chinesepid

November 30, 2021

```
In [1]: P.<x> = PolynomialRing(QQ)
```

```
In [2]: f = x^2+1
```

```
In [3]: g= x^2+2
```

```
In [4]: R=P.quotient_by_principal_ideal(ideal(f*g))
```

```
In [5]: a = R(x)
```

```
In [6]: a
```

```
Out[6]: xbar
```

```
In [7]: for j in range(12):
            print(j,a^j)
```

```
0 1
1 xbar
2 xbar^2
3 xbar^3
4 -3*xbar^2 - 2
5 -3*xbar^3 - 2*xbar
6 7*xbar^2 + 6
7 7*xbar^3 + 6*xbar
8 -15*xbar^2 - 14
9 -15*xbar^3 - 14*xbar
10 31*xbar^2 + 30
11 31*xbar^3 + 30*xbar
```

```
In [8]: S=P.quotient_by_principal_ideal(f)
```

```
In [9]: T=P.quotient_by_principal_ideal(g)
```

```
In [10]: for j in range(12):
             print(j,S(x)^j,T(x)^j,R(x)^j)
```

```
0 1 1 1
1 xbar xbar xbar
2 -1 -2 xbar^2
3 -xbar -2*xbar xbar^3
4 1 4 -3*xbar^2 - 2
5 xbar 4*xbar -3*xbar^3 - 2*xbar
6 -1 -8 7*xbar^2 + 6
7 -xbar -8*xbar 7*xbar^3 + 6*xbar
8 1 16 -15*xbar^2 - 14
9 xbar 16*xbar -15*xbar^3 - 14*xbar
10 -1 -32 31*xbar^2 + 30
11 -xbar -32*xbar 31*xbar^3 + 30*xbar
```

```
In [11]: gcd(f,g)

Out[11]: 1

In [12]: ?xgcd

In [13]: r,c,d = xgcd(f,g)

In [14]: r,c,d

Out[14]: (1, -1, 1)

In [15]: c*f+d*g

Out[15]: 1

In [16]: h1= 1

In [17]: h2=16

In [18]: u=d*g*h1+ c*f*h2; u

Out[18]: -15*x^2 - 14

In [23]: c,f,h2

Out[23]: (-1, x^2 + 1, 16)

In [19]: d,g,h1

Out[19]: (1, x^2 + 2, 1)

In [ ]:

In [ ]:

In [21]: nu = x^3; nu
```

```
Out[21]: x^3

In [22]: nu.parent()

Out[22]: Univariate Polynomial Ring in x over Rational Field

In [23]: rnu =R(nu); rnu

Out[23]: xbar^3

In [24]: rnu.inverse_of_unit()

Out[24]: 3/4*xbar^3 + 7/4*xbar

In [26]: xgcd(x^2+1,x^3)

Out[26]: (1, -x^2 + 1, x)

In [27]: xgcd(x^2+2,x^3)

Out[27]: (1, -1/4*x^2 + 1/2, 1/4*x)

In [28]: v = 1*g*x -1*f*x/4; v

Out[28]: 3/4*x^3 + 7/4*x

In [29]: v*x^3

Out[29]: 3/4*x^6 + 7/4*x^4

In [30]: R(v*x^3)

Out[30]: 1

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [31]: C=matrix(QQbar,[[3,1,1,1],[0,3,0,0],[0,0,3,0],[0,0,0,3]]); C

Out[31]: [3 1 1 1]
         [0 3 0 0]
         [0 0 3 0]
         [0 0 0 3]

In [32]: cc=C.characteristic_polynomial(); cc

Out[32]: x^4 - 12*x^3 + 54*x^2 - 108*x + 81
```

```
In [33]: cc.factor()

Out[33]: (x - 3)^4

In [ ]:

In [32]: cm =C.minimal_polynomial(); cm

Out[32]: x^2 - 6*x + 9

In [33]: cm.factor()

Out[33]: (x - 3)^2

In [34]: C*C

Out[34]: [9 6 6 6]
         [0 9 0 0]
         [0 0 9 0]
         [0 0 0 9]

In [36]: C^2-6*C

Out[36]: [-9  0  0  0]
         [ 0 -9  0  0]
         [ 0  0 -9  0]
         [ 0  0  0 -9]

In [36]: B=C-3*identity_matrix(4); B

Out[36]: [0 1 1 1]
         [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]

In [37]: B^2

Out[37]: [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]

In [38]: R.<x> =PolynomialRing(QQ)

In [39]: S = R.quotient_by_principal_ideal(x^2); S

Out[39]: Univariate Quotient Polynomial Ring in xbar over Rational Field with modulus x^2

In [43]: S(x).inverse_of_unit()
```

```
---------------------------------------------------------------------------

ArithmeticError                           Traceback (most recent call last)

/tmp/ipykernel_15611/1346418135.py in <module>
----> 1 S(x).inverse_of_unit()


/opt/math/sage/sage/local/var/lib/sage/venv-python3.8/lib/python3.8/site-packages/sage/
   1311             try:
   1312                 if not self.is_unit():
-> 1313                     raise ArithmeticError("element is not a unit")
   1314             except NotImplementedError:
   1315                 # if an element does not implement is_unit, we just try to


ArithmeticError: element is not a unit
```

In [44]:
```
S(x+1).inverse_of_unit()
```

Out[44]:
```
-xbar + 1
```

In [45]:

In [ ]: