一、資料結構的實作

1. 三種資料結構的比較：

Doubly linked list 是一種線性的資料結構，其中的每一個 node 都有指向前後的指標。看起來是很直覺也好實作的資料結構，因為新增或刪除 node 都只需要重新連結指標即可；然而缺點是因為無法直接 access 中間的 node，只能用複雜度O(n) 的 sequential search，不能用複雜度 O(log n) 的 binary search。

Array 因為常常被使用，直覺上覺得他好像是新手在用的，不好用，例如無法動態配置。但是這次作業寫得其實比較像 vector，是動態陣列，加上陣列可以任意存取任一筆資料，可以使用 binary search，效能實際上應該比同為線性的 doubly linked list 好。

Binary Search Tree 則非線性資料結構了。基本的規則是，一個 node 的左邊一定比 node 的值小，而右邊一定比較大。主要的好處是，可以用遞迴來實作 search 和 traverse，既直覺 code 又簡潔，更享有 divide and concur 的好處。不過找 successor/processor 就比較冗長一些。

2. 我的實作方式與原因：

dlist 和 array 因為有 code 的框架，又有 pdf 的提示，基本上沒有發揮太多創意。Dlist 利用 _next 與 _prev 指向下一個與上一個 node，並且用一個 dummy node 來表示最後，並將 list 連起來。Array 則是用一個 class 將一個 array 包起來，並且在插入值而發現空間不夠時，擴大 array 的容量，達到動態的效果。

比較有實驗的部分是，寫 array 的 insertion 時，一開始採取的方法是先把資料塞進去再 merge sort。然而發現這種 implementation 不好，insert 5000 個值就要花 50 秒左右；後來改為直接 sequential search 後，找到恰當的位置塞入。前者的複雜度應該是 O(n log n)，而後者應為 O(n)，果然有差。

bst 我選用有 parent 的方式。好處是寫的時候比較直覺，不過每次有 insert 或 delete 就要多整理一次指標；也要小心 parent 為空的情況 (root)。另外，end 我則是在 iterator 裡加上 bool _isEnd 的 member，如果是在最後一個的下一個，值為 true，反之則為 false。好處也是直覺，而且因為不是多製造一個 node，比較不會有 segmentation fault 的問題。

二、實驗比較

1. 實驗設計

寫一個 do file 如下：

```
 adta -r 1000
usage
adtp
usage
adtd -all
usage

adta -r 10000
usage
adtp
usage
adtd -all
usage

adta -r 100000
usage
adtp
usage
adtd -all
usage

q -f
```

2. 實驗預期

所花時間：dlist > array >bst

所用記憶體：array > bst > dlist

3. 結果比較與討論

```
dlist
adt> adta -r 1000

adt> usage
Period time used : 0.01 seconds

adt> adtp
adt> usage
```

Period time used : 0 seconds

adt> adtd -all

adt> usage
Period time used : 0 seconds

adt>
adt> adta -r 10000

adt> usage
Period time used : 1.51 seconds

adt> adtp
adt> usage
Period time used : 0 seconds

adt> adtd -all

adt> usage
Period time used : 0 seconds

adt>
adt> adta -r 100000

adt> usage
Period time used : 209 seconds

adt> adtp
adt> usage
Period time used : 0.04 seconds

adt> adtd -all

adt> usage
Period time used : 0.02 seconds

Total time used : 210.6 seconds
Total memory used: 3.871 M Bytes

array
  adt> adta -r 1000
  adt> usage

```
Period time used : 0.01 seconds

adt> adtp
adt> usage
Period time used : 0.01 seconds

adt> adtd -all

adt> usage
Period time used : 0 seconds

adt>
adt> adta -r 10000

adt> usage
Period time used : 0.83 seconds

adt> adtp
adt> usage
Period time used : 0 seconds

adt> adtd -all

adt> usage
Period time used : 0 seconds

adt>
adt> adta -r 100000

adt> usage
Period time used : 101.2 seconds

adt> adtp
adt> usage
Period time used : 0.04 seconds

adt> adtd -all

adt> usage
Period time used : 0 seconds

Total time used  : 102.1 seconds
Total memory used: 5.027 M Bytes
```

```
bst
adt> adta -r 1000

adt> usage
Period time used : 0 seconds

adt> adtp
adt> usage
Period time used : 0 seconds

adt> adtd -all

adt> usage
Period time used : 0 seconds

adt>
adt> adta -r 10000

adt> usage
Period time used : 0.08 seconds

adt> adtp
adt> usage
Period time used : 0 seconds

adt> adtd -all

adt> usage
Period time used : 0.01 seconds

adt>
adt> adta -r 100000

adt> usage
Period time used : 1.11 seconds

adt> adtp
adt> usage
Period time used : 0.06 seconds

adt> adtd -all

adt> usage
```

Period time used : 0.06 seconds

Total time used ：1.32 seconds
Total memory used: 4.516 M Bytes

結果的確是 bst 大勝 array，而 dlist 墊底而且輸很多。而記憶體方面，由於 list 幾乎沒有沒用到的東西（除了 dummy node），而 array 一旦刪除便會空了很多空間， bst 則是會有空的指標，因此記憶體使用上，array > bst > dlist。