Monday, November 4, 2019

Explanation

Problem 1: Least Recently Used Cache

LRU Cache uses dictionary (map) to store the data. The cost for inserting, deleting, and searching is O(1), which is a pretty efficient data structure.

The capacity is defined in LRU Cache class. If the capacity is full and a new data is pushed into, dictionary follows FIFO rule like queue structure by removing the most left key & value so that new key & value can be added into the dictionary.

Problem 2: File Recursion

File Recursion uses dynamic array to store the file names under its directory. Since it is unknown how many files are included in sub directory, dynamic array is a better structure than static array since it can adjust its size easily. The cost for inserting, deleting, and searching is O(n) for dynamic array.

Recursion technique is used to iterate the searching until all the files with .c suffix are found. Since recursion is a technique in which a problem is solved in-terms of itself, so it can help to reduce unnecessary amount of callings.

Problem 3: Huffman Code

In this problem I used three types of data structure: linked list, queue, and binary tree.

Linked list has the cost of O(1) for insertion and deletion. It is used to link the left and right child leaf to their root.

Queue is the data structure that follows FIFO rule. The advantage of using queue to merge every nodes into root is that we can make sure the lower level leaves are merged and assigned binary code by upper level leaves/roots.

Binary tree has the cost of O(log(n)) for insertion, deletion, and searching. In this problem all the sub-functions in the searching class define the binary search tree structure. Each leaf has a binary code (1 or 0) assigned. Recursion technique is used to assigned binary code to every single leaf/root until traversing to the top root

Problem 4: Active Directory

Active Directory uses dynamic array structure, which has the cost of O(n) for insertion, deletion, and searching. The reason for using dynamic array is that we don't know how many new child or sub child will be added in the future. Therefore, we need to use the structure that can change its size easily.

Recursion technique is used to traverse and search all of the child/sub-child of a parent until the user is found.

Problem 5: Blockchain

Blockchain is a linked list that has the cost of O(1) for insertion, deletion, and searching. Every new node is attached as the tail of previous node. The search is conducted from tail to head until the certain data is found.

Problem 6: Union and Intersection of Two Linked Lists

In this problem the linked list structure is used that has the cost of O(1) for insertion, deletion and O(n) for searching. After two linked lists are created, we need to find the union and intersection group between two linked lists.

The logic for union group is that by first assigning all the unique data from linked list 1 into union group, only the data from linked list 2 that doesn't appear in union group will be assigned.

The logic for intersection group is that only the unique data from linked list 1 that can also be found in linked list 2 will be assigned to intersection group.