CS152-Homework9

Hongchen Cao 2019533114

2020.11.21

1

$262063 = x_{35} \times x_{70}$, where $x_{35} = 503, x_{70} = 521$, so iterations needed= 34.

$9420457 = x_{50} \times x_{100}$, where $x_{50} = 2351, x_{100} = 4007$, so iterations needed= 49.

$181937053 = x_{165} \times x_{330}$, where $x_{165} = 12391, x_{330} = 14683$, so iterations needed= 164.

```python
from math import gcd, sqrt
import numpy as np
import copy
from sage.all import inverse_mod
```

```python
def PollardRho(n: int, x1: int):
    iterations = 1
    x = x1
    x_ = (x ** 2 + 1) % n
    p = gcd(x - x_, n)
    while p == 1:
        x = (x ** 2 + 1) % n
        x_ = (x_ ** 2 + 1) % n
        x_ = (x_ ** 2 + 1) % n
        p = gcd(x - x_, n)
        iterations += 1
    if p == n:
        return None
    else:
        return "{} = x{} * x{}, where x{} = {}, \
        x{} = {}".format(n, iterations, 2 * iterations, iterations, p, 2 * iterations,
                                                     int(n / p))



print(PollardRho(262063, 1))
print(PollardRho(9420457, 1))
print(PollardRho(181937053, 1))
```

2

$256961 = 293 \times 877$

```python
from math import gcd, sqrt
import numpy as np
import copy
from sage.all import inverse_mod
```

```python
def DixonRandomSquare(B: list, n: int):
    related_squares = list()
    for i in range(500, n):
        left = i ** 2 % n
        for j in range(len(B)):
            right = B[j] ** 2 % n
            if left == right:
                related_squares.append((i, B[j]))
    res = list()
    for i in range(len(related_squares)):
        factor = gcd(related_squares[i][0] - related_squares[i][1], n)
        if factor != 1:
            res.append(factor)
    return np.unique(np.array(res))


print(DixonRandomSquare([-1, 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31], 256961))
```

3

$$317940011 = 25523 \times 12457$$

```python
from math import gcd, sqrt
import numpy as np
import copy
from sage.all import inverse_mod
```

```python
def Wiener(n: int, b: int):
    tempB = copy.deepcopy(b)
    tempN = copy.deepcopy(n)
    continued_fraction_expansion = [0]
    while b:
        continued_fraction_expansion.append(n // b)
        n, b = b, n % b
    b = tempB
    n = tempN

    c = list()
    d = list()
    c.append(1)
    c.append(continued_fraction_expansion[0])
    d.append(0)
    d.append(1)
    for j in range(2, len(continued_fraction_expansion) + 1):
        c.append(continued_fraction_expansion[j - 1] * c[j - 1] + c[j - 2])
        d.append(continued_fraction_expansion[j - 1] * d[j - 1] + d[j - 2])
        n_ = (d[j] * b - 1) / c[j]
        if n_ == int(n_):
            p = ((n - n_ + 1) + sqrt((n - n_ + 1) ** 2 - 4 * n)) / 2
            q = ((n - n_ + 1) - sqrt((n - n_ + 1) ** 2 - 4 * n)) / 2
            if 0 < q < n and 0 < p < n and p == int(p) and q == int(q):
                return int(p), int(q)
    return None


print(Wiener(317940011, 77537081))
```

4

shestandsupinthegardenwhereshehasbeenworkingandlooksintothedistanceshehassensedashiftintheweatherthereisanothergustofwindabuckleofnoiseintheairandthetallcypressesswaysheturnsandmovesuphilltowardthehouseclimbingoveralowwallfeelingthefirstdropsofrainonherbarearmsshecrossestheloggiaandquicklyentersthehouse

```python
from math import gcd, sqrt
import numpy as np
import copy
from sage.all import inverse_mod
```

```python
def ElGamalDecrtpter(ciphertext: list, a: int, p: int):
    res = str()
    for pair in ciphertext:
        textplain = (pair[1] * inverse_mod(pow(pair[0], a), p)) % p
        textplain0 = textplain // (26 ** 2)
        textplain1 = (textplain - textplain0 * (26 ** 2)) // 26
        textplain2 = textplain - textplain0 * (26 ** 2) - textplain1 * 26
        res += (chr(97 + textplain0) + chr(97 + textplain1) + chr(97 + textplain2))
    return res


print(ElGamalDecrtpter(ciphertext, 7899, 31847))
```