Explicación del Problema: Simple Repetition
En este documento analizaremos el enunciado del problema número 7, titulado Simple Repetition, proveniente de la plataforma Codeforces. El objetivo es comprender qué pide el problema, por qué el número generado casi nunca puede ser primo y cuál es el método eficiente para determinar la primalidad del valor resultante.
1. Enunciado del problema
A Pasha le encantan los números primos. Investigando en internet encontró un algoritmo que genera un nuevo número y de la siguiente forma:

1. Se toma un número entero x sin ceros a la izquierda.
2. Se repite su representación decimal k veces.
3. El resultado de esa repetición es el número y.

Por ejemplo:

- Si x=52 y k=3, entonces:

y=52 52 52

- Si x=6 y k=7:

y=6666666
El reto consiste en determinar si el número y es un número primo.
Límites:

- x puede ser tan grande como 10^9
- k es como máximo 7
- Se debe imprimir YES si y es primo y NO si no lo es.

2. ¿Es necesario construir realmente el número?
El número y se obtiene repitiendo x varias veces, pero esto produce valores enormes. Por ejemplo:

- Si x=1000000 y k=6, el número resultante tendría 6 millones de dígitos, imposible de almacenar en un tipo de dato normal.

Por lo tanto, antes de pensar en técnicas de primalidad, debemos analizar si es necesario generar el número repetido.
La respuesta es no.
3. ¿Cuándo un número repetido puede ser primo?
Analicemos qué sucede al repetir un número:
Caso 1: k = 1
Si k=1, entonces simplemente debemos verificar si x es primo.
Esto es completamente válido.
Caso 2: k > 1
Repetir un número más de una vez es equivalente a multiplicarlo por un número compuesto.
Por ejemplo:

- Repetir “7” dos veces forma “77”, que equivale a:

77=7 x 11
Como resultado, 77 ya no puede ser primo.
Lo mismo ocurre con “777”, que equivale a 7×111, y también es compuesto.
Generalizando:

- Repetir un número x más de una vez genera un valor que siempre es divisible por x
- También es divisible por el número formado por puros unos (como 11, 111, 1111…)

Por lo tanto:
Si k>1, el número no será primo
La única excepción
Existe un único caso donde repetir un número genera un valor primo:

- x=1
- k=2

Esto produce:
11
Y 11 sí es primo.
Con esto ya se puede establecer una condición fundamental:
4. Conclusión clave
Solo hay dos casos donde puede haber un número primo
Cuando k=1
— Simplemente probamos si x es primo.

1. Cuando x=1 y k=2
   — El resultado es 11, que es primo.

Para todos los demás casos:

La respuesta es NO

5. ¿Cómo verificar si un número es primo?

Cuando debamos verificar directamente la primalidad de x (solo cuando k=1), usamos el método tradicional:

Método ingenuo

El método básico consiste en revisar si x es divisible por cualquier número desde 2 hasta x−1.

Pero esto es demasiado lento si x es grande (hasta 10^9).

Método optimizado por raíz cuadrada

Sabemos que los divisores de un número se presentan en parejas:

Ejemplo con 100:

- (2, 50)
- (4, 25)
- (5, 20)
- (10, 10)

Observamos que todos los divisores pequeños están antes de la raíz:

$\sqrt{100}$=10

Si un divisor mayor existe, su pareja menor está antes de la raíz.

Por tanto, si buscamos divisores solo hasta la raíz cuadrada, es suficiente para determinar si un número es compuesto.

Procedimiento

Para verificar si x es primo:

1. Si x≤1x : no es primo.
2. Revisar divisiones desde 2 hasta $|\sqrt{x}|$:
   - Si algún residuo da 0 → no es primo.
3. Si ninguno divide perfectamente → es primo.

Este método es eficiente incluso para números de hasta 10^9.

6. Estructura final de la solución

1. Si x=1 y k=2
   ⇒ imprimir YES
2. Si k>1
   ⇒ imprimir NO
3. Si k=1
   ⇒ verificar si x es primo usando la raíz cuadrada
   ⇒ imprimir YES/NO según el resultado

Con esto se resuelve completamente el problema.