Divide an Conquer and Running Time Analysis

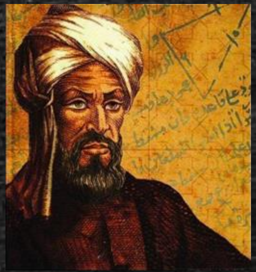
Integer Multiplication

Today’s goal

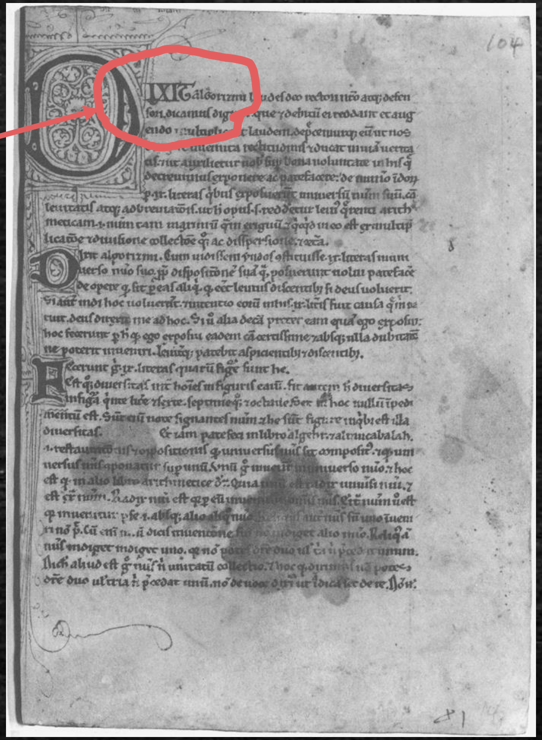
- Karatsuba Integer Multiplication
- Algorithmic Technique
  - Divide and conquer
- Algorithmic Analysis tool
  - Intro to asymptotic analysis

Start at very beginning.

- al-Khwarizmi



- Dixit algorizmi
- “Algorisme” [old French]
  - Arabic number system
  - “Algorithm”



Integer Multiplication

- How to calculate 44×34

$$\begin{array}{r} 44 \\ \times\ \ 34 \\ \hline \end{array}$$

- How to calculate $123555589 \times 987555321$

$$\begin{array}{r} 123555589 \\ \times\ \ 987555321 \\ \hline \end{array}$$

How fast is it?

$$\begin{array}{r} \overbrace{1235555891244352345234653 24}^{\boldsymbol{n}} \\ \times\ 8755532112312312312312 33123 \\ \hline \end{array}$$

How many 1-digit operation we need to make?

Roughly

- n^2 1-multiplication
- n^2 1-addition for carries
- n $2n$-addition finally

How fast is it?

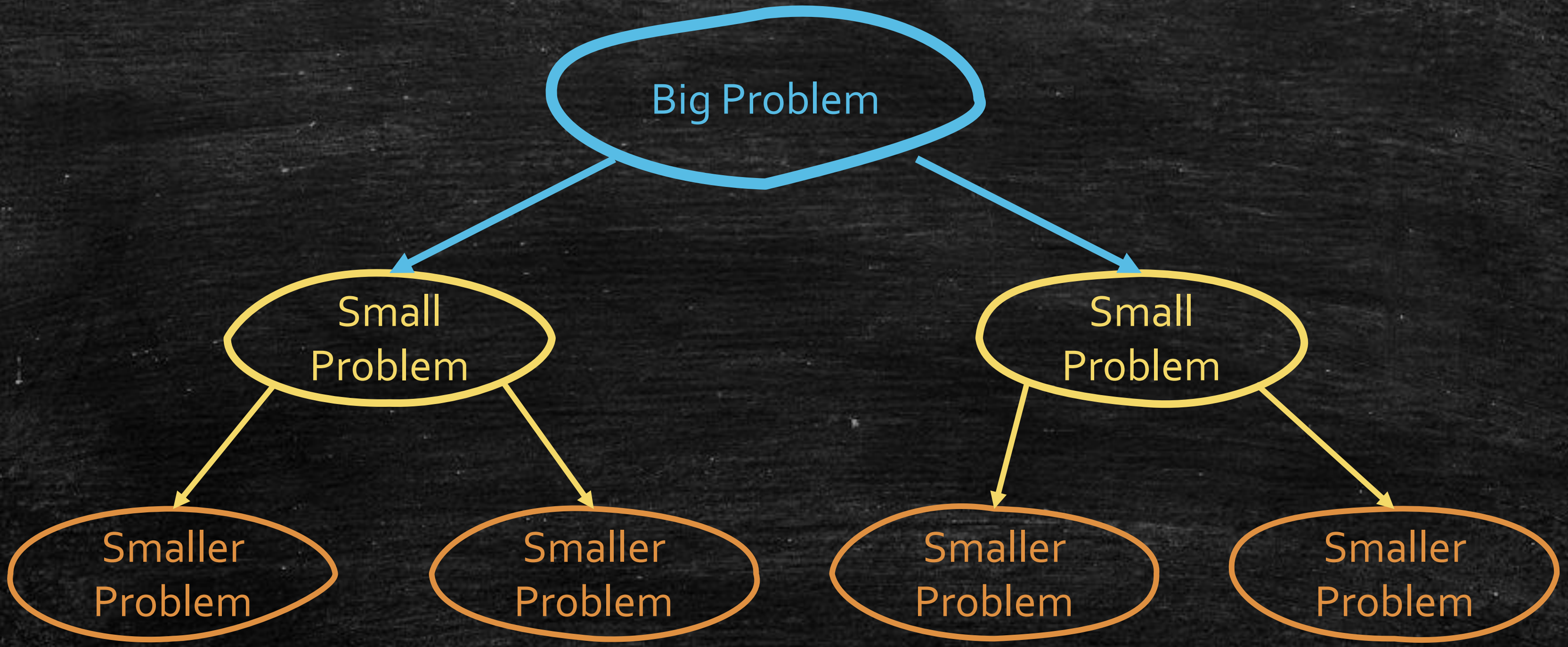
- How many 1-digit operation we need to make?
- We roughly need $5\mathrm{n}^2$ 1-digit operations.
- We know we can write $5n^2, 4n^2, 100000n^2$ to be
  - $\boldsymbol{O}(\boldsymbol{n}^{\mathbf{2}})$
- Plan B:
  - What if we roughly need $100n^{1.6}$ 1-digit operations?
  - Which is better?
  - $100n^{1.6}$ or $5n^2$?
  - We think $100n^{1.6}$ is better than $5n^2$ because we cares **large** n!
  - $O(n^{1.6})$ is better than $O(n^2)$!
  - Formal Definition of O later…

Can we do better?

Let us buy our first tool! 

Divide and conquer

Divide and Conquer



Divide and conquer for multiplication

- 1234×5678
- $1234 = 12 \times 100 + 34$
- $1234 \times 5678 = (12 \times 100 + 34)(56 \times 100 + 78)$ $= (12 \times 56) \cdot 10000 + (12 \times 78 + 34 \times 56) \cdot 100 + 34 \times 78$
- 1 four-digit → 4 two-digit

Generally?

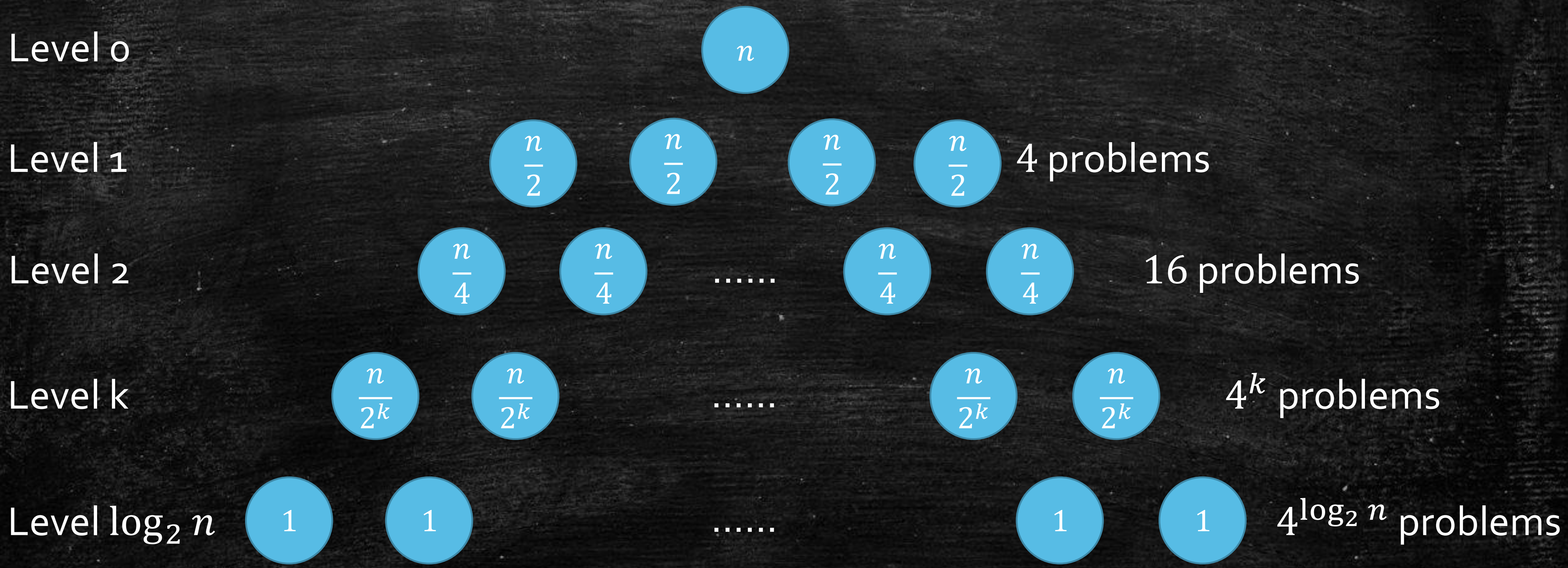
- Can we make it generally?
- To do n digit multiplications, suppose n is even
- Design a recursive algorithm for n, suppose n is 2's power.
- $$xy = \left(a \cdot 10^{\frac{n}{2}} + b\right)\left(c \cdot 10^{\frac{n}{2}} + d\right)$$
  $$= ac \cdot 10^n + (ad + bc) \cdot 10^{\frac{n}{2}} + bd$$

What if n is **not** $2's$ power?

Running time, analytically

- Main question: **Is it better than before?**
  - ~~**Yes! Because we learn it in SJTU!**~~
  - how many 1-digit multiplications we need for an n-digit multiplication?
    - A: n^2 B: n^3 C: n D: $nlogn$
      - Run the algorithm for 1234×5678, how many 1-digit multiplications we need?
      - how many 1-digit multiplications we need for 1 8-digit multiplication?

Analysis


Level 0
n
Level 1
$\frac{n}{2}$ $\frac{n}{2}$ $\frac{n}{2}$ $\frac{n}{2}$
4 problems
Level 2
$\frac{n}{4}$ $\frac{n}{4}$ ...... $\frac{n}{4}$ $\frac{n}{4}$
16 problems
Level k
$\frac{n}{2^k}$ $\frac{n}{2^k}$ ...... $\frac{n}{2^k}$ $\frac{n}{2^k}$
4^k problems
Level $\log_2 n$
1 1 ...... 1 1
$4^{\log_2 n}$ problems

Analysis

- Claim: we need n^2 1-digit multiplications for 1 n-digit multiplication.
- How many levels we need?
  - $\log_2 n$
- How many multiplications we need in level $t = \log_2 n$?
  - Level 0: 1 $\quad n \times n$
  - Level 1: 4 $\quad \frac{n}{2} \times \frac{n}{2}$
  - Level 2: 16 $\frac{n}{4} \times \frac{n}{4}$
  - Level t: 4^t $\quad 1 \times 1$
- Conclusion: $4^{\log_2 n} = \boldsymbol{n^2}$

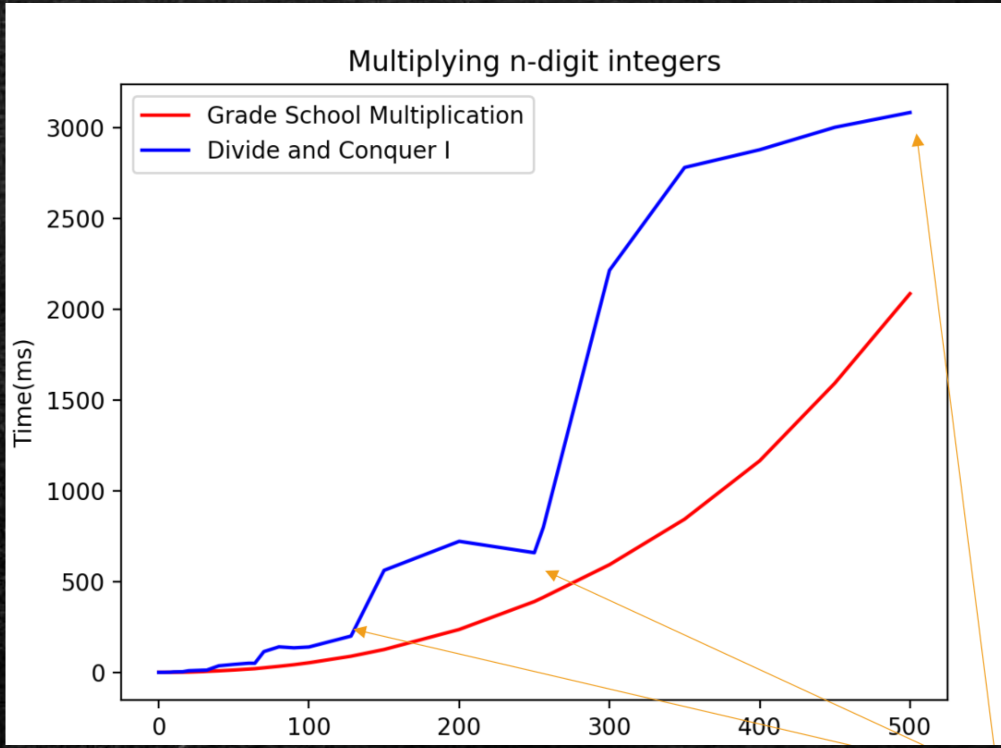
We need more than n^2 one-digit operations!

Even only consider multiplication!

It is just an analysis!

Experiments

- Claim: the grade school multiplication is better!



What’s wrong?

- $xy = \left(a \cdot 10^{\frac{n}{2}} + b\right) \times \left(c \cdot 10^{\frac{n}{2}} + d\right)$

  $= ac \cdot 10^{n} + (ad + bc) \cdot 10^{\frac{n}{2}} + bd$
- What do we need?
  - ac
  - $\boldsymbol{ad + bc}$
  - bd
- What do we calculate
  - ac
  - $\boldsymbol{ad}$
  - **bc**
  - bd

Karatsuba Algorithm

Improve!

- What do we need?
  - ac
  - $\boldsymbol{ad + bc}$
  - bd
- How to get $\boldsymbol{ad + bc}$ without $\boldsymbol{ad}$ and $\boldsymbol{bc}$?
- Solution:
  - Calculate: ac, bd
  - One more multiplication: $\mathrm{z} = (a+b)(c+d)$
  - Get $ad + bc = (a+b)(c+d) - ac - bd$
  - $x \times y = \left(a \cdot 10^{\frac{n}{2}} + b\right) \times \left(c \cdot 10^{\frac{n}{2}} + d\right)$

$$= ac \cdot 10^n + (ad + bc) \cdot 10^{\frac{n}{2}} + bd$$

$$= ac \cdot 10^n + (z - ac - bd) \cdot 10^{\frac{n}{2}} + bd$$

Improve!

- What is the difference?
  - We now calculate
    - ac
    - $\boldsymbol{z = (a + b)(c + d)}$
    - bd
  - One n-digit → Three $\frac{n}{2}$-digit

Make a guess!

How fast is it?

Is it fast?

- Claim: we need $n^{1.6}$ 1-digit multiplication for 1 n-digit multiplication.
- How many levels we need?
  - $\log_2 n$
- How many multiplications we need in level t?
  - Level 0: 1 $n \times n$
  - Level 1: 3 $\frac{n}{2} \times \frac{n}{2}$
  - Level 2: 9 $\frac{n}{4} \times \frac{n}{4}$
  - Level t: 3^t 1×1
- Conclusion: $3^{\log_2 n} = n^{\log_2 3} \approx n^{1.6}$

How to consider additions?

Is it fast?

- Claim: we need $n^{1.6}$ 1-digit multiplication for 1 n-digit multiplication.
- How many levels we need?
  - $\log_2 n$
- How many multiplications we need in level t?
  - Level 0: 1 $n \times n$
  - Level 1: 3 $\frac{n}{2} \times \frac{n}{2}$
  - Level 2: 9 $\frac{n}{4} \times \frac{n}{4}$
  - Level t: 3^t $\frac{n}{2^t} \times \frac{n}{2^t}$
  - Level $\log_2 n$: $3^{\log_2 n}$ 1×1

$8n$ for addition.

$8 \cdot 3 \cdot \frac{n}{2}$ for addition.

$8 \cdot 3^t \cdot \frac{n}{2^t}$ for addition.

Totally $O(n^{1.6})$

- Conclusion: $3^{\log_2 n} = n^{\log_2 3} \approx n^{1.6}$

Can we do better **again**?

Better algorithms

- Toom-Cook (1963): Breaking into size $\frac{n}{3}$-size problems make it better! → $O(n^{1.465})$
- Think:
  - how to break $n \times n$ into $5\ \frac{n}{3} \times \frac{n}{3}$?
  - Given it is true, why it is $n^{1.465}$?
- Schonhage-Strassen (1971): $O(n \log n \log \log n)$
- Furer (2007): $O(n \log n \log^* n)$
- Harvey and van der Hoeven (2019): $O(n \log n)$

$$\log^* n := \begin{cases} 0 & \text{if } n \le 1; \\ 1 + \log^*(\log n) & \text{if } n > 1 \end{cases}$$

FFT! We will make it next week! (maybe next next)

Our work is expected to be the end of the road for this problem, although we don't know yet how to prove this rigorously.

What about matrix?

- How to multiply two matrices
- $\begin{bmatrix} 2 & 9 \\ 7 & 5 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 2 \times 1 + 9 \times 3 & 2 \times 2 + 9 \times 4 \\ 7 \times 1 + 5 \times 3 & 7 \times 2 + 5 \times 4 \end{bmatrix} = \begin{bmatrix} 29 & 40 \\ 22 & 34 \end{bmatrix}$
- $Z = XY$
- $z_{ik} = \sum_{1 \le j \le n} x_{ij} y_{jk}$
- How many integer multiplications?
  - n^2 entries of Z to calculate
  - Each takes n multiplications
  - Totally n^3

How to divide and conquer?

Divide and conquer

- Key fact: If $X = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$, $Y = \begin{bmatrix} E & F \\ G & H \end{bmatrix}$.
  - $\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} E & \mathrm{F} \\ G & H \end{bmatrix} = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & \mathrm{C}F + DH \end{bmatrix}$
- How to divide and conquer?
  - 1 n-size multiplication → 8 $\frac{n}{2}$-size multiplications
    - $AE, BG, AF, BH, CE, DG, BF, DH$
  - How many integer multiplications?
  - $8^{\log_2 n} = n^3$
  - **The same problem as before!**

Do you have any approach?

Strassen's magical idea

- $P_1 = A(F - H)$
- $P_2 = (A + B)H$
- $P_3 = (C + D)E$
- $P_4 = D(G - E)$
- $P_5 = (A + D)(E + H)$
- $P_6 = (B - D)(G + H)$
- $P_7 = (A - C)(E + F)$

- $XY = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} E & \mathrm{F} \\ G & H \end{bmatrix}$

  $= \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$

  $= \begin{bmatrix} P_5 + P_4 - P_2 + P_6 & P_1 + P_2 \\ P_3 + P_4 & P_1 + P_5 - P_3 - P_7 \end{bmatrix}$
- How many integer multiplications now?
- $7^{\log_2 n} = n^{\log_2 7} \approx n^{2.81}$

Running Time: Make It Simple

Time Complexity and Big-O notations

Computation Model

- When we talk about "how fast", what do we mean?
- Integer multiplication
  - Number of 1-digit multiplication
- Matrix multiplication
  - Number of Integer multiplication
- General Algorithms?
  - What are the **unit-time operations** of our computers?

Turing Machine?

In Our Class (Not So Formal)

- Word RAM Model
  - RAM: Random Access
- The model was created by Michael Fredman and Dan Willard in 1990 to simulate programming languages like C.
- Unit-time operations
  - $a[1] \leftarrow a[5]$
  - $a[2] \leftarrow a[3] + a[5]$
  - $a[5] \leftarrow a[4] \times a[6]$
  - ……

RAM

	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$	$i = 6$	…	$i = 2^w$
$a[i]$	$\leq 2^w$	$\leq 2^w$	$\leq 2^w$	$\leq 2^w$	$\leq 2^w$	$\leq 2^w$	…	$\leq 2^w$

Most Problems in The Course

- Input: n different integers/words.
  - Sorting
    - Sort integers $a_1, a_2, \dots, a_n$.
  - Find Max
    - Find the max integer among $a_1, a_2, \dots, a_n$
- We can not use a fix w to handle arbitrarily large n?
- Transdichotomous
  - $n \le 2^w$ (We need a pointer to store the address.)
  - $a_i \le 2^w$ (We want to store each integer in a word.)
  - We do not want large w help improving the running time.
  - So, we assume $w = O(\max\{\log n, \log a_i\})$ to match the minimum request!

Turing Machine (informal)

- Two main difference
- $w = O(1)$
- Unit Operations
  - Move one step.
  - Only logical operations.

	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$	$i = 6$	…	$i = 2^w$
$a[i]$	$\{0,1\}$	$\{0,1\}$	$\{0,1\}$	$\{0,1\}$	$\{0,1\}$	$\{0,1\}$	…	$\{0,1\}$

Sample Program

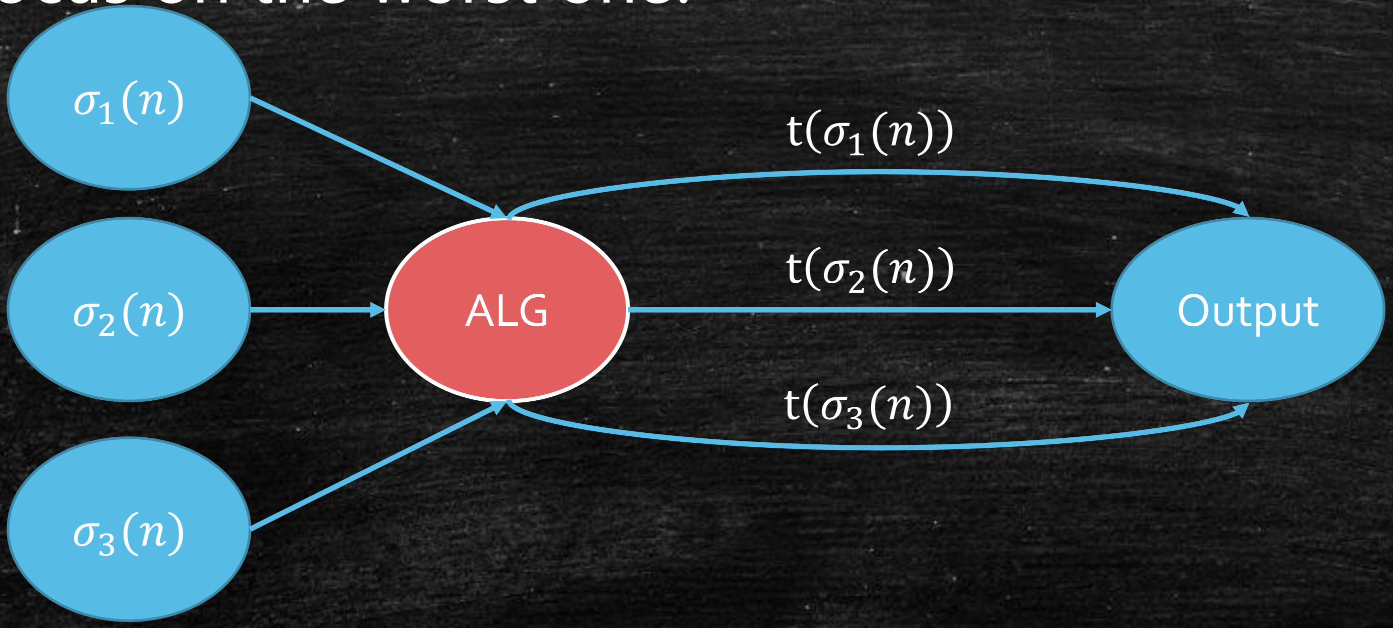
int sum=0,n;	$+3$
input(n);	$+1$
int a[n];	$+n$
input(a);	$+n$
for i=0 to n	$+n$
sum+=a[i];	$+n$
output(sum);	$+1$

Time Complexity

- What is the running time of an algorithm?
- Given an algorithm + Input $\sigma(n)$ of size n.
- Running Time $\mathrm{t}(\sigma(n))$: Number of unit-time operations.

Time Complexity

- $T(n) = \max_{\sigma(n)} t(\sigma(n))$
- Focus on the worst one!



Still Complicated!

Do you wan to see $T(n) = 103n^2 + 5n + 101$

Big O Notation

- $T(n) = 103n^2 + 5n + 101 \to O(n^3)$
- What is it exactly mean?
- $T(n) = O(n^2)$?
- $T(n)$ is at most at the same level as n^2 when n is large.
- $T(n) = O(n^2)$ may equals to
  - $n^2 + 100n + 10000$
  - $1000n^2 + 10n$
  - $n+100$
- It is to say, $T(n) \leq Cn^2$, when n is large.

Big O Notation: how to define

- Big O Notation (Upper Bound)
  - $T(n) = O(g(n))$
  - $\exists C, n_0$, s.t. $\forall n > n_0$, $T(n) \leq C \cdot g(n)$
- Big Ω Notation (Lower Bound)
  - $T(n) = \Omega(g(n))$
  - $\exists C, n_0$, s.t. $\forall n > n_0$, $T(n) \geq C \cdot g(n)$
- Big Θ Notation (Exact!)
  - $T(n) = O(g(n))$
  - $T(n) = \Omega(g(n))$

Discussion

- Show n^2 is not $O(n)$.
- Let $p(n) = a_k n^k + a_{k-1} n^{k-1} + a_{k-2} n^{k-2} \ldots + a_1 n + a_0$
  - $a_k \geq 0$,
- Show $p(n) = O(n^k)$?
- Show $p(n) = \Theta(n^k)$?
- Show
  - $3^n \neq O(2^n)$?
  - $\log_2 n = \Omega(\ln n)$?
- Can we find two functions f and g,
  - $f(n) \neq O(g(n))$ and $f(n) \neq \Omega(g(n))$?
  - Require non-decreasing?

Little o Notations

- Little O Notation (Upper Bound)
  - $T(n) = o(g(n))$
  - $\forall C > 0, \exists n_0, \forall n > n_0, T(n) < C \cdot g(n)$
- Little ω Notation (Lower Bound)
  - $T(n) = \omega\left(g(n)\right)$
  - $\forall C > 0, \exists n_0,\ \forall n > n_0, T(n) > C \cdot g(n)$

Karatsuba

Karatsuba$(n, x[n], y[n])$:

1. if $n = 1$ return $x \times y$
2. $a \leftarrow x[1{:}\frac{n}{2}], b \leftarrow x[\frac{n}{2} + 1{:}n]$
3. $c \leftarrow y\left[1{:}\frac{n}{2}\right], b \leftarrow y[\frac{n}{2} + 1{:}n]$
4. $ac \leftarrow karatsuba(\frac{n}{2}, a, c)$
5. $bd \leftarrow karatsuba(\frac{n}{2}, b, d)$
6. $z \leftarrow karatsuba(\frac{n}{2}, (a - b), (d - c))$
7. $xy \leftarrow ac \cdot 10^n + (z + ac + bd) \cdot 10^{\frac{n}{2}} + bd$

9. return xy

Karatsuba with Word-RAM and Big O

Karatsuba$(n, x[n], y[n])$:	$\boldsymbol{T(n)}$
1. if $n = 1$ return $x \times y$	$O(1)$
2. $a \leftarrow x[1{:}\frac{n}{2}], b \leftarrow x[\frac{n}{2} + 1{:}n]$	$O(\frac{n}{2})$
3. $c \leftarrow y\left[1{:}\frac{n}{2}\right], b \leftarrow y[\frac{n}{2} + 1{:}n]$	$O(\frac{n}{2})$
4. $ac \leftarrow karatsuba(\frac{n}{2}, a, c)$	$T(\frac{n}{2})$
5. $bd \leftarrow karatsuba(\frac{n}{2}, b, d)$	$T(\frac{n}{2})$
6. $z \leftarrow karatsuba(\frac{n}{2}, (a-b), (d-c))$	$T(\frac{n}{2})$
7. $xy \leftarrow ac \cdot 10^n + (z + ac + bd) \cdot 10^{\frac{n}{2}} + bd$	$O(n)$
9. return xy	$O(n)$

Karatsuba with Word-RAM and Big O

- $T(n) = 3T\left(\frac{n}{2}\right) + O(n)$
- Level 1: $+O(n)$
- Level 2: $+O(1.5n)$
- Level 3: $+O\left(1.5^2 n\right)$
- Level t: $+O(1.5^t n)$
- Level $\log_2 n - 1$: $+O(1.5^{\log_2 n - 1} n)$
- Level $\log_2 n$: $+O(n^{\log_2 3})$

$O(n^{\log_2 3})$

$O(n^{\log_2 3})$

Goals!

- Course goals
  - Think **analytically** about algorithms
  - Clearly **communicate** your algorithmic idea
  - Equip with an **algorithmic toolkit**
- Today's goals
  - Karatsuba Integer Multiplication
  - Algorithmic Technique
    - Divide and conquer
  - Algorithmic Analysis tool
    - Intro to asymptotic analysis
    - Word RAM and Big O Notations

How about the pace today?

Next time

- More divide and conquer

- Before next time
  - Think the questions in the slides.
  - Join the wechat group!
  - Try the Online Judge System!

Welcome to discuss research problems with us!