DSnP HW#5 Report

- Implementation

1. Array

   透過一般已固定大小的 array 實現，當已達資料量的存放上限(_capacity)，若欲再存入資料，則會再跟系統要一塊 2*_capacity 大小的新記憶體，將當前記憶體的資料 copy 給新的記憶體後，並釋放當前記憶體空間，故為動態配置。

   ✓ Iterator: 與一般 array 基本上無異。

2. Dlist

   將不連續的記憶體透過雙向 pointer 串聯成一列，並利用尾端的 dummy node 與_head 連接，可直接以_head -> _prev 走到 dummy node，因此實際上比較像是一圈而非一列。

   ✓ Iterator: 透過_prev 和_next 迭代。

   ✓ Sorting: 因已規定不可 copy data (i.e. in-place)，我選擇以 insertion sort 實作，理論上 best case: $O(n)$，average case: $O(n^2)$，worst case: $O(n^2)$

   ✓ Other notice: 最重要的是如何維持_head 指向正確的位置。
   push_back(): 當 empty()時，會讓_head 由原本的 dummy node 指向新增的 node。
   pop_back()在_size == 1 時，會讓_head 指回 dummy node。

3. BSTree

   每個 node 的 left subtree 中所有 node 的 data 皆 < 本身 data；每個 node 的 right subtree 中所有 node 的 data 皆 >= 本身 data。

```
T _data;
BSTreeNode<T>* _parent
BSTreeNode<T>* _left;
BSTreeNode<T>* _right;
```

   每個 node 都存著其 parent 和 children 的位置，_parent 可供我們往上搜尋資料。

```
private:
  BSTreeNode<T>* _root;
  BSTreeNode<T>* _dum;
  size_t _size;
```

   在 bst 中，當 constructor 被呼叫，_dum 即同時產生，_root 暫且指到 NULL。

   ✓ Iterator:
   ++ : 先往右下找 successor，若_right == NULL，再往上是否有任何 node 為 left child，並 return 其 parent。
   --: 先往左下找 predecessor，若_left == NULL，再往上是否有任何 node 為 right child，並 return 其 parent。
   begin(): data 最小的 node (i.e. leftmost)，但當 tree 為 empty()，begin() = _dum。

end(): _dum，data 最大的 node 的 right child，但當 tree 為 empty()，其_parent = NULL。
其中需要注意_root -> _parent = NULL。

- ✓ Other notice:
  bst 所需 maintain 的性質繁多，尤其是_root 與_dum 的指向。
  insert(x): 當 tree 為 empty()時，_root 指向新的 node。Insert(x)可能改變_dum 的_parent，因此我會先將_dum 拔掉，之後 insert 完再把_dum 接回。
  erase(pos): 同樣我先將_dum 拔除，待整個 erase process 結束再接回。這裡我們分三個 case 討論：two children、one child、no child，其中 case 1 最後都會化成 case 2 or 3 因為 successor 不可能有 two children。還要小心最後真正 erase 的 node 是否為 root，若是則必須調整_root 的指向。

- Experiment

Command	Array	Dlist	BSTree
adta –r 10000	0.01 s	1.2 s	0.01 s
	1.672M bytes	1.426M bytes	1.801M bytes
adts	0.02 s	2.2 s	0 s
adtd –r 5000	0 s	0.25 s	0.72 s
adta –r 20000	0.01 s	6.99 s	0.03 s
	2.609M bytes	2.629M bytes	2.633M bytes
adts	0.05 s	13.92 s	0 s
adtd –r 10000	0 s	1.51 s	4.33 s

- Analysis

1. Array
   - ✓ 在新增資料時，跟系統要的空間是以指數成長
   - ✓ 用 STL::sort(…)約為$O(n \log n)$
   - ✓ 隨機存取能力佳
2. Dlist
   - ✓ 記憶體主要用在存 pointers
   - ✓ Insertion sort 約為$O(n^2)$
   - ✓ 隨機存取透過 pointer 從頭找尋，約$O(n)$
3. BSTree
   - ✓ 記憶體主要用在存 pointers
   - ✓ Insert 的 process 即經過 sorting，故不需要額外花時間
   - ✓ 隨機存取能力不佳