Assignment 14

Alex Clemmer
Student number: u0458675

1

1.1

The first implementation of `explore()` calls `postvisit()` before the children nodes are visited. This has the obvious effect of breaking the exploration of the dfs: note that in the book `explore()` is called recursively on the child nodes before `postvisit()` is called upon the final traversal of that particular node.

2

2.1

No it does not. It searches depth-first via the rightmost node. The book clearly goes from the leftmost node to the rightmost.

2.2

In the second implementation of `explore()`, we only call `postvisit()` in the event that we've pushed `-v` to the stack for some `v` we just visited. Before we actually pop that element, though, we will have pushed all that node's children to the stack, and all *that* node's children, and so on, until the first leaf. Only then can we recurse back and call that method, as only then will can we encounter a `-v` on the stack.

3

The worst case is $\Theta(|V|^2)$. If all the nodes are connected, then we will add every node for every node in the list. There is no alternative worst case, so this gives us not just $O(|V|^2)$, but also $\Theta(|V|^2)$.

Just like any $O(n)$ vs. $O(n^2)$ relationship, the $O(n^2)$ could represent an improvement over the linear algorithm if the linear algorithms constants are sufficiently high. In our case, for example, if the stack frames caused a prohibitive overhead, it may be some time before the $O(n^2)$ algorithm fell behind.

4

Asymptotically, no. In a fully-connected graph, if we don't add elements that are visited, then each iteration will give us only one less than the previous total. This is a well-known relation: $\sum_j (n-j) = \frac{n(n-1)}{4}$, which is very clearly $O(n^2)$