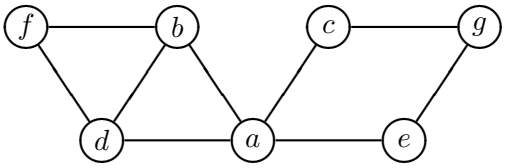
Exercises for Algorithms and Data Structures Lecture 6

Question 1. Consider the following undirected graph G:



a. (Levitin, exercise 3.5.1.b)
Make a DFS traversal through G, starting in node a. Consider nodes with multiple neighbours in alphabethic order.

Construct the corresponding DFS tree. Indicate in which order the nodes are being visited for the first time (added to the stack) and in which order they are completely handled (removed from the stack).

b. (Levitin, exercise 3.5.4)
Make a BFS traversal through G, starting in node a. Consider nodes with multiple neighbours in alphabethic order.

Construct the corresponding DFS tree. Indicate in which order the nodes are being visited for the first time.

Question 2.

a. Adjust the pseudo-code for DFS traversals in an undirected graph (see slides or Levitin paragraaf 3.5) in such a way, that the algorithm checks whether a graph is acyclic. If this is the case, the algorithm has to return **true**. If this is not the case (i.e., the graph has at least one cycle), the algorithm has to return **false**, and print the nodes of the cycle, in the order of the cycle.

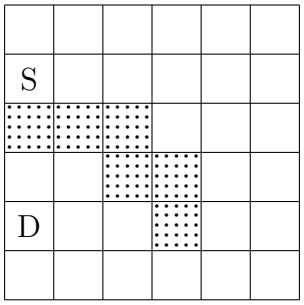
b. (based on Levitin, exercise 3.5.6.a)
Adjust the pseudo-code for BFS traversals in an undirected graph (see slides or Levitin paragraaf 3.5) in such a way, that the algorithm checks whether a graph is acyclic. If this is the case, the algorithm has to return **true**. If this is not the case (i.e., the graph has at least one cycle), the algorithm has to return **false**.

c. (Levitin, exercise 3.5.6.b)
Note that we can use both DFS as well as BFS to detect cycles in an undirected graph. Is one always preferable over the other?

If so, which of the two is as fast as the other, and why is that the case? If not, please give an example of a graph in which BFS is preferable, and an example of a graph in which DFS is preferable.

Question 3. Adjust the psuedo-code for the DFS traversal of an undirected graph in such a way that the algorithm checks whether the graph is connected. If this is the case, the algorithm will return **true**, and the algorithm will return **false** otherwise. Additionally, the connected components of the graph need to be numbered in such a way, that each nodes that are connected receive the same number.

Question 4. Consider the following 6x6 board:



You can move in one step to each adjacent square (horizontal, vertical and diagonal). The shaded squares are not accessible.

a. Execute a Breadth First Search on this board, starting at square S. Note for each (non-shaded) square what the resulting distance is from S.

b. Determinem based on the answer of **a.**, all shortest paths from S to D. How many of these shortest paths are there?