Digitos Diferentes

|  |  |
| --- | --- |
| Fecha de Elaboración: | Octubre 3, 2013 |
| Autores: | Diego Serrano |
| Colaboradores: | Diana Gómez |
| Fuente: | 12527 - Different Digits  UVa Online Judge |

## Problema

Los habitantes de Nlogonia son muy supersticiosos. Una de sus creencias es que los números de la casa de calle que tienen un dígito repetido traen mala suerte para los residentes. Por lo tanto, nunca vivirían en una casa que tiene un número de la calle, como 838 o 1004.

La reina de Nlogonia ordenó construir una nueva vía costera, y quiere asignar a las nuevas viviendas sólo números sin dígitos repetidos, para evitar malestar entre sus súbditos. Usted ha sido nombrado por Su Majestad para escribir un programa que, dados dos enteros *N* y *M*, determine el número máximo de viviendas a las que se pueden asignar números de calle entre *N* y *M*, inclusive, que no tengan dígitos repetido.

**Entrada**

Cada caso de prueba es descrito en una sola línea. La línea contiene dos enteros *N* y *M*, como se describió anteriormente (*1<=N<=M<=5000*).

**Salida**

Para cada caso de prueba, imprima una línea con un entero representando el número de números de calle entre *N* y *M*, sin dígitos repetidos.

**Ejemplo de Entrada**

87 104

989 1022

22 25

1234 1234

**Ejemplo de Salida**

14

0

3

1

## PLANTEAMIENTO DE SOLUCION BASICA

Intuitivamente el problema llama a que cada vez que recibamos un intervalo, hacemos un ciclo desde N hasta M, inclusive, explorando cada número y detectando si tiene dígitos repetidos. En caso de encontrar dígitos repetidos, se añadirá uno más a un contador.

El problema con la solución básica es que la complejidad para responder cada caso de prueba es *O(N)*, lo cual puede tomar una cantidad considerable cuando hay muchos casos de prueba. En Uva Online Judge, el tiempo máximo para este problema es de 1 segundo, y esta solución excede éste límite.

## CODIGO FUENTE

import java.io.BufferedReader;

import java.io.InputStream;

import java.io.InputStreamReader;

import java.util.StringTokenizer;

public class Naive {

public static void main(String[] args) {

try {

InputStream input = System.in;

BufferedReader reader = new BufferedReader(new

InputStreamReader(input));

StringTokenizer tokenizer = new StringTokenizer(" ");

String params;

while ((params = reader.readLine()) != null) {

tokenizer = new StringTokenizer(params);

int n = Integer.parseInt(tokenizer.nextToken());

int m = Integer.parseInt(tokenizer.nextToken());

int c = 0;

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

c += different(i);

}

System.out.println(c);

}

reader.close();

} catch (Exception e) {

e.printStackTrace();

}

}

public static int different(int arg) {

int[] dig = new int[12];

for (int i = 0; i < 12; i++) {

dig[i] = 0;

}

while (arg > 0) {

dig[arg % 10]++;

if (dig[arg % 10] > 1)

return 0;

arg /= 10;

}

return 1;

}

}

## Planteamiento de la Solución ÓPTIMA

Usando programación dinámica podemos pre-calcular cuántos números con dígitos diferentes hay de *1* a *K*, para cada *K* tal que *1<=K<=N*.

El pre-cálculo de numero de dígitos diferentes *dig(K)* se hace sumando 1 al elemento de *dig(K-1)*, si *K* no tiene dígitos repetidos, de otra manera, *dig(K) = dig(K-1)*.

Para responder cada caso de prueba, solo tenemos que hacer *dig(M) – dig(N-1)*.

La complejidad para el pre-procesamiento es lineal *O(N)*, y el tiempo para responder cada caso de prueba es constante *O(1)*.

La solución fue enviada a Uva Online Judge el 1 de octubre de 2013, con un tiempo de ejecución de 0.205 segundos.

## Código Fuente

import java.io.BufferedReader;

import java.io.InputStream;

import java.io.InputStreamReader;

import java.util.StringTokenizer;

public class Main {

public static final int MAX = 5001;

public static int[] dig = new int[MAX];

public static void main(String[] args) {

try {

dig[0] = 0;

for (int i = 1; i < MAX; i++)

dig[i] = dig[i - 1] + different(i);

InputStream input = System.in;

BufferedReader reader = new BufferedReader(

new InputStreamReader(input));

StringTokenizer tokenizer = new StringTokenizer(" ");

String params;

while ((params = reader.readLine()) != null) {

tokenizer = new StringTokenizer(params);

int n = Integer.parseInt(tokenizer.nextToken());

int m = Integer.parseInt(tokenizer.nextToken());

System.out.println(dig[m] - dig[n - 1]);

}

reader.close();

} catch (Exception e) {

e.printStackTrace();

}

}

public static int different(int arg) {

int[] dig = new int[12];

for (int i = 0; i < 12; i++) {

dig[i] = 0;

}

while (arg > 0) {

dig[arg % 10]++;

if (dig[arg % 10] > 1)

return 0;

arg /= 10;

}

return 1;

}

}