Homework8

November 3, 2022

[1]:

```
#Probelm 1 a
def isElliptic(E,p):
    A=E[0]
    B=E[1]
    if (4*pow(A,3)+27*B*B)%p==0 :
        return False
    else:
        return True
```

[2]:

```
#Problem 1 b
def pointOnCurve(P,E,p):
    x=P[0]
    y=P[1]
    A=E[0]
    B=E[1]
    if P==0:
        return True
    else:
        if (pow(x,3)+A*x+B)%p==(y*y)%p:
            return True
        else:
            return False
```

[3]:

```
#Problem 1 c
p=pow(2,256)-pow(2,32)-977
E=[0,7]
x=0x79BE667EF9DCBBAC55A06295CE870B07029BFCDB2DCE28D959F2815B16F81798
y=0x483ADA7726A3C4655DA4FBFC0E1108A8FD17B448A68554199C47D08FFB10D4B8
P=[x,y]
pointOnCurve(P,E,p)
```

[3]:

```
True
```

[0]:

```
```

[6]:

```
#Problem 2 a
def fastPower(a,n,m):
    amutiplier=1
    list=[]
    while n!=0:
        n0=n%2
        n=int((n-n0)/2)
        list.append(n0)
    for x in list:
        if x==0:
            amodulo=a%m
            a=(amodulo*amodulo)%m
        else:
            amodulo=a%m
            a=(amodulo*amodulo)%m
            amutiplier=(amutiplier*amodulo)%m
    return amutiplier

#I change the function a liitle
def findSquareRoot(a,p):
    list_answer=[]
    if p%4==3:
        if fastPower(a,(p-1)//2,p)==1:
            a=fastPower(a,(p+1)//4,p)
            if a>p-a:
                list_answer.append(p-a)
                list_answer.append(a)
                return list_answer
            else:
                list_answer.append(a)
                list_answer.append(p-a)
                return list_answer
        else:
            return False
    if p%4==1:
        m=(p-1)//2
        j=0
        a_initial=1
        while j<=m:
            if (j*j)%p==a:
                if j==0:
                    list_answer.append(j)
                    return list_answer
                if j>p-j:
                    list_answer.append(p-j)
                    list_answer.append(j)
                    return list_answer
                else:
                    list_answer.append(j)
                    list_answer.append(p-j)
                    return list_answer
            j=j+1
        return False

def generateElliptic(E,p):
    answerlist=['O']
    A=E[0]
    B=E[1]
    #When discriminant is not 0
    if isElliptic(E,p):
        x=0
        while x<p:
            a=pow(x,3)+A*x+B
            a=a%p
            if findSquareRoot(a,p)!=False:
                if len(findSquareRoot(a,p))==2:
                    answerlist.append([x,findSquareRoot(a,p)[0]])
                    answerlist.append([x,findSquareRoot(a,p)[1]])
                else:
                    answerlist.append([x,findSquareRoot(a,p)[0]])
                x=x+1
            else:
                x=x+1
    else:
        return "discriminant is 0"
    return answerlist
```

[7]:
```
#Problem 2 b
E=[5,12]
p=13
generateElliptic(E,p)
```

[7]:
```
['O', [0, 5], [0, 8], [2, 2], [2, 11], [7, 0], [10, 3], [10, 10]]
```

[0]:

[31]:
```
#Problem 3 a
def multInverse(a,p):
    # use p0 to save the ordinary prime
    p0=p
    b=a%p
    list3=[]
    s0=1
    s1=0
    t0=0
    t1=1
    while p%b!=0:
        r1=p%b
        q=p//b
        s2=s0-(q*s1)
        s0=s1
        s1=s2
        t2=t0-(q*t1)
        t0=t1
        t1=t2
        p=b
        b=r1
        list3=[b,s1,t1]
    return (p0+t1)%p0

def addPoints(P,Q,E,p):
    if P=='O' and Q=='O':
        return 'O'
    if P=='O' and Q!='O':
        return Q
    if P!='O' and Q=='O':
        return P
    x1=P[0]
    y1=P[1]
    x2=Q[0]
    y2=Q[1]
    A=E[0]
    B=E[1]
    if P!='O' and Q!='O':
        if P==Q:
            if (2*y1)%p==0:
                return 'O'
            else:
                lamda=((3*pow(x1,2)+A)*multInverse(2*y1,p))%p
                x3=(lamda*lamda-x1-x2)%p
                y3=(lamda*(x1-x3)-y1)%p
                return [x3,y3]
        if P!=Q:
            if x1==x2 and y1!=y2:
                return 'O'
            else:
                lamda=((y2-y1)*multInverse((x2-x1),p))%p
                x3=(lamda*lamda-x1-x2)%p
                y3=(lamda*(x1-x3)-y1)%p
                return [x3,y3]
```

[60]:

```
#Problem 3 b
def additionTable(E,p):
    points=generateElliptic(E,p)
    print("{:^10}".format("*"),end="")
    for i in range(len(points)):
        print("{:^10}".format(str(points[i])),end="")
    print()  #To change the line
    for i in range(len(points)):
        print("{:^10}".format(str(points[i])),end="")
        for j in range(len(points)):
            print("{:^10}".
 ↪format(str(addPoints(points[i],points[j],E,p))),end="")
        print()


E=[5,12]
p=13
additionTable(E,p)
```

```
    *         0       [0, 5]    [0, 8]    [2, 2]   [2, 11]    [7, 0]   [10, 3]
[10, 10]
    0         0       [0, 5]    [0, 8]    [2, 2]   [2, 11]    [7, 0]   [10, 3]
[10, 10]
  [0, 5]    [0, 5]   [10, 3]      0      [10, 10]   [7, 0]    [2, 2]   [2, 11]
[0, 8]
  [0, 8]    [0, 8]      0      [10, 10]   [7, 0]   [10, 3]   [2, 11]    [0, 5]
[2, 2]
  [2, 2]    [2, 2]   [10, 10]   [7, 0]   [10, 3]      0       [0, 5]    [0, 8]
[2, 11]
 [2, 11]   [2, 11]    [7, 0]   [10, 3]      0      [10, 10]   [0, 8]    [2, 2]
[0, 5]
  [7, 0]    [7, 0]    [2, 2]   [2, 11]    [0, 5]    [0, 8]      0      [10, 10]
[10, 3]
 [10, 3]   [10, 3]   [2, 11]    [0, 5]    [0, 8]    [2, 2]   [10, 10]   [7, 0]
0
 [10, 10]  [10, 10]   [0, 8]    [2, 2]   [2, 11]    [0, 5]   [10, 3]      0
[7, 0]
```

[33]:

```
#Probelm 4 a
def doubleAndAdd(P,n,E,p):
    res='O'
    while n>0:
        if n%2==1:
            res=addPoints(res,P,E,p)
        P=addPoints(P,P,E,p)
        n=n//2
    return res
```

[34]:

```
#Problem 4 b
n = 165717357988647532
p=pow(2,256)-pow(2,32)-977
E=[0,7]
x=0x79BE667EF9DCBBAC55A06295CE870B07029BFCDB2DCE28D959F2815B16F81798
y=0x483ADA7726A3C4655DA4FBFC0E1108A8FD17B448A68554199C47D08FFB10D4B8
P=[x,y]

doubleAndAdd(P,n,E,p)
```

[34]:

```
[51524656361346136203439460631008348936841590752868015863048885409956560359198,
 79203802285035089814150287439488171030915061256999835886514121114625556531371]
```

[0]:

[0]: