**Problem 1: LRU cache**

The program uses an OrderedDict which is a dictionary subclass that remembers the order the keys were first inserted in.

The difference between [dict()](http://quiz.geeksforgeeks.org/python-set-4-dictionary-keywords-python/) and OrderedDict() is that while a regular dict does not track the insertion order, and iterating thought it returns values in an arbitrary order, OrderedDict preserves the order in which the keys are inserted i.e. the order the items are inserted is remembered by OrderedDict.

OrderedDict  has O(1) getitem, setitem, and delitem just like regular dict, hence an OrderedDict proves a useful data type for performing LRU cache operations and thus has a time complexity of O (1) and the space complexity is O(n) where n is the size of the LRU cache implemented

**Problem 2: File Recursion**

The os.walk() function used here follows a tree traversal, so if there are n total directory entries in the subtree, os.walk will take O(n) time with a space complexity is O(n).

**Problem 3: Huffman coding**

Huffman coding -Here again, The tree traversal time will increase linearly based on the input string size needed be encoded/decoded. If there are n nodes in the tree then it takes O(log n) time complexity to decode and the decoded text would have a space complexity of O(length) where length is length of input

**Problem 4 : Active Directory**

The first step was to make sure the second parameter enter in the is\_user\_in\_group(user, group) method is actually a valid group and then I am checking for the group name and if it is the same as the user and then checking the user list of the groups and then the same is repeated for child groups and if a user is found then a True is returned else False. Since it searches the entire depth of the groups, the time & space complexity here is O(m \* n) Space where m is the depth and n the number of users)

**Problem 5: Blockchain**

Here, a linked list is used to store the block chain with the next link pointing to the previous linked list element and has a Space complexity: O(n) - total space occupied by the blockchain Time complexity: O(1) - for appending a block

**Problem 6: Union and Intersection**

The codes for function union & function intersection will loop through each of the items in the list joining and comparing the items between the 2 linked lists of interest and the hence would have a (O(n) or linear complexity space complexity of O(n) where n is the sum of number of elements in the two linked lists

The function remove\_Duplicate loops through each element in the list and creates a dictionary of items that were already visited to create a set of items and hence has a linear time space complexity of O(n) where n is the sum of number of elements in the linked lists.