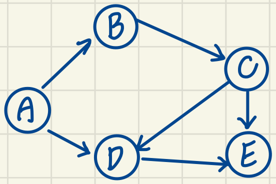
11) Define an iterator class *TopoIterator* for iterating through the vertices of a directed acyclic graph in topological order.



indegree: A=0 B=1 C=1 D=2 E=2.

E(G)={(A.B).(B.C).(C.D).(C.E).(A.D).(D.E)}.

indegree means # times each element be point to.

In Topological order. the element who's indegree is 0 may be the head of the sequence. after sort into sequence. then delete the node. and so on, until all the node be delete.

```
class TopoIterator{
  private:
    int V;
    list<int>*adj; // Pointer to an array containing adjacency list
  public:
    Graph(int V);
    void addEdge(int u, int v);
    //Function to add an edge to graph.
    void topologicalSort();
};
```

```
TopoIterator::Graph(int V)
{
   this->V=V
   adj= new list<int>[V];
}

void TopoIterator::addEdge(int u, int v)
{
   adj[u].push back(v);
}

void TopoIterator::topologicalSort()
{
   vector<int> indegree(V,0); //initial all indegrees as 0.
   for(int u=0; u<V; u++)
   {
      list<int>::iterator itr;
      for(itr=adj[u].begin(); itr!=adj[u].end(); itr++)
         indegree[*itr]++;
   }

   queue<int> q;
   // Create an queue and enqueue all vertices with indegree 0
```

```
for(int i=0; i<V; i++)
    if(indegree[i]==0) q.push(i)
int cnt=0;
vector<int> top_order;
while(!q.empty())
{
    int u=q.front();
    q.pop();
    top_order.push_back(u);
    list<int>::iterator itr;
    for(itr=adj[u].begin(); itr!=adj[u].end(); itr++)
        if(--indegree[*irt]==0) q.push(*itr);
    cnt++;
}
if(cnt!=V)
{
    cout<<"There exists a cycle in the graph\n";
    return;
}
```

```
  for (int i=0; i<top_order.size(); i++)
     cout<< top_order[i]<<"   ";
  cout<<endl;
};
```