Evaluación Final Matemática Discreta I

Karlos Alejandro Alfonso Rodríguez
Correo: karlos.alfonso@estudiantes.matcom.uh.cu
Grupo: C-213

13 de julio de 2021

1. Problema 1485C

- Link del Problema: https://codeforces.com/problemset/problem/1485/C
- Link del submit (Accepted) en Codeforces: https://codeforces.com/problemset/submission/1485/121584796

1.1. Resumen en Términos Matemáticos

Dados dos enteros positivos x e y, el problema consiste en contar la cantidad de pares (a, b) que cumplen lo siguiente:

- $1 \leq a \leq x$
- $1 \leq b \leq y$
- $\lfloor \frac{a}{b} \rfloor = a \bmod b$

1.2. Problema Original

C. Floor and Mod

time limit per test: 2 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

A pair of positive integers (a, b) is called special if $\lfloor \frac{a}{b} \rfloor = a \bmod b$. Here, $\lfloor \frac{a}{b} \rfloor$ is the result of the integer division between a and b, while $a \bmod b$ is its remainder.

You are given two integers x and y. Find the number of special pairs (a, b) such that $1 \leq a \leq x$ and $1 \leq b \leq y$.

Input

The first line contains a single integer t $(1 \leq t \leq 100)$ - the number of test cases.

The only line of the description of each test case contains two integers x, y $(1 \leq x, y \leq 10^9)$.

Output

For each test case print the answer on a single line.

Example

input	output
9	
3 4	1
2 100	0
4 3	2
50 3	3
12 4	5
69 420	141
12345 6789	53384
123456 789	16090
12345678 9	36

Note

In the first test case, the only special pair is $(3, 2)$.

In the second test case, there are no special pairs.

In the third test case, there are two special pairs: $(3, 2)$ and $(4, 3)$.

1.3. Solución Teórica del Problema

Si aplicamos el Algoritmo de la División sobre los enteros a y b ($b \geq 1$), obtenemos que $a = bq + r$, con $0 \leq r < b$; pero si el par (a, b) es especial, entonces $q = r$, resultando:

$$a = br + r$$
$$a = r(b + 1) \tag{1}$$

Si r fuera igual a 0, por (1) tenemos que $a = 0$ lo que entra en contradicción con una de las condiciones iniciales del problema que dice $1 \leq a$, por lo tanto $r > 0$.
Partiendo de $r < b$:

$$\begin{array}{rcl} r & < & b \\ r & < & b+1 \\ r^2 & < & r(b+1) \;\; \text{multiplicamos por } r > 0 \\ r^2 & < & a \leq x \;\; \text{aplicamos (1)} \\ r^2 & < & x \\ |r| & < & \sqrt{x} \\ r & < & \sqrt{x} \;\; \text{como } r > 0 \end{array}$$

Para mayor comodidad digamos $r \leq \sqrt{x}$, esto no afecta la veracidad de la desigualdad ni la correctitud de la solución.
Si fijamos a r podemos saber cuantos pares de números especiales existen para ese r. Veamos como:
Primeramente trabajemos con $1 \leq a \leq x$.

$$\begin{array}{rcl} 1 \leq a & \leq & x \\ br + r & \leq & x \;\; \text{por } a = br + r \\ br & \leq & x - r \\ b & \leq & \dfrac{x}{r} - 1 \end{array}$$

Debemos contar los enteros b que cumplen las siguientes condiciones:

1. $b > r$
2. $1 \leq b \leq y$
3. $1 \leq b \leq \frac{x}{r} - 1$

Como r es fijo, siempre que encontremos un entero b que cumpla las condiciones anteriores, se garantiza que habrá un entero a tal que (a, b) sea un par especial, porque a está expresado en función de b y r.

1.4. Algoritmos

1.4.1. Fuerza Bruta:

La solución por fuerza bruta consiste en hallar todos los pares (a, b) e ir comprobando si cumplen o no con la condición de ser especial. Esta solución es la más intuitiva, pero es muy ineficiente y no hace uso de las facilidades que nos brindan las condiciones iniciales del problema.

A continuación el algoritmo por fuerza bruta:

```
def CountSpecialsPairs_BF(x, y):
    count = 0
    for i in range(1, x + 1):
        for j in range(1, y + 1):
            div = i / y
            mod = i % y
            if (div == mod)
                count += 1
    return count
```

La complejidad temporal de este algoritmo es $O(xy)$, ya que se realizan dos recorridos de rango x e y respectivamente.

1.4.2. Utilizando Teoría de Números:

Anteriormente demostramos que contando los enteros b que cumplían ciertas condiciones para un r fijo, hallaríamos la cantidad de pares (a, b) que son especiales. Veamos como se refleja esto en el algoritmo.

Para un r fijo la cantidad de enteros b estará definida por las cotas superiores halladas anteriormente ($b \leq y$ y $b \leq \frac{x}{r} - 1$). Como estas cotas son potencialmente distintas, nos quedaremos con la menor de ellas, o sea: $min(y, \frac{x}{r} - 1)$. Hasta aquí hay un problema, no estamos teniendo en cuenta que $b > r$, para solucionar esto debemos restar r a la cantidad de enteros b hallada, porque haciendo esto garantizamos que empezamos a contar los $b_s > r$, entonces será: $min(y, \frac{x}{r} - 1) - r$.

Finalmente la solución consiste en iterar por los enteros r tales que $1 \leq r \leq \sqrt{x}$ e ir calculando para cada r la cantidad de pares especiales que existen. Teóricamente r es menor que y, pero en la práctica puede darse el caso de que al iterar por los enteros r estos sean mayores que y, esto quiere decir que $\nexists b$ que sea mayor que r y para ese caso la fórmula obtenida daría como resultado un número negativo, lo cual no tiene sentido. Una forma de solucionar esto es hacerle una modificación a dicha fórmula, resultando: $max(0, min(y, x/r - 1) - r)$. A continuación el algoritmo:

```
def CountSpecialsPairs_NT(x, y):
    count = 0
    for r in range(1, sqrt(x) + 1):
        count += max(0, min(y, x/r - 1) - r)
    return count
```

La complejidad temporal de este algoritmo es $O(\sqrt{x})$, una mejora considerable respecto al algoritmo por fuerza bruta.

1.4.3. Tester:

Se ha implementado un Tester para comprobar la correctitud del algoritmo implementado utilizando teoría de números. El tester genera valores random de x e y, y valida la respuesta en relación a la obtenida del algoritmo por fuerza bruta, ya que este algoritmo garantiza que se comprueban todos los pares posibles. Por cada caso de prueba generado se imprime si fue correcto o no, de encontrarse un caso incorrecto este se imprime.

A continuación la implementación:

```
def Tester(count_tests, ran_x, ran_y):
    results = []
    incorrects = []
    for i in range(0, count_tests):
        x = random(1, ran_x)
        y = random(1, ran_y)
        if (CountSpecialsPairs_NT(x,y) == CountSpecialPairs_BF(x,y)):
            results.Insert("Correct")
        else:
            results.Insert("Incorrect")
            incorrects.Insert((x,y))
    print(results)
    print(incorrects)
```

La variable *count_tests* indica la cantidad de casos de prueba que queremos generar, *ran_x* será la cota superior para x, mientras que *ran_y* será la cota superior de y. Como el Tester se apoya en el algoritmo por fuerza bruta(O(xy)) cabe señalar que para valores muy elevados de *ran_x* y *ran_y* demorará validar los casos.

2. Teoremas Utilizados:

2.1. Algoritmo de la División:

Dados dos enteros a y b ($b > 0$), existen enteros únicos q y r tales que: $a = qb + r$ y ($0 \leq r < b$).

3. Bibliografía:

- David M. Burton. DIVISIBILITY THEORY IN THE INTEGERS. Elementary Number Theory