Mersenne

January 9, 2021

1 Nombres de Mersenne

Marc Lorenzi

18 juillet 2016

[1]:
```
import csv
from math import log, exp
import matplotlib.pyplot as plt
```

[17]:
```
plt.rcParams['figure.figsize'] = (12, 6)
```

1.1 1 Introduction

Soit $p \in \mathbb{N}$. Le $p$ième nombre de Mersenne est $M_p = 2^p - 1$.

[2]:
```
def mersenne(p):
    return 2 ** p - 1
```

[4]:
```
[(k,mersenne(k)) for k in range(20)]
```

[4]:
```
[(0, 0),
 (1, 1),
 (2, 3),
 (3, 7),
 (4, 15),
 (5, 31),
 (6, 63),
 (7, 127),
 (8, 255),
 (9, 511),
 (10, 1023),
 (11, 2047),
 (12, 4095),
 (13, 8191),
 (14, 16383),
 (15, 32767),
```

```
 (16, 65535),
 (17, 131071),
 (18, 262143),
 (19, 524287)]
```

Proposition : Soit $p \in \mathbb{N}$. Si M_p est premier, alors p est premier.

Démonstration : supposons que p est composé : $p = ab$ où $2 \leq a, b < p$. On a alors

$$M_p = (2^a)^b - 1 = (2^a - 1) \sum_{k=0}^{b-1} 2^{ka}$$

Or, $a \geq 2$ donc $2^a - 1 \geq 3 > 1$ et $a < p$ donc $2^a - 1 < 2^p - 1 = M_p$. Ainsi, $2^a - 1$ est un diviseur non trivial de M_p, qui est donc composé.

La réciproque de cette proposition est hélas fausse et on connaît des nombres de Mersenne M_p avec p premier qui, eux, ne sont pas premiers. En fait on en connaît même beaucoup, et les nombres de Mersenne premiers sont assez rares. À l'heure où est écrit ce notebook, on en connaît 51. Le dernier en date a été découvert en décembre 2018.

Histoire de pouvoir faire quelques petits tests, voici une fonction (assez inefficace) qui renvoie `True` lorsque son paramètre est premier et `False` sinon.

[5]:
```
def est_premier(p):
    if p <= 1: return False
    k = 2
    while k * k <= p and p % k != 0: k += 1
    return k * k > p
```

[6]:
```
print([p for p in range(100) if est_premier(p)])
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73,
79, 83, 89, 97]
```

Voici la liste des nombres premiers p inférieurs à 60 tels que M_p soit premier.

[7]:
```
print([p for p in range(60) if est_premier(p) and est_premier(mersenne(p))])
```

```
[2, 3, 5, 7, 13, 17, 19, 31]
```

Pourquoi s'arrêter à 60 ? Et M_{61} ? En fait notre algorithme **est_premier** est beaucoup trop naïf pour pouvoir tester la primalité d'un nombre d'une vingtaine de chiffres en un temps raisonnable …

[8]:
```
mersenne(61)
```

[8]:
```
2305843009213693951
```

Heureusement, il existe un test efficace pour déterminer si un nombre de Mersenne est premier. Il s'agit du test de Lucas.

1.2 2 Le test de Lucas

Proposition : soit p un nombre premier. On pose $L_1 = 4$ puis, pour tout i entre 2 et $p-1$, $L_i = L_{i-1}^2 - 2$. Alors, M_p est premier si et seulement si $L_{p-1} \equiv 0[M_p]$.

La preuve de ce résultat est loin d'être facile, je l'admettrai ici. Contentons-nous d'écrire la fonction et de tester.

[9]:
```
def lucas(p):
    Mp = mersenne(p)
    L = 4
    for k in range(1, p-1):
        L = (L * L - 2) % Mp
    return L == 0
```

Voici tous les entiers p premiers inférieurs à 2000 pour lesquels M_p est premier.

[10]:
```
[p for p in range(2000) if est_premier(p) and lucas(p)]
```

[10]:
```
[3, 5, 7, 13, 17, 19, 31, 61, 89, 107, 127, 521, 607, 1279]
```

Le dernier nombre de la liste est déjà un très grand nombre.

[11]:
```
mersenne(1279)
```

[11]:
```
104079321946643990819252403273640855386152622472667048053191123504036080596733602980122394417323241848424216139542810077913835662483234649081399066056773207629241295093892203457731833496615835504729594205476898112116936771475484788669625013844382602917323488853111608285384165850282556046662248318909188018470682222031405210266984354887329580288780508697361869007147207105557031687290870
```

Terminons sur un exemple encore plus gros.

[12]:
```
lucas(9941)
```

[12]:
```
True
```

[13]:
```
mersenne(9941)
```

[13]:
```
346088282490851215242960395767413316722628668900238547790489283445006220809834114464364375544153707533664486747635050186414707093323739706083766904042292657896479937097603584695523190454849100503041498098185402835071596835622329419680597622813345447397208492609048551927706260549117935903890607959811638387214329942787636330953774381948448664711249676857988881722120330008214696844649561469971941269212843362064633138595375772004624420290646813260875582574884704893842439892702368849786430630930044229396033700105465953863020090730439444822025590974067005973305707995078329631309387398850801984162586351945229130425629366798595874957210311737477964188950607019417175060019371524300323636319342657985162360474512090898647074307803622983070381934454864937566479918042587755749738339033157350828910293920
```

359352758617185019942554834671861074548772439880729606244911940066680112823824095816458261761861746604034802056466823143718255492784779380991749580255263323326536457743894150848953969902818530057870876229329803338285735419228259022169602665532210834789602051686546011466737981306056247480055071718250333737502267307344178512950738594330684340802698228963986562732597175372087295649072830289749771358330867951508710859216743218522918811670637448496498549094430541277444079407989539857469452772132166580885754360477408842913327292948696897496141614919739845432835894324473601387609643750514699215032683744527071718684091832170948369396280061184593746143589068811190253101873595319156107319196071150598488070027088705842749605203063194191166922106176157609367241948160625989032127984748081075324382632093913796444665700601391278360323002267434295194325607280661260119378719405151497555187549252134264394645963853964913309697776533329401822158003182889278072368602128982710306618115118964131893657845400296860012420391376964670183983594954112484565597312460737798777092071706710824503707457220155015899591766244957768006802482976673920392995410164224776445671222149803657927708412925555542817045572430846389988129960519227313987291200902060882060733762075892299473666405897427035811786879875694315078654420055603469625309399653955932310466430039146465805452965014040019423897552675534768248624631951431493188170905972588780111850281190559073677771187432814088678674286302108275149258477101296451833651979717375170900505673645964696355331369819296000267389583289299126738345726980325998955997501176664201042888546085699446442834195232948787488410595750197438786353119204210855804692460582533832967771946911459901921324984968810021189968284941331573164056304725480868921823442538199590383852412786840833479611419970101792978355653650755329138298654246225346827207503606740745956958127383748717825918527473164970582095181312905519242710280573023145554793628499010509296055849712377978984921839997037415897674154830708629145484724536724572622450131479992681684310464449439022250504859250834761894788889552527898400988196200014868575640233136509145628127191354858275083907891469979019426224883789463551

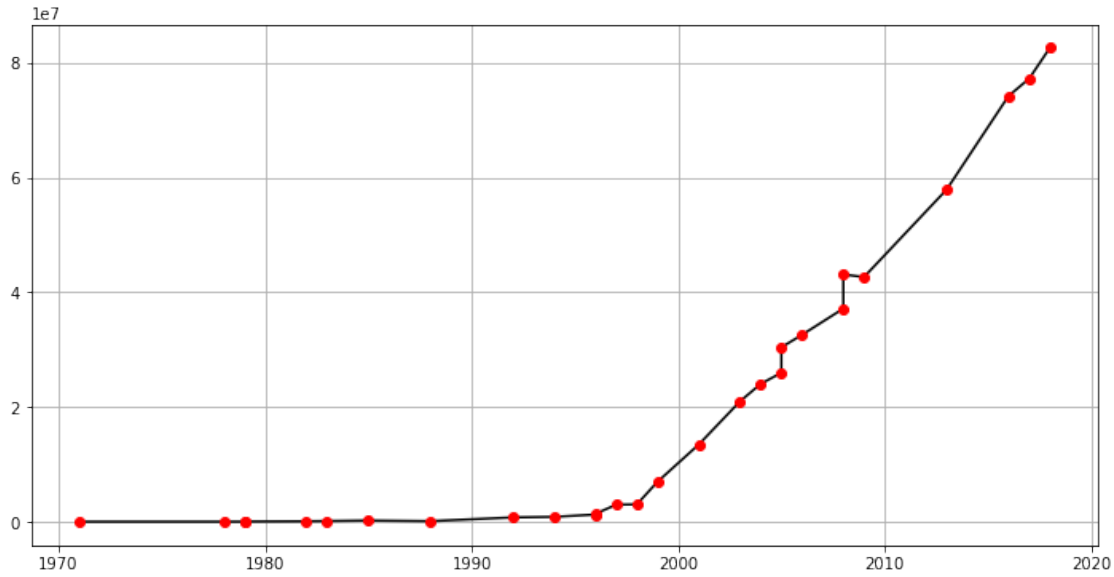
1.3 3 Les nombres de Mersenne connus

En janvier 2019 on connaît 51 nombres de Mersenne premiers. Le fichier `Mersenne_Status.csv` contient un certain nombre d'informations sur ces nombres : leur valeur, leur date de découverte, le nom du découvreur et la méthode utilisée. Vous pouvez y jeter un coup d'oeil.

[14]:
```
f = open('Mersenne_Status.csv', 'r')
rd = csv.reader(f, delimiter=';')
zs = []
for row in rd:
    try:
        zs.append((int(row[1]), int(row[2])))
    except: continue
zs.sort(key=lambda z: z[0])
```

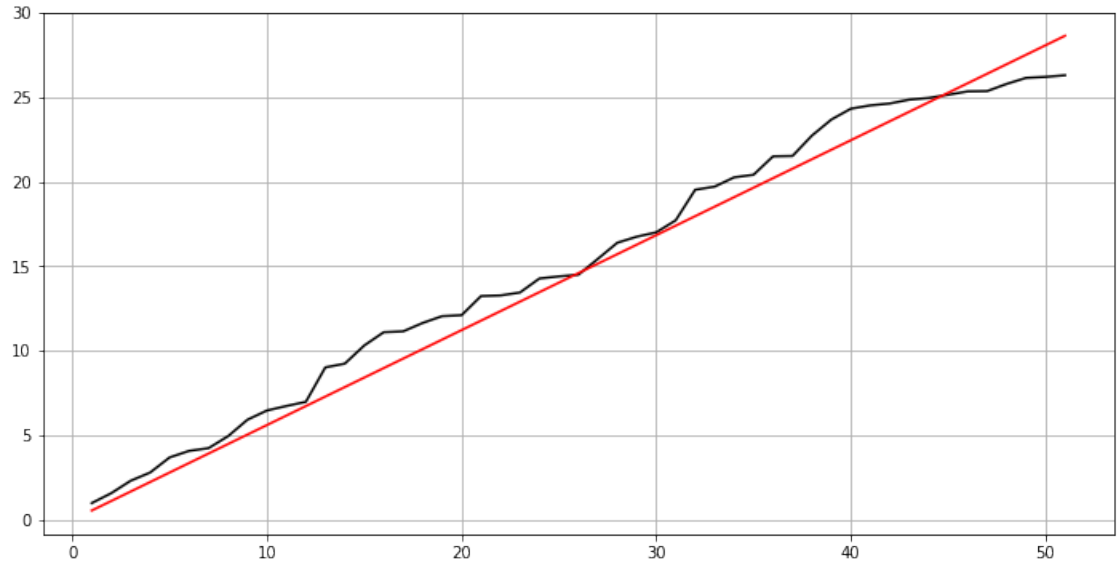
Traçons l'indice p en fonction de l'année de la découverte de M_p. On se restreint aux nombres découverts après 1970.

[18]:
```
N = 23
xs = [x for (x, y) in zs]
ys = [y for (x, y) in zs]
plt.plot(xs[N:], ys[N:], 'k')
plt.plot(xs[N:], ys[N:], 'or')
plt.grid()
plt.show()
```



Traçons pour finir $n \mapsto \lg\lg M_p \simeq \lg p$, où M_p est le $n$ième nombre de Mersenne et lg désigne le logarithme en base 2. En rouge la courbe de $n \mapsto ne^{-\gamma}$, où γ est la constante d'Euler.

[19]:
```
gamma = 0.577215
ys.sort()
ps = [log(p, 2) for p in ys]
xs = range(1, 52)
plt.plot(xs, ps, 'k')
us = [exp(-gamma) * x for x in xs]
plt.plot(xs, us, 'r')
plt.grid()
plt.show()
```



Faut-il attacher de l'importance à l'apparente concordance entre ces deux courbes ? Question ouverte …

[ ]: