Fundamental Algorithms 10

Exercise 1: Modified Depth/Breadth-First Traversal

Consider the following, modified traversal algorithm for graphs and trees:

ModTraversal(V: Node)
{
    // assume Mark[V.key] = 1 at entry
    // init (local) list for "active" nodes
    Queue active = { };
    // visit all (non-visited) nodes connected to V
    forall (V, W) in V.edges do {
        if Mark[W.key] = 0 then {
            // visit node W and mark as visited:
            Visit(W);
            Mark[W.key] := 1;  // append node W to "active" nodes
            append(active, W);
        }
    }
    // perform traversal from all "active" nodes connected to V
    forall W in active do {
        ModTraversal(W);
    }
}

Exercise 1a:

Consider the graph given in Figure 1: in what order are the nodes "visited" by the modified traversal? (Number the nodes in the graph accordingly.) The traversal shall be called by

Mark[S.key] := 1;
ModTraversal(S);

(S being the start node for the traversal).
Solution:

Due to the recursive call of the function ModTraversal, the traversal is similar to a depth-first traversal. However, the approach to first mark the nodes adjacent to the current node and append them to a list of active nodes is similar to breadth-first traversal. Hence, the traversal is a mixture between DFT and BFT: first, all nodes adjacent to the current node are visited, but the traversal then proceeds in depth-first manner. See Figure 1 for the resulting order.

Exercise 1b:

In the same graph, mark the edges that are part of the spanning tree computed by ModTraversal.

Solution:

See Figure 1.

Exercise 1c:

Now assume that the second forall-loop is changed into a parallel loop:

```cpp
// perform traversal from all (non-visited) nodes connected to V
forall W in active do in parallel {
    ModTraversal(W);
};
```

Figure 1: Graph for Exercise 1a) and 1b). It it not specified, in which order edges outgoing from a node V are stored in the list V.edges – you may assume any order you like.
Discuss whether there can be concurrent read or write access to the elements of the array Mark. Discriminate between the two cases that the traversed graph is a tree and that it is not a tree.

Solution:

For an arbitrary graph (not a tree): Here, concurrent access are possible. Consider nodes 2 and 3 in Fig. 1: they will trigger concurrent accesses to the nodes 13 and 15, for example.

For a tree: In that case, no concurrent accesses can happen – subtrees are traversed in parallel, but as the subtrees are not allowed to share nodes (this would violate the tree property!), parallel accesses are exclusive.