# Coding Ninjas

## Problem Level: Easy

## Problem Description:

Given a NxM matrix containing Uppercase English Alphabets only. Your task is to tell if there is a path in the given matrix which makes the sentence “**CODINGNINJA**” .
There is a path from any cell to all its neighbouring cells. For a particular cell, neighbouring cells are those cells that share an edge or a corner with the cell.

**Sample Input 1:**

```
2 11
CXDXNXNXNXA
XOXIXGXIXJX
```

**Sample Output 1:**

```
1
```

## Approach to be followed:

Our goal is to find if the word exists in the matrix or not.The idea behind this is to search for the starting character and then use DFS and backtracking.  Traverse through the matrix starting from **C** and check for the next character in all the possible 8 directions. Maintain a visited array to ensure if each cell in the matrix is visited only once. Do it recursively for all the adjacent cells of each character of the string “CODINGNINJAS” by checking if the cell is visited or not, adjacent directions are valid to move. If we get the string return True otherwise False.

## Steps:

1. Create a helper function to check for the validity of movement in all the possible 8 directions.
2. Start from the first index of the string and search it in the matrix.
3. Recursively call on its adjacent cells by checking if the direction is valid and the cell is unvisited.

4. Do this until we reach the end of the string.
5. If all the characters of the string are connected return true else false.

## Pseudo Code:

```
dir = [[-1, -1], [-1, 0], [-1, 1], [0, -1], [0, 1], [1, -1], [1, 0], [1,
1]]

pattern = ['C', 'O', 'D', 'I', 'N', 'G', 'N', 'I', 'N', 'J', 'A']

function validPosition(x, y, n, m) :

     If (x >= 0 and x < n and y >= 0 and y < m)

         Return true

     Return false


Function DFS(arr, visited, x, y, index, n, m) :

    if (index equal to 11) : //last index of string "CODINGNINJAS"

        return 1

   visited[x][y] = True  // maintain a visited array

    found = 0

   Loop from i = 0 till i=8 ://all possible directions

        newx = x + dir[i][0]

        newy = y + dir[i][1]

        if(validPosition(newx, newy, n, m) and arr[newx][newy] equals
pattern[index] and visited[newx][newy] is False) :

            found = found | DFS(arr, visited, newx, newy, index + 1, n,
m)


    visited[x][y] = False

    return found

Function solve(arr, n, m) :


    found=0

    Declare visited

    Loop from i=0 till n
```

```
        Loop from j=0 till m
            Visited[i][j] = False


   Loop from i=0 till n
        Loop from j=0 till m
            if (arr[i][j] is 'C') :
                  found = DFS(arr, visited, i, j, 1, n, m)
                if (found is not equal to 0) :
                    break


        if (found is not equal to 0) :
            break
    return found
```

**Time Complexity: O(N*M)** where N is the number of rows and M is the number of columns of the given matrix.