**Problem :**

**​​**Find count of numbers in range [L,R] which has the sum of its digit = x.

Constraints :

0<=L<=R<=10^18

x<=180

**Approach :**

-> We find **count(r,sum) - count(l-1,sum) ;**

where count(x,sum) means count of numbers from 0 to x where sum of its digits=”sum”

-> Now we find out how many digits the highest allowed digit has, eg if the number is 4937 , which has 4 digits then we try to form all 4 digit numbers which form required sum means

0001

0002

0003

..

4937 , and see which has required sum.

-> If we imagine the problem with all 9s at all 4 places like 9999 , then problem is easy,

At every place we can put digit from **i=0 to 9, and recur to check how many ways are there to form “requiredSum-i” using the rest 3 digits .**

->For this simpler problem : the highest sum that can be asked in question is 180 as per constraints, and at most there can be a 19 digit number (10^18), so for all 19 digits at most it can be called on to check for sum upto 180.

-> So total 180\*19 states at max, and transition time for each of these is 10 as it puts a digit from 0 to 9 at each place to transition to some other state,

-> So max time = 180\*19\*9 ~ 10^5 which pass in 1 sec.

-> So dp[n][sum] = dp[n-1][sum-i] , for i=0….9 , where dp is no of ways to get “sum” using “n” digits.

**-> For our question , we need 1 more condition:**

**If number is 4537 , at 1st place, we are BOUNDED & can put only 1...4 , if we put anything from 1...3 , we are not further bounded and will never be bounded,as we know we can never exceed the required upper bound,so we can put any number from 0...9 at rest of the places.**

-> **But if we put a 4 , this BOUND carries on, means on 2nd place we can put only digits from 0 to 5 .So now in 2nd place if we put any of 0...4 we are not further bounded but putting a 5 carries on the Bound and so on,....**

-> So just maintaining a BOUND boolean variable at every instant in the recursion will do the job.

-> **But apart from that another imp thing is that we need 1 more dimension to our DP table,**

**2 possibilities :**

**dp[n][sum][0] and dp[n][sum][1] ,**

The 3rd dimension = 0, means no of ways to form a sum of “sum”,with n digits given that we were not bounded at that place, and 1 means we were bounded when we returned the answer.

-> So again the Time remains same : It's just that some place might be called once with bound & once without bound , so 2 times at max which makes time = 2\*10^5 which is almost same.

**Code :** [**https://ideone.com/eNX6nL**](https://ideone.com/eNX6nL)