COME ON CODE ON

A blog about programming and more programming.

Pollard Rho Brent Integer Factorization

with 10 comments

Pollard Rho is an integer factorization algorithm, which is quite fast for large numbers. It is based on Floyd's cycle-finding algorithm and on the observation that two numbers x and y are congruent modulo p with probability 0.5 after $1.177\sqrt{p}$ numbers have been randomly chosen.

Algorithm

```
Input : A number N to be factorized
Output : A divisor of N
If x mod 2 is 0
        return 2

Choose random x and c
y = x
g = 1
while g=1
        x = f(x)
        y = f(f(y))
        g = gcd(x-y,N)
return g
```

Note that this algorithm may not find the factors and will return failure for composite n. In that case, use a different f(x) and try again. Note, as well, that this algorithm does not work when n is a prime number, since, in this case, d will be always 1. We choose f(x) = x*x + c. Here's a python implementation :

```
1  def pollardRho(N):
2          if N%2==0:
3                  return 2
4          x = random.randint(1, N-1)
5          y = x
6          c = random.randint(1, N-1)
7          g = 1
8          while g==1:
9                  x = ((x*x)%N+c)%N
```

```
                y = ((y*y)%N+c)%N
                y = ((y*y)%N+c)%N
                g = gcd(abs(x-y),N)
        return g
```

In 1980, Richard Brent published a faster variant of the rho algorithm. He used the same core ideas as Pollard but a different method of cycle detection, replacing Floyd's cycle-finding algorithm with the related Brent's cycle finding method. It is quite faster than pollard rho. Here's a python implementation :

```
def brent(N):
        if N%2==0:
                return 2
        y,c,m = random.randint(1, N-1),random.randint(1, N-1),random.ra
        g,r,q = 1,1,1
        while g==1:
                x = y
                for i in range(r):
                        y = ((y*y)%N+c)%N
                k = 0
                while (k<r and g==1):
                        ys = y
                        for i in range(min(m,r-k)):
                                y = ((y*y)%N+c)%N
                                q = q*(abs(x-y))%N
                        g = gcd(q,N)
                        k = k + m
                r = r*2
        if g==N:
                while True:
                        ys = ((ys*ys)%N+c)%N
                        g = gcd(abs(x-ys),N)
                        if g>1:
                                break

        return g
```

-fR0DDY

Written by fR0DDY

September 18, 2010 at 11:51 PM

Posted in Algorithm, Programming

Tagged with algorithm, brent, cycle, integer, pollard, rho

10 Responses

Subscribe to comments with RSS.

Found your note about Pollard's method. It always sounded intimidating…and then I read about it; found your page; it all doesn't seem so bad.

sympy, a CAS in pure python, has this method as part of it's number theory module. You might be interested in checking it out.

C Smith

October 12, 2010 at 1:02 AM

Reply
Thanks, very Useful.
Though I need analyse these codes… but the more Description, the more useful…

Tnx

Afshin

February 2, 2011 at 2:58 AM

Reply
Great blog bro.

Ahmet Alp Balkan

March 31, 2011 at 4:47 PM

Reply
how do we get all the distinct factors of an integer by this method?
Thanks.

pranay

August 1, 2011 at 11:27 PM

Reply
This is the best idea about integer factorization, written here is to let more people know and participate.
A New Way of the integer factorization
1+2+3+4+……+k=Ny,(k<N/2),"k" and "y" are unknown integer,"N" is known Large integer.
True gold fears fire, you can test 1+2+3+…+k=Ny(k<N/2).
How do I know "k" and "y"?
"P" is a factor of "N",GCD(k,N)=P.

Two Special Presentation:
N=5287
1+2+3+…k=Ny
Using the dichotomy
1+2+3+…k=Nrm
"r" are parameter(1;1.25;1.5;1.75;2;2.25;2.5;2.75;3;3.25;3.5;3.75)
"m" is Square
(K^2+k)/(2*4)=5287*1.75 k=271.5629(Error)
(K^2+k)/(2*16)=5287*1.75 k=543.6252(Error)
(K^2+k)/(2*64)=5287*1.75 k=1087.7500(Error)
(K^2+k)/(2*256)=5287*1.75 k=2176(OK)
K=2176,y=448
GCD(2176,5287)=17
5287=17*311

N=13717421
1+2+3+…+k=13717421y
K=4689099,y=801450
GCD(4689099,13717421)=3803
13717421=3803*3607

The idea may be a more simple way faster than Fermat's factorization method(x^2-N=y^2)!
True gold fears fire, you can test 1+2+3+…+k=Ny(k<N/2).
More details of the process in my G+ and BLOG.

My G+ :https://plus.google.com/u/0/108286853661218386235/posts
My BLOG:http://hi.baidu.com/s_wanfu/item/00cd4d3c5a2fd089f5e4ad0a
Email:wanfu.sun@gmail.com

s-987618

November 22, 2012 at 7:23 PM

Reply
hey thanks a lot for this, I appreciate it. I looked through a couple of different pages and this is the first one that really made sense.

alexr1090

May 1, 2013 at 10:13 PM

Reply
Glad I could help.

fR0DDY

May 1, 2013 at 10:45 PM

Reply

Hey! This is a great blog. I went ahead and took your code and was trying some stuff more out on it (More euler problems actually) and i somehow ended up with a weird case. Sometimes, just sometimes when im giving in 9 as the input, i get a 9 as the output. Essentially it should be 3 right?

Is this a problem with my understanding? Any insight would be helpful.

Cheers & great work!

shrayas

August 30, 2013 at 11:03 PM

Reply

Hi!
In your code you calculate q with q = q*(abs(x-y))%N. Since you using mod N then q must be between 0 and N-1 (but could also be N-1 or 0). Then you use determine g with g = gcd(q,N) and g cant be N since q is smaller than N. So why are you checking if g=N in row 19?

Jesper

November 13, 2013 at 4:56 PM

Reply

Thanks! Note that to get gcd, you also need to add "from fractions import gcd"

nealmcb

December 15, 2013 at 8:44 PM

Reply