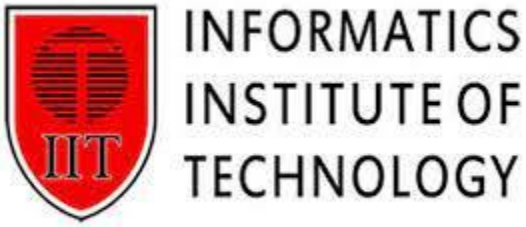


Informatics Institute of Technology

BEng. Software Engineering

Module: 5SENG003C

Algorithms: Theory, Design, and Implementation

Module Leader: Mr. Ragu Sivaraman

COURSE WORK REPORT

IIT ID: 20200688

UOW ID: w1913278

Name: Gorakapitiyage Induranga Kawishwara

A brief description of the data structure (Adjacency Matrix) that was utilized in the implementation.

Data structures are integral to many algorithms because of how effectively they organize and handle data. An adjacency matrix can be used to display a graph in a tabular format. The table's rows and columns are represented by 2D arrays, and each cell contains data. If a line can be drawn between two locations, that cell will contain a True value. We enter False if there is no link between them. It's much easier to maintain track of the graph's connections in this way.

A major benefit of using an adjacency matrix to describe a network is the ease with which we can determine whether a link exists between any two nodes. This is so because a single glance at a table cell is all that's required to determine whether a link exists. Because of how quickly adjacency matrices can be constructed, they are particularly useful for graphs with many edges. Adjacency matrices are useful for representing graphs, but which one is chosen depends on the nature of the graph and the operations that will be applied to it.

An Overview of the Implemented Algorithm (Sink Elimination)

If it's necessary to know if a directed graph is acyclic, the sink elimination procedure provides a quick and easy approach to it. The approach is effective because it iteratively removes sinks (vertices with no outgoing edges) from the graph. There will always be a sink that can be eliminated on each iteration if the graph is acyclic.

The elimination of a sink does not introduce any new cycles into the graph; hence this procedure can be repeated indefinitely. The graph is now empty; hence it may be safely assumed to be acyclic. However, we will not be able to delete all vertices from the graph if it contains a cycle, as there will be at least one vertex that is not a sink. If there remains at least one vertex left in the graph after the algorithm finishes, then the graph is cyclic.

The function's helper employs two Boolean arrays to monitor which vertices have been visited and which are in the call stack. The system use a list to maintain track of the route that has been taken thus far. A cycle is identified and the route from the beginning vertex of the cycle to its ending vertex is returned as an integer array if a visited vertex is found on the call stack. If the algorithm cannot find a cycle in the graph, it will return null.

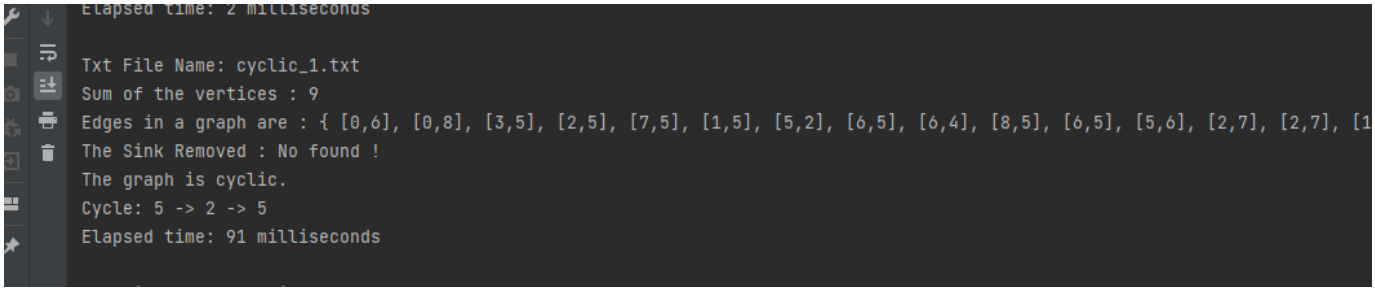
Tests of the Algorithm on Two Selected Case Studies.

Cyclic:

```
0 6
0 8
3 5
2 5
7 5
1 5
5 2
6 5
6 4
8 5
6 5
5 6
2 7
2 7
1 3
2 7
5 6
7 9
1 5
6 4
1 7
3 9
1 3
4 5
1 6
4 6
7 6
7 6
7 6
7 3
7 2
9 7
```

```
1 4
3 4
9 4
7 3
7 2
9 5
3 7
5 3
```

Graph edges

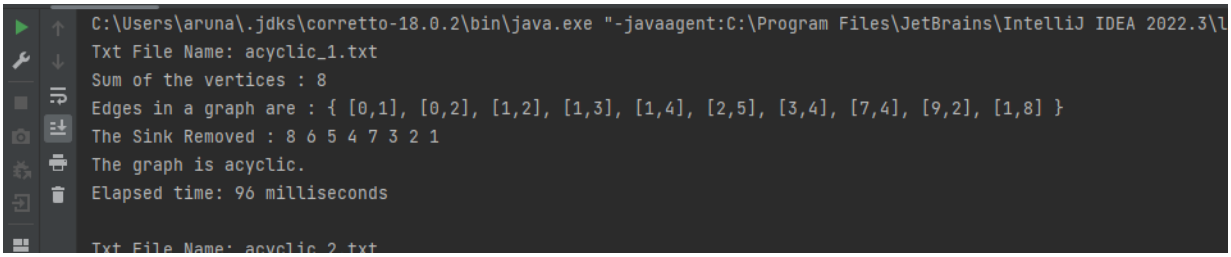


Output

Acyclic:

```
0 1
0 2
1 2
1 3
1 4
2 5
3 4
7 4
9 2
1 8
```

Graph edges


C:\Users\aruna\.jdks\corretto-18.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2022.3\l
Txt File Name: acyclic_1.txt
Sum of the vertices : 8
Edges in a graph are : { [0,1], [0,2], [1,2], [1,3], [1,4], [2,5], [3,4], [7,4], [9,2], [1,8] }
The Sink Removed : 8 6 5 4 7 3 2 1
The graph is acyclic.
Elapsed time: 96 milliseconds
Txt File Name: acyclic_2.txt

output

Evaluation of Algorithm Design and Implementation Performance

Where n is the size of the input array, the code's method has a time complexity of O(n2). This occurs because the outer loop only goes through the complete array once, whereas the inner loop goes through progressively smaller increments of the array with each iteration of the outer loop.

The doubling hypothesis may be used to do an empirical analysis of the algorithm's efficiency. This is done by timing how long the method needs to process arrays that are progressively larger, and then comparing the ratios of those durations. Theoretical research suggests that the procedure has a quadratic time complexity if the running time roughly doubles for each doubling of the input size.