Factorització

Artur Travesa

(versió 2021-04)

Introducció general

L'origen d'aquestes notes es remunta a diferents cursos d'Aritmètica o de Criptografia, a càrrec de l'autor, per a estudiants de Matemàtiques o d'Informàtica de la Universitat de Barcelona.

La idea bàsica és descriure algunes aplicacions importants de l'Aritmètica bàsica (diguem, de nivell de primer curs) a les transmissions xifrades d'informació.

En cap cas es tracta d'un curs de Criptografia, que caldria encabir en espais més amplis de coneixements, que haurien d'incloure, probablement, parts de teoria de la comunicació, de complexitat algorítmica o computacional, d'aprenentatge automàtic, o d'estudi de teories de compartició de secrets, entre d'altres.

El format triat per a la presentació és el d'un *notebook* de *Mathematica*, per la facilitat que té aquest programari per a poder desenvolupar els càlculs no trivials de manera prou senzilla i entenedora, d'una banda, i per a permetre fer una presentació escrita prou raonable des del punt de vista de material escrit, de l'altra. En particular, la possibilitat d'incloure els càlculs dins del text de manera natural en fan una bona eina comunicativa i, alhora, facilita molt el càlcul amb exemples no trivials.

A fi de veure tot el contingut del *notebook* convé executar-lo. Això es pot fer de cop o bé, més recomanable, cel·la a cel·la a mesura que s'avança en la lectura i comprensió dels diferents continguts.

Observació: Per al cas en què no es disposi del programari, hi ha una versió executada del *notebook* en format pdf.

Amb la finalitat doble d'una banda, de no fer textos molt llargs o amb molts continguts, i de l'altra de poder ampliar de manera senzilla els continguts que s'hi tractin, el material s'ha dividit en diferents *notebooks*, que desrivim a continuació, en l'apartat de referències.

Referències

[Cripto- 1]: **Criptografia bàsica (1)**.
Travesa, A.:CriptografiaBasica-1 ; accessible en forma notebook o en format pdf des de https://travesa.cat/notes/
Contingut: Una iniciació a la codificació de missatges. El criptosistema de Cèsar. El criptosistema de Vigenère.

[Cripto- 2]: **Criptografia bàsica (2)**.
Travesa,CriptografiaBasica-2; accessible en forma notebook o en format pdf des de https://travesa.cat/notes/
Contingut: Els criptosistemes lineals. Els criptosistemes afins. Sufixació de missatges. Farciment de missatges.

[Eratostenes]: **Un garbell d'Eratòstenes.**
Travesa, A.: Eratostenes; accessible en forma notebook o en format pdf des de https://travesa.cat/notes/
Contingut: Un garbell d'Eratòstenes.

[Cripto- 3]: **Primeritat. Construcció de primers**.
Travesa, A.: ConstruccioDePrimers; accessible en forma notebook o en format pdf des de https://travesa.cat/notes/
Contingut: Test de primeritat de Solovay-Strassen. Test de primeritat de Miller-Rabin. Un certificat congruencial de primeritat. Construcció certificada de nombres primers de mida prefixada. Aplicació al càlcul de claus RSA. Aplicació (exercici) al càlcul de claus ElGamal.

[Cripto- 4]: **Factorització**.
Travesa, A.: Factoritzacio; accessible en forma notebook o en format pdf des de https://travesa.cat/notes/
Contingut: Un garbell d'Eratòstenes. Tests de primeritat de Solovay-Strassen i de Miller-Rabin. Un certificat congruencial de primeritat. Un algoritme bàsic de divisó per nombres primers petits. Un algoritme bàsic de divisó per nombres petits. El mètode de factorització de Fermat. El mètode de factorització p-1 de Pollard. El mètode de factorització rho de Pollard.

[RSA]: **Criptosistemes de tipus RSA**.
Travesa, A.: RSA; accessible en forma notebook o en format pdf des de https://travesa.cat/notes/
Contingut: Una descripció bàsica dels criptosistemes de tipus RSA: les claus; xifratge; desxifratge; observacions sobre la seguretat.

[ElGamal]: **El criptosistema ElGamal**.
Travesa, A.: ElGamal; accessible en forma notebook o en format pdf des de https://travesa.cat/notes/
Contingut: Logaritmes discrets. Una descripció bàsica del criptosistema ElGamal: el grup cíclic; les claus; xifratge i desxifratge.

[Tr-1]: Travesa, A.: *Aritmetica*. Edicions de la Universitat de Barelona, col·lecció UB, n. 25. Barcelona, 1998. ISBN:84-8338-031-5.

Introducció

Presentació

En uns altres *notebooks*, hem iniciat el tractament del problema de la factorització de nombres naturals; de fet, hem implementat un garbell d'Eratòstenes per a calcular llistes completes de nombres primers fins a fites determinades, i hem implementat alguns tests i algun certificat de primeritat. Més avall proposem una implementació bàsica d'un algoritme de factorització per divisió per primers petits.
Així, donat un nombre natural N>1, que volem descompondre com a producte de nombres primers, podrem usar aquesta funció per a trobar-ne els divisors primers més petits que una certa fita F. En acabar aquest procés, poden passar dues coses diferents: o bé el nombre que volíem descompondre ja ha estat descompost completament (i, en aquest cas, hem acabat), o bé encara resta un factor de N, diem-ne M, per a descompondre. I per a aquest factor M encara poden passar dues coses: si M és menor que el quadrat del màxim dels nombres primers pels quals hem provat la divisió (i aquests són tots els nombres primers menors que F), aleshores M és primer i també hem acabat la descomposició.

Però pot succeir (i, de fet, succeeix la majoria de vegades) que el nombre M sigui més gran que el quadrat del darrer nombre primer pel qual hem provat la divisió. En aquest cas, sabem que M no és divisible per primers menors que la fita F (en particular, si fem que sigui F>3, podem suposar que M és senar i no divisible per 3).

Llavors, cal saber si M és o no és primer i, si no ho és, descompondre'l. En aquestes notes es tracten aquests dos problemes.

Observació 1

Les funcions següents no han estat optimitzades en cap cas; ni tan sols s'han considerat filtres per a comprovar les dades d'entrada.
Per tant, només es pot garantir mínimament el resultat del càlcul si les dades d'entrada són correctes.
Per exemple, s'ha suposat que la fita d'entrada per al garbell d'Eratòstenes és un nombre positiu; no s'ha intentat preveure què succeeix si és un nombre negatiu, o bé no és un nombre enter, o bé ni tan sols és un nombre... (per exemple, una llista o una variable, o una tira de caràcters, etcètera).
Consideracions semblants s'apliquen a totes les altres funcions.

Observació 2

Convé començar per executar les funcions de la secció següent, perquè es fan servir més avall.

Funcions necessàries

Un garbell d'Eratòstenes

```
Eratostenes[ff_] := Module[{pr, i, j, f, k},
  f = Floor[(ff + 1) / 2];
  pr = Table[1, {i, 1, f}];
  i = 2;
  k = Floor[(Sqrt[ff] + 1) / 2];
  While[i ≤ k,
   If[pr[[i]] == 1, For[j = 2 i (i - 1) + 1, j ≤ f, j += 2 i - 1, pr[[j]] = 0 ]];
   i = i + 1;
  ];
  Complement[Union[{2}, Table[(2 i - 1) pr[[i]], {i, 2, f}]], {0}]
 ]
```

Tests de primeritat

```
SolovayStrassen[nn_, ff_] := Module[{n = Abs[nn], f, g, x, y},
  If[Not[IntegerQ[n]], Print["Cal que n sigui un nombre enter."]; Return[Null], Null];
  If[ff < 1, Print["Cal fer alguna prova."]; Return[Null], Null];
  If[n == 2 || n == 3 || n == 5 || n == 7, Return[True], Null];
  If[n == 1 || EvenQ[n], Return[False], Null];
  f = 0;
  While[f < ff,
   g = RandomInteger[{2, n - 2}];
   x = PowerMod[g, (n - 1) / 2, n];
   If[x == 1 || x == n - 1, y = Mod[JacobiSymbol[g, n], n];
    If[x ≠ y, Return[False], Null], Return[False]];
   f++];
  Return[Indeterminate]
 ]
```

```
MillerRabin[nn_, ff_] := Module[{n = Abs[nn], v, m, f, g, x, k},
  If[Not[IntegerQ[n]], Print["Cal que n sigui un nombre enter."]; Return[Null], Null];
  If[ff < 1, Print["Cal fer alguna prova."]; Return[Null], Null];
  If[n == 2 || n == 3 || n == 5 || n == 7, Return[True], Null];
  If[n == 1 || EvenQ[n], Return[False], Null];
  v = 0; m = n - 1; While[EvenQ[m], v++; m = m / 2];
  f = 0;
  While[f < ff,
   g = RandomInteger[{2, n - 2}];
   x = PowerMod[g, m, n];
   If[x ≠ 1 && x ≠ n - 1,
    k = 1;
    x = PowerMod[x, 2, n];
    While[x ≠ n - 1 && k < v - 1, x = PowerMod[x, 2, n]; k++];
    If[k ≥ v - 1 && x ≠ n - 1, Return[False], Null], Null];
   f++];
  Return[Indeterminate]
 ]
```

Un certificat de primeritat i una funció per a comprovar-los

```
Certificat[pp_, f_, n_] := Module[{p = Abs[pp], fppmu, l, m, g, mm, s},
  If[IntegerQ[p], Null, Print["Cal que p sigui un nombre enter."]; Return[Null]];
  If[n < 1, Print["Cal fer alguna prova."]; Return[Null], Null];
  If[p == 2 || p == 3, Return[{True, p - 1, {p - 1}}], Null];
  If[p == 1 || EvenQ[p], Return[{False, 2}], Null];
  If [f == {}, fppmu = Transpose[FactorInteger[p - 1]][[1]], fppmu = Sort[f]];
  l = Length[fppmu]; m = 0;
  While[m < n, g = RandomInteger[{2, p - 2}]; If[(s = PowerMod[g, (p - 1) / 2, p]) == p - 1,
    i = 2; While[i ≤ l && PowerMod[g, (p - 1) / fppmu[[i]], p] ≠ 1, i++];
    If[i == l + 1, Return[{True, g, fppmu}], Null], If[s ≠ 1, Return[{False, g}], Null]]; m++];
  Return[Indeterminate]
 ]
```

```
Comprova[pp_, cert_] := Module[{p = pp, g = cert[[2]], lta = cert[[3]], x, q, d},
  If[p == 2, Return[cert == {True, 1, {1}}], Null];
  If[p == 3, Return[cert == {True, 2, {2}}], Null];
  x = Product[lta[[i]], {i, 1, Length[lta]}];
  If[Mod[p - 1, x], Return[False], Null];
  q = (p - 1) / x;
  d = GCD[q, x];
  While[d > 1, q = q / d; d = GCD[q, x]];
  If[q ≠ 1, Return[False], Null];
  If[PowerMod[g, p - 1, p] ≠ 1, Return[False], Null];
  If[MemberQ[Table[PowerMod[g, (p - 1) / lta[[i]], p ], {i, 1, Length[lta]}], 1],
   Return[False], Return[True]]
 ]
```

Un algoritme bàsic de divisió per primers petits

```
Divisio0[nn_, ff_] := Module[{n = Abs[nn], pr, l, p, a, b, lta, i},
  If[nn == 1, Print["Factorització completa."]; Return[{{1, 1}}]];
  If[nn == -1, Print["Factorització completa."]; Return[{{-1, 1}}]];
  If[IntegerQ[ff], pr = Eratostenes[Min[ff, 10^5]], pr = ff];
  l = Length[pr];
  lta = {};
  If[nn < 0, AppendTo[lta, {-1, 1}]];
  i = 1;
  p = pr[[i]];
  While[n >= p^2 && i < l,
   {a, b} = QuotientRemainder[n, p];
   If[b == 0, AppendTo[lta, {p, 1}]; n = a; {a, b} = QuotientRemainder[n, p];
    While[b == 0, lta[[-1, 2]]++ ; n = a; {a, b} = QuotientRemainder[n, p]]];
   i++;
   p = pr[[i]];
  ];
  If[n >= p^2 && i == l,
   {a, b} = QuotientRemainder[n, p];
   If[b == 0, AppendTo[lta, {p, 1}]; n = a; {a, b} = QuotientRemainder[n, p];
    While[b == 0, lta[[-1, 2]]++ ; n = a; {a, b} = QuotientRemainder[n, p]]];];
  If[n < p^2 && n > 1, AppendTo[lta, {n, 1}]; n = 1;
   Print["Factorització completa."]; Return[lta]];
  If[n < p^2 && n == 1, Print["Factorització completa."]; Return[lta]];
  Print["El darrer factor pot ésser primer o no."];
  Return[{lta, n}]
 ]
```

Estudi de la primeritat del darrer factor

Un cop hem provat la factorització per divisió d'acord amb les funcions anteriors, i en el cas que no puguem assegurar directament que la factorització és completa, pot ser que encara es pugui determinar si el darrer factor és o no és primer.

Convé afegir un test de primeritat; si el nombre no el passa, tindrem una prova que el factor és compost.

Exercici 1

Es demana afegir a la funció Divisio0 anterior un test de primeritat per al darrer factor i modificar la sortida d'acord amb si podem assegurar que és compost o no.

```
Divisio1[nn_, ff_] := Module[{n = Abs[nn], pr, l, p, a, b, lta, i, fita = 50},
  If[nn == 1, Print["Factorització completa."]; Return[{{1, 1}}]];
  If[nn == -1, Print["Factorització completa."]; Return[{{-1, 1}}]];
  If[IntegerQ[ff], pr = Eratostenes[Min[ff, 10^5]], pr = ff];
  l = Length[pr];
  lta = {};
  If[nn < 0, AppendTo[lta, {-1, 1}]];
  i = 1;
  p = pr[[i]];
  While[n >= p^2 && i < l,
   {a, b} = QuotientRemainder[n, p];
   If[b == 0, AppendTo[lta, {p, 1}]; n = a; {a, b} = QuotientRemainder[n, p];
    While[b == 0, lta[[-1, 2]]++ ; n = a; {a, b} = QuotientRemainder[n, p]]];
   i++;
   p = pr[[i]];
  ];
  If[n >= p^2 && i == l,
   {a, b} = QuotientRemainder[n, p];
   If[b == 0, AppendTo[lta, {p, 1}]; n = a; {a, b} = QuotientRemainder[n, p];
    While[b == 0, lta[[-1, 2]]++ ; n = a; {a, b} = QuotientRemainder[n, p]]];];
  If[n < p^2 && n > 1, AppendTo[lta, {n, 1}]; n = 1;
   Print["Factorització completa."]; Return[lta]];
  If[n < p^2 && n == 1, Print["Factorització completa."]; Return[lta]];
  If[SolovayStrassen[n, fita] == False,
   Print["El darrer factor és compost."]; Return[{lta, n}]];
  Print["El darrer factor probablement és primer."];
  Return[{lta, n}]
 ]
```

Provem la nova funció en tots els casos.

```
Divisio1[Factorial[10] 103 , 10]
```

El darrer factor probablement és primer.

```
{{{2, 8}, {3, 4}, {5, 2}, {7, 1}}, 103}
```

```
Divisio1[Factorial[100] , 85]
```

El darrer factor és compost.

```
{{{2, 97}, {3, 48}, {5, 24}, {7, 16}, {11, 9}, {13, 7}, {17, 5}, {19, 5}, {23, 4}, {29, 3}, {31, 3}, {37, 2},
  {41, 2}, {43, 2}, {47, 2}, {53, 1}, {59, 1}, {61, 1}, {67, 1}, {71, 1}, {73, 1}, {79, 1}, {83, 1}}, 8633}
```

```
Divisio1[Factorial[100] , 89]
```

Factorització completa.

```
{{2, 97}, {3, 48}, {5, 24}, {7, 16}, {11, 9}, {13, 7}, {17, 5}, {19, 5}, {23, 4}, {29, 3}, {31, 3}, {37, 2},
 {41, 2}, {43, 2}, {47, 2}, {53, 1}, {59, 1}, {61, 1}, {67, 1}, {71, 1}, {73, 1}, {79, 1}, {83, 1}, {89, 1}, {97, 1}}
```

Exercici 2

Es demana afegir a la funció anterior un certificat de primeritat per al darrer factor i modificar la sortida d'acord amb si podem assegurar que és primer o no.

```
Divisio[nn_, ff_] := Module[{n = Abs[nn], pr, l, p, a, b, lta, i, fita = 50, fita2 = 10, cert},
  If[nn == 1, Print["Factorització completa."]; Return[{{1, 1}}]];
  If[nn == -1, Print["Factorització completa."]; Return[{{-1, 1}}]];
  If[IntegerQ[ff], pr = Eratostenes[Min[ff, 10^5]], pr = ff];
  l = Length[pr];
  lta = {};
  If[nn < 0, AppendTo[lta, {-1, 1}]];
  i = 1;
  p = pr[[i]];
  While[n >= p^2 && i < l,
   {a, b} = QuotientRemainder[n, p];
   If[b == 0, AppendTo[lta, {p, 1}]; n = a; {a, b} = QuotientRemainder[n, p];
    While[b == 0, lta[[-1, 2]]++ ; n = a; {a, b} = QuotientRemainder[n, p]]];
   i++;
   p = pr[[i]];
  ];
  If[n >= p^2 && i == l,
   {a, b} = QuotientRemainder[n, p];
   If[b == 0, AppendTo[lta, {p, 1}]; n = a; {a, b} = QuotientRemainder[n, p];
    While[b == 0, lta[[-1, 2]]++ ; n = a; {a, b} = QuotientRemainder[n, p]]];];
  If[n < p^2 && n > 1, AppendTo[lta, {n, 1}]; n = 1;
   Print["Factorització completa."]; Return[lta]];
  If[n < p^2 && n == 1, Print["Factorització completa."]; Return[lta]];
  If[SolovayStrassen[n, fita] == False,
   Print["El darrer factor és compost."]; Return[{lta, n}]];
  cert = Certificat[n, {}, fita2];
  If[cert[[1]] == True, AppendTo[lta, {n, 1}];
   Print["Factorització completa."]; Return[lta]];
  If[cert[[1]] == False, Print["El darrer factor és compost."]; Return[{lta, n}]];
  Print["El darrer factor pot ésser primer o compost."];
  Return[{lta, n}]
 ]
```

Provem la nova funció.

```
Divisio[Factorial[10] 103 , 10]
```

Factorització completa.

```
{{2, 8}, {3, 4}, {5, 2}, {7, 1}, {103, 1}}
```

```
Divisio[Factorial[100] 103, 10]
```

El darrer factor és compost.

```
{{{2, 97}, {3, 48}, {5, 24}, {7, 16}}, 383 939 088 209 963 459 650 577 437 583 591 321 716 996 212 133 864 662 618 960 360 090 576 574 593 691}
```

```
Divisio[Factorial[100] , 89]
```

Factorització completa.

```
{{2, 97}, {3, 48}, {5, 24}, {7, 16}, {11, 9}, {13, 7}, {17, 5}, {19, 5}, {23, 4}, {29, 3}, {31, 3}, {37, 2},
 {41, 2}, {43, 2}, {47, 2}, {53, 1}, {59, 1}, {61, 1}, {67, 1}, {71, 1}, {73, 1}, {79, 1}, {83, 1}, {89, 1}, {97, 1}}
```

Mètode de factorització de Fermat

Si coneixem que el darrer factor encara és compost, encara podem intentar factoritzar-lo amb altres mètodes.

La funció següent implementa l'algoritme de factorització de Fermat amb una restricció en el nombre de passos.

```
Fermat[nn_, ff_] := Module[{f, v, an, u, r},
  f = 2 (ff + 1);
  v = 1;
  an = Floor[Sqrt[nn]];
  u = 2 (an + 1) + 1;
  r = (an + 1)^2 - nn;
  While[r > 0 && v < f, r -= v; v += 2];
  While[r < 0, r += u; u += 2; While[r > 0 && v < f, r -= v; v += 2]];
  If[v < f, Return[{(u - v) / 2, (u + v - 2) / 2}], Return[{nn, 1}]]
 ]
```

Exercici 3

Quin és el sentit de ff en aquesta funció?

El valor de ff limita el nombre de passos de l'algoritme a un màxim de ff+1.

Segons el valor de ff, per a quins nombres compostos nn la funció Fermat[] trencarà nn en dos factors?

Per als nombres que siguin producte de dos factors r, s tals que 0< s-r <= 2 ff. En particular, si nn és el quadrat d'un nombre primer, no el factoritzarà en pocs passos.

Si nn és el producte de dos nombres primers bessons, quants passos d'algoritme seran necessaris per a factoritzar nn?

Un sol pas.

Es demana utilitzar aquesta funció per a factoritzar alguns nombres compostos.

```
Fermat[41 × 43, 0]
```

```
{1763, 1}
```

```
Fermat[41 × 43, 1]
```

```
{41, 43}
```

```
Fermat[41 × 47, 2]
```

```
{1927, 1}
```

```
Fermat[41 × 47, 3]
```

```
{41, 47}
```

A continuació, alguns exemples que no semblen trivials.

```
p = NextPrime[RandomInteger[10^500]];
```

```
q = NextPrime[p];
```

```
Block[{$MaxExtraPrecision = 500}, Fermat[p q, 100 000]]
```

```
{45 532 341 863 126 570 183 555 533 956 393 700 366 965 605 270 761 421 358 764 952 601 380 391 748 814 733 705 539 477 204 277 765 883 425 682 859 ↓
  469 801 319 852 920 708 967 916 939 040 938 526 949 974 735 279 742 644 909 816 323 772 908 617 298 376 272 452 162 878 640 156 611 272 860 001 ↓
  770 509 113 958 727 729 632 693 113 191 938 782 090 885 994 930 751 461 633 388 886 794 820 080 536 468 914 175 343 680 525 740 733 580 838 117 ↓
  900 125 655 049 251 416 446 676 532 352 967 577 402 110 994 950 118 910 203 974 843 810 041 759 161 339 602 983 016 380 239 111 284 160 392 133 ↓
  343 136 279 938 174 008 343 289 598 919 642 964 444 253 526 653 022 952 079 751 832 359,
 45 532 341 863 126 570 183 555 533 956 393 700 366 965 605 270 761 421 358 764 952 601 380 391 748 814 733 705 539 477 204 277 765 883 425 682 ↓
  859 469 801 319 852 920 708 967 916 939 040 938 526 949 974 735 279 742 644 909 816 323 772 908 617 298 376 272 452 162 878 640 156 611 272 860 ↓
  001 770 509 113 958 727 729 632 693 113 191 938 782 090 885 994 930 751 461 633 388 886 794 820 080 536 468 914 175 343 680 525 740 733 580 838 ↓
  117 900 125 655 049 251 416 446 676 532 352 967 577 402 110 994 950 118 910 203 974 843 810 041 759 161 339 602 983 016 380 239 111 284 160 392 ↓
  133 343 136 279 938 174 008 343 289 598 919 642 964 444 253 526 653 022 952 079 751 832 809}
```

Mètode de factorització p-1 de Pollard

La funció següent implementa una versió bàsica de l'algoritme de factorització p-1 de Pollard.

```
PollardPM1[nn_, ff_] := Module[{a, x, r, d},
  a = 2;
  x = a;
  r = 2;
  While[r <= ff, x = PowerMod[x, r, nn]; d = GCD[x - 1, nn];
   If[d == nn, Return[nn]]; If[d > 1, Return[Sort[{d, nn / d}]]]; r += 1];
  Return[nn]
 ]
```

Exercici 4

Es demana explicar quins nombres pot factoritzar aquesta funció i dir si es pot usar algun valor de *a* diferent de 2.

Els nombres n que pot factoritzar aquesta funció són els divisibles per algun nombre primer p tal que p-1 divideix el factorial de ff (de fet, que l'ordre de a mòdul p divideix el factorial de ff), excepte que tots els divisors primers de p-1 tinguin el mateix màxim nombre primer que els divideixi.
En lloc de a=2 es pot utilitzar qualsevol altre nombre enter diferent de 0, -1 i 1.

Intentem factoritzar el producte (Factorial[27] + 1) (Factorial[37] + 1) (Factorial[80] + 1).

Quin valor de ff és adequat?

Per a un primer pas, és suficient ff=27, però no ff=26.

```
PollardPM1[(Factorial[27] + 1) (Factorial[37] + 1) (Factorial[80] + 1), 26]
```

```
10 726 236 951 242 006 090 923 123 085 756 600 979 624 212 197 394 719 451 941 989 075 196 626 186 232 831 669 138 579 330 170 920 079 875 265 194 ∖
 370 769 310 909 856 276 384 260 328 418 087 261 122 396 579 810 107 647 708 442 827 043 133 063 168 000 001
```

```
PollardPM1[(Factorial[27] + 1) (Factorial[37] + 1) (Factorial[80] + 1), 27]
```

```
{10 888 869 450 418 352 160 768 000 001,
 985 064 335 657 904 530 906 571 232 094 964 195 854 994 970 463 220 635 338 590 103 922 625 894 509 086 874 776 297 727 178 187 611 990 324 319 ∖
  000 833 407 895 121 002 613 197 004 327 130 059 581 580 902 400 000 001}
```

Notem que Factorial[27] + 1 i Factorial[37] + 1 són primers, i que Factorial[80] + 1 és compost.

```
MillerRabin[Factorial[27] + 1, 10]
```

```
Indeterminate
```

```
Certificat[Factorial[27] + 1, {}, 100]
```

```
{True, 2 112 935 056 414 625 440 554 007 735, {2, 3, 5, 7, 11, 13, 17, 19, 23}}
```

```
MillerRabin[Factorial[37] + 1, 10]
```

```
Indeterminate
```

```
Certificat[Factorial[37] + 1, {}, 100]
```

```
{True, 10 275 223 045 201 227 890 217 506 818 942 722 950 794 928, {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37}}
```

```
MillerRabin[Factorial[80] + 1, 10]
```

```
False
```

El factor que hem trobat és Factorial[27] + 1; per a trobar els altres cal augmentar el valor de ff.

```
PollardPM1[ (Factorial[37] + 1) (Factorial[80] + 1), 50]
```

```
{13 763 753 091 226 345 046 315 979 581 580 902 400 000 001,
 71 569 457 046 263 802 294 811 533 723 186 532 165 584 657 342 365 752 577 109 445 058 227 039 255 480 148 842 668 944 867 280 814 080 000 000 ∴
  000 000 000 001}
```

El factor que hem trobat és Factorial[37] + 1. Resta factoritzar Factorial[80] + 1. Caldrà augmentar el valor de ff.

```
PollardPM1[Factorial[80] + 1, 5000]
```

```
{672 937,
 106 353 874 205 555 352 573 586 433 385 571 802 658 472 720 837 709 551 677 362 732 407 680 123 481 812 040 120 648 656 363 494 374 778 025 283 ∖
  198 873}
```

Notem que (Factorial[80] + 1)/672937 encara és compost.

```
MillerRabin[(Factorial[80] + 1) / 672 937, 10]
```

```
False
```

Notem que (Factorial[80] + 1)/672937 no es factoritza fàcilment amb el mètode p-1. (De fet, la fita ff=1000000 no és suficient.)

```
PollardPM1[ (Factorial[80] + 1) / 672 937, 1 000 000]
```

```
106 353 874 205 555 352 573 586 433 385 571 802 658 472 720 837 709 551 677 362 732 407 680 123 481 812 040 120 648 656 363 494 374 778 025 283 198 ∖
 873
```

I el mètode de Fermat tampoc no el factoritza amb la fita ff=10000000.

```
Fermat[ (Factorial[80] + 1) / 672 937, 10 000 000]
```

```
{106 353 874 205 555 352 573 586 433 385 571 802 658 472 720 837 709 551 677 362 732 407 680 123 481 812 040 120 648 656 363 494 374 778 025 283 ∖
  198 873, 1}
```

Mètode de factorització rho de Pollard

La funció següent implementa una versió bàsica de l'algoritme de factorització rho de Pollard.

```
PollardRho[nn_, tt_, ff_] := Module[{a, x, y, t, f},
  f = ff;
  While[f > 0,
   a = RandomInteger[nn];
   x = RandomInteger[nn];
   y = x;
   t = tt;
   While[t > 0, x = Mod[x^2 + a, nn]; y = Mod[(y^2 + a)^2 + a, nn];
    d = GCD[x - y, nn]; If[d > 1 && d < nn, Return[Sort[{d, nn / d}]]];
    If[d == 1, t -= 1]];
   f -= 1];
  Return[nn]
 ]
```

Exercici 5

Expliqueu quins valors de tt i de ff són raonables per a usar la funció i quins nombres pot factoritzar.

Els nombres nn que pot factoritzar aquesta funció són els divisibles per algun nombre primer p prou petit a fi que els períodes de les successions recursives donades per x^2+a mòdul p siguin menors que tt. La prova es fa per a ff successions aleatòries. Per tant, no cal fer ff gaire gran; val més fer tt més gran.

Intentem factoritzar de nou el nombre (Factorial[27]+1) (Factorial[37]+1) (Factorial[80]+1).

```
PollardRho[(Factorial[27] + 1) (Factorial[37] + 1) (Factorial[80] + 1), 10^6, 15]
```

```
{672 937,
 15 939 437 051 673 494 087 742 423 266 600 886 828 372 064 840 237 227 930 611 616 057 961 779 759 818 276 702 185 463 617 204 760 742 647 922 ∴
  754 092 536 613 248 872 147 592 211 943 195 406 495 886 534 073 486 979 684 143 452 993 434 947 198 873}
```

Notem que ara apareix com a factor un nombre que, abans, costava una miqueta i que, en canvi, els factors que apareixien abans fàcilment, ara no apareixen de manera senzilla.

Però n'apareix un de nou! (que ho és, per força, de Factorial[80]+1).

```
PollardRho[(Factorial[27] + 1) (Factorial[37] + 1) (Factorial[80] + 1) / 672 937, 10^6, 15]
```

```
{351 900 745 811,
 45 295 263 625 938 147 096 297 241 347 141 291 490 702 804 398 090 301 183 535 095 380 701 218 481 562 431 797 745 501 872 220 130 492 413 128 ×
  948 463 831 857 033 165 734 100 149 554 500 616 948 641 395 255 184 265 472 893 539 043}
```

```
MillerRabin[(Factorial[80] + 1) / (672 937 × 351 900 745 811), 10]
```

```
False
```

```
PollardRho[(Factorial[80] + 1) / (672 937 × 351 900 745 811), 10^6, 15]
```

```
302 226 907 648 204 411 931 246 850 043 840 081 307 353 909 970 965 053 486 630 367 974 989 354 040 997 196 506 077 104 204 669 539 043
```

Però no podem acabar de factoritzar Factorial[80]+1 amb la funció PollardRho amb 15 successions i una fita de 1000000 per als períodes.

Observacions finals

Hi ha altres mètodes de factorització més elaborats i més potents que permeten factoritzar aquest nombre.
Notem que podem dir de seguida que el nombre és compost.

```
SolovayStrassen[(Factorial[80] + 1) / (672 937 × 351 900 745 811), 10]
```

```
False
```

I *Mathematica* és capaç de factoritzar-lo de manera prou ràpida.

```
FactorInteger[(Factorial[80] + 1) / (672 937 × 351 900 745 811)]
```

```
{{1 374 851 388 985 363, 1},
 {219 825 146 244 531 300 827 618 434 834 380 439 599 661 150 333 525 461 306 467 196 762 698 338 736 604 216 421 361, 1}}
```

Notem, però, que *Mathematica* no és capaç de factoritzar de manera prou ràpida el nombre inicial. Per tant, potser no té implementat el mètode p-1 de Pollard?

```
FactorInteger[(Factorial[27] + 1) (Factorial[37] + 1) (Factorial[80] + 1)]
```

```
$Aborted
```

Observació: Aquest *notebook* ha estat treballat amb la versió 7.0 (en aquests moments ja obsoleta) de *Mathematica*. Per tant, pot ser que en la versió actual aquest càlcul ja sigui possible o, fins i tot, senzill, per causa de la millora constant del programari. També pot ser que en noves versions hi hagi implementats més mètodes de factorització.