COMP20007 Assignment 2: Written Problems

Problem 3

a) Melbourne City Library

In this case stability and speed are the core requirements, because of this I would recommend Merge sort, as it has a best and worst case time complexity of $\Theta(n \log n)$ and is stable to ensure duplicate books maintain their order.

b) Phone Manufacturer

For this situation I would suggest Selection Sort, as although the algorithm has a time complexity of $\Theta(n^2)$ it only takes $O(n)$ swaps to produce a sorted array. This is excellent for the situation as it will significantly reduce power consumption over other algorithms and the $\Theta(n^2)$ time complexity is not a large issue for the small sets of items; selection sort is also easily generalised to any input that has a concept of a sorted order.

c) Astronomy

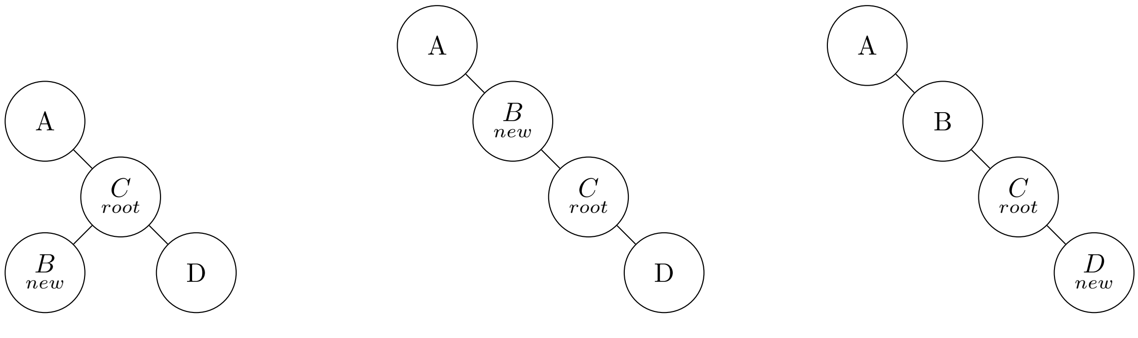
Considering the space and speed requirements of this problem I am recommending the use of Counting Sort, although Counting sort is often implemented such that the resulting sorted array is copied out into a newly allocated array of equal size to the original data, it is possible to write the values over the values in the original array. Such an implementation only requires an array large enough to store k integers, where k is the range of values the data can take, in this case 11 for the integers 0-10, this meets the requirement for not taking additional space greater than the size of the original array as $11 < 50$ and also meets the requirement for speed, as counting sort executes in $O(n + k)$ time.

Problem 4

To convert the binary tree into a right stick we simply need to rotate right at a node whenever a node is added as it's left child. The algorithm for such an operation is as follows:

```
function Stickify(new, root):
    if $new.value < root.value$ then
        RotateRight(root)
```

Figure 1 shows the 2 cases that can be passed into Stickify. In 1a the guard on the **if** statement returns **true** so a rotation is performed to achieve the stick in 1b, however in 1c, the guard evaluates to **false** and Stickify returns having performed no rotation.



(a) The stick when a new node is inserted as a left child.

(b) The reformed stick after Stickify is used on the tree from fig 1a.

(c) A case where *new.value* is greater than *root.value*.

Figure 1: Examples of the 2 cases that Stickify has to deal with.