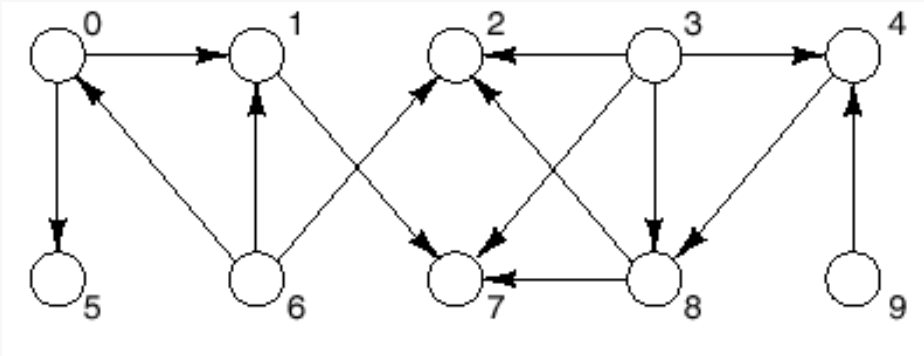
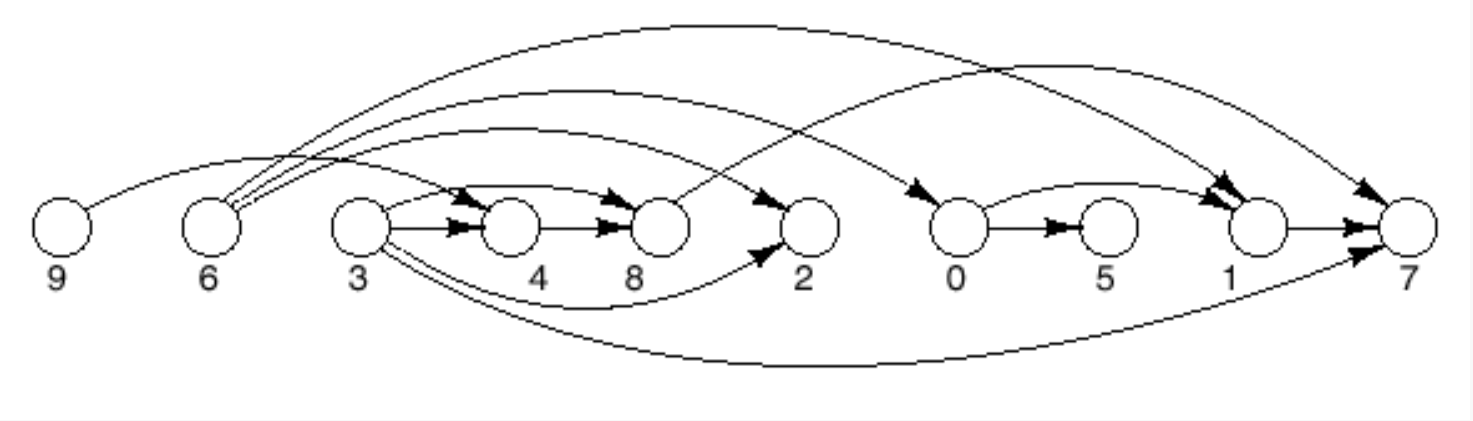
Topological Ordering

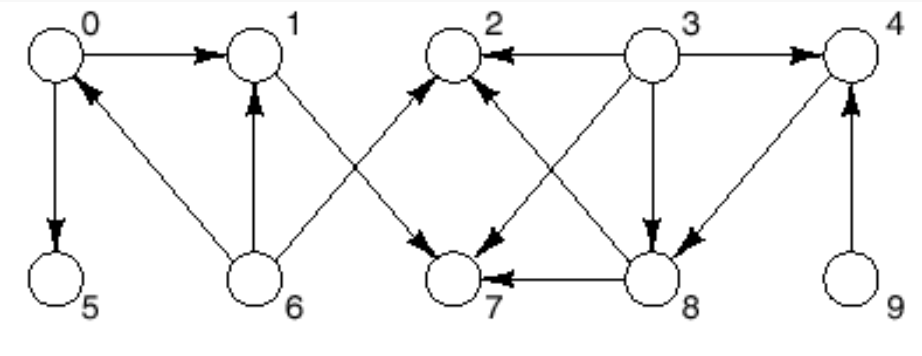
Suppose that G is a directed graph which contains no directed cycles:



Then, a topological ordering of the vertices in G is a sequential listing of the vertices such that for any pair of vertices, v and w in G, if (v,w) is an edge in G then v precedes w in the sequential listing.



Computing a Topological Ordering



A topological ordering of the vertices of G may be obtained by performing a generalized depth-first traversal.

We build a list L of vertices of G.

Begin with vertex 0.

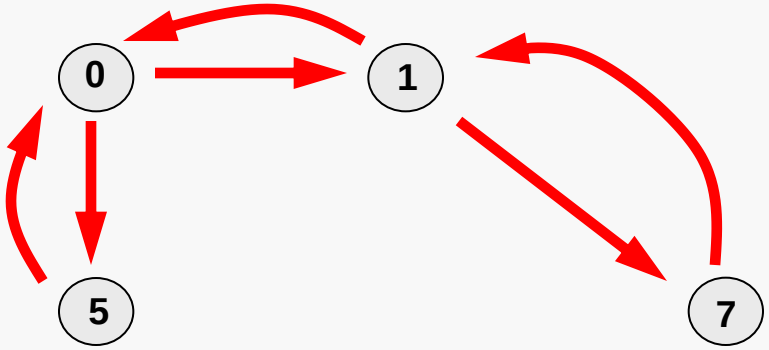
Do a depth-first traversal, marking each visited vertex and adding a vertex to L only if it has no unmarked successors.

When the traversal adds its starting vertex to L, pick the first unmarked vertex as a new starting point and perform another depth-first traversal.

Stop when all vertices have been marked.

Depth-First Traversal Trace

Initially we probe from 0 to 1 to 7, which has no successors.



L: 7

Next the recursion backs out to 1, which has no unmarked successors.

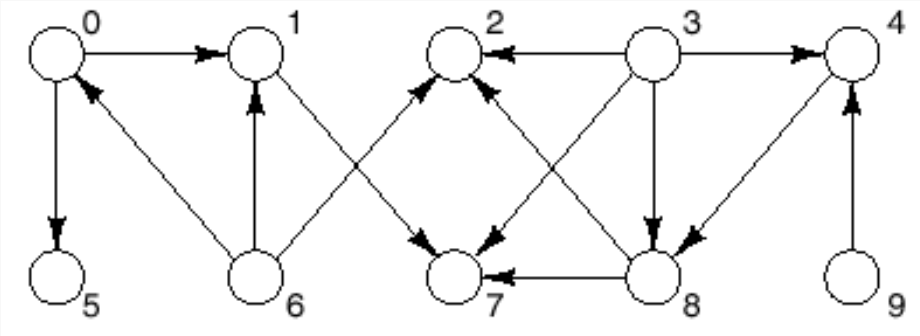
L: 1 7

Next the recursion backs out to 0, and probes to 5, which has no successors.

L: 5 1 7

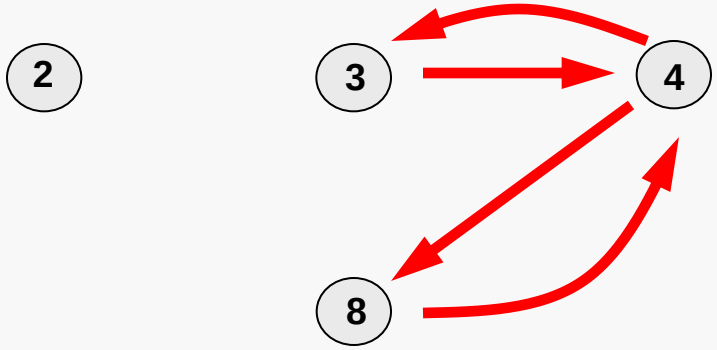
Next the recursion backs out to 0 again, which now has no unmarked successors.

L: 0 5 1 7



Depth-First Traversal Trace

Now we pick the first unmarked vertex, 2, and continue the process. 2 has no successors.



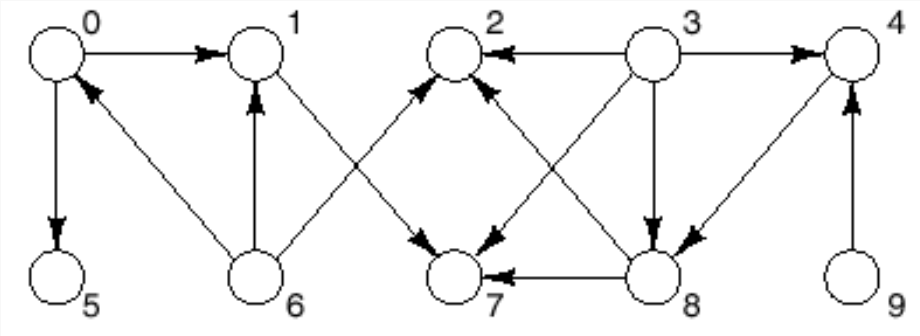
L: 2 0 5 1 7

Next start with vertex 3, and probe to 4 and then to 8, which has no unmarked successors.

L: 8 2 0 5 1 7

The recursion backs out, adding vertices 4 and 3.

L: 3 4 8 2 0 5 1 7



Depth-First Traversal Trace

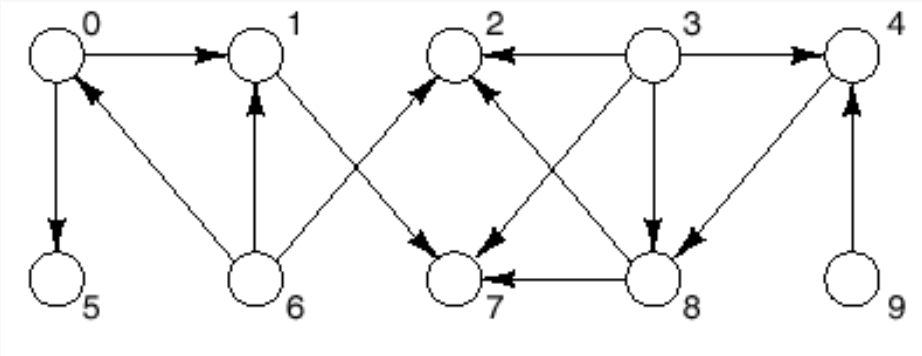
Next we pick the unmarked vertex, 6, which has no unmarked successors.

L: 6 3 4 8 2 0 5 1 7

Next we pick the unmarked vertex, 9, which has no unmarked successors.

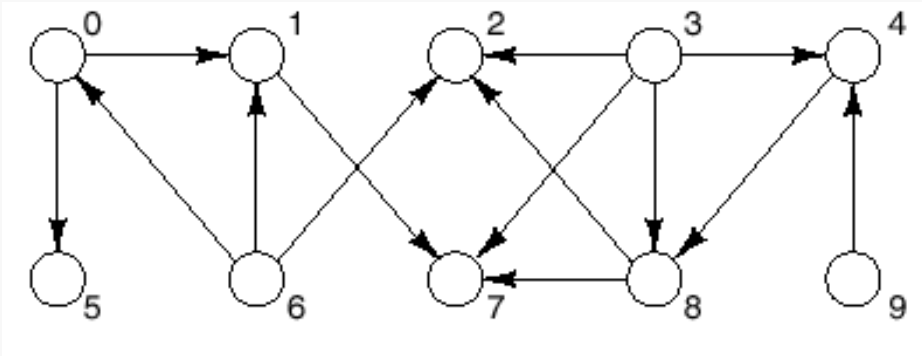
L: 9 6 3 4 8 2 0 5 1 7

At this point, all the vertices have been marked and the algorithm terminates.

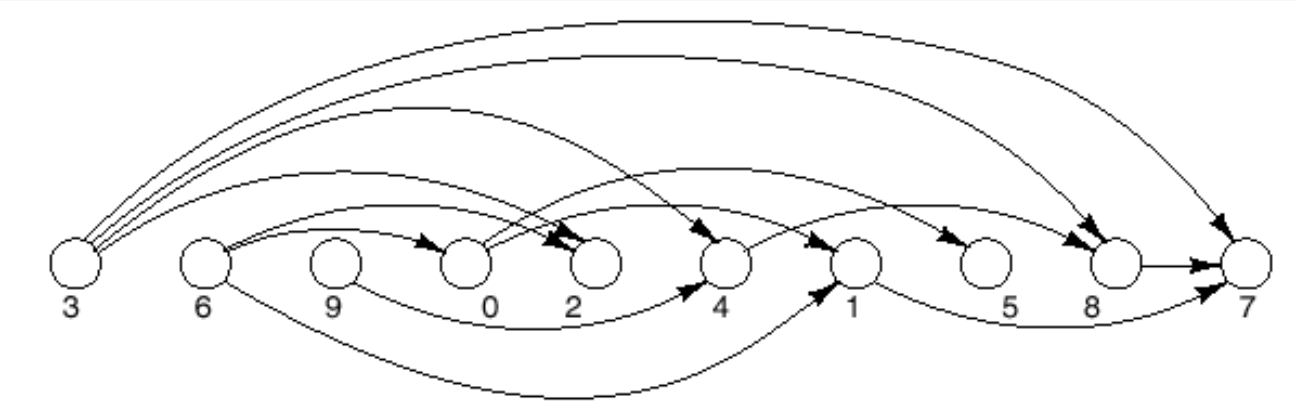


Multiple Topological Orderings

There is usually more than one topological ordering for a graph. Choosing a different rule for picking the starting vertices may yield a different ordering.



Also, a generalized breadth-first traversal can be used instead. For the graph above, a breadth-first traversal yields the ordering:



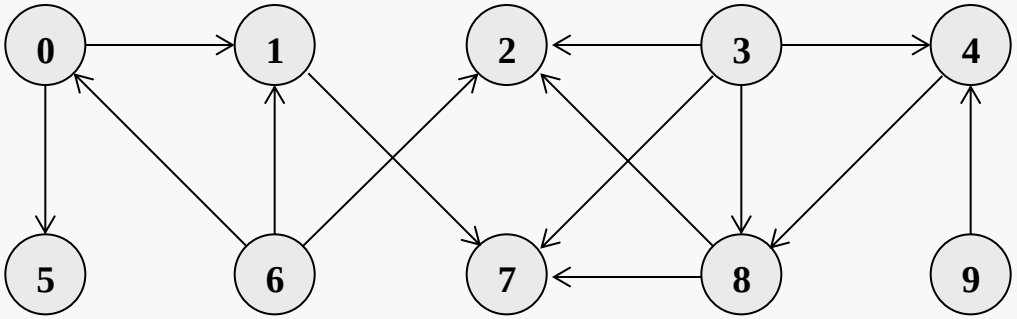
Topological Sort Trace

Algorithm

Find any vertex with no incoming edges adding the vertex to the list.

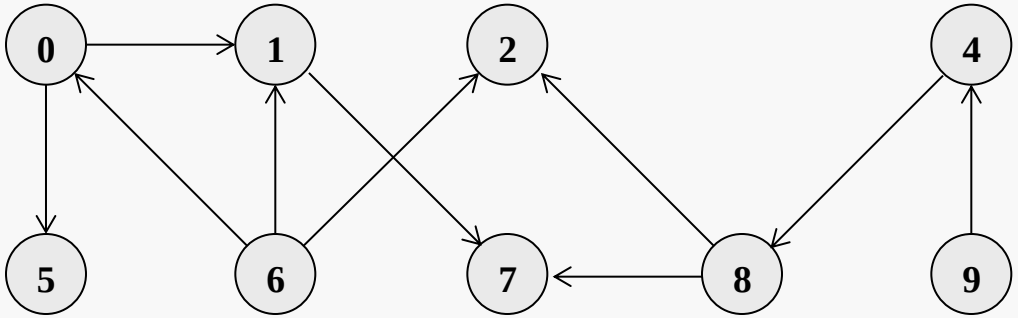
Remove the vertex and its edges from the graph.

Apply the same strategy to the rest of the graph.

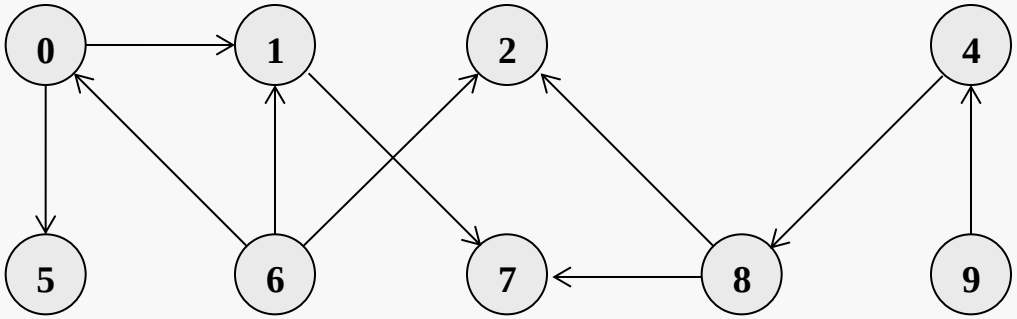


Vertex 3 is the first vertex with no incoming edges.

L: 3

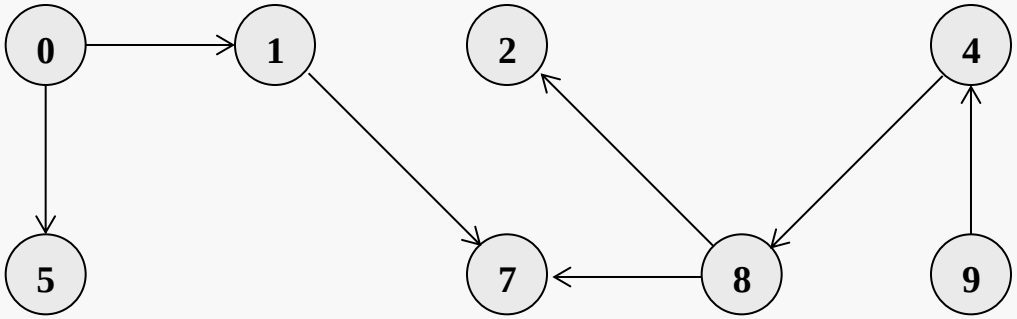


Topological Sort Trace



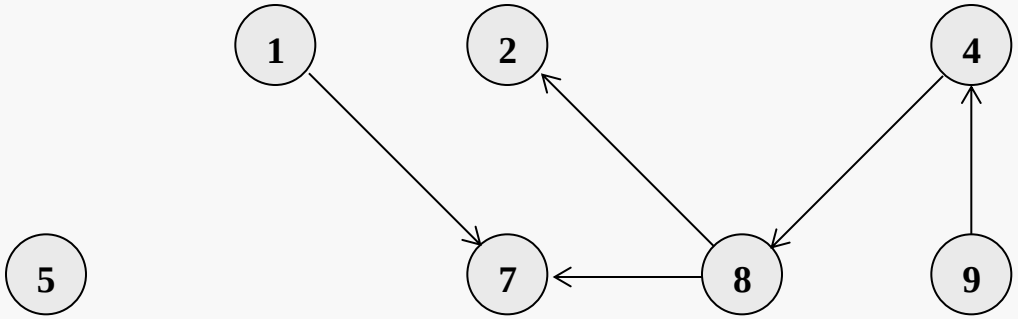
Vertex 6 is the next vertex with no incoming edges.

L: 3 6

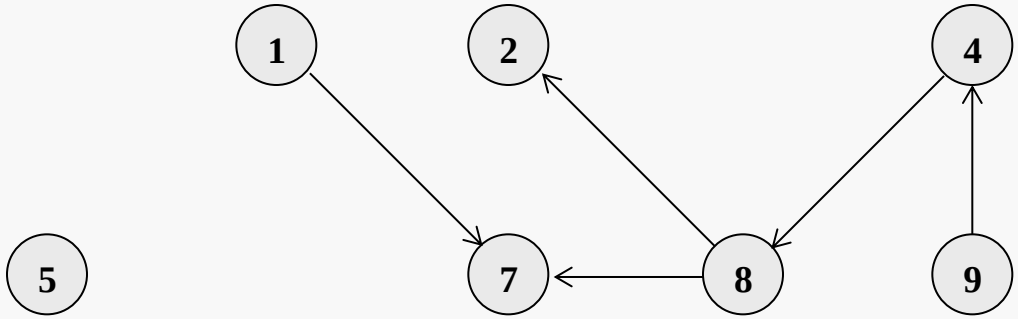


Vertex 0 is the next vertex with no incoming edges.

L: 3 6 0

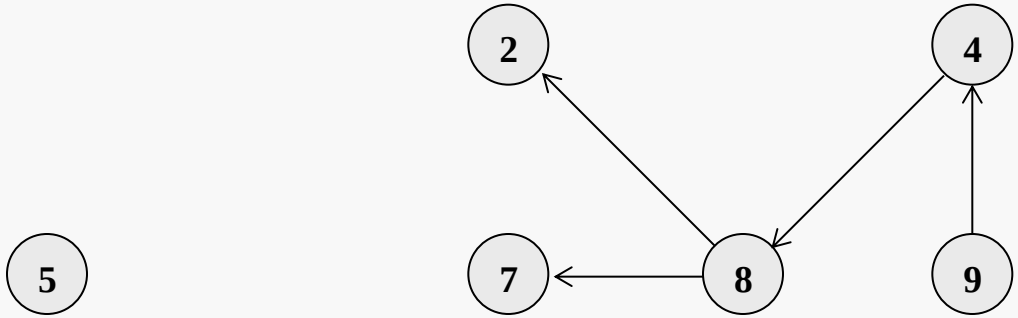


Topological Sort Trace



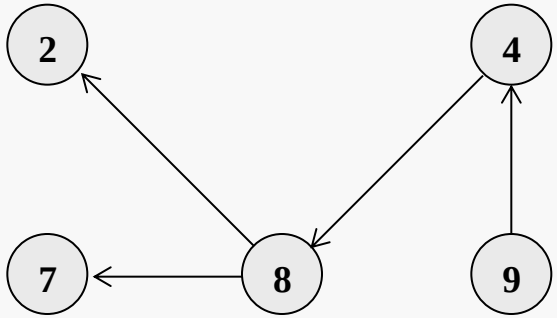
Vertex 1 is the next vertex with no incoming edges.

L: 3 6 0 1

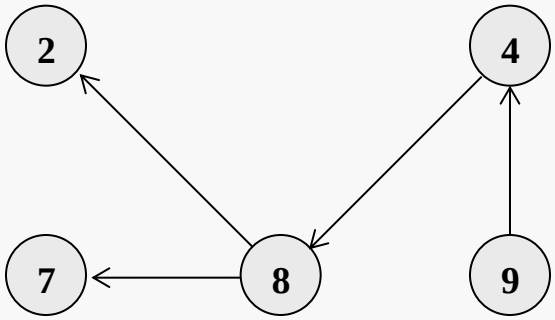


Vertex 5 is the next vertex with no incoming edges.

L: 3 6 0 5

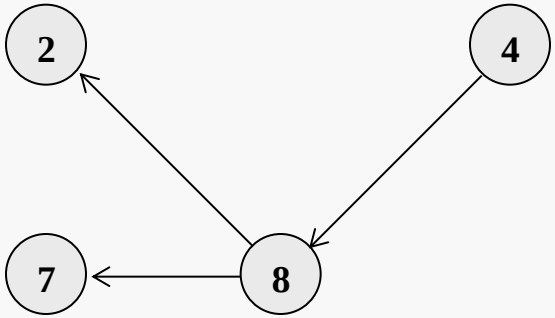


Topological Sort Trace



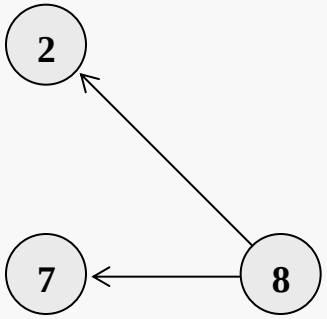
Vertex 9 is the next vertex with no incoming edges.

L: 3 6 0 5 9

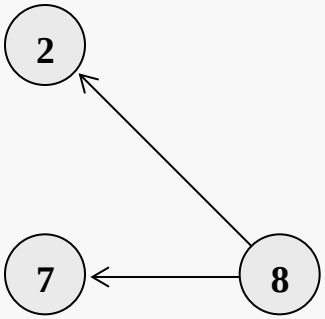


Vertex 4 is the next vertex with no incoming edges.

L: 3 6 0 5 9 4



Topological Sort Trace



Vertex 8 is the next vertex with no incoming edges.

L: 3 6 0 5 9 4 8

Vertex 2 & 7 are the next verticies with no incoming edges.

L: 3 6 9 0 1 4 5 8 2 7

Applications of Topological Orderings

Applications of topological orderings are relatively common…

- prerequisite relationships among courses
- glossary of technical terms whose definitions involve dependencies
- organization of topics in a book or a course