Beating the Birthday Paradox: Order Computations in Generic Groups

Andrew V. Sutherland

Massachusetts Institute of Technology

April 20, 2007

Outline

1. Introduction
2. Algorithms
   - Primorial Steps
   - Multi-Stage Seive
3. Results

Outline

1. Introduction
2. Algorithms
   - Primorial Steps
   - Multi-Stage Seive
3. Results

Hard Problems vs Easy Problems

Hard Problems

- Factoring Integers: $N = pq$
- Discrete Logarithm: $DL(\alpha, \beta)$
- Order Computation: $|\alpha|$

Easy Problems

- Multiplying: $pq = N$
- Exponentiating: $\alpha^k = \beta$
- Fast Order Computation: $|\alpha|$ given $\alpha^E = 1_G$

Generic Groups and Black Boxes

Generic Groups

- Isomorphic groups are equivalent.
- Algorithms work in any finite group.
- Complexity measured by group operations.

Black Boxes

- Opaque representation.
- Unique identifiers.
- Good software engineering.

Order Computations

Applications

- Black-box group recognition.
- Abelian group structure.
- Factoring.

Order Computation Theorem

The total cost of all order computations is at most

$$(1 + o(1))\, T(\lambda(G)),$$

where $T(N)$ is the cost of computing $|\alpha| = N$.

Order Computations

Applications

- Black-box group recognition.
- Abelian group structure.
- Factoring.

Order Computation Theorem

The total cost of all order computations is at most

$$(1 + o(1))\, T(\lambda(G)),$$

where $T(N)$ is the cost of computing $|\alpha| = N$.

Order Computations

Problem

- Find the least positive N such that $\alpha^N = 1_G$.
- No upper bound on N.
- $\alpha^k = \alpha^j \iff k \equiv j \mod N$.

Solutions

- Birthday paradox.
- Shanks baby-steps giant-steps $\approx 2\sqrt{2N}$.
- Pollard rho method $\approx \sqrt{2\pi N}$.

Order Computations

Problem

- Find the least positive N such that $\alpha^N = 1_G$.
- No upper bound on N.
- $\alpha^k = \alpha^j \quad \Longleftrightarrow \quad k \equiv j \mod N$.

Solutions

- Birthday paradox.
- Shanks baby-steps giant-steps $\approx 2\sqrt{2N}$.
- Pollard rho method $\approx \sqrt{2\pi N}$.

Lower Bounds?

Babai

Exponential lower bound in black-box groups.

Shoup

$\Omega(\sqrt{N})$ lower bound for discrete logarithm in generic groups.

Terr

$\sqrt{2N}$ lower bound on addition chains.

Birthday Paradox

$\sqrt{(2\log 2)N}$ lower bound for a random algorithm (?)

Outline

1. Introduction
2. **Algorithms**
   - Primorial Steps
   - Multi-Stage Seive
3. Results

The Basic Idea

What if we knew $|\alpha|$ were odd?

What if we knew $|\alpha| \perp 6$?

What if we knew $|\alpha| \perp \prod_{p \leq L} p$?

The Basic Idea

What if we knew $|\alpha|$ were odd?

What if we knew $|\alpha| \perp 6$?

What if we knew $|\alpha| \perp \prod_{p \leq L} p$?

The Basic Idea

What if we knew $|\alpha|$ were odd?

What if we knew $|\alpha| \perp 6$?

What if we knew $|\alpha| \perp \prod_{p \leq L} p$?

Key Fact #1

Orders Can Be Factored

For any $\beta = \alpha^k$:

$$|\beta| = N_1 \quad \text{and} \quad |\alpha^{N_1}| = N_2 \qquad \Longrightarrow \qquad |\alpha| = N_1 N_2.$$

Primorial Steps Algorithm

1. Let $E = \prod p^h$ for $p \leq L$, $p^h \leq M < p^{h+1}$, and let $P = \prod p$.
2. Compute $\beta = \alpha^E$.
3. Use baby-steps $\perp P$ and giant-step multiples of P to find $N_1 = |\beta|$.
4. Use a fast order algorithm to find $N_2 = |\alpha^{N_1}|$ given E.
5. Return $N_1 N_2$.

Primorials

w	p_w	P_w	$\phi(P_w)$	$\phi(P_w)/P_w$	$P_w/\phi(P_w)$
1	2	2	1	0.5000	2.0000
2	3	6	2	0.3333	3.0000
3	5	30	8	0.2667	3.7500
4	7	210	48	0.2286	4.3450
5	11	2310	480	0.2078	4.8125
6	13	30030	5760	0.1918	5.2135
7	17	510510	92160	0.1805	5.5394
8	19	9699690	1658880	0.1710	5.8471
9	23	223092870	36495360	0.1636	6.1129
10	29	6469693230	1021870080	0.1579	6.3312

Table: The First Ten Primorials

Complexity

Worst Case

$$O\left(\sqrt{\frac{N}{\log\log N}}\right)$$

Best Case

$$O(L)$$

Average Case

???

Complexity

Worst Case

$$O\left(\sqrt{\frac{N}{\log\log N}}\right)$$

Best Case

$$O(L)$$

Average Case

???

Complexity

Worst Case

$$O\left(\sqrt{\frac{N}{\log\log N}}\right)$$

Best Case

$$O(L)$$

Average Case

???

Key Fact #2

Numbers Have Smooth Parts and Coarse Parts

Let $\sigma_y(x)$ be the largest y-smooth divisor of x.
Define $\kappa_y(x) = x/\sigma(x)$ to be the y-coarse part of x,

$$x = \sigma_y(x)\kappa_y(x).$$

Typically $y = x^{1/u}$.

How Big is the Coarse Part?

Few numbers are y-smooth, but for most numbers, $\kappa_y(x) \ll x$.

The Multi-Stage Sieve

Factoring in the Dark

Problem: We don't know any factors until we find them all.

Play the Odds

Solution: Alternate sieving and searching until we do.

The Multi-Stage Sieve

Factoring in the Dark

Problem: We don't know any factors until we find them all.

Play the Odds

Solution: Alternate sieving and searching until we do.

How Numbers Are Made

Random Bisection Model

How to generating random integers with known factorizations (Bach).

Distribution of Smooth Numbers

$$\Psi(x, x^{1/u}) \sim \rho(u)x.$$

Distribution of Semismooth Numbers

Semismooth probability function: $G(r, s)$.

Complexity

Median Complexity

$O(N^{0.344})$, assuming uniform distribution of $N = |\alpha|$.
Typically better.

More generally...

$$Pr\left[T(N) \leq cN^{1/u}\right] \geq G(1/u, 2/u)$$

Semismooth and Smooth Probabilities

u	$G(1/u, 2/u)$	$\rho(u)$
2.2	0.8958	0.2203
2.5	0.7302	0.1303
2.9	0.5038	0.0598
3.0	0.4473	0.0486
4.0	0.0963	0.0049
6.0	1.092e-03	1.964e-05
8.0	3.662e-06	3.232e-08
10.0	5.382e-09	2.770e-11

Outline

1. Introduction
2. Algorithms
   - Primorial Steps
   - Multi-Stage Seive
3. **Results**

What does it all mean?

Reference Problem for Generic Algorithms - Ideal Class Groups

Compute the ideal class group of $\mathbb{Q}[\sqrt{D}]$ for negative D.
Interesting problem for number theorists, and cryptographers.

Comparison to Generic Algorithms: $D = -4(10^{30} + 1)$

Rho algorithm: 200 million gops, 15 days (Teske 1998).
Multi-stage sieve: 200,000 gops, 6 seconds.

Comparison to Non-Generic Algorithms: $D = -4(10^{54} + 1)$

Subexponential MPQS algorithm: 9 hours (Buchmann 1999).
Multi-stage sieve: 800,000 gops, 25 seconds.

What does it all mean?

Reference Problem for Generic Algorithms - Ideal Class Groups

Compute the ideal class group of $\mathbb{Q}[\sqrt{D}]$ for negative D.
Interesting problem for number theorists, and cryptographers.

Comparison to Generic Algorithms: $D = -4(10^{30} + 1)$

Rho algorithm: 200 million gops, 15 days (Teske 1998).
Multi-stage sieve: 200,000 gops, 6 seconds.

Comparison to Non-Generic Algorithms: $D = -4(10^{54} + 1)$

Subexponential MPQS algorithm: 9 hours (Buchmann 1999).
Multi-stage sieve: 800,000 gops, 25 seconds.

What does it all mean?

Reference Problem for Generic Algorithms - Ideal Class Groups

Compute the ideal class group of $\mathbb{Q}[\sqrt{D}]$ for negative D.
Interesting problem for number theorists, and cryptographers.

Comparison to Generic Algorithms: $D = -4(10^{30} + 1)$

Rho algorithm: 200 million gops, 15 days (Teske 1998).
Multi-stage sieve: 200,000 gops, 6 seconds.

Comparison to Non-Generic Algorithms: $D = -4(10^{54} + 1)$

Subexponential MPQS algorithm: 9 hours (Buchmann 1999).
Multi-stage sieve: 800,000 gops, 25 seconds.

Recipe for Subexponential Algorithms

Lottery Problem

Given a random sequence of problems, how long does it take to solve one? You only have to win once.

Subexponential Approach

Choose u so that $cN^{1/u}G(1/u, 2/u) \approx 1$.
Running time is "aysmptotically" $L(1/2, \sqrt{2})$ or $L(1/2, 1)$.

Generic Solution

Works for any problem that can be reduced to random order computations.

Subexponential Result

Example: $D = -(10^{80} + 1387)$

Computed using 2×10^9 gops (u = 6.7).
$L(1/2, 1)$ bound would predict 10^{13} gops.

Points to Ponder...

What is the right bound for order computation?

Known: $\Omega\left(N^{1/3}\right)$ $O\left(\sqrt{N/\log\log N}\right)$

Unknown: $\Omega\left(N^{1/2}/\log N\right)$? $O\left(\sqrt{N/logN}\right)$?

Space efficient worst case?

$o\left(\sqrt{N}\right)$ algorithm using polylogarithmic space?

Subexponential Applications

Which problems can be reduced to random order computations in finite groups?