**Section 1: Theory Questions [26 points]**

**1.1 The deque module is part of which python library? 1 point**

Collections

**1.2 What are 2 differences that distinguish a tree from a graph? 2 points**

1. In a tree there is a unique node called root, however there is no unique node called root in graph.

2. A cycle can be formed in a graph but in a tree there cannot be any cycle.

**1.3 Give the definitions of time complexity and space complexity 2 points**

Time complexity quantifies the amount of time required by an algorithm to solve a given problem.

Space complexity is the amount of memory required to solve an algorithm’s given problem.

**1.4 Describe the bubble sort algorithm and its complexity. What is guaranteed at the end of the first pass? 5 points**

A bubble sort algorithm repeatedly steps through a list comparing adjacent elements then swapping them if they are in the incorrect order.

At the end of the first pass, the largest item will be moved to the end of the list (its correct position).

The time complexity would be O(n^2).

The space complexity would be O(1).

**1.5 Explain what LIFO and FIFO are and how each works in practice with a named data structure 8 points**

LIFO – Last in first out.

Stacks data structure.

Elements are stacked on top of each other with the most recent addition at the top of the stack. The last element is the first to be accessed or removed whilst the first would be last. Pop is used to remove and return the top element from the stack. Push is used to add an element to the top of the stack.

FIFO – First in first out.

Queue data structure.

Like a queue the first element added will be the first to be accessed or removed. The last element would be last. Enqueue is used to add an element to rear of queue and deque removes an element from front of queue.

**1.6 What is a Balanced Binary Tree and what would be the best root? Walkthrough how you search using this structure. 8 points**

A balanced binary tree is a binary tree data structure where the heights of the left and right subtrees do not differ by more than one. This helps with efficiency of searching, inserting and deleting elements, and keeping the tree balanced.

The best root for this type of binary tree would be the median(middle) element from the sorted list.

To search this structure, you would start at the root of the tree which is ideally the median value. You would then compare the element that you are searching for with the one at the current node. If they are the same, you can return the value. If not, you can move to a subtree. The node on the left should be lower than the current node, and the one on the right higher. This process would continue until the value is found. The element may not be present in the tree and therefore not returned.