**Analysis and Design of Algorithms**

**Semester III**, Year **2021-22**

**Lab - 5**  Date : 01-11-2021

Name: E. Sai Manoj MIS. No: 112015044 Branch: CSE

**AIM:**

1. Implement a backtracking algorithm for solving N Queen problem. Compute all possible solution for N Queen and also compute the number of backtracks. Perform the experiment from N = 2 to 9.

2. Given a set of candidate numbers (candidates) (without duplicates) and a target number (target), find all unique combinations in candidates where the candidate numbers sums to target. eg. W : [5, 10, 12, 13, 15, 18] and target = 30.

**Question 1:**

**Pseudo Code:**

START

inSafe(sol,row,col,n) // checks if the position has any Queens in the same column, and the same diagonals

For i = 0 to col

If sol[row][i] == ‘Q’ //checking if the column has a Queen

Return False

For i = row to 0 and j = col to 0

If sol[i][j] == ‘Q’ //checking if one of the diagonal has a Queen

i = i-1, j = j-1

Return False

For i = row to n and j = col to 0

If sol[i][j] == ‘Q’ //checking if the other diagonal has a Queen

i = i+1, j = j-1

Return False

Return True // If the position is safe we return true

QInsert(sol,col,n) // We insert the queens in the rows, one in each column

If col == n // All the queens are placed add it to the solution list and return true

Solx = []

For i in sol

For j = 0 to len(i)

If i[j] == ‘Q’

solx.append(j+1)

j = j+1

i = i+1

Posns.append(solx) //adding all the solutions to the final solution list

Return true

q = false

For i = 0 to n // In that column we keep insertion Queen in each row and check

If inSafe(sol,i,col,n)

Sol[i][col] = ‘Q’

q = QInsert(sol, col+1,n) or q

Sol[i][col] = ‘0’ //if the insertion of the queen doesn't give us a solution we backtrack and remove Queen from that position

bt = bt + 1 // backtracking counter

Return res

solve(n)

posns.clear() // we clear it so that when we go for next n value we have a clear solutions list

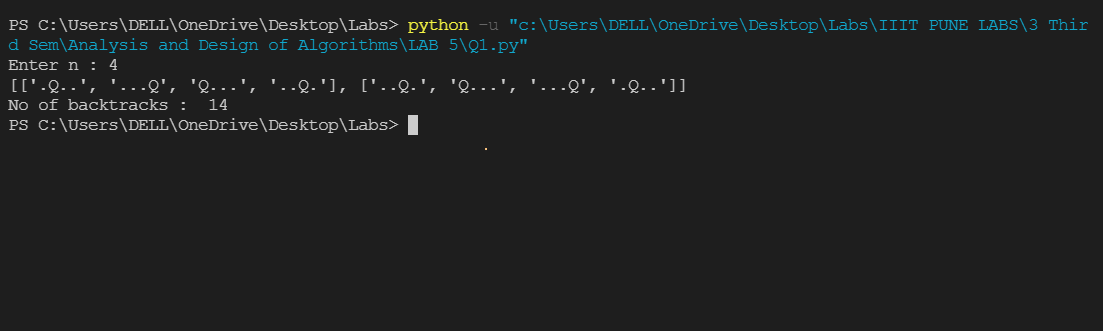
sol = empty board(all positions ‘0’)

QInsert(sol, 0 ,n) // starting from the first column

Return posns //final list of all the solution lists

END

**Output:**

****

**Question 2:**

**Pseudo Code:**

START

FUNCTION combinationSum(List, target):

result = [ ] # empty list to store output of code

tempList = [ ]

List = sorted(list(set(list)) # sort the list

Call the function findNumbers(result, List, tempList, target, 0)

RETURN result

END FUNCTION

FUNCTION findNumbers(result, List, tempList, target, index):

IF target == 0:

Append tempList to result # result.append(list(tempList))

RETURN

ENDIF

FOR i = index till i = len(tempList):

IF target - List[i] >= 0:

Append List[i] to tempList # tempList.append(List[i])

Call function findNumbers(result, List, tempList, target – List[i], i)

Remove List[i] from tempList # tempList.remove(List[i])

END FOR

END FUNCTION

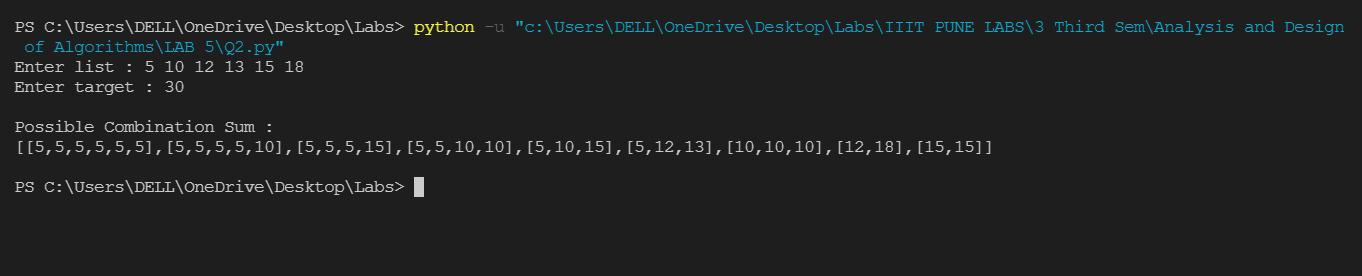
List = input List

Target = Input target

Call the function combinationSum(List, target)

END

**Output:**

****