Dixon's Factoring Algorithm

Basic (Important) Idea (Not Just For Dixon's Algorithm)

- **Suppose**

$$n = p_1^{e_1} p_2^{e_2} \cdots p_r^{e_r}$$

with p_j "odd" distinct primes and $e_j \in \mathbb{Z}^+$.

- **Then $x^2 \equiv 1 \pmod{p_j^{e_j}}$ has two solutions which implies $x^2 \equiv 1 \pmod{n}$ has 2^r solutions.**

- **If x and y are random and $x^2 \equiv y^2 \pmod{n}$, then with probability $(2^r - 2)/2^r$ we can factor n (nontrivially) by considering $\gcd(x + y, n)$.**

Dixon's Factoring Algorithm

1. Randomly choose a number $a > \sqrt{n}$ and compute $s(a) = a^2 \mod n$.

2. A bound $B = B(n)$ is chosen (specified momentarily). Determine if $s(a)$ has a prime factor $> B$. We choose a new a if it does. Otherwise, we obtain a complete factorization of $s(a)$.

3. Let $p_1, \ldots, p_t$ denote the primes $\leq B$. We continue steps (1) and (2) until we obtain $t+1$ different a's, say $a_1, \ldots, a_{t+1}$.

4. From the above, we have the factorizations

$$s(a_i) = p_1^{e(i,1)} p_2^{e(i,2)} \cdots p_t^{e(i,t)} \quad \text{for } i \in \{1, 2, \ldots, t+1\}.$$

Dixon's Factoring Algorithm

4. From the above, we have the factorizations

$$s(a_i) = p_1^{e(i,1)} p_2^{e(i,2)} \cdots p_t^{e(i,t)} \quad \text{for } i \in \{1, 2, \ldots, t+1\}.$$

For $i \in \{1, 2, \ldots, t+1\}$, compute the vectors

$$\vec{v}_i = \langle e(i,1), e(i,2), \ldots, e(i,t) \rangle \mod 2.$$

These vectors are linearly dependent modulo 2. Use Gaussian elimination (or something better) to find a non-empty set $S \subseteq \{1, 2, \ldots, t+1\}$ such that $\sum_{i \in S} \vec{v}_i \equiv \vec{0} \pmod 2$. Calculate $x \in [0, n-1] \cap \mathbb{Z}$ (in an obvious way) satisfying

$$\prod_{i \in S} s(a_i) \equiv x^2 \pmod n.$$

4. From the above, we have the factorizations

$$s(a_i) = p_1^{e(i,1)} p_2^{e(i,2)} \cdots p_t^{e(i,t)} \quad \text{for } i \in \{1, 2, \ldots, t+1\}.$$

For $i \in \{1, 2, \ldots, t+1\}$, compute the vectors

$$\vec{v}_i = \langle e(i,1), e(i,2), \ldots, e(i,t) \rangle \mod 2.$$

These vectors are linearly dependent modulo 2. Use Gaussian elimination (or something better) to find a non-empty set $S \subseteq \{1, 2, \ldots, t+1\}$ such that $\sum_{i \in S} \vec{v}_i \equiv \vec{0} \pmod 2$. Calculate $x \in [0, n-1] \cap \mathbb{Z}$ (in an obvious way) satisfying

$$\prod_{i \in S} s(a_i) \equiv x^2 \pmod n.$$

5. Calculate $y = \prod_{i \in S} a_i \mod n$. Then $x^2 \equiv y^2 \pmod n$. Compute $\gcd(x + y, n)$. Hopefully, a nontrivial factorization of n results.

Small Example: $n = 1189$ **and** $B = 11$.

Homework: (due October 26 by class time)
page 14, problem (1) about (1) on page 12
page 16 on Dixon's Factoring Algorithm
New Problem below (not in Notes)

New Problem.

(a) Calculate accurate to 4 decimal places the value of

$$\lim_{x \to \infty} \frac{|\{n \le x : \forall \text{ primes } p \text{ dividing } n, \text{ we have } p \le x^{1/3}\}|}{x}.$$

(b) Calculate accurate to 4 decimal places the value $a \in (0, 1)$ such that

$$\lim_{x \to \infty} \frac{|\{n \le x : \forall \text{ primes } p \text{ dividing } n, \text{ we have } p \le x^{a}\}|}{x} = \frac{1}{2}.$$

Small Example: $n = 1189$ **and** $B = 11$.

Homework: (due October 26 by class time)
page 14, problem (1) about (1) on page 12
page 16 on Dixon's Factoring Algorithm
New Problem below (not in Notes)

Use Dixon's Algorithm to factor $n = 80099$. **Suppose** $B = 15$ **and the** a_j**'s from the first three steps are the numbers 1392, 58360, 27258, 39429, 12556, 42032, and 1234. (Each of these squared mod** n **should have all of its prime factors** $\leq B$.)

MAPLE EXAMPLE

What bound B on the primes is optimal (or at least good)?

What is the running time for Dixon's algorithm?

$$\psi(x, y) = |\{n \le x : p|n \implies p \le y\}|$$
$$\psi\left(x, \sqrt{x}\,\right) \sim (1 - \log 2)x$$

Theorem (Dickman). *For u fixed, $\psi(x, x^{1/u}) \sim \rho(u)x$ where $\rho(u)$ satisfies:*

(i) $\rho(u)$ is continuous for $u > 0$

(ii) $\rho(u) \to 0$ as $u \to \infty$

(iii) $\rho(u) = 1$ for $0 < u \le 1$

(iv) for $u > 1$, $\rho(u)$ satisfies the differential delay equation $u\rho'(u) = -\rho(u - 1)$.

What bound B on the primes is optimal (or at least good)?

What is the running time for Dixon's algorithm?

$$\psi(x, y) = |\{n \le x : p|n \implies p \le y\}|$$

$$\psi\left(x, \sqrt{x}\,\right) \sim (1 - \log 2)x$$

Theorem (Dickman). ***For u fixed, $\psi(x, x^{1/u}) \sim \rho(u)x$ where $\rho(u)$ satisfies:***

(i) $\rho(u)$ is continuous for $u > 0$

(ii) $\rho(u) \to 0$ as $u \to \infty$

(iii) $\rho(u) = 1$ for $0 < u \le 1$

(iv) for $u > 1$, $\rho(u)$ satisfies the differential delay equation $u\rho'(u) = -\rho(u - 1)$.

$$\psi(x,y) = |\{n \le x : p|n \implies p \le y\}|$$

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

Comment: The following estimate was obtained by deBruijn:

$$\rho(u) = \exp\left(-(1+o(1))u\log u\right) \approx \frac{1}{u^u}.$$

Maier showed that u does not need to be fixed in any of the above and instead one can take

$$u < (\log x)^{1-\varepsilon} \quad \text{for any fixed } \varepsilon > 0.$$

Note that $x^{1/\log x} = e$.

$$\psi(x,y) = |\{n \le x : p|n \implies p \le y\}|$$

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

Comment: The following estimate was obtained by deBruijn:

$$\rho(u) = \exp\left(-(1+o(1))u\log u\right) \approx \frac{1}{u^u}.$$

Maier showed that u does not need to be fixed in any of the above and instead one can take

$$u < (\log x)^{1-\varepsilon} \quad \text{for any fixed } \varepsilon > 0.$$

Take $u = \log n / \log B$ so that (if $u < (\log n)^{1-\varepsilon}$)

$$\psi(n,B) = \psi(n, n^{1/u}) = n\exp\left(-(1+o(1))\log n \log u / \log B\right).$$

The number of different a's we expect to consider before we get enough good $s(a)$'s in the algorithm is

$$(\pi(B)+1)\exp\left((1+o(1))\log n \log u / \log B\right).$$

$$\psi(x,y) = |\{n \le x : p|n \implies p \le y\}|$$

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

Take $u = \log n / \log B$ so that (if $u < (\log n)^{1-\varepsilon}$)

$$\psi(n,B) = \psi(n, n^{1/u}) = n \exp\left(-(1+o(1)) \log n \log u / \log B\right).$$

The number of different a's we expect to consider before we get enough good $s(a)$'s in the algorithm is

$$(\pi(B)+1) \exp\left((1+o(1)) \log n \log u / \log B\right).$$

We also expect $\le B$ steps to factor each value of $s(a)$.

$$\psi(x,y) = |\{n \le x : p|n \implies p \le y\}|$$

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

Take $u = \log n / \log B$ so that (if $u < (\log n)^{1-\varepsilon}$)

$$\psi(n,B) = \psi(n, n^{1/u}) = n \exp\left(-(1+o(1)) \log n \log u / \log B\right).$$

The number of different a's we expect to consider before we get enough good $s(a)$'s in the algorithm is

$$(\pi(B)+1)\exp\left((1+o(1))\log n \log u / \log B\right).$$

We also expect $\le B$ steps to factor each value of $s(a)$. This means we should take

$$B = \exp\left(\sqrt{\log n}\sqrt{\log u}/\sqrt{2}\right) = \exp\left(\sqrt{\log n}\sqrt{\log\log n}/2\right),$$

$$\psi(x,y) = |\{n \le x : p|n \implies p \le y\}|$$

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

Take $u = \log n/\log B$ so that (if $u < (\log n)^{1-\varepsilon}$)

$$\psi(n,B) = \psi(n,n^{1/u}) = n\exp\left(-(1+o(1))\log n \log u/\log B\right).$$

The number of different a's we expect to consider before we get enough good $s(a)$'s in the algorithm is

$$(\pi(B)+1)\exp\left((1+o(1))\log n \log u/\log B\right).$$

We also expect $\le B$ steps to factor each value of $s(a)$. This means we should take

$$B = \exp\left(\sqrt{\log n}\sqrt{\log u}/\sqrt{2}\right) = \exp\left(\sqrt{\log n}\sqrt{\log\log n}/2\right),$$

$$\psi(x,y) = |\{n \le x : p|n \implies p \le y\}|$$

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

Take $u = \log n / \log B$ so that (if $u < (\log n)^{1-\varepsilon}$)

$$\psi(n,B) = \psi(n, n^{1/u}) = n \exp\left(-(1+o(1))\log n \log u / \log B\right).$$

The number of different a's we expect to consider before we get enough good $s(a)$'s in the algorithm is

$$(\pi(B)+1)\exp\left((1+o(1))\log n \log u / \log B\right).$$

We also expect $\le B$ steps to factor each value of $s(a)$. This means we should take

$$B = \exp\left(\sqrt{\log n}\sqrt{\log u}/\sqrt{2}\right) = \exp\left(\sqrt{\log n}\sqrt{\log\log n}/2\right),$$

and the expected running time for Dixon's Algorithm is about

$$\exp\left(2\sqrt{\log n}\sqrt{\log\log n}\right),$$

including Gaussian elimination.

The number of different a's we expect to consider before we get enough good $s(a)$'s in the algorithm is

$$(\pi(B)+1)\exp\left((1+o(1))\log n\log u/\log B\right).$$

We also expect $\leq B$ steps to factor each value of $s(a)$. This means we should take

$$B=\exp\left(\sqrt{\log n}\sqrt{\log u}/\sqrt{2}\right)=\exp\left(\sqrt{\log n}\sqrt{\log\log n}/2\right),$$

and the expected running time for Dixon's Algorithm is about

$$\exp\left(2\sqrt{\log n}\sqrt{\log\log n}\right),$$

including Gaussian elimination.

Comment: This is a rough estimate. A closer analysis would give a running time of

$$\exp\left((2\sqrt{2}+o(1))\sqrt{\log n}\sqrt{\log\log n}\,\right).$$

With some more work, Pomerance and later Vallée reduced the constant $2\sqrt{2}$ so that now we know it can be replaced to $\sqrt{4/3}$.

The CFRAC Algorithm

Every real number α can be written uniquely as a *simple continued fraction*

$$\alpha = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \ddots}}}$$

where $a_0 \in \mathbb{Z}$ and $a_j \in \mathbb{Z}^+$ for $j \geq 1$. The *convergents* obtained by truncating the above give approximations a/b to α satisfying

$$\left|\alpha - \frac{a}{b}\right| < \frac{1}{b^2}.$$

The CFRAC Algorithm

$$\left|\alpha - \frac{a}{b}\right| < \frac{1}{b^2}$$

$$\left|\alpha^2 - \frac{a^2}{b^2}\right| \ll \frac{\alpha}{b^2}$$

$$\left|b^2\alpha^2 - a^2\right| \ll \alpha$$

Comment: Every convergent a/b of $\sqrt{n}$ satisfies

$$\left|b^2 n - a^2\right| < 2\sqrt{n}.$$

The CFRAC Algorithm

Compute the numerators a_j of the convergents of $\sqrt{n}$. If the corresponding denominators are b_j, then $|a_j^2 - nb_j^2| < 2\sqrt{n}$. Recall $s(a) = a^2 \mod n$. Repeat Dixon's algorithm but now

- Define $s(a)$ to be in $(-n/2, n/2]$ with $s(a) \equiv a^2 \pmod{n}$. Then $|s(a_j)| < 2\sqrt{n}$.
- Treat -1 (the possible negative sign in $s(a)$) as a prime.

How is the running time of the algorithm affected?

The chance that a_j has the property that all its prime divisors are $\leq B$ is $\psi(2\sqrt{n}, B)$ instead of $\psi(n, B)$. The expected running time is

$$O\left(\exp(\sqrt{2}\sqrt{\log n}\sqrt{\log\log n})\right).$$

$$\psi(x,y) = |\{n \le x : p|n \implies p \le y\}|$$

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

Take $u = \log n / \log B$ so that (if $u < (\log n)^{1-\varepsilon}$)

$$\psi(n,B) = \psi(n, n^{1/u}) = n \exp\left(-(1+o(1)) \log n \log u / \log B\right).$$

The number of different a's we expect to consider before we get enough good $s(a)$'s in the algorithm is

$$(\pi(B)+1) \exp\left((1+o(1)) \log n \log u / \log B\right).$$

We also expect $\le B$ steps to factor each value of $s(a)$. This means we should take

$$B = \exp\left(\sqrt{\log n}\sqrt{\log u}/\sqrt{2}\right) = \exp\left(\sqrt{\log n}\sqrt{\log\log n}/2\right),$$

and the expected running time for Dixon's Algorithm is about

$$\exp\left(2\sqrt{\log n}\sqrt{\log\log n}\right),$$

including Gaussian elimination.

The CFRAC Algorithm

Compute the numerators a_j of the convergents of $\sqrt{n}$. If the corresponding denominators are b_j, then $|a_j^2 - nb_j^2| < 2\sqrt{n}$. Recall $s(a) = a^2 \mod n$. Repeat Dixon's algorithm but now

- Define $s(a)$ to be in $(-n/2, n/2]$ with $s(a) \equiv a^2 \pmod{n}$. Then $|s(a_j)| < 2\sqrt{n}$.
- Treat -1 (the possible negative sign in $s(a)$) as a prime.

How is the running time of the algorithm affected?

The chance that a_j has the property that all its prime divisors are $\leq B$ is $\psi(2\sqrt{n}, B)$ instead of $\psi(n, B)$. The expected running time is

$$O\left(\exp(\sqrt{2}\sqrt{\log n}\sqrt{\log\log n})\right).$$

The CFRAC Algorithm

Compute the numerators a_j of the convergents of $\sqrt{n}$. If the corresponding denominators are b_j, then $|a_j^2 - nb_j^2| < 2\sqrt{n}$. Recall $s(a) = a^2 \mod n$. Repeat Dixon's algorithm but now

- Define $s(a)$ to be in $(-n/2, n/2]$ with $s(a) \equiv a^2 \pmod{n}$. Then $|s(a_j)| < 2\sqrt{n}$.

- Treat -1 (the possible negative sign in $s(a)$) as a prime.

The chance that a_j has the property that all its prime divisors are $\leq B$ is $\psi(2\sqrt{n}, B)$ instead of $\psi(n, B)$. The expected running time is

$$O\left(\exp(\sqrt{2}\sqrt{\log n}\sqrt{\log\log n})\right).$$

Comment: Brillhart and Morrison (1970) used the CFRAC algorithm to factor $F_7 = 2^{2^7} + 1$ (having 39 digits).

A Further Idea

An "early abort" strategy can be combined with the above ideas to reduce the running time of the algorithms.

A Further Idea

An "early abort" strategy can be combined with the above ideas to reduce the running time of the algorithms. Given a, one stops trying to factor $s(a)$ if it has no "small" prime factors.

A Further Idea

An "early abort" strategy can be combined with the above ideas to reduce the running time of the algorithms. Given a, one stops trying to factor $s(a)$ if it has no "small" prime factors. This leads to a running time of the form

$$O\left(\exp\left(\sqrt{3/2}\sqrt{\log n}\sqrt{\log\log n}\,\right)\right).$$