CoCoNuT Assignment Two

January 15, 2015

1 More Sage

Groups of integers modulo n are created with the `Integers` command. You can then use the group name to map an integer into the group. You can use the usual addition and multiplication operations and `-a` and `1/a` for inversion in the additive resp. multiplicative groups.

```
sage: G=Integers(7)
sage: a=G(5)
sage: b=G(2)
sage: a+b
0
sage: -a
2
sage: 1/a
3
```

Vectors and matrices work as one would expect. To use matrices over a structure (such as a group), use the

```
MatrixSpace(struct,rows,cols)
```

constructor. The `^-1` operation inverts a matrix, if possible.

```
sage: v=vector([1,2,3])
sage: w=vector([1,1,0])
sage: v+w
(2, 3, 3)

sage: m=matrix([[1,2],[3,4]])
sage: n=identity_matrix(2)
sage: m+n
[2 2]
[3 5]
sage: m*n
[1 2]
[3 4]

sage: M=MatrixSpace(Integers(5),2,2)
sage: M([[2,3],[3,2]])+M([[4,2],[1,2]])
[1 0]
[4 4]

sage: M([[2,1],[1,2]])^-1
```

```
[4 3]
[3 4]
```

Elliptic curve groups are useful in number theory and cryptography. The basic idea is to start with the set of points (x, y) satisfying an equation of the form[1] $y^2 = x^3 + ax + b$ where all computation is done modulo a prime p. We then add a "point at infinity" $\mathcal{Z}$ as a neutral element, i.e. $\mathcal{Z} + \mathcal{Z} = \mathcal{Z}$ and $\mathcal{Z} + (x, y) = (x, y) = (x, y) + \mathcal{Z}$ for all points (x, y) on the curve. It turns out that this gives a group for a particular addition law. All we need to know for now is that these points form a group and the sage command to generate such a group is `EllipticCurve(P, [a, b])` where `P` is the structure of integers modulo p.

```
sage: E=EllipticCurve(Integers(7),[3, 1])
sage: E
Elliptic Curve defined by y^2 = x^3 + 3*x + 1 over
         Ring of integers modulo 7
```

Sage outputs elliptic curve points in the format `(x:y:1)` or `(0:1:0)` for the point at infinity (there are reasons for this format, which do not concern us here). To input the point at infinity we use `E(0)`.

```
sage: E([5, 1])
(5 : 1 : 1)
sage: E(0)
(0 : 1 : 0)
sage: E([5,1])+E([5,1])
(6 : 2 : 1)
sage: E([5,1])+E(0)
(5 : 1 : 1)
sage: -E([5,1])
(5 : 6 : 1)
sage: E([5,1])+E([5,6])
(0 : 1 : 0)
```

[1] For this to work, the discriminant $\Delta = 4a^3 + 27b^2$ must be nonzero. Also for simplicity, we assume $p > 3$ as the cases $p = 2, 3$ have some exceptions to the rules given here.

2 Assignment Two Questions

1. Consider the following groups:

   (a) The group of 3×2 matrices modulo 4 with matrix addition as the group operation.

   (b) The group of invertible 2×2 matrices modulo 3 with matrix multiplication as the group operation.

   (c) The group of permutations of the set $S = \{0, 1, 2, 3, 4\}$.

   For each group, find

   (a) The neutral element.

   (b) The group order.

   (c) Is the group Abelian?

   (d) A generator, if one exists.

   Answer:

   (a) The neutral element is the all-zero matrix (3×2 zeros). The group has $4^{3 \cdot 2} = 4096$ elements and is Abelian (addition modulo 4 is Abelian and you add component-wise). There is no single generator; each element has order at most 4 since for a matrix m, the element $4m = m+m+m+m$ must be all zeros.

   (b) The neutral element is the 2×2 identity matrix (1s on the diagonal, 0s elsewhere). There are $3^4 = 81$ matrices of the required size but not all are invertible - for a matrix $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ the condition for invertibility is that the determinant $ad - bc$ is nonzero. If $a \neq 0$ we get, for any values of b, c that two values of d (out of 3) yield a nonzero determinant — 36 matrices in all. If $a = 0$ then for $b = 0$ the determinant will be zero in any case; if $a = 0$ and $b \neq 0$ then d can be anything and there are exactly two (nonzero) values of c that make the determinant nonzero. This gives a further 12 matrices, making the group order 48. Matrix multiplication (except in the 1×1 case) is not Abelian. This implies that there cannot be any generators.

   (c) The neutral element is the identity map that sends each element of S to itself. The group order is $4! = 24$ and the group is not Abelian, for example $(ab)(bc) \neq (bc)(ab)$. Again, this means that the group cannot have a single generator.

2. (a) Find the order of the group of the elliptic curve given by $a = 7$ and $b = 3$ modulo $p = 1009$.

   Answer:

   980

(b) Is the above group Abelian? ✔

(c) Write a function that takes an elliptic curve point P and an integer n and adds P to itself n times, using only the group structure of the curve and commands from the previous assignment.

Answer:

```
def pmul(n, P):
    Q = P - P    # the neutral element of the curve
    while (n > 0):
        if (n % 2):
            Q = Q + P
        P = P + P
        n = n // 2
    return Q
```

(d) Use your algorithm to compute $512 \cdot (9064, 6692)$ on the curve defined by $a = 11, b = 4$ modulo $p = 10037$.

Answer:

(5496,7337)

3. Give an algorithm that takes a permutation as a $2 \times n$ matrix (as in the lecture notes) and outputs it as a list of disjoint cycles. Do not use sage's permutation group library.

Answer:

```
def IsValidArray(m):
    if len(m.rows()) <> 2: return false
    l1=list(m.row(0))
    l2=list(m.row(1))
    for i in range(len(l1)):
        if l1.count(l1[i])<>1 or l2.count(l1[i]) <>1: return False
    return True
#----------------------------------------------------------------
#----------------------------------------------------------------
def FromMattoCycles(m):
    if not IsValidArray(m):
```

```
        print "Sorry, invalid permutation matrix"
        return []
l1=list(m.row(0))
l2=list(m.row(1))
cycles=[]
checked=[]
ind=0
while ind < len(l1):
     curitem = l1[ind]
     if(checked.count(curitem) == 0):
               if l2[ind]==l1[ind]:
                       checked +=  [l1[ind]]
               else:
                       newitem = curitem
                       curcycle=[]
                       while True:
                             curcycle += [newitem]
                             newitem = l2[l1.index(newitem)]
                             if newitem == curitem: break
                       checked +=  curcycle
                       cycles  += [vector(curcycle)]
     ind += 1
return cycles
```

4. (a) Write a function that takes as input integers a, b, c and finds a solution to the equation $a^x = b \pmod{c}$. (You will have to do an exhaustive search given what you currently know.)

Answer:

```
def grp_DLP(a, b, c):
    b = b % c
    r = 1
    for x in range(c):
        if r = b:
            return x
        r = (r * a) % c
    raise Exception("No solution exists")
```

(b) Solve $34091202317940^x = 46461034929471 \bmod 61704897745301$.

Answer:

29393

(c) Adapt your algorithm from part a) above as necessary for the case of an elliptic curve group.

Answer:

```
def EC_DLP(P, Q, p):
    R = P - P    # == E(0)
    for x in range(p):
        if R == Q:
            return x
        R = R + P
    raise Exception("No solution exists")
```

(d) Solve the equation $x \cdot P = Q$ on the elliptic curve defined by $a = 5, b = 1$ modulo $p = 138451$ with $P = (74030, 23679)$ and $Q = (33643, 90060)$.

Answer:

63612