Algoritmi u teoriji brojeva

Zadaća 2

Mihael Petrinjak

Zadatak 1.

In [34]:

```
R = 1000
m = 279
```

Pronađimo $m' :=$ `m_` , $R^{-1} \mod m :=$ `R_` i definirajmo $U(T) :=$ `U(T)` .

In [35]:

```
for i in range(0,R):
    if i * m % R == -1 + R:
        m_ = i
        break

for i in range(0,m):
    if R * i % m == 1 :
        R_ = i
        break

def U(T):
    return T * m_ % R

print("m_ =", m_)
print("R_ =",R_)
```

```
m_ = 681
R_ = 190
```

In [36]:

```
T = 2001
while(True):
    if (T + U(T) * m) / R - (T * R_ % m) == m :
        break
    T += 1
print(T)
```

```
2116
```

Dakle najmanji traženi broj je 2116.

Zadatak 2.

Prošireni Euklidov algoritam:

In [60]:

```
def EGCD(g,w):

    # osiguravamo g > w
    if(g < w):
        print("g > w !")

    # inicijalizacija
    x, y, u, v = 1, 0, 0, 1

    # petlja
    while w > 0:
        q = g // w
        x, y, g, u, v, w = u, v, w, x - q*u, y - q*v, g - q*w

    return x, y, g
```

7, 11, 13 su relativno prosti u parovima, jer su prosti, pa možemo koristiti *Kineski teorem o ostatcima*.

In [46]:

```
ms = [7, 11, 13]
xs = [4, 1, 0]
M = ms[0] * ms[1] * ms[2]
Ms = [M/ms[i] for i in range(0,3)]
Ms
```

Out[46]:

```
[143.0, 91.0, 77.0]
```

Algoritam za pronalaženje a_i takvih da $a_i \cdot M_i \equiv 1 \pmod{m_i}$. Koristimo prvu povratnu vrijednost iz `EGCD`.

In [62]:

```
x = sum([EGCD(Ms[i],ms[i])[0]*Ms[i]*xs[i] for i in range(0,len(xs))])
x
```

Out[62]:

```
-780.0
```

In [65]:

```
x += M
x
```

Out[65]:

```
1222.0
```

```
Provjera rezultata.
```

In [69]:

```
(x % 7, x % 11, x % 13)
```

Out[69]:

```
(4.0, 1.0, 0.0)
```

Dakle, traženi broj je 1222.

Zadatak 3.

Budući da će u ovom zadatku za sve brojeve d koje promatramo vrijediti

$$\sqrt{d} = \alpha = \alpha_0 = \frac{s_0 + \sqrt{d}}{t_0}$$

slijedi da $s_0 = 0, t_0 = 1$

In [70]:

```
import numpy as np
```

In [213]:

```
def VR(d, length):

    a = list()
    s = list()
    t = list()
    a.append( np.floor(np.sqrt(d)) )
    s.append(0)
    t.append(1)

    for i in range(1,length):
        if t[i-1] == 0 : return -1
        s.append( a[i-1] * t[i-1] - s[i-1] )
        t.append( (d - s[i]**2) / t[i-1] )
        if t[i] == 0: return -1
        alpha = (s[i] + np.sqrt(d)) / t[i]
        a.append( np.floor(alpha) )

    return a
```

Pišem funkciju `provjeri` koja vraća istinu ako dani broj zadovoljava uvjet zadatka.

In [318]:

```
# provjerava je li dana lista palindrom
def palindrom(komad):
    for i in range(0, len(komad)//2):
        if komad[i] != komad[-1-i]:
            return False
    return True

# provjerava nalazi li se u danoj listi niz s periodom
# duljine "duljina"
def period(komad, duljina):
    # izlazim ako su svi isti
    prvi = komad[0]
    isti = True
    for x in komad:
        if x != prvi:
            isti = False
            break
    if isti: return False

    # provjeravam postoji li period duljine "duljina"
    for i in range(0,duljina):
        if komad[i] != komad[duljina+i]:
            return False
    return True

def provjeri(razvoj, r):
    # prvi uvjet je da je palindrom i daima period duljine r
    if palindrom(razvoj[1:r]) and period(razvoj[1:], r):
        # izlazim ako postoji period manje duljine od r
        for k in range(2,r):
            if period(razvoj[1:], k):
                return False
        return True
    return False
```

Računamo kandidate do 1000.

In [321]:

```
kandidati = list()
for d in range(2,1000):
    razvoj = VR(d,61)
    if razvoj != -1:
        if provjeri(razvoj,30):
            kandidati.append(d)

kandidati
```

Out[321]:

```
[379, 436, 649, 772, 849, 946, 981]
```

Provjeravam vizualno za broj 379.

In [317]:

```
np.array(VR(379,61))
```

Out[317]:

```
array([19.,  2.,  7.,  3.,  2.,  2.,  6., 12.,  1.,  4.,  1.,  1.,  1.,
        3.,  4., 19.,  4.,  3.,  1.,  1.,  1.,  4.,  1., 12.,  6.,  2.,
        2.,  3.,  7.,  2., 38.,  2.,  7.,  3.,  2.,  2.,  6., 12.,  1.,
        4.,  1.,  1.,  1.,  3.,  4., 19.,  4.,  3.,  1.,  1.,  1.,  4.,
        1., 12.,  6.,  2.,  2.,  3.,  7.,  2., 38.])
```

Konačno, najmanji takav je 379.