# Implementation – time and space complexity

## Problem 1 – LRU Cache

The requirement was to have a lookup table ordered according to the frequency of access. The dictionary data structure was favoured due to the hashing mechanism. Even though the newest versions of Python dictionary allows an ordered lookup table, the OrderedDict was chosen since it has been specifically designed for this.

The OrderedDict data structure has to effectively maintain an array to keep the data and a hash table for data lookup which allows O(1) lookups using the keys assuming no hash collisions. However, it comes with the additional requirement of having to keep the array synchronised with the hashtable. Due to this, insert/delete operations have a worstcase time complexity of O(n).

Space complexity of the OrderedDict is O(n).

## Problem 2 – Finding Files

Time complexity of the file search algorithm depends largely on the number of files and folders in the file system staring from the folder given when the function is first called. The search is carried out as a DFS search. It is a linear traversal. The number of search iterations carried out depends on the width and depth of the folders. After every level, only a segment of the entire file space is searched in linear time. Therefore the time complexity is O(n).

According to the algorithm, the file and folder objects are kept in memory until the next level of recursion call is completed. They are released after that. If there are n files in the system, the worst case in terms of memory management is that they are stored n different subfolders which makes the search n deep. This requires to keep n folder objects in memory which gives the worst case space complexity of O(n).

## Problem 3 – Huffman Coding

Huffman Coding is implemted using Node, Priority Queue and Tree data structures. When a character string is received, the frquency of occurance of each character is calculated, stored inside a Node, and sent into the Queue. The Queue is sorted using the Python command, *sorted* which has a time complexity of O(n log n) where n is the number of characters. Theses Nodes become leaf nodes in a tree in which merged nodes are intermediate nodes with only two child nodes. Encoding is done by traversing through the tree until a leaf node is found.

Given this configuration, the overall time complexity of the algorithm is O($n$ log $n$) where n is the number of characters in the given string.

The space complexity is O(n) where n is the size of the entire character set.

## Problem 4 – Active Directory

This implementation is a hierarchical tree structure. Each node (group) has a set of users and links to a set of groups beneath it. Search is effectively a recursive search that depends on the number of users (u) and the number of groups (g). This gives time complexity of O(u*g) or O(n) where n is the total number of items to search. Space complexity is O(1) since it has to only hold the item to be returned.

**Problem 5 - BlockChain**

This has been implemented as a linked list of Blocks. Each Block contains a time stamp and data. It also contains a function to calculate the hash code. Time complexity of append operations is O(1) while the search operation has O(n) worst case time complexity where n is the number of items stored in the linked list. The function to convert to convert it to a python list and calculate its size have time complexity of O(n).

Space complexity directly depends on the number of items stored in the llinked list. Hence it is O(n).

**Problem 6 – Union and Intersection operations**

All items are first stored in two linked lists. For the Union operation they are appended into separate lists (O(n)) and Python set operation is executed. Its worst case lookup time complexity is O(n). This makes the overall time complexity of our Union operation is O(n).

For the Intersection operation, in addition to the list appends (O(n)) and and set operation (O(n)), we carry out a search in a double loop which has a time complexity of $O(n^2)$. This makes the overall time complexity of the Intersection operation $O(n^2)$.

The space complexity of the operations are O(3n) since three lists are used which gives overall time complexity of O(n).