Wydział
Informatyki

Teoria de Cribas

Introduccion y ejemplos

Jose Lucha (18904)

Agenda

Ideas Básicas

1. Distinción por propiedades: Crear un mecanismo capaz de diferenciar elementos de un conjunto de interés en base a una propiedad de interés.
2. Generación o búsqueda: No solo distinguir que números poseen una propiedad sino generar o encontrar y listar estos números.
3. Conteo de números: En base a la capacidad de nuestros métodos y el punto de paro contar cuantos números con la propiedad han sido encontrados.

Formalización de la teoría

Nota: De manera general se usaran p, q para denotar números primos, n, m, d, entre otros para denotar números naturales.

Definition

Sea $\pi(x)$ como una función para contar números primos de la forma:

$$\pi(x) = \sum_{p \leq x} 1$$

Formalización de la teoría

Definition

Sea A un conjunto finito de números naturales, P un conjunto de números primos y z tal que $z > 1$. Llamaremos a A como el conjunto cribado, P el rango de cribado y z el nivel de cribado. Definimos la función de cribado

$$S(A, P, z) = \sum_{a \in A \,; (a, P(z) = 1)} 1$$

donde

$$P(z) = \Pi_{p \in P \,; p < z} p$$

Criba de Sundaram

1. Sean $n, i, j \in N$ tales que $1 \leq i \leq j$, y generamos la lista de números de 1 a n.
2. Seleccionamos un valor $k \in N$ para que sea una cota para i, j con el fin de optimizar el proceso.
3. Generamos todos los números posibles de la forma $i + j + 2ij$ tal que sean menores o iguales a n.
4. Eliminamos de nuestro listado original todos los números anteriores.
5. Cada elemento x de nuestra lista original es reemplazado por $2x + 1$.
6. Nuestra lista contiene solamente números primos.

Idea detrás del algoritmo

1. Sabemos de antemano que 2 es un numero primo y es el único primo par.
2. Nos enfocamos en encontrar primos impares.
3. Consideramos los números de la forma $i + j + 2ij$ donde $i \leq j \leq n$ y la fórmula de números impares $2x + 1$
4. Nótese que $2(i + j + 2ij) + 1 = 4ij + 2i + 2j + 1 = (2i + 1)(2j + 1)$. De esto tenemos una manera de identificar números compuestos por impares.
5. Con esto en mente removemos de nuestra lista los números de la forma $i + j + 2ij$ y evaluamos los números restantes en $2x + 1$ para obtener números primos.

Ejemplo

Tomemos la lista de números

$[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25]$

Ahora producimos nuestra lista de números de la forma $i + j + 2ij$ variando i, j tomando en cuenta aquellos menores o iguales a 25.

Nótese que al generar nuestra lista de números es posible que estos se repitan. Para ahorrar espacio escribiremos solamente la primera vez que un numero aparezca.

$1 + 1 + (2 * 1 * 1) = 4$; $2 + 3 + (2 * 2 * 3) = 17$

$1 + 2 + (2 * 1 * 2) = 7$; $1 + 6 + (2 * 1 * 6) = 19$

$1 + 3 + (2 * 1 * 3) = 10$; $1 + 7 + (2 * 1 * 7) = 22$

$2 + 2 + (2 * 2 * 2) = 12$; $3 + 3 + (2 * 3 * 3) = 24$

$1 + 4 + (2 * 1 * 4) = 13$; $1 + 8 + (2 * 1 * 8) = 25$

$1 + 5 + (2 * 1 * 5) = 16$;

Ejemplo

Al remover los números que generamos de nuestra lista original no quedamos con $[1, 2, 3, 5, 6, 8, 9, 11, 14, 15, 18, 20, 21, 23]$. Calculamos entonces:

$(2 * 1) + 1 = 3$; $(2 * 2) + 1 = 5$

$(2 * 3) + 1 = 7$; $(2 * 6) + 1 = 13$

$(2 * 8) + 1 = 17$; $(2 * 9) + 1 = 19$

$(2 * 11) + 1 = 23$; $(2 * 14) + 1 = 29$

$(2 * 15) + 1 = 31$; $(2 * 18) + 1 = 37$

$(2 * 20) + 1 = 41$; $(2 * 21) + 1 = 43$

$(2 * 23) + 1 = 47$

Y obtenemos la lista de números primos:

$[3, 5, 7, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]$

Implementación del algoritmo

```python
def Sundaram(n):
    a = []
    for i in range(n):
        a.append(i+1)
    k = n//2
    for j in range(k):
        for i in range(k):
            b = (i+1)+(j+1)+(2*(i+1)*(j+1))

            if b <= n:
                if b in a:
                    a.remove(b)
                    #print(a)
    leng = len(a)
    for i in range(leng):
        a[i] = (2*a[i]) +1
    return a
```

Rysunek: Criba de Sundaram

Prueba del algoritmo

```
Se encontraron  25 numeros primos.
Se realizaron  98  iteraciones.
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79,
83, 89, 97]

Se encontraron  45 numeros primos.
Se realizaron  2500  iteraciones.
[3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83,
 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 1
79, 181, 191, 193, 197, 199]
```

Rysunek: Prueba de las Cribas de Eratosthenes y Sundaram

Comparación

Con el propósito de comparar la eficiencia de la Criba de Sundaram hacemos mención de la antigua Criba de Eratosthenes y la más moderna Criba de Atkin. La Criba de Eratosthenes ya fue discutida en clase, por lo que no profundizaremos en ella. La Criba de Atkin es un algoritmo identificador de números primos como Eratosthenes y Sundaram pero es mucho más reciente. Fue creada en 2003 por Arthur Atkin y Daniel Bernstein. La idea detrás de esta criba es nuevamente el descarte de números compuestos, en este caso por el cuadrado de primos, pero conlleva una preparación de los datos distinta a las dos cribas previamente mencionadas.

Comparacion

En términos de complejidad tenemos que dependiendo de la implementación cada método tiene complejidad:

1. Criba de Eratosthenes: $O(n)$ o $O(log(log(n)))$
2. Criba de Sundaram: $O(nlog(n))$
3. Criba de Atkin: $O(n)$ o $O(\frac{n}{log(log(n)}$)

En este Sundaram es posiblemente el más pesado, mientras que Atkin es el más optimo, con Erathosthenes como un intermedio.

Referencias

1. Sariols, M. (2017) Sieve Theory and Applications, Facultad de Matematicas de la Universidad de Barcelona
2. Ford, K. (2020) Sieve Methods Lecture Notes
3. D. Abdullah, R. Rahim, D. Apdilah, S. Efendi, T. Tulus and S. Suwilo (2018) Prime Numbers Comparison using Sieve of Eratosthenes and Sieve of Sundaram Algorithm, IOP Conf. Series: Journal of Physics: Conf. Series 978 (2018) 012123
4. Havil, J. (2009) Sundaram's Sieve, Millennium Mathematics Project, University of Cambridge