

Chapter 16

Directed graphs

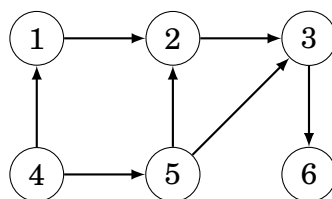
In this chapter, we focus on two classes of directed graphs:

- **Acyclic graphs:** There are no cycles in the graph, so there is no path from any node to itself¹.
- **Successor graphs:** The outdegree of each node is 1, so each node has a unique successor.

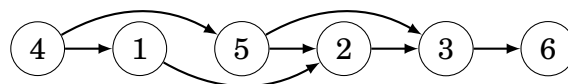
It turns out that in both cases, we can design efficient algorithms that are based on the special properties of the graphs.

16.1 Topological sorting

A **topological sort** is a ordering of the nodes of a directed graph such that if there is a path from node a to node b , then node a appears before node b in the ordering. For example, for the graph



a possible topological sort is [4, 1, 5, 2, 3, 6]:



An acyclic graph always has a topological sort. However, if the graph contains a cycle, it is not possible to form a topological sort, because no node in the cycle can appear before the other nodes in the cycle. It turns out that depth-first search can be used to both check if a directed graph contains a cycle and, if it does not contain a cycle, to construct a topological sort.

¹Directed acyclic graphs are sometimes called DAGs.

Algorithm

The idea is to go through the nodes of the graph and always begin a depth-first search at the current node if it has not been processed yet. During the searches, the nodes have three possible states:

- state 0: the node has not been processed (white)
- state 1: the node is under processing (light gray)
- state 2: the node has been processed (dark gray)

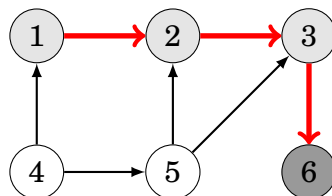
Initially, the state of each node is 0. When a search reaches a node for the first time, its state becomes 1. Finally, after all successors of the node have been processed, its state becomes 2.

If the graph contains a cycle, we will find out this during the search, because sooner or later we will arrive at a node whose state is 1. In this case, it is not possible to construct a topological sort.

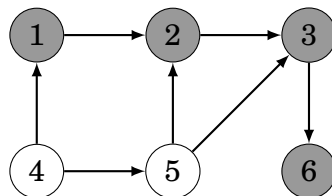
If the graph does not contain a cycle, we can construct a topological sort by adding each node to a list when the state of the node becomes 2. This list in reverse order is a topological sort.

Example 1

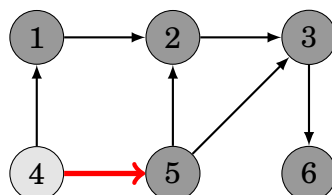
In the example graph, the search first proceeds from node 1 to node 6:



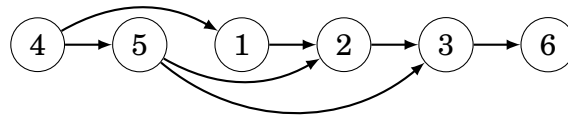
Now node 6 has been processed, so it is added to the list. After this, also nodes 3, 2 and 1 are added to the list:



At this point, the list is [6,3,2,1]. The next search begins at node 4:



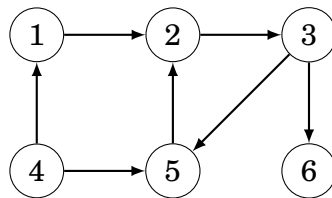
Thus, the final list is [6, 3, 2, 1, 5, 4]. We have processed all nodes, so a topological sort has been found. The topological sort is the reverse list [4, 5, 1, 2, 3, 6]:



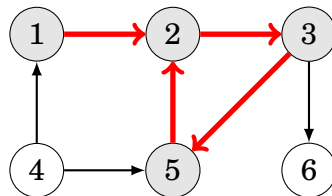
Note that a topological sort is not unique, but there can be several topological sorts for a graph.

Example 2

Let us now consider a graph for which we cannot construct a topological sort, because there is a cycle in the graph:



The search proceeds as follows:



The search reaches node 2 whose state is 1, which means the graph contains a cycle. In this example, the cycle is $2 \rightarrow 3 \rightarrow 5 \rightarrow 2$.

16.2 Dynamic programming

If a directed graph is acyclic, dynamic programming can be applied to it. For example, we can efficiently solve the following problems concerning paths from a starting node to an ending node:

- how many different paths are there?
- what is the shortest/longest path?
- what is the minimum/maximum number of edges in a path?
- which nodes certainly appear in any path?