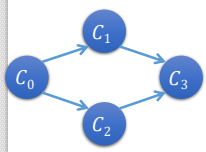
6.5 Activity Networks

6.5.1 Activity-on-Vertex (AOV) Networks

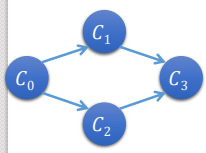
- A digraph G with the vertices represent tasks or activities and the edges represent precedence relations between tasks.



Predecessor :
Vertex i is a predecessor of vertex j, iff there is a directed path from vertex i to vertex j.

Topological Order

- A **linear ordering** of the vertices of a graph such that, for any two vertices i and j, if i is a predecessor of j in the network, then i precedes j in the linear ordering.



$C_0 \rightarrow C_1 \rightarrow C_2 \rightarrow C_3$ (O)
$C_0 \rightarrow C_2 \rightarrow C_1 \rightarrow C_3$ (O)
$C_0 \rightarrow C_2 \rightarrow C_3 \rightarrow C_1$ (X)

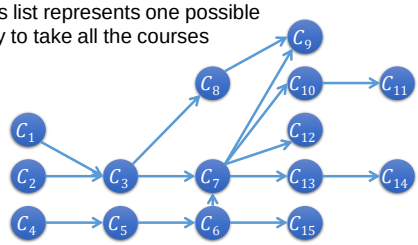
Application

Course No.	Course	Prerequisites
C1	Programming I	None
C2	Discrete Mathematics	None
C3	Data Structures	C1, C2
C4	Calculus I	None
C5	Calculus II	C4
C6	Linear Algebra	C5
C7	Analysis of Algorithms	C3, C6
C8	Assembly Language	C3
C9	Operating Systems	C7, C8
C10	Programming Languages	C7
C11	Compiler Design	C10
C12	Artificial Intelligence	C7
C13	Computational Theory	C7
C14	Parallel Algorithms	C13
C15	Numerical Analysis	C5

81

AOV Network of Courses

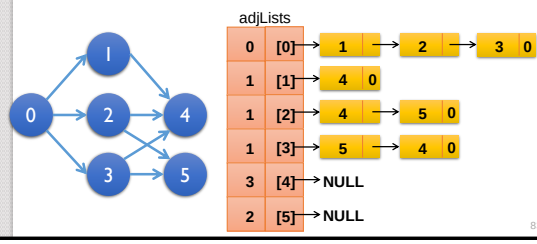
Use **topological ordering** to generate a linear order list. This list represents one possible way to take all the courses



82

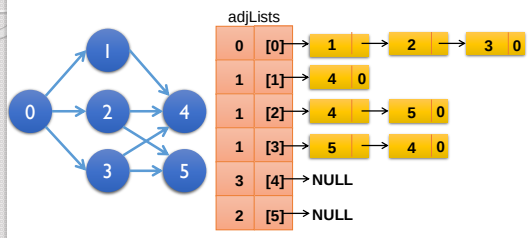
Topological Ordering

- Iteratively pick a vertex v that has no predecessors.
  - Use an additional field "count" to record the "in-degree" value of each vertex.



83

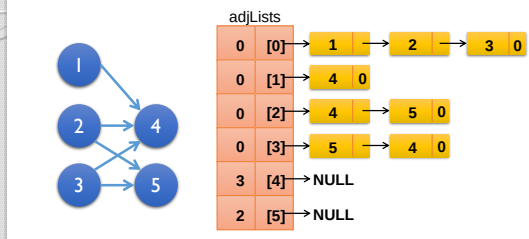
Running Example



Ordered list:

84

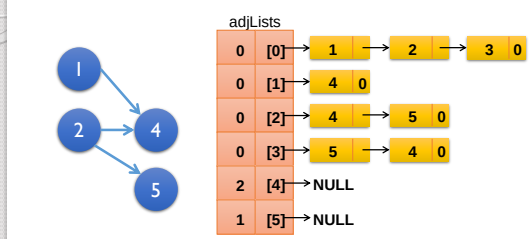
Running Example



Ordered list: 0

85

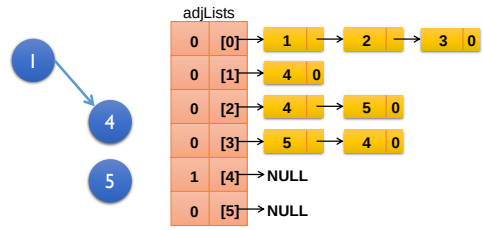
Running Example



Ordered list: 0 3

86

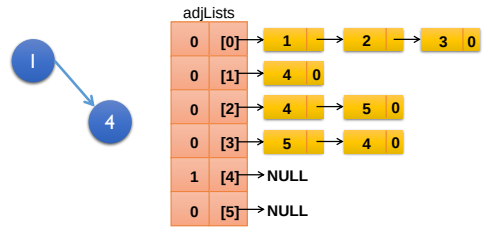
Running Example



Ordered list: 0 3 2

87

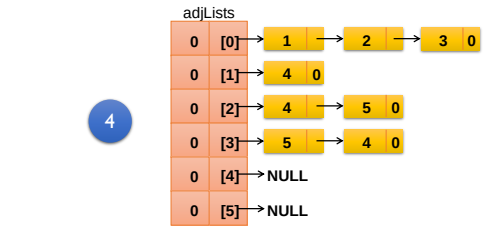
Running Example



Ordered list: 0 3 2 5

88

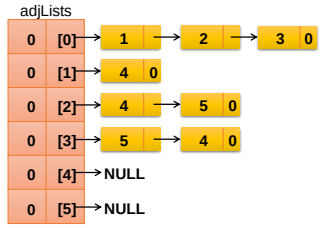
Running Example



Ordered list: 0 3 2 5 1

89

Running Example



Ordered list: 0 3 2 5 1 4

90

Self-Study Topics

- Graph representations
  - Sequential lists
  - Adjacency multilists
- Graph operation
  - Biconnected components
- Single source shortest path
  - **Bellman-Ford's algorithm** (Digraph with negative edge costs)
- Activity-on-Edge (AOE) Networks
  - Critical path analysis

91