Pseudo code

```
DFS()
    Same as ppt
    Difference:
        If conflict = true
            Then return
DFS_VISIT(Vertex &u)
    Same as ppt
    Difference:
        If visit a gray node
            Then conflict → true
            Return

        Topological_sort
            When a vertex is finished, push_back in out vector
main()
    save vertex into arr by information in first line
    every vertex has a near vector
    push_back the index of classes which have to be taken after the current class
    ex, 1 0 → vertex0.push_back(1)
    DFS()

    If conflict = true
        Then print "Conflict"
        Else
            Then print out vector from behind
```

Result and Analysis

The sooner a vertex turns into black, the later the class should be, the last vertex turn into black should be the root, the most essential class that should be taken before any class is taken.

At first, when we build up the tree(before entering the DFS), we spend θ(V) to push_back every vertex, and θ(E) to push_back every edge, so the time complexity building will be θ(V+E).

Every vertex will only be visited once, and after that, all the edges(ex, if 2 in vertex0.near == 0 → 2, → is an edge) pointing out from it will never be visit again, as an edge will only exist in one vertex's **near** vector, if exist in both, it means that there are two edges between the two vertex, which will cause a conflict, the bonus part that I'll explain in the next paragraph. Furthermore, the vertex will become black to keep others from visiting, so DFS() will at most be **O(V+E)**, as all vertices and edges will only be visited once.

Bonus:

Any connected component should get black from the leaf all the way to root, moreover, when we are working on one connected component, the other component should be all black(already finished) or all white(hasn't start working on); therefore, the colors a edge can find on the other side should only be white and black, if the edge find a gray vertex on the other side, that will be the current vertex's ancestor, which will cause a conflict.