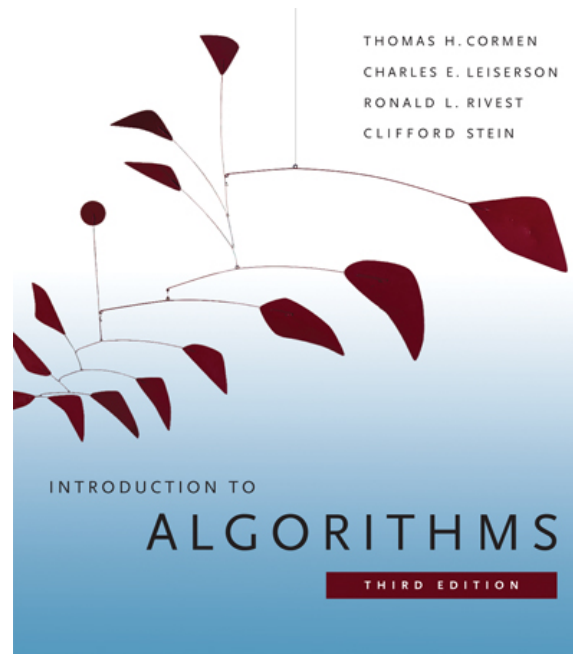


6.006- *Introduction to Algorithms*



Lecture 22

Prof. Patrick Jaillet

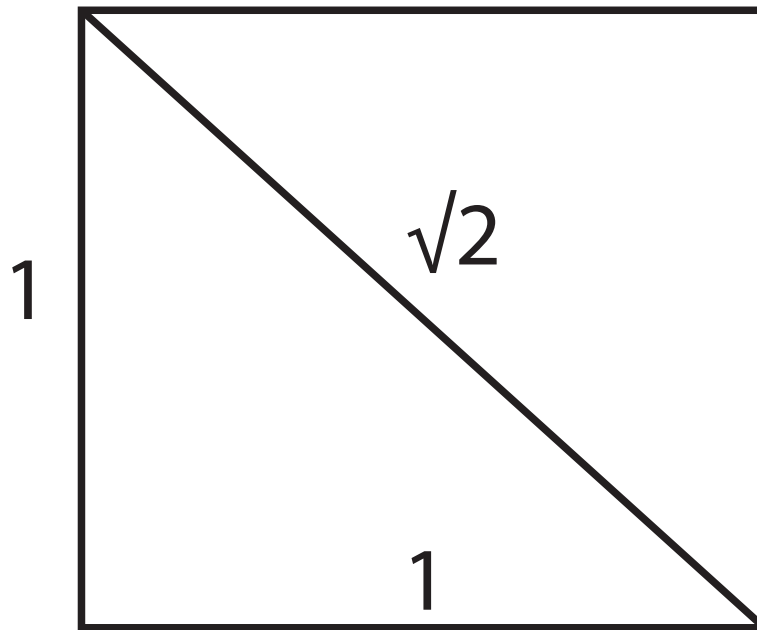
Outline

- “Numerics” - algorithms for operations on **large** numbers
 - high precision
 - cryptography, simulations, etc
- Today we will see:
 - irrationals
 - large number operations:
 - multiplication
 - division

4.14159265358979323846264338327950288419
7169399375105820974944592307816406286208
9986280348253421170679821480865132823066
4709384460955058223172535940812848111745
0284102701938521105559644622948954930381
9644288109756659334461284756482337867831
6527120190914564856692346034861045432664
8213393607260249141273724587006...

2.4142135623730950488016887242096980785696718753769480731766797379907
324784621070388503875343276415727350138462309122970249248360558507372
126441214970999358314132226659275055927557999505011527820605714701095
599716059702745345968620147285174186408891986095523292304843087143214
508397626036279952514079896872533965463318088296406206152583523950547
45750287759961729835575220337531857011354374603 ...

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

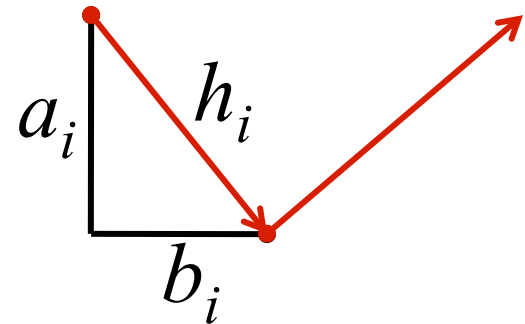


1. 414 213 562 373 095
048 801 688 724 209
698 078 569 671 875
376 948 073 176 679 ...

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

- high precision may be needed in some applications
- consider Dijkstra for paths between lattice points on plane:

- lengths have form $\sum_i \sqrt{h_i}$
- where $h_i^2 = a_i^2 + b_i^2$



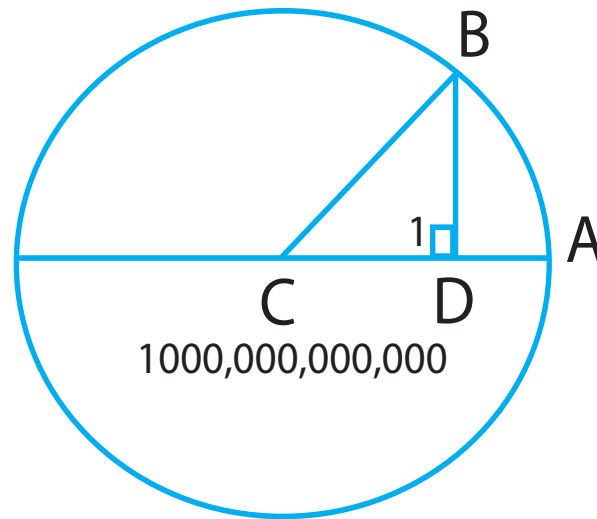
- is $\sqrt{1} + \sqrt{40} + \sqrt{60} > \sqrt{12} + \sqrt{17} + \sqrt{56}$?

$$\sqrt{1} + \sqrt{40} + \sqrt{60} = 15.07052201275$$

$$\sqrt{12} + \sqrt{17} + \sqrt{56} = 15.07052201430$$

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

- geometry problem



- $BD = 1$
- what is AD ?

$$AD = AC - CD = 500,000,000,000 - \sqrt{500,000,000,000^2 - 1}$$

Computing \sqrt{h} ; Heron's method

- Iterative approach; also called the Babylonian method
- can be seen to be equivalent to the Newton method (more later)
- $y_0 = h; x_0 = 1$
- $y_1 = (x_0 + y_0)/2; x_1 = h/(y_1)$
- *in general*
 - $y_{i+1} = (x_i + y_i)/2$
 - $x_{i+1} = h/(y_{i+1})$



(c. 10–70 AD)

Computing \sqrt{h} ; Heron's method

- $y_0 = h; x_0 = 1$
- $y_1 = (x_0 + y_0)/2; x_1 = h/(y_1)$
- *in general*
 - $y_{i+1} = (x_i + y_i)/2$
 - $x_{i+1} = h/(y_{i+1})$
- converges quickly

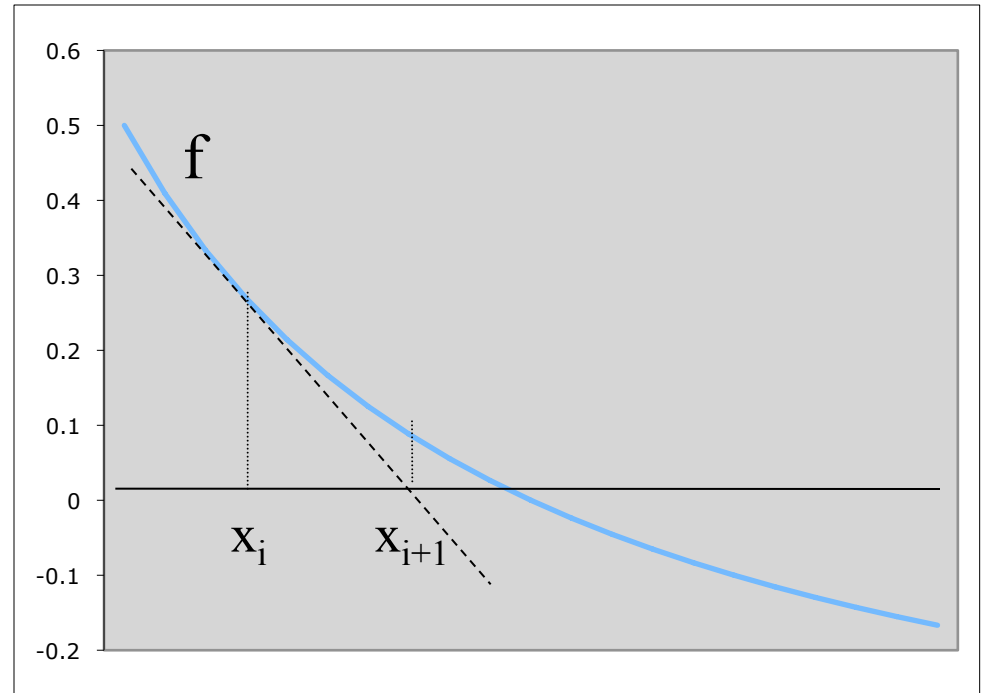
| | x | y |
|---|--------------------|--------------------|
| 0 | 1.0000000000000000 | 2.0000000000000000 |
| 1 | 1.3333333333333330 | 1.5000000000000000 |
| 2 | 1.411764705882350 | 1.4166666666666670 |
| 3 | 1.414211438474870 | 1.414215686274510 |
| 4 | 1.414213562371500 | 1.414213562374690 |
| 5 | 1.414213562373090 | 1.414213562373090 |



(c. 10–70 AD)

Computing \sqrt{h} ; 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 $f(x_i) + f'(x_i)(x - x_i) = 0$
 $\Rightarrow x_{i+1} = x_i - f(x_i)/f'(x_i)$



for computing \sqrt{h}

$$f(x) = x^2 - h$$

$$x_{i+1} = x_i - (x_i^2 - h) / 2x_i = (x_i + h/x_i) / 2$$

Computing \sqrt{h} ; Newton's method

- $x_{i+1} = x_i - (x_i^2 - h) / 2x_i = (x_i + h/x_i) / 2$
- convergence?

| | |
|---|--------------------|
| 0 | 1.0000000000000000 |
| 1 | 1.5000000000000000 |
| 2 | 1.4166666666666670 |
| 3 | 1.414215686274510 |
| 4 | 1.414213562374690 |
| 5 | 1.414213562373090 |

Large number addition

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

$$\begin{array}{ccccccc} X_{n-1} & X_{n-2} & \cdots & X_1 & X_0 & & \\ & & & & & + & \\ Y_{n-1} & Y_{n-2} & \cdots & Y_1 & Y_0 & & \end{array}$$

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 ?

$$\begin{array}{ccccccc} X_{n-1} & X_{n-2} & \cdots & X_1 & X_0 & & \\ & & & & & * & \\ Y_{n-1} & Y_{n-2} & \cdots & Y_1 & Y_0 & & \end{array}$$

Divide and conquer: attempt I

- Split each number into “high” and “low” parts, each $n/2$ digits long

$$- x = x_H r^{n/2} + x_L$$

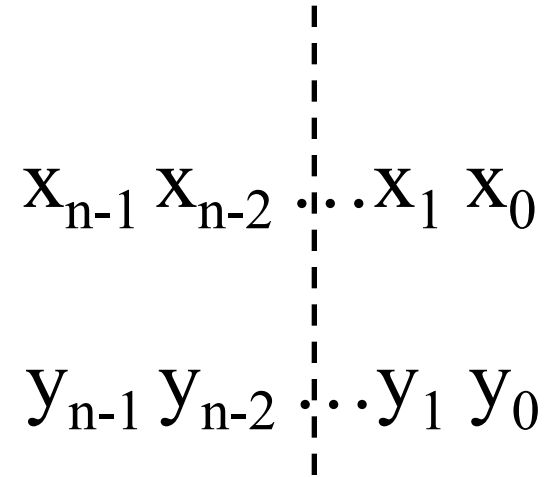
$$- y = y_H r^{n/2} + y_L$$

- We have

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

- Algorithm for $\text{mult}(x, y)$:

$a = \text{mult}(x_H, y_H)$, $b = \text{mult}(x_H, y_L)$, $c = \text{mult}(x_L, y_H)$, $d = \text{mult}(x_L, y_L)$
 return $\text{add}(\text{lshift}(a, n), \text{lshift}(b, n/2), \text{lshift}(c, n/2), d)$



Attempt I: analysis

- Algorithm:

$a = \text{mult}(x_H, y_H)$, $b = \text{mult}(x_H, y_L)$, $c = \text{mult}(x_L, y_H)$, $d = \text{mult}(x_L, y_L)$
return $\text{add}(\text{lshift}(a, n), \text{lshift}(b, n/2), \text{lshift}(c, n/2), d)$

- Running time $T(n)$?

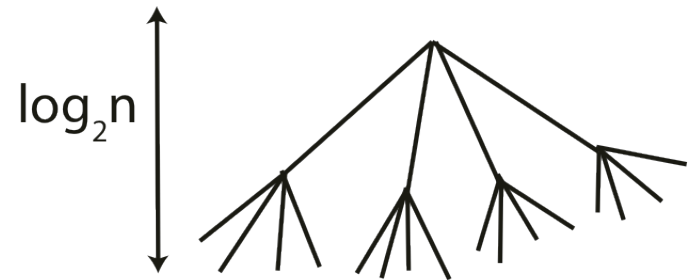
– recurrence

- $T(n) = 4T(n/2) + O(n)$

– this solves to

- $T(n) = O(n^2)$

- How to improve on this ?



$$4T(n/2)$$

$$4^{\log_2 n} = n^{\log_2 4} = n^2$$

Attempt II

- Need to 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)

$$a = x_H*y_H$$

$$d = x_L*y_L$$

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

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

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

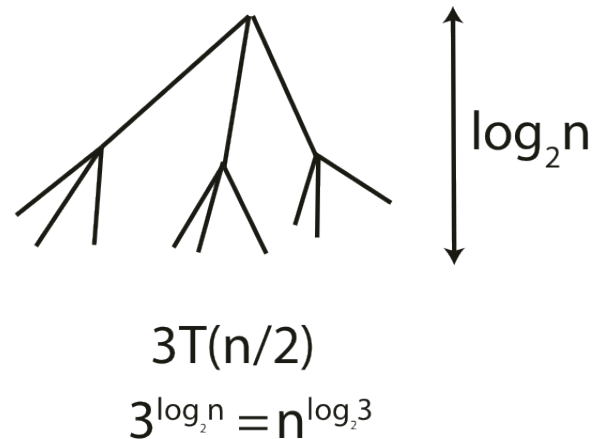
$$x*y = a r^n + e r^{n/2} + d$$

- This leads to

$$- T(n) = 3 T(n/2) + O(n)$$

which solves to

$$T(n) = O(n^{\log_2 3}) = O(n^{1.58496\dots})$$

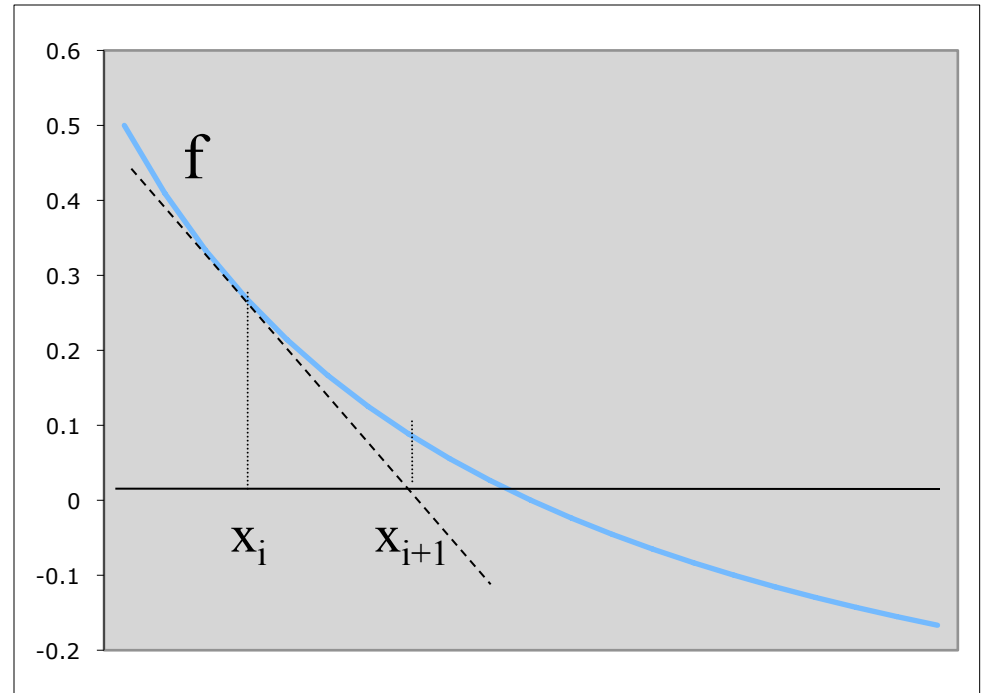


Division $1/b$

- Inversion:
 - given positive n -digit number b (radix r)
 - compute $1/b$
 - i.e., for some $R=r^k$, compute $\lfloor R/b \rfloor$
- Iterative algorithm:
 - start from some x_0
 - keep computing x_{i+1} from x_i
 - ~~hope~~ prove this converges to the answer

Division ... again ... 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 $f(x_i) + f'(x_i)(x - x_i) = 0$
$$\Rightarrow x_{i+1} = x_i - f(x_i)/f'(x_i)$$



here we have (next slide)

$$f(x) = 1/x - b/R$$

$$x_{i+1} = 2x_i - x_i^2 b/R$$

Division

- want to solve $f(x)=1/x-b/R=0$
- we have $f'(x)=-1/x^2$
- iterative step $x_{i+1}=x_i-f(x_i)/f'(x_i)$ yields

$$x_{i+1}=x_i+ x_i^2 (1/x_i -b/R)$$

i.e.,

$$x_{i+1}=2x_i - x_i^2 b/R$$

- convergence ?

Convergence of $x_{i+1} = 2x_i - x_i^2 b/R$

- Assume $x_i = R/b (1+e_i)$, $e_i = \text{error}$
- Assumptions:
 - $|e_i|$ is small
 - Ignore the round-off errors caused by the “/R” operation
- How does each iteration affect e_i ?

$$\begin{aligned}x_{i+1} &= 2x_i - x_i^2 b/R \\ &= 2R/b (1+e_i) - [R/b (1+e_i)]^2 b/R \\ &= R/b [2+2e_i - 1 - 2e_i - e_i^2] \\ &= R/b [1 - e_i^2] \\ &= R/b [1 + e_{i+1}], \text{ where } e_{i+1} = -e_i^2\end{aligned}$$

- We have $|e_{i+1}| = |e_i|^2$ - “quadratic” convergence