Addition and Subtraction

How fast do we add (or subtract) two numbers n and m?

How fast can we add (or subtract) two numbers n and m?

Definition. Let $A(d)$ denote the maximal number of steps required to add two numbers with $\leq d$ bits.

Theorem. $A(d) \asymp d$.

Theorem. $S(d) \asymp d$.

Multiplication

How fast do we multiply two numbers n and m?

How fast can we multiply two numbers n and m?

Definition. Let $M(d)$ denote the number of steps required to multiply two numbers with $\leq d$ bits.

Theorem. $M(d) \ll d^2$.

Can we do better? Yes

How can we see "easily" that something better is possible?

Attempt 2

Definition. Let $M(d)$ denote the number of steps required to multiply two numbers with $\leq d$ bits.

- Suppose $M(d) \gg d^{1.5}$.
- Let d be large, and let $\varepsilon > 0$.
- Let n and m have $\leq d$ bits, and write $n = a_n \times 2^r + b_n$ and $m = a_m \times 2^r + b_m$, where $r = \lfloor d/2 \rfloor$ and the a_j and b_j are integers with $b_j < 2^r$.
- From

  $$nm = a_n a_m 2^{2r} + \big((a_n+b_n)(a_m+b_m) - a_n a_m - b_n b_m\big)2^r + b_n b_m,$$

  deduce $M(d) \leq 3M(r+2) + O(r) \leq (3+\varepsilon)M(r+2)$.
- Hence, $M(d) \leq (3+\varepsilon)^s M\big((d + 2^{s+2} - 4)/2^s\big)$.
- Take $s = \lfloor \log_2 d \rfloor - C$ (with C big). Then $2^s \geq d/2^{C+1}$.
- Conclude, $M(d) \ll (3+\varepsilon)^{\log_2 d} = d^{\log(3+\varepsilon)/\log 2}$.

Theorem. $M(d) \ll d^2$.

- Conclude, $M(d) \ll (3+\varepsilon)^{\log_2 d} = d^{\log(3+\varepsilon)/\log 2}$.

$$\frac{\log 3}{\log 2} = 1.5849625$$

Theorem. $M(d) \ll d^{1.585}$.

HW: Due September 7 (Friday)
Page 3, Problems 1 and 2
Page 5, unnumbered homework (first set)
(you may use $(\log 5/\log 3) + \varepsilon$ instead of $\log 5/\log 3$)

Idea for Doing Better

- Let n and m have $\leq d$ bits, and write $n = a_n \times 2^r + b_n$ and $m = a_m \times 2^r + b_m$, where $r = \lfloor d/2 \rfloor$ and the a_j and b_j are integers with $b_j < 2^r$.

- From

  $$nm = a_n a_m 2^{2r} + \big((a_n + b_n)(a_m + b_m) - a_n a_m - b_n b_m\big) 2^r + b_n b_m,$$

  deduce $M(d) \leq 3M(r+2) + O(r) \leq (3+\varepsilon) M(r+2)$.

Think in terms of writing

$$n = a_n 2^{2r} + b_n 2^r + c_n \quad \text{and} \quad m = a_m 2^{2r} + b_m 2^r + c_m,$$

where $r = \lfloor d/3 \rfloor$.

How many multiplications does it take to expand nm?

Theorem. *For every* $\varepsilon > 0$*, we have* $M(d) \ll_\varepsilon d^{1+\varepsilon}$*.*

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

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

Lagrange Interpolation:

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

Theorem. *For every* $\varepsilon > 0$, *we have* $M(d) \ll_\varepsilon d^{1+\varepsilon}$.

- Suppose n and m have $\leq kr$ digits. Write

$$n = \sum_{u=0}^{k-1} a_u 2^{ur} \quad \text{and} \quad m = \sum_{v=0}^{k-1} b_v 2^{vr}.$$

- Then $nm = f(2^r)$, where $f(x) = \left(\sum_{u=0}^{k-1} a_u x^u\right)\left(\sum_{v=0}^{k-1} b_v x^v\right)$.

- Compute the $2k-1$ numbers $y_j = f(j)$, for $0 \leq j \leq 2k-2$

Theor

- Sup

Lagrange Interpolation:

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

- Then $nm = f(2^r)$, where $f(x) = \left(\sum_{u=0}^{k-1} a_u x^u \right) \left(\sum_{v=0}^{k-1} b_v x^v \right)$.

- Compute the $2k-1$ numbers $y_j = f(j)$, for $0 \le j \le 2k-2$, using $2k-1$ multiplications of two $\le r + c_k$ digit numbers.

- Compute the coefficients of $f(x)$ expanded, using Lagrange interpolation, in $O_k(r)$ steps.

Theorem. *For every $\varepsilon > 0$, we have $M(d) \ll_\varepsilon d^{1+\varepsilon}$.*

- Suppose n and m have $\leq kr$ digits. Write

$$n = \sum_{u=0}^{k-1} a_u 2^{ur} \quad \text{and} \quad m = \sum_{v=0}^{k-1} b_v 2^{vr}.$$

- Then $nm = f(2^r)$, where $f(x) = \left(\sum_{u=0}^{k-1} a_u x^u\right)\left(\sum_{v=0}^{k-1} b_v x^v\right)$.

- Compute the $2k-1$ numbers $y_j = f(j)$, for $0 \leq j \leq 2k-2$, using $2k-1$ multiplications of two $\leq r + c_k$ digit numbers.

- Compute the coefficients of $f(x)$ expanded, using Lagrange interpolation, in $O_k(r)$ steps.

- Deduce $M(kr) \leq (2k-1)M(r+c_k) + c_k' r$ so that

$$M(d) \ll (2k-1)^{\log_k d} \ll d^{\log(2k-1)/\log k}. \qquad \blacksquare$$

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

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

Definition. Let $M'(d)$ denote an upper bound on the number of steps required to multiply two numbers with $\le d$ bits. Let $D'(d)$ denote an upper bound on the number of steps required to obtain q and r given n and m each have $\le d$ binary digits.

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

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**

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

We need only compute $1/m$ to sufficient accuracy.

Suppose n and m have $\le$ s digits.

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

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

We need only compute $1/m$ to sufficient accuracy.

Suppose n and m have $\le s$ digits. If $1/m = 0.d_1d_2d_3d_4\ldots$ (base 2) with $d_1, \ldots, d_s$ known, then

$$\frac{n}{m} = \frac{1}{2^s}(n \times d_1d_2\ldots d_s) + \theta, \quad \text{where } 0 \le \theta \le 1.$$

Write this in the form

$$\frac{n}{m} = \frac{1}{2^s}(q'2^s + q'') + \theta,$$

so $n = mq' + \theta'$ where $0 \le \theta' < 2m$. Try $q = q'$ and $q = q' + 1$.

Newton's Method

Say we want to compute $1/m$.

Newton's Method

Say we want to compute $1/m$. Take a function $f(x)$ which has root $1/m$. If x' is an approximation to the root, then how can we get a better approximation? Take

$$f(x) = m - 1/x.$$

Starting with $x' = x_0$, this leads to the approximations

$$x_{n+1} = 2x_n - mx_n^2.$$

Note that if $x_n = (1 - \varepsilon)/m$, then $x_{n+1} = (1 - \varepsilon^2)/m$.

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

Algorithm R. Let v in binary be $v = (0.v_1v_2v_3\ldots)_2$, with $v_1 = 1$. The algorithm outputs z satisfying

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

$z \in [0, 2]$

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

R2. [Newton iteration] (At this point, $z \le 2$ has the binary form $(**.** \cdots *)_2$ with 2^k+1 places after the radix point.) Calculate z^2 exactly. Then calculate V_kz^2 exactly, where $V_k = (0.v_1v_2\ldots v_{2^{k+1}+3})_2$. Then set $z \leftarrow 2z - V_kz^2 + r$, where $0 \le r < 2^{-2^{k+1}-1}$ is added if needed to "round up" z so that it is a multiple of $2^{-2^{k+1}-1}$. Finally, set $k \leftarrow k+1$.

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