Question :

Geek is participating in a coding contest. His score, N, can only consist of digits 7 and 9. He has the power to update his score by changing at most one digit. Help him find the maximum score.  
**Note:** You can only convert 7 to 9, and 9 to 7.

Input : 9779

Output : 9979

Answer :

class Solution {  
    public int maximumNumber(int num) {  
        int a[] = new int[1000];  
        int k=0;  
        while(num!=0){  
            a[k++]=num%10;  
            num/=10;  
        }int max=0;  
        for(int i=k-1;i>=0;i--){  
            int sum=0;

            for(int j=k-1;j>=0;j--){  
               if(i==j && a[i]==7){  
                    sum=sum\*10+9;  
            
                }  
                else sum=sum\*10+a[j];  
            }if(sum>max) max=sum;  
        }return max;  
    }  
}

Question 2 :

Geek is given an array **nums** of length **n**and two integers **x**and **y**. Geek is interested to find the total number of pairs (i, j) such that x <= a[i]\*a[j] <= y (1 <= i < j <= n).  
Help geek to find the total number of such pairs.

**Input:** nums[] = {5,3,7,9,7,9,7,7},

x = 7, y = 19

**Output:** 1

**Explanation:** There is only one pair which

satisfies the given conditions. The pair is

(1,2).

Answer :

class Solution  
{  
    public int TotalPairs(int[] nums, int x, int y)  
    {  
        int count = 0;  
    int n = nums.length;  
    Arrays.sort(nums);  
    for (int i = 0; i < n; i++) {         
        int left = i + 1;  
        int j = n - 1;  
        while (left <= j) {  
            int mid = left + (j - left) / 2;  
            if (nums[i] \* nums[mid] <= y) {  
                left = mid + 1;  
            } else {  
                j = mid - 1;  
            }  
        }       
        int right = i + 1;  
        int k = n - 1;  
        while (right <= k) {  
            int mid = right + (k - right) / 2;  
            if (nums[i] \* nums[mid] >= x) {  
                k = mid - 1;  
            } else {  
                right = mid + 1;  
            }  
        }  
        count += Math.max(0, j - k);  
    }  
    return count;  
    }  
}