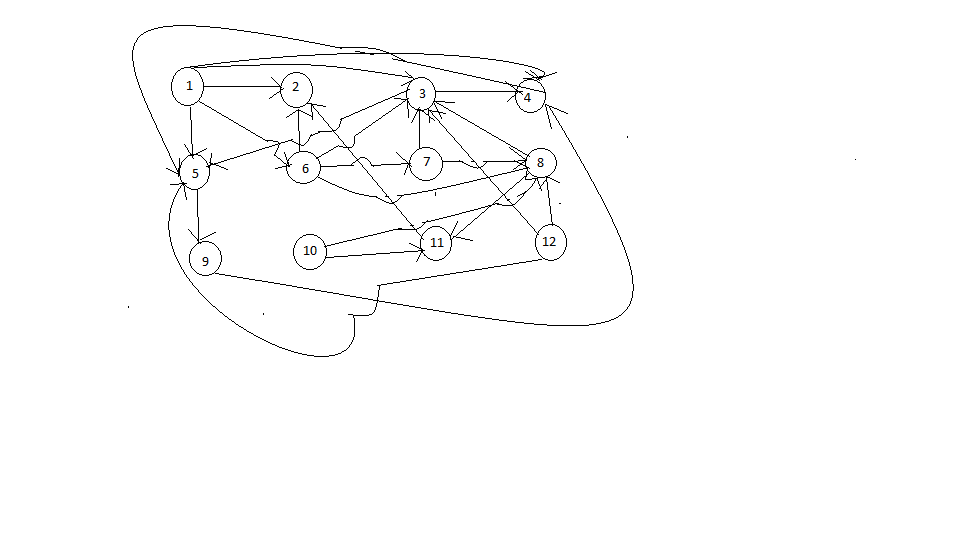
The challenge, which I faced here is with courses Algorithms 1, Algorithms 2 and Programming in Perl, as it has circular dependency problem. I over came the problem with topological sorting method.

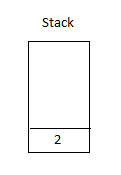
Here you go for the steps which I followed.



I drew a Directed Acyclic Graph from 1 to 12 vertices and an empty stack. At first all the vertices are unvisited. The purpose of the stack is to store the visited vertex which has no neighboring vertex in ascending order.

The first step starts with visiting vertex 1. For loop coursearray[v] holds all the vertices. Since vertex 1 is not visited, continue with other vertices. If the vertex is visited, then set visited[v] = true. Next will find its neighbors. For example, vertex 1 has five neighbors, i.e. vertex 2, 3, 4, 5 and 6. Forward with ascending order, so next neighbor we choose is vertex 2. Here there is not any child to explore from vertex 2. So, vertex 2 push into stack by using command Stack.push(v).

When the program starts, it calls courseSort() method. If the vertex is unvisited, the it calls topologicalCourseSorting() method else push that into stack.



Next, we check for the vertex 3, its neighbor vertex is 4 again vertex 4’s neighbor is 5 and vertex 5’s child is 9. So now we push 9 to stack. Then after 4 followed by 5 and finally 3. Now we pushed vertex 1’s, all neighboring vertices except 6. Vertex 6, points to vertex 8 and this leads to vertex 11. As vertex 11 has no other child to explore, push that into stack. Then after it inserts vertex 8, comes backwards and start to fill stack with vertex 7 as well as 6. It’s time to insert vertex 1 as it’s all neighbors pushed back already in a stack. Vertices left are 10 and 12. These two vertices pushed into stack in ascending order. The final picture of stack will be shown as below with inserted values.

|  |
| --- |
| 12 |
| 10 |
| 1 |
| 6 |
| 7 |
| 8 |
| 11 |
| 3 |
| 4 |
| 5 |
| 9 |
| 2 |

The course order will be 12,10,1,6,7,8,11,3,4,5,9,2