Descripción

Últimamente has estado estudiando matemáticas y te tiene asombrado la conjetura de los primos gemelos, que dice lo siguiente:

"Existe un número infinito de primos p tales que $p + 2$ también es primo."

No puedes comprender cómo es posible que esta aseveración no ha sido demostrada. De hecho, estás convencido de que no solamente se cumple para p y $p + 2$, sino que se cumple para p y $p + d$, para cualquier valor de d par. Una noche, sin poder dormir pensando en esta conjetura, te lanzas a escribir un programa que de todas las parejas de primos con diferencia d, donde d es cualquier entero positivo (no necesariamente par).

Entrada

- En la primera línea el valor de n.
- En la segunda línea el valor de d (no necesariamente par).

Salida

Todas las parejas de primos menores a n con diferencia d. Cada línea consiste de dos números separados por un espacio, representando una pareja de números con diferencia d, con el primer número siendo el menor de la pareja. La salida debe cumplir que el primer número de cada línea es menor al primer número de la línea siguiente.

Notas

- Puedes asumir que la salida siempre tendrá al menos una pareja de primos.

Límites

- $n \leq 10000000$.
- $d \leq 10000000$.

Subtarea 1 [10 puntos]

$n \leq 50000, d = 2$.

Subtarea 2 [10 puntos]

$n \leq 500000, d = 2$.

Subtarea 3 [10 puntos]

$d = 2$.

Subtarea 4 [10 puntos]

$n \leq 50000, d \leq 50000$.

Subtarea 5 [20 puntos]

$n \leq 500000, d \leq 500000$.

Subtarea 6 [40 puntos]

Sin restricciones.

Solución:

Para poder buscar parejas de primos con diferencia dada, tendremos que pre-computar un arreglo con todos los primos menores a n. Hay varias formas de hacer esto, y explicaremos brevemente tres de ellas. Cada forma nos ayudará a afrontar los límites de $n \leq 50000$, $n \leq 500000$ y $n \leq 10000000$ respectivamente. La primera forma es iterar sobre cada número menor a n, evaluando si es primo o no, y añadiéndolo en un arreglo en caso de serlo. Para evaluar si un número en particular es primo o no, basta contar la cantidad de sus divisores y evaluar si es igual a 2. Para contar los divisores de un número x, podemos iterar sobre todos los enteros positivos menores o iguales a x, usando el operador módulo para determinar si el número es un divisor de x. Esta solución tiene complejidad cuadrática $O(n^2)$. Podemos optimizar esto dándonos cuenta que no es necesario evaluar todos los números menores a x y es suficiente deternos al llegar hasta $\lceil \sqrt{x} \rceil$, ya que si no hay divisores (quitando el valor de 1) más antes de este valor, es imposible que existan luego de él ya que los divisores vienen en parejas $(y, \frac{x}{y})$. Esta solución tiene complejidad $O(n\sqrt{n})$

Para afrontar el mayor valor de n posible, debemos usar el algoritmo de la Criba de Eratóstenes que es óptimo para pre-computar primos. El algoritmo comienza con una lista de todos los números $[2, n]$. Repetitivamente se eliminan todos los múltiplos del primer número no eliminado (exceptuando a él). Al final, los primos son los números que quedaron sin eliminar. Se realizan dos optimizaciones como son: empezar a eliminar los múltiplos de p desde p^2 en adelante. Se puede hacer esto pues todos los múltiplos anteriores $2p, 3p, ...$, ya fueron eliminados al considerar los primos 2, 3, ..., etc. Similarmente, podemos detener la eliminación al llegar hasta $p \geq \lceil \sqrt{n} \rceil$, por un argumento similar a que los divisores pueden ser emparejados. La complejidad de este algoritmo es $O(n \log \log n)$.

Lo siguiente es realizar la búsqueda de los pares de primos una vez que se tenga el arreglo pre-computado. La forma obvia es para cada primo, iterar sobre cada primo mayor que él, verificando si la diferencia es igual a d. Esta solución es claramente cuadrática en el número de primos. Sin embargo cuando $d = 2$, podemos detenernos al chequear el siguiente número solamente, ya que los números primos a partir del 3 son todos impares y tienen como diferencia mínima el valor de 2. Para optimizar esta búsqueda, podemos iterar sobre cada primo p y realizar búsqueda binaria para el valor de $p+d$ en el arreglo de los primos, pues está ordenado por construcción. Esta solución tiene complejidad $O(p \log p)$ si p es la cantidad de primos menor a n.