DSnP HW5

Implementation of some ADT

* Dynamic Array

Just an normal STL-vector-like container.

Uses <algorithm>::sort().

Time complexties:

**Insert random place**: O( n), since we may need to move all our data to other place.

**Delete random place**: O( 1), since we just swap it with back(), and then pop\_back().

**Sort**: O( ln( n )), since we use STL sort, and didn’t utilize the mutable variable is\_sorted.

(which shall not be used, since we have functions that return non-const iterators, and thus our data could be modified even when there’s no element deleted/inserted – they are modified through these iterators.)

**Size**: O(1), since dynamic arrays itself have to maintain its size and capacity, and we could just return them.

* Doubly Linked List

Basically it’s a ring. Contains a dummy node.

Uses merge-sort variant for linked lists.

**Push Back**: O(1), since we just need to modify fixed amount of pointers.

**Delete random place**: O(n), since we have to find where to delete. After the item was found, it’s constant time operation.

**Sort**: O( ln( n)), since we used merge sort. BTW we just modify the pointers to sort the list, so there’s no copy or move (c++11 or later) constructor used when sort. Which I think is handy.

**Size**: O(n), since I didn’t maintain the size, I have to traverse the whole list.

* Binary Search Tree

A Red-Black tree variant. Uses nullptr instead of NIL, that is, there’s no dummy tree node.

Shall not be very stable actually, but at least it passed do1 to do4.

**Insert**: O( ln(n)), since R-B tree is balanced, so insert time complexity for trees O(height) is just O( ln(n)).

**Delete random index**: O(n), since I used in-order traversal.

**Delete random key**: O( ln(n) ), since R-B tree is balanced, so delete time complexity for trees O(height) degenerates to O(ln(n)).

**Sort**: O(1), since R-B tree itself is a binary search tree, which is sorted.

**Size**: O(1), since I have an size data field for the whole tree.

* Some Experiments: (uses g++ -g –O2)
* Doubly linked list, random add, sort, quit. String length is 6.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Item # | 1000000 | 2000000 | 3000000 | 4000000 |
| Period Time (ms) | 1320 | 2780 | 4410 | 5900 |

* Binary Search tree, random add. String length is 6.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Item # | 100w | 200w | 300w | 400w |
| Period Time  (ms) | 1690 | 3770 | 6130 | 8660 |