CS-430

HW-5

① Let w be given maximum weight

Algorithm:

```
TSP (i, v')
  {
   if |v'| = 0 then return c(i,1)
    else
        result ← ∞
        for (k = 1 to |v'|) do
         { cost = TSP(v_{ik}, V' - {v_{ik}}) + C(i,k)
            if (cost < w)
               result = min (result, cost)
           }
        return result
  }
```

② V - value for which we need to check if change exists.

$d_1, d_2 \cdots d_n$ → list of denominations.

let d[ ] be an array that contains $d_1, d_2 \cdots d_n$.

To find out if change exists for value V, we can check if

1) change exist by including a coin d_i
(or)
2) change exists by excluding a coin d_i.

That is

IsChangeExist (V, n, d[ ]) =

IsChangeExists (V - d[n], n, d[ ])
(or)
IsChangeExists (V, n-1, d[ ])

Algorithm :

```
IxChargeExists (V, n, d[ ])
  {
    if (V = = 0)
        return true
    else if (V<0)
        return false
    else
        return IxchargeExist (V - d[n], n, d[ ])
                   (or)
        return IxchargeExist (V, n-1, d [ ] )

  }
```

In the above recursive solution, multiple sub problems are calculated many times.

To avoid this, create a table of size (V+1), n

t [ ] [ ] = new boolean [V+1] [n]

```
/* Initialize all values to false initially */
for(i = 0, i < V+1 ; i++)
  {
    for (j = 0 ; j < n ; j++)
        t[i][j] = false .
  } }
```

Scanned by CamScanner

```
/* For v = 0, set to true */

for (j = 0 ; j<n ; j++ )
    t [0][j] = true.

/* Build table bottom up */

for (i = 1 ; i< v+1 ; i++ )
   for (j = 0 ; j<n ; j++)
     {
      if (i - d[j] >= 0)
          x = t [i - d[j]][j]
      else
          x = false
      if (j>1)
          y = t [i][j -1]
      else
          false .
      t [i][j] = x or y
     }
}
return t [v][n-1]
```

③ The maximum cheese the mouse can gain at any point (i,j) in the grid will be equal to the weight of cheese at (i,j) + the maximum of cheese it can get by either going down (or) by going right.

a)

$$W(i,j) = W(i,j) + \max\begin{cases} W(i+1,j), \\ W(i,j+1) \end{cases}$$

where $i,j \in (0,n)$
and $w(i,n+1) = 0$ and $w(n+1,j) = 0$

b) let $t[\,][\,]$ be a $n \times n$ array which holds the maximum cheese that the mouse can get starting from the point (i,j) where i and j ranges from 1 to n.

i) When the mouse is in the last column, it can only get the cheese by going down.

ii) When the mouse is in the last row, it can only get the cheese by going right

iii) For any other position, maximum cheese = the maximum cheese it can get by going down or by going right

Tabulating

Filling the table in bottom-up manner.
Assuming the array indices start from 1 for convenience.
Algorithm :

```
 t [n][n] = w(n, n)
for j = n-1 to 1      /* for last row */
   t [n][j] = w(n, j) + t [n][j+1]

for i = n-1 to 1      /* for last column */
   t [i][n] = w(i, n) + t [i+1][n]

for i = n-1 to 1      /* for other values bottom
   for j = n-1 to 1                         up */
      {
         t [i][j] = w(i, j) + max[ t [i+1][j],
                                    t [i][j+1] ]
      }
 return t [1][1]
 ~~return t[0][0]~~
```