CSUN | CALIFORNIA STATE UNIVERSITY NORTHRIDGE

Word Checker

Shiva Ramezani

Armineh Karamian

Data structures:

1- Binary search tree

2- AVL tree

3- Hash table

Binary Search Tree

Binary Search Tree

BST is a special type of binary tree in which left child of a node has value less than the parent and right child has value greater than parent.

Time Complexity O(n)

Space Complexity O(n)

Justification

BST algorithms makes it easier to store data and move from one storage space to another whenever it is needed. Hence, we decided to use this algorithm as one of our choice to implement.

AVL Tree

AVL Tree

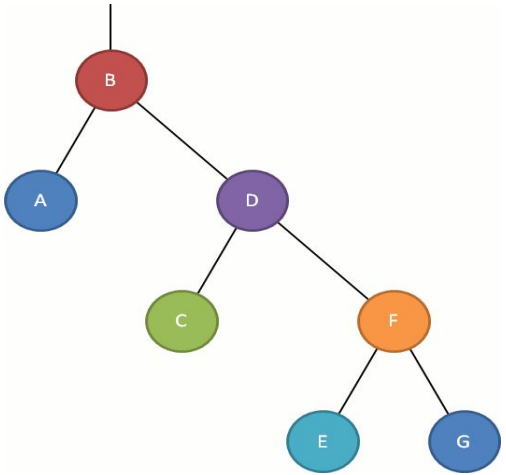
The AVL Tree is a height balanced binary search tree in which each node has a balance factor that is computed by subtracting the height of the right sub-tree from the height of the left sub-tree. If the balance factor of each node is between -1 and 1, the tree is said to be balanced; otherwise, the tree is imbalanced and must be balanced.

Time Complexity O(log n)

Space Complexity O(n)

justification

Our second choice is AVL tree which is a self-balancing binary tree with logarithmic time guarantee, but that will cause each node store some extra information on the storage end. If we don't need the logarithmic time guarantee, BST will work on the average logarithmic time for query and linear time in worst case, but for sure we will save some memory.



Hash Table

Hash Table

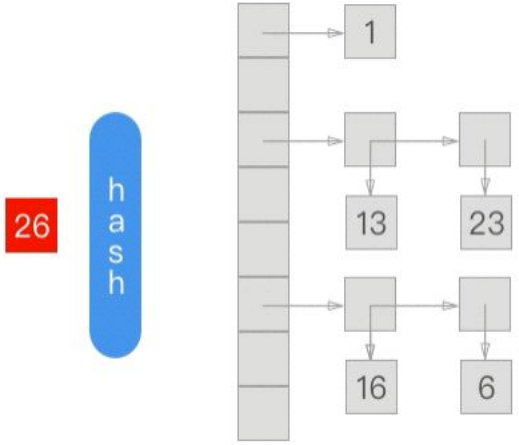
Hash Table is a data structure which stores data in an associative manner. In a hash table, data is stored in an array format, where each data value has its own unique index value. Access of data becomes very fast if we know the index of the desired data.

Time Complexity O(1)

Space Complexity O(n)

Justification

If we want constant time lookup (average runtime), hash table is a good option. Therefore, we used this one.



Performances

File / Algorithm	Binary Search Tree	AVL Tree	Hash Table
1000	0	0	0
10000	0	0	0
100000	0	0	0
1000000	1	3	0
10000000	20	51	5
100000000	688	2008	230

Table 1. Execution time of different data structure in seconds

Analyzing the Result

Based on our result Hash Table has the best run-time efficiency. The AVL tree was the worst.

Hash Table > BST > AVL Tree

Theoretically, AVL tree has a better run-time complexity than BST, but for random access data, hash tables are superior to both trees, also the given data is not dynamic and it is unsorted. Therefore, the runtime result in this algorithm is opposite.