Baby-Step/Giant Step DL Algorithm
Expostion by William Gasarch

1 A Bad Way to Compute Discrete Log

Definition 1.1 The *Discrete Log Problem* is as follows. Let p be a prime and g be a generator. These are considered parameters. Given $y \in \{1, \ldots, p-1\}$ find x such that $y = g^x$.

Throughout this paper p is a prime and g is a generator for it.
Here is an algorithm for DL that is slow.

1. Input(y)
2. For $x = 0$ to $p - 1$
   (a) Compute g^x.
   (b) If $g^x = y$ then output(x) and STOP

Note that in the worst case this could take roughly p steps. But worse than that—its average case also is not very good (we will not go into this). What we mean is that in MANY cases it will take a lot of time.

2 Fast(er) Algorithm for Calculating Discrete Logs: the Baby-Step / Giant-Step Algorithm

Can we do better than p steps? How much better? Recall that we think of p as large and $\log p$ as small. Hence if DL could be solved in $\log p$ steps even $(\log p)^2$ steps or even $(\log p)^{10}$ steps that would be fast. Alas we cannot do that. However, we can do DL in roughly $\sqrt{p}$ steps.

First we do some informal reasoning. Define $m = \lceil \sqrt{p} \rceil$. Thus whenever we do something m times, we are doing it roughly $\sqrt{p}$ times.

Given c we want to find x such that $c = g^x$.

KEY IDEA ONE: Think of dividing x by m. Thus $x = qm + r$. Since we divided by m we know that $0 \leq r \leq m$. Since $m \times m$ is roughly p we can assume $0 \leq q \leq m$.

KEY IDEA TWO: Lets say we GUESS what r is. If we are CORRECT then there is some q such that

$c = g^{qm+r}$
$c = g^{qm} g^r$
$c^{-1} g^r = g^{-qm}$

AND recall that $0 \leq r \leq m$. So if we guess r correctly then $c \times g^{-r}$ has to be of the form g^{-qm} for some q.

KEY IDEA THREE: Prepossess! Before doing any discrete logs we will create tables of some powers of g. We will certainly have tables of $g^0, g^1, g^2, \ldots, g^m$. (actually this is not really needed since g^i can be computed in roughly $\log i$ steps which is fast) So if the DL is $\leq m$ we will be able to find it. Can't prepossess too much since that would take too much time.

Of more importance: We preprocess and SORT g^{-qm} for $0 \leq q \leq m$. We give an example below during the procedure.

KEY: If we want to know if (say) 19 is of the form 2^{-6q} we can do a binary search on the sorted table and find out quickly.

We will now do the PREPOSSESS and ALGORITHM.

We will do a running example with $p = 29$, $m = 6$, and $g = 2$. The numbers given are not true.

PREPROCESSING

1. For $r = 0$ to m compute g^{-r} and put in a table.

   The table will look like this TABLE r:

r	2^{-r}
0	1
1	4
2	17
3	9
4	27
5	12
6	11

   No need to sort this table as we will be using it to just compute cg^{-r}. Note that we first computer g^{-1} and then raise it to powers.

2. For $i = 0$ to m compute g^{-im}. Put them in a table that looks like this:

q	2^{-6q}
0	1
1	19
2	3
3	22
4	8
5	18
6	10

   But we then SORT THIS ON THE SECOND COLUMN to have TABLE qm:

q	2^{-6q}
0	2
2	3
4	8
6	10
5	18
1	19
3	22

SO, we have both of these tables.
ALGORITHMS

1. Input(c) (We seek x such that $c = g^x$. We think of x as $qm+r$ with $0 \leq q, r \leq m$ and hence we seek q, r.)

2. For $r = 0$ to m we TRY to see if r is a good guess for r: (Comment- if r is correct then $c = g^{qm+r} = g^{rm}g^r$, so $cg^{-r} = g^{-qm}$)

   (a) Look on TABLE r to find g^{-r}. Multiply it by c.

   (b) By using Binary Search try to find cg^{-r} on TABLE qm.
   IF find it then find the q such that $cg^{-r} = g^{-qm}$. OUTPUT $x = qm + r$.
   IF do not find it then FINE, just go to next value of r.

How long does this take. The preprocessing takes roughly $2m$ steps. But you may not want to really count that since you do it once and may use the data over and over again.

The algorithm also takes roughly m iterations, each one takes $\log p$ steps. So this takes $m \log p$ steps.

Since that m is roughly $\sqrt{p}$. So the entire algorithm takes $\sqrt{(p)} \log p$ steps.

3 What of it?

So DL can be solved in $\sqrt{p}$ steps. This is still not that fast. However, its far faster than we thought. Lets say that 2^{100} is too big for a computer to do that many steps.

OLD MENTALITY: Need p such that $\log p$ is small and p is large. Take $p \sim 2^{512}$.

NEW MENTALITY: Need p such that $\log p$ is small and $\sqrt{p}$ is large. Take $p = 2^{1024}$.

This is a warning. Clever math has not yet made us think that DL is easy (and hence Diffie Helman is breakable but this clever math has made us increase our parameter.

Is this algorithm actually used? The time bound $\sqrt{p}$ is not so bad. But the space bound of $\sqrt{p}$ is bad. There are algorithms with the same time bound but a much better space bound, and they are used.

4 Can we Save on Space? Part I

Ignoring log factors, the Baby Step/Giant Step algorithm takes $O(\sqrt{p})$ space and $O(\sqrt{p})$ time. The time, while a problem, perhaps can be dealt with by using faster machines or parallel machines. But is there a way to use less space. We first eliminate one of the tables. Intuitively we don't need the g^{-r} table since we can always compute $g^{-r}r$. We do this formally:
PREPROCESSING

1. For $i = 0$ to m compute g^{-im}. Put them in a table that looks like this:

q	2^{-mq}
0	1
1	19
2	3
3	22
4	8
5	18
6	10

But we then SORT THIS ON THE SECOND COLUMN to have TABLE qm:

q	2^{-6q}
0	2
2	3
4	8
6	10
5	18
1	19
3	22

SO, we have one table, the qm table.
ALGORITHMS

1. Input(c) (We seek x such that $c = g^x$. We think of x as $qm+r$ with $0 \leq q, r \leq m$ and hence we seek q, r.)

2. For $r = 0$ to m we TRY to see if r is a good guess for r: (Comment- if r is correct then $c = g^{qm+r} = g^{rm}g^r$, so $cg^{-r} = g^{-qm}$)

(a) Compute g^{-1}. Then, by using repeated squaring (which is fast!) computer Compute $(g^{-1})^r = g^{-r}$. Multiply it by c. We now have cg^{-r}.

(b) By using Binary Search try to find cg^{-r} on TABLE qm.
IF find it then find the q such that $cg^{-r} = g^{-qm}$. OUTPUT $x = qm + r$.
IF do not find it then FINE, just go to next value of r.

How long does this take. The preprocessing takes roughly m steps. But you may not want to really count that since you do it once and may use the data over and over again.

The algorithm also takes roughly m iterations, each one takes $2 \log p$ steps. So this takes $2m \log p$ steps.

Since that m is roughly $\sqrt{p}$. So the entire algorithm takes $2\sqrt{(p)} \log p$ steps.

5 Can we Save on Space? Part II

The big space usage, in fact the only space usage, is in Table qm. This table has m elements on it. What if m is NOT $\sqrt{p}$. Lets just keep it at m for now. We revisit the KEY IDEAS.

KEY IDEA ONE: Think of dividing x by m. Thus $x = qm + r$. Since we divided by m we know that $0 \leq r \leq m$. Since $x \leq p$ we know that $0 \leq q \leq \frac{p}{m}$.

KEY IDEA TWO: Lets say we GUESS what r is. If we are CORRECT then there is some q such that

$c = g^{qm+r}$

$c = g^{qm} g^r$

$c^{-1} g^r = g^{-qm}$

AND recall that $0 \leq r \leq m$. So if we guess r correctly then $c \times g^{-r}$ has to be of the form g^{-qm} for some $q \leq \frac{p}{m}$.

KEY IDEA THREE: PREPROCESS g^{-qm} for $0 \leq q \leq \frac{p}{m}$. This will take SPACE $\frac{p}{m}$.

KEY IDEA FOUR: Take m LARGE so that TABLE qm is SMALL. We will pick m later.

We now give the entire algorithm with m as a parameter.

WHAT IS BELOW WOULD BE A FINE ANSWER ON THE MIDTERM.

PREPROCESSING

1. For $i = 0$ to $\frac{p}{m}$ compute g^{-im}. Put them in a table and sort on g^{-im} as we've done before. Call this table TABLE qm.

SO, we have one table, the qm table.

ALGORITHMS

1. Input(c) (We seek x such that $c = g^x$. We think of x as $qm + r$ with $0 \leq r \leq m$ and $0 \leq q \leq \frac{p}{m}$. Hence we seek q, r.)

2. For $r = 0$ to m we TRY to see if r is a good guess for r: (Comment- if r is correct then $c = g^{qm+r} = g^{rm}g^r$, so $cg^{-r} = g^{-qm}$)

   (a) Compute g^{-1}. Then, by using repeated squaring (which is fast!) computer Compute $(g^{-1})^r = g^{-r}$. Multiply it by c. We now have cg^{-r}.

   (b) By using Binary Search try to find cg^{-r} on TABLE qm.
   IF find it then find the q such that $cg^{-r} = g^{-qm}$. OUTPUT $x = qm + r$.
   IF do not find it then FINE, just go to next value of r.

How long does this take. The preprocessing takes roughly $\frac{p}{m}$ steps. But you may not want to really count that since you do it once and may use the data over and over again.

The algorithm also takes roughly m iterations, each one takes $2 \log p$ steps. So this takes $2m \log p$ steps.

SO, ignoring log factors and constant factors:
TIME: m steps
SPACE: p/m space.
Take $m = p^{9/10}$ to get
TIME: $p^{9/10}$ (WORSE than before)
SPACE: $p^{1/10}$ (BETTER THAN BEFORE)