

Pisano-Estrada algorithm to generate prime numbers (PEXPi)

2020 Guianluca R. Pisano & Víctor Estrada Díaz

PEXPi algorithm

Abstraction:

An optimal algorithm for constructing a table of prime numbers, capable of only the record of primes found and a counter associated with the prime, is capable of generating all the prime numbers without more than adding 1 and comparing. There are no divisions or multiplications or other mathematical operations.

Causes:

```
n = n+1
primo(n) = true
P[1]=2 > C[1]=[+1] -> [0] => first(n)=false
P[2]=3 > C[2]=[+1] -> [0] => first(n)=false
P[3]=5 > C[3]=[+1] -> [0] => first(n)=false
P[4]=7 > C[4]=[+1] -> [0] => first(n)=false
...
P[p]=P > C[5]=[+1] -> [0] => prime(n)=false
...
if prime(n) => I add n to the list of primes and go to the next one (n+1)
Repeating.
```

We only have the array of discovered primes P and its cyclic counters C, the only mathematical operation is to add one in each C of each prime and if it has the value of the prime it is set to zero and n is not prime.

When all the counters are passed without setting any to zero then we have a new prime number.

Code in python:

```

#(c) 2020 Guianluca R. Pisano & Víctor Estrada Díaz
#author Victor Estrada according to an idea by
#Gianluca R. Pisano
#It is about keeping an accountant for
#every cousin found and with only the
#function counter (sum) determine if
#a number is prime or not
#create the array of primes up to max
#to run install python and run:
#py primos.py
#where primos.py is this code.
```

```
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)
        # uncomment next line to print all
        # print(i,n)

    n=n+1

print(i,p[i])
t1=time.time()-t0
print('calculated ',i,' primes and en ',t1,'sec')

```

Salida:

Running with QPython 3L on a Samsung Galaxy A50 phone

```

/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
calculated 1000 primes
in 4.53965950012207 seconds
```

```
#[QPython] Press enter to exit ...
```