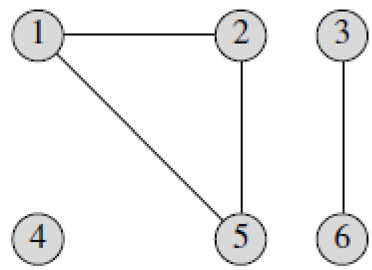
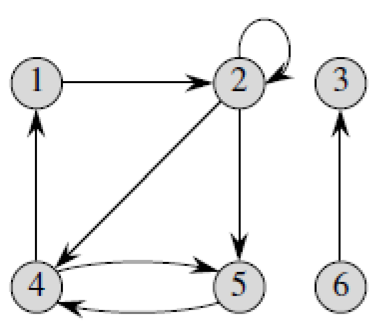
Connectedness:

- An undirected graph is connected if every pair of vertices is connected by a path.
  Below graph has three connected components: {1, 2, 5}, {3, 6}, and {4}. Every vertex in {1, 2, 5} is reachable from every other vertex in {1, 2, 5}.



- The connected components of a graph are the **equivalence classes** of vertices under the "**is reachable from**" relation.
- An undirected graph is connected if it has exactly **one connected component**, that is, if every vertex is reachable from every other vertex.

- A directed graph is **strongly connected** if every two vertices are reachable from each other.
  Below graph has three strongly connected components: {1, 2, 4, 5}, {3}, and {6}. All pairs of vertices in {1, 2, 4, 5} are mutually reachable. The vertices {3, 6} do not form a strongly connected component, since vertex 6 cannot be reached from vertex 3.

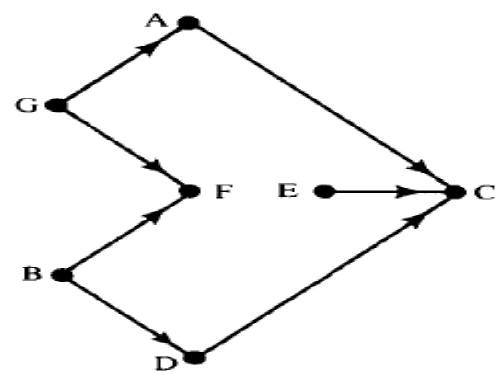


- The ***strongly connected components*** of a directed graph are the **equivalence classes** of vertices under the "**are mutually reachable**" relation.
- A directed graph is strongly connected if it has only **one strongly connected** component.

Topological Sorting:

- Let S be a directed graph with the following two properties:
    1) Each vertex v_i of S represents a task.
    2) Each (directed) edge (u, v) of S means that task u must be completed before beginning task v.
- We note that such a graph S cannot contain a cycle, such as P = (u, v, w, u), since, otherwise, we would have to complete u before beginning v, complete v before beginning w, and complete w before beginning u. That is, we cannot begin any of the three tasks in the cycle.

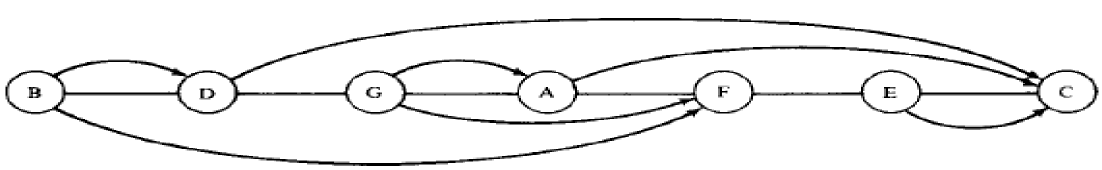
- Such a graph S, which represents **tasks and a prerequisite relation** and which cannot have any cycles, is said to be cycle-free or acyclic. A directed acyclic (cycle-free) graph is called a **dag** for short. Figure below is an example of such a graph.



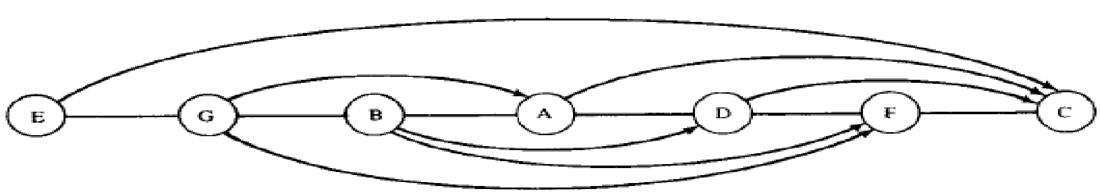
Adjacency lists	
A:	C
B:	D, F
C:	
D:	C
E:	C
F:	
G:	A, F

- A fundamental operation on a dag S is to process the vertices one after the other so that the vertex u is always processed before vertex v whenever (u, v) is an edge. Such a **linear ordering** T of the vertices of S, which may not be unique, is called a topological sort.

- Below Figure shows two topological sorts of the above graph S. We have included the edges of S in the below figure to show that they agree with the direction of the linear ordering.



(a)



(b)

- A topological sort of a dag G = (V, E) is a linear ordering of all its vertices such that if G contains an edge (u, v), then u appears before v in the ordering (If the graph is not acyclic, then no linear ordering is possible).

- A topological sort of a graph can be viewed as an ordering of its vertices along a horizontal line so that all directed edges go from left to right (Topological sorting is thus different from the usual kind of "sorting").

- Usage: Directed acyclic graphs are used in many applications to indicate precedences among events.

Theorem: Let S be a finite directed cycle-free graph. Then there exists a topological sort T of the graph S.

Note that the theorem states only that a topological sort exists. We now give an algorithm which will find a topological sort. The **main idea** of the algorithm is that any vertex (node) N with zero **indegree** may be chosen as the first element in the sort T. The algorithm essentially repeats the following two steps until S is empty:

(1) Find a vertex N with zero indegree.
(2) Delete N and its edges from the graph S.

We use an auxiliary QUEUE to temporarily hold all vertices with zero degree.

Algorithm 9.4: The algorithm finds a topological sort T of a directed cycle-free graph S.

Step 1. Find the indegree INDEG(N) of each vertex N of S.

Step 2. Insert in QUEUE all vertices with zero degree.

Step 3. Repeat Steps 4 and 5 until QUEUE is empty.

Step 4. Remove and process the front vertex N of QUEUE.

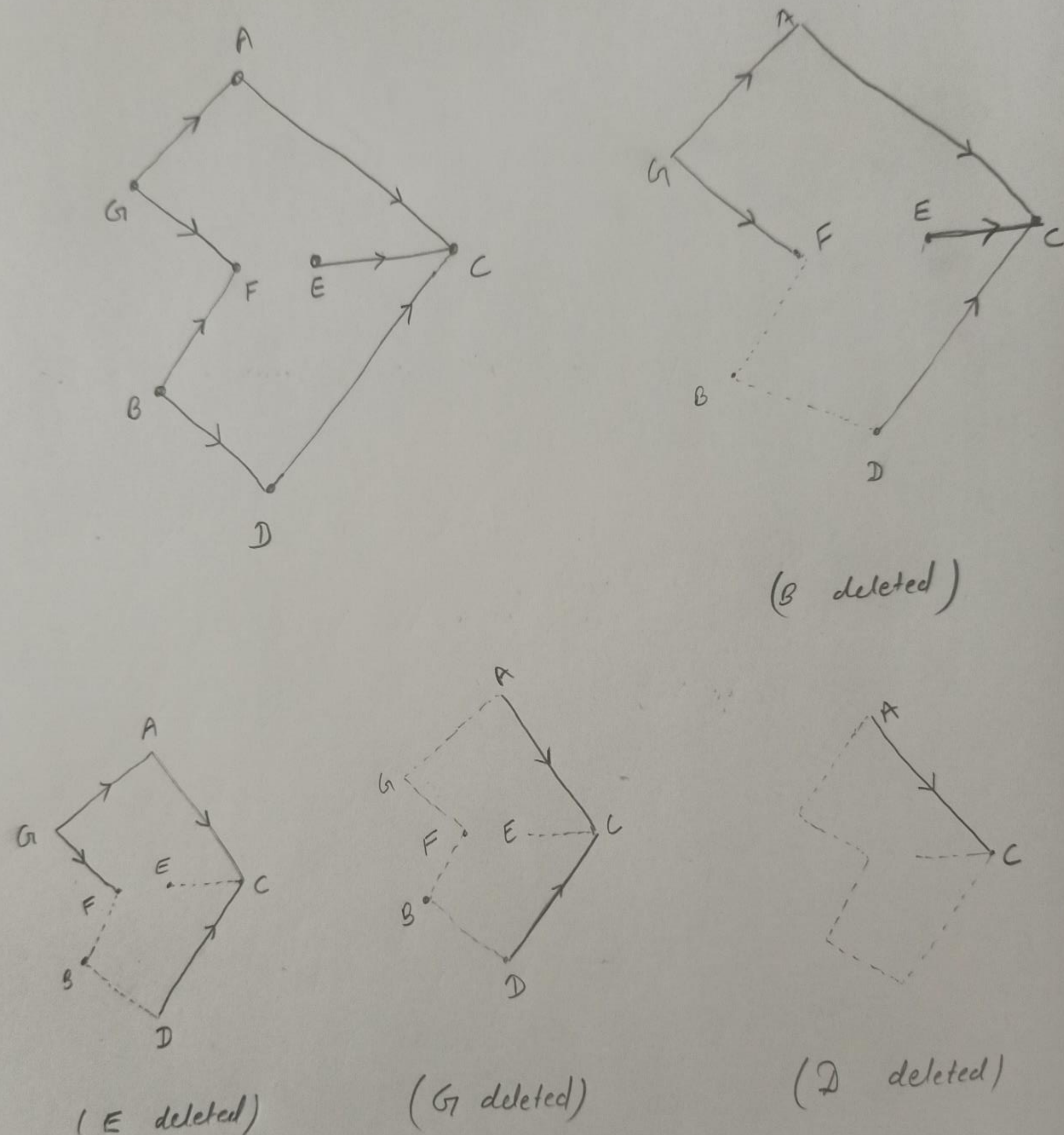
Step 5. Repeat for each neighbor M of the vertex N.
(*a*) Set INDEG(M) : = INDEG(M) − 1.
[This deletes the edge from N to M.]

(*b*) If INDEG(M) = 0, add M to QUEUE.
[End of loop.]
[End of Step 3 loop.]

Step 6. Exit.

Example:

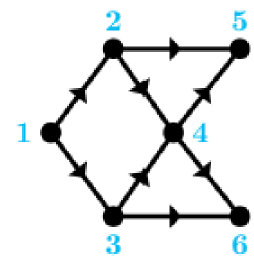
Vertex		B	E	G	D	A	F	C
Queue	GEB	DGE	DG	FAD	FA	CF	C	ϕ



Problems:

Q1.

Consider the DAG with $V = \{1,2,3,4,5,6\}$ shown below.

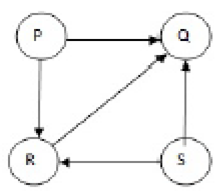


Which of the following is not a topological ordering?

A. 1 2 3 4 5 6   B. 1 3 2 4 5 6   C. 1 3 2 4 6 5   D. 3 2 4 1 6 5

Q2.

Consider the directed graph below given.



Which one of the following is **TRUE**?

A. The graph does not have any topological ordering.
B. Both PQRS and SRQP are topological orderings.
C. Both PSRQ and SPRQ are topological orderings.
D. PSRQ is the only topological ordering.