Topological Sort

**Actual Output for graph.txt**

G1: 0 2 3 4 1

G2: 1 2 4 0 3

G3: 1 -> no more in-degree 0 vertex; not an acyclic graph.

G4: 0 1 2 4 3

G5: 1 4 0 3 2

G6: 1 4 0 3 2 8 9 5 6 7

G7: 7 -> no more in-degree 0 vertex; not an acyclic graph.

G8: 4 2 6 0 5 9 8 7 1 3

G9: 1 4 5 6 7 2 3 8 9 0

G10: 4 0 7 8 3 5 6 1 9 2

G11: 3 7 13 10 6 9 11 14 1 0 2 8 12 4 5

G12: 9 10 0 1 6 7 11 2 5 8 14 4 13 3 12

G13: 2 7 13 9 11 4 6 12 5 3 8 14 1 10 0

G14: 8 9 1 4 7 10 13 3 5 11 6 0 14 2 12

G15: 2 14 7 12 9 8 6 10 11 1 5 13 0 3 4

G16: 10 18 11 6 0 15 17 4 8 2 3 12 5 14 19 7 16 9 13 1

G17: 9 8 12 18 14 6 7 0 10 1 3 4 15 16 11 17 13 2 19 5

G18: 3 15 14 18 19 1 4 16 0 5 11 17 6 2 10 7 8 12 9 13

G19: No in-degree 0 vertex; not an acyclic graph.

G20: 14 4 6 13 5 10 8 18 7 12 0 3 16 1 17 9 2 19 15 11

G21: 13 1 12 21 22 4 7 2 18 15 17 19 14 23 16 24 0 3 5 10 9 8 11 6 20

G22: No in-degree 0 vertex; not an acyclic graph.

G23: 13 8 3 7 10 6 1 18 2 4 5 14 16 24 21 22 19 20 9 23 0 11 12 15 17

G24: No in-degree 0 vertex; not an acyclic graph.

G25: 15 0 5 14 7 4 12 17 11 20 3 13 16 8 6 23 22 10 2 19 21 1 9 24 18

G26: 7 14 25 10 6 22 15 19 1 0 17 18 20 27 9 11 26 8 21 12 3 28 5 13 29 2 16 24 23 4

G27: 7 12 21 20 25 11 23 27 28 1 4 16 8 18 14 19 5 24 15 22 26 2 6 29 0 17 9 13 3 10

G28: 0 13 3 1 29 8 15 11 16 6 17 19 4 18 26 20 28 27 23 5 21 22 14 24 10 7 12 9 25 2

G29: 23 25 3 17 2 5 9 4 29 18 11 1 13 6 15 10 20 8 14 16 24 21 7 19 28 0 12 22 26 27

G30: No in-degree 0 vertex; not an acyclic graph.

G31: 6 15 33 34 1 4 16 2 8 12 20 30 7 9 25 13 19 26 11 3 32 10 24 17 14 31 23 5 29 22 27 21 0 18 28

G32: No in-degree 0 vertex; not an acyclic graph.

G33: 0 18 7 29 25 33 6 3 34 10 19 17 20 28 11 21 32 8 5 22 1 23 27 15 4 26 30 9 12 24 13 31 14 2 16

G34: 15 7 13 5 4 19 27 14 0 23 28 25 21 6 11 31 12 3 34 20 1 9 8 29 26 16 10 32 33 2 18 22 17 30 24

G35: 26 25 19 11 16 8 13 14 22 6 31 32 0 21 10 5 24 3 30 18 28 34 15 29 33 27 9 23 7 1 4 12 2 17 20

G36: 4 17 19 18 10 0 24 30 12 26 25 8 7 31 27 21 36 22 23 3 15 9 11 20 37 28 1 14 16 38 5 33 29 2 34 39 6 13 35 32

G37: 3 10 16 33 12 34 13 15 11 21 1 30 29 4 5 37 17 20 32 2 39 6 35 14 25 27 38 18 24 28 36 8 22 26 7 9 19 31 0 23

G38: 39 11 17 5 9 1 35 8 23 34 12 26 18 15 28 31 36 10 24 27 7 2 13 21 30 0 38 6 16 29 37 32 22 14 20 33 4 19 25 3

G39: 13 20 0 16 6 12 7 8 37 36 18 26 31 32 29 11 17 21 34 39 3 30 1 23 24 25 5 4 33 28 19 10 38 2 35 27 14 22 9 15

G40: 4 20 39 11 14 24 9 8 17 18 29 34 31 33 16 22 36 6 5 21 23 3 27 13 25 30 38 2 32 12 1 35 10 15 28 37 19 0 26 7

G41: 21 8 39 0 23 37 11 12 13 33 1 28 15 34 18 38 19 25 2 5 41 3 44 16 14 20 35 22 31 4 36 26 32 9 43 24 27 42 7 30 10 6 17 29 40

G42: 39 2 21 16 0 14 37 7 41 15 19 11 12 20 22 40 25 31 27 13 30 8 23 36 42 3 5 32 1 26 4 29 34 44 18 28 38 35 9 17 43 33 10 24 6

G43: No in-degree 0 vertex; not an acyclic graph.

G44: 22 30 44 4 40 24 13 9 43 0 5 23 14 42 32 18 36 17 31 10 27 39 16 19 33 37 21 34 11 2 28 29 35 6 15 8 25 7 41 20 38 26 1 3 12

G45: 19 28 23 34 36 12 41 5 30 3 16 43 11 17 13 26 7 18 35 1 22 14 0 6 42 40 9 32 44 24 38 10 8 20 25 37 21 31 4 2 29 39 15 33 27

G46: 38 14 41 37 11 22 47 5 7 17 23 31 48 29 39 3 26 28 46 1 44 8 19 45 42 43 33 27 36 12 0 21 15 34 2 49 9 32 40 20 25 4 10 6 30 16 18 24 13 35

G47: No in-degree 0 vertex; not an acyclic graph.

G48: 31 38 1 24 16 12 27 35 10 18 2 32 26 40 6 37 19 48 9 20 0 22 33 13 36 4 47 15 39 49 46 45 29 7 3 23 11 21 5 14 17 44 43 28 30 42 25 41 8 34

G49: 4 5 33 21 30 37 24 8 23 22 15 3 6 32 9 14 1 18 7 11 2 29 34 0 28 19 38 42 47 27 10 39 36 12 49 20 31 16 48 35 46 41 43 17 25 13 26 44 45 40

G50: No in-degree 0 vertex; not an acyclic graph.

**Summary**

There are 3 helper methods in the program. They are addEdge, displayTopologicalOrder, topologicalSort. Purpose of the addEdge method is to add an edge between given 2 vertices. displayTopologicalOrder is used to display the final output (topological order) to the screen. displayTopologicalOrder method is used to determine the topological order of the graph.

To determine topological sort following algorithm is used.

* Determine a node that has no incoming edges.
* Include that node in the ordering.
* Take it out of the graph.
* Repeat the same process until, no more nodes have indegree zero.

If all the nodes are included in the ordering, the result has been already obtained. If there are some nodes left, but they are all connected by incoming edges, this indicates that the graph has a cycle and no topological ordering.

The graph is represented by an adjacency list. ArrayList of ArrayList is used for that. Another ArrayList is used to store the topological ordering of the graph. Integer Array is used to store number of incoming edges for each vertex. Vertices with no incoming edges are stored inside a queue.

Time complexity of the algorithm is O(V+E), where V = Vertices, E = Edges. Space complexity of the algorithm is O(V). To store the in-degrees of all the nodes, we must create an array. This required O(V) space.