Data Structures Assignment 4 report

2021100045 컴퓨터학과 박승원

1) How did I modify test cases in main.cpp

As the instruction described, I need to make 4 cases that is differed from sizes of key sets(1,000 / 10,000 / 100,000 / 1,000,000).

Then, I should divide it into keys that are listed in skewed order or in random order. Next, I divided it into insert operation and find operation. And then, I divided it into Search Tree's timing analysis and AVL Tree's. Also I fixed skewed order as an increasing order.

So the schematic of my test case implementation is like this.

```
Test Cases Implementation
 1. size = 1000
  1-(1). Skewed Order(Increasing Order)
    (a). Insert Operation
        (i) Search Tree : n seconds.
        (ii) AVL Tree : n seconds.
            -> AVL Tree Insertion was faster than Search Tree's.
    (b). Find Operation
        (i) Search Tree : n seconds.
        (ii) AVL Tree : n seconds.
            -> AVL Tree Find was faster than Search Tree's.
  1-(2). Random Order
    (a). Insert Operation
        (i) Search Tree : n seconds.
        (ii) AVL Tree : n seconds.
            -> Search Tree Insertion was faster than AVL Tree's.
    (b). Find Operation
        (i) Search Tree : n seconds.
        (ii) AVL Tree : n seconds.
            -> AVL Tree Find was faster than Search Tree's.
------------
2. size = 10000
  2-(1). Skewed Order(Increasing Order)
    (a). Insert Operation
        (i) Search Tree : n seconds.
        (ii) AVL Tree : n seconds.
            -> AVL Tree Insertion was faster than Search Tree's.
    (b). Find Operation
        (i) Search Tree : n seconds.
        (ii) AVL Tree : n seconds.
            -> AVL Tree Find was faster than Search Tree's.
  2-(2). Random Order
    (a). Insert Operation
        (i) Search Tree : n seconds.
        (ii) AVL Tree : n seconds.
            -> Search Tree Insertion was faster than AVL Tree's.
    (b). Find Operation
        (i) Search Tree : n seconds.
        (ii) AVL Tree : n seconds.
            -> AVL Tree Find was faster than Search Tree's.
------------
3. size = 100000
  3-(1). Skewed Order(Increasing Order)
    (a). Insert Operation
        (i) Search Tree : n seconds.
```

(ii) AVL Tree : n seconds.
-> AVL Tree Insertion was faster than Search Tree's.
(b). Find Operation
(i) Search Tree : n seconds.
(ii) AVL Tree : n seconds.
-> AVL Tree Find was faster than Search Tree's.
3-(2). Random Order
(a). Insert Operation
(i) Search Tree : n seconds.
(ii) AVL Tree : n seconds.
-> Search Tree Insertion was faster than AVL Tree's.
(b). Find Operation
(i) Search Tree : n seconds.
(ii) AVL Tree : n seconds.
-> AVL Tree Find was faster than Search Tree's.

---

4. size = 1000000
4-(1). Skewed Order(Increasing Order)
(a). Insert Operation
(i) Search Tree : n seconds.
(ii) AVL Tree : n seconds.
-> AVL Tree Insertion was faster than Search Tree's.
(b). Find Operation
(i) Search Tree : n seconds.
(ii) AVL Tree : n seconds.
-> AVL Tree Find was faster than Search Tree's.
4-(2). Random Order
(a). Insert Operation
(i) Search Tree : n seconds.
(ii) AVL Tree : n seconds.
-> Search Tree Insertion was faster than AVL Tree's.
(b). Find Operation
(i) Search Tree : n seconds.
(ii) AVL Tree : n seconds.
-> AVL Tree Find was faster than Search Tree's.

2) Time comparison by table

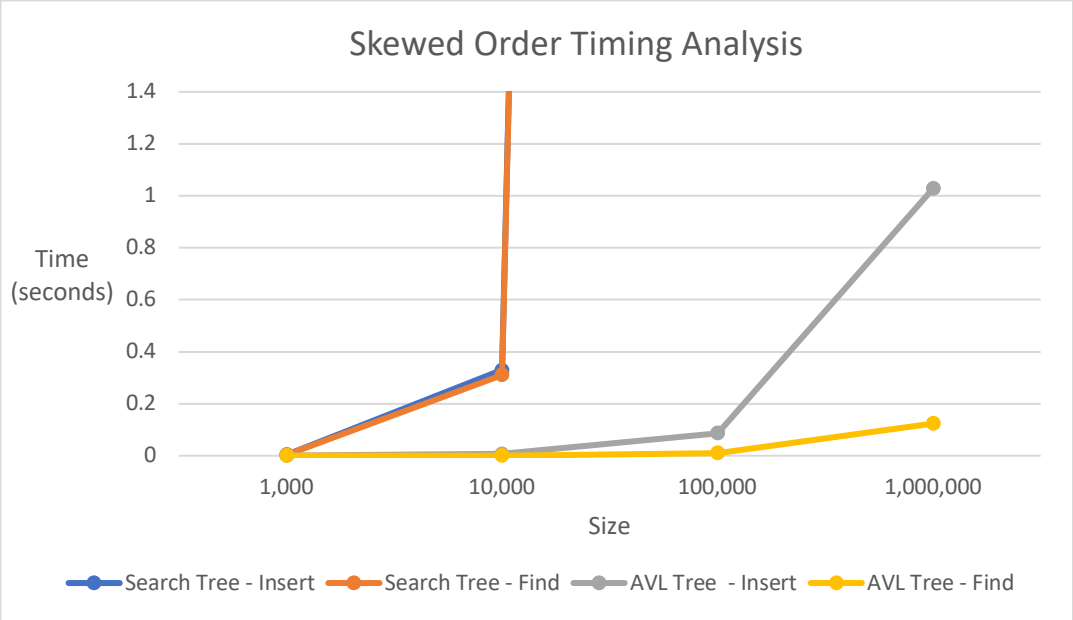
(*Test cases ran for 5 times and this is the average of the each case's results.)

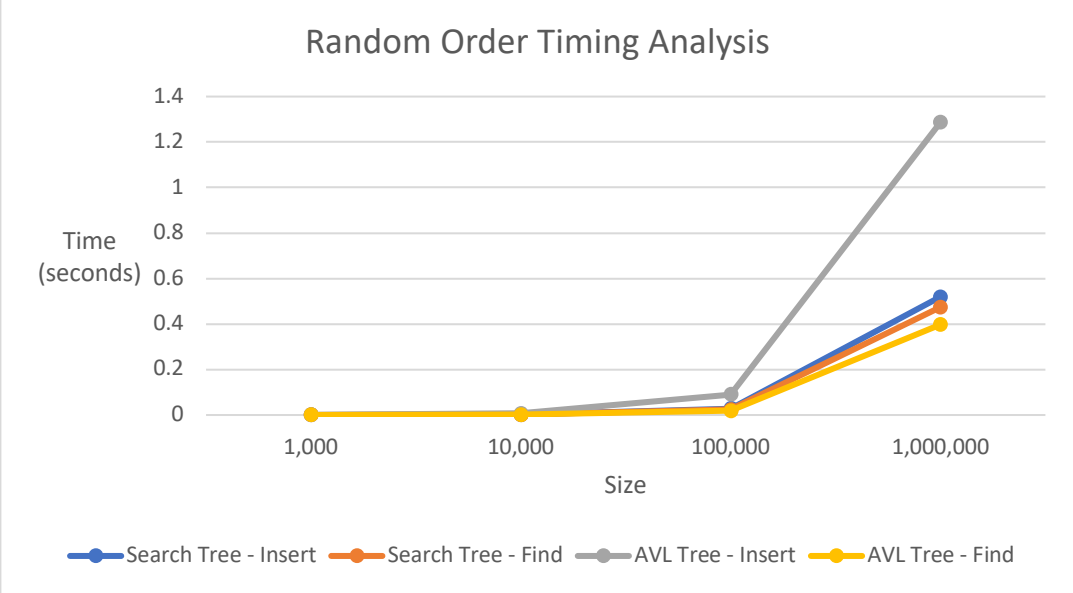
size	Tree Type	Skewed order - Insert (seconds)	Skewed order - Find (seconds)	Random order – Insert (seconds)	Random order - Find (seconds)
1,000	Search Tree	0.0032658	0.0029438	0.0001614	0.0001368
	AVL Tree	0.0005858	0.000116	0.0005536	0.000119
10,000	Search Tree	0.3288356	0.3103856	0.0019996	0.0016824
	AVL Tree	0.0072258	0.0009308	0.0070476	0.001398
100,000	Search Tree	34.32824	34.13608	0.0287628	0.0243844
	AVL Tree	0.0861992	0.0109368	0.0897476	0.0190552
1,000,000	Search Tree	Unable to compute!	Unable to compute!	0.5196918	0.4742352
	AVL Tree	1.027568	0.1237196	1.286722	0.3974736

(*Faster one)

3) Time comparison by graph

(*Test cases ran for 5 times and this is the average of the each case's results.)





4) Conclusion

Macroscopically, in every case, except "Random Order Insert Operation", AVL tree showed better performance than BST. In insert operation, AVL tree showed almost same performance no matter its keys were in skewed or random order. However, BST showed huge difference in insert operation. Because skewed case is the worst case of BST, skewed order insertion took way more time than random order insertion. Same as the insertion operation, find operation of BST in skewed ordered keys took more time than the case which keys are in skewed order. When finding random ordered keys in AVL Tree, it took more time than when finding skewed ordered keys. Because, it is much easier to find keys that is a little different with the former one. The reason why BST was faster in random order insert operation is that BST doesn't care about the height balance. AVL tree should care about height balance, so it should restruct itself when the height balance is collapsed. However, BST doesn't need to do that. That is the reason why BST is faster than AVL tree in random order insert operation.