Primeritat i factorització

Artur Travesa

(versió 2024-07)

Capítol 6. Mètode de Fermat

Artur Travesa

(versió 2024-04)

6.0. Introducció

No hi ha (com a mínim, jo no en sé de cap) mètodes de factorització que funcionin bé per a tots els nombres; generalment, cada mètode funciona bé en alguns casos particulars molt específics.

Això fa difícil implementar un mètode de factorització general que tingui en compte totes les possibilitats.

Un d'aquests mètodes particulars és el mètode de factorització anomenat de Fermat. Funciona bé per a trencar en dos factors un nombre n=a·b de manera que els factors a, b, siguin tals que el valor absolut de la seva diferència (o sigui a-b o b-a) sigui un nombre natural "petit" (i aquí "petit" depèn de la capacitat de càlcul o del temps que estiguem disposats a invertir en intentar la descomposició de n).

Notem que si els factors són "grans" però la diferència "petita", aquest mètode trencarà el producte; però si canviem un dels factors, per exemple, a, per 3a (per mantenir que els factors siguin senars), la diferència 3a-b=2a+(a-b) és "gran" més "petit"="gran" i el mètode ja no funciona.

I a l'inrevés, si tenim el producte (3·a)·b i primerament dividim per 3 (per exemple, dividim per primers petits i només apareix el 3 com a factor petit i amb multiplicitat 1), el mètode funcionarà sobre el producte a·b.

En qualsevol cas, el mètode de Fermat és a la base d'altres mètodes més potents, i convé comprendre'l bé.

En farem una funció, però no la incorporarem a l'algoritme general de factorització.

6.1. El mètode

El mètode de Fermat es pot aplicar a un nombre natural senar, n>1 i, evidentment, si el volem factoritzar cal que sigui compost.

El menor nombre natural senar compost és n=9=3·3, i és un quadrat. I notem que si n=a^2, el càlcul de l'arrel quadrada de n, a, ja proporciona una factorització de n.

Per tant, un primer pas de l'algoritme serà mirar si el nombre és un quadrat.

En qualsevol cas, el mètode de Fermat es basa en el resultat següent, de demostració immediata com a exercici (cf. **[Tr-Ar,** cap. VII, punt 2.1**]**).

Sigui N un nombre natural senar qualsevol. Hi ha una bijecció entre el conjunt format per les parelles (a, b) de nombres enters tals que $N = ab$, amb $1 \leq b < a$, i el conjunt format per les parelles (x, y) de nombres enters tals que $N = x^2 - y^2$, amb $0 \leq y < x$. La bijecció és donada per les fórmules $x = \frac{a+b}{2}$, $y = \frac{a-b}{2}$, $a = x + y$, $b = x - y$.

A partir d'aquí, l'algoritme consisteix a avaluar successivament la diferència $r := x^2 - y^2 - N$, a partir de valors inicials adequats, i augmentar y o x segons que r sigui positiu o negatiu, fins a obtenir $r = 0$. De fet, es poden calcular fàcilment els increments de r a partir dels increments de x i de y, amb la utilització només de sumes i restes, fet que fa l'algoritme molt ràpid. Si no posem límit al nombre de proves, podrem factoritzar el nombre en tants passos (essencialment) com la diferència entre els dos factors més propers, l'un, per sota i l'altre, per sobre, de l'arrel quadrada de a·b.

Observació. Notem que si b=1 (o sigui, si n=a és primer), llavors el nombre de passos és, essencialment, a, de manera que si el nombre és primer, haurem de fer tants passos com el nombre; i això, a la pràctica, és molt pitjor que intentar factoritzar per divisió fins a l'arrel quadrada...

Per tant, cal posar un límit al nombre de passos que estem disposats a fer; o sigui, a la diferència a-b.

Podeu veure els detalls en **[Tr-Ar,** cap. VII, secció 2**]**.

6.2. La funció

Implementem l'algoritme.

In [1]:
```
def FermatFact(nn,ff):
    if not nn in ZZ:
        return 'Cal que el nombre sigui enter.'
    if nn==0:
        return [0]
    if nn==1:
        return [1]
    if nn==-1:
        return [-1]
    if nn<0:
        n=-nn
        signe=-1
    else:
        n=nn
        signe=1
    if n==2 or n==3 or n==5 or n==7:
        return [nn]
    if is_even(n):
        return [signe*2, n//2]
    if ff<1:
        return 'Cal fer alguna prova.'
    f=2*(floor(ff)+1)
    an=floor(sqrt(n))
    if n==an^2:
        return [signe*an,an]
    v=1
    u=2*(an+1)+1
    r=(an+1)^2-n
    while r>0 and v<f:
        r=r-v
        v=v+2
    while r<0:
        r=r+u
        u=u+2
        while r>0 and v<f:
            r=r-v
            v=v+2
    if v<f:
        return [signe*(u-v)//2,(u+v-2)//2]
    return [nn,'fita petita']

```

6.3. Exemples

6.3.0. Trivials

In [2]:
```
FermatFact(0,5)
```

Out[2]: [0]

In [3]:
```
FermatFact(1,5)
```

Out[3]: [1]

In [4]:
```
1 FermatFact(-1,5)
```

Out[4]: [-1]

In [5]:
```
1 FermatFact(2,5)
```

Out[5]: [2]

In [6]:
```
1 FermatFact(-2,5)
```

Out[6]: [-2]

In [7]:
```
1 FermatFact(3,5)
```

Out[7]: [3]

In [8]:
```
1 FermatFact(-3,5)
```

Out[8]: [-3]

In [9]:
```
1 FermatFact(4,5)
```

Out[9]: [2, 2]

In [10]:
```
1 FermatFact(-4,5)
```

Out[10]: [-2, 2]

In [11]:
```
1 FermatFact(5,5)
```

Out[11]: [5]

In [12]:
```
1 FermatFact(-5,5)
```

Out[12]: [-5]

In [13]:
```
1 FermatFact(6,5)
```

Out[13]: [2, 3]

In [14]:
```
1 FermatFact(-6,5)
```

Out[14]: [-2, 3]

In [15]:
```
1 FermatFact(7,5)
```

Out[15]: [7]

In [16]:
```
1 FermatFact(-7,5)
```

Out[16]: [-7]

In [17]:
```
1 FermatFact(8,5)
```

Out[17]: [2, 4]

In [18]:
```
1 FermatFact(-8,5)
```

Out[18]:
```
[-2, 4]
```

In [19]:
```
1 FermatFact(9,5)
```

Out[19]:
```
[3, 3]
```

In [20]:
```
1 FermatFact(-9,5)
```

Out[20]:
```
[-3, 3]
```

In [21]:
```
1 FermatFact(11,5)
```

Out[21]:
```
[1, 11]
```

In [22]:
```
1 FermatFact(13,5)
```

Out[22]:
```
[13, 'fita petita']
```

In [23]:
```
1 FermatFact(13,6)
```

Out[23]:
```
[1, 13]
```

6.3.1. Senzills

In [24]:
```
1 FermatFact(32947562394567*32947562394571, 15)
```

Out[24]:
```
[32947562394567, 32947562394571]
```

In [25]:
```
1 p=random_prime(10^50)
```

In [26]:
```
1 p
```

Out[26]:
```
85011264563285560195603215279327990782339394022717
```

In [27]:
```
1 q=next_prime(p)
```

In [28]:
```
1 q
```

Out[28]:
```
85011264563285560195603215279327990782339394022811
```

In [29]:
```
1 FermatFact(p*q,1000)
```

Out[29]:
```
[85011264563285560195603215279327990782339394022717,
 85011264563285560195603215279327990782339394022811]
```

In [30]:
```
1 q-p
```

Out[30]:
```
94
```

In [31]:
```
1 p*q
```

Out[31]:
```
72269151026489312476432554715941044850683067763593233227945727941875246449682921686255839305501974 87
```

In [32]:
```
1 %time FermatFact(p*q,1000)
```

```
CPU times: user 2.43 s, sys: 15.9 ms, total: 2.45 s
Wall time: 2.01 s
```

Out[32]:
```
[85011264563285560195603215279327990782339394022717,
 85011264563285560195603215279327990782339394022811]
```

In [33]:
```
1 %time FermatFact(p*q,100)
```

```
CPU times: user 2.27 s, sys: 26.4 ms, total: 2.29 s
Wall time: 1.84 s
```

Out[33]:
```
[85011264563285560195603215279327990782339394022717,
 85011264563285560195603215279327990782339394022811]
```

6.3.2. Més complicats

In [34]:
```
1 n=1+2*ZZ.random_element(10^500)
```

In [35]:
```
1 m=n+2*ZZ.random_element(1000)
```

In [36]:
```
1 %time FermatFact(m*n,10000)
```

```
CPU times: user 9.76 s, sys: 87.7 ms, total: 9.84 s
Wall time: 5.63 s
```

Out[36]:
```
[10854365488838075497016601521853975522919298243761917609905438029
0831288609744495793717087712833921068188742717991439116233053814523
1829660421437757460856521787101967353374037103655041879043342379654
0877057518173568998636664365162877996440554761281679920543482803470
1917514881325223710053113554523076133891221366168515396843490738774
6601749416559509696930108194217819487091160314736052968523011631454
8553121740537968719769788503270521453413267972611514015640120500218
8753591914840997616845009740418 05,
 10854365488838075497016601521853975522919298243761917609905438029
0831288609744495793717087712833921068188742717991439116233053814523
1829660421437757460856521787101967353374037103655041879043342379654
0877057518173568998636664365162877996440554761281679920543482803470
1917514881325223710053113554523076133891221366168515396843490738774
6601749416559509696930108194217819487091160314736052968523011631454
8553121740537968719769788503270521453413267972611514015640120500218
87535919148409976168450097404 1957]
```

A fi de no incorporar els tests de primeritat, farem servir la funció is_primer(n) del *SageMath*, però podríem usar igualment la funció SolovayStrassenTest(n,20).

In [37]:
```
1 is_prime(n)
```

Out[37]:
```
False
```

In [38]:
```
1 is_prime(m)
```

Out[38]:
```
False
```

In [39]:
```
1 m-n
```

Out[39]: 152

6.4. Observació

Per als valors

p=130618917572945862433731712064533144408005740745 83,

q=130618917572945862433731712064533144408005740747 17,

que són tals que q-p=134, he executat la comanda

%time factor(p*q)

Després de més d'una hora, l'he aturat sense que ho hagi fet!!!

En canvi, amb la funció FermatFact, ha trigat menys de 3 segons a factoritzar!

Òbviament, això és perquè els nombres són molt especials. A més a més, probablement, el *SageMath* no té implementada aquesta funció, o, almenys, no de manera automàtica.

Fi del capítol 6