DSnP HW5 Report

B03901026 許凱傑

一、資料結構的實作

	DList	Array	BST
Node	每個 node 有3個 data member：_prev, _next, data	不用 implement node	每個 node 有4個 data member：_p, _left, _right, data
資料結構	_head, _isSorted 有額外 dummy node ，串成一個圈	Data array's pointer, _size, _capacity, _isSorted Random access	_root, _size 有額外 dummy node
iterator	++：_node 指向_next --：_node 指向_prev	因為 random access 可以 overload+，++只需要將_node 指向下一記憶體位址；--則是上一記憶體位址	++：find successor --：find predecessor
size()	O(n)	O(1)	O(1)
add()	push_back(data) O(1)	push_back(data) O(1)	insert(data) O(h)~O(log n)
pop_front()	O(1)	O(1)	O(1)
pop_back()	O(1)	O(1)	O(1)
empty()	O(1)	O(1)	O(1)
erase(pos)	O(1)	O(1) always move last element to the deleted one's location	O(h)~O(log n) However, command line getPos is O(n), so erase is far slower than add.
find()	O(n)	O(n)	O(h)~O(log n)
sort()	Quicksort O(n log n)	STL::sort	No need to implement
			preOrderPrint()
優點	不須使用連續記憶體空間。 不須事先指定大小。	Memory overhead 小。 記憶體位址都在同一 page 上。	所有資料都已排序好，無須 sort。 find()較快。 不須使用連續記憶體空間。 不須事先指定大小。
缺點	find()較慢。 Memory overhead 大。	find()較慢。 需事先指定大小(capacity)。	Add 速度較慢，erase 速度很慢(更明顯)。 Memory overhead 大。

DSnP HW5 Report

B03901026 許凱傑

QuickSort: O(n log n)

使用 Divide and Conquer 的演算法來實作。從數列中挑選一個基準點，大於基準的放一邊，小於的放一邊，如此循環最後可完成排序。Performance 跑 do2 很好只需大約 2 秒
若是 BubbleSort 大約 17 秒。

```
void quicksort(DListNode<T>* left , DListNode<T>* right , size_t size) const{
    if(size <= 1 )return;
    //find the medium data
    DListNode<T>* index = left;
    for(size_t i=1; i<= size_t(size/2) ; i++)
        index = index->_next;
    //set pivot
    T pivot = index->_data;
    //put the pivot data in the rightmost node
    myswap(index,right);
    //keep the position where new smaller data insert to
    DListNode<T>* swapindex = left;
    //use index run from left to right->_prev
    index = left;
    size_t leftlength = 0;
    while(index != right){
        if(index->_data <= pivot){
            //cerr<<index->_data<<endl;
            myswap(index , swapindex);
            swapindex = swapindex->_next;
            leftlength++;
        }
        index = index->_next;
    }
    //put the pivot data in the swapindex node
    //all left nodes have smaller data; right nodes have bigger
    myswap(swapindex,right);
    quicksort(left,swapindex->_prev,leftlength);
    quicksort(swapindex->_next , right , size-leftlength-1);
}
```

BST—delete 分成三種情形，只有左右都有 children 比較麻煩(詳細在註解裡面)

```
//only deal with parent
void transplant(BSTreeNode<T>* _delete, BSTreeNode<T>* _replace){
    if(_delete->_p == 0)
        _root = _replace;
    else if(_delete == (_delete->_p)->_left)
        (_delete->_p)->_left = _replace;
    else
        (_delete->_p)->_right = _replace;
    if(_replace != 0)
        _replace->_p = _delete->_p;
}
```

```
void BSTDelete(BSTreeNode<T>* target){
    if(target == 0)    return;
    if(target->_left == 0)
        transplant(target, target->_right);
    else if(target->_right == 0)
        transplant(target, target->_left);
    else if(target->_right == _dummy){
        BSTreeNode<T>* newmax = maximum(target->_left);
        newmax->_right = _dummy;
        _dummy->_p = newmax;
        transplant(target, target->_left);
    }
    else{
        //let y be the successor of target in the right subtree
        BSTreeNode<T>* y = target->_right;
        while( y->_left != 0 )
            y = y->_left;
        //if y's parent is not target, need to bridge the gap between target->_right & replacing node
        //y->_p & y->_right(since there is no y->_left) shoule be connected
        if(y->_p != target){
            transplant(y, y->_right);
            y->_right = target->_right;
            (target->_right)->_p = y;
        }
        //deal with target->_left
        y->_left = target->_left;
        (target->_left)->_p = y;
        transplant(target,y);
    }
    delete target;
    _size-- ;
}
```

DSnP HW5 Report

B03901026 許凱傑

二、實驗比較

1.實驗設計

```
adta -r 10000000
usage
adta -s kevin
usage
adtd -r 1
usage
adtd -f 1
usage
adtd -b 1
usage
adts
usage
adtd -s kevin
usage
adtd -a
usage
q -f
```

2.實驗預期

- 速度方面除了 sort、erase 特定 data、add 應該都是 ARRAY>DLIST>BST
- add 部分因為 array 可能會需要重開更大的記憶體空間，把資料複製過去，所以可能會比較慢，造成 DLIST> ARRAY>BST
- 記憶體用量應該是 DLIST≅BST>ARRAY

3.結果比較

- adta –r 10000000 的速度

	DLIST	ARRAY	BST
Time used	1.86 s	5.44s	18.24s
Memory used	609.6 MB	767.1MB	609.5 MB

推測 ARRAY 不像預測的用到最小記憶體是因為，會有短暫的時間新開的空間與舊的記憶體空間同時存在，而記錄該值。

- adtd –r 1

DLIST	ARRAY	BST
0.08s	0s	0.16s

- adta –r 1、adtd –f 1、adtd –b 1 速度幾乎都一樣(0s)
- adts

DLIST	ARRAY	BST
9.18s	4.23s	0s(不須 sort))

ARRAY 是 random access

- adtd –s kevin(跟 find 正相關)

DLIST	ARRAY	BST
0.02s	0.02s	0s

- adtd –a (跟 adtd 趨勢差不多)

DLIST	ARRAY	BST
0.2s	0s	2.31s