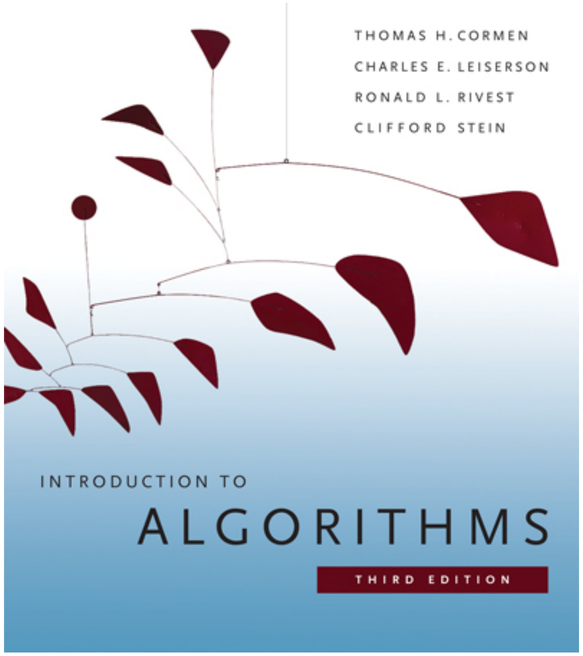
6.006- *Introduction to Algorithms*



Lecture 22

Piotr Indyk

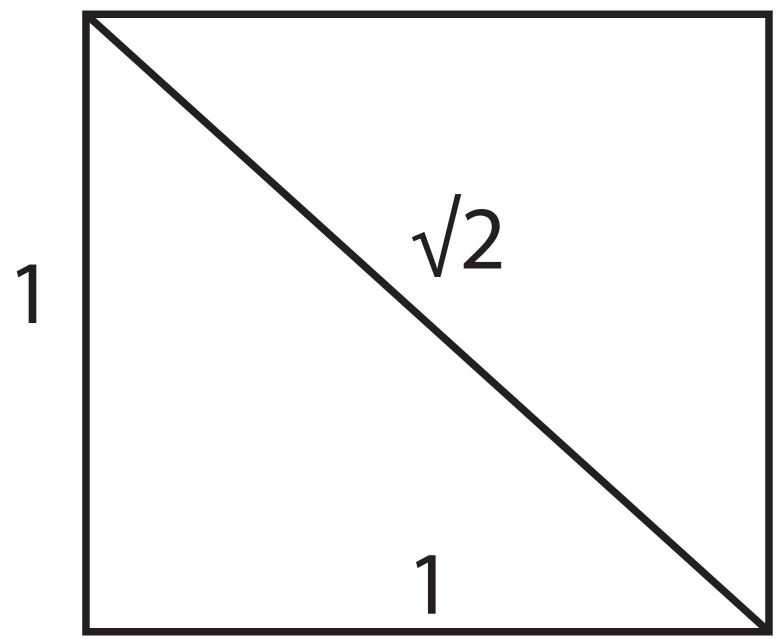
Outline

- "Numerics" - algorithms for operations on large numbers
  - high precision
  - cryptography, simulations, etc
- We will see:
  - irrationals
  - large number operations:
    - multiplication
    - division
    - matrix multiplication

3.14159265358979323846264338327950288419
7169399375105820974944592307816406286208
9986280348253421170679821480865132823066
4709384460955058223172535940812848111745
0284102701938521105559644622948954930381
9644288109756659334461284756482337867831
6527120190914564856692346034861045432664
8213393607260249141273724587006…

2.4142135623730950488016887242096980785696718753769
480731766797379907324784621070388503875343276415727
350138462309122970249248360558507372126441214970999
358314132226659275055927557999505011527820605714701
095599716059702745345968620147285174186408891986095
523292304843087143214508397626036279952514079896872
533965463318088296406206152583523950547457502877599
61729835575220337531857011354374603 ...

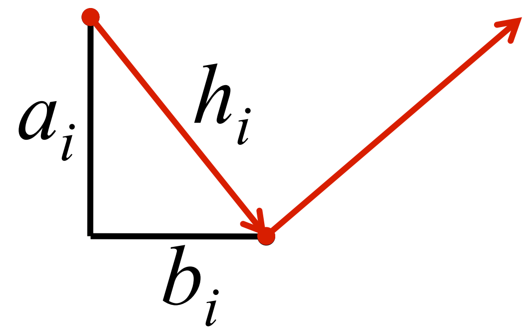
Computing $\sqrt{h}$ to lots of digits ... why?



1. 414 213 562 373 095
048 801 688 724 209
698 078 569 671 875
376 948 073 176 679 ...

Computing $\sqrt{h}$ to lots of digits ... why?

- High precision may be needed in some applications
- Consider Dijkstra for paths between points on plane:
  - lengths have form $\sum_i \sqrt{h_i}$
  - where $h_i^2 = a_i^2 + b_i^2$



- Is $\sqrt{1}+\sqrt{40}+\sqrt{60} > \sqrt{12}+\sqrt{17}+\sqrt{56}$?

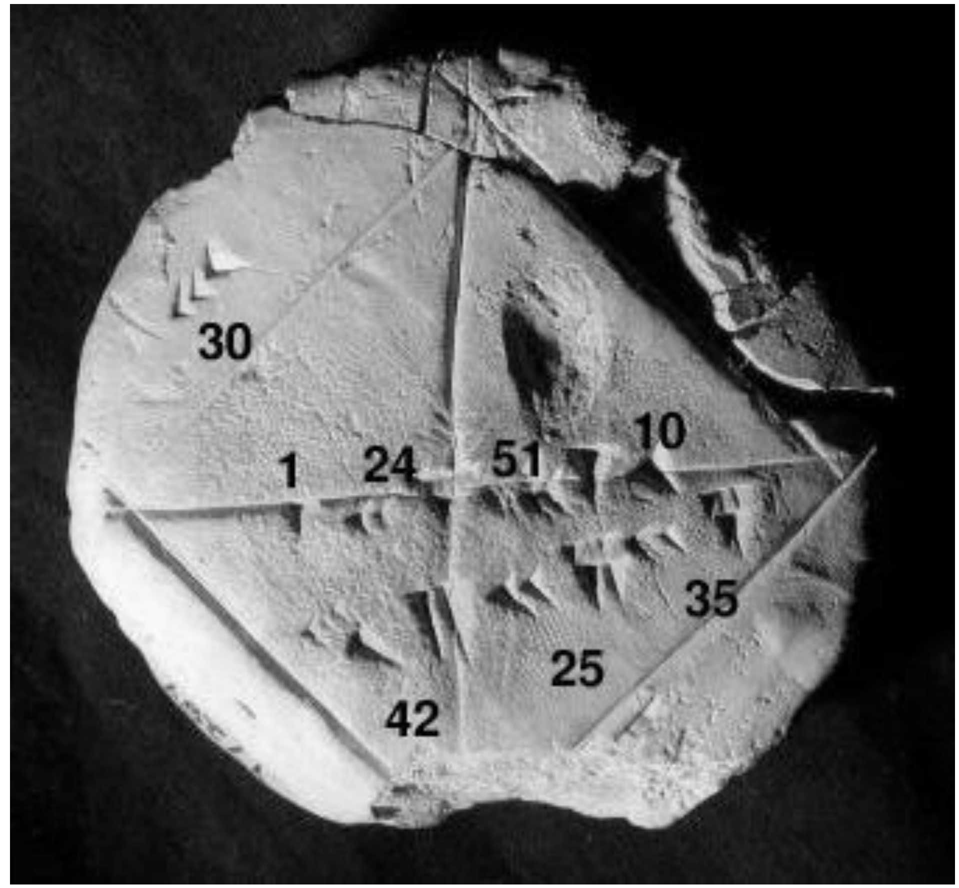
$$\sqrt{1}+\sqrt{40}+\sqrt{60} = 15.07052201275$$
$$\sqrt{12}+\sqrt{17}+\sqrt{56} = 15.07052201430$$

Computing $\sqrt{h}$; Babylonian method

- Iterative approach
- Also called the Heron's method

- $y_0 = h;\ x_0 = 1$
- $y_1 = (x_0 + y_0)/2;\ x_1 = h/(y_1)$

- In general
  - $y_{i+1} = (x_i + y_i)/2$
  - $x_{i+1} = h/(y_{i+1})$

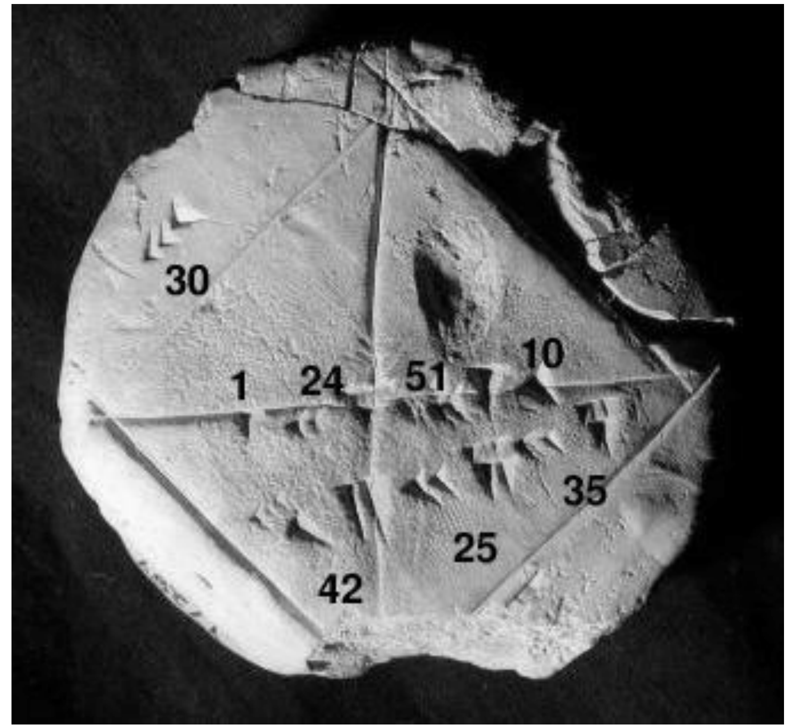


c. 1700 BC

Since $x_0 + y_0 - 2x_0^{1/2} y_0^{1/2} = (x_0^{1/2} - y_0^{1/2})^2 \geq 0$,
we have $(x_0 + y_0)/2 \geq x_0^{1/2} y_0^{1/2} = h^{1/2}$

Computing $\sqrt{h}$; Babylonian method

- $y_0 = h; x_0 = 1$
- $y_1 = (x_0 + y_0)/2; x_1 = h/(y_1)$
- In general
  - $y_{i+1} = (x_i + y_i)/2$
  - $x_{i+1} = h/(y_{i+1})$
- Convergence ?
- "Fast"
- Example for $h=2$



	x	y
0	1.000000000000000	2.000000000000000
1	1.333333333333330	1.500000000000000
2	1.411764705882350	1.416666666666670
3	1.414211438474870	1.414215686274510
4	1.414213562371500	1.414213562374690
5	1.414213562373090	1.414213562373090

Computing $\sqrt{h}$; Babylonian method

- Formula:
  - $y_{i+1} = (x_i + y_i)/2$
  - $x_{i+1} = h/(y_{i+1})$
- Need to be able to:
  - Add
  - Divide

    (or multiply + compute the inverse)

Large number addition

- Given: two positive n-digit numbers x,y with radix r
  (e.g., $r=2$, $r=10$, $r=60$)
- Goal: compute $x+y$, using only operations on single digits
- Algorithm: keep adding digits from right to left, keeping track of carryovers
- Time = $O(n)$

$$\begin{array}{c} x_{n-1}\, x_{n-2} \ldots x_1\; x_0 \\ + \\ y_{n-1}\, y_{n-2} \ldots y_1\; y_0 \end{array}$$

Large number multiplication

- Given: two positive n-digit numbers x,y with radix r
- Goal: compute $x*y$, using only operations on single digits
- Ideas ?

$$x_{n-1}\, x_{n-2} \ldots x_1\, x_0$$

$$*$$

$$y_{n-1}\, y_{n-2} \ldots y_1\, y_0$$

Divide and conquer: attempt I

$$x_{n-1}\ x_{n-2}\ \ldots\ x_1\ x_0$$

$$y_{n-1}\ y_{n-2}\ \ldots\ y_1\ y_0$$

- Split each number into "high" and "low" parts, each $n/2$ digits long
  - $x = x_H r^{n/2} + x_L$
  - $y = y_H r^{n/2} + y_L$
- We have

  $$x*y = (x_H r^{n/2} + x_L)*(y_H r^{n/2} + y_L)$$
  $$= x_H*y_H\, r^n + (x_H*y_L + x_L*y_H) r^{n/2} + x_L*y_L$$
- Algorithm for mult(x,y):

  a= mult(x_H,y_H), b= mult(x_H,y_L), c= mult(x_L,y_H), d= mult(x_L,y_L)

  return add(lshift(a,n), lshift(b,n/2), lshift(c,n/2), d)

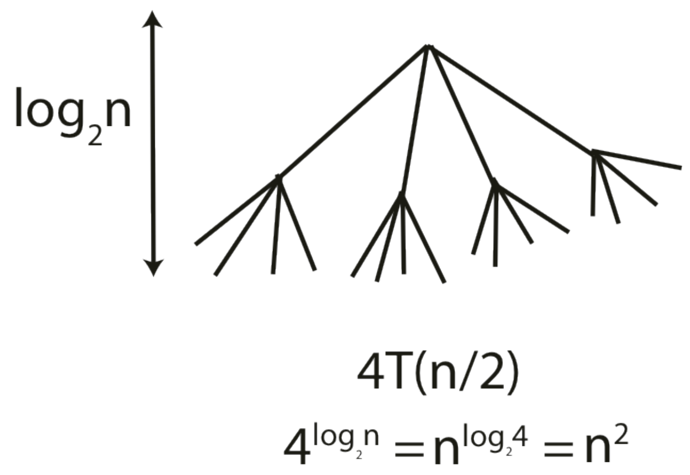
Attempt I: analysis

- Algorithm:

  a= mult(x_H,y_H), b= mult(x_H,y_L), c= mult(x_L,y_H), d= mult(x_L,y_L)

  return add(lshift(a,n), lshift(b,n/2), lshift(c,n/2), d)

- Running time T(n) ?
  - recurrence
    - $T(n)=4\ T(n/2)+O(n)$
  - this solves to
    - $T(n)=O(n^2)$
- How to improve on this ?



Attempt II

- Can we compute

$$x*y = x_H*y_H\, r^n + (x_H*y_L + x_L*y_H) r^{n/2} + x_L*y_L$$

  using fewer (than 4) recursive multiplications ?
- Here is how to do it (Karatsuba'62)

  $a = x_H*y_H$

  $d = x_L*y_L$

  $e = (x_H+x_L) * (y_H+y_L) - a - d$

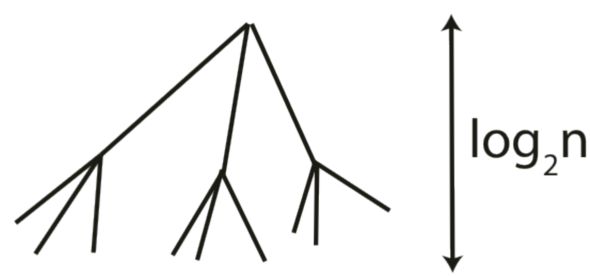
  $\quad = x_H*y_H + (x_H*y_L+x_L*y_H) + x_L*y_L - a - d$

  $\quad = x_H*y_L+x_L*y_H$

  $x*y = a\, r^n + e\, r^{n/2} + d$
- This leads to
  - $T(n)=3\, T(n/2) + O(n)$

  which solves to

$T(n)=O(n^{\log 3}) = O(n^{1.58496\ldots})$



$3T(n/2)$

$3^{\log_2 n} = n^{\log_2 3}$

Multiplication: further results

- Toom-Cook:
  - Split into 3 parts
  - 5 multiplications
  - $O(n^{\log 3(5)}) = O(n^{1.46..})$
- Schönhage and Strassen'71:
  - Uses Fast Fourier Transform
  - $O(n \log(n) \log(\log(n)))$
- Furer'07
  - $O(n \log(n)\, 2^{\Theta(\log^*(n))})$
  - $\log^*(n)$: number of times we need to apply logarithm to n until we get a number ≤ 1
  - E.g.,

  $\log^*(2^{65536}) = 1+\log^*(65536)=2+\log^*(16)=3+\log^*(4)=4+\log^*(2)=5$

Division: Newton's method

- Newton's method: Iterative approach to solving $f(x)=0$
- Iterative step:
  - find a line $y= f(x_i)+f'(x_i)(x- x_i)$ tangent to $f(x)$ at x_i
  - set x_{i+1} to the solution of $f(x_i)+f'(x_i)(x- x_i) =0$
    $\Rightarrow x_{i+1}=x_i-f(x_i)/f'(x_i)$