Exercise 1 (Modified Graph Traversal)

Consider the modified traversal algorithm for graphs and trees \texttt{ModTrav}.

\begin{algorithm}
\textbf{Input:} \(V,:\) Node
\(act \leftarrow \emptyset;\) // Local queue of active nodes
\textbf{for} \((V,W) \in V\text{.}\, \text{edges}\) \textbf{do}
  \hspace{1em} \textbf{if} \ \text{mark}[W\text{.\,key}] = 0 \textbf{then}
    \hspace{2em} \text{visit}(W);
    \hspace{2em} \text{mark}[W\text{.\,key}] \leftarrow 1;
    \hspace{2em} act \leftarrow act \circ W;
  \textbf{end}
\textbf{end}
\textbf{for} \ W \in \text{act} \textbf{do} \text{ModTrav}(W);
\end{algorithm}

1. Given the graph above, in which order is the function \texttt{visit} called on the nodes by this algorithm? Number the nodes accordingly. It is given that, initially, the start node \(S\) is marked and the rest of the nodes are unmarked. It is not specified in which order edges outgoing from a node \(V\) are stored in the list \(V\text{.}\, \text{edges}\) – you may assume any order you like.

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

3. Now assume that the second for-loop is changed into a parallel loop. Discuss whether there can be concurrent read or write access to the elements of the array \texttt{mark}. Think about what happens if the graph is a tree.
Solution:

1. Due to the recursive call of the function ModTrav, 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. Together, 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.

2. Concurrent access is possible. Consider nodes 2 and 3 in the example graph. The recursion starting from these nodes concurrently accesses the nodes 13 and 15, for example. In case of tree, two nodes having the same parent node may cause concurrent access.