Homework - 3

Chetna Ajay Nainani
Rachana Vimal Gandhi

1) Algorithm:->

```
bool containsSimpleCycle (Matrix graph[noOfVertices]
                                        [noOfVertices])
    for i = 0 to i != noOfVertices
        flag[noOfVertices] = {0}
        for j = 0 to j != noOfVertices
            if (j == i OR graph[i][j] == 0)
                Continue
            for k = 0 to k != noOfVertices
                if (k == j OR k == i OR
                        graph[j][k] == 0)
                    continue
                if (flag[k] == 1)
                    return true
                else
                    flag[k] = 1
    return false
```

Time Complexity:->

↳ If two vertices share more than one neighbour, then the graph contains a square

↳ for any vertices; v1 and v2, we will compare the row of v1 with row of v2 in O|V| time

↳ We will iterate all through V1 and V2 which will take O|V^2| time

↳ Thus, the overall time complexity of the algorithm is O|V^3| as O|V^2| for the iteration and O|V| for the Comparision.

Explamation :->

↳ We will use 3 loops and the flag array to check if the ~~graph~~ vertices contains more than one common neighbour

↳ We will iterate till index is not equal to number of vertices. We will ~~mark the ind~~ update the value of element in flag array as 1 to check if the row has common neighbour.

↳ first, we will create adjacency matrix and update the value of element as 1 if there is an edge from vertex 1 to vertex. The size of matrix will be no. of Vertices * no of Vertices.

↳ flag array is used to keep track of the neighbour vertices.
~~If the~~

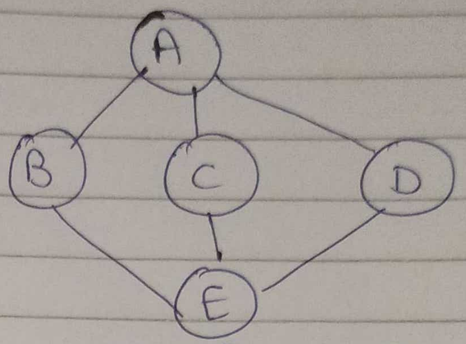
↳ We will Compare the row of vertex 1 and vertex 2 ; if there is more than one Common neighbour then we will return true else false.

↳ If ~~there~~ 1 is already present in the flag array and the value of another row is also 1, then ~~We check~~ that means both the vertices share ~~the~~ common neighbour and hence a square cycle is found.

↳ We will iterate first for loop till it does not becomes equal to noof Vertices. We will reset the flag array in every iteration of first for loop. We will iterate second for loop till it does not become equal to the noof Vertices. if index of first for loop & second for loop is equal, we will Continue the loop. We will use third loop for Comparision. The index of third for loop is used to update the value of flag array.

Example

for Graph:→



Adjacency Matrix:→

$$\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 1 & 1 & 1 & 0 \\ B & 1 & 0 & 0 & 0 & 1 \\ C & 1 & 0 & 0 & 0 & 1 \\ D & 1 & 0 & 0 & 0 & 1 \\ E & 0 & 1 & 1 & 1 & 0 \end{array}$$

By using Algorithm:→

Two Vertices , A & E Share more than neighbour and B,C,D also Share more than neighbour. So in flag array (Refer. the above algorithm) ,the flag[K] will be set to 1 for vertex A and for vertex E , we will get the flag of k^{th} index as 1 , the result will be returned as true.

∴ The Algorithm mentioned is correct.