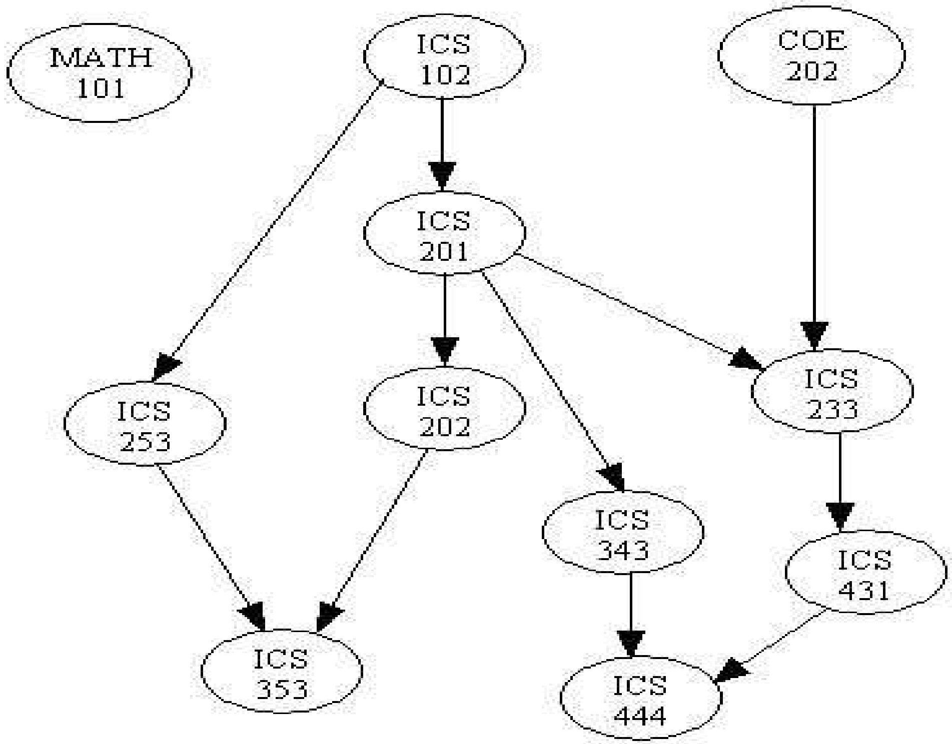
Topological Sort

Introduction

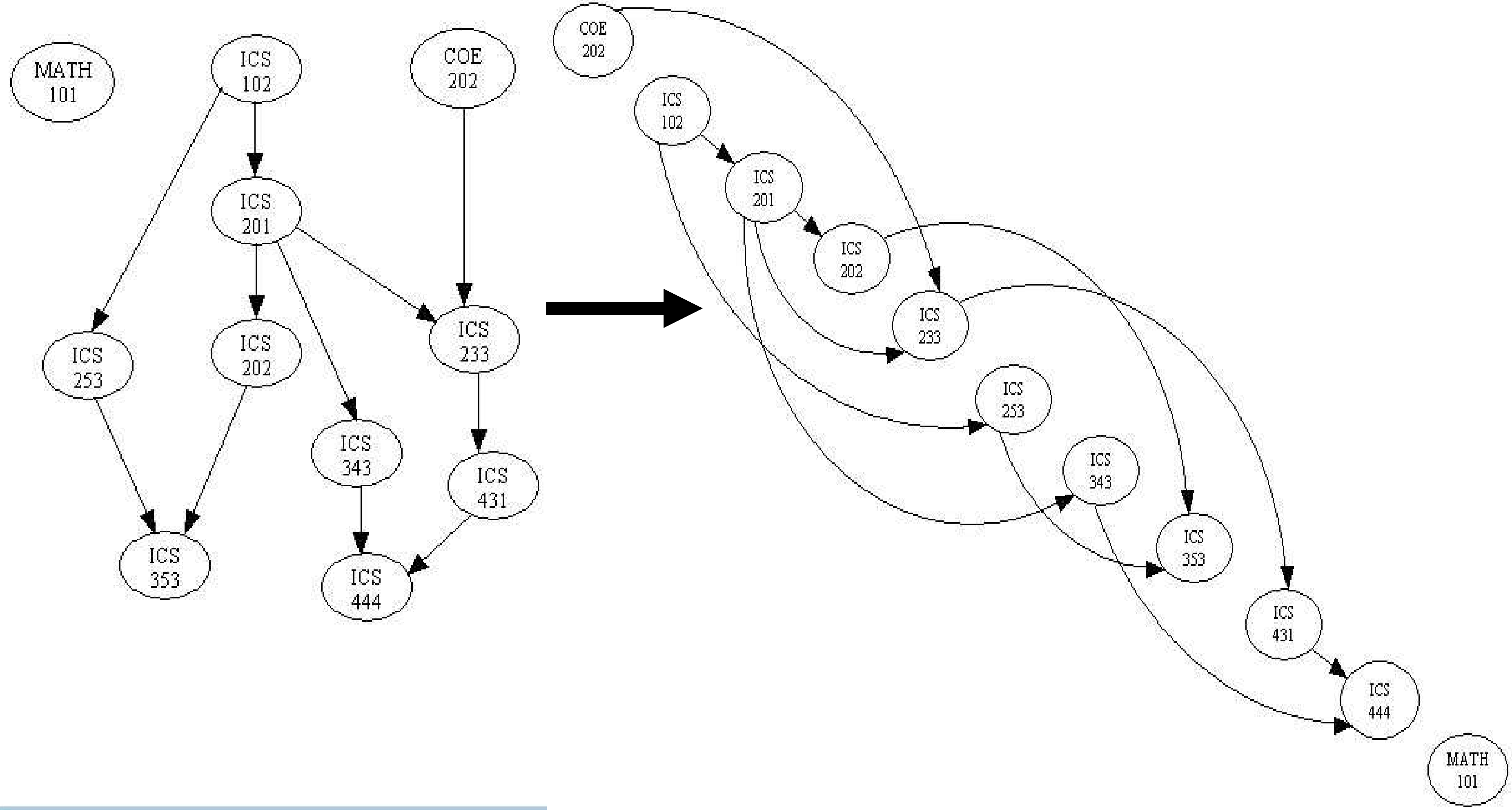
- There are many problems involving a set of tasks in which some of the tasks must be done before others.
- For example, consider the problem of taking a course only after taking its prerequisites.
- Is there any systematic way of linearly arranging the courses in the order that they should be taken?



Yes! - Topological sort.

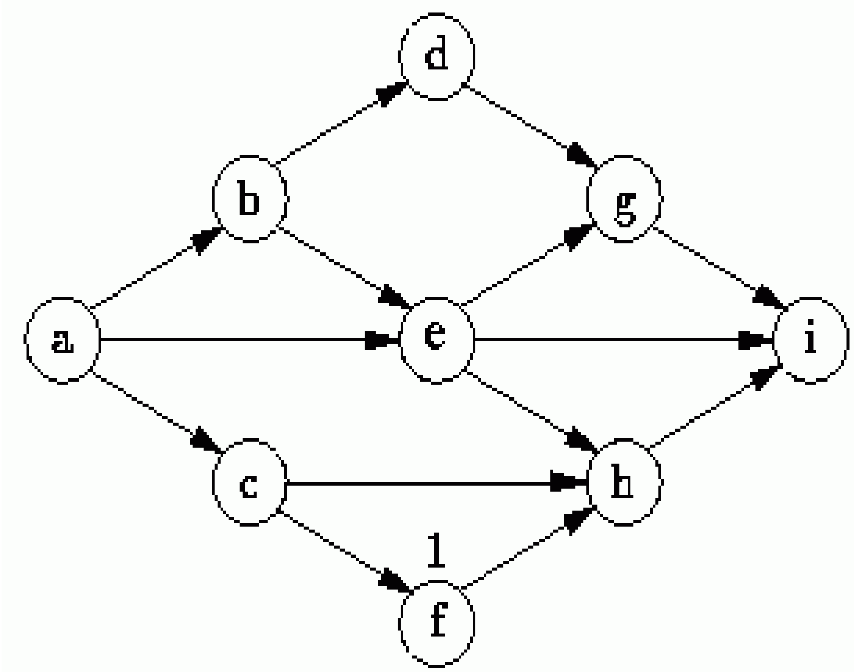
Definition of Topological Sort or Topological Ordering

- Topological sort is a method of ordering the vertices in a directed acyclic graph (DAG), as a sequence, such that if there is a path from v_i to v_j, then v_j appears after v_i in the ordering.
- The graph in (a) can be topologically sorted as in (b)



Topological Sort is not unique

- Topological sort is not unique.
- The following are all topological sort of the graph below:



s1 = {a, b, c, d, e, f, g, h, i}

s2 = {a, c, b, f, e, d, h, g, i}

s3 = {a, b, d, c, e, g, f, h, i}

s4 = {a, c, f, b, e, h, d, g, i}

etc.

Topological Sort Algorithm

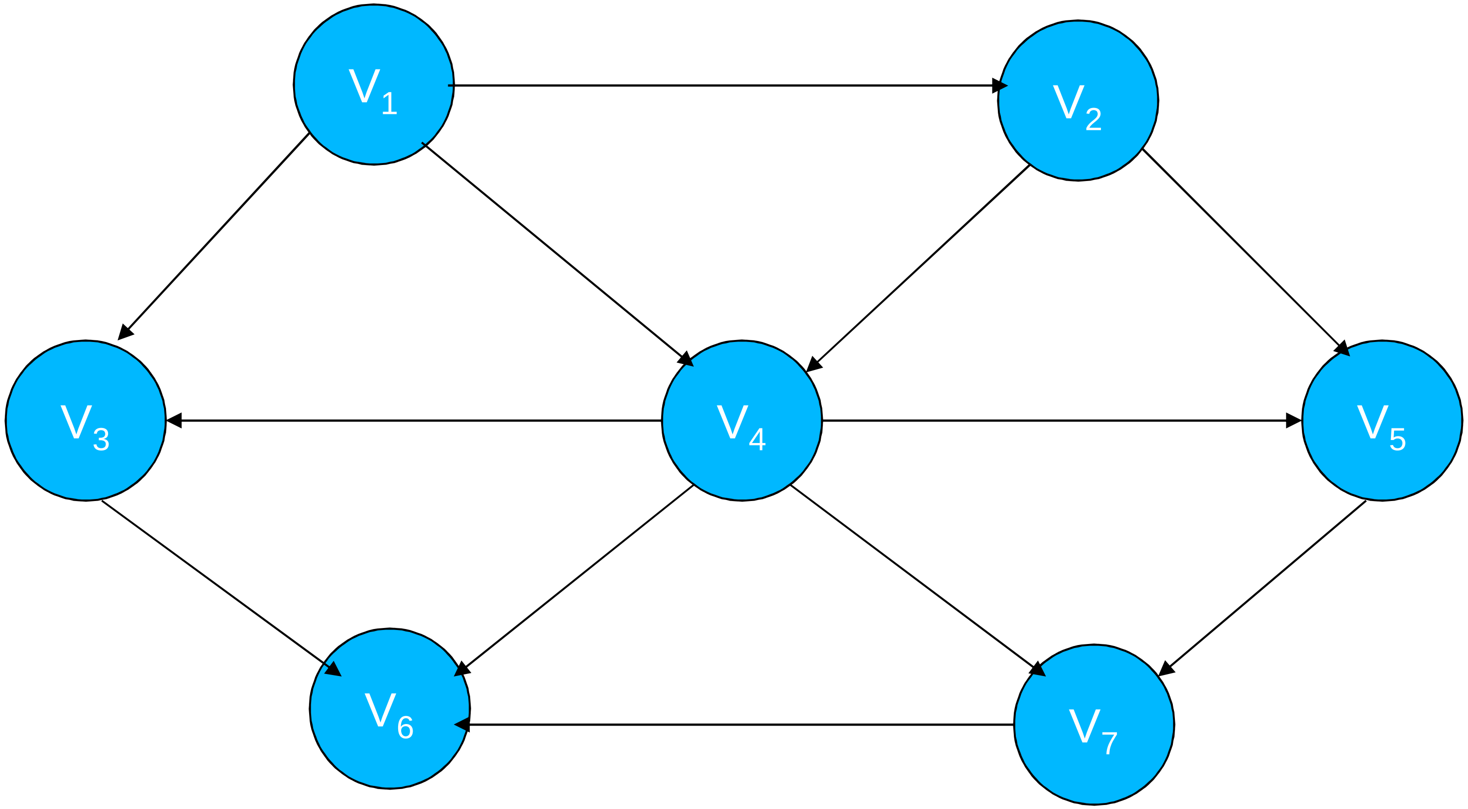
- One way to find a topological sort is to consider in-degrees of the vertices.
- The first vertex must have in-degree zero -- every DAG must have at least one vertex with in-degree zero.
- The Topological sort algorithm is

```
int topologicalOrderTraversal( )
{
    int numVisitedVertices = 0;
    while(there are more vertices to be visited)
       {
        if(there is no vertex with in-degree 0)
             break;
        else
        {
         select a vertex v that has in-degree 0;
         visit v;
         numVisitedVertices++;
         delete v and all its emanating edges;
        }
       }
return numVisitedVertices;
}
```

Pseudocode to perform Topological Sort

```
void toposort (graph g)
{
    queue q;
    int counter=0;
    vertex v,w;
    q=createqueue(numvertex);
    makeempty(q);
    for each vertex v
            if(indegree[v]==0)
                    enqueue(v,q);
    while(!isempty(q))
    {
            v=dequeue(q);
            topnum[v]=++counter;
            for each w adjacent to v
                    if(--indegree[w]==0)
                            enqueue(w,q);
    }
    if(counter!=numvertex)
      printf("Graph is cyclic");
    disposequeue(q);
}
```

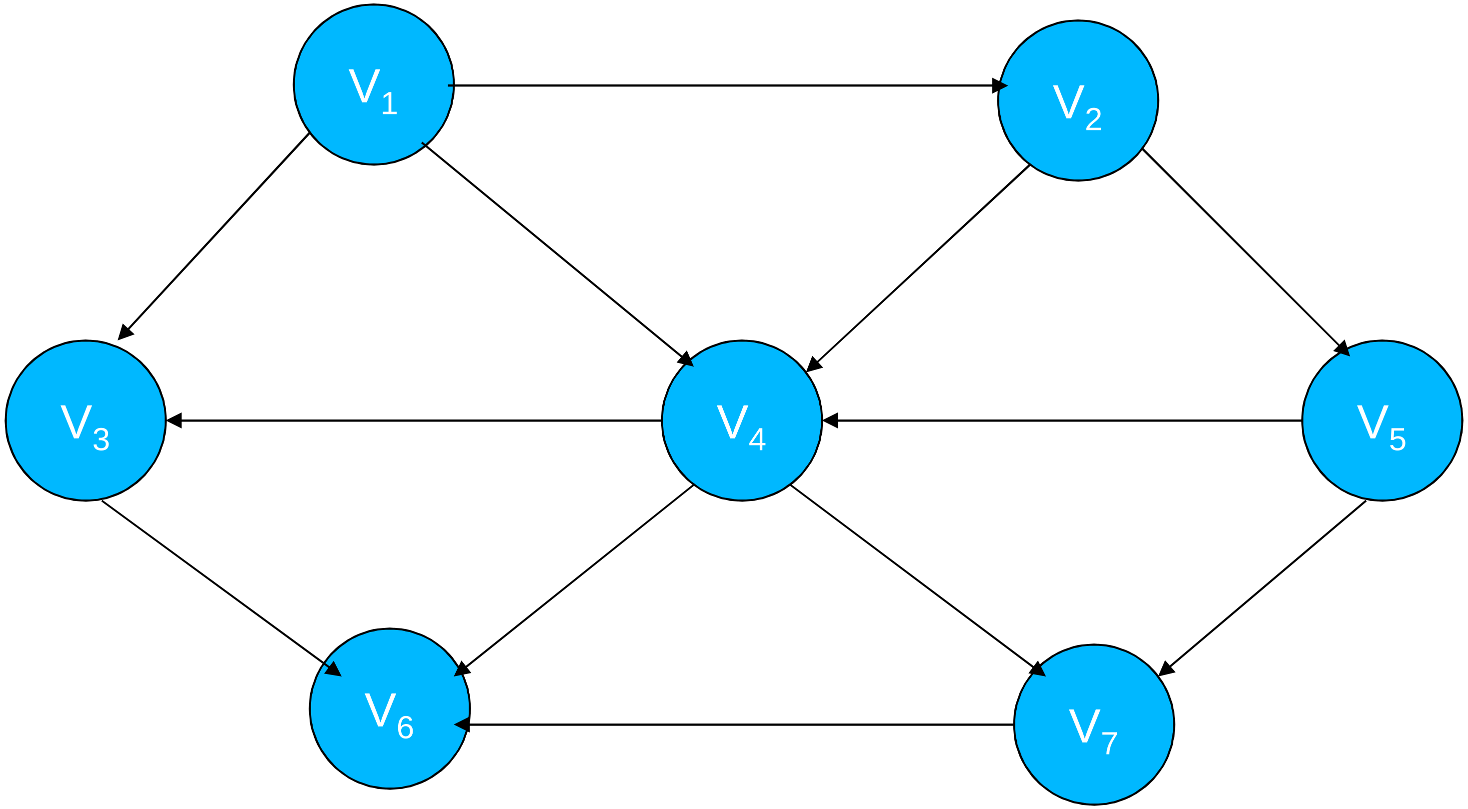
Example 1-An Acyclic Graph



Result of applying Toposort to the graph shown in previous slide

	Indegree Before Dequeue						
Vertex	**I**	**II**	**III**	**IV**	**V**	**VI**	**VII**
V_1	0	0	0	0	0	0	0
V_2	1	0	0	0	0	0	0
V_3	2	1	1	0	0	0	0
V_4	2	1	0	0	0	0	0
V_5	2	2	1	0	0	0	0
V_6	3	3	3	2	1	1	0
V_7	2	2	2	1	1	0	0
Enqueue	V_1	V_2	V_4	V_3, V_5		V_7	V_6
Dequeue	V_1	V_2	V_4	V_3	V_5	V_7	V_6

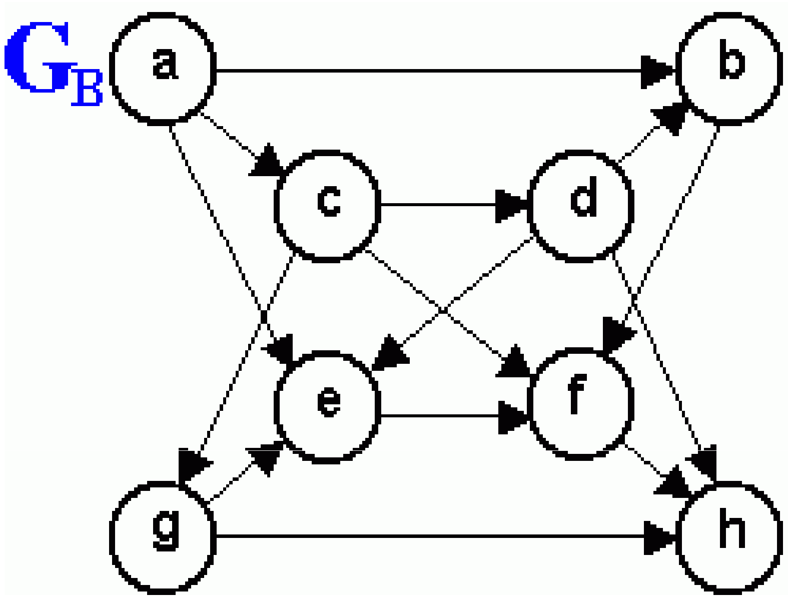
Example 2-An Acyclic Graph



Result of applying Toposort to the graph shown in previous slide

	Indegree Before Dequeue						
Vertex	**I**	**II**	**III**	**IV**	**V**	**VI**	**VII**
V_1	0	0	0	0	0	0	0
V_2	1	0	0	0	0	0	0
V_3	2	1	1	1	0	0	0
V_4	3	2	1	0	0	0	0
V_5	1	1	0	0	0	0	0
V_6	3	3	3	3	2	1	0
V_7	2	2	2	1	0	0	0
Enqueue	V_1	V_2	V_5	V_4	V_3, V_7		V_6
Dequeue	V_1	V_2	V_5	V_4	V_3	V_7	V_6

Exercise



List the order in which the nodes of the directed graph G_B are visited by topological order traversal that starts from vertex a.