Ramaneek Gill 1000005754
CSC263 problem set 6

```
BFS(matrix):
        num_of_nodes = 0;

        //count the num of unique nodes in the matrix
        for each row in matrix
                for each element in row
                        if element == 1 //it points to a vertex
                                look at that vertex at this position
                                if vertex.visted == 0
                                        vertex.visited = 1
                                        num_of_nodes++;

        //count how many times we can count the nodes adjacent from each connection point
        for each row in matrix
                for each element in row //row and element and indexs for the matrix
                                                                //ie A[row][element] == A[i][j]
                        Search_Coloumn(matrix, element)
                        if(count > num_of_nodes^2)
                                return True;
        return False;

Search_Coloumn(matrix, col_index)
        for i = 0
                if matrix[i][col_index] == 1
                        count++;
```

This algorithm relies on the fact in a matrix of size n we can visit a vertice at most n*n times. This occurs when they are all
connected to one another. This algorithm will go through each vertice in each row of the matrix and count up how many times
a node is visited by following what that node is connected to y looking at the array of its corresponding coloumn slice.

If there are cycles these nodes will be counted many times over because we are essentially counting the times we can reach all nodes
from each connection point. Since this connection point can be revisited when we look to see if the count is greater than the upper
bound n*n to determine a cycle.