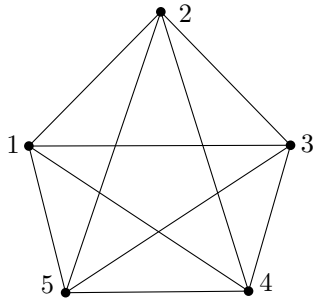
University of Freiburg
Dept. of Computer Science
Prof. Dr. F. Kuhn

Algorithms and Data Structures
Summer Term 2021
Sample Solution Exercise Sheet 8

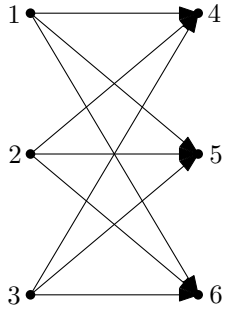
Exercise 1: Graph Representations

Give the following graphs as *adjacency matrix* and *adjacency list*.

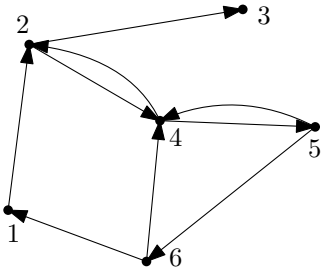
(a)



(b)



(c)



Sample Solution

(a)
$$\begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$1 : 2 \to 3 \to 4 \to 5 \to \bot$
$2 : 1 \to 3 \to 4 \to 5 \to \bot$
$3 : 1 \to 2 \to 4 \to 5 \to \bot$
$4 : 1 \to 2 \to 3 \to 5 \to \bot$
$5 : 1 \to 2 \to 3 \to 4 \to \bot$

(b)
$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$1 : 4 \to 5 \to 6 \to \bot$
$2 : 4 \to 5 \to 6 \to \bot$
$3 : 4 \to 5 \to 6 \to \bot$
$4 : \bot$
$5 : \bot$
$6 : \bot$

(c)
$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

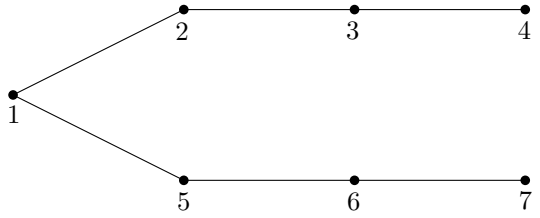
$1 : 2 \to \bot$
$2 : 3 \to 4 \to \bot$
$3 : \bot$
$4 : 2 \to 5 \to \bot$
$5 : 4 \to 6 \to \bot$
$6 : 1 \to 4 \to \bot$

Exercise 2: Breadth First (BFS) and Depth First Search (DFS)

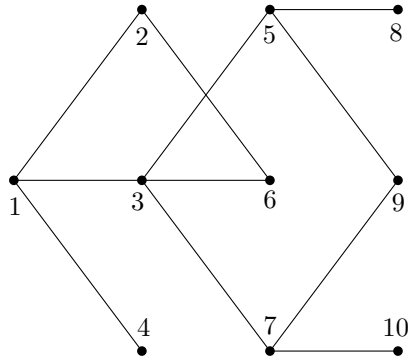
For the following graphs, give the order in which nodes are visited (marked) when running BFS and DFS. Moreover, mark the resulting spanning trees in the respective graph. Start with the node with

identifier 1. Whenever there is a choice, mark the node with smallest identifier first.

(a)

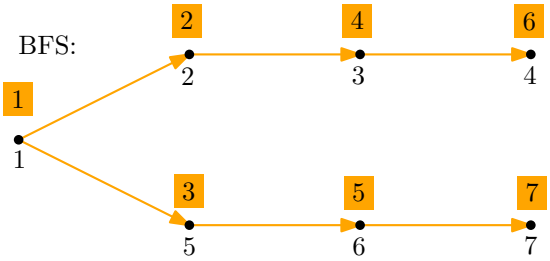


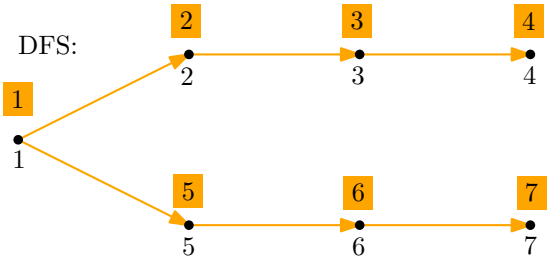
(b)



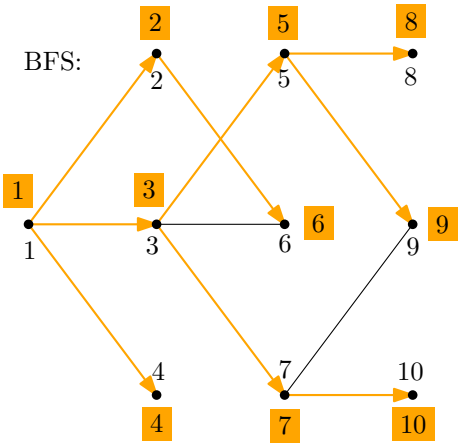
Sample Solution

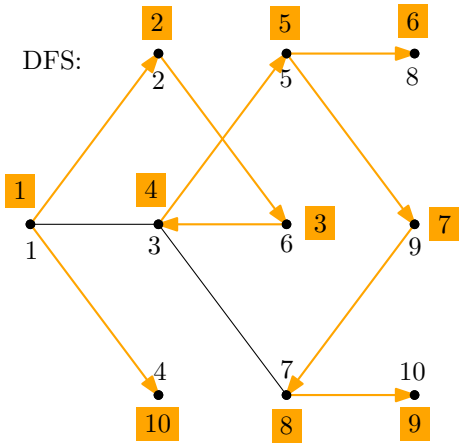
(a)





(b)





Exercise 3: Check for Cycles

(a) Let $G = (V, E)$ be an undirected graph represented by an *adjacency list*. Describe an algorithm that tests in $\mathcal{O}(|V|)$ steps whether G has a cycle.

(b) Argue why your algorithm is correct and has the desired running time.

Sample Solution

We do a BFS (or DFS) on the graph. We return "cycle found" as soon as a node is found that is already marked. BFS on a graph given as adjacency list takes $\mathcal{O}(m+n)$. However we argue that the algorithm already stops after $O(n)$ steps.

If there is no cycle in the graph, we have $m = \mathcal{O}(n)$ and all nodes are visited after $\mathcal{O}(n)$ steps. If there is a cycle, one must be found after looking at the first n edges, since n edges must necessarily have a cycle in a graph with n nodes. Thus a cycle is detected after $\mathcal{O}(n)$ steps.