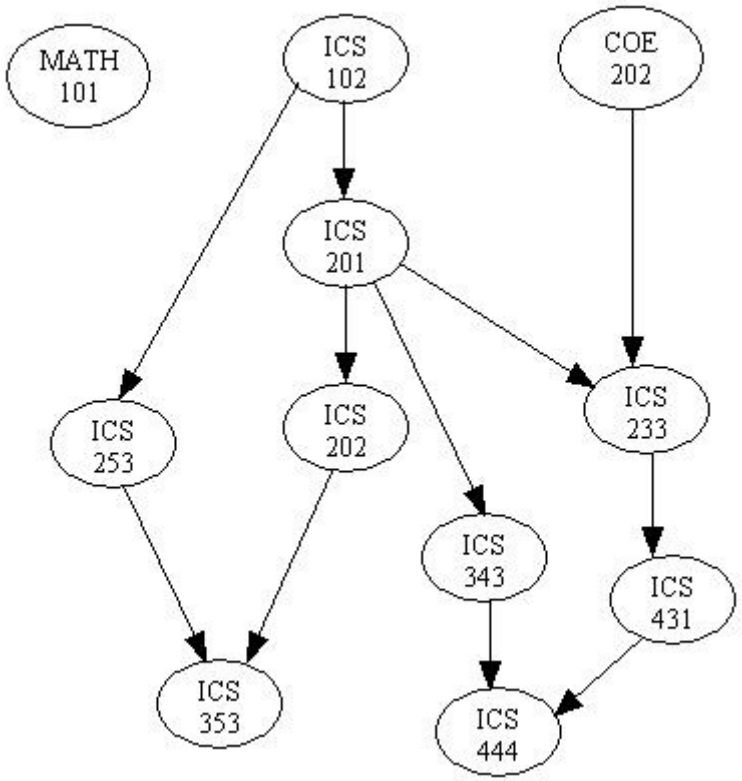
Topological Sort

- Introduction.
- Definition of Topological Sort.
- Topological Sort is Not Unique.
- Topological Sort Algorithm.
- An Example.
- Implementation.
- Review Questions.

Introduction

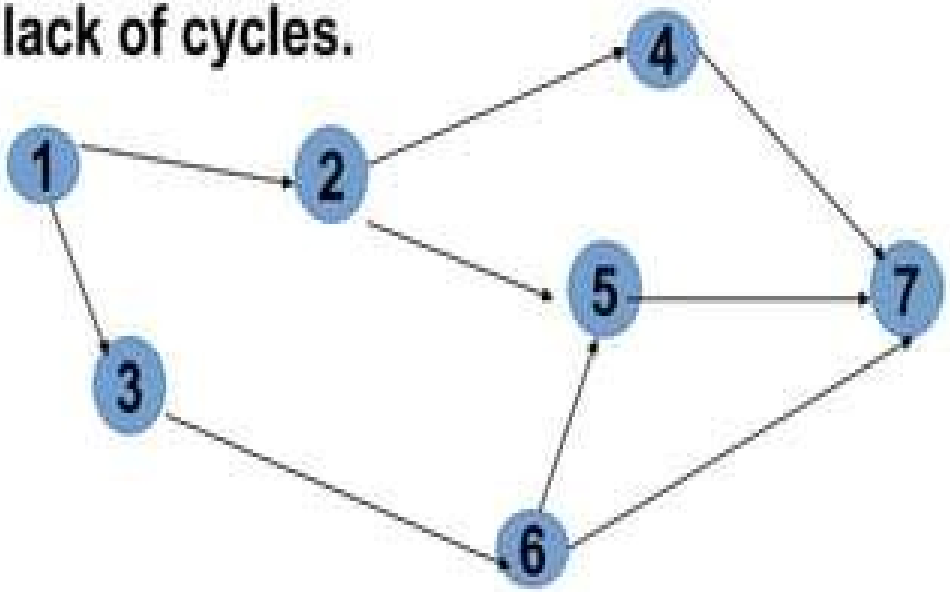
- There are many problems involving a set of tasks in which some of the tasks must be done before others.
- For example, consider the problem of taking a course only after taking its prerequisites.
- Is there any systematic way of linearly arranging the courses in the order that they should be taken?



.Yes! - Topological sort

Directed Acyclic Graph

A directed acyclic graph is an acyclic graph that has a direction as well as a lack of cycles.



Vertices Set:
{1,2,3,4,5,6,7}

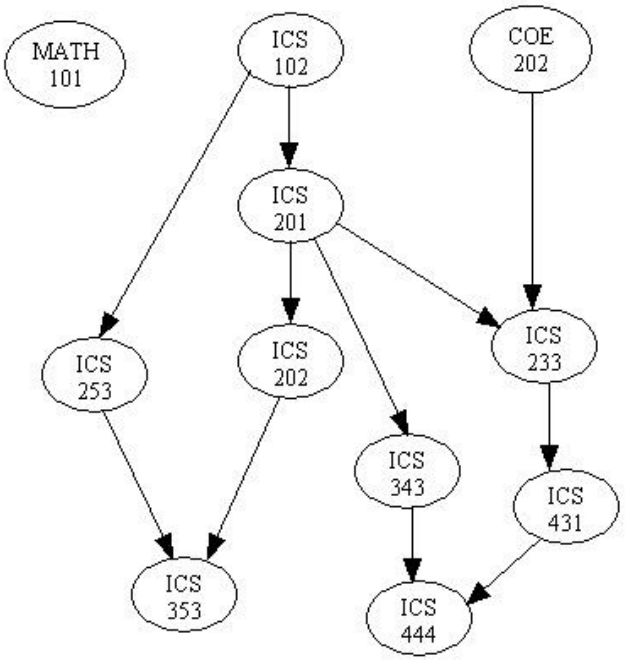
Edge Set: {(1,2),(1,3),(2,4),
(2,5),(3,6),(4,7),
(5,7),(6,7)}

A directed acyclic graph has a topological ordering. This means that the nodes are ordered so that the starting node has a lower value than the ending node.

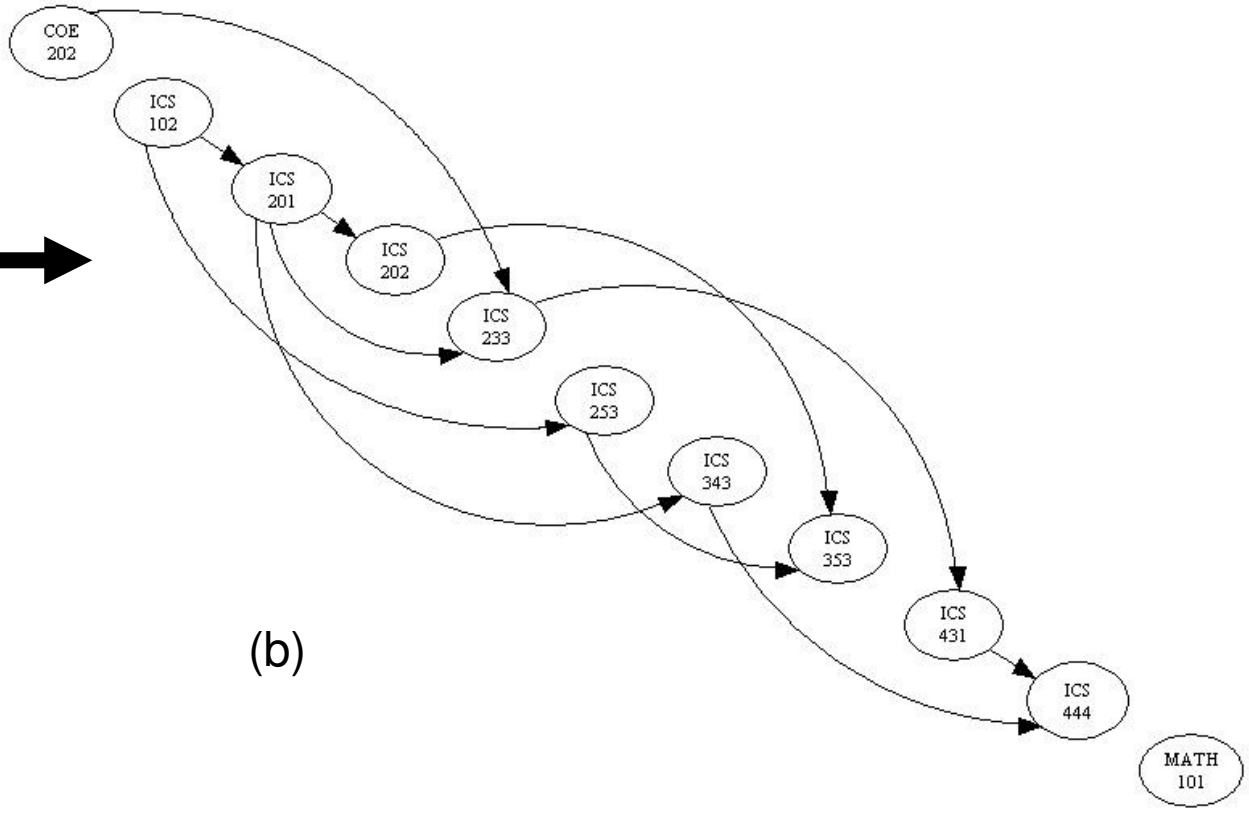
Definition of Topological Sort

- Topological sort is a method of arranging the vertices in a directed acyclic graph (DAG), as a sequence, such that no vertex appear in the sequence before its predecessor.

- The graph in (a) can be topologically sorted as in (b)



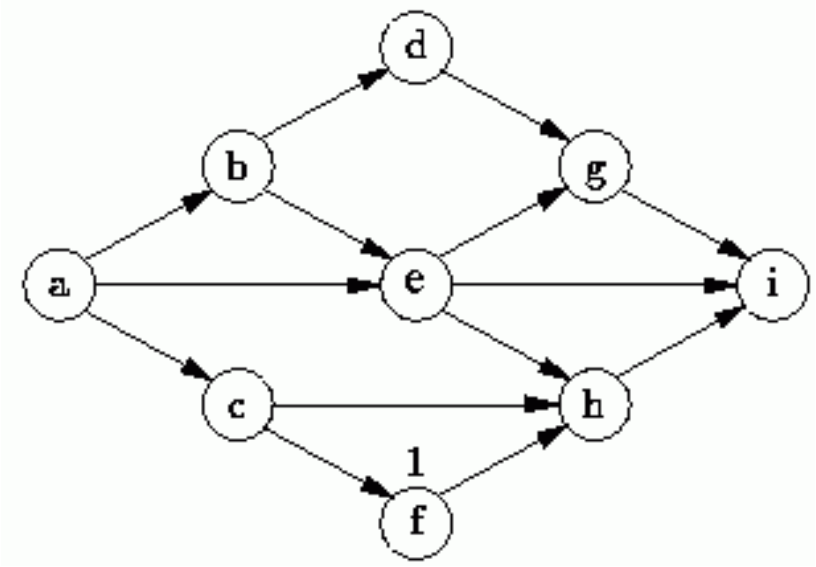
(a)



(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.

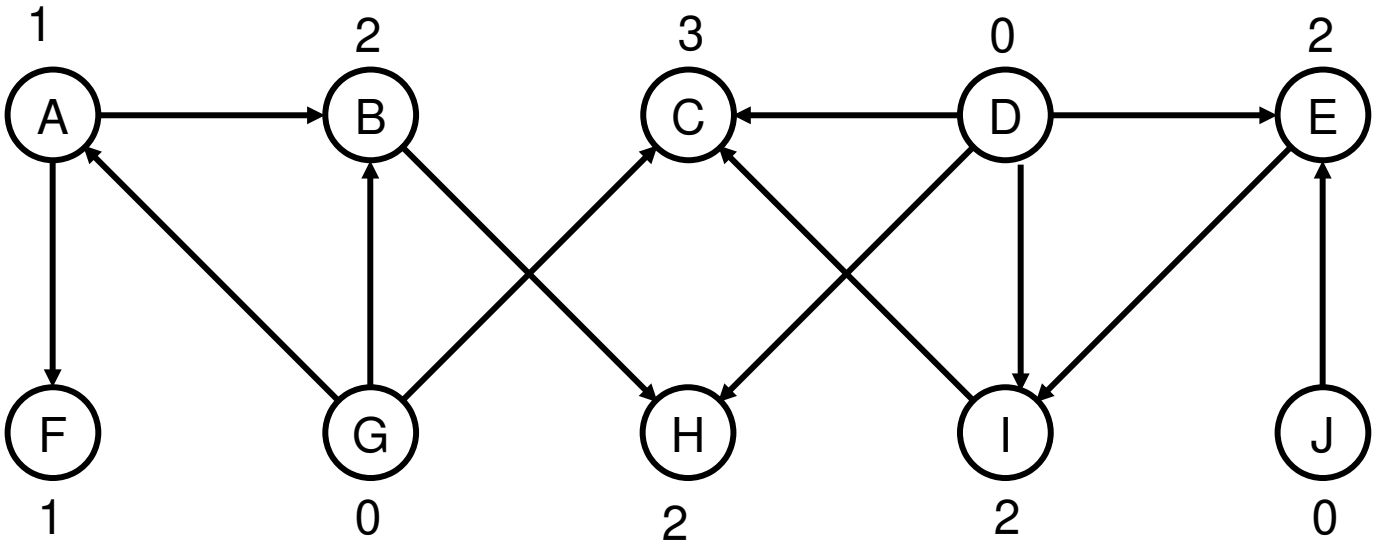
Topological Sort Algorithm

- 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:

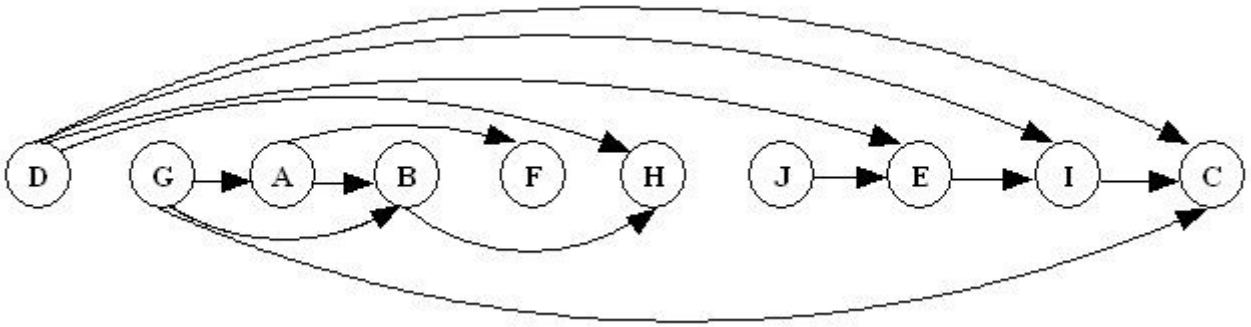
```
int topologicalOrderTraversal( ){
    int numVisitedVertices = 0;
    while(there are more vertices to be visited){
        if(there is no vertex with in-degree 0)
            break;
        else{
         select a vertex v that has in-degree 0;
         visit v;
         numVisitedVertices++;
 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

   return numVisitedVertices;
}
```

Topological Sort Example

- **Demonstrating Topological Sort.**

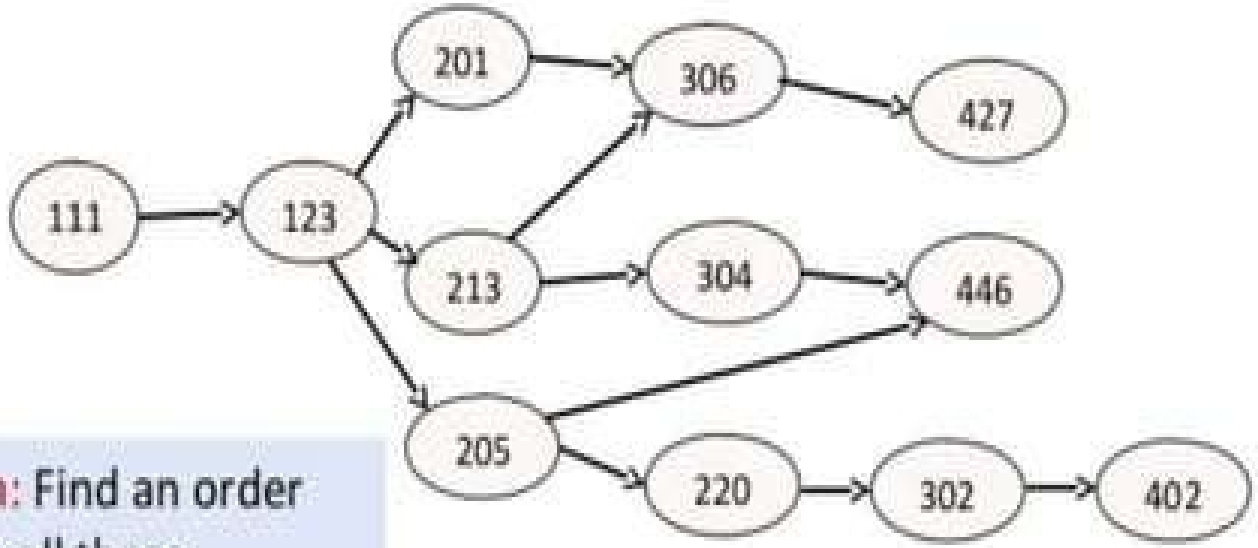


D	G	A	B	F	H	J	E	I	C



Contd...

- Consider the following graph of course prerequisities



Problem: Find an order in which all these courses can be taken.

Example: 111, 123, 201, 213, 304, 306, 427, 205, 446, 220, 302, 402

- To take a course, all of its prerequisites must be taken first

10

Algorithm

1 - Compute the indegrees of all vertices

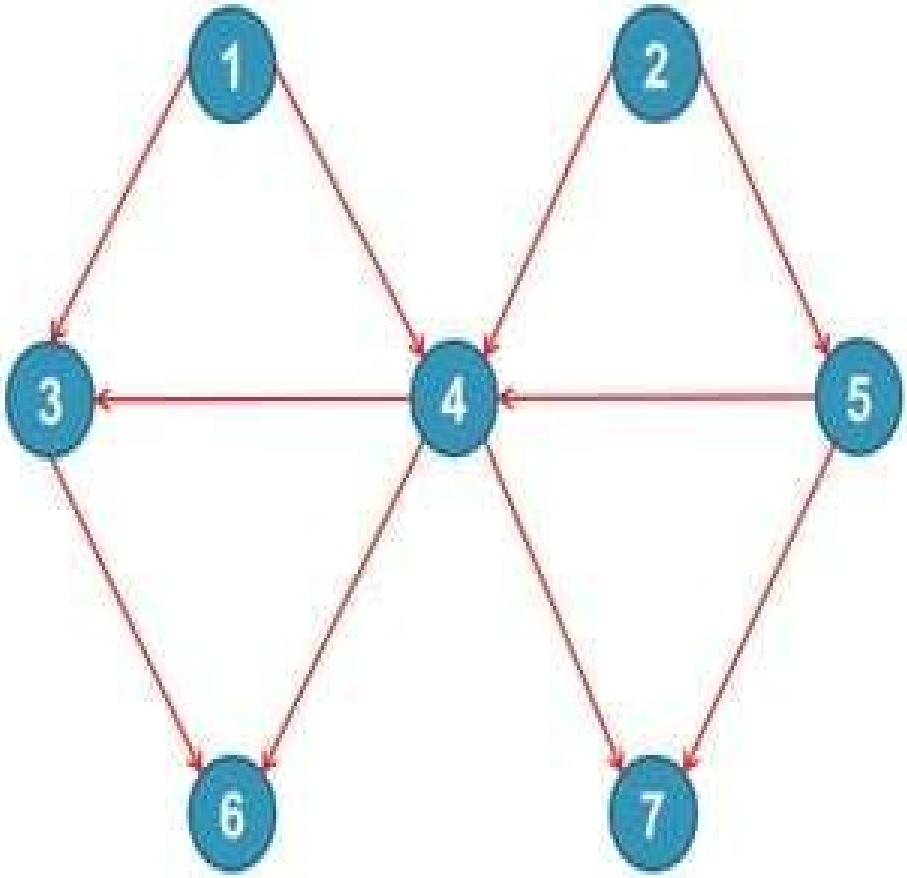
2 - Find a vertex U with indegree 0 and print it (store it in the ordering)If there is no such vertex then there is a cycle and the vertices cannot be ordered. Stop.

3 - Remove U and all its edges (U,V) from the graph.

4 - Update the indegrees of the remaining vertices.

5 - Repeat steps 2 through 4 while there are vertices to be processed

Example



Identify nodes having in degree '0'

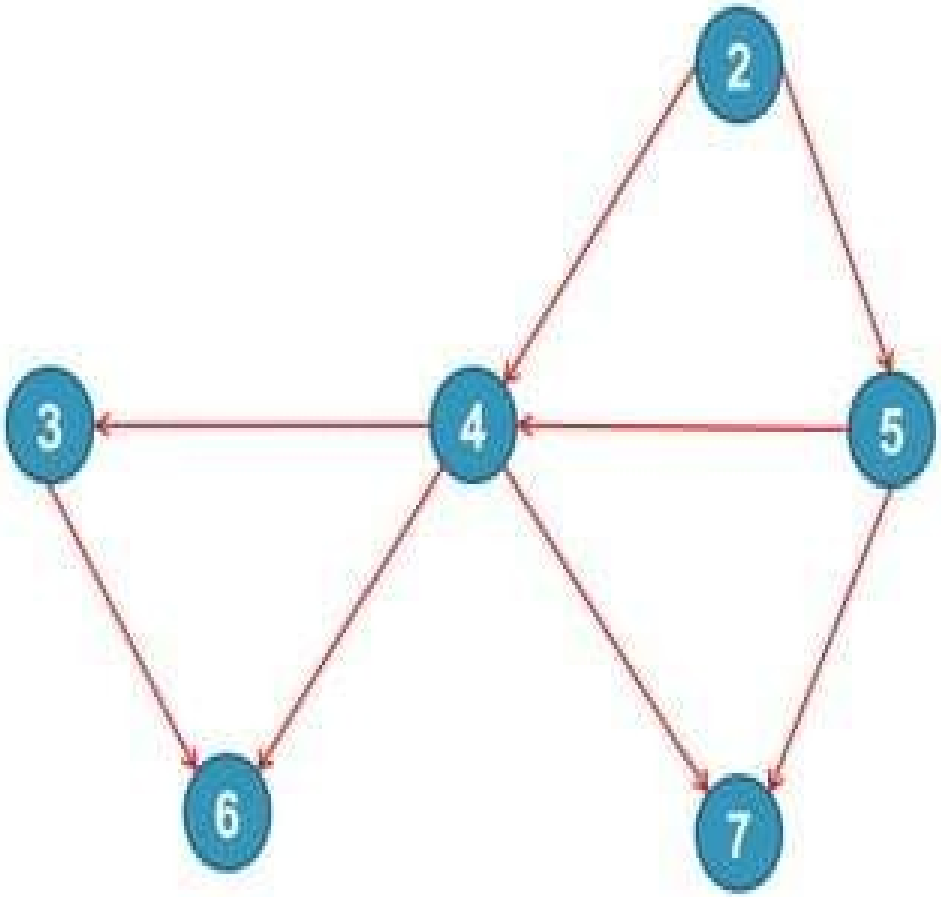
Select a node and delete it with its edges then add node to output

Select Node : 1

Output :

1

Contd...



Identify nodes having in degree '0'

Select a node and delete it with its edges then add node to output

Select Node : 2

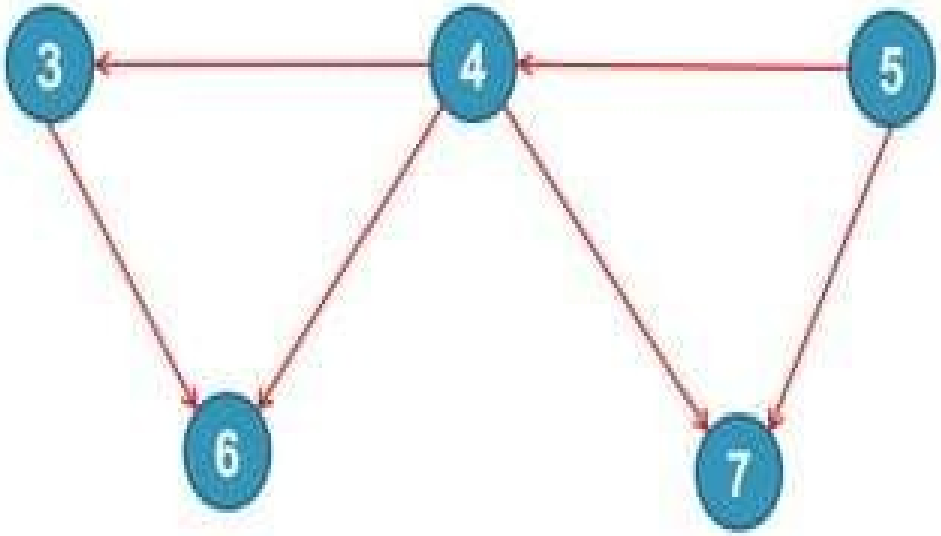
Output :

1 2

Contd...

Identify nodes having in degree '0'

Select a node and delete it with its edges then add node to output

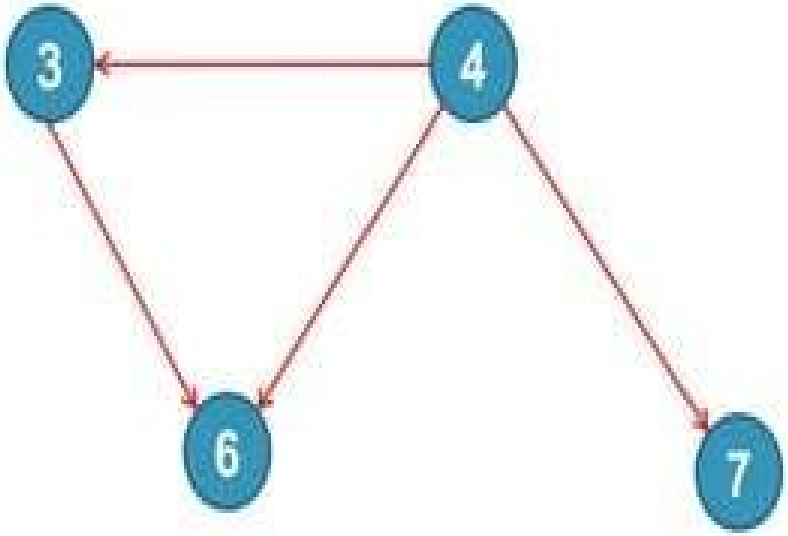


Select Node : 5

Output :

1 2 5

Contd...



Identify nodes having in degree '0'

Select a node and delete it with its edges then add node to output

Select Node : 4

Output :

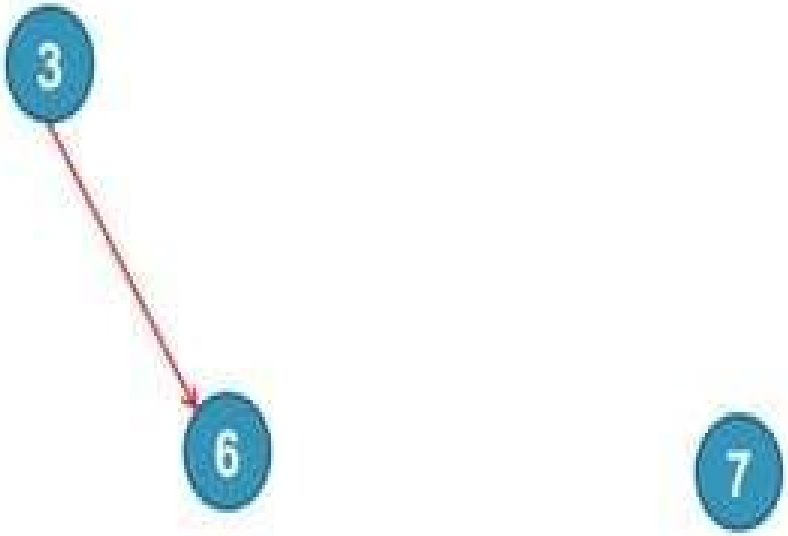
1 2 5 4

Contd...

Identify nodes having in degree '0'

Select a node and delete it with its edges then add node to output

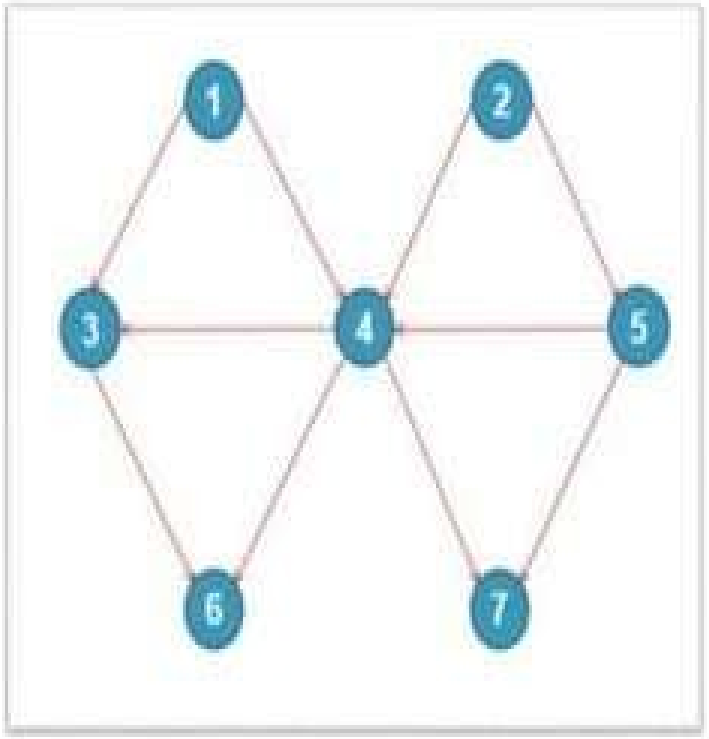
Select Node : 3



Output :

1 2 5 4 3

Contd...



Identify nodes having in degree '0'

Select a node and delete it with its edges then add node to output

Select Node : 7

6

7

Output :

1 2 5 4 3 6 7

Applications

- ***1] Build Systems*** :
- We have various IDE's like Ecllipse , Netbeans etc. We have to build a project which has many libraries dependent on each
- other then IDE uses Topological Sort to decide which library to
- include or build first.

- ***2] Advanced-Packaging Tool(apt-get) :***
- In Linux apt-get is used to install softwares in the system. The command
- "apt-get install VLC" is used. It is the way in Linux to install and remove
- softwares.

Contd...

- ***3] Task Scheduling*** :
- Topological Sort is helpful in scheduling interdependent task to know which rask should proceed which one.

- ***4] Pre- Requisite Problems:***
- We often come across situations where we need to finish one job in order to
- proceed the next one. For ex, In University structure, we need to complete
- basic Algorithm course to study an Advance Algorithm course. So, there
- exist a pre- requisite and we can know this by doing a topological sort on all.

Implementation of Topological Sort

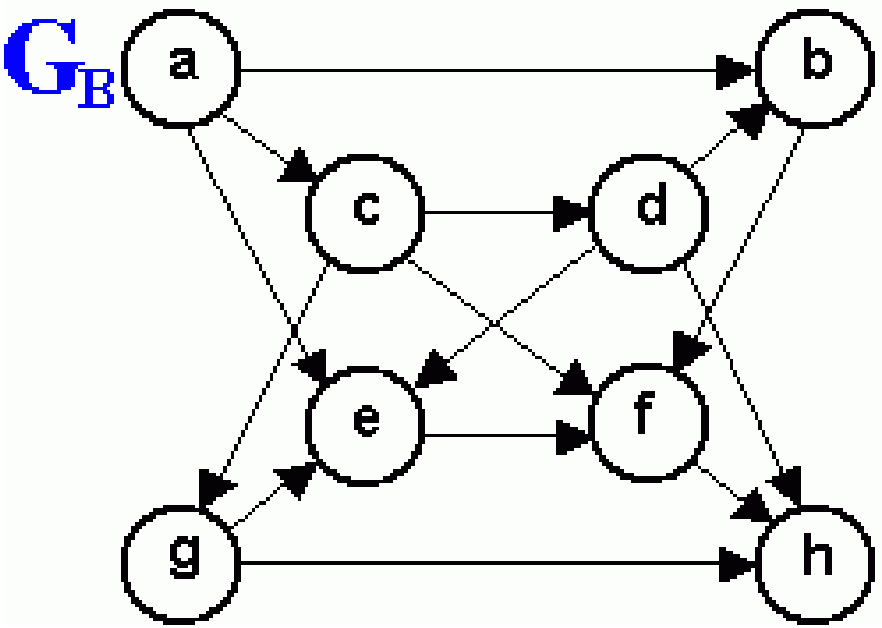
- The algorithm is implemented as a traversal method that visits the vertices in a topological sort order.
- An array of length |V| is used to record the in-degrees of the vertices. Hence no need to remove vertices or edges.
- A priority queue is used to keep track of vertices with in-degree zero that are not yet visited.

```
public int topologicalOrderTraversal(Visitor visitor){
   int numVerticesVisited = 0;
   int[] inDegree = new int[numberOfVertices];
   for(int i = 0; i < numberOfVertices; i++)
      inDegree[i] = 0;

   Iterator p = getEdges();
   while (p.hasNext()) {
      Edge edge = (Edge) p.next();
      Vertex to = edge.getToVertex();
      inDegree[getIndex(to)]++;
   }
```

Implementation of Topological Sort

```
   BinaryHeap queue = new
BinaryHeap(numberOfVertices);
   p = getVertices();
   while(p.hasNext()){
      Vertex v = (Vertex)p.next();
      if(inDegree[getIndex(v)] == 0)
         queue.enqueue(v);
   }

   while(!queue.isEmpty() && !visitor.isDone()){
      Vertex v = (Vertex)queue.dequeueMin();
      visitor.visit(v);
      numVerticesVisited++;
      p = v.getSuccessors();
      while (p.hasNext()){
         Vertex to = (Vertex) p.next();
         if(--inDegree[getIndex(to)] == 0)
            queue.enqueue(to);
      }
   }
   return numVerticesVisited;
}
```

Review Questions



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