

Algoritmo Pisano-Estrada para generar números primos (PExPi)



2020 Guianluca R. Pisano & Víctor Estrada Díaz

Algoritmo PExPi

Abstracción:

Un algoritmo optimo para construcción de una tabla de números primos, capaz de solo con el registro de primos encontrados y un contador asociado al primo es capaz de generar todos los números primos sin más que el sumar 1 y comparar. No hay divisiones ni multiplicaciones ni otras operaciones matemáticas.

Itera:

```
n = n+1
primo(n) = true
P[1]=2 > C[1]=[+1] -> [0] => primo(n)=false
P[2]=3 > C[2]=[+1] -> [0] => primo(n)=false
P[3]=5 > C[3]=[+1] -> [0] => primo(n)=false
P[4]=7 > C[4]=[+1] -> [0] => primo(n)=false
....
P[p]=P > C[5]=[+1] -> [0] => primo(n)=false
...
si primo(n) => añado n a la lista de primos y voy al siguiente (n+1)
Iterando.
```

Sólo tenemos el array de primos descubiertos P y sus contadores cíclicos C, la única operación matemática es sumar uno en cada C de cada primo y si tiene el valor del primo se pone a cero y n no es primo.

Cuando todos los contadores se pasan sin poner ninguno a cero entonces tenemos un nuevo numero primo.

Código en python:

```
 #(c) 2020 Guianluca R. Pisano & Víctor Estrada Díaz
 #autor Victor Estrada segun una idea de
 #Gianluca R. Pisano
 #Se trata de llevar un contador para
 #cada primo encontrado y con solo la
 #función contador (suma) determinar si
 #un numero es primo o no
 #crea el array de primos hasta el max
 #para ejecutar instale python y ejecute:
 #py primos.py
 #donde primos.py es este código.
```

```
import time
```

```

n=2
print(n)
max=1000
p = []
c = []
p.append(1)
c.append(0)
p.append(2)
c.append(1)
i=1
t0=time.time()
while i<max:
    primo=1
    j=1
    while j<=i:
        c[j]=c[j]+1
        if c[j] == p[j]:
            primo=0
            c[j]=0

    j=j+1

    if primo:
        i=i+1
        p.append(n)
        c.append(0)
        # descomente siguiente línea para imprimir todos
        # print(i,n)

    n=n+1

print(i,p[i])
t1=time.time()-t0
print('calculados ',i,' primos en ',t1,'seg')

```

Salida:

Ejecutado con QPython 3L en un teléfono Samsung Galaxy A50

```

/emulated/0/qpython/.last_tmp.py && exit <
2
2 3
3 5
4 7
5 11
6 13

```

7 17

8 19

.....

982 7741

983 7753

984 7757

985 7759

986 7789

987 7793

988 7817

989 7823

990 7829

991 7841

992 7853

993 7867

994 7873

995 7877

996 7879

997 7883

998 7901

999 7907

1000 7919

1000 7919

calculados 1000 primos

en 4.53965950012207 segundos

#[QPython] Press enter to exit ...