**GRAPH THEORY AND IT’S APPLICATIONS**

**ASSIGNMENT – 3**

NAME : Sanmat Sanjayakumar Payagoudar

SRN : PES1UG20CS385

**PROBLEM STATEMENT :**

C) Find the solution to 9X9 sudoku problem(with/without initial hints).

**ALGORITHM EXPLANATION :**

* Create a function that checks after assigning the current index the grid becomes unsafe or not. Keep Hashmap for a row, column and boxes. If any number has a frequency greater than 1 in the hashMap return false else return true; hashMap can be avoided by using loops.
* Create a recursive function that takes a grid.
* Check for any unassigned location.
  + If present then assigns a number from 1 to 9.
  + Check if assigning the number to current index makes the grid unsafe or not.
  + If safe then recursively call the function for all safe cases from 0 to 9.
  + If any recursive call returns true, end the loop and return true. If no recursive call returns true then return false.
* If there is no unassigned location then return true.

**CODE :**

N = 9

def print\_solution(array):

    for i in range(N):

        for j in range(N):

            print(array[i][j], end = " ")

        print()

def is\_safe(input\_grid, Row, Column, num):

    for x in range(9):

        if input\_grid[Row][x] == num:

            return False

    for x in range(9):

        if input\_grid[x][Column] == num:

            return False

    startRow = Row - Row % 3

    startColumn = Column - Column % 3

    for i in range(3):

        for j in range(3):

            if input\_grid[i + startRow][j + startColumn] == num:

                return False

    return True

def solve\_sudoku(input\_grid, Row, Column):

    if (Row == N - 1 and Column == N):

        return True

    if Column == N:

        Row += 1

        Column = 0

    if input\_grid[Row][Column] > 0:

        return solve\_sudoku(input\_grid, Row, Column + 1)

    for num in range(1, N + 1, 1):

        if is\_safe(input\_grid, Row, Column, num):

            input\_grid[Row][Column] = num

            if solve\_sudoku(input\_grid, Row, Column + 1):

                return True

        input\_grid[Row][Column] = 0

    return False

input\_grid = [[5, 3, 0, 0, 7, 0, 0, 0, 0],

        [6, 0, 0, 1, 9, 5, 0, 0, 0],

        [0, 9, 8, 0, 0, 0, 0, 6, 0],

        [8, 0, 0, 0, 6, 0, 0, 0, 3],

        [4, 0, 0, 8, 0, 3, 0, 0, 1],

        [7, 0, 0, 0, 2, 0, 0, 0, 6],

        [0, 6, 0, 0, 0, 0, 2, 8, 0],

        [0, 0, 0, 4, 1, 9, 0, 0, 5],

        [0, 0, 0, 0, 8, 0, 0, 7, 9]]

if (solve\_sudoku(input\_grid, 0, 0)):

    print\_solution(input\_grid)

else:

    print("No solution exists")