4/11 Topological Sorting for DAGS:

DAGS (directed acyclic graphs) are directed graphs with no directed cycle.
suppose there is a set of tasks that need to be performed one at a time. Some tasks depend on other tasks and they cannot be started until the other tasks are completed. All the dependencies are known and we want to arrange a schedule for performing the tasks which is consistent with the dependencies.
We can associate a directed graphs whose vertices are the tasks and whose edges are the dependencies. The graph is acyclic otherwise some tasks can never be started. This problem is the topological sorting.
The problem: given a DAG with n vertices, label the vertices from 1 to n such that if v is labeled k, then all vertices that can be reached from v by a directed path are labeled with labels > k.
First one lemma: A DAG always contains a vertex of indegree 0.
Pf: the proof is similar to that of trees have leaves. otherwise we can traverse the grah for ever if we do not have a cycle.
Note that by the same argument, there is a vertex with out degree 0.
Thus the algorithm is as follows: first we find a vertex of indegree 0. once we find it, we label it with 1 and remove its adjacent edges and label the rest of the graph, which is still acyclic, by induction.

```
Algorithm Topological-sorting:
Input: DAG G; output: Labeling of G;
begin
  Initialize v.indegree for all vertices; {e.g., by DFS}
  G_label := 0;
  for i:=1 to n do
    if v_i.Indegree = 0 then put v_i in Queue;
  while Queue is not empty
    remove a vertex v from Queue;
    G_label := G_label + 1; v.label := G_label;
    for all edges (v,w) do
        w.Indegree := w.Indegree - 1;
        if w.Indegree = 0 then put w in Queue;
end end end {end-for}
```

Time complexity: Initializing indegrees with DFS takes $O(|V|+|E|)$. Finding a vertex of indegree 0 takes constant time by queue. We consider each edge (v,w) once when we bring v out of the queue. Thus we update Indegree variables at most $|E|$ times. The total running time is $O(|V|+|E|)$ then.