Qn Link : <https://docs.google.com/document/d/1taIf64o4VlUOsPXqt8wG--OpsOSo-xyWE0MupwLf4Ck/edit>

Summary :

* Given an array of size “N” , find the sum of all pairs such that fun(I , j) = ij
* Find the sum of these pairs
* Func(a[1] , a[2] ) + Func(a[1] , a[3] )+……..+ Func(a[1] , a[n] )+ Func(a[2] , a[3] ) + …….. + Func(a[2] , a[n] ) + …………+ Func(a[n - 1] , a[n] )

Brute force :

// Function to concatenate integers i and j and convert to an integer

public static int f(int i, int j) {

String s1 = Integer.toString(i);

String s5 = Integer.toString(j);

String l = s1 + s5;

int r = Integer.parseInt(l);

return r;

}

int sum = 0;

for (int i = 1; i <= n; ++i) {

for (int j = i + 1; j <= n; ++j) {

sum += f(b[i], b[j]);

}

}

Optimized Approach :

* [1 , 2 , 500 , 800 , 20 , 50]

For index 0 & 5 🡪 func(1, 5) = 150

For index 2 & 5 🡪 func(2 , 5) = 50050

For index 3 & 5 🡪 func(3 , 5) = 80050

For index 4 & 5 🡪 func(4 , 5) = 2050

I can split it like ,

* 100 + 50
* 50000 + 50
* 80000 + 50
* 2000 + 50

So for index i & j , the func (I , j ) returns 🡪 nums[i] \* 10 ^ y + nums[j] , where y is length of nums[j]

1 \* 10 ^ 2 + 50

500 \* 10 ^ 2 + 50

800 \* 10 ^ 2 + 50

….

And the sum of prev index after calculation is = 132100 + 4 \*50

The sum of prev index before calc is = 1321 + 4 \* 50

That is it , create an prefix array and the formula is

Sum = sum + prefix[j – 1 ] \* 10 ^ y + (j – 1) \* nums[j]

class Solution {

    public int solution(int [] nums , int X , int Y) {

        int n =  nums.length;

        int sum = nums[0]

        int [] prefix = new int[n];

        perfix[0] = nums[0];

        for(int i = 1 ; i < n ; i++){

           prefix[i] = prefix[i - 1] + nums[i];

        }

        for(int i = 1 ; i < n ; i++){

             int length = Integer.toString(nums[i]).length();

       sum = (sum + (prefix[i - 1] \* (long)Math.pow(10 , length) ) % 1e7 + (nums[i] \* j-1) % 1e7)%1e7;

        }

        return sum;

    }

}