REPORT

IT- 427

PROG ASSIGNMENT -1

ULID- [SPRAMK](mailto:SPRAMK@ILSTU.EDU)

SECRET DIRECTORY- “marvel”

The input file will be “graphs.txt”

Command- java topSort inputFilename.txt

The output generated is pasted below-

Topological Orders:

G1: 0 2 3 4 1

G2: 1 2 4 0 3

G3: 1 2 3 4 0

G4: 0 1 2 3 4

G5: 1 4 0 3 2

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

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

G8: 4 6 2 5 0 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 11 6 9 14 1 2 12 0 4 5 8

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

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

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

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

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

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

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

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

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

G23: 13 8 3 7 10 6 18 1 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 12 4 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 17 20 0 18 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 18 8 14 19 24 5 15 22 26 29 2 6 0 17 9 13 3 10

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

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

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

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

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

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

G47: No in-degree 0 vertex; not an acyclic graph.G48: 31 38 1 12 16 24 27 35 10 18 32 2 26 40 6 37 48 9 19 20 0 22 33 36 13 4 47 15 39 46 49 45 29 7 3 23 11 21 5 14 17 43 44 28 30 42 25 41 8 34G49: 4 5 21 30 33 37 24 8 22 23 15 3 6 9 32 14 18 1 7 11 29 34 0 2 28 38 19 42 47 27 10 39 36 12 49 20 31 16 48 35 46 41 43 17 25 26 44 45 13 40

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

Methods used and algorithm-

Main() method

In the main method, I have written the code for parsing the text file.

In the method I am taking arguments from command line

And then parsing the text file to generate the adjacency matrix

Once the nodes are read, we will call the **findTopSort()** method which takes some required parameters.

findTopSort() method-

In this I am having a while loop which checks if there is any node left having 0-in degree

Then based on the algorithm told in the class-

I am selecting a vertex with zero in-degree and adding to result

And updating the adjacency matrix and deleting the vertex from the vertices list

Once the while loop breaks , im printing the output

InDegree() method

I am using this method to check whether a vertex has any 0 indegree or not

If a vertex has atleast 1 in-degree then it returns false and vice versa.

CompleteSearch() method

This method used to check zero in-degree for all vertices.

* **Data structures-**

Arrays,

List

* **Time And Space Complexity:**

·       **Time Complexity:**

**1.    O(V):**Iterating over each vertex of graph for result

**2.    O(E):**Traversingover each edge for specific vertex to find indegree

**3.    O(E):**Traversing over each edge for vertex to removing its outdegree

**4.    Total Complexity = O(V+E)**

·       **Space Complexity:**

1. Auxiliary space required to store adjacency matrix is O(V\*V),

v = no of vertices.

2.    Auxiliary space to store result in List is O(V)

**3.    Total = O(V\*V) + O(V) = O(V^2)**