Números primos

Gastón Fontenla, David Lescano

Universidad Nacional de La Matanza

Octubre 2019

Definiciones

- **Números primos:** los números naturales que tiene únicamente dos divisores distintos: él mismo y el 1 Ej: **2, 5, 97**
- **Números compuestos:** los números naturales que tienen algún divisor natural aparte de sí mismos y del 1, y, por lo tanto, pueden factorizarse. Ej: **8, 21, 63**
- **¿Qué pasa con el 1?** El número 1, por convenio, no se considera ni primo ni compuesto.
- Clasificando los naturales del 1 al 15:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

¿Cómo saber si un número es primo?

- Dado N, determinar si es primo
- ¡Recordemos definición de número primo!
- Probamos los números desde el 2 hasta el N-1, y nos fijamos si alguno divide a N, utilizando el operador módulo (%). Si alguno de estos números divide a N, entonces N no es primo.
- A este algoritmo se lo conoce como **Test de Primalidad**

Código Test de Primalidad

```
1  bool esPrimo(int n)
2  {
3      if(n < 2)
4          return false;
5
6      for(int i=2; i<n; i++)
7      {
8          if(n % i == 0)
9          {
10             return false;
11         }
12     }
13
14     return true;
15 }
```

Extendiendo el Test de Primalidad

- Ahora, por cada número menor o igual a N, determinar cuales son primos

```
1 vector <int> primos;
2 for(int i=2; i<=n; i++)
3 {
4     if(esPrimo(i) == true)
5     {
6         cout << i << " es primo" << endl;
7         primos.push_back(i);
8     }
9 }
```

- Con este código, guardamos los primos menores o iguales a N en el vector primos

Otra forma de pensarlo

- En una cuadrícula, ponemos los números de 1 a N
- Tomamos el primer número en blanco, y lo marco como **primo**. Luego, marcamos como **compuesto** a sus múltiplos.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50

Otra forma de pensarlo

- En una cuadrícula, ponemos los números de 1 a N
- Tomamos el primer número en blanco, y lo marco como **primo**. Luego, marcamos como **compuesto** a sus múltiplos.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50

Otra forma de pensarlo

- En una cuadrícula, ponemos los números de 1 a N
- Tomamos el primer número en blanco, y lo marco como **primo**. Luego, marcamos como **compuesto** a sus múltiplos.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50

Otra forma de pensarlo

- En una cuadrícula, ponemos los números de 1 a N
- Tomamos el primer número en blanco, y lo marco como **primo**. Luego, marcamos como **compuesto** a sus múltiplos.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50

Otra forma de pensarlo

- En una cuadrícula, ponemos los números de 1 a N
- Tomamos el primer número en blanco, y lo marco como **primo**. Luego, marcamos como **compuesto** a sus múltiplos.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50

Otra forma de pensarlo

- En una cuadrícula, ponemos los números de 1 a N
- Tomamos el primer número en blanco, y lo marco como **primo**. Luego, marcamos como **compuesto** a sus múltiplos.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50

Otra forma de pensarlo

- En una cuadrícula, ponemos los números de 1 a N
- Tomamos el primer número en blanco, y lo marco como **primo**. Luego, marcamos como **compuesto** a sus múltiplos.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50

Otra forma de pensarlo

- En una cuadrícula, ponemos los números de 1 a N
- Tomamos el primer número en blanco, y lo marco como **primo**. Luego, marcamos como **compuesto** a sus múltiplos.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50

Otra forma de pensarlo

- En una cuadrícula, ponemos los números de 1 a N
- Tomamos el primer número en blanco, y lo marco como **primo**. Luego, marcamos como **compuesto** a sus múltiplos.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50

Otra forma de pensarlo

- En una cuadrícula, ponemos los números de 1 a N
- Tomamos el primer número en blanco, y lo marco como **primo**. Luego, marcamos como **compuesto** a sus múltiplos.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50

Otra forma de pensarlo

- En una cuadrícula, ponemos los números de 1 a N
- Tomamos el primer número en blanco, y lo marco como **primo**. Luego, marcamos como **compuesto** a sus múltiplos.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50

¿Cómo se implementa?

- La cuadrícula en realidad es un vector, lo hacemos como cuadrícula para poder verlo en pantalla con N grande.
- Dicho vector es un vector de bool
- **true** significa que es primo
- **false** significa que es compuesto
- Inicialmente todos son primos (exceptuando el 0 y el 1)
- A este algoritmo se lo conoce como **Criba de Eratóstenes**

Código Criba de Eratóstenes

```
1  int N;
2  cin >> N;
3
4  vector <bool> esPrimo(N+1, true);
5  esPrimo[0] = false;
6  esPrimo[1] = false;
7
8  for(int i=2; i<=N; i++)
9  {
10     if(esPrimo[i])
11     {
12         for(int j=i*2; j<=N; j+=i)
13         {
14             esPrimo[j] = false;
15         }
16     }
17 }
```

Tiempos y cotas

- La función esPrimo(N) internamente hace N operaciones. Se puede hacer más rápido, haciendo una observación en la definición de un número compuesto.
- Sin entrar en detalles, la Criba de Eratóstenes es más rápida que hacer muchas llamadas a la función esPrimo(), incluso si la comparamos con la versión rápida de esta función.
- El **Test de Primalidad** funciona para números hasta 100.000.000
- La **Criba de Eratóstenes** funciona para números hasta 10.000.000

¡Material extra post-charla!

- Mencionamos una mejora en la función esPrimo(N)
- Recordemos que un número N compuesto se puede representar como el producto de dos números A y B (sin contar al 1 y a N).
- N = A*B. Si probamos los números desde 2 hasta N-1, en busca de algún divisor de N, nos toparemos primero con A o con B (el mas chico de los dos)
- Está garantizado que este número lo encontraremos en la raiz de N o menos.
- Supongamos que A es el menor de los divisores de N. Si A > raiz(N) entonces A*A es mayor a N (¡esto no puede ocurrir, es falso!). Asi que deducimos que A <= raiz(N), para que A*A <= N.
- En definitiva, no necesitamos probar si algún número en el rango [2, N-1] divide a N. Con probar el rango [2, raiz(N)] alcanza

Función esPrimo(N) mejorada

```
1  bool esPrimo(int n)
2  {
3      if(n < 2)
4          return false;
5
6      for(int i=2; i*i<=n; i++)
7      {
8          if(n  % i == 0)
9          {
10             return false;
11         }
12     }
13
14     return true;
15 }
```

¿Otra mejora más?

- Los números pares son compuestos, a excepción del 2
- Esto quiere decir, que podemos preguntar en un if si un número es par
- ¡Con esto, nos ahorramos mucho cálculo!
- Esto significa que solo preguntaremos si algún impar divide a N
- Preguntaremos por los impares en el rango [3, raiz(N)]
- Esta mejora hace que la función esPrimo(N) se ejecute en la mitad del tiempo que la versión presentada anteriormente
- Para comprobarlo, hacer llamadas a esPrimo(N) con todos los N: 1 <= N <= 10.000.000 en ambas versiones, y comparar el tiempo de ejecución

Función esPrimo(N) doblemente mejorada

```
1  bool esPrimo(int n)
2  {
3      if(n == 2)
4          return true;
5
6      if(n < 2 || n%2 == 0)
7          return false;
8
9      for(int i=3; i*i <= n; i+=2)
10     {
11         if(n%i == 0)
12         {
13             return false;
14         }
15     }
16     return true;
17 }
```

¡Problemas!

- Buscando números primos:
  http://juez.oia.unsam.edu.ar/#/task/busca_primos/statement
- Contando pares sumaprimos:
  http://juez.oia.unsam.edu.ar/#/task/cuenta_primos/statement
- ¡A encontrar primos! (o casi):
  http://juez.oia.unsam.edu.ar/#/task/casi-primo/statement

Referencias

- https://es.wikipedia.org/wiki/Test_de_primalidad
- https://es.wikipedia.org/wiki/Criba_de_Eratóstenes
- http://www.oia.unsam.edu.ar/wp-content/uploads/2013/10/temario-orientativo-oia.pdf