Primeritat i factorització

Artur Travesa

(versió 2024-07)

Capítol 2: Certificats de primeritat

2.0. Introducció

Continuem amb un altre fragment de l'article 329 de les *Disquisicions Aritmètiques*, de C.F. Gauss, en la versió de la Dra. Griselda Pascual Xufré, editada per l'IEC, el 1996 (cf. **[Ga-DA]**).

"*[...] està fonamentat en la naturalesa del problema que* qualssevol *mètodes sortiran contínuament més prolixos com més grans són els nombres als quals s'apliquin; [...]"*

2.1. Un certificat bàsic

2.1.0. El fonament teòric

Per al càlcul d'arrels primitives mòdul un nombre primer p, es pot utilitzar el resultat següent, que assegura que g és un element invertible mòdul p, d'ordre exactament p-1.

Sigui p>3 un nombre enter senar. El nombre p és primer si, i només si, existeix un nombre enter g, $2 \leq g \leq p-2$*, tal que* $g^{p-1} \equiv 1 \pmod{p}$ *i que per a tot nombre primer q que divideixi p-1 sigui* $g^{\frac{p-1}{q}} \not\equiv 1 \pmod{p}$. (Això és, un element d'ordre exactament p-1.)

En aquest cas, la successió (p, True, g, {q: q és primer i q divideix p-1}) es podria considerar un certificat de primeritat per a p, perquè permet comprovar que, efectivament, p és primer, si la llista és la dels divisors primers de p-1.

Se'n pot veure una demostració en **[Tr-Ar**, cap. VII, punt 1.3**]**.

2.1.1. Observació

D'altra banda, si p és primer, la probabilitat que un element g, $2 \leq g \leq p-2$ triat a l'atzar i amb probabilitat equirepartida, sigui una arrel primitiva mòdul p és més gran que el quocient $\dfrac{\varphi(p-1)}{p-1} = \prod_{q \mid p-1} \left(1 - \dfrac{1}{q}\right)$, amb el producte estès al conjunt dels nombres naturals primers q que divideixen p-1. I aquesta probabilitat només és molt petita si p-1 és divisible per molts nombres primers diferents (cf. la discussió en **[Tr-Ar**, cap. VII, punts 1.6,

1.7**]**).

Notem que aquesta probabilitat no depèn de si sabem calcular, o no, la llista dels divisors primers de p-1. Això serà útil més avall.

Aquest fet fa possible dissenyar un algoritme probabilístic per a resoldre la tasca de crear certificats de nombres primers p per als quals es conegui la llista dels divisors primers de p-1. En efecte, la recerca a l'atzar d'un element g que satisfaci les propietats que volem és, probablement, senzilla.

A més a més, el factor 1/2 que apareix per al divisor primer 2 de p-1 es pot detectar directament en el primer pas de l'algoritme, de manera que, essencialment, l'algoritme, després del primer pas, té probabilitat doble de reeixir.

2.1.2. Definició

Un *certificat de composició* és una llista (n, False, g), on n és un nombre natural compost i g és un nombre enter entre 1 i n-1 tal que $g^{n-1} \not\equiv 1 \pmod{n}$.

Efectivament, un tal valor de g certifica que n és compost.

Donarem més avall la definició de *certificat de primeritat*.

2.2. La funció Certifica(pp,fppmu,ff)

Es tracta de deinir una funció per a certificar la primeritat d'un nombre natural primer no nul. Aquesta funció només es podrà aplicar a nombres primers per als quals coneguem molta informació complementària, i més avall generalitzarem el certificat de manera que la nova funció es pugui aplicar a nombres amb no tantes restriccions.

2.2.0. Les dades

La funció **Certifica(pp,fppmu,ff)** es podrà aplicar a un nombre enter pp>0, una fita ff>0 per al nombre de proves a l'atzar, i una llista fppmu de nombres naturals.

• Aquí, pp és el nombre que es vol certificar que és primer. Ha de ser un nombre natural, i hauria d'haver passat un test de primeritat, de manera que si fos compost, no intentéssim certificar que és primer. (Encara que això no és necessari per a usar la funció, sí que és molt convenient, perquè si pp és compost, segur que no podrem certificar que és primer!)

• A més a més, fppmu ha de ser la llista de tots els nombres primers que divideixen el nombre pp-1 (**f**actors **p**rimers de **p m**enys **u**). En el cas que no es disposi d'aquesta llista (o sigui, que es proporcioni la llista buida, fppmu=[ ]), la funció Certifica(pp,fppmu,ff) intentarà calcular-la, però això fa que pugui no acabar el càlcul en temps "raonable" (hores?, dies?, setmanes?, mesos?, anys?, segles?, ...).

• **ATENCIÓ!** En cas que la funció disposi d'aquesta llista, *suposarà* que és la llista de tots els divisors primers de pp-1; per tant, cal que la llista sigui efectivament la llista dels divisors primers de pp-1; en cas contrari, el retorn de la funció pot ser erroni i el certificat,

fals.

2.2.1. El retorn

En cas que la funció disposi d'aquesta llista correcta (o la pugui calcular), la funció retorna:

• [pp, True, g, lta], si pp és primer, on g és una arrel primitiva mòdul pp i lta és la llista ordenada en sentit creixent dels factors primers de pp-1;

• [pp, False, 1], si pp=1;

• [pp, False, 2], si pp és parell, compost;

• [pp, False, g], si pp és senar i compost, on 1<g<pp-1 és un nombre d'ordre no divisor de pp-1; o bé

• 'Indeterninat', si no ha pogut certificar que pp és primer amb, com a màxim, ff elements g triats a l'atzar.

Observació. Notem, doncs, que la funció també pot retornar un certificat de composició.

2.2.2. La funció

In [1]:
```
def Certifica(pp,fppmu,ff):
    if pp==1:
        return [pp,false,1]
    if pp==2 or pp==3:
        return [pp,true,pp-1,[pp-1]]
    if is_even(pp):
        return [pp,false,2]
    if ff<1:
        return ["Cal fer alguna prova."]
    if len(fppmu)==0:
        lta1=factor(pp-1)
        lta=[lta1[i][0] for i in range(len(lta1))]
    else:
        lta=sorted(fppmu)
    l=len(lta)
    f=0
    while f<ff:
        g=ZZ.random_element(2,pp-2)
        if (s:=Mod(g,pp)^((pp-1)//2))==pp-1:
            i=1
            while i<= l-1 and Mod(g,pp)^((pp-1)//lta[i])!=1:
                i=i+1
            if i==l:
                return [pp,true,g,lta]
        else:
            if s!=1:
                return [pp,false,g]
        f=f+1
    return [pp,'Indeterminat']
```

2.2.3. Proves

Nombres compostos, sense llista, o amb llista (òbviament, incorrecta):

```
In [2]: Certifica(1995467,[],2)
```

```
Out[2]: [1995467, False, 343662]
```

```
In [3]: factor(1995467-1)
```

```
Out[3]: 2 * 11 * 90703
```

```
In [4]: Certifica(3*7*13,[],5)
```

```
Out[4]: [273, False, 62]
```

```
In [5]: Certifica(3*7*13,[2,3],5)
```

```
Out[5]: [273, False, 58]
```

```
In [6]: factor(3*7*13-1)
```

```
Out[6]: 2^4 * 17
```

```
In [7]: Certifica(3*7*13,[2,17],5)
```

```
Out[7]: [273, False, 166]
```

Poques proves?

```
In [8]: Certifica(90703,[],1)
```

```
Out[8]: [90703, 'Indeterminat']
```

```
In [9]: Certifica(90703,[],2)
```

```
Out[9]: [90703, True, 34781, [2, 3, 5039]]
```

```
In [10]: Certifica(90703,[],2)
```

```
Out[10]: [90703, True, 37304, [2, 3, 5039]]
```

Certifica sense llista. Notem que factoritzar (n!+1)-1 és molt senzill, per divisió.

```
In [11]: Certifica(90703,[],5)
```

```
Out[11]: [90703, True, 10328, [2, 3, 5039]]
```

```
In [12]: Certifica(factorial(27)+1,[],10)
```

```
Out[12]: [10888869450418352160768000001,
          True,
          7783394867997569949923072301,
          [2, 3, 5, 7, 11, 13, 17, 19, 23]]
```

```
In [13]:  1  Certifica(factorial(37)+1,[],10)
```

```
Out[13]: [13763753091226345046315979581580902400000001, 'Indeterminat']
```

I certifica nombres prou grans sense llista:

```
In [14]:  1  Certifica(219825146244531300827618434834380439599661150333525461
```

```
Out[14]: [219825146244531300827618434834380439599661150333525461306467196762
         698338736604216421361,
          True,
          144864957102304393308843885866673755661145337492371774063940864141
         948479690476195293987,
          [2,
           5,
           311,
           313,
           1896019,
           2679283,
           26117587,
           212759192599164428602949807972793547430520108198963916077031]]
```

Si es proporciona una llista errònia, el resultat pot ser correcte o no.

```
In [15]:  1  factor(229-1)
```

```
Out[15]: 2^2 * 3 * 19
```

```
In [16]:  1  Certifica(229,[2,19],5)
```

```
Out[16]: [229, True, 195, [2, 19]]
```

```
In [17]:  1  Certifica(229,[2,4,19],3)
```

```
Out[17]: [229, True, 155, [2, 4, 19]]
```

```
In [18]:  1  Certifica(229,[2,4,17],3)
```

```
Out[18]: [229, True, 115, [2, 4, 17]]
```

En alguna prova, la funció ha proporcionat 84 com a valor de g; però:

```
In [19]:  1  Mod(84,229)^(228/3)
```

```
Out[19]: 1
```

Notem que els nombres compostos es detecten bastant bé, sense llista.

```
In [20]:  1  Certifica((factorial(27)+1)*(factorial(37)+1),[],5)
```

```
Out[20]: [149871710558155707455980685999557247034058974987628747043133063168
         000001,
          False,
          134422904520598198160656694270685063409088712184963224459610542741
         360638]
```

In [21]:
```
1 Certifica((factorial(27)+1)*(factorial(37)+1),[],5)
```

Out[21]:
```
[14987171055815570745598068599955724703405897498762874704313306316800000 1,
 False,
 3912979799707421810307756907203157146078459017195697029876785773210565]
```

In [22]:
```
1 Certifica((factorial(27)+1)*(factorial(37)+1),[],5)
```

Out[22]:
```
[14987171055815570745598068599955724703405897498762874704313306316800000 1,
 False,
 5711011056001043187355932125544056093264777686573643266090893138504052 5]
```

2.2.4. Un exemple no trivial

Les llistes de primers que es proporcionen han estat construïdes a partir de primers certificats. I els primers, han estat construïts a partir de les llistes de primers. Això permet certificar-los fàcilment (perquè les llistes són bones), i no cal fer gaire proves.

In [23]:
```
1 q1=213301290870756148157681949376793849934023192047284579482211 9(
```

In [24]:
```
1 q2=160435421908126022771978733679599623714565815438642056043113 38
```

In [25]:
```
1 ltaq1=[2, 7, 103, 433889, 263795224239548516807553360364750 7, 30
2 350811628298740339512859664364897 1, 43285623664976199866617093529
```

In [26]:
```
1 ltaq2=[2, 19913, 66763, 26299997615281926533677951166058 43, 2917 8
2 383086575260705284437136091876145 7, 513131373235869678004648744 21
```

In [27]:
```
1 Certifica(q1,ltaq1,5)
```

Out[27]:
```
[213301290870756148157681949376793849934023192047284579482211902406587469483732439405687857078447451959422365100846598108928350487337197871461477 7,
 True,
 6029982134954162165128530265498710850344336468675576255534882417628527722035871632351650476829384760260259424257038777030695190308252822750759 90,
 [2,
  7,
  103,
  433889,
  263795224239548516807553360364750 7,
  303953370473077444397374012886255 9,
  350811628298740339512859664364897 1,
  432856236649761998666170935298498 3]]
```

In [28]:
```
1 Certifica(q2,ltaq2,5)
```

Out[28]:
```
[160435421908126022771978733679599623714565815438642056043113385296
939906267598800342640388971028995466673342267929043233332906023329 4
859805838217,
 'Indeterminat']
```

2.3. El certificat de Pocklington

2.3.0. La base teòrica

La dificultat més gran del certificat bàsic anterior és que per a provar la primeritat de pp necessita de la factorització de pp-1. El teorema següent (de H.C. Pocklington) permet relaxar aquesta condició i considerar una factorització parcial prou bona.

Teorema: *Sigui $N > 3$ un nombre enter senar. Llavors, el nombre N és primer si, i només si, existeix una descomposició $N - 1 = T \cdot U$, amb $T, U \geq 1$, enters, per als quals se satisfan les condicions següents: (a) $\gcd(T, U) = 1$, (b) $U < \sqrt{N}$, i (c) per a tot divisor primer p de T existeix un nombre enter g tal que $g^{N-1} \equiv 1 \pmod{N}$ i $\gcd(g^{\frac{N-1}{p}} - 1, N) = 1$.*

Se'n pot veure una demostració en **[Tr-Ar**, cap. VII, punt 2.5**]**.

Notem que per a l'aplicació del teorema no cal la factorització completa de N-1; només cal conèixer la llista de primers que divideixen una part prou gran de la factorització.

Notem també que les propietats anteriors se satisfan per a una arrel primitiva mòdul N, g; per tant, és fàcil que es pugui prendre el mateix valor de g per a tots els divisors primers p de T. Però tampoc no cal que g sigui una arrel primitiva mòdul N, perquè per a un element g que sigui una potència d'exponent divisor de U d'una arrel primitiva, també se satisfan totes les propietats de la proposició i no és una arrel primitiva mòdul N.

Això permet donar una definició de certificat de primeritat que, tot i ser més restrictiva que no caldria, fa que sigui prou senzill obtenir-ne un i la seva comprovació.

Definició. Anomenarem *certificat de primeritat* d'un nombre natural (primer) $N > 1$ una successió $[N, \text{True}, g, \texttt{lta}]$ tal que $N - 1$ es pugui escriure en la forma $N - 1 = T \cdot U$, amb $\gcd(N, T) = 1$, $U^2 < N$, `lta` sigui la llista dels nombres primers divisors de T, i g sigui un nombre natural per al qual se satisfà que $g^{N-1} \equiv 1 \pmod{N}$ i que per a tot element p de la llista `lta` és $\gcd(g^{\frac{N-1}{p}} - 1, N) = 1$.

Notem que si `lta` és la llista de tots els divisors primers de $N - 1$, llavors $U = 1$, i el certificat de primeritat és exactament del mateix tipus que el certificat que s'obté amb l'aplicació del criteri bàsic.

En el cas particular en què N-1 sigui el producte d'un nombre primer p més gran que l'arrel quadrada de N i altres factors (que no cal conèixer), podem pensar en T=p com la part factoritzada i U=(N-1)/p com la resta. I només cal fer una prova (per a p).

Observació. En qualsevol cas, com que 2 és primer (i un certificat és [2,true,1,[1]]), només cal preocupar-se de certificar nombres primers senars. En particular, per a calcular el certificat, sempre podem afegir el primer 2 a la llista, si no hi és, de manera que això encara fa més petit el valor de U. A canvi, caldrà fer una comprovació més (la que correspon al primer 2). Però això farà més senzilla la resta!

En efecte, com que N és un primer senar, si g és una arrel primitiva mòdul N, l'ordre de g és parell i $g^{\frac{N-1}{2}} \equiv -1 \pmod{N}$. Així, la comprovació del valor d'aquesta potència justifica d'una banda que $g^{N-1} \equiv 1 \pmod{N}$, propietat que havíem de comprovar, i, de l'altra, que $\gcd(g^{\frac{N-1}{2}} - 1, N) = \gcd(-2, N) = 1$; per tant, no afegeix feina a les comprovacions que cal fer.

Però si un element g és un quadrat mòdul N (i N és primer, com volem provar), la potència proporciona $1 \pmod{N}$, i no $-1 \pmod{N}$. I això exclou la meitat dels elements g mòdul N, de manera que aquesta sola comprovació fa que la probabilitat d'encertar una arrel primitiva (a partir d'aquí) es multipliqui per 2.

2.3.1. La funció Pocklington(pp,tt,ff)

Es tracta, doncs, de definir una funció **Pocklington(pp,tt,ff)** que s'apliqui a un nombre (primer que volem certificar) pp, una llista tt de factors primers de pp-1, i una fita ff per al nombre màxim de valors de g que provarem a l'atzar, i que proporcioni un certificat de primeritat per a pp, o bé 'Indeterminat' si en ff proves no l'aconsegueix, o un certificat de composició si pp és compost.

In [29]:

```
def Pocklington(pp,tt,ff):
    if not pp in ZZ or pp<1:
        return ['Cal que el nombre P sigui enter positiiu.']
    if pp==1:
        return [pp,false,1]
    if pp==2 or pp==3:
        return [pp,true,pp-1,[pp-1]]
    if is_even(pp):
        return [pp,false,2]
    if ff<1:
        return 'Cal fer alguna prova.'
# Comprovació que la llista tt és de divisors de pp-1, i càlcul d
# però no que són primers.
    if false in [(r in ZZ and r>1) for r in tt]:
        return 'La llista T no és de nombres enters >1.'
# Si 2 no pertany a la llista tt, li afegim (per a millora del cà
    t=tt
    if not (2 in t):
        t=[2]+t
    x=prod(t)
    q,r=divmod(pp-1,x)
    if r:
        return 'La llista T no és correcta.'
    d=gcd(q,x)
    while d>1:
        q=q//d
        d=gcd(q,x)
    uu=q
    q=uu^2
    if q==pp:
        return [pp,false,uu]
    if q>pp:
        return 'U és massa gran.'
    t=sorted(t)
# Si hem arribat aquí, és que P, T, F i U són correctes (excepte
# potser, que alguns elements de T no siguin primers).
    l=len(t)
    f=0
    while f<ff:
        g=ZZ.random_element(2,pp-2)
        if (s:=Mod(g,pp)^((pp-1)//2))==pp-1:
            i=1
            while i<= l-1 and gcd((s:=Mod(g,pp)^((pp-1)//t[i]))-1
                i=i+1
            if i==l:
                return [pp,true,g,t]
        else:
            if s!=1:
                return [pp,false,g]
        f=f+1
    return [pp,'Indeterminat']

```

2.3.2. Proves

In [30]:
```
1 Pocklington(103,[17],5)
```

Out[30]:
```
[103, True, 74, [2, 17]]
```

In [31]:
```
1 Pocklington((factorial(27)+1)*(factorial(37)+1),[],5)
```

Out[31]:
```
'U és massa gran.'
```

In [32]:
```
1 Pocklington(q1,ltaq1,5)
```

Out[32]:
```
[213301290870756148157681949376793849934023192047284579482211902406
587469483732439405687857078447451959422365100846598108928350487337 1
978714614777,
 True,
 260409316044678262760838360536651314506028866369961581312411722092
962600364695096463862190372837947122026341474598366727992984193083 5
37956394551,
 [2,
  7,
  103,
  433889,
  263795224239548516807553360364750 7,
  303953370473077444397374012886255 9,
  350811628298740339512859664364897 1,
  432856236649761998666170935298498 3]]
```

In [33]:
```
1 Pocklington(q2,ltaq2,5)
```

Out[33]:
```
[160435421908126022771978733679599623714565815438642056043113385296
939906267598800342640388971028995466673342267929043233332906023329 4
859805838217,
 True,
 148292964794993216033865593942857549736561122211935031619859255717
365961076286519545175784945241083674341935645746335322771321064492 5
965012764991,
 [2,
  19913,
  66763,
  262999976152819265336779511660584 3,
  291780675785049631015070136768080 3,
  383086575260705284437136091876145 7,
  513131373235869678004648744219021 1]]
```

Traiem els primer i els dos darrers nombres primers de la llista. Enccara certifica. I notem que afegeix el 2.

In [34]:
```
1 Pocklington(q1,ltaq1[1:-2],5)
```

Out[34]:
```
[213301290870756148157681949376793849934023192047284579482211902406
587469483732439405687857078447451959422365100846598108928350487337 1
978714614777,
 'Indeterminat']
```

In [35]:
```
1 Pocklington(q2,ltaq2[1:-2],5)
```

Out[35]:
```
[160435421908126022771978733679599623714565815438642056043113385296
939906267598800342640388971028995466673342267929043233332906023329 4
859805838217,
 True,
 132287367046212506817571997350988238050233633437463486173293061696
738782182116272653020777263251852723880474858191707303758734383047 8
869587978069,
 [2,
  19913,
  66763,
  262999976152819265336779511660584 3,
  291780675785049631015070136768080 3]]
```

Traiem els tres primers i els dos darrers nombres primers de la llista. En un cas encara certifica; en l'altre, no. I notem que afegeix el 2.

In [36]:
```
1 Pocklington(q1,ltaq1[3:-2],5)
```

Out[36]:
```
[213301290870756148157681949376793849934023192047284579482211902406
587469483732439405687857078447451959422365100846598108928350487337 1
978714614777,
 True,
 162643180269708833020324859638843315093717633363525054538793404191
620384626309597051659407371878220794621800783422734683454764506870 7
798678671725,
 [2,
  433889,
  263795224239548516807553360364750 7,
  303953370473077444397374012886255 9]]
```

In [37]:
```
1 Pocklington(q2,ltaq2[2:-2],5)
```

Out[37]:
```
[160435421908126022771978733679599623714565815438642056043113385296
939906267598800342640388971028995466673342267929043233332906023329 4
859805838217,
 True,
 888661756089016436655163811981292486518149683024830043244972796010
415131453139079889624598494790339370162677220588111494170878854045 3
84268809173,
 [2,
  66763,
  262999976152819265336779511660584 3,
  291780675785049631015070136768080 3]]
```

In [38]:
```
1 Pocklington(q2,ltaq2[3:-2],5)
```

Out[38]:
```
'U és massa gran.'
```

2.4. Comprovació de certificats

2.4.0. Certificats de composició

La comprovació d'un certificat de composició és immediata. En efecte, donada una llista [n,

False, g] que, pretesament, és un certificat que n és compost, és suficient fer dues coses. En primer lloc, comprovar que 1<g<n, cosa que és immediata. Ara, si el càlcul $g^{n-1} \pmod{n}$ no proporciona 1, llavors o bé g no és invertible mòdul n, o bé l'ordre de g mòdul n no és un divisor de n-1, com hauria de ser si n fos primer. Per tant, efectivament, n és compost.

2.4.1. Certificats de primeritat

En canvi, la comprovació d'un certificat de primeritat [n,true,g,tt] com l'hem definit presenta una dificultat.

En efecte. Suposem que la llista tt és de nombres primers. En aquest cas, la comprovació del certificat és senzilla. En efecte, en primer lloc, es pot comprovar que n-1 és un producte, T, de nombres de la llista (amb repeticions, si cal) i un nombre U coprimer amb T i menor que l'arrel quadrada de n, i que tots els nombres p de la llista divideixen n-1. I, a continuació, també és senzill comprovar les propietats demanades per a g, perquè és senzill calcular $\gcd(g^{\frac{n-1}{p}} - 1, n)$ per a cada element p de la llista, i $g^{n-1} \pmod{n}$ i veure que tots són 1.

Ara bé, la dificultat rau en el fet que *cal* comprovar que els nombres p de la llista tt són primers.

Per tant, per a fer una funció que comprovi efectivament que el certificat és correcte, hauríem d'afegir, al certificat de n, certificats dels factors primers de n-1, certificats dels factors primers dels factors primers de p-1 per a cada factor primer p de n-1, i així recursivament; i, després, comprovar, també recursivament tots aquests certificats.

De la mateixa manera que per a crear el certificat ho fem a partir d'una funció que *suposa* que la llista dels divisors primers de n-1 és correcta, si suposem això mateix per a la funció de comprovació dels certificats, la programació és molt més planera.

Certament, només podrem dir que el certificat és bo si la funció dóna com a bo el certificat i, a més a més, aquesta llista és de nombres *primers* divisors de n-1.

2.4.2. La funció

La funció **ComprovaCert(cert)** comprova que la llista cert és o bé un certificat ce composició, o bé un certificat de primeritat, donant per fet que els nombres de la llista lta que apareix a la llista cert, si és el cas, són primers.

In [39]:
```
def ComprovaCert(cert):
    pp=cert[0]
    if not pp in ZZ or pp<1:
        return 'S\'ha d\'aplicar a un nombre enter positiu.'
    if pp==1:
        return cert==[pp,false,1]
    if pp==2 or pp==3:
        return cert==[pp,true,pp-1,[pp-1]]
    if is_even(pp):
        return cert==[pp,false,2]
    if cert[1]==false:
        return 1<cert[2]<pp and mod(cert[2],pp)^(pp-1)!=1
    if cert[1]=='Indeterminat' or cert[1]=='indeterminat':
        return'Indeterminat'
    if cert[1]!=true:
        return 'La llista no és cap certificat.'
# Sembla que la llista pot ser un certificat de primeritat de P.
# Comprovem que la llista ho és de nombres naturals entre 1 i n-
# i divisors de n-1.
    tt=cert[3]
    if false in [v in ZZ and 1<v and v<pp and ((pp-1)%v)==0 for v
        return 'La llista T és incorrecta.'
# Suposem que els elements de la llista són primers.
# Comprovem que g és un natural entre 1 i n-1.
    g=cert[2]
    if not g in ZZ or g<2 or g>pp-1:
        return 'La llista no és cap certificat: g incorrecte.'
# Calculem el cofactor U.
    x=prod(tt)
    q,r=divmod(pp-1,x)
    if r:
        return 'La llista T no és correcta.'
    d=gcd(q,x)
    while d>1:
        q=q//d
        d=gcd(q,x)
    uu=q
    q=uu^2
    if q==pp:
        return 'P és un quadrat.'
    if q>pp:
        return 'U és massa gran.'
# Comprovem les propietats de g.
    if Mod(g,pp)^(pp-1)!=1:
        return False, 'L\'ordre de g no divideix p-1.'
    if false in [gcd(Mod(g,pp)^((pp-1)//tt[v])-1,pp)==1 for v in
        return False, 'Algun g és incorrecte.'
    return true

```

2.4.3. Proves amb Certifica

In [40]:
```
c1=Certifica(229,[2,3,19],5)
```

In [41]:
```
1 print(c1)
```

```
[229, True, 162, [2, 3, 19]]
```

In [42]:
```
1 ComprovaCert(c1)
```

Out[42]: True

In [43]:
```
1 c2=Certifica(229,[2,3,17],10)
```

In [44]:
```
1 print(c2)
```

```
[229, True, 137, [2, 3, 17]]
```

In [45]:
```
1 ComprovaCert(c2)
```

Out[45]: 'La llista T és incorrecta.'

In [46]:
```
1 ComprovaCert(Certifica(q1,ltaq1,5))
```

Out[46]: True

2.4.4. Proves amb Pocklington

In [47]:
```
1 ComprovaCert(Pocklington(q1,ltaq1,5))
```

Out[47]: True

In [48]:
```
1 Pocklington(q1,ltaq1[0:-1],5)
```

Out[48]:
```
[213301290870756148157681949376793849934023192047284579482211902406
587469483732439405687857078447451959422365100846598108928350487337 1
978714614777,
 True,
 136025107850475047883021820548087598430461405021474807871074926414
371340038525822804473340298364125801213170679532157180876137156452 9
777646364548,
 [2,
  7,
  103,
  433889,
  263795224239548516807553360364750 7,
  303953370473077444397374012886255 9,
  350811628298740339512859664364897 1]]
```

In [49]:
```
1 Pocklington(q1,ltaq1[0:-2],5)
```

Out[49]:
```
[213301290870756148157681949376793849934023192047284579482211902406
587469483732439405687857078447451959422365100846598108928350487337 1
978714614777,
 True,
 391027496615282703039787317677444228089369371349686177510835350255
740108890064406713489440007762166980757776286789704755583972309594 2
1120917997,
 [2,
  7,
  103,
  433889,
  263795224239548516807553360364750 7,
  303953370473077444397374012886255 9]]
```

In [50]:
```
1 ComprovaCert(Pocklington(q1,ltaq1[0:-1],5))
```

Out[50]:
```
True
```

In [51]:
```
1 ComprovaCert(Pocklington(q1,ltaq1[0:-2],5))
```

Out[51]:
```
True
```

In [52]:
```
1 Pocklington(q2,ltaq1[0:-2],5)
```

Out[52]:
```
'La llista T no és correcta.'
```

In [53]:
```
1 Pocklington(q2,ltaq2[0:-2],5)
```

Out[53]:
```
[160435421908126022771978733679599623714565815438642056043113385296
939906267598800342640388971028995466673342267929043233332906023329 4
859805838217,
 True,
 138318769430808966386816214174724017038143741761498768168340816721
544560543666863501400708421672242047028604761523773782665708298632 2
235024104365,
 [2,
  19913,
  66763,
  262999976152819265336779511660584 3,
  291780675785049631015070136768080 3]]
```

In [54]:
```
1 ComprovaCert(Pocklington(q2,ltaq2[0:-2],5))
```

Out[54]:
```
True
```

Fi del capítol 2