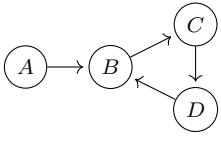
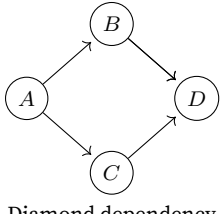
Dependency hell

In modern software development, it is common for an application to depend on many *libraries*, which in turn may depend on other libraries. A library can be seen as a collection of code that is designed for reuse. The libraries used to build an application are called the application's *dependencies*. The dependencies between libraries are given as a directed graph $G = (V, E)$ where a node represents a library, and where there is an edge from u to v if u uses v as a library.

The examples below are situations often referred to as *dependency hell*.



Circular dependency



Diamond dependency

Cyclic dependencies are generally unwanted because an update of one of the libraries in a cycle could affect all the other libraries in the cycle. *Diamond dependency* is also generally unwanted because an update can lead to being dependent on different versions of the same library simultaneously.

You must develop algorithms that can help identify such unwanted situations.

(a) More precisely, we say that the graph contains a *cyclic dependency* if there is a path from a node v to the same node v which consists of at least two nodes.

Write a procedure that, given a graph $G = (V, E)$, returns **`true`** if the graph contains *cyclic dependencies*, and **`false`** otherwise. If your procedure uses algorithms known from the curriculum, you must briefly explain how they work.

(b) In this sub-problem, you may assume that the graph does not contain cyclic dependencies.

More precisely, we say that the graph contains a *diamond dependency* if there are two distinct paths from u to v. Remember that two paths are distinct if they do not share any nodes or edges other than u and v.

Write a procedure that, given a directed acyclic graph $G = (V, E)$, returns **`true`** if the graph contains *diamond dependencies*, and **`false`** otherwise.