CS 170

Lecture 1

Sanjam Garg.

Logistics.

* Website: cs170.org
* Send OSP LoAs asap
* Lecture attendance is required (4 drops allowed)
  - use iClicker (test today)
  - join.iclicker.com/RGWR
* No discussion this week
* HW1 released on Sunday

Algorithms – Integer arithmetic (Efficiency)

MCDXLIII

DCCCXII

Decimal system: ① once you can write 0...9 you can write integers as large as possible.

② $a+b$ and $a \times b$ for $a \& b$ be single digit you can + & × any two integers

Fibonnacci Sequence: 0, 1, 1, 2, 3, 5, 8 .....

$$F_n = \begin{cases} 0 & \text{if } n=0 \\ 1 & \text{if } n=1 \\ F_{n-1}+F_{n-2} & \text{otherwise} \end{cases}$$

$n>4 \quad 2F_{n-2} < F_n < 2F_{n-1}$

$2^{\frac{n}{2}} \quad 2^n \quad 2^{0.694n}$

$F_n = F_{n-1} + F_{n-2}$ & $F_n > F_{n-1}$

Problem: Compute F_n (e.g., $n = 100$ or 200)

```
Fib(n):
    if n = 0    return 0
    if n = 1    return 1
    return      Fib(n-1) + Fib(n-2)
```

→ 1, c

$c_1 > c_2 > c_3$

$T(n)$ = # of computer steps it takes to compute $Fib(n)$

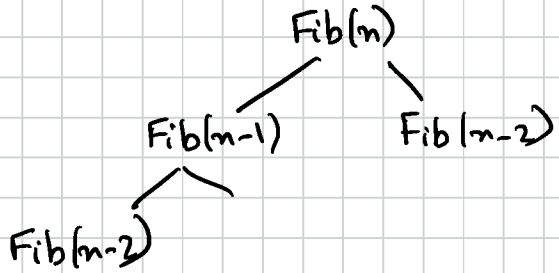
$T(n) \to c_1$ if $n = 0$; c_2 if $n = 1$; $c_3 + T(n-1) + T(n-2)$ ow.

$T(n) > F_n$

between $2^{\frac{n}{2}}$ & 2^n

$O(2^n)$

* is this correct? ✓
* how long does it take? → basic computer steps. → bigger picture.
* can we do better?




```
Fib(n)
    if n = 0          return 0
    create an array   F[0 ... n]
    F(0) = 0          F(1) = 1
    for i = 2 ... n
        F[i] = F[i-1] + F[i-2]
    return F[n]
```

constant work + $\underbrace{c + c \cdots c}_{n-2}$

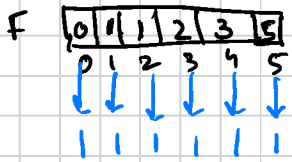
$\approx cn$ time

* $c' \times 1 + c' \times 2 + c' \times 3 \cdots + c'n$

$\approx c'' n^2$

$O(n^2)$

Fib(5)



Efficiency of the algorithm → rough approximation that gives the right perspective.

↳ running time as the input gets larger.

* don't care about constants
* among multiple terms - n, n^2, n^3 - n & n^2 don't matter.

Big-Oh-Notation

Let $f(n)$ & $g(n)$ be : $\mathbb{Z}^+ \longrightarrow \mathbb{R}$

($\mathbb{Z}^+$: size of the input; $\mathbb{R}$: # of computer steps)

Define: $f = O(g)$ iff $f \leq g$ (ignoring constants)

if $\exists$ a constant c s.t $\forall n \in \mathbb{Z}^+$ we have $f(n) \leq c\,g(n)$

$a n^3 + b n^2 < (a+b) n^3$ → $O(n^3)$

Integer Addition & Multiplication.

n ← #of digits

$$\begin{array}{cccc} & & 1 & \\ & & 5 & 3 \\ + & & 3 & 9 \\ \hline & & 9 & 2 \end{array}$$

↳ $O(n)$

↳ working in base 10

↳ base 2

$$\begin{array}{cccc} & 1 & 4 & 3 \to a \\ & b_2 & b_1 & b_0 \\ \times & 3 & 4 & 1 \to b \\ \hline & 1 & 4 & 3 \end{array}$$

$$\begin{array}{ccccc} & 5 & 7 & 2 & x \\ + & 4 & 2 & 9 & x & x \\ \hline & 4 & 8 & 7 & 6 & 3 \end{array}$$

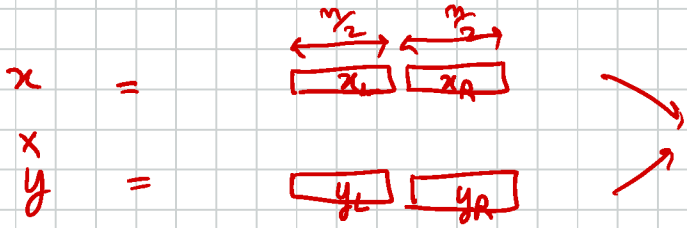
$a \times b = a \times (b_0 + b_1 \times 10 + b_2 100 \ldots)$

$= a \times b_0 + a \times b_1 \times 10 + a \times b_2 \times 10^2 \ldots$

→ $O(n^2)$

Multiplication using Divide and Conquer Approach

↳ ① break the problem into smaller sub-problems
② Solve the subproblems
③ Combine the answers.



n digit numbers → $n = 2^k$

$x = x_L 2^{\frac{n}{2}} + x_R$

$y = y_L 2^{\frac{n}{2}} + y_R$

$$x \times y = x_L y_L 2^n + (x_L y_R + x_R y_L) 2^{\frac{n}{2}} + x_R y_R$$

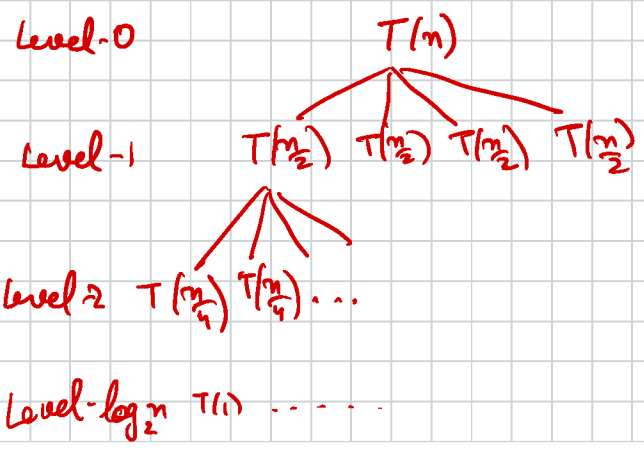
MULTIPLY (x, y)

Input: $x, y \rightarrow$ n-bit positive integers
Output: $x \times y$

```
MUL(x,y)
  if n=1 return x×y
  x_L, x_R = left ⌈n/2⌉, right ⌊n/2⌋ bits of x
  y_L, y_R = left ⌈n/2⌉, right ⌊n/2⌋ bits of y
  P1, P2, P3, P4 = MUL(x_L, y_L), MUL(x_L, y_R), MUL(x_R, y_L), MUL(x_R, y_R)
  return P1 2^n + (P2+P3) 2^(n/2) + P4
```

$T(n)$ = # of steps to multiply two n-digit numbers

$= 4\,T(\frac{n}{2}) + O(n)$



# of subproblems	Total work
1	$1 \times O(n)$
4	$+ 4 \times O(\frac{n}{2})$
4^2	$+ 4^2 \times O(\frac{n}{2^2})$
$4^{\log_2 n}$	$+ 4^{\log_2 n} O(\frac{n}{2^{\log_2 n}})$

Facts ① For a geometric progression $1, \alpha, \alpha^2 \cdots \alpha^k$ with $\alpha > 1$
we have that $1 + \alpha + \alpha^2 \cdots \alpha^k = O(\text{the last term})$
$= O(\alpha^k)$

② (a) $2^{\log_2 n} = n$ (b) $4^{\log_2 n} = \left(2^{\log_2 n}\right)^2 = n^2$

③ $a^{\log b} = b^{\log_b a \times \log b} = b^{\log a}$

$1 \times O(n) + 4\, O\left(\frac{n}{2}\right) \cdots\cdots$ $4^{\log_2 n}\, O\left(\frac{n}{2^{\log n}}\right)$

$= O\left(4^{\log_2 n} \times \frac{n}{2^{\log n}}\right)$ (by Fact 1)

$= O(n^2)$ (by fact 2a, b)

Can we improve it?

Observe that $\underbrace{(x_L y_R + x_R y_L)}_{\text{need to compute}} = (x_L + x_R)(y_L + y_R) - \underbrace{x_L y_L - x_R y_R}_{\text{already computed}}$

Insight
* one multiplication rather than two suffices.

* $T(n) = 3\, T\left(\frac{n}{2}\right) + O(n)$

* $T(n) = O(n) + 3\, O\left(\frac{n}{2}\right) + 3^2\, O\left(\frac{n}{2^2}\right) \cdots 3^{\log n}\, O\left(\frac{n}{2^{\log n}}\right)$

$= O\left(3^{\log n} \times \frac{n}{2^{\log n}}\right)$ (by Fact 1)

$= O\left(n^{\log 3}\right)$ (by Fact 2a, 3)

$= O(n^{1.59})$