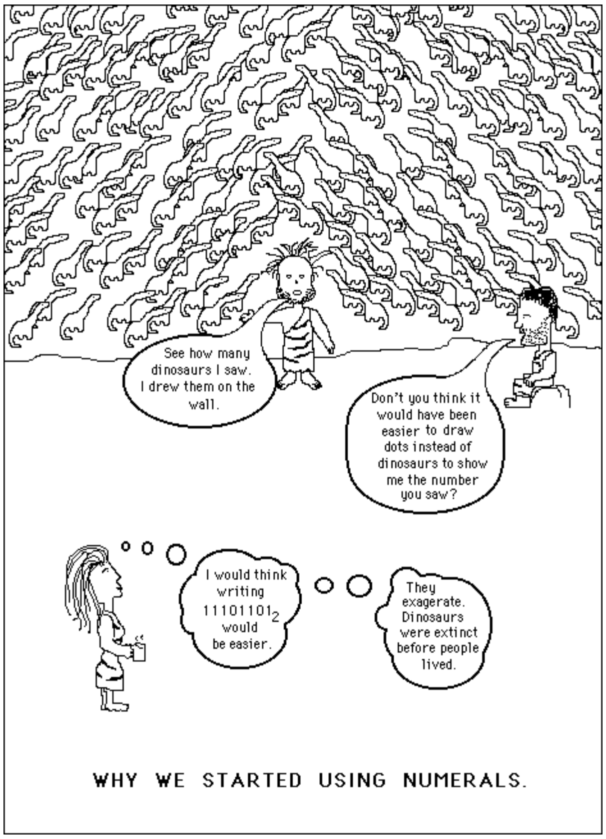
Math 788M: Computational Number Theory
(Instructor's Notes)*

The Very Beginning:

- A positive integer n can be written in n steps.
- The role of numerals (now $O(\log n)$ steps)
- Can we do better? (Example: The largest known prime contains 258716 digits and doesn't take long to write down. It's $2^{859433} - 1$.)



WHY WE STARTED USING NUMERALS.

Running Time of Algorithms:

- A positive integer n in base b contains $[\log_b n] + 1$ digits.
- Big-Oh & Little-Oh Notation (as well as $\ll$, $\gg$, $\sim$, $\asymp$)

Examples 1: $\log(1 + (1/n)) = O(1/n)$

Examples 2: $[\log_b n] + 1 \asymp \log n$

Examples 3: $1 + 2 + \cdots + n \ll n^2$

Examples 4: f a polynomial of degree $k \implies f(n) = O(n^k)$

---

*These notes are from a course taught by Michael Filaseta in the Spring of 1996.

Examples 5: $(r+1)^\pi \sim r^\pi$

• We will want algorithms to run quickly (in a small number of steps) in comparison to the length of the input. For example, we may ask, "How quickly can we factor a positive integer n?" One considers the length of the input n to be of order $\log n$ (corresponding to the number of binary digits n has). An algorithm runs in polynomial time if the number of steps it takes is bounded above by a polynomial in the length of the input. An algorithm to factor n in polynomial time would require that it take $O\left((\log n)^k\right)$ steps (and that it factor n).

Addition and Subtraction (of n and m):

- We are taught how to do binary addition and subtraction in $O(\log n + \log m)$ steps.
- We aren't going to do better than this.
- Converting to base 10 (or any other base) is another story.

Multiplication (of n and m):

- We are taught how to do multiplication in $O\left((\log n)(\log m)\right)$ steps.
- Better is Possible

Example: Let $M(d)$ denote an upper bound on the number of steps required to multiply two numbers with $\le d$ binary digits. For simplicity, we suppose n and m both have $2r$ digits. We show that nm can be calculated in $\ll r^{\log 3/\log 2} \ll r^{1.585}$ steps. Write $n = a \times 2^r + b$ and $m = c \times 2^r + d$. Then $nm = x \times 2^{2r} + y + z \times 2^r$ where $x = ac$, $y = bd$, and $z = (a+b)(c+d) - x - y$. We deduce $M(2r) \le 3M(r+2) + kr$ for some constant k (where we have allowed for the possibility that r is not an integer). Etc.

- Even Better is Possible

Theorem. $M(d) \ll d(\log d)\log\log d$.

- Note that multiplying a d digit number by 19 takes $O(d)$ steps.

Sketch of proof that $M(d) \ll d^{1+\varepsilon}$.

• **Theorem.** *Given distinct numbers $x_0, x_1, \dots, x_k$ and numbers $y_0, y_1, \dots, y_k$, there is a unique polynomial f of degree $\le k$ such that $f(x_j) = y_j$ for all j.*

• Lagrange Interpolation:

$$f(x) = \sum_{i=0}^{k} \left(\prod_{\substack{0 \le j \le k \\ j \ne i}} \frac{x - x_j}{x_i - x_j} \right) y_i$$

• Mimic the Example above. Suppose n and m have $\le kr$ digits. Write

$$n = \sum_{j=0}^{k-1} a_j 2^{jr} \quad \text{and} \quad m = \sum_{j=0}^{k-1} b_j 2^{jr}.$$

Consider

$$f(x) = \left(\sum_{j=0}^{k-1} a_j x^j\right)\left(\sum_{j=0}^{k-1} b_j x^j\right),$$

and note $nm = f(2^r)$. The coefficients of $f(x)$ can be determined by using $x_j = j$ in the Lagrange interpolation formula, taking $2k-1$ multiplications of $\le r + c_k$ (for some constant c_k) digit numbers (to obtain the y_j). This leads to

$$M(kr) \le (2k-1)M(r+c_k) + c_k' r \quad \Longrightarrow \quad M(d) \ll (2k-1)^{\log_k d} \ll d^{\log(2k-1)/\log k},$$

for some constant c_k', which implies what we want.

- What about the required division?

Homework:

(1) Let

$$f_1(n) = \log n, \quad f_2(n) = \log\log n, \quad f_3(n) = \log(3n+5),$$

$$f_4(n) = 1, \quad \text{and} \quad f_5(n) = n.$$

Determine the largest subset S of $\{1, 2, \dots, 5\}$ for which each of (a), (b), and (c) holds (each letter below is a separate problem requiring a possibly different set S). You do not need to justify your answers.

(a) $f_1(n) = O(f_j(n))$ for $j \in S$.
(b) $f_1(n) = o(f_j(n))$ for $j \in S$.
(c) $f_1(n) \sim f_j(n)$ for $j \in S$.

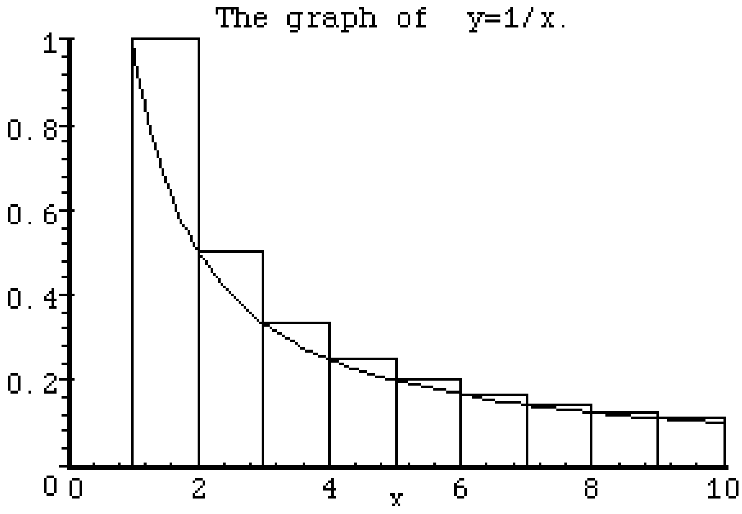
(2) Find an explicit function $f(x)$ involving logarithms, powers, and/or exponentials which has both of the properties (at the same time):

(i) $f(n) \ll n^{\varepsilon}$ for every $\varepsilon > 0$.
(ii) $f(n) \gg (\log n)^k$ for every $k > 0$.

(Hint: I did put exponentials in the list above for a reason.) Justify your answer.

(3) The value of $f(n) = \sum_{k=1}^{n} \frac{1}{k}$ can be estimated by comparing it's value to an integral. For example, by comparing the sum of the areas of the rectangles indicated in the graph on the top of the next page with the area under the graph of $y = 1/x$, one obtains

$$f(9) \ge \int_1^{10} \frac{1}{x}\, dx = \log 10.$$



(a) Prove that $f(n) \gg \log n$.

(b) Prove that $f(n) \ll \log n$. (Hint: Try another picture as above with the rectangles completely under the curve.)

(c) Prove that $f(n) \sim \log n$.

Division:

• Problem: Given two positive integers n and m, determine the quotient q and the remainder r when n is divided by m. These should be integers satisfying

$$n = mq + r \quad \text{and} \quad 0 \leq r < m.$$

• Let $D(d)$ denote an upper bound on the number of steps required to obtain q and r given n and m each have $\leq d$ binary digits.

• **Theorem.** *Suppose $M(d)$ has the form $df(d)$ where $f(d)$ is an increasing function of d. Then $D(d) \ll M(d)$.*

• We need only compute $1/m$ to sufficient accuracy. Suppose n and m have $\leq r$ digits. If $1/m = 0.d_1 d_2 ...$ (in binary) with $d_1, \dots, d_r$ known, then $n/m = (1/2^r)(n \times d_1 d_2 \dots d_r) + \theta$ where $0 \leq \theta \leq 1$. If q' represents the number formed from all but the last r digits of $n \times d_1 d_2 \dots d_r$, then $n = mq' + \theta'$ where $0 \leq \theta' < 2m$. Try $q = q'$ and $q = q' + 1$.

• Preliminaries to the Proof – Newton's Method

Example: Discuss computing $1/m$. Consider $f(x) = mx - 1$ and $f(x) = m - 1/x$. Develop the idea of beginning with a good approximation x_0 to $1/m$ and obtaining successively better ones by using the recursion $x_{n+1} = 2x_n - mx_n^2$. Note that if $x_n = (1 - \varepsilon)/m$, then $x_{n+1} = (1 - \varepsilon^2)/m$. The role of multiplication already has replaced the role of division.

• Algorithm from Knuth, Vol. 2, pp. 295-6:

Algorithm R *(High-precision reciprocal)***.** Let v have the binary representation $v = (0.v_1 v_2 v_3 \dots)_2$, where $v_1 = 1$. This algorithm computes an approximation z to $1/v$, such that

$$|z - 1/v| \leq 2^{-n}.$$

R1. [Initial Approximation.] Set $z \leftarrow \frac{1}{4}\lfloor 32/(4v_1 + 2v_2 + v_3)\rfloor$ and $k \leftarrow 0$.

R2. [Newtonian iteration.] (At this point we have a number z of the binary form $(xx.xx\ldots x)_2$ with $2^k + 1$ places after the radix point, and $z \le 2$.) Calculate $z^2 = (xxx.xx\ldots x)_2$ exactly, using a high-speed multiplication routine. Then calculate $V_k z^2$ exactly, where $V_k = (0.v_1v_2\ldots v_{2^{k+1}+3})_2$. Then set $z \leftarrow 2z - V_k z^2 + r$, where $0 \le r < 2^{-2^{k+1}-1}$ is added if necessary to "round up" z so that it is a multiple of $2^{-2^{k+1}-1}$. Finally, set $k \leftarrow k+1$.

R3. [Test for end.] If $2^k < n$, go back to step R2; otherwise the algorithm terminates.

- Relate algorthm to our problem (note notational differences).
- Let z_k be the value of z after k iterations of step R2. Show by induction that

$$(*) \qquad z_k \le 2 \quad \text{and} \quad |z_k - 1/v| \le 2^{-2^k}.$$

For $k = 0$, try each possibility for v_1, v_2, and v_3. For the RHS of $(*)$, use

$$\frac{1}{v} - z_{k+1} = v\left(\frac{1}{v} - z_k\right)^2 - z_k^2(v - V_k) - r.$$

For the LHS of $(*)$, consider each of the cases $V_k = 1/2$, $V_{k-1} = 1/2 \ne V_k$, and otherwise.

- Deduce that the number of steps is

$$2M(4n) + 2M(2n) + 2M(n) + 2M(n/2) + \cdots + O(n) \ll M(n).$$

Homework:

We have sketched the argument that for every $\varepsilon > 0$, $M(d) \ll d^{1+\varepsilon}$. Most of the details given dealt with showing $M(kr) \le (2k-1)M(r + c_k) + c'_k r$. For $k = 2$, we gave a more detailed argument. Do the same (give details) for $k = 3$ to prove that $M(d) \ll d^{\log 5/\log 3}$. Your account should explain the connection between the estimates $M(3r) \le 5M(r+c) + c'r$ and $M(d) \ll d^{\log 5/\log 3}$. Also, clarify what the 5 multiplications are and determine a specific value for c (not c').

Elementary Number Theory:

- Modulo Arithmetic (definition, properties, & different notation)
- Computing $a^m \pmod{n}$
- Euler's Phi Function (definition, formula)
- Euler's Theorem, Fermat's Little Theorem, and Existence of Inverses
- Computing Inverses (later – see two sections from now)
- Chinese Remainder Theorem
- Generators exist modulo 2, 4, p^e, and $2p^e$

Homework:

(1) A very wealthy person buys items priced at \$2,259.29, \$4,855.07, \$9,921.23, \$12,009.91, and \$20,744.39. How many of each item did he purchase if the total purchase without taxes comes to \$749,518.05? (Hint: Factor and use a calculator or computer.)

(2) Show that for every positive integer k, there is an integer n such that each of the numbers $n+1, n+2, \dots, n+k$ is divisible by a square > 1. For example, for $k = 3$, one can take $n = 47$ since 48 is divisible by 4, 49 is divisible by 49, and 50 is divisible by 25.

Greatest Common Divisors:

- Algorithm from Knuth, Vol. 2, p. 320:

Algorithm A *(Modern Euclidean algorithm)***.** Given nonnegative integers u and v, this algorithm finds their greatest common divisor. (*Note:* The greatest common divisor of *arbitrary* integers u and v may be obtained by applying this algorithm to $|u|$ and $|v|$ ....)

A1. [$v = 0$?] If $v = 0$, the algorithm terminates with u as the answer.

A2. [Take $u \mod v$.] Set $r \leftarrow u \mod v$, $u \leftarrow v$, $v \leftarrow r$, and return to A1. (The operations of this step decrease the value of v, but they leave $\gcd(u, v)$ unchanged.)

- Why does the algorithm work?
- How long does it take? Explain the worse case and establish the following:

Theorem (Lamé). *Let $\phi = (1+\sqrt{5})/2$. Let $0 \le u, v < N$ in Algorithm A. Then the number of times step A2 is repeated is $\le \lceil \log_\phi(\sqrt{5}N) \rceil - 2$.*

- Inverses and a method to compute them
- **Theorem.** *Given integers a and b, not both 0, there exist integers u and v such that $au + bv = \gcd(a, b)$.*
- The average number of times step A2 is repeated is $\asymp \log N$.
- The average value of $\gcd(u, v)$ is $\asymp \log N$ but "usually" it's much smaller.
- Justification. For the first part use that

$$\sum_{d|n} \phi(d) = n \quad \text{and} \quad \sum_{d \le N} \frac{\phi(d)}{d^2} \asymp \log N.$$

Omit the details on the second of these asymptotics. For the first, note that for each divisor d of n, the positive integers having greatest common divisor d with n are precisely the $m \le n$ of the form kd where $1 \le k \le n/d$ and $\gcd(k, n/d) = 1$. This implies

$$n = \sum_{d|n} \phi(n/d) = \sum_{d|n} \phi(d).$$

The average value of $\gcd(u, v)$ is $\asymp \log N$ follows by using $\displaystyle\sum_{1 \le n \le N} \sum_{1 \le m \le N} \sum_{d|n,\, d|m} \phi(d)$. For the second part, observe that the number of pairs (n, m) with $\gcd(n, m) > z$ is bounded above by

$$\sum_{d>z} \frac{N^2}{d^2} \le \frac{N^2}{z^2} + N^2 \int_z^\infty \frac{1}{t^2}\, dt \le \frac{2N^2}{z}.$$

Probable Primes:

- The use of Fermat's Little Theorem
- The example $341 = 11 \times 31$ but note that $3^{340} \equiv 56 \pmod{341}$
- The example $561 = 3 \times 11 \times 17$
- Some noteworthy estimates:

$$P_2(x) \le x^{1-\frac{\log\log\log x}{2\log\log x}} \quad \text{and} \quad P_2(x) \ge x^{2/7} \quad \forall x \ge x_0$$

$$\pi(x) \ge \frac{x}{\log x} = x^{1-\frac{\log\log x}{\log x}} \quad \forall x \ge 17$$

$$P_2(2.5 \times 10^{10}) = 21853 \quad \text{and} \quad \pi(2.5 \times 10^{10}) = 1091987405$$

- Teminology: pseudoprime, probable prime, industrial grade prime, absolute pseudoprime, Carmichael number
- The equivalence of two different definitions for absolute pseudoprimes
- There are infinitely many absolute pseudoprimes
- Strong pseudoprimes. Suppose n is an odd composite number and write $n-1 = 2^s m$ where m is an odd integer. Then n is a *strong pseudoprime to the base* b if either (i) $b^m \equiv 1 \pmod{n}$ or (ii) $b^{2^j m} \equiv -1 \pmod{n}$ for some $j \in [0, s-1]$.
- Strong pseudoprimes base b are pseudoprimes base b.
- Primes p satisfy (i) and (ii) for any b relatively prime to p.
- There are no n which are strong pseudoprimes to every base b with $1 \le b \le n$ and $\gcd(b, n) = 1$. To see this, assume otherwise. Note that n must be squarefree. Next, consider a prime divisor q of n, and note $n/q > 1$. Let $c \in [1, q-1]$ be such that c is not a square modulo q. Let b satisfy $b \equiv 1 \pmod{n/q}$ and $b \equiv c \pmod{q}$. Then (i) cannot hold modulo q and (ii) cannot hold modulo n/q.
- $5^{280} \equiv 67 \pmod{561}$
- The number $3215031751 = 151 \times 751 \times 28351$ is simultaneously a strong pseudoprime to each of the bases 2, 3, 5, and 7. It's the only such number $\le 2.5 \times 10^{10}$.

Homework:

(1) Using the Euclidean algorithm, calculate $\gcd(7046867, 1003151)$.

(2) Calculate integers x and y for which $7046867x + 1003151y = \gcd(7046867, 1003151)$.

(3) Prove that if n is an odd pseudoprime, then $2^n - 1$ is a pseudoprime.

(4) Prove that 1729 is an absolute pseudoprime.

The Lucas-Lehmer Primality Test:

- Fix integers P and Q. Let $D = P^2 - 4Q$. Define recursively u_n and v_n by

$$u_0 = 0, \quad u_1 = 1, \quad u_{n+1} = Pu_n - Qu_{n-1} \text{ for } n \geq 1,$$

$$v_0 = 2, \quad v_1 = P, \quad \text{and} \quad v_{n+1} = Pv_n - Qv_{n-1} \text{ for } n \geq 1.$$

If p is an odd prime and $p \nmid PQ$ and $D^{(p-1)/2} \equiv -1 \pmod{p}$, then $p | u_{p+1}$.

- Compute modulo p by using

$$\begin{pmatrix} u_{n+1} & v_{n+1} \\ u_n & v_n \end{pmatrix} = M^n \begin{pmatrix} 1 & P \\ 0 & 2 \end{pmatrix} \quad \text{where} \quad M = \begin{pmatrix} P & -Q \\ 1 & 0 \end{pmatrix}.$$

- The formulas

$$u_n = \frac{\alpha^n - \beta^n}{\alpha - \beta} \quad \text{and} \quad v_n = \alpha^n + \beta^n \quad \text{for } n \geq 0,$$

where $\alpha = (P + \sqrt{D})/2$ and $\beta = (P - \sqrt{D})/2$

- The formula

$$2^{n-1} u_n = \binom{n}{1} P^{n-1} + \binom{n}{3} P^{n-3} D + \binom{n}{5} P^{n-5} D^2 + \cdots$$

- Prove $p | u_{p+1}$ (let $n = p + 1$ above)

Maple's "isprime" Routine (Version 5, Release 3):

- Don't try isprime(1093^2) or isprime(3511^2) in Maple V, Release 3. The algorithm will end up in an infinite loop. This is not a concern in the latest version of Maple.
- What is isprime doing?
- The help output for isprime:

FUNCTION: isprime - primality test

CALLING SEQUENCE:
isprime(n)

PARAMETERS:
n - integer

SYNOPSIS:
- The function isprime is a probabilistic primality testing routine.

- It returns false if n is shown to be composite within within one strong pseudo-primality test and one Lucas test and returns true otherwise. If isprime returns true, n is "very probably" prime - see Knuth "The art of computer programming", Vol 2, 2nd edition, Section 4.5.4, Algorithm P for a reference and H. Reisel, "Prime numbers and computer methods for factorization". No counter example is known and it has been conjectured that such a counter example must be hundreds of digits long.

SEE ALSO: nextprime, prevprime, ithprime

- The Maple program:

```
proc (n)
   local btor, nr, p, r;
   options remember, system,
     `Copyright 1993 by Waterloo Maple Software`;
   if not type(n,integer) then
     if type(n,numeric) then
       ERROR(`argument must be an integer`)
     else
       RETURN('isprime(n)')
     fi
   fi;
   if n < 2 then
     false
   elif has(`isprime/w`,n) then
     true
   elif igcd(2305567963945518424753102147331756070,n) <> 1 then
     false
   elif n < 10201 then
     true
   elif igcd(84969694892334181105323399091873499659260625866489327366
     1154542634220389327076939090906947730950913750978691711866802886149933382
     5097682386722983737962963066757674131126736578936440788157186969893730633
     1130664786204486249492573240226273954373636390387526081667586612559568346
     3069722044751229884822222855006268378634251996022599630131594564447006472
     0696621750477244528915927867113,n) <> 1 then
     false
   elif n < 1018081 then
     true
   else nr := igcd(408410100000,n-1);
     nr := igcd(nr^5,n-1);
     r := iquo(n-1,nr);
     btor := modp(power(2,r),n);
       if `isprime/cyclotest`(n,btor,2,r) = false
         or irem(nr,3) = 0 and `isprime/cyclotest`(n,btor,3,r) = false
         or irem(nr,5) = 0 and `isprime/cyclotest`(n,btor,5,r) = false
         or irem(nr,7) = 0 and `isprime/cyclotest`(n,btor,7,r) = false then
         RETURN(false)
       fi;
     for p from 3 while (numtheory[jacobi])(p^2-4,n) <> -1 do od;
     evalb(`isprime/TraceModQF`(p,n+1,n) = [2, p])
   fi
end
```

Maple's Initial Steps:

- The list 'isprime/w' consists of the primes < 100.
- The number 2305567 ... 6070 is the product of the primes < 100. The next prime is 101 and $101^2 = 10201$.
- The number 8496969 ... 7113 is the product of the primes in the interval $(100, 1000)$. The next prime is 1009 and $1009^2 = 1018081$.
- If n is a prime < 1018081, the initial steps will declare it to be prime. If n is a composite number with a prime factor < 1000, then the initial steps will declare it composite. In particular, all $n < 1018081$ will have been properly dealt with.

Maple's Version of the Strong-Pseudoprime Test (or is it?):

- $408410100000 = 2^5 3^5 5^5 7^5$
- If $n-1 = 2^{e_1}3^{e_2}5^{e_3}7^{e_4}m$ where $\gcd(210, m) = 1$, then $nr = 2^{e_1'}3^{e_2'}5^{e_3'}7^{e_4'}$ where $e_j' = \min\{e_j, 25\}$. Also, $r = (n-1)/nr$ and $btor = 2^r \mod n$.
- What is 'isprime/cyclotest'(n,btor,2,r) doing? It computes $x = 2^y \mod n$ where $y = (n-1)/2^{e_1'}$. If $x \equiv \pm 1 \pmod{n}$, then it tells the algorithm to go on to the next step (involving 'isprime/cyclotest'(n,btor,3,r)). Otherwise, it replaces x with $x^2 \mod n$. If now $x = 1$, then it declares n composite. If $x \equiv -1 \pmod{n}$, then it tells the algorithm to go on to the next step. If $x \not\equiv \pm 1 \pmod{n}$, then it again replaces x with $x^2 \mod n$. It continues like this, declaring n composite if $x = 1$, telling the algorithm to go on if $x \equiv -1 \pmod{n}$, and replacing x with $x^2 \mod n$ otherwise, until e_1' squarings of x have occured. At that point, if the algorithm has not declared n to be a prime or composite number and it has not told the algorithm to go on, then it declares that n is composite. This is equivalent to a strong pseudoprime test to the base 2 if $e_1' = e_1$ (for most numbers this will be the case). Otherwise it is a little weaker.
- The steps involving 'isprime/cyclotest'(n,btor,i,r) for $i \in \{3, 5, 7\}$ are variations on a strong pseudoprime test to the base 2 (yes, I mean base 2). For example, 'isprime/cyclotest'(n,btor,3,r) makes use of the fact that if $x^3 \equiv 1 \pmod{p}$, then either $x \equiv 1 \pmod{p}$ or $x^2 + x + 1 \equiv 0 \pmod{p}$ (this is analogous to using for the strong pseudoprime test that if $x^2 \equiv 1 \pmod{p}$, then either $x \equiv 1 \pmod{p}$ or $x \equiv -1 \pmod{p}$).
- The "help" for isprime is somewhat misleading. The references cited do not suggest (as far as I noticed - OK, I didn't read every word in these references) that one should use e_1' above rather than e_1. Algorithm P of Knuth's book is not used (it involves choosing an integer $b \in (1, n)$ at random and checking if n is a strong pseudoprime base b).

Maple's Version of the Lucas-Lehmer Test:

- Take $Q = 1$. Then

$$v_{2n} = v_n^2 - 2 \quad \text{and} \quad v_{2n+1} = v_{n+1}v_n - P \quad \text{for } n \geq 1.$$

Also, $Du_n = 2v_{n+1} - Pv_n$. If p is a prime, then $v_p \equiv P \pmod{p}$. Isprime checks if (v_{n+1}, v_n) is congruent to $(2, P)$ modulo n.

- Prove that if it is, then $n | u_{n+1}$.

- How is (v_{n+1}, v_n) computed modulo n? Beginning with $\vec{w} = (v_1, v_0)$ and considering the binary digits of n beginning from the left-most digit, $\vec{w} = (v_{m+1}, v_m)$ is replaced by (v_{2m+2}, v_{2m+1}) whenever the digit 1 is encountered and by (v_{2m+1}, v_{2m}) whenever the digit 0 is encountered. In this way, the subscript of the second coordinate of $\vec{w}$ corresponds to the number obtained by considering a left portion of the binary representation of n.

Mersenne Primes:

- Definition.
- It's connection to perfect numbers.
- **The Lucas-Lehmer Test.** *Let p be an odd prime, and define recursively*

$$L_0 = 4 \quad \textit{and} \quad L_{n+1} = L_n^2 - 2 \mod (2^p - 1) \quad \textit{for} \quad n \geq 0.$$

Then $2^p - 1$ is a prime if and only if $L_{p-2} = 0$.

- The Lucas sequence with $Q = 1$ and $P = 4$. Here, $L_n = v_{2^n} \mod (2^p - 1)$. Note that

$$u_n = \frac{(2+\sqrt{3})^n - (2-\sqrt{3})^n}{\sqrt{12}} \quad \text{and} \quad v_n = (2+\sqrt{3})^n + (2-\sqrt{3})^n.$$

- ($\Longrightarrow$) Suppose $N = 2^p - 1$ is a prime. We will use that (from the theory of quadratic reciprocity) $3^{(N-1)/2} \equiv -1 \pmod{N}$ (where here N prime and $N \equiv 7 \pmod{12}$ is important). Also, $2^p \equiv 1 \pmod{N}$ easily implies there is an x such that $x^2 \equiv 2 \pmod{N}$. Hence, $2^{(N-1)/2} \equiv x^{N-1} \equiv 1 \pmod{N}$. We want $v_{(N+1)/4} \equiv 0 \pmod{N}$. From $v_{2n} = v_n^2 - 2$, it follows that need only show $v_{(N+1)/2} \equiv -2 \pmod{N}$. Observe that $2 \pm \sqrt{3} = ((\sqrt{2} \pm \sqrt{6})/2)^2$. Hence,

$$\begin{aligned} v_{(N+1)/2} &= \left(\frac{\sqrt{2}+\sqrt{6}}{2}\right)^{N+1} + \left(\frac{\sqrt{2}-\sqrt{6}}{2}\right)^{N+1} \\ &= 2^{-N} \sum_{j=0}^{(N+1)/2} \binom{N+1}{2j} \sqrt{2}^{N+1-2j} \sqrt{6}^{2j} = 2^{(1-N)/2} \sum_{j=0}^{(N+1)/2} \binom{N+1}{2j} 3^j. \end{aligned}$$

Using

$$\binom{N+1}{2j} = \binom{N}{2j} + \binom{N}{2j-1},$$

we deduce

$$2^{(N-1)/2} v_{(N+1)/2} \equiv 1 + 3^{(N+1)/2} \equiv -2 \pmod{N},$$

and the result follows.

- ($\Longleftarrow$, the more important case) We make use of the following two identities:

$$v_n = u_{n+1} - u_{n-1} \quad \text{and} \quad u_{m+n} = u_m u_{n+1} - u_{m-1} u_n, \tag{1}$$

where the subscripts are all assumed to be non-negative integers. Establish

$$\text{(2)} \qquad \text{if } u_n \equiv 0 \pmod{p^e}, \text{ then } u_{pn} \equiv 0 \pmod{p^{e+1}},$$

where e is assumed to be a positive integer. To obtain (2), use induction to show that if $a = u_{n+1}$, then

$$u_{kn} \equiv k a^{k-1} u_n \pmod{p^{e+1}} \quad \text{and} \quad u_{kn+1} \equiv a^k \pmod{p^{e+1}}$$

and then take $k = p$; for example, observe that for $k = 2$, we have

$$u_{2n} = u_n u_{n+1} - u_{n-1} u_n = u_n u_{n+1} + u_n(u_{n+1} - 4u_n) \equiv 2a u_n \pmod{p^{e+1}}$$

and $u_{2n+1} = u_{n+1}^2 - u_n^2 \equiv a^2 \pmod{p^{e+1}}$.

Next, we use that

$$u_n = \sum_{k=0}^{n} \binom{n}{2k+1} 2^{n-2k-1} 3^k \quad \text{and} \quad v_n = \sum_{k=0}^{n} \binom{n}{2k} 2^{n-2k+1} 3^k$$

to obtain that

$$\text{(3)} \qquad u_p \equiv 3^{(p-1)/2} \pmod{p} \quad \text{and} \quad v_p \equiv 4 \pmod{p}.$$

Fermat's Little Theorem implies for $p > 3$ that $u_p \equiv \pm 1 \pmod{p}$. Using (1), (3), and the definition of the u_n, we obtain that if $u_p \equiv 1 \pmod{p}$, then $u_{p-1} \equiv 4u_p - u_{p+1} \equiv 4u_p - v_p - u_{p-1} \equiv -u_{p-1} \pmod{p}$ implying $u_{p-1} \equiv 0 \pmod{p}$. Similarly, if $u_p \equiv -1 \pmod{p}$, then $u_{p+1} \equiv 4u_p - u_{p-1} \equiv 4u_p + v_p - u_{p+1} \equiv -u_{p+1} \pmod{p}$ implying $u_{p+1} \equiv 0 \pmod{p}$. Thus, for every prime $p > 3$, there is an integer $\epsilon = \epsilon(p) = \pm 1$ such that

$$\text{(4)} \qquad u_{p+\epsilon} \equiv 0 \pmod{p}.$$

Also, $u_p \equiv 0 \pmod{p}$ if $p = 2$ and $p = 3$.

Observe that

$$\text{(5)} \qquad \gcd(u_n, u_{n+1}) = 1 \quad \text{and} \quad \gcd(u_n, v_n) \le 2.$$

The former follows from the recursive definition of u_n. The second follows from the first by first noting $2u_{n+1} = 4u_n + v_n$ (obtained by combining (1) with the recursive relation on u_n).

For any positive integer m, denote by $\alpha = \alpha(m)$ the smallest positive integer for which $u_\alpha \equiv 0 \pmod{m}$ (it is not needed, but such an α always exists). Then

$$\text{(6)} \qquad u_n \equiv 0 \pmod{m} \iff \alpha | n.$$

This follows by considering $u_\alpha, u_{\alpha+1}, \ldots$ modulo m. The necessity also requires using (5).

By assumption, $v_{2^{p-2}} \equiv L_{p-2} \equiv 0 \pmod{2^p - 1}$. Thus, (5) $\Longrightarrow$ $u_{2^{p-2}} \not\equiv 0 \pmod{2^p - 1}$. Also, $u_{2n} = u_n v_n$ implies $u_{2^{p-1}} \equiv 0 \pmod{2^p - 1}$. It follows that $\alpha(2^p - 1) = 2^{p-1}$.

Write $2^p - 1 = p_1^{e_1} p_2^{e_2} \cdots p_r^{e_r}$ with p_j distinct primes and e_j positive integers. Each $p_j \ge 3$. Set

$$k = \operatorname{lcm}\left\{ p_j^{e_j - 1}(p_j + \epsilon_j) : j = 1, \ldots, r \right\}.$$

Here $\epsilon_j = \pm 1$ are chosen so that (4) holds with $p = p_j$ and $\epsilon = \epsilon_j$. Observe that (2), (4), and (6) imply $u_k \equiv 0 \pmod{2^p - 1}$. In particular, k is a multiple of $\alpha(2^p - 1) = 2^{p-1}$. By the definition of k, it follows that 2^{p-1} divides $p_j^{e_j - 1}(p_j + \epsilon_j)$ for some j. For such j, we have $p_j \ge 2^{p-1} - 1$. The inequality

$$3p_j \ge 2^p + 2^{p-1} - 3 > 2^p - 1$$

now implies that $2^p - 1$ must be prime.

Homework:

(1) Prove that each of the identities in (1) holds for arbitrary positive integers n and m.

(2) Prove that for every positive integer m, the number $\alpha = \alpha(m)$ in the proof above exists.

General Primality Tests:

- A polynomial time algorithm may exist:

Theorem (Selfridge-Weinberger). *Assume the Extended Riemann Hypothesis holds. Let n be an odd integer > 1. A necessary and sufficient condition for n to be prime is that for all positive integers $a < \min\{70(\log n)^2, n\}$, we have $a^{(n-1)/2} \equiv \pm 1 \pmod{n}$ with at least one occurrence of -1.*

- For $n = 1729$ and all integers $a \in [1, n-1]$ with $\gcd(a, n) = \gcd(a, 7 \times 13 \times 19) = 1$, $a^{(n-1)/2} \equiv 1 \pmod{n}$.

- Observe that $a < n$ exists when n is prime (consider a primitive root).

- **Theorem (Lucas).** *Let n be a positive integer. If there is an integer a such that $a^{n-1} \equiv 1 \pmod{n}$ and for all primes p dividing $n-1$ we have $a^{(n-1)/p} \not\equiv 1 \pmod{n}$, then n is prime.*

- For primes such an a exists (consider a primitive root).

- Prove the theorem.

- **Theorem (Pepin Test).** *Let $F_n = 2^{2^n} + 1$ with n a positive integer. Then F_n is prime if and only if $3^{(F_n - 1)/2} \equiv -1 \pmod{F_n}$.*

- Prove the theorem.

- **Theorem (Proth, Pocklington, Lehmer Test).** *Let n be a positive integer. Suppose $n - 1 = FR$ where all the prime factors of F are known and $\gcd(F, R) = 1$. Suppose further that there exists an integer a such that $a^{n-1} \equiv 1 \pmod{n}$ and for all* *primes p dividing F we have* $\gcd(a^{(n-1)/p} - 1, n) = 1$. *Then every prime factor of n is congruent to* 1 *modulo F.*

• Consider the case $F \geq \sqrt{n}$.

• Prove the theorem. (Let q be a prime divisor of n. Consider $m = \mathrm{ord}_q(a)$. Show $F|m$ by showing $p^e||F$ and $p^e \nmid m$ is impossible.)

• In 1980, Adleman, Pomerance, and Rumely developed a primality test that determines if n is prime in about $(\log n)^{c \log\log\log n}$ steps (shown by Odlyzko).

Factoring Algorithms (Part I):

• Given a composite integer $n > 1$, the general problem is to find some nontrivial factorization of n, say $n = uv$ where each of u and v is an integer > 1. If this can be done effectively and one has a good primality test, one will have a good method for completely factoring n.

• The expectation is that a random number n will have on the order of $\log\log n$ prime factors. Describe what this means but don't prove it.

• Most numbers n have a prime factor $> \sqrt{n}$. Prove using

$$\sum_{p \leq x} \frac{1}{p} = \log\log x + A + O(1/\log x).$$

• One expects typically small prime factors, so it is reasonable to first do a quick "sieve" to determine if this is the case.

• **Pollard's $p - 1$ Factoring Algorithm**
This algorithm determines a factorization of a number n if n has a prime factor p where $p - 1$ factors into a product of small primes. A simple form of the algorthm is to compute $2^{k!} \mod n$ successively for $k = 1, 2, \ldots$ until some prescribed amount (say 10^6 or 10^7), for each k checking $\gcd(2^{k!} - 1 \mod n, n)$ to possibly obtain a nontrivial factor of n. If $(p-1)|k!$, then $2^{k!} \equiv 1 \pmod{p}$ so that p will divide $\gcd(2^{k!} - 1 \mod n, n)$ and the chances of obtaining a nontrivial factor of n will be good.

• **Pollard's ρ-Algorithm**
This method typically finds a prime factor p of n in about $\sqrt{p}$ steps (so $O(N^{1/4})$ steps). Note that small prime factors will usually be found first.

• Preliminary observation. Suppose we roll a fair die with n faces k times. We claim that if $k \geq 2\sqrt{n}+2$, then with probability $> 1/2$ two of the numbers rolled will be the same. (Mention the birthday problem.) Use that there are at least $\sqrt{n}$ integers in $[\sqrt{n}, 2\sqrt{n}+1]$. The probability the numbers rolled are all different is

$$\prod_{j=1}^{k-1} \left(\frac{n-j}{n}\right) \leq \left(1 - \frac{\sqrt{n}}{n}\right)^{\sqrt{n}} \leq \frac{1}{e}.$$

So the result follows.

• The algorithm. Let $f(x) = x^2 + 1$. Let $f^{(j)}$ be defined by $f^{(1)}(x) = f(x)$ and $f^{(j+1)}(x) = f(f^{(j)}(x))$ for $j \geq 1$. Compute $a_j = f^{(j)}(1) \mod n$ for $1 \leq j \leq k$. The idea is that if $k \geq 100\sqrt{p}$ where p is a prime factor of n, then one can expect to find i and j with $1 \leq i < j \leq k$ such that $f^{(i)}(1) \mod p = f^{(j)}(1) \mod p$. In this case, we would have $a_i \equiv a_j \pmod{p}$ so that p will divide $\gcd(a_i - a_j, n)$. Hopefully then by computing $\gcd(a_i - a_j, n)$ we can determine a factorization of n. This will not be so good however if we actually compute $\binom{k}{2}$ different gcd's.

We instead use Floyd's cycle-finding algorithm. Observe that if i and j are as above, then (since there are $j - i$ integers in $(i, j]$) there is an integer $t \in (i, j]$ for which $(j - i)|t$. Now, $a_i \equiv a_j \pmod{p}$ implies $a_{i+u} \equiv a_{j+u} \pmod{p}$ for every positive integer u. In particular,

$$a_t \equiv a_{t+(j-i)} \equiv a_{t+2(j-i)} \equiv a_{t+3(j-i)} \equiv \cdots \equiv a_{2t} \pmod{p}.$$

Thus, rather than checking $\binom{k}{2}$ different gcd's as above, one can compute $a_1, a_2, \ldots$ and check as one progresses the values of $\gcd(a_{2t} - a_t, n)$ for $t = 1, 2, \ldots$. One continues until one finds a factorization of n, noting again that this should take $O(\sqrt{p})$ steps to find a given prime factor p.

• Brent and Pollard factored $F_8 = 2^{2^8} + 1$ using this method with $f(x) = x^{1024} + 1$. Discuss why such a choice for $f(x)$ would be appropriate here.

Dixon's Factoring Algorithm:

• The basic idea. 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 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 $\gcd(x + y, n)$.

• The algorithm.

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

(2) A bound $B = B(n)$ to be specified momentarily is chosen. 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

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

Compute the vectors

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

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.

• A (non-realistic) example. Take $n = 1189$ and $B = 11$. Suppose after step (3) we have for the a_i's the numbers 151, 907, 449, 642, 120, and 1108 with the $s(a_i)$'s being $210 = 2 \times 3 \times 5 \times 7$, $1050 = 2 \times 3 \times 5^2 \times 7$, $660 = 2^2 \times 3 \times 5 \times 11$, $770 = 2 \times 5 \times 7 \times 11$, $132 = 2^2 \times 3 \times 11$, and $1108 = 2^3 \times 7 \times 11$, respectively. Execute the rest of the algorithm.

Homework:

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 reduced modulo n should have all of its prime factors $\leq B$.)

• What about B? To analyze the optimal choice for B, we use

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

From previous discussions, $\psi(x, \sqrt{x}) \sim (1 - \log 2)x$. In general, $\psi(x, x^{1/u}) \sim \rho(u)x$ for some number $\rho(u)$.

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 $0 < u \leq 1$
(iv) for $u > 1$, $\rho(u)$ satisfies the differential delay equation $u\rho'(u) = -\rho(u-1)$.

Helmut Maier, improving on work of deBruijn, essentially removes any restriction on u by establishing the result above whenever $u < (\log x)^{1-\varepsilon}$ for any fixed $\varepsilon > 0$. Note that $x^{1/\log x} = e$. Also, deBruijn established that

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

We are interested in $\psi(n, B)$. Thus, $u = \log n / \log B$. Note that $u \leq \log n$. Suppose $u < (\log n)^{1-\varepsilon}$ as above. We deduce

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

The number of different times we expect to go through steps (1) and (2) of the algorithm is

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

We expect $\leq B$ steps to factor each value of $s(a)$. We are led to considering

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

The number of steps expected for Dixon's Algorithm is therefore $\exp\left(2\sqrt{\log n}\sqrt{\log\log n}\right)$ (take into account the Gaussian elimination).

• Preliminaries to the CFRAC Algorithm. Discuss simple continued fractions (scf). Mention that if a/b is a reduced convergent of the scf for α, then $|\alpha - (a/b)| < 1/b^2$. Deduce that $|a^2 - \alpha^2 b^2| \ll \alpha$.

• The CFRAC Algorithm. Use Dixon's Algorithm with the a_j's chosen so that a_j is the numerator of a reduced convergent of the scf for $\sqrt{n}$. If b_j is the denominator, then $|a_j^2 - nb_j^2| < 2\sqrt{n}$. In other words, if one modifies $s(a_j)$ so that it might be negative, we can take $|s(a_j)| < 2\sqrt{n}$. One can deal with negative $s(a_j)$ by treating -1 as a prime. The running time is improved as $\psi(n, B)$ above gets to be replaced by $\psi(2\sqrt{n}, B)$. One obtains here a running time of $O\left(\exp(\sqrt{2}\sqrt{\log n}\sqrt{\log\log n})\right)$. On the other hand, it has not been established that the values of $s(a_j)$ here are just as likely to have all prime factors $\leq B$ as random numbers their size, so this running time is heuristic. We note that this algorithm was used by Brillhart and Morrison to factor $F_7 = 2^{2^7} + 1$.

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

Homework:

Use the CFRAC Algorithm to factor $n = 135683$. The first 10 numerators of the convergents for the simple continued fraction for $\sqrt{n}$ and their squares modulo n (in factored form) are shown below. You may use this information. As far as B goes, ignore it (take it to be 1000 if you want). You do not need to use Gaussian elimination. Instead you can simply try to find the right combination of factors mentally.

Numerators	**Squares Mod n**
737	19×23
1105	$-1 \times 2 \times 61$
6262	257
13629	$-1 \times 2 \times 193$
19891	11×23
53411	$-1 \times 2 \times 7 \times 11$
233535	277
520481	$-1 \times 2 \times 7 \times 19$
1274497	11×19
4343972	-1×83

• The Quadratic Sieve Algorithm. Consider $F(x) = (x + [\sqrt{n}])^2 - n$. Then $|F(x)| \ll$

$|x|\sqrt{n}$ for $0 < |x| \le \sqrt{n}$. The idea of the quadratic sieve algorithm is to consider a in Dixon's Algorithm to be of the form $a = x + [\sqrt{n}]$ with $|x|$ small. Here, we allow for $s(a)$ to be negative as in the CFRAC Algorithm. Thus, $|s(a)| \ll |x|\sqrt{n}$.

• Why would this be better than the CFRAC Algorithm? In the CFRAC Algorithm, we had $|s(a)| < 2\sqrt{n}$, so this is a reasonable question. The advantage of the Quadratic Sieve Algorithm is that one can "sieve" prime divisors of $s(a)$ to determine how $s(a)$ factors for many a at once. To clarify, for a small prime p, one can solve the quadratic $F(x) \equiv 0 \pmod{p}$. If there are solutions to the congruence, there will usually be two incongruent solutions modulo p, say x_1 and x_2. Thus, one knows that if $x \equiv x_1$ or x_2 modulo p and $a = x + [\sqrt{n}]$, then $p|s(a)$. Otherwise, $p \nmid s(a)$.

• The running time is $O\left(\exp(\sqrt{9/8}\sqrt{\log n}\sqrt{\log\log n})\right)$ for the Quadratic Sieve Algorithm. The running time is heuristic.

• There are other variations of the above algorithms. In particular, a version of the Quadratic Sieve Algorithm suggested by Peter Montgomery reduces the running time to $O\left((1/2)\sqrt{\log n}\sqrt{\log\log n})\right)$.

The Number Field Sieve:

• Let f be an irreducible monic polynomial with integer coefficients. Let α be a root of f. Let m be an integer for which $f(m) \equiv 0 \pmod{n}$. The mapping $\phi : \mathbb{Z}[\alpha] \to \mathbb{Z}_n$ with $\phi(g(\alpha)) = g(m) \mod n$ for all $g(x) \in \mathbb{Z}[x]$ is a homomorphism. The idea is to find a set S of polynomials $g(x) \in \mathbb{Z}[x]$ such that (i) $\prod_{g\in S} g(m) = y^2$ for some $y \in \mathbb{Z}$, and (ii) $\prod_{g\in S} g(\alpha) = \beta^2$ for some $\beta \in \mathbb{Z}[\alpha]$. Taking $x = \phi(\beta)$, we deduce

$$x^2 \equiv \phi(\beta)^2 \equiv \phi(\beta^2) \equiv \phi\left(\prod_{g\in S} g(\alpha)\right) \equiv \prod_{g\in S} g(m) \equiv y^2 \pmod{n}.$$

Thus, we can hope once again to factor n by computing $\gcd(x+y, n)$.

• What do we choose for f? We determine an f of degree $d > 1$ and with $n > 2^{d^2}$ as follows. Set $m = [n^{1/d}]$. Write n in base m; in other words, compute $c_0, c_1, \dots, c_d$ each in $\{0, 1, \dots, m-1\}$ with

$$n = c_d m^d + c_{d-1} m^{d-1} + \cdots + c_1 m + c_0.$$

Set $f(x) = \sum_{j=0}^{d} c_j x^j$. Then f is monic and $f(m) \equiv 0 \pmod{n}$. Next, we attempt to factor f in $\mathbb{Z}[x]$. (Discuss the running time for this.) If it is irreducible, then we are happy. If it is reducible, then we use the factorization of f to determine a factorization of n (explain).

Homework:

Prove that f is monic. In other words, show that $c_d = 1$.

• What do we choose for the g? We take the g to be of the form $a - bx$ where $|a| \leq D$ and $0 < b \leq D$. We want $g(m)$ to have only small prime factors. This is done by first choosing b and then, with b fixed, letting a vary and sieving to determine the a for which $g(m)$ has only small prime factors.

• How do we obtain the desired square in $\mathbb{Z}[\alpha]$? Let $\alpha_1, \ldots, \alpha_d$ be the distinct roots of f with $\alpha = \alpha_1$. We consider the norm map $\mathrm{N}(g(\alpha)) = g(\alpha_1) \cdots g(\alpha_d)$, where $g \in \mathbb{Z}[x]$. It has the two properties: **(i)** if g and h are in $\mathbb{Z}[x]$, then $\mathrm{N}(g(\alpha)h(\alpha)) = \mathrm{N}(g(\alpha))\,\mathrm{N}(h(\alpha))$, and **(ii)** if $g \in \mathbb{Z}[x]$, then $\mathrm{N}(g(\alpha)) \in \mathbb{Z}$. It follows that the norm of a square in $\mathbb{Z}[\alpha]$ is a square in $\mathbb{Z}$. On the other hand,

$$\mathrm{N}(a - b\alpha) = b^d \prod_{j=1}^{d} \left(\frac{a}{b} - \alpha_j \right) = b^d f(a/b) = a^d + c_{d-1}a^{d-1}b + \cdots + c_1 ab^{d-1} + c_0 b^d.$$

The idea is to try to obtain a set S of pairs (a, b) as above. As we force the product $\prod(a - bm)$ to be a square (products over $(a, b) \in S$), we also force $\prod \left(a^d + c_{d-1}a^{d-1}b + \cdots + c_0 b^d\right)$ to be a square.

• Have we really answered the previous question? No. We have found an element of $\mathbb{Z}[\alpha]$ with its norm being a square in $\mathbb{Z}$. This does not mean the element is a square in $\mathbb{Z}[\alpha]$. (For example, consider $\alpha = \mathrm{i}$ and $S = \{(2, 1), (2, -1)\}$.) There is quite a bit of work left in the algorithm in terms of modifying the exponent vectors to produce the squares we want, but a small bit of the ideas have been demonstrated.

• The running time for the number field sieve is $\exp\left(c(\log n)^{1/3}(\log\log n)^{2/3}\right)$ where c is a constant and $c = 4/(3^{2/3})$ will do.

• The number field sieve was used to factor $F_9 = 2^{2^9} + 1$ in 1993 by Lenstra, Lenstra, Manasse, and Pollard.

Public-Key Encryption:

• **Problem:** *Describe how to communicate with someone you have never met before through the personals without anyone else understanding the private material you are sharing with this stranger.*

• What they don't know can hurt them. Find two large primes p and q; right now 100 digit primes will suffice. Compute $n = pq$. If you are secretive about your choices for p and q, then you can tell the world what n is and you will know something no one else in the world knows, namely how n factors. Now, don't you feel special?

• What to publish first. Now that you know something no one else knows, the problem becomes a bit easier to resolve. You (and you alone) can determine $\phi(n)$. Do so, and choose some positive integer s (the "encrypting exponent") with $\gcd(s, \phi(n)) = 1$. You publish n and s in the personals. You also describe to someone how to correspond with you in such a way that no one else will understand the message (at this point everyone can read what you are writing, but that's OK as it is only a temporary situation). What you tell them is something like this: to form a message M concatenate the symbols 00 for blank, 01 for a, 02 for b, ..., 26 for z, 27 for a comma, 28 for a period, and whatever else you might want. For example, $M = 0805121215$ means "hello". Next, tell the person to publish (back in the personals) the value of $E = M^s \mod n$. (The person should be told to make sure that $M^s > n$ by adding extra blanks if necessary and that $M < n$ by breaking up a message into two or more messages if necessary.)

• What can you do with the encoded message? You calculate t with $st \equiv 1 \pmod{\phi(n)}$ (one can use $t \equiv s^{\phi(\phi(n))-1} \mod \phi(n)$). Then compute $E^t \mod n$. This will be the same as M modulo n (unless p or q divides M, which isn't likely). So now you know the message.

• Computing $\phi(n)$ is seemingly as difficult as factoring n. Here one needs to compute $\phi(\phi(n))$ which for you is only as difficult as factoring $p-1$ and $q-1$. If these each have 100 digits, then n has around 200 digits and all is reasonable (for the times we live in). One can also try to construct p and q with $p-1$ and $q-1$ of some nice form (for example, having small prime factors).

Homework:

Let $p = 193$ and $q = 257$, so $n = pq = 49601$. Let $s = 247$.
(a) Someone sends the encrypted message $E = 48791$. Determine the word sent.
(b) Encrypt the message, "No". In other words, tell me the value of E.

• Certified signatures. Imagine person A has published n and s in the personals, person B is corresponding with person A in private, and person C really dislikes person B. C decides to send A a message in the personals that reads something like, "Dear A, I think you are a jerk. Your dear friend, B." This of course would make A very upset with B and would make C very happy. What would be nice is if there were a way for B to sign his messages so that A can see the signature and know whether a message supposedly from B is really from B. This is done as follows. First, if B is really corresponding with A, he (B stands for boy) should have his very own n and s which he has shared with at least A. Let's call them n' and s', and let the corresponding t be t'. Now, B doesn't have to use his name, but he informs A of some signature (name) he will use, say S. He can change S regularly if he wishes, but in any case, it is given to A as part of an encrypted message. At the end of the encrypted message, he gives A the number $T = S^{t'} \mod n'$. After A decodes the message, he computes $T^{s'} \mod n'$ (remember n' and s' are public). The result will be S. Since only B knows t', only B could have determined T and A will know that the message really came from B.

Factoring Polynomials:

• **Problem:** *Given a polynomial in $\mathbb{Z}[x]$, determine if it is irreducible over $\mathbb{Q}$. If it is reducible, find a non-trivial factorization of it in $\mathbb{Z}[x]$.*

• **Berlekamp's Algorithm.** This algorithm determines the factorization of a polynomial $f(x)$ in $\mathbb{Z}_p[x]$ where p is a prime (or more generally over finite fields). For simplicity, we suppose $f(x)$ is monic and squarefree in $\mathbb{Z}_p[x]$. Let $n = \deg f(x)$. Let A be the matrix with jth column derived from the coefficients reduced modulo p of $x^{(j-1)p} \mod f(x)$.

Specifically, write

$$x^{(j-1)p} \equiv \sum_{i=1}^{n} a_{ij} x^{i-1} \mod f(x) \qquad \text{for } 1 \le j \le n.$$

Then we set $A = (a_{ij} \mod p)_{n \times n}$. Note that the first column consists of a one followed by $n-1$ zeroes. In particular, $\langle 1, 0, 0, \dots, 0 \rangle$ will be an eigenvector for A associated with the eigenvalue 1. We are interested in determining the complete set of eigenvectors associated with the eigenvalue 1. In other words, we would like to know the null space of $B = A - I$ where I represents the $n \times n$ identity matrix. It will be spanned by $k = n - \text{rank}(B)$ linearly independent vectors which can be determined by performing column operations on B. Suppose $\vec{v} = \langle b_1, b_2, \dots, b_n \rangle$ is one of these vectors, and set $g(x) = \sum_{j=1}^{n} b_j x^{j-1}$. Observe that $g(x^p) \equiv g(x) \pmod{f(x)}$ in $\mathbb{Z}_p[x]$. Moreover, the $g(x)$ with this property are precisely the $g(x)$ with coefficients obtained from the components of vectors $\vec{v}$ in the null space of B.

Claim. $f(x) \equiv \prod_{s=0}^{p-1} \gcd\left(g(x) - s, f(x)\right) \pmod{p}$.

The proof follows by using that $f(x)$ divides $g(x)^p - g(x) \equiv \prod_{s=0}^{p-1} \left(g(x) - s\right)$ in $\mathbb{Z}_p[x]$ and that each irreducible factor of $f(x)$ in $\mathbb{Z}_p[x]$ divides at most one of the $g(x) - s$. Observe that if $g(x)$ is not a constant, then $1 \le \deg(g(x) - s) < \deg f(x)$ for each s so the above claim implies we get a non-trivial factorization of $f(x)$ in $\mathbb{Z}_p[x]$. On the other hand, $f(x)$ will not necessarily be completely factored. One can completely factor $f(x)$ by repeating the above procedure for each factor obtained from the claim; but it is simpler to use (and not difficult to show) that if one takes the product of the greatest common divisors of each factor of $f(x)$ obtained above with $h(x) - s$ (with $0 \le s \le p - 1$) where $h(x)$ is obtained from another of the k vectors spanning the null space of B, then one will obtain a new non-trivial factor of $f(x)$ in $\mathbb{Z}_p[x]$. Continuing to use all k vectors will produce a complete factorization of $f(x)$ in $\mathbb{Z}_p[x]$. (As an example of Berlekamp's algorithm, factor $x^7 + x^4 + x^3 + x + 1$ in $\mathbb{Z}_2[x]$.)

- **Hensel Lifting Algorithm.** This algorithm gives a method for using the factorization of $f(x)$ in $\mathbb{Z}_p[x]$ (p a prime) to produce a factorization of $f(x)$ in $\mathbb{Z}_{p^k}[x]$. Suppose that $u(x)$ and $v(x)$ are relatively prime polynomials in $\mathbb{Z}_p[x]$ for which

$$f(x) \equiv u(x)v(x) \pmod{p}.$$

Then Hensel Lifting will produce for any positive integer k polynomials $u_k(x)$ and $v_k(x)$ in $\mathbb{Z}[x]$ satisfying

$$u_k(x) \equiv u(x) \pmod{p}, \quad v_k(x) \equiv v(x) \pmod{p},$$

and

$$f(x) \equiv u_k(x)v_k(x) \pmod{p^k}.$$

When $k = 1$, it is clear how to choose $u_k(x)$ and $v_k(x)$. For $k \ge 1$, we determine values of $u_{k+1}(x)$ and $v_{k+1}(x)$ from the values of $u_k(x)$ and $v_k(x)$ as follows. We compute

$$w_k(x) \equiv \frac{1}{p^k}\left(f(x) - u_k(x)v_k(x)\right) \pmod{p}.$$

Since $u(x)$ and $v(x)$ are relatively prime in $\mathbb{Z}_p[x]$, we can find $a(x)$ and $b(x)$ in $\mathbb{Z}_p[x]$ (depending on k) such that

$$a(x)u(x) + b(x)v(x) \equiv w_k(x) \pmod{p}.$$

It follows that we can take

$$u_{k+1}(x) = u_k(x) + b(x)p^k \quad \text{and} \quad v_{k+1}(x) = v_k(x) + a(x)p^k.$$

A complete factorization of $f(x)$ modulo p^k can be obtained from a complete factorization of $f(x)$ modulo p by modifying this idea. However, note that $f(x) = 2x^2 + 5x + 3 = (x+1)(2x+3)$ satisfies

$$f(x) \equiv \left(x + \frac{3^k+3}{2}\right)(2x+2) \pmod{3^k}.$$

Claim. For $f(x) \in \mathbb{C}[x]$, $M(f) \leq \|f\| \leq 2^{\deg f} M(f)$.

For the claim, we define for a given $w(x) \in \mathbb{C}[x]$, $\tilde{w}(x) = x^{\deg w} w(1/x)$.

• **Mignotte's Inequality.** This inequality gives an upper bound on the "size" of the factors of a given polynomial in $\mathbb{Z}[x]$. Given $f(x) = \sum_{j=0}^n a_j \in \mathbb{Z}[x]$, we measure it's size with $\|f\| = \left(\sum_{j=0}^n a_j^2\right)^{1/2}$. Thus for a fixed $f(x) \in \mathbb{Z}[x]$, we want an upper bound on $\|g\|$ where $g(x)$ is a factor of $f(x)$ in $\mathbb{Z}[x]$.

Theorem. *If $f(x)$, $g(x)$, and $h(x)$ in $\mathbb{Z}[x]$ are such that $f(x) = g(x)h(x)$, then $\|g\| \leq 2^{\deg g}\|f\|$.*

For $f(x) = a_n \prod_{j=1}^n (x - \alpha_j)$ with $a_n \neq 0$ and α_j complex but not necessarily distinct, we define the Mahler measure of f as $M(f) = |a_n| \prod_{j=1}^n \max\{1, |\alpha_j|\}$. The following two properties are easily seen to hold: (i) if $g(x)$ and $h(x)$ are in $\mathbb{C}[x]$, then $M(gh) = M(g)M(h)$; (ii) if $g(x)$ is in $\mathbb{Z}[x]$, then $M(g) \geq 1$.

Claim. For $f(x) \in \mathbb{R}[x]$, $M(f) \leq \|f\| \leq 2^{\deg f} M(f)$.

For the claim, we define for a given $w(x) \in \mathbb{C}[x]$, $\tilde{w}(x) = x^{\deg w} w(1/x)$. The coefficient of $x^{\deg w}$ in the expanded product $w(x)\tilde{w}(x)$ is $\|w\|^2$. For $f(x) = \sum_{j=0}^n a_j x^j = a_n \prod_{j=1}^n (x - \alpha_j)$, we consider

$$w(x) = a_n \prod_{\substack{1 \leq j \leq n \\ |\alpha_j| > 1}} (x - \alpha_j) \prod_{\substack{1 \leq j \leq n \\ |\alpha_j| \leq 1}} (\alpha_j x - 1).$$

One checks that

$$w(x)\tilde{w}(x) = a_n^2 \prod_{j=1}^n (x - \alpha_j) \prod_{j=1}^n (1 - \alpha_j x) = f(x)\tilde{f}(x).$$

By comparing coefficients of x^n, we deduce that $\|w\| = \|f\|$. Also, observe that from the definition of w, $|w(0)| = M(f)$. Thus, if $w = \sum_{j=0}^n c_j x^j$, then

$$M(f) = |c_0| \leq (c_0^2 + c_1^2 + \cdots + c_n^2)^{1/2} = \|w\| = \|f\|,$$

establishing the first inequality. For the second inequality observe that for any $k \in \{1, 2, \ldots, n\}$, the product of any k of the α_j has absolute value $\leq M(f)/|a_n|$. It follows that $|a_{n-k}|/|a_n|$, which is the sum of the products of the roots taken k at a time, is $\leq \binom{n}{k} \times M(f)/|a_n|$. Hence, $|a_{n-k}| \leq \binom{n}{k} M(f) = \binom{n}{n-k} M(f)$. The second inequality now follows from

$$\|f\| = \left(\sum_{j=0}^{n} a_j^2 \right)^{1/2} \leq \sum_{j=0}^{n} |a_j| \leq \sum_{j=0}^{n} \binom{n}{j} M(f) = 2^n M(f).$$

To prove the theorem, just use the Claim and properties (i) and (ii) of Mahler measure to deduce

$$\|g\| \leq 2^{\deg g} M(g) \leq 2^{\deg g} M(g) M(h) = 2^{\deg g} M(gh) = 2^{\deg g} M(f) \leq 2^{\deg g} \|f\|.$$

• **Combining the above ideas.** We factor a given $f(x) \in \mathbb{Z}[x]$ with the added assumptions that it is monic and squarefree. The latter we can test by computing $\gcd(f, f')$, which will give us a nontrivial factor of f if f is not squarefree. If f were not monic, a little more needs to be added to the ideas below (but not much).

Let $B = 2^{(\deg f)/2} \|f\|$. Then if f has a nontrivial factor g in $\mathbb{Z}[x]$, it has such a factor of degree $\leq (\deg f)/2$ so that by Mignotte's inequality, we can use B as a bound on $\|g\|$.

Next, we find a prime p for which f is squarefree modulo p. There are a variety of ways this can be done. There are only a finite number of primes for which f is not squarefree modulo p (these primes divide the resultant of f and f'). Working with $\gcd(f, f')$ modulo p can resolve the issue or simply using Berlekamp's factoring algorithm until a squarefree factorization occurs is fine.

We choose a positive integer r as small as possible such that $p^r > 2B$. One factors f modulo p by Berlekamp's algorithm and uses Hensel lifting to obtain the factorization of f modulo p^r. Given our conditions on f, we can suppose all irreducible factors are monic and do so.

Now, we can determine if $f(x) = g(x)h(x)$ for some monic g and h in $\mathbb{Z}[x]$ with $\|g\| \leq B$ as follows. We observe that the coefficients of g are in $[-B, B]$. We use a residue system modulo p^r that includes this interval, namely $(-p^r/2, p^r/2]$, and consider each factorization of f modulo p^r with coefficients in this residue system as a product of two monic polynomials $u(x)$ and $v(x)$. Since $f = gh$, there must be some factorization where $g \equiv u \pmod{p^r}$ and $h \equiv v \pmod{p^r}$. On the other hand, the coefficients of g and u are all in $(-p^r/2, p^r/2]$ so that the coefficients of $g - u$ are each divisible by p^r and are each $< p^r$ in absolute value. This implies $g = u$. Thus, we can determine if a factor g exists as above by simply checking each monic factor of f modulo p^r with coefficients in $(-p^r/2, p^r/2]$.