1) First Repeating Character - 1

description

Given a string of characters, find the first repeating character.

**Input Format**

The first line of input contains T - the number of test cases. It's followed by T lines, each line contains a single string of characters.

**Output Format**

For each test case, print the first repeating character, separated by a new line. If there are none, print '.'.

**Constraints**

1 <= T <= 1000

'a' <= str[i] <= 'z'

1 <= len(str) <= 104

**Example**

**Input**

4

datastructures

algorithms

smartinterviews

hackerrank

**Output**

a

.

s

a

import java.io.\*;

import java.util.\*;

public class Main {

    public static void main(String[] args) {

        /\* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class should be named Main. \*/

        Scanner sc=new Scanner(System.in);

        int n=sc.nextInt();

        for(int i=0;i<n;i++){

            char a='.';

            String s=sc.next();

            HashMap<Character,Integer> h=new HashMap<>();

            for(int j=0;j<s.length();j++){

                h.put(s.charAt(j),h.getOrDefault(s.charAt(j),0)+1);

            }

            for(int j=0;j<s.length();j++){

                if(h.get(s.charAt(j))>1){

                    a=s.charAt(j);

                    break;

                }

            }

            System.out.println(a);

        }

    }

}

2). Number of Anagramic Groups

description

You are given N strings of length M, count the number of anagramic groups. An anagramic group is a list of strings which are anagrams of each other.

**Input Format**

The first line of input contains T - the number of test cases. In each test case, the first line contains N - the number of strings and M - the length of each string, separated by a space. The N subsequent lines each contain a string of length M.

**Output Format**

For each test case, print the count of the number of anagramic groups in the given N strings, separated by a new line.

**Constraints**

1 <= T <= 100

1 <= N <= 100

1 <= M <= 1000

'a' <= str[i] <= 'z'

**Example**

**Input**

2

5 4

arts

rank

star

rant

rats

9 5

tesla

start

slate

salte

tarts

aster

arson

astle

norse

**Output**

3

5

**Explanation**

**Test-Case 1**

The strings "arts", "star" and "rats" can be grouped together because they are anagrams of each other.

The total number of such groups is 3 and the groups are {"arts", "rats", "star"}, {"rant"} and {"rank"}.

import java.io.\*;

import java.util.\*;

public class Main {

    public static void main(String[] args) {

        /\* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class should be named Main. \*/

        Scanner sc=new Scanner(System.in);

        int t=sc.nextInt();

        for(int x=0;x<t;x++){

            int n=sc.nextInt();

            int m=sc.nextInt();

            HashMap<String,Integer> h=new HashMap<>();

            for(int i=0;i<n;i++){

                String s=sc.next();

                char a[] = s.toCharArray();

                Arrays.sort(a);

                String s1=new String(a);

                h.put(s1,h.getOrDefault(s1,0)+1);

            }

            System.out.println(h.size());

        }

    }

}

3) Enclosing Substring

description

Given 2 strings A and B, find the smallest substring of B having all the characters of A, in any order.

**Input Format**

The first line of input contains T - the number of test cases. It's followed by T lines, each line containing 2 space-separated strings - A and B.

**Output Format**

For each test case, print the length of the smallest substring of B having all the characters of A, separated by newline. If no such substring is found, print -1.

**Constraints**

20 points

1 <= T <= 100

1 <= size(A), size(B) <= 100

60 points

1 <= T <= 100

1 <= size(A), size(B) <= 1000

120 points

1 <= T <= 100

1 <= size(A), size(B) <= 10000

**General Constraints**

'a' <= A[i], B[i] <= 'z'

**Example**

**Input**

4

fkqyu frqkzkruqmfqyuzlkyg

onmwvytbytn uqhmfjaqtgngcwkuzyamnerphfmw

bloets lwbcrsfothplxseplrtbshbtstjloxsf

dzpd dclzztpjldkndgbdqqzmbp

**Output**

7

-1

13

9

**Explanation**

**Test Case 1:**

The smallest substring containing all characters of A is "fqyuzlk", which has a length of 7.

**Test Case 2:**

Despite considering all possible substrings of B, we cannot find any substring containing all characters of A.

**Test Case 3:**

The smallest substring containing all characters of A is "bcrsfothplxse", which has a length of 13.

**Test Case 4:**

The smallest substring containing all characters of A is "ztpjldknd", which has a length of 9.

import java.io.\*;

import java.util.\*;

public class Main {

    static boolean valid(int a1[],int b1[]){

        for(int i=0;i<26;i++){

            if(a1[i]>=b1[i])

            continue;

            else

            return false;

        }

            return true;

        }

    public static void main(String[] args) {

        /\* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class should be named Main. \*/

        Scanner sc=new Scanner(System.in);

        int t=sc.nextInt();

        for(int x=0;x<t;x++){

            String b=sc.next();

            String a=sc.next();

            int b1[]=new int[26];

            int a1[]=new int[26];

            int ans=a.length();

            for(int i=0;i<b.length();i++){

                b1[b.charAt(i)-97]+=1;

            }

            int p1=0,p2=0;

            while(p2<a.length()){

                if(valid(a1,b1)){

                    ans=Math.min(ans,p2-p1);

                    a1[a.charAt(p1)-97]-=1;

                    p1++;

                }

                else{

                    a1[a.charAt(p2)-97]+=1;

                    p2+=1;

                }

            }

            while(p1<a.length() && valid(a1,b1)){

                ans=Math.min(ans,a.length()-p1);

                a1[a.charAt(p1)-97]-=1;

                p1+=1;

            }

            if(ans==a.length())

            System.out.println(-1);

            else

            System.out.println(ans);

        }

    }

}

4). Substring Matching

description

You are given two strings A and B. You are also given Q queries with 4 indices i, j, k, and l. Check whether the substring of A[i:j] matches the substring of B[k:l].

**Input Format**

The first line of input contains T - the number of test cases. In each test case, the first line contains the string A and the second line contains the string B.

The next line contains an integer Q - the number of queries. The Q subsequent lines each contain 4 integers i, j, k, and l, separated by a space.

**Output Format**

For each query, on a new line, print "Yes" if the substring of A matches the substring of B, print "No" otherwise.

**Constraints**

30 points

1 <= T <= 100

1 <= len(A), len(B) <= 100

0 <= Q <= 1000

120 points

1 <= T <= 100

1 <= len(A), len(B) <= 10000

0 <= Q <= 10000

General Constraints

'a' <= A[i], B[i] <= 'z'

0 <= i <= j < len(A)

0 <= k <= l < len(B)

**Example**

**Input**

2

smartinterviews

intermediate

2

5 9 0 4

1 3 2 4

hackerrank

hackerearth

1

0 3 0 3

**Output**

Yes

No

Yes

**Explanation**

**Test-Case 1**

The substring of "smartinterviews" from index 5 to 9 is "inter". The substring of "intermediate" from 0 to 4 is also "inter". Since these both are equal, the output is "Yes".

The substring of "smartinterviews" from index 1 to 3 is "mar". The substring of "intermediate" from 2 to 4 is "ter". Since these both are not equal, the output is "No".

import java.io.\*;

import java.util.\*;

public class Main {

    static long mod=1000000007;

    static long power[]=new long[10001];

    static void po(){

    int p=37;

    power[0]=p;

    for(int i=1;i<10001;i++){

        power[i]=(power[i-1]\*p)%mod;

    }

}

    public static void main(String[] args) {

        /\* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class should be named Main. \*/

        Scanner sc=new Scanner(System.in);

        po();

        int t=sc.nextInt();

        for(int x=0;x<t;x++){

            String a=sc.next();

            String b=sc.next();

            //System.out.println(a+b);

            int n=a.length();

            int m=b.length();

            long sa[]=new long[a.length()];

            long sb[]=new long[b.length()];

            sa[0]=((a.charAt(0)-97)\*power[0])%mod;

            sb[0]=((b.charAt(0)-97)\*power[0])%mod;

            for(int i=1;i<n;i++)

                sa[i]=(sa[i-1]+((a.charAt(i)-97)\*power[i])%mod)%mod;

            for(int i=1;i<m;i++)

                sb[i]=(sb[i-1]+((b.charAt(i)-97)\*power[i])%mod)%mod;

            int q=sc.nextInt();

            for(int i=0;i<q;i++){

                int a1=0,a2=0,b1=0,b2=0;

                a1=sc.nextInt();

                a2=sc.nextInt();

                b1=sc.nextInt();

                b2=sc.nextInt();

                long ha=0,hb=0;

                if(a1!=0)

                    ha=(sa[a2]-sa[a1-1]+mod)%mod;

                else

                    ha=sa[a2];

                if(b1!=0)

                    hb=(sb[b2]-sb[b1-1]+mod)%mod;

                else

                    hb=sb[b2];

                int d=a1-b1;

                if(d>0){

                    hb=(hb\*power[d-1])%mod;

                }

                else if(d<0){

                    ha=(ha\*power[Math.abs(d)-1])%mod;

                }

                if(ha==hb){

                    System.out.println("Yes");

                }

                else

                    System.out.println("No");

            }

        }

    }

}

5. Rabin Karp String Matching Algorithm

description

Given 2 strings A and B, find the number of occurrences of A in B as a substring.

Note:

Solve using the Rabin-Karp string matching algorithm. Do not use an inbuilt library.

**Input Format**

The first line of input contains T - the number of test cases. It's followed by T lines. Each line contains 2 strings - A and B, separated by space.

**Output Format**

For each test case, print the number of occurrences of A in B, separated by a new line.

**Constraints**

30 points

1 <= len(A), len(B) <= 1000

70 points

1 <= len(A), len(B) <= 10000

General Constraints

1 <= T <= 2000

'a' <= A[i], B[i] <= 'z'

**Example**

**Input**

4

smart yekicmsmartplrplsmartrplplmrpsmartrpsmartwmrmsmartsmart

interviews interviewseiwcombvinterviewskrenlzp

ds dsdsajdsrjjdsjjj

algo yalgoalgoalgopalgoaxalgoasaxalgolalgoalgoalgo

**Output**

6

2

4

9

import java.io.\*;

import java.util.\*;

public class Main {

    static int mod=1000000007;

    static int valid(String a,String b){

        long ha=0,hb=0,temp1=0,temp2=0,p=91;

        int count=0;

        temp1=p;

        temp2=p;

        int n=a.length();

        int m=b.length();

        for(int i=0;i<n;i++){

            ha=(ha+((b.charAt(i)-97)\*temp1)%mod)%mod;

            hb=(hb+((a.charAt(i)-97)\*temp1)%mod)%mod;

            temp1=(temp1\*p)%mod;

        }

        if(ha==hb){

             count++;

        }

            for(int i=n;i<m;i++){

               ha=(ha-((b.charAt(i-n)-97)\*temp2)%mod+mod)%mod;

               ha=(ha+((b.charAt(i)-97)\*temp1)%mod)%mod;

                if(ha==(hb\*temp2)%mod){

                    count++;

                }

                temp1=(temp1\*p)%mod;

                temp2=(temp2\*p)%mod;

            }

        return count;

    }

    public static void main(String[] args) {

        /\* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class should be named Main. \*/

        Scanner sc=new Scanner(System.in);

        int t=sc.nextInt();

        for(int x=0;x<t;x++){

            String a=sc.next();

            String b=sc.next();

            System.out.println(valid(a,b));

        }

    }

}