GUYS REVIEW JAROOR DENA PADH KAR :P

A Brief Tut On Digit DP (WORKCHEF JULY 16 isi pe tha)

I will go by example:-

question
How many numbers are less than or equal to N and has sume of digit equal to K.

So what To do:-
Brute force approach suggest us to iterate over all numbers less than N and find sum of digits of each number.
If we iterate from 0 to N and find sum of digit of each number complexity would be O(NlogN) but some question has constraints like 10^18

So now what :-

we will use digit dp to solve this

Lets take N=12345 and K=60

now we have a number of 5 digits and we want to calculate how many number less than that 5-digit number has sum of digit=60.

so first we make an array "Y" and store all the digits of the given number in that array for N=12345 the array would be
Y[0]=1,Y[1]=2,Y[2]=3 and so on

now we start making Digits using recursion first we will take a digit and then keep appending digits to it until we get a number of 5-digits
for example:-
0
01
010
0102
01023

code for it:-

```
void generate (int pos , int num)
{
    if(pos==5)
    {
        cout<<num<<endl;
        return 0;
    }
```

```
    for (int i=0 ; i<=9 ; i++)
    {
        int new_num=num*10+i;
        generate(pos+1,new_num);
    }
    return;
}
```

so this way we can generate all numbers which has <=5 digits but we don't need all of them we need only numbers <=12345 so in our function we will define a new boolian variable say check.

Now if the digit which was last appended to our number is equal to the digit of N at the same position then the check wouldd be 1.
for example
1 //as 1 is appended at 1st position also in 12345 1st digit is 1 so check will be 1

But in the below case check wouldn't be 1:-
1
10
103 //even 3 is equal to the digit on 3rd position in 12345 but 103 <123 and we know this as check at 10 is 0.

what will this do ??
see if our check will be 1 then we will append digits from) to digit which is at same position in 12345 like

1 //check is now 1
here we will append 0,1,2 in 1 only which gives 10,11,12 and so on

This way we are able to generate all those numbers which are less than equal to N;

Code

```
void generate (int pos , int num , bool check)
{
    if(pos==5)
    {
        cout<<num<<endl;
        return 0;
    }

    int till=9;
    if(check)
        till=Y[pos];
```

```
    for (int i=0 ; i<=till; i++)
    {
        int new_num=num*10+i;
        bool bcheck=0;
        if(i==Y[pos] && check)
            bcheck=1;
        generate(pos+1,new_num,bcheck);
    }
    return;
}
```

the complexity of above recursive function is ON)

but we can see that its not necessary to generate all these numbers instead we can get sum of digits by adding a new variable sumofdig and as we don't need exact numbers we can leave the variable num.
for example:-

num	sumofdigit
1	1
12	1+2=3
123	3+3=6
1234	6+4=10
12345	10+5=15

After appending N digits we can check whether sumofdigit==k or not
Code:-

```
int solve(int pos , int sumofdig, bool check)
{
    if(pos==N+1)
    {
        if(sumofdigit==K)
            return 1;
        return 0;
    }

    int till=9;
    if(check)
        till=Y[pos];

    int num=0;
    for (int i=0 ; i<=till; i++)
    {
        int sumo=sumofdigit+i;
        bool bcheck=0;
        if(i==Y[pos] && check)
```

```
            bcheck=1;
        num+=generate(pos+1,sumo,bcheck);
    }
    return num;
}
```

compexity of above code :-O(N)

now we can easily add memoization to this function and make the complexity O(K*2) as sum of digit for an 18 digit number can't exceed 18*9 so K will always be less than 18*9 and hence complexity would be O(324)

```
int solve(int pos , int sumofdig, bool check)
{
    if(pos==N+1)
    {
        if(sumofdigit==K)
            return 1;
        return 0;
    }

    if(DP[sumofdigit][check]!=-1)
        return DP[sumofdigit][check];

    int till=9;
    if(check)
        till=Y[pos];

    int num=0;
    for (int i=0 ; i<=till; i++)
    {
        int sumo=sumofdigit+i;
        bool bcheck=0;
        if(i==Y[pos] && check)
            bcheck=1;
        num+=generate(pos+1,sumo,bcheck);
    }
    DP[sumofdigit][check]=num;
    return num;
}
```

complexity of above code is O(K*2) as the DP[K][2] will be filled and if recursion reaches a unique (sumofdigit ,check) more than once then DP table will return the value so overall complexity will be no. of iteraion to fill the table which is O(K*2)

Hope You Understand this a very nice concept

PS:- Workchef had some logic part too apart from Digit DP . It had Bitmasking involved in it too will write about those concepts some other time :)