯現以 traverse 效能
   (3) adts, usage　顯現 sort()/quickSort() 效能
   (4) adts, usage　顯現排序後的資料再排序效能
   (5) adtd –f 10000, usage　顯現 pop_front() 效能
   (6) adtd –b 10000, usage　顯現 pop_back() 效能
   (7) adtd –a, usage　顯現 clear() 效能

2. **實驗結果：**

Array

(1)
```
Period time used : 0.07 seconds
Total time used  : 0.07 seconds
Total memory used: 6.117 M Bytes
```

(2)
```
Period time used : 0.09 seconds
Total time used  : 0.16 seconds
Total memory used: 6.121 M Bytes
```

(3)
```
Period time used : 0.19 seconds
Total time used  : 0.35 seconds
Total memory used: 6.137 M Bytes
```

(4)
```
Period time used : 0 seconds
Total time used  : 0.35 seconds
Total memory used: 6.137 M Bytes
```

(5)
```
Period time used : 0 seconds
Total time used  : 0.35 seconds
Total memory used: 6.137 M Bytes
```

(6)
```
Period time used : 0 seconds
Total time used  : 0.35 seconds
Total memory used: 6.137 M Bytes
```

(7)
```
Period time used : 0 seconds
Total time used  : 0.35 seconds
Total memory used: 6.137 M Bytes
```

Array–ref

(1)
```
Period time used : 0.01 seconds
Total time used  : 0.01 seconds
Total memory used: 6.102 M Bytes
```

(2)
```
Period time used : 0.07 seconds
Total time used  : 0.08 seconds
Total memory used: 6.102 M Bytes
```

(3)
```
Period time used : 0.03 seconds
Total time used  : 0.11 seconds
Total memory used: 6.121 M Bytes
```
(4)
```
Period time used : 0 seconds
Total time used  : 0.11 seconds
Total memory used: 6.121 M Bytes
```

(5)
```
Period time used : 0 seconds
Total time used  : 0.11 seconds
Total memory used: 6.133 M Bytes
```
(6)
```
Period time used : 0 seconds
Total time used  : 0.11 seconds
Total memory used: 6.133 M Bytes
```

(7)
```
Period time used : 0 seconds
Total time used  : 0.11 seconds
Total memory used: 6.133 M Bytes
```

Array 的整體速度都很快，除了 sort 相對要花較久的時間，但第二次 sort 由於已經標記為已排序，便沒有再多花時間。

DList

(1)
```
Period time used : 0.02 seconds
Total time used  : 0.02 seconds
Total memory used: 4.77 M Bytes
```
(2)
```
Period time used : 0.09 seconds
Total time used  : 0.11 seconds
Total memory used: 4.773 M Bytes
```

(3)
```
Period time used : 0.22 seconds
Total time used  : 0.33 seconds
Total memory used: 4.773 M Bytes
```
(4)
```
Period time used : 0 seconds
Total time used  : 0.33 seconds
Total memory used: 4.773 M Bytes
```

(5)
```
Period time used : 0 seconds
Total time used  : 0.33 seconds
Total memory used: 4.773 M Bytes
```
(6)
```
Period time used : 0.01 seconds
Total time used  : 0.34 seconds
Total memory used: 4.773 M Bytes
```

(7)
```
Period time used : 0 seconds
Total time used  : 0.34 seconds
Total memory used: 4.773 M Bytes
```

DList-ref

(1)
```
Period time used : 0.01 seconds
Total time used  : 0.01 seconds
Total memory used: 4.723 M Bytes
```
(2)
```
Period time used : 0.06 seconds
Total time used  : 0.07 seconds
Total memory used: 4.734 M Bytes
```

(3)
```
Period time used : 102.7 seconds
Total time used  : 102.8 seconds
Total memory used: 4.734 M Bytes
```
(4)
```
Period time used : 0 seconds
Total time used  : 102.8 seconds
Total memory used: 4.734 M Bytes
```

(5)
```
Period time used : 0 seconds
Total time used  : 102.8 seconds
Total memory used: 4.734 M Bytes
```
(6)
```
Period time used : 0 seconds
Total time used  : 102.8 seconds
Total memory used: 4.734 M Bytes
```

(7)
```
Period time used : 0.01 seconds
Total time used  : 102.8 seconds
Total memory used: 4.734 M Bytes
```

DList 的整體速度都很快，但 sort 相對要花較久的時間，與 reference 比較花費時間的話，在此實驗中使用的 Quick Sort 顯然對於效能的提升有顯著影響，reference 可能使用較為費時的 Bubble Sort。

BST

(1)
```
Period time used : 0.25 seconds
Total time used  : 0.25 seconds
Total memory used: 4.789 M Bytes
```
(2)
```
Period time used : 0.11 seconds
Total time used  : 0.36 seconds
Total memory used: 4.789 M Bytes
```

(3)
```
Period time used : 0 seconds
Total time used  : 0.36 seconds
Total memory used: 4.789 M Bytes
```
(4)
```
Period time used : 0 seconds
Total time used  : 0.36 seconds
Total memory used: 4.789 M Bytes
```

(5)
```
Period time used : 0.01 seconds
Total time used  : 0.37 seconds
Total memory used: 4.789 M Bytes
```
(6)
```
Period time used : 0 seconds
Total time used  : 0.37 seconds
Total memory used: 4.789 M Bytes
```

(7)
```
Period time used : 0.02 seconds
Total time used  : 0.39 seconds
Total memory used: 4.789 M Bytes
```

BST-ref

(1)
```
Period time used : 0.04 seconds
Total time used  : 0.04 seconds
Total memory used: 4.762 M Bytes
```
(2)
```
Period time used : 0.19 seconds
Total time used  : 0.23 seconds
Total memory used: 4.793 M Bytes
```

(3)
```
Period time used : 0 seconds
Total time used  : 0.23 seconds
Total memory used: 4.793 M Bytes
```
(4)
```
Period time used : 0 seconds
Total time used  : 0.23 seconds
Total memory used: 4.793 M Bytes
```

(5)
```
Period time used : 0 seconds
Total time used  : 0.23 seconds
Total memory used: 4.793 M Bytes
```
(6)
```
Period time used : 0 seconds
Total time used  : 0.23 seconds
Total memory used: 4.793 M Bytes
```

(7)
```
Period time used : 0.02 seconds
Total time used  : 0.25 seconds
Total memory used: 4.793 M Bytes
```

BST 整體來說速度快且穩定，沒有特別費時的效能。

結論：三種資料結構的兩種 program 作比較，Dynamic Array 的效能表現最佳，Binary Search Tree 優化實作後將會次之（可能我的寫爆了 QQ），Doubly Linked List 雖然墊底，但只要找到適當的排序法，效能將明顯提升。