Drew Meseck

10/27/2019

Homework #4

Part 1: Algorithm

The algorithm I used is as follows (pseudocode):

```
Is_Yucca(M):                    Where M is an Adjacency matrix for Graph M
    For(i = 0; I < M.Length; i++)        Where M.Length gives the length of any row or column
        Count_out_degree = 0             Tracks the count of out edges from a node
        In_Flag = True                   Tracks the logical truth of the In-degree = 0
        For(j = 0; j < M.Length; j++)
            If( M[j][i] == 1)
                In_flag = False          In degree > 0 (this row is not YUCCA determinant)
                Break out of this loop (go back and check the next column).
        If(In_Flag == True)              If In Degree = 0
            For(f = 0; f < M.Length; f++)
                If( M[i][f] == 1)
                    Increment Count_out_Degree
            If( Count_Out_Degree == M.Length – 1)
                Return True              Both Conditions for YUCCA have been Satisfied
    Return False                         None of the Conditions for YUCCA have been satisfied
```

Discussion of Algorithm:

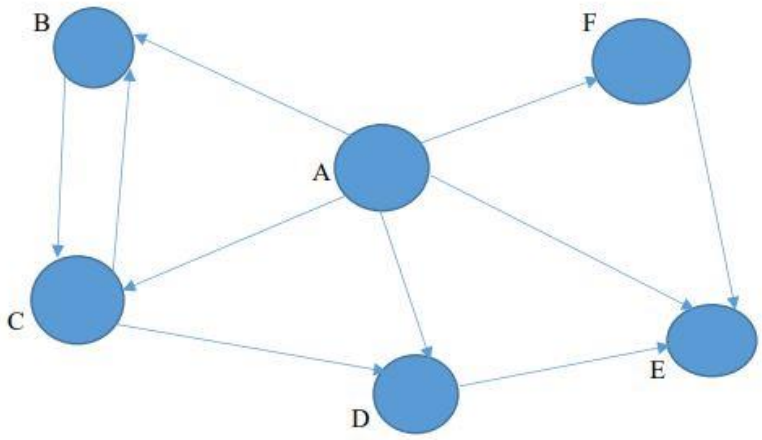
The Algorithm Above, worst case has 3 nested loops that all complete after *n* iterations making the worst-case time complexity for this algorithm $O(n^3)$. However, if the graph is not YUCCA, the algorithm can only have the worst-case time complexity of $O(n^2)$ since if the in-degree condition is not satisfied first, the tertiary for loop can never be reached. Only in rare cases (when the graph is YUCCA and the last node that is checked satisfies those conditions) will the algorithm run in its true worst-case complexity of $O(n^3)$.

Example:

TO

FROM	A	B	C	D	E	F
A	0	1	1	1	1	1
B	0	0	1	0	0	0
C	0	1	0	1	0	0
D	0	0	0	0	1	0
E	0	0	0	0	0	0
F	0	0	0	0	1	0

The above adjacency matrix denotes the directed graph as given by the problem:



The above algorithm would only end up having to check the first row and column, since node A is where the YUCCA conditions are satisfied. First it would check the in degree by looping through the A column to make sure there are no edges coming FROM another node TO node A. Since there is none, the In_Flag would never be set to false (no 1 is detected in that column). The next If statement would then trigger since the In_Flag is True. It would then loop through the A row and count the number of edges that go out from node A. Since it will count 5 such edges, and the condition it must meet for YUCCA is n – 1 edges, the algorithm would return true in this case.

Part 2: Program

The program itself is attached in the Program.cs file. The following pairs of images relate to an input graph and the programs output:

Valid Yucca on node B:

```
//                  A  B  C  D  E  F
int[,] matrix = { { 0, 0, 1, 1, 1, 1 }, //A
                  { 1, 0, 1, 1, 1, 1 }, //B
                  { 0, 0, 0, 1, 0, 0 }, //C
                  { 1, 0, 0, 0, 1, 0 }, //D
                  { 0, 0, 0, 0, 0, 0 }, //E
                  { 0, 0, 0, 0, 1, 0 }, //F
                };
bool result = Is_YUCCA(matrix);
```

Microsoft Visual Studio Debug Console

```
Is the Matrix Given a YUCCA? : True

C:\Program Files\dotnet\dotnet.exe (process 16188)
To automatically close the console when debugging s
le when debugging stops.
Press any key to close this window . . .
```

Invalid Yucca - B has an in edge:

```
//                  A  B  C  D  E  F
int[,] matrix = { { 0, 0, 1, 1, 1, 1 }, //A
                  { 1, 1, 1, 1, 1, 1 }, //B
                  { 0, 0, 0, 1, 0, 0 }, //C
                  { 1, 0, 0, 0, 1, 0 }, //D
                  { 0, 0, 0, 0, 0, 0 }, //E
                  { 0, 0, 0, 0, 1, 0 }, //F
                };
bool result = Is_YUCCA(matrix);
```

Microsoft Visual Studio Debug Console

```
Is the Matrix Given a YUCCA? : False

C:\Program Files\dotnet\dotnet.exe (proce
To automatically close the console when d
le when debugging stops.
Press any key to close this window . . .
```

Invalid Yucca – B has insufficient out edges:

```
//                  A  B  C  D  E  F
int[,] matrix = { { 0, 0, 1, 1, 1, 1 }, //A
                  { 0, 0, 1, 1, 1, 1 }, //B
                  { 0, 0, 0, 1, 0, 0 }, //C
                  { 1, 0, 0, 0, 1, 0 }, //D
                  { 0, 0, 0, 0, 0, 0 }, //E
                  { 0, 0, 0, 0, 1, 0 }, //F
                };
bool result = Is_YUCCA(matrix);
```

Microsoft Visual Studio Debug Console

```
Is the Matrix Given a YUCCA? : False

C:\Program Files\dotnet\dotnet.exe (proces
To automatically close the console when de
le when debugging stops.
Press any key to close this window . . .
```