Problem Pattern

1. Since the projects are dependent on one another and all of a project's dependencies must be built before the project is, so what is the underlying ordering of projects? Topological Sort.
2. When will be the answer -1, i.e. when will the ordering of projects will not be possible? Topological Sorting for a graph is not possible if the graph is not a Directed Acyclic Graph.

Problem Approach

Steps to find the topological ordering of graph:

- Compute the in-degree (number of incoming edges) for each of the vertices present in the graph.
- Pick all the vertices with in-degree = 0 and add them into a queue (Enqueue operation)
- Remove a vertex from the queue (Dequeue operation) and then.

1. Decrease in-degree by 1 for all its neighboring nodes.
2. If the in-degree of any neighboring node is reduced to zero, then add it to the queue.

- Repeat Step 3 until the queue is empty.

The order in which the nodes are dequeued is the topological ordering of graph.
Note: If the total number of nodes dequeued is not equal to the number of vertices, then the answer is -1.

Problem Pseudocode

```
buildOrder(project, depend){
   n = project.size();
   for i=0 to n:
       mpp[project[i]]=i+1;
       revmpp[i+1]=project[i];

   m=depend.size();
   for i=0 to m :
       u=mpp[depend[i][0]];
       v=mpp[depend[i][1]];
       adj[u].insert(v);
       indeg[v]++;

   queue<int> q;
   for i = 1 to n:
        if indeg[i]==0:
                q.push(i);

   while !q.empty():
       u=q.front();
       q.pop();
       ans.insert(revmpp[u]);
       for i = 0 to adj[u].size() :
           v = adj[u][i];
           indeg[v]--;
           if indeg[v]==0 :
                       q.push(v);
```

```
 string t='';
   t+='-1';
   a.insert(t);
   if ans.size()!=n :
return a;
   return ans;
}
```