Computational Number Theory and Algebra June 20, 2012

Lecture 18

Lecturers: Markus Bläser, Chandan Saha *Scribe: Chandan Saha*

In the last class, we have discussed the Pollard-Strassen's integer factoring algorithm that has a running time of $\tilde{O}(s(N)^{1/2})$ bit operations, where $s(N)$ is the smallest prime factor of the input (composite) number N. This algorithm has a space complexity of $\tilde{O}(s(N)^{1/2})$ bits. In today's class, we will discuss a heuristic, randomized algorithm - namely, *Pollard's rho method*, whose 'believed' expected time complexity is $\tilde{O}(s(N)^{1/2})$ bit operations, and which uses only $\tilde{O}(1)$ space! Although, a rigorous analysis of Pollard's rho method is still open, in practice it is observed that Pollard's rho method is more efficient than Pollard-Strassen factoring algorithm (partly because of the low memory usage of the former). Today, we will cover the following topics:

- Pollard's rho method, and
- Introduction to smooth numbers.

1 Pollard's rho method

Let N be a number that is neither a prime nor a perfect power, and p the smallest prime factor of N. If we pick numbers $x_0, x_1, x_2, \ldots$ from $\mathbb{Z}_N$ uniformly, independently at random then after at most $p+1$ such pickings we have (for the first time) two numbers x_i, x_s with $i < s$ such that $x_i = x_s \mod p$. Since N is not a perfect power, there is another prime factor $q > p$ of N. Since the numbers x_i and x_s are randomly chosen from $\mathbb{Z}_N$, by the Chinese remaindering theorem, $x_i \neq x_s \mod q$ with probability $1 - 1/q$ even under the condition that $x_i = x_s \mod p$. Therefore, $\gcd(x_i - x_s, N)$ is a nontrivial factor of N with probability at least $1 - 1/q$. We refer to the fact that $x_i = x_s \mod p$ for some $i < s$ as a *collision*. What can we say about the value of s, i.e. the length of the sequence $x_0, x_1, x_2, \ldots$ till we encounter a collision? It is obvious that we can upper bound this length by $p+1$. But, we can do better - it turns out that on expectation, this length is only $O(\sqrt{p})$.

Since we are only bothered about collisions modulo p, we can assume for the moment that the numbers $x_0, x_1, x_2, \ldots$ are chosen randomly from $\mathbb{Z}_p$.

Lemma 1 (Birthday problem) *Let $x_0, x_1, x_2, \ldots$ be a sequence where x_i is uniformly randomly (and independently) chosen from $\{0, 1, \ldots, p-1\}$, and s be the smallest index such that $x_s = x_i$ for some $i < s$. Then the expected value of s is $O(\sqrt{p})$.*

Proof Clearly, for any $j \geq 1$, $\Pr[s \geq j] = \prod_{i=0}^{j-1}(1 - i/p) \leq \prod_{i=0}^{j-1} e^{-i/p} \leq e^{-(j-1)^2/2p}$. Therefore, expectation of s, $E[s] = \sum_{j=0}^{\infty} \Pr[s \geq j] = 1 + \sum_{j=1}^{\infty} \Pr[s \geq j]$. Hence,

$$E[s] \leq 1 + \sum_{j=1}^{\infty} e^{-(j-1)^2/2p} \leq 2 + \sqrt{2p} \int_0^{\infty} e^{-x^2} dx \leq 2 + \sqrt{2p} \int_0^{\infty} e^{-x} dx = 2 + \sqrt{2p}$$

Hence, $E[s] = O(\sqrt{p})$. (Exercise 1 shows how to get a slightly better bound.) ■

The above discussion immediately gives a randomized integer factoring algorithm: Choose $x_0, x_1, x_2, \ldots$ uniformly at random from $\mathbb{Z}_N$ until we pick a x_s such that $\gcd(x_i - x_s, N)$ yields a proper factor of N. Lemma 1 implies that this algorithm has expected time complexity $\tilde{O}(\sqrt{p})$ bit operations (which is the same as the Pollard-Strassen's algorithm). But, if we store the numbers in the sequence $x_0, x_1, x_2, \ldots$ until a collision happens then the expected space complexity will also be $\tilde{O}(\sqrt{p})$ bits which is not an asymptotic improvement on the Pollard-Strassen's algorithm. Reduction in space usage is achieved using Floyd's cycle detection trick.

1.1 Floyd's cycle detection trick

Suppose, we have a sequence $x_0, x_1, x_2, \ldots$ such that $x_i = f(x_{i-1})$, where f is a function from $\{0, \ldots, p-1\}$ to itself. Let t be the minimum index for which $x_t = x_{t+\ell}$, where ℓ is also minimal (meaning, ℓ is the length of the cycle on top of the 'rho' that is 'sketched out' by the sequence). Given x_0, Floyd's algorithm outputs an index i such that $x_i = x_{2i}$ and $i \leq t + \ell$.

Algorithm 1 Floyd's cycle detection algorithm

```
Input:  A number x_0 ∈ {0,...,p-1}.
Output:  An index i such that x_i = x_{2i}, where x_j = f(x_{j-1}) for all j ≥ 1.
1.  Set i = 1 and y_0 = x_0.
2.  repeat
3.      x_i = f(x_{i-1}) and y_i = f(f(y_{i-1})),
4.      if x_i = y_i, output i and stop.
5.      else set i ← i + 1.
```

Lemma 2 *The above algorithm outputs an $i \leq t + \ell$, where t and ℓ are as defined above.*

Proof It is clear that at all stages of the algorithm, $y_i = x_{2i}$. Now let's see what happens for $i = t + \ell - (t \mod \ell)$. Since, $i \geq t$ and $\ell | (2i - i) = i$, it must be the case that $x_i = x_{2i}$. Therefore, the output of the algorithm is less than or equal to $t + \ell$. ■

In other words, Floyd's algorithm detects a cycle in at most $t + \ell$ iterations by storing only *two* values of the sequence in every iteration.

In the randomized factoring algorithm discussed earlier, the function f returns a value uniformly at random from $\{0, \ldots, p-1\}$ irrespective of the function argument. In Pollard's rho algorithm, f is chosen to be the 'magic' function $f(x) = (x^2 + 1) \mod N$, where $x \in \mathbb{Z}_N$. Apparently, this choice of f behaves like a 'random' function for all practical purposes. As to why it behaves like a random function no one really knows - indeed, a rigorous analysis of Pollard's rho algorithm is an open problem.

Algorithm 2 Pollard's rho method for integer factoring

```
1.  Choose x_0 uniformly at random from {0,...,N-1}.  Set i = 1 and y_0 = x_0.
2.  repeat
3.      x_i = (x_{i-1}^2 + 1) mod N and y_i = (y_{i-1}^2 + 1)^2 + 1 mod N,
4.      if m = gcd(x_i - y_i, N) is nontrivial, output m and stop.
5.      else set i ← i + 1.
```

Theorem 3 *Under the assumption that the sequence $\{x_i | x_i = x_{i-1}^2 + 1 \mod p\}_{i \geq 1}$ behaves like a random sequence, for a uniformly random value of $x_0 \in \{0, \ldots, p-1\}$, Pollard's rho method outputs the smallest prime factor of N, namely $p = s(N)$, in expected $\tilde{O}(s(N)^{1/2})$ bit operations and using only $\tilde{O}(1)$ bits of space.*

The proof of the theorem follows easily from the above discussion. We leave the details as an exercise.

2 Introduction to Smooth numbers

Smooth numbers are integers with *small* prime factors. So, by definition, they are easy to factor. Study on the density of these numbers dates back to the work of Dickman [Dic30]. It is easy to see that most integers

have at least one small prime factor. For example, every second integer has 2 as its factor, every third has 3 as factor and so on. After we divide an integer by its small factors we are left with another integer with large prime factors. Factoring an integer with large prime factors is potentially a difficult task. Quite intriguingly, many such hard factoring instances can be *reduced* to factoring few smooth numbers. We will see the details of such reductions, in subsequent lectures, in Dixon's *random squares* algorithm and the Quadratic Sieve algorithm.

Smooth numbers find important applications in other factoring algorithms, like the Number Field Sieve [LJMP90] and Lenstra's elliptic curve method [Jr.87]. As another application, we will discuss the *index-calculus method* for computing discrete logarithms in a later lecture. Smooth numbers also play a crucial role in proving many analytic number theoretic results. To cite an example, the result on infinitude of Carmichael numbers [AGP94] depends on the density of *smooth primes*, that is prime p such that $p-1$ is smooth.

We now state the result by Dickman on the density of smooth numbers. In fact, this particular estimate is due to Canfield, Erdös and Pomerance [CEP83]. An integer is called *y-smooth* if all its prime factors are less than y. Let $\psi(x,y)$ be the total number of y-smooth integers between 1 and x.

Lemma 4 *If $u = O(\frac{\log x}{\log\log x})$ then $\frac{\psi(x,x^{\frac{1}{u}})}{x} = u^{-u(1+o(1))}$, where $o(1) \to 0$ as $u \to \infty$.*

We would use this estimate in the Quadratic Sieve algorithm. But before we move on to the applications, let us get a feel of how to find $\psi(x,y)$. Let $y^u = x$ and $p_1, \ldots, p_k$ be the primes less than y. Then, by the Prime Number Theorem, $k \approx \frac{y}{\ln y}$. Since, $p_k^u \leq x$, all numbers of the form $p_1^{e_1} \ldots p_k^{e_k}$, with $\sum_{i=1}^{k} e_i \leq u$, are y-smooth and less than x. Hence, $\psi(x,y) \geq \binom{k+u}{u} \geq (\frac{k}{u})^u \approx \frac{x}{(u \ln y)^u}$ (taking u to be an integer). Therefore, $\frac{\psi(x,x^{\frac{1}{u}})}{x} \geq (u \ln y)^{-u}$. The purpose of presenting this coarse estimate is not merely to show a similarity with the expression in Lemma 4. A sophisticated version of this argument would be used in Dixon's algorithm.

For further details on the density of smooth numbers, the reader may refer to the article by Andrew Granville [Gra08] in [BS08]. An exposition to smooth numbers and their various applications can also be found in the article by Carl Pomerance [Pom94].

2.1 Smooth numbers in integer factoring

The two factoring algorithms, we are going to discuss in this course, are based on one simple idea. Let N be the given odd integer that we want to factor and α, β be two other integers less than N such that $\alpha^2 = \beta^2 \bmod N$. If it happens that neither $\alpha = \beta \bmod N$ nor $\alpha = -\beta \bmod N$ then $\gcd(\alpha+\beta, N)$ (as well as, $\gcd(\alpha-\beta, N)$) yields a nontrivial factor of N. This basic idea is the cornerstone of many modern day factoring algorithms and was first introduced as a general scheme by Kraitchik [Kra26, Kra29]. To make the scheme work, we are faced with two immediate questions.

- How to find a square modulo N?
- How to find two 'distinct' roots of the square modulo N?

By 'distinct' we mean the condition, $\alpha \neq \pm\beta \bmod N$. The first question is rather easy to answer. A random integer is a square modulo N with high probability if N has few prime factors. Besides, one can always take an integer and simply square it modulo N. However, it is the second question that demands more attention. Much of the effort in both Dixon's algorithm and the Quadratic Sieve is centered around efficiently finding 'distinct' roots of a square and it is here that smooth numbers enter the scene. The idea of involving smooth numbers is based on an earlier work by Morrison and Brillhart [MB75].

Exercises:

1. Show that $\int_0^\infty e^{-x^2} dx = \sqrt{\pi}/2$. This gives a slightly better bound on $E[s]$ in Lemma 1.

References

[AGP94] W. R. Alford, A. Granville, and C. Pomerance. There are Infinitely Many Carmichael Numbers. *Annals of Mathematics*, 139:703–722, 1994.

[BS08] Joseph P. Buhler and Peter Stevenhagen, editors. *Algorithmic Number Theory: Lattices, Number Fields, Curves and Cryptography*, volume 44 of *Mathematical Sciences Research Institute Publications*. Cambridge University Press, 2008.

[CEP83] Earl Rodney Canfield, Paul Erdös, and Carl Pomerance. On a problem of Oppenheim concerning "factorisatio numerorum". *Journal of Number Theory*, 17:1–28, 1983.

[Dic30] Karl Dickman. On the frequency of numbers containing prime factors of a certain relative magnitude. *Arkiv för Mathematik, Fysik*, 22:1–14, 1930.

[Gra08] Andrew Granville. Smooth numbers: computational number theory and beyond. *Algorithmic Number Theory: Lattices, Number Fields, Curves and Cryptography, MSRI Publications*, 44:267–323, 2008.

[Jr.87] Hendrik W. Lenstra Jr. Factoring integers with elliptic curves. *Annals of Mathematics*, 126:649–673, 1987.

[Kra26] M. Kraitchik. Théorie des Nombres, Tome II. 1926.

[Kra29] M. Kraitchik. Recherches sur la Théorie des Nombres, Tome II. 1929.

[LJMP90] Arjen K. Lenstra, Hendrik W. Lenstra Jr., Mark S. Manasse, and John M. Pollard. The Number Field Sieve. In *STOC*, pages 564–572, 1990.

[MB75] M. Morrison and J. Brillhart. A method of factoring and the factorization of F_7. *Mathematics of Computation*, 29:183–205, 1975.

[Pom94] Carl Pomerance. The Role of Smooth Numbers in Number Theoretic Algorithms. In *Proceedings of the International Congress of Mathematicians*, 1994.