Programmation linéaire

Segalen Philippe

28 juin 2017

Exponentiation modulaire

LtoR dichotomic exponentiation

```
def modexp_lr(a, b, m):
    r = 1
    for bit in reversed(_bits_of_n(b)):
        r = r * r % m
        if bit == 1:
            r = r * a % m
    return r
```

Fonction bits_of_n

```
def _bits_of_n(m):

    bits = []

    while m:
        bits.append(m % 2)
        m /= 2

    return bits
```

RtoL dichotomic exponentiation

```
def modexp_rl(a, b, m):
    r = 1
    while 1:
        if b % 2 == 1:
            r = r * a % m
        b /= 2
        if b == 0:
            break
        a = a * a % m

    return r
```

Tests

```
loukaine@loukaine-G75VX:~/Documents/MathRSA/test$ python test_math.py
first test
0.00205302238464
second test
0.00145196914673
```

Figure 1: Test 1

```
first test -  LtoR
[0.8344700336456299, 0.7609200477600098, 0.7624950408935547, 0.7613940238952637, 0.7834620475769043]
second test -  RtoL
[0.547105073928833, 0.5442409515380859, 0.5494589805603027, 0.5520210266113281, 0.5445020198822021]
```

Figure 2: Test 2