CSCI B505 - Fall 2018

Written Assignment - 3

1. A binomial coefficient $C(n,k)$ can be defined as the coefficient of x^k in the expansion of $(1+x)^n$.

We need to solve this problem with dynamic approach where we will save the value of already calculated coefficients in an array and trace back that value and use it when we need that value.

The algorithm is -

```
binomial_coefficient(n,k)
    For x in range(0,k+1) and x in range(0,n+1).
Take C = 0
# Bottom up approach of dynamic program.
for i in range(n+1):
    for j in range(min(i,k)+1):   # min will take the lower value as we don't need to calculate the values of k where k is greater than min(k).
        # Base cases.
        if j==0 or j==1:
            C[i][j] = 1
        else:
            C[i][j] = C[i-1][j-1] + C[i-1][j]
    return C[n][k]
```

2. Paving a road of n meters with stones of 2, 3, 5 meters long.

Ans: Suppose we choose the 1st tile as 2 or 3 or 5. Then the remaining road to be paved is n-2 or n-3 or n-5. We need to repeat this process until we get n=0.

We can write an algorithm as.

```
Paving road (n)
 if (n==0):
     return 1.        # There is no more way (if any) to pave the remaining road,
 elif (n==1):           else if n=0 then there is 1 way by not paving any stone on the road
     return 0
 elif (n==3 or n==2):
     return 1
 else:
     return paving road (n-5) + paving road (n-3)
          + paving road (n-2)
```

We can use dynamic programming to do the same much more efficiently.

```
Paving road (n)
res = [0] * (n+1)
  res [0] = 1
  res[1] = 0
  res[2] = 1
  res[3] = 1
  res[4] = 1
for i in range (5, n+1):
  res[i] = res[i-2] + res[i-3] + res[i-5]
 return res[n]
```

3. Algorithm to find number of ways to partition n numbers.

Ans 5. Ans:- number_of_partition (n)   # We can take a function which takes n as input and returns us the number of partition on that number

for i in range(n+1) :
 for j in range (n+1):
n_partition =[0]* n   ——> to initialize our array with 0's.

<u>base cases</u>:

n_partition [0] [0] = 1

for i in range(1, n+1): #(fill the matrix with the values)

n_partition [i] [0] = n_partition[i-1] [i-1]
base case for j=0 filling it manually

for j in range (1, i+1);   # for the other values of j

n_partition [i] [j] = n_partition[i-1] [j-1]
+ n_partition [i] [j-1]

return n_partition [n] [0]

4. Grid pathway problem to find number of shortest path in n x m size of the grid.

Ans:

Grid (n, m):- #(We need to give input n and m value as input)

let's ~~row~~ row = 0
Column = 0

(grid [0] [0] is ~~the~~ the left most bottom most value where we start)

if (row = n-1 and column == m): (if it can reach goal state in 1 step)
return 1

elif (row == n) and column < m : #(edge cases where ~~it~~ it already achieved the row move and can move columnwise only)
return grid (row, column +1)

elif (row < n and column == m): #(only row move possible)
return grid (row +1 , column)

else : #(if both of them are less i.e. row < n and Column < m)
return (grid (~~row~~, column+1)
+ grid (row+1 , column))

Scanned by CamScanner