Question 1. Any algorithm that solves $\mathcal{P}$ can be used to sort L by first using it to place the items of L in T, and then traversing T in linear time and printing an inorder listing of the items in it (the inorder listing produced is a sorted version of L). In other words if $\mathcal{P}$ could be solved faster than the claimed lower bound, then we could sort faster than the known $\Omega(n \log n)$ lower bound for sorting, a contradiction.

Question 2. Without loss of generality assume $n_A < n_B$ (otherwise simply interchange the roles of A and B in what follows).

1. Initialize the answer set S to be empty.
2. Sort A in $O(n_A \log n_A)$ time.
3. For every element x of B, binary search for x in the sorted version of A: If that binary search finds x then include x in S. This step takes $O(n_B \log n_A)$ time, because there are n_B binary searches each of which takes $O(\log n_A)$ time.
4. S now contains the elements that are in both A and B.

The total time is $O(n_A \log n_A + n_B \log n_A)$, which is $O(n_B \log n_A)$ because $n_A \leq n_B$.

Question 3. For every vertex v do the following.

1. Do a breadth-first search starting at v and stop that search as soon as you encounter a non-tree edge (this is why this search from v takes $O(n)$ time). Suppose the non-tree edge encountered is between vertices x and y, and let T be the breadth-first tree created before edge (x, y) was encountered.
2. Let $f(v)$ be the length of the cycle consisting of edge (x, y) together with the x-to-y path along the edges of T.

The length of a shortest cycle is simply $\min_v f(v)$. There can be no shorter cycle because if there were, then the breadth-first search started at a vertex w on that cycle would have given an $f(w)$ equal to the length of that cycle. Note, however, that if G contained odd cycles, then we would no longer be justified in stopping the breadth-first as soon as a non-tree edge is encountered (because a better odd-length cycle may lie ahead and we would miss it).

Question 4. For part 1 see the figure on the next page. Forward edges are (d, g), (b, e). Backward edges are (b, a), (f, d), (j, h), (k, c). Cross edges are (g, f), (h, d), (j, d), (l, j), (k, g), (m, k). The strongly connected components are $\{a, b\}$, $\{c, k, i, m\}$, $\{h, j, l\}$, $\{d, f, e, g\}$.

Question 5. See the figure on the next page.

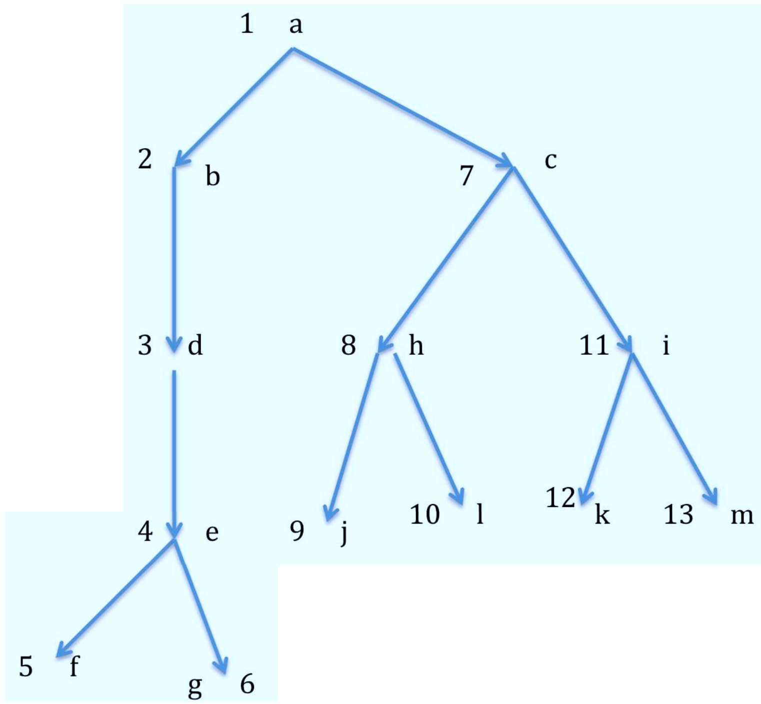

Figure for Question 4.1

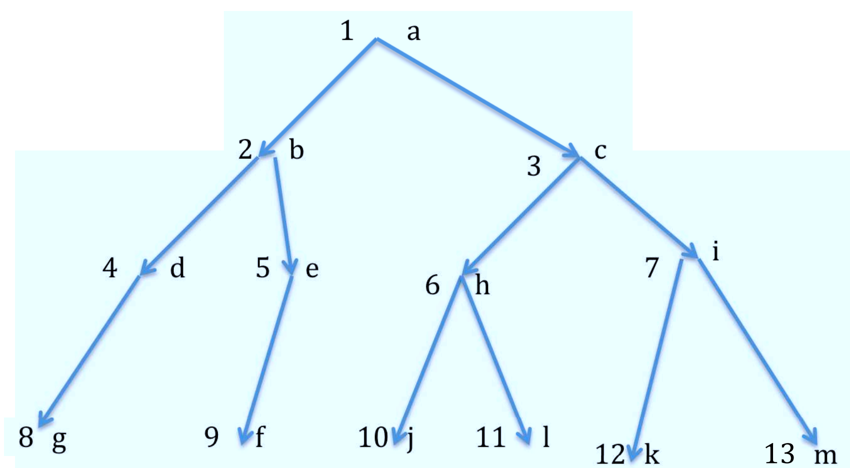

Figure for Question 5

Figure 1: The figures for questions 4.1 and 5