is\_fancy\_number(num):

* Creates a array of digits(digits) in square of num and pass it to rec() function for finding whether num is fancy or not

Logic :

* If num is fancy then there must exist a way to partition the array digits such that the splitted sum=num
* How did I achieve going through all possible partitions ?
* I used the technique of recursion to do that.
* Since each value in the digits array can be either the starting of a new splitted number or either it will combine with the previous built number making it more larger. Hence , there are 2 possibilities for each value in the digits array.
* I handled those recursive calls in the rec() function.
* In this way all possibilities would be covered.
* Finally when no more value in digits array is left we check if sum1+sum2==num or not. And return accordingly.
* Then there is optimization in the rec function as well . If sum1+sum2 > val then it cannot be a fancy number using that split . So, I return false without going through all further possibilities.
* Then there is an optimization in the generate\_result function : If an answer for num or less is already calculated then I don't calculate it again. I have stored it in an array and returned the answer using that array.

cnt = []

*def* rec(*digits*, *pos*, *sum1*, *sum2*,*val*):

'''

rec :checks digits can be split into several numbers such that the sum of these is equal to the original number.

digits array contains the digits of the square of the number

pos is the current position in the digits array

sum1 is the sum of the numbers we have added by splitting the digits array

sum2 is the number we are currently building

val is the original number's square

'''

if *sum1* + *sum2* > *val*: # not fancy

return False

if *pos* == len(*digits*): # end of digits

if *sum1* + *sum2* == *val*: # if fancy

return True

return False

# if we add the current digit to sum2 without updating total sum

# if we make sum2=digit and update total sum1 = sum1 + sum2

return rec(*digits*, *pos*+1, *sum1*+*sum2*, *digits*[*pos*],*val*) or rec(*digits*, *pos*+1, *sum1*, *sum2*\*10 + *digits*[*pos*],*val*)

*def* is\_fancy\_number(*num*):

'''

is\_fancy\_number : checks if the number is fancy

'''

square = *num* \*\* 2

digits = list(map(int, str(square))) # convert to array of digits

return rec(digits, 0, 0, 0,*num*) # call recursive function

*def* generate\_result(*x*):

'''

Your logic goes here

'''

if *x*<=len(cnt):

return cnt[*x*-1]

count = 0

if len(cnt) !=0:

count = cnt[-1]

for num in range(len(cnt)+1, *x*+1):

if is\_fancy\_number(num):

count += 1

cnt.append(count)

return count