Integer Factorization

Alex Sundling

Algorithms

- Shanks' square forms algorithm
- Williams' p+1
- Quadratic Sieve
- Dixon's Random Squares Algorithm

Shanks' Square Forms

- Created by Daniel Shanks as an improvement on Fermat's factorization method
- Fermat's method
  - Tries to find integers x and y such that $x^2 - y^2 = N$
- Shanks' method
  - Tries to find integers x, y such that $x^2 \equiv y^2 \pmod{n}$
  - GCD(x-y, N) has a great chance of giving you a factor of N

Shanks' Square Forms Algorithm

- First Step
  - Initialize Values

$$P_0 = \lfloor \sqrt{N} \rfloor, Q_0 = 1, Q_1 = N - P_0^2.$$

- Second step
  - Repeat the following until Q_i is a perfect square

$$b_i = \left\lfloor \frac{\lfloor \sqrt{N} \rfloor + P_{i-1}}{Q_i} \right\rfloor, P_i = b_i Q_i - P_{i-1}, Q_{i+1} = Q_{i-1} + b_i (P_{i-1} - P_i)$$

Shanks' Square Forms Algorithm

- Third Step
  - Initialize values

$$b_0 = \left\lfloor \frac{\lfloor \sqrt{N} \rfloor - P_{i-1}}{\sqrt{Q_i}} \right\rfloor, P_0 = b_0\sqrt{Q_i} + P_{i-1}, Q_0 = \sqrt{Q_i}, Q_1 = \frac{N - P_0^2}{Q_0}$$

- Fourth Step
  – Repeat this step until $P_{i+1} = P_i$

$$b_i = \left\lfloor \frac{\lfloor \sqrt{N} \rfloor + P_{i-1}}{Q_i} \right\rfloor, P_i = b_i Q_i - P_{i-1}, Q_{i+1} = Q_{i-1} + b_i(P_{i-1} - P_i)$$

Shanks' Square Forms Algorithm

- Final Step
  - Take the gcd(N, P_i) and this will be a factor of N

Shanks' Square Forms Algorithm

- N = 11111
  $P_0 = 105$ $Q_0 = 1$ $Q_1 = 86$
  $P_1 = 67$ $Q_1 = 86$ $Q_2 = 77$
  $P_2 = 87$ $Q_2 = 77$ $Q_3 = 46$
  $P_3 = 97$ $Q_3 = 46$ $Q_4 = 37$
  $P_4 = 88$ $Q_4 = 37$ $Q_5 = 91$
  $P_5 = 94$ $Q_5 = 91$ $Q_6 = 25$
  Here Q_6 is a perfect square
  $P_0 = 104$ $Q_0 = 5$ $Q_1 = 59$
  $P_1 = 73$ $Q_1 = 59$ $Q_2 = 98$
  $P_2 = 25$ $Q_2 = 98$ $Q_3 = 107$
  $P_3 = 82$ $Q_3 = 107$ $Q_4 = 41$
  $P_4 = 82$
  Here $P_3 = P_4$
  gcd(11111, 82) = 41, which is a factor of 11111.

$$b_i = \left\lfloor \frac{\lfloor\sqrt{N}\rfloor + P_{i-1}}{Q_i} \right\rfloor, P_i = b_iQ_i - P_{i-1}, Q_{i+1} = Q_{i-1} + b_i(P_{i-1} - P_i)$$

$$b_i = \left\lfloor \frac{\lfloor\sqrt{N}\rfloor + P_{i-1}}{Q_i} \right\rfloor, P_i = b_iQ_i - P_{i-1}, Q_{i+1} = Q_{i-1} + b_i(P_{i-1} - P_i)$$

Shanks Square Forms

- Strengths
  - It is a small algorithm that can be implemented on devices that have memory constraints
- Weaknesses
  - Not made for big numbers

Williams’s p+1

- Invented by H.C. Williams in 1982
- Similar to Pollard’s p-1 algorithm
- Parameters
  - N, an Integer to factor
  - A, an number greater than 2

William's p+1 Algorithm

- First step
  - Initialize values
    - m = 3
    - $V_0 = 2$
    - $V_1 = A$
    - $V_i = [A*V_{i-1} - V_{i-2}] \bmod N$

Williams' p+1 Algorithm

- Second step
  - Keep computing the next V_i in the sequence until $i = m!$
- Third step
  - Get the gcd of (V_i-2, n)
  - If the gcd doesn't equal 1 or N go to Step 4 otherwise increase m by 1 and go back to the Second step
- Fourth step
  - Check to make sure m! is a multiple of p+1
    - If it is the gcd(V_i-2, n), is a factor of N

Williams' p+1

- Notes
  - Jacobi Symbol
    - $D = A^2 - 4$
    - We want $(D/p) = -1$
    - Don't know p beforehand though
  - Prime + 1 (p+1)
    - $(D/p) = -1$
    - Works best when this is the case
  - Prime – 1 (p-1)
    - $-(D/p) = +1$
    - Algorithm will turn into a slow version of Pollard's p-1 algorithm

William’s p+1

- Strengths
  - Works very well when the number p+1
- Weaknesses
  - Doesn’t work well when the number is p-1

Quadratic Sieve

- Invented by Carl Pomerance in 1981
  - An improvement to Dixon's factorization method
- Second behind general number field sieve for fastest method in integer factorization
- Basic idea
  - Tries to set up a congruence of squares modulo N to find a factor

Quadratic Sieve Algorithm

- Parameters
  - N, number to be factored
  - t, number of primes to populate the factor base with
- Split up into two parts
  - Data collection phase
  - Data processing phase

Quadratic Sieve

- Step 1
  - Select a factor base of size t
    - -1 always included in factor base
    - Then next primes are chosen based on whether N is a quadratic residue modulo p
      - Legendre(N, prime number) == 1
    - Example: factor base when N=24961 when t = 6 would be {-1, 2, 3, 5, 13, 23}
(7, 11, 17 and 19) have been omitted from it because for each one Legendre(N, prime number) == -1
- Step 2
  - Compute

$$m = \lfloor \sqrt{n} \rfloor$$

Quadratic Sieve

- Step 3
  - Collect (t + 1) pairs
    - Compute b
      - $b = q(x) = (x + m)^2 - N$
        - x values are go in this order
        - (0, 1, -1, 2, -2, 3, -3. ....,)
    - Next check to make sure b is p_t-smooth using Trial Division if not go to next x and compute b again

Quadratic Sieve

- Step 3 (continued)
  - Do a prime factorization of b
  - Create a vector e_i that consists of all the exponents of the prime factorization of b

$$b = \prod_{j=1}^{t} p_j^{e_{ij}}$$

- Next you will create another vector v_i and put in the values from e_i mod 2
  - Vector will consist of only 1s and 0s

Quadratic Sieve

- Example ($N = 24961$, $t = 6$)
  - Factor base = {-1, 2, 3, 5, 13, 23}
  - $x = 0$, $q(x) = -312$
  - $-1^1 * 2^3 * 3^1 * 5^0 * 13^1 * 23^0$
  - $e_i = <1, 3, 1, 0, 1, 0>$
  - $v_i = <1, 1, 1, 0, 1, 0>$

Quadratic Sieve

i	x	$q(x)$	factorization of $q(x)$	a_i	v_i
1	0	-312	$-2^3 \cdot 3 \cdot 13$	157	$(1,1,1,0,1,0)$
2	1	3	3	158	$(0,0,1,0,0,0)$
3	-1	-625	-5^4	156	$(1,0,0,0,0,0)$
4	2	320	$2^6 \cdot 5$	159	$(0,0,0,1,0,0)$
5	-2	-936	$-2^3 \cdot 3^2 \cdot 13$	155	$(1,1,0,0,1,0)$
6	4	960	$2^6 \cdot 3 \cdot 5$	161	$(0,0,1,1,0,0)$
7	-6	-2160	$-2^4 \cdot 3^3 \cdot 5$	151	$(1,0,1,1,0,0)$

- Step 4
  - Using Linear Algebra find a subset that when added together equals 0
    - Using binary logic
      - $1 + 1 = 0$
      - $1 + 0 = 1$
  - Example: $v_1 + v_2 + v_5 = 0$
  - Now we know our subset

Quadratic Sieve

- Step 5
  - Compute x

$$x = \prod_{i \in T} a_i \bmod n.$$

- Example:
  – ($a^1*a^2*a^5$ mod n)

Quadratic Sieve

- Step 6
  - Calculate

$$l_j = (\sum_{i \in T} e_{ij})/2$$

- Example
  - $V_1 = <1, 3, 1, 0, 1, 0>$
  - $V_2 = <0, 0, 1, 0, 0, 0>$
  - $V_5 = <1, 3, 2, 0, 1, 0>$
  - Add up each Column and divide by 2
  - $<1, 3, 2, 0, 1, 0>$
  - $l1 = 1, l_2 = 3, l_3 = 2, l_4 = 0, l_5 = 1, l_6 = 0$

Quadratic Sieve

- Step 7
  - Calculate

$$y = \prod_{j=1}^{t} p_j^{l_j} \bmod n$$

- Example
  - $y = -1^1 * 2^3 * 3^2 * 5^0 * 13^1 * 23^0$

Quadratic Sieve

- Step 8
  - If $x \equiv \pm y \pmod{n}$ holds true then go back to step 4 and find another subset
- Step 9
  - Compute d = gcd(x – y, n)
  - d is a factor of n!

Quadratic Sieve

- Strengths
  - It's very good with large numbers
    - Still fastest for integers under 100 decimal digits
- Weaknesses
  - Overkill on small numbers
  - Continued Fractions is a better algorithm for numbers up until around 30 decimal digits

Dixon’s Random Square Algorithm

- Invented by John Dixon, a mathematician at Carleton University
- Works somewhat like Fermat’s factorization algorithm
- Basic Idea
  - Based on finding a congruence of squares modulo N

Dixon’s Algorithm

- Parameters
  - N, number to be factored
  - t, smoothness bound and number of primes
- Since Quadratic Sieve is an improvement to Dixon’s Algorithm basically everything is calculated the same way except for how b is calculated

Dixon's Algorithm

- Quadratic Sieve
  - b is calculated using this equation
    - $b = Q(x) = (x + m)^2 - n$
- Dixon's Algorithm
  - b is calculated by choosing an an a_i value at random and then seeing if b is p_t-smooth
  - $B = (a_i^2) \bmod n$

Dixon’s Algorithm

- Strength
  - Was the reason Quadratic Sieve was invented

Comparisons

- Quadratic Sieve is the best overall
- If you have memory constraints Shanks is probably the best Algorithm to use

Critique

- The three algorithms that use square root can't handle big numbers
- QS could be faster, by using faster Prime and GCD functions.

Future Work

- Add a Big Integer class to QS
- Turn the data gathering part of QS into a parallel process.

Questions?