The Index Calculus Problem in $\mathbb{Z}_p^*$

Implementation

```
zipMult:=(A,B)->zip((x,y)->x*y,A,B):
listAdd:=A->foldr((x,y)->x+y,0,op(A)):

h_primPow:=proc(p,alpha,ans)
    if (alpha mod p = 0) then
        return h_primPow(p,alpha/p,ans+1):
    else
        return ans:
    end if:
end proc:

primPow:=(p,a)->h_primPow(p,a,0):

myFactor:=proc(inalpha,B::list) local i,x,listx,alpha:
    alpha:=inalpha:
    for i from 1 to nops(B) do
        x[i]:=primPow(B[i],alpha):
        alpha:=alpha/(B[i]^x[i]):
    end do:

    listx:=[seq(x[i],i=1..nops(B))]:

# if myFactor can't factor alpha in the base it will return NULL
    if(foldr((x,y)->x*y,1,op(zip((x,y)->x^y,B,listx)))=inalpha) then;
        return listx:
    else return NULL:
    end if:
end proc:

makeFun:=(var,coef)->listAdd(zipMult(var,coef)):

indexCalc:=proc(alpha,beta,p,sizeB) local B,i, temp, congEqns,mytry,var,back,n,x,logOf,listlogOf:
# solve log[alpha] beta mod p
    B:=[seq(ithprime(i),i=1..sizeB)]:

    var:=[seq(a[i],i=1..nops(B))]:

    for i from 1 to nops(B) do back[var[i]]:=B[i]: end do:

    congEqns:={}:
    n:=1:
    while (n<=nops(B)+10) do
        x:=rand(1..p)():
        temp:=myFactor(alpha&^x mod p,B):
        if (temp <> NULL) then
            congEqns:={makeFun(var,temp)=x} union congEqns:
            n:=n+1:
            print("n:",n);
        end if:
    end do:
```

```
    temp:=msolve(congEqns,p-1);
    print("minSolutions",nops(temp));

    while(nops(temp)<nops(B)) do
        n:=1:
        while (n<=10) do
            x:=rand(1..p)():
            temp:=myFactor(alpha&^x mod p,B):
            if (temp <> NULL) then
                congEqns:={makeFun(var,temp)=x} union congEqns:
                n:=n+1:
                print("n:",n);
            end if:
        end do:
        temp:=msolve(congEqns,p-1);
        print("minSolutions",nops(temp));
    end do;

    logOf:=table();

    for i from 1 to nops(B) do
        logOf[back[lhs(temp[i])]]:=rhs(temp[i]):
    end do:

    listlogOf:= [seq(logOf[B[i]],i=1..nops(B))]:

    while (true) do
        x:=rand(1..p)():
        mytry:=beta*alpha&^x mod p:
        temp:=myFactor(mytry,B);
        if (temp<>NULL) then
            return listAdd(zipMult(temp,listlogOf)) -x mod (p-1):
        end if:
    end do:
end proc:
```

Question 6.1

```
> aa:=indexCalc(alpha,beta,p,4); alpha^aa mod p = beta;
                                     aa := 7916
                                     356 = 356

> aa:=indexCalc(alpha,beta,p,4); alpha&^aa mod p = beta;
                                    aa := 232836
                                  248388 = 248388
```

Factor Base Study

For this study we let $p = 100000000379 = 2 \times 50000000189 + 1$ and use the primitive element $\alpha = 2$ in $\mathbb{Z}_p$ for the base in our discrete log. Picking $\beta = 356$ we time `indexCalc(alpha,beta,p,baseSize)` (which solves $\log_\alpha \beta \bmod (p)$) for increasing values of `baseSize`. The results are below:

$\|B\|$	10	20	40	80	160	320
Time	2506.895s	481.87s	112.40s	139.16s	426.98s	2747.04s

Table 1: Timing of `indexCalc`

Here's how it looks like in UNIX:

```
> p:=100000000379; beta:=356; alpha:=2;
> sizeB:=40:
> order(alpha,p)=(p-1);
                          100000000378 = 100000000378

> aa:=indexCalc(alpha,beta,p,sizeB):
> alpha&^aa mod p = beta;
                                   356 = 356
```

The Index Calculus Problem in $GF(p^k)^*$

Design

The index calculus algorithm in $GF(p^k)$ is similar to its analogue in $\mathbb{Z}_p$with the exception that we rewrite

$$\alpha^{x_j} \equiv p_1^{a_{1j}} p_2^{a_{2j}} \cdots p_B^{a_{Bj}}$$

as

$$x_j \equiv a_{1j} \log_\alpha p_1 + \cdots + a_{1B} \log_\alpha p_B \mod (p^k - 1)$$

and of course do all of our work with polynomials in $\mathbb{Z}_p[x]/g(x)$ (where $g(x)$ is an irreducible polynomial of degree k) instead of integer numbers. To be more explicit α is a primitive element in $\mathbb{Z}_p[x]/g(x)$ and our prime numbers (p_i's) are now just irreducible polynomials in $\mathbb{Z}_p[x]/g(x)$.

Implementation

```
zipPow:=(A,B,g,p)->zip((xx,yy)->(Powmod(xx,yy,g,x)mod p),A,B):
zipMult:=(A,B)->zip((x,y)->x*y,A,B):
foldMult:=(A,p)->foldr( (n,m)->funMult(n,m,p), 1, op(A)):
listAdd:=A->foldr((x,y)->x+y,0,op(A)):
funMult:=(f,g,p)->Expand(f*g) mod p;
funDiv:=(f,g,p)->Quo(f,g,x) mod p:
funMod:=(f,g,p)->Rem(f,g,x) mod p:

funcOrd := proc(f,g,p) local i,ft,gt:
    i:=0:
    ft:=f: gt:=g:
    while (true) do
        ft:=funMod(funMult(ft,f,p),g,p):
        i:=i+1:
        if (ft=f) then break: end if:
    end do:
    return i:
end proc:

h_primPow:=proc(f,alpha,p,ans)
if (Rem(alpha,f,x) mod p = 0) then
```

```
return h_primPow(f,Quo(alpha,f,x) mod p,p,ans+1):
else
return ans:
end if:
end proc:

primPow:=(f,alpha,p)->h_primPow(f,alpha,p,0):

makeB:=proc(n,p) local a,i,temp:
    a[1]:=x:
    i:=2;
    temp:=Nextprime(a[1],x) mod p:
    while (i<=n) do
        if ( coeff(temp,x,degree(temp,x))=1 ) then
            a[i]:=temp:
            i:=i+1;
            temp:=Nextprime(a[i-1],x) mod p:
        else
            temp:=Nextprime(temp,x) mod p:
        end if:

    end do:
    return [seq(a[i],i=1..n)]:
end proc:

myFactor:=proc(inalpha,B,g,p) local alpha, i, a, lista:
    alpha:=inalpha;
    for i from 1 to nops(B) do
        a[i]:=primPow(B[i],alpha,p);
        alpha:=funDiv(alpha,Powmod(B[i],a[i],g,x) mod p,p):
    end do:

   lista:=[seq(a[i],i=1..nops(B))]:

   if (foldMult(zipPow(B,lista,g,p),p)=inalpha) then
        return lista:
    else
        return NULL:
    end if:

end proc:

makeFun:=(var,coef)->listAdd(zipMult(var,coef)):


indexCalc:=proc(alpha,beta,g,p,k,sizeB) local B,i, temp, congEqns,mytry,var,back,n,y,logOf,listlogOf:
# solve log[alpha] beta mod p
    B:=makeB(sizeB,p):
    if(degree(B[sizeB],x)>=k) then return "error factor base too large"; end if;
    print("degree",degree(B[sizeB],x));
    var:=[seq(a[i],i=1..nops(B))]:

    for i from 1 to nops(B) do back[var[i]]:=B[i]: end do:
```

```
    congEqns:={}:
    n:=1:
    while (n<=nops(B)+20) do
        y:=rand(1..p^k-1)():
        temp:=myFactor( Powmod(alpha,y,g,x) mod p ,B,g,p):

        if (temp <> NULL) then
            congEqns:={makeFun(var,temp)=y} union congEqns:
            n:=n+1:
            print("n:",n);
        end if:
    end do:

    temp:=msolve(congEqns,p^k-1);
    print("minSolutions",nops(temp));

    while(nops(temp)<nops(B)) do
        n:=1:
        while (n<=10) do
            y:=rand(1..p^k-1)():
            temp:=myFactor( Powmod(alpha,y,g,x) mod p ,B,g,p):
            if (temp <> NULL) then
                congEqns:={makeFun(var,temp)=y} union congEqns:
                n:=n+1:
                print("n:",n);
            end if:
        end do:
        temp:=msolve(congEqns,p^k-1);
        print("minSolutions",nops(temp),nops(congEqns));
    end do;


    logOf:=table();

    for i from 1 to nops(B) do
        logOf[back[lhs(temp[i])]]:=rhs(temp[i]):
    end do:

    listlogOf:= [seq(logOf[B[i]],i=1..nops(B))]:

    while (true) do
        y:=rand(1..p^k-1)():
        mytry:=Rem(funMult(beta,Powmod(alpha,y,g,x) mod p,p),g,x) mod p;
        print(mytry);
        temp:=myFactor(mytry,B,g,p);
        if (temp<>NULL) then
            return listAdd(zipMult(temp,listlogOf)) -y mod (p^k-1):
        end if:
    end do:
end proc:
```

Testing

We check a couple smaller cases to make sure that this code is valid.

An example in $GF(2^4)$

The following code solves $\log_{x+1}(1+x+x^2)$ mod $(x+x+1)$

```
> p:=2: k:=4: beta:=1+x+x^2: g:=Nextprime(x^k,x) mod p: alpha:=x+1: sizeB:=4:

> aa:=indexCalc(alpha,beta,g,p,k,sizeB); Powmod(alpha,aa,g,x) mod p = beta;

                                         2              2
                                1 + x + x  = 1 + x + x
```

An example in $GF(3^4)$

$\log_{x+1}(1+2*x+x^2)$ mod (x^4+x+2)

```
> p:=3: k:=4: beta:=1+2*x+x^2: g:=Nextprime(x^k,x) mod p; alpha:=x+1: sizeB:=4:

> aa:=indexCalc(alpha,beta,g,p,k,sizeB); Powmod(alpha,aa,g,x) mod p = beta;
                                           2              2
                              1 + 2 x + x  = 1 + 2 x + x
```

The Big One

For $g = x^{50} + x^4 + x^3 + x^2 + 1$ which is irreducilbe in $\mathbb{Z}_2[x]$ and $\alpha = x+1$ a primitive element in $GF(2^{50}) \cong \mathbb{Z}_2[x]/g$ (easily found using theorem 5.8) the following code finds $\log_{x+1}(1+x+x^2)$ mod (g).

```
> p:=2: k:=50:
> beta:=1+x+x^2:
> g:=Nextprime(x^k,x) mod p:
> alpha:=x+1: sizeB:=200:
> aa:=indexCalc(alpha,beta,g,p,k,sizeB); Powmod(alpha,aa,g,x) mod p = beta;
                          aa := 200420092568080

                                    2              2
                           1 + x + x  = 1 + x + x
```