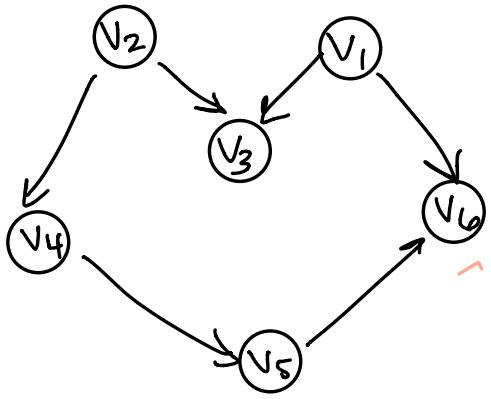
Lecture 36

TOPOLOGICAL SORTING

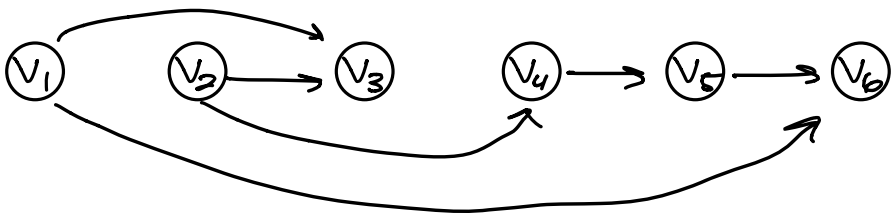
A topological order of a directed graph $G = (V, E)$ is an ordering of its vertices as $v_1, v_2, \ldots, v_n$ so that for every edge (v_i, v_j) we have $i < j$

Example

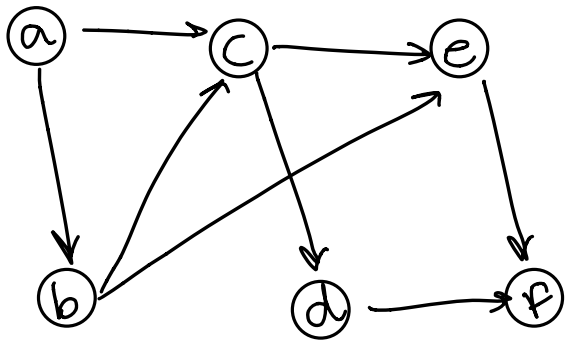
A)



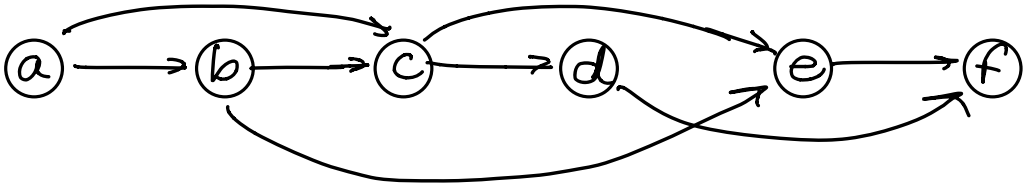
Topological Order



B)



Topological Orders



* In a task scheduling diagraph, a topological sorting of a task sequence satisfies the precedence constraints

* Scheduling problem:

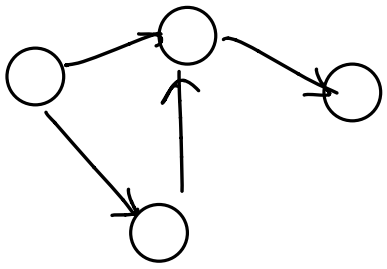
Edge (a,b) means task a must be completed before task b can be started

* course prerequisite graph
course v_i must be taken before course v_j
* Pipeline of computing jobs: output of job v_i needed to determine input of job v_j

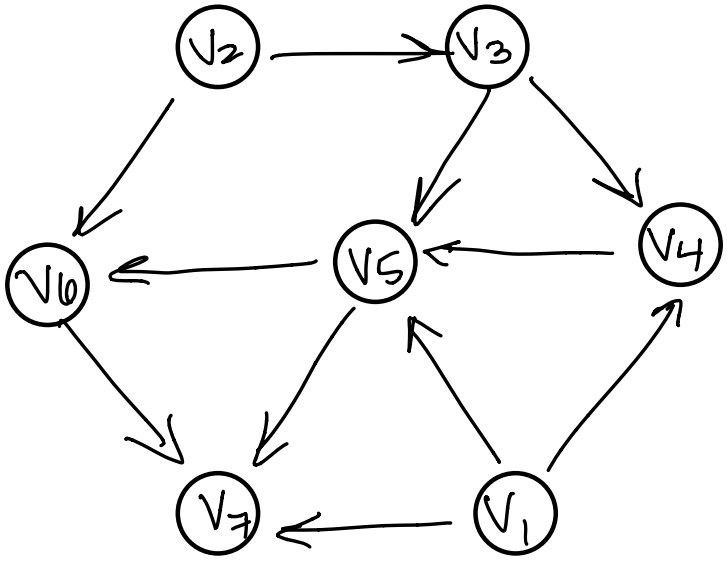
If a directed graph has a directed cycle, you can't have a topological order

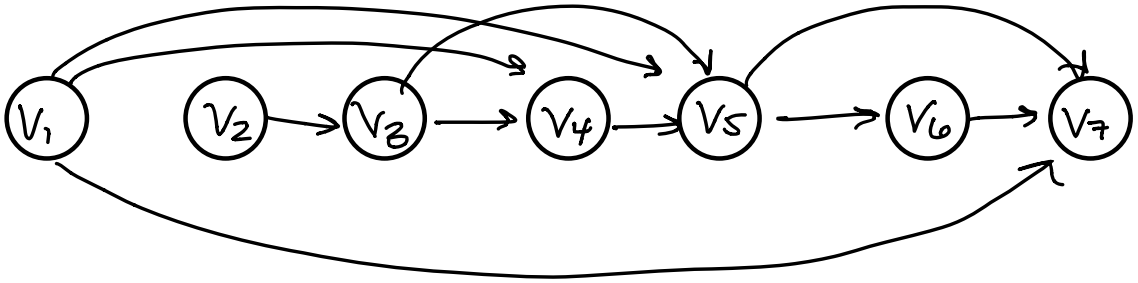
A directed acyclic graph (DAG) is a digraph w/ no directed cycles

Thm. A diagraph has a topological order IFF it is a DAG



= a DAG



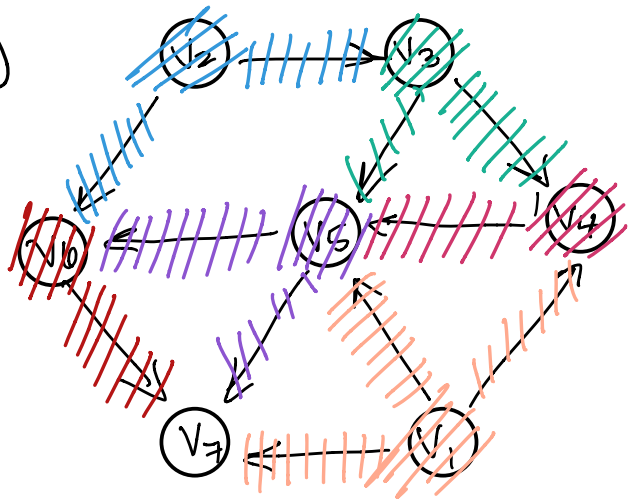


Computing a Top. Ordering from a DAG

- Find a vertex w/ no incoming edges and order it first
- delete it from G
- Recursively compute a topological ordering of $G - \{v\}$ and append this order after v.

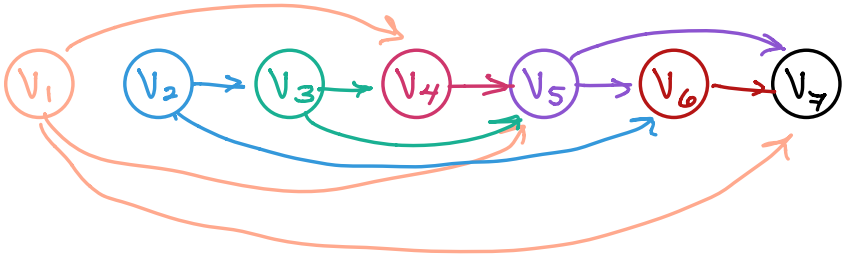
Draw the topological ordering for each DAG

A)

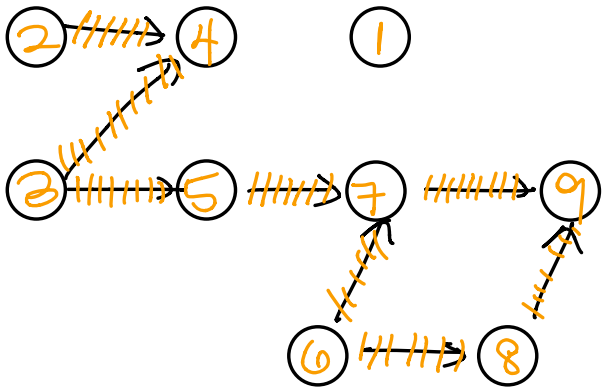


① vertices w/out incoming edges

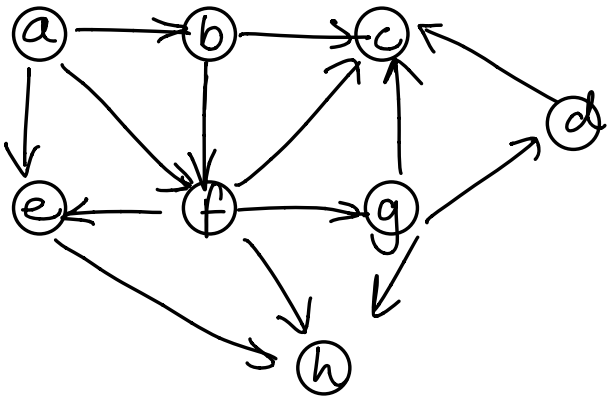
V_2 & V_1



B)



c)



we find a topological order in $O(V+E)$ time!