# Aritmética: Números primos

Programación Competitiva UPC

# Problema de motivación

**¿Cómo determinamos si un número es primo?**

# Definición

Un número primo es un natural que tiene exactamente **dos divisores distintos**: 1 y el mismo.

La propiedad de ser primo se denomina **primalidad**

# Test de primalidad

```
int test_primalidad_1(int n){

    if(n==1) return 0;

    for(int i=2; i<n; i++){
        if( n%i==0 ) return 0;
    }

    return 1;
}

```

Complejidad: O(n)

# Test de primalidad

```
int test_primalidad_2(int n){

    if(n==1) return 0;

    limite = sqrt(n)
    for(int i=2; i<=limite; i++){
        if( n%i==0 ) return 0;
    }

    return 1;
}
```

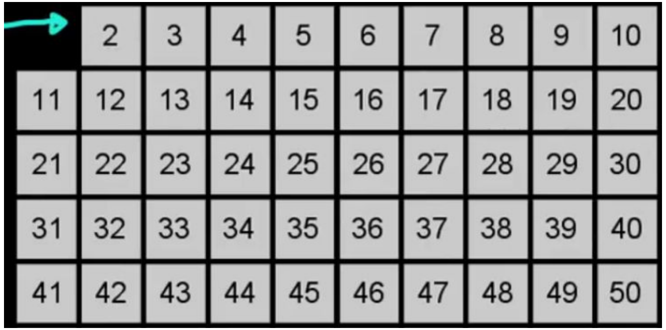
Complejidad: O( $\sqrt{n}$

# Problema de motivación

**¿Y si ahora nos consultan por todos los números de 2 a n?**

# Criba de Eratóstenes

Empezamos con todos los números a partir de 2 marcados como primos

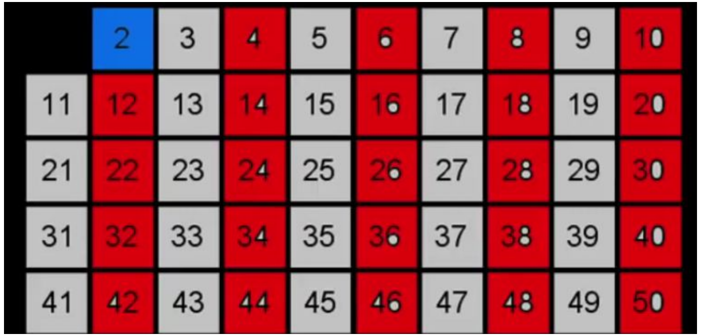



Si el número analizado sigue marcado como primo significa que es primo finalmente

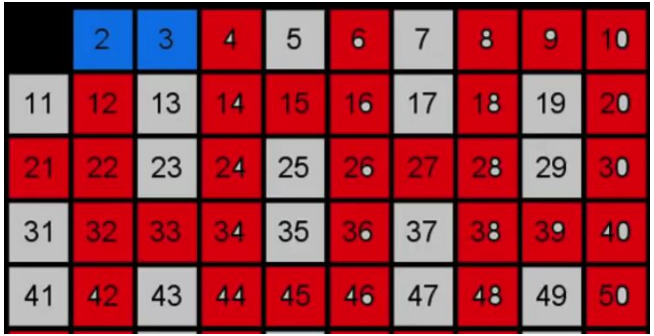

[Visualización]()

# Criba de Eratóstenes

Y además marcamos todos sus múltiplos como no primos



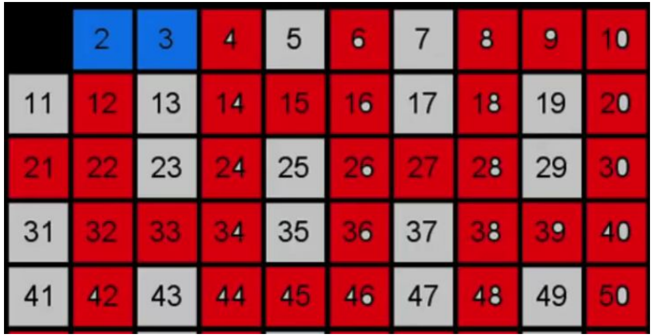

Seguimos con 3 de igual manera



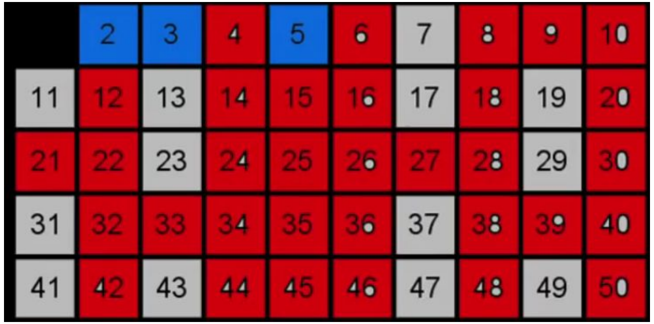

Visualización

# Criba de Eratóstenes

Cuando llegamos a 4 ya no es primo, así que finalmente no es primo



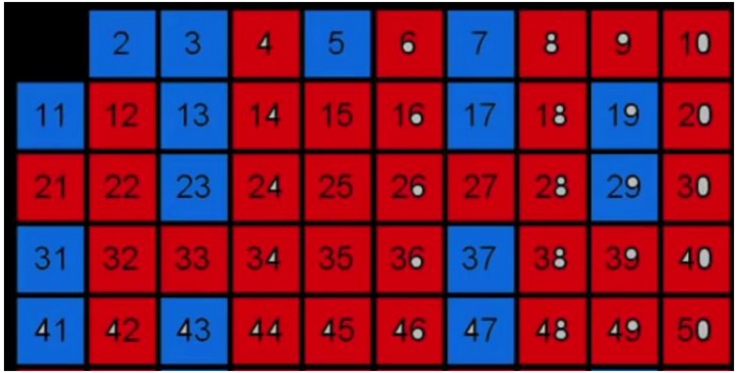

Seguimos con 5




[Visualización]

# Criba de Eratóstenes

Así sucesivament, hasta finalizar




[Visualización completa]

# Criba de Eratóstenes

```
int es_primo[n]

int limite=sqrt(n);      // Solo debemos iterar hasta la raiz

for(int i=2;i<n;i++) es_primo[i]=1; // Inicializamos todos como primos

es_primo[0]=es_primo[1]=0     // Solo para recordar que 0 y 1 no son primos

for(int i=2;i<=limite;i++){

    if (es_primo[i]){
        // Recorremos los múltiplos de i desde i^2
        for(int j=i*i; j<n; j+=i ){
            es_primo[j]=0;
        }
    }
}

// Al finalizar el arreglo es_primo tendra los resultados

```

Complejidad: O( n log log n )

n <= 10^7

# Pregunta

**¿Qué podemos hacer queremos saber la primalidad de varios números?**

**Pero estos son > 10^8**

**¿Podemos optimizar en algo los primeros test de primalidad que vimos?**

# Teorema fundamental de la aritmética

Todo número natural se puede representar de forma única mediante sus factores primos, salvo el orden

$$6936 = 2^3 \cdot 3 \cdot 17^2$$

$$1200 = 2^4 \cdot 3 \cdot 5^2$$

Forma general

$$n = \prod_{i=1}^{r} p_i^{a_i}$$

# Fórmulas generales

$$n = \prod_{i=1}^{r} p_i^{a_i}$$

Cantidad de divisores de n

$$= \prod_{i=1}^{r} (a_i + 1).$$

Ejemplo para 60 = 2^2 x 3 x 5

Cantidad de divisores = (2+1)(1+1)(1+1) = 12

Suma de divisores de n

$$= \prod_{i=1}^{r} (1 + p_i^{x} + p_i^{2x} + \cdots + p_i^{a_i x}).$$

Ejemplo para 60 = 2^2 x 3 x 5

Suma de divisores = (1+2+4)(1+3)(1+5) = 168

# Criba de Eratóstenes modificada

Nos sirve para guardar el primo divisor del número para agilizar cálculos de divisores

```
int criba_primos[n];

int limite = sqrt(n);   //Solo iteramos hasta la raiz

for(int i=2;i<n; i++) criba_primos[i]=1; //inicializamos la criba

criba_primos[0]=criba_primos[1]=0; // 0 y 1 casos especiales

for(int i=2; i<=limite; i++){

    if (criba_primos[i]==1){ //Siginifica que es primo
        for(int j=i*i; j<n; j+=i){
            criba_primos[j]=i; //Guardamos ese factor primo en la criba
        }
    }
}

// Al finalizar los números con criba_primos[i] igual a 1 son primos
// Los compuestos tendrán alguno de sus factores primos en la criba

```

Complejidad: O( n log log n )

n <= 10^7

# Problemas

10001 prime

Largest prime factor

Smith numbers

Twins **Solución**

highly divisible triangular number Solución

Gracias !