Topological Sort

Dicu Cristian Paul
CEN 1.1B (Dobby)

Year 2017- 2018
Semester II

Problem Statement

Topological sort. Implement two different algorithms to determine the minimun path between two vertexes ityn weighted directed graphs, e.g., Moore and Dijkstra

Application Design. For the first Method

The algorithm uses a number of functions that represent the process they are used for:

- Function for indegree.
- Function for queues, checking if its empty, inserting, and deleting queue.
- Function for creating a graph.

The input.

In the input is required to enter the number of vertices (example: 6), then the edges (example: 0 1). After writing all the edges that you want, you can type ”-1 -1” in order to stop the writing edges and show you the result.

The output.

In the output, if the edges are correct than the algorithm will show the vertices in topological order.
Example:

```
Number of vertices: 6
Enter edge : 0 1
Enter edge : 0 2
Enter edge : 0 3
Enter edge : 0 5
Enter edge : 1 2
Enter edge : 1 4
Enter edge : -1 -1
Vertices in topological order are : 0 1 3 5 2 4
```

How the algorithm works.

For this method I used indegree array .

The function insert queue verifies if the queue is initially empty, and also if the queue reaches the max, if it is it will print the message : Queue Overflow .

```
Data: The queue:
Result: Inserts if needed to queue:
if queue is full then
    print queue is full;
else
    if queue is initially empty then
        print add to queue;
    end
end
```

Algorithm 1: Inserting in queue.

The function create graph is where all the inputs are. If the origin or the destination is greater than n (which is the number of vertices) than it will show the message ”Invalid edge!”, but it will move on to the next input.

```
Data: The graph:
Result: The input, number of vertices, and the edges:
forall max edges do
    print origin and destination;
    if origin and destination == -1 then
        break;
    end
    if origin and destination greater than the number or vertices then
        print invalid edge;
    end
end
```

Algorithm 2: Creating the graph.

The function indegree calculates the indegree, which is the number of edges directed into a vertex in a directed graph.

```
Data: The graph
Result: Prints the indegree
forall every vertice in the graph do
    if adjacency matrix == 1 then
        integree ++;
    end
end
```

Algorithm 3: Finding the Indegree.

The main function is where I used the indegree, using a for to find the indegree of each vertex, and a while where it adds verteces (v) to the topological array (topo order), after that delete all the edges going from vertex (v).

```
Data: The graph
Result: Prints the vertices in topological order
forall find the integree of each vertex do
end
while the queue is not empty do
    add vertex to topological order array;
end
if count greater than the number of vertices then
    print no topological ordering possible;
end
forall vertices in topological order are do
    print all vertices in topological order;
end
```

Algorithm 4: Printing the board.

Application Design. For the secound Method.

The algorithm uses a number of functions that represent the process they are used for :

- Function for the topological sort.
- Function for initialize list, adjacent lists, and degrees.
- Function for entries adjacent lists.
- Function for creat adjacent lists.

The input

In the input is required to enter the number of vertices(example 4), then the edges (example vertex 5 has edges 4 and 3 : 5 4 3).

The output

It should enter the vertices in topological order.

How the program works

For this method I used a weird algorithm, which I am pretty sure its not the best.

I used a function called topological sort which is supposed to calculate the topological sort of the lists.

The function entries adjacent list is where the input takes place, using a while which verifies if the vertices are greater than the counter.