

Algorithm Design and Analysis

Introduction & multiplication

How was your break!?

Today: Hunshichang Algorithm

The Big Questions

- Who are we?
- Why are we here?
- What is going on?

Who are we?

We are ...

- Instructors

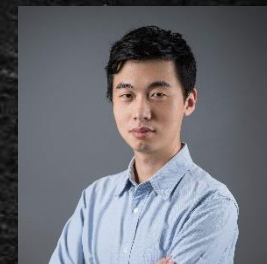
- 张宇昊 (Yuhao Zhang)
 - Zhang_yuhao@sjtu.edu.cn
 - www.zyhwtc.com
- 陶表帅 (Biaoshuai Tao)
 - bstao@sjtu.edu.cn
 - jhc.sjtu.edu.cn/~bstao
- 张驰豪 (Chihao Zhang)
 - chihao@sjtu.edu.cn
 - Chihaozhang.com/



张宇昊



陶表帅



张驰豪

- TAs

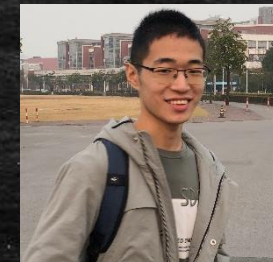
- Writing
 - 杨若峰 wanshuiyin@sjtu.edu.cn
 - 王文茜 wangwenqian@sjtu.edu.cn
 - 宋家鑫 sjtu_xiaosong@sjtu.edu.cn
- Programming
 - 王玉林 yulin_wang@sjtu.edu.cn



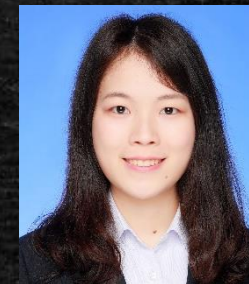
杨若峰



王文茜



宋家鑫



王玉林



步晓霖



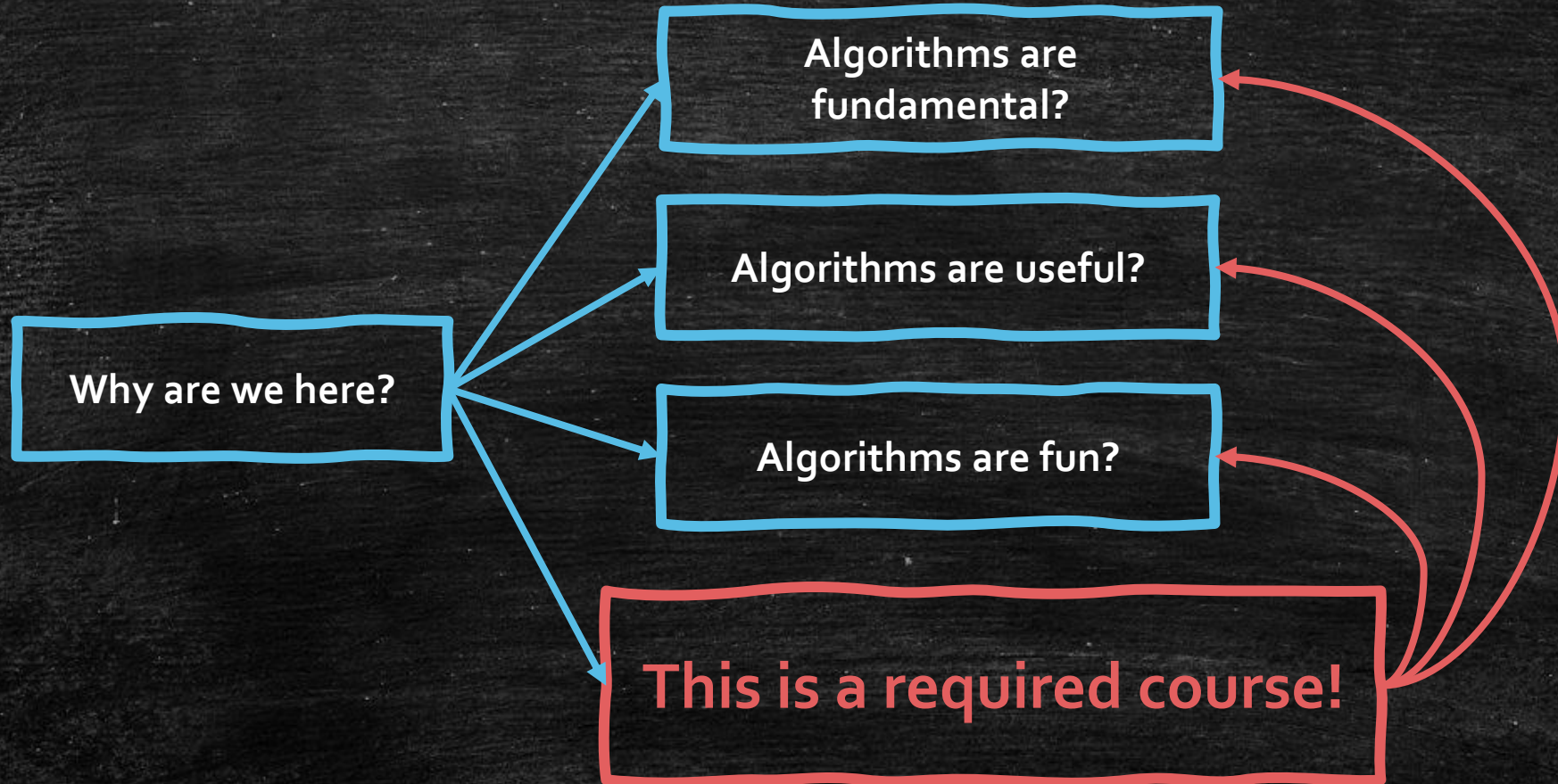
薛峥嵘



朱梓臣

Why are we here?

Algorithm!



Algorithms are fundamental!



Operating Systems

How to choose
data in the
cache?



Cryptography

How to design
an encoding
algorithm?



Algorithms are useful!

- We may need to sort something?
- We may need to find the best bundle?
- ...
- When the input is larger and larger,
- Algorithms are more and more important!

Algorithms are fun!

- Algorithm design is also an **art**!
- You will feel **excited** when you see a surprising algorithm!
- You will feel **thrilled** when you have created a surprising algorithm!
- Also, many interesting **research problems**...

What is going on?

Course goals

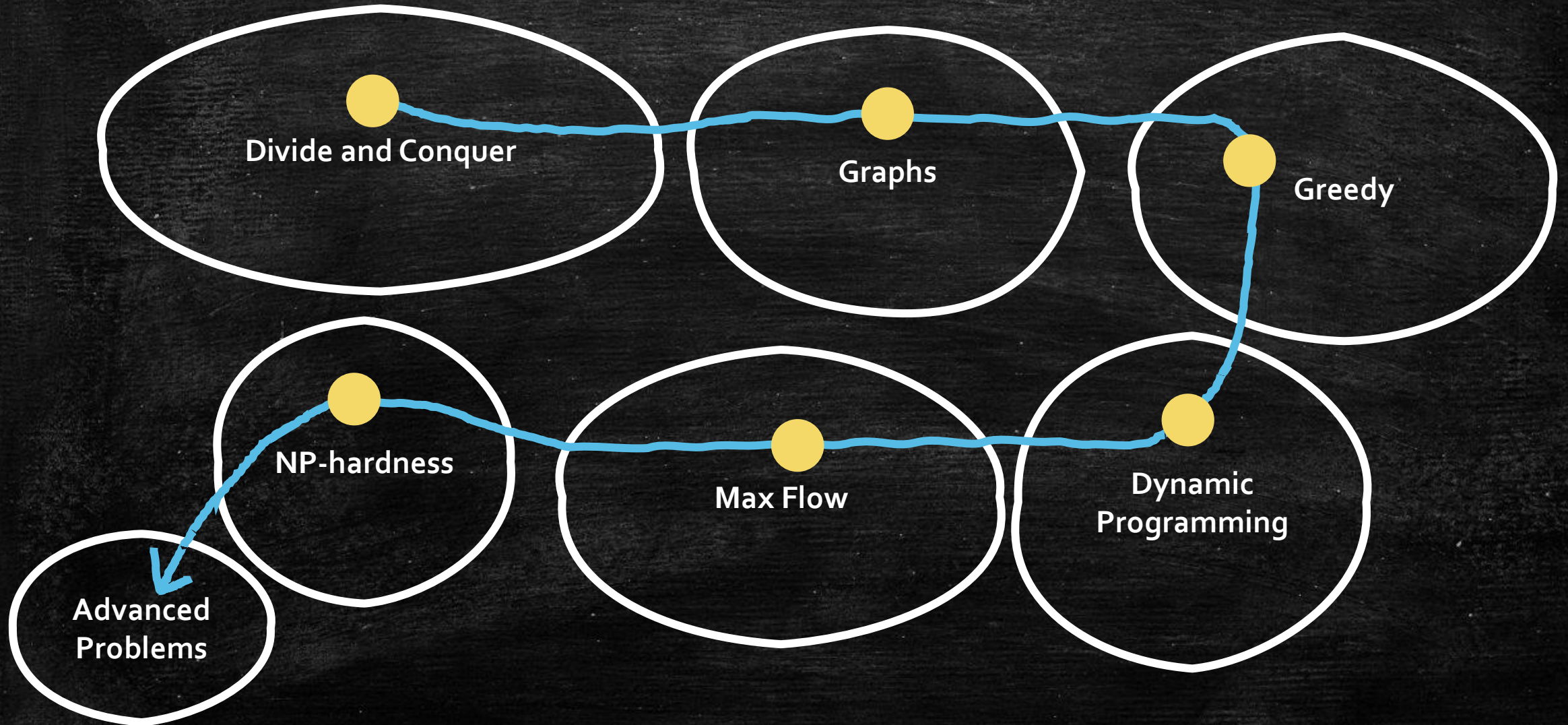
- The **design** and **analysis** of algorithms
- After this course, you will
 - Think **analytically** about algorithms
 - Clearly **communicate** your algorithmic idea
 - Equip with an **algorithmic toolkit**



- Use them **correctly**




Roadmap




Guide questions

- Does the algorithm **work**?
- Is it **fast**?
- Can I do **better**?

How to think?

- 
- What is work?
 - What is better?
 - Do we need to consider worst case?
 - Is there any corner case?

- Detail-oriented
- Precise
- Rigorous



Listen to my idea, it is quite intuitive! It should work if everything goes well, trust me!

- Big-picture
- Intuitive
- Hand-wavey

Both side are necessary!

How to think in most of this course?

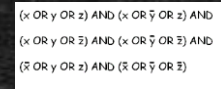
- We usually talk about **Exact Algorithms**.
- Dose the algorithm **work**?
 - Return the optimal/correct answer
- Is it **fast**?
 - Time complexity
 - Worst case
- Can I do **better**?
 - More efficient
 - Better time complexity

Aside the course: Talk about Exactness.

- What if the problem is so hard to get the solution?

- Np-hard problems: take too long time
- Online problems: not enough information

Included in
advanced topics



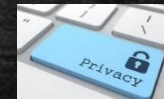
SAT Problem



Online Matching

- What if a more efficient algorithm is not better?

- More efficient → make private data public
- More efficient → focus on the majority population



Data Privacy



Fairness

- What if you can not control player's behavior?

- Auction
- Public resource allocation



Auction



Public Resources

Advanced Topic

- Approximation Algorithms
 - Sometimes, we can not have both **efficiency** and **exactness**, unless **P=NP**.
 - Design Approximation Algorithms in Polynomial Time.
 - How to evaluate?
 - Algorithm A achieve Approximation Ratio Γ .
 - Minimizing Problem
 - For **all** inputs σ , $A(\sigma) \leq \Gamma \cdot OPT(\sigma)$.
 - A is a Γ –approximate algorithm.
 - Exact Algorithm: $\Gamma = 1$.

Advanced Topic

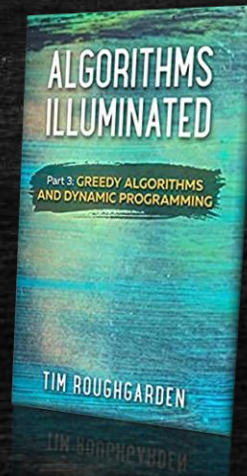
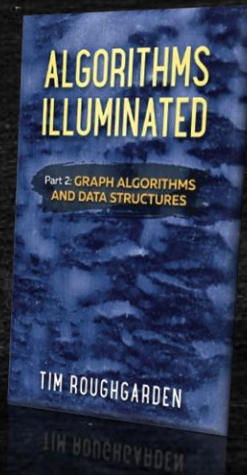
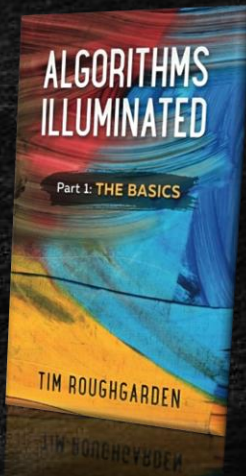
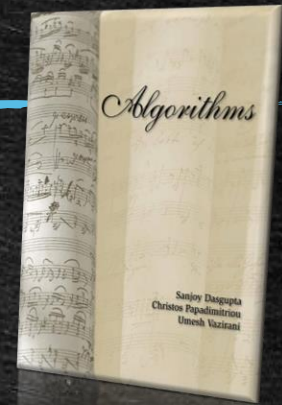
- Online Algorithm

- Sometimes, we can not have **exactness**, if we are making **online decision**, even we have super computational power.
- Example: Ski-rental
 - Rent: \$1
 - Buy: \$10
 - Buy or rent?
- How to evaluate?
- Algorithm A achieve Competitive Ratio Γ .
 - For **all** input sequences σ , $A(\sigma) \leq \Gamma \cdot OPT(\sigma)$.
 - A is a Γ –competitive algorithm.

About the course?

References (optimal)


- **Algorithms** by Dasgupta, Papadimitriou, Vazirani
- **Algorithms Illuminated**, Vols 1,2 and 3 by Tim Roughgarden



Homework

- Homework: **70%**
 - 5 programming + 6 writing homework: $a \leq 55\%$
 - 1 midterm (in-class): $b \leq 15\%$
- 1 final exam: $c \leq 30\%$
- Overall: $a + b + c$
- We encourage **discussion**, but please try them on your **own** before discussion, and conclude them on your **own** after discussion.

Talk to us and each other!

- You can discuss with us at office hours.
 - Question: I do not know how to do it? 
 - Question: This is my approach, but I got a stuck here...
- Office hours
 - Any time on wechat
 - Regular Office Hour: TBD
- Wechat group
 - Check **CANVAS**

Policy

- We encourage discussion on homework, but you should **write down** your solution **on your own**.
- You must **cite** all collaborators, as well as all sources used (e.g., online materials).
- Late policy
 - Within 3 days: **50%** of your score
 - Out of 3 days: **0%**
 - Special Issue

Feedback

- It's my **first course**, so please tell me
 - The **pace** of the lecture
 - The **difficulty** of the homework
 - The **tpyos** in the sldies

Today

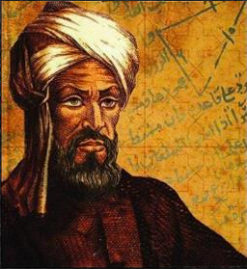
Integer Multiplication

Today's goal

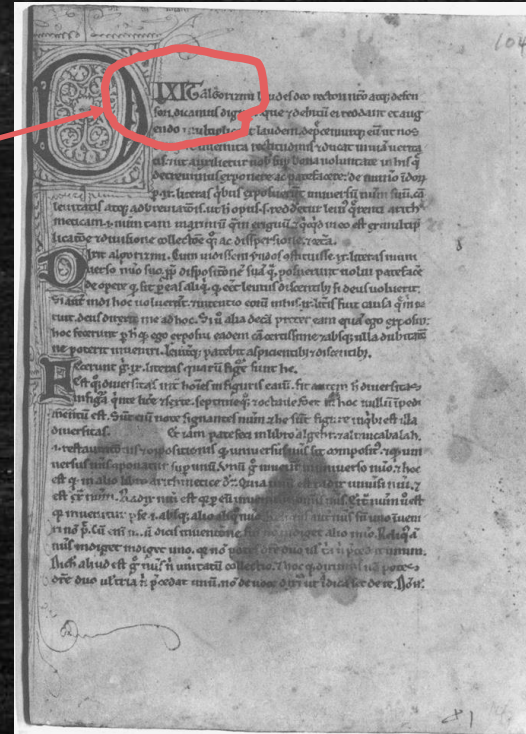
- Karatsuba Integer Multiplication
- Algorithmic Technique
 - Divide and conquer
- Algorithmic Analysis tool
 - Intro to asymptotic analysis

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?

$$\begin{array}{r} \overbrace{123555589124435234523465324}^n \\ \times 875553211231231231231233123 \\ \hline \end{array}$$



How many 1-digit
operation we
need to make?

Roughly

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

$O(n^2)$



How fast is it?

- How many 1-digit operation we need to make?
- We roughly need $5n^2$ 1-digit operations.
- We know we can write $5n^2, 4n^2, 100000n^2$ to be
 - $O(n^2)$
- 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...

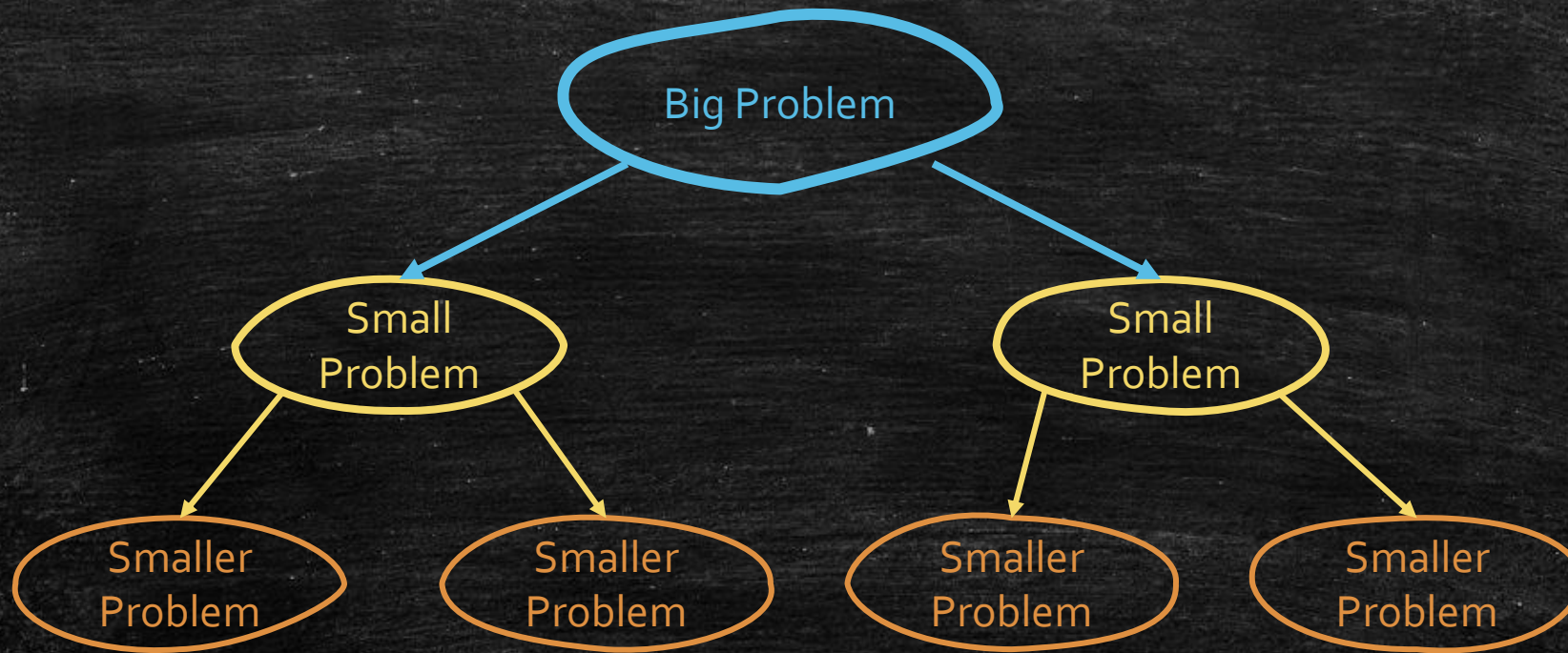
Can we do better?

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 \rightarrow 4 two-digit

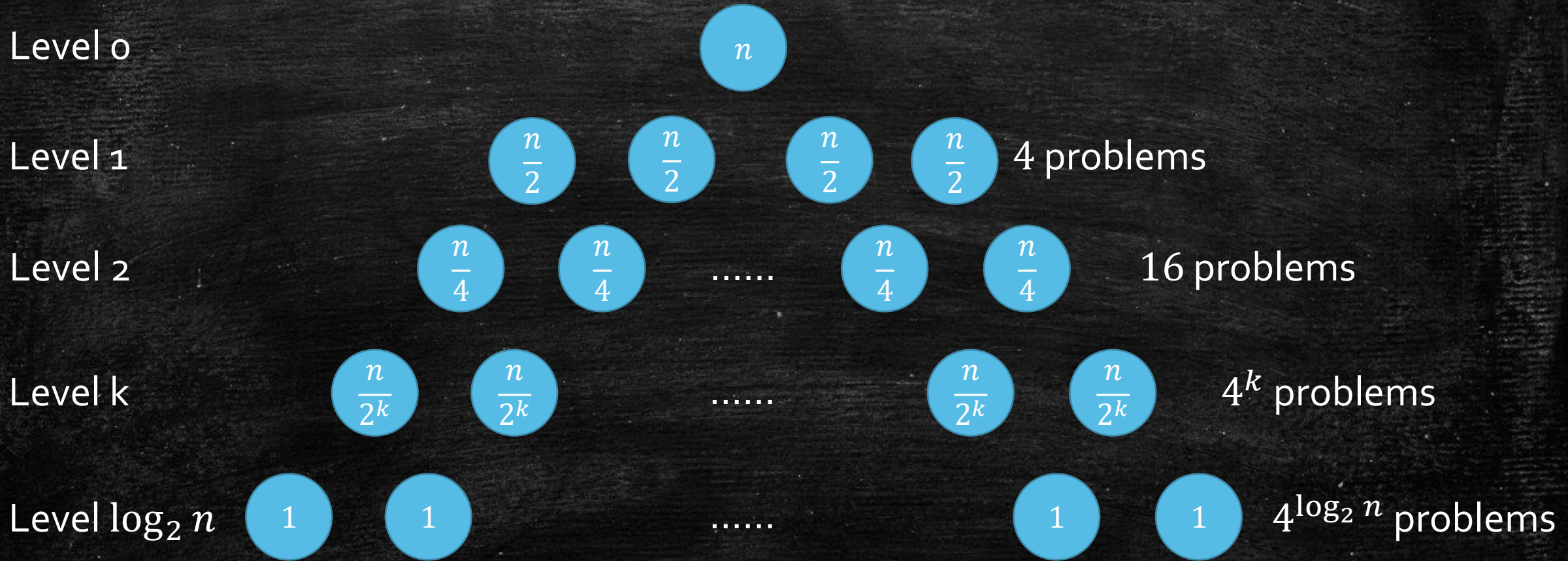
Generally?

- 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.
- $$xy = \left(a \cdot 10^{\frac{n}{2}} + b\right) \left(c \cdot 10^{\frac{n}{2}} + d\right)$$
$$= ac \cdot 10^n + (ad + bc) \cdot 10^{\frac{n}{2}} + bd$$

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 1 n-digit multiplication?
 - **A: n^2 B: n^3 C: n D: $n \log n$**
 - 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



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 $n \times n$
 - Level 1: 4 $\frac{n}{2} \times \frac{n}{2}$
 - Level 2: 16 $\frac{n}{4} \times \frac{n}{4}$
 - Level t : 4^t 1×1
- Conclusion: $4^{\log_2 n} = n^2$

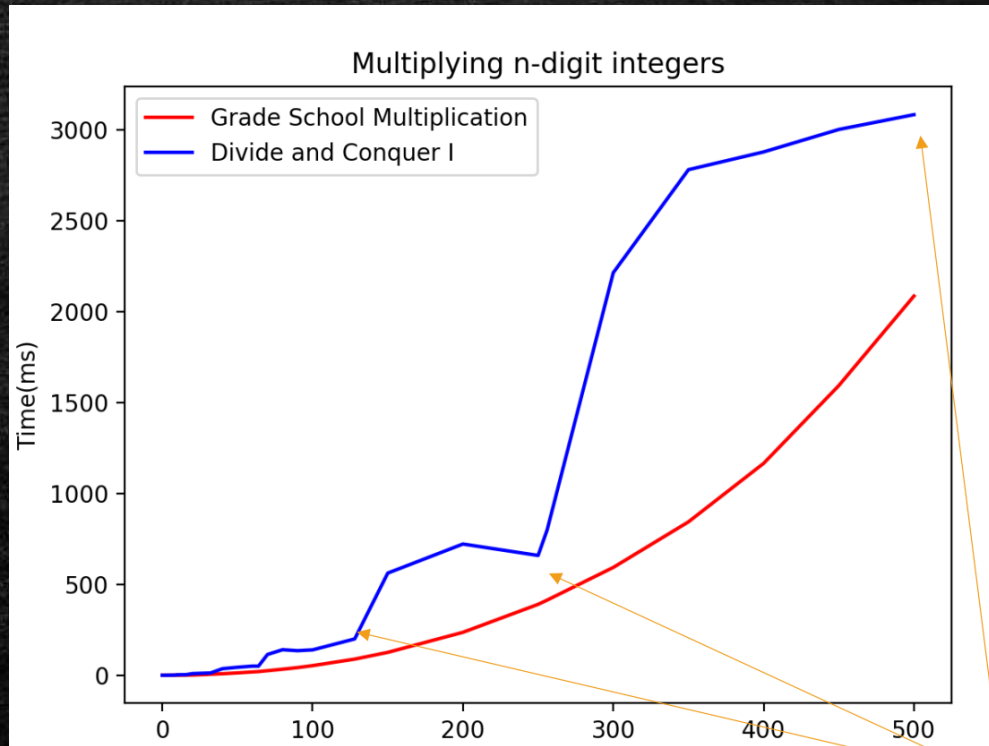
We need more than n^2
one-digit operations!

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
 - $ad + bc$
 - bd
- What do we calculate
 - ac
 - ad
 - bc
 - bd

Karatsuba Algorithm

Improve!

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

Improve!

- What is the difference?
 - We now calculate
 - ac
 - $z = (a + b)(c + d)$
 - bd
 - One n -digit \rightarrow 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: $3^{\log_2 n} = n^{\log_2 3} \approx n^{1.6}$

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

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 $\log_2 n$: $3^{\log_2 n} 1 \times 1$

$8n$ 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! $\rightarrow O(n^{1.465})$
- 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): $O(n \log n)$

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

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

- $\begin{bmatrix} 2 & 9 \\ 7 & 5 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 2 \times 1 + 9 \times 3 & 2 \times 2 + 9 \times 4 \\ 7 \times 1 + 5 \times 3 & 7 \times 2 + 5 \times 4 \end{bmatrix} = \begin{bmatrix} 29 & 40 \\ 22 & 34 \end{bmatrix}$

- $Z = XY$

- $z_{ik} = \sum_{1 \leq j \leq n} x_{ij} y_{jk}$

- How many integer multiplications?

- n^2 entries of Z to calculate
- Each takes n multiplications
- Totally n^3

- What about running time?

Word Ram model?
Turing model?

How to divide and conquer?

Divide and conquer

- Key fact: If $X = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$, $Y = \begin{bmatrix} E & F \\ G & H \end{bmatrix}$.
 - $\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$
- How to divide and conquer?
 - 1 n -size multiplication $\rightarrow 8 \frac{n}{2}$ -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

- $P_1 = A(F - H)$

- $P_2 = (A + B)H$

- $P_3 = (C + D)E$

- $P_4 = D(G - E)$

- $P_5 = (A + D)(E + H)$

- $P_6 = (B - D)(G + H)$

- $P_7 = (A - C)(E + F)$

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

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

$$= \begin{bmatrix} P_5 + P_4 - P_2 + P_6 & P_1 + P_2 \\ P_3 + P_4 & P_1 + P_5 - P_3 - P_7 \end{bmatrix}$$

- 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?

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
 - $n \leq 2^w$ (We need a pointer to store the address.)
 - $a_i \leq 2^w$ (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
- $w = 1$
- Unit Operations
 - Move **one** step.
 - Only **logical** operations.

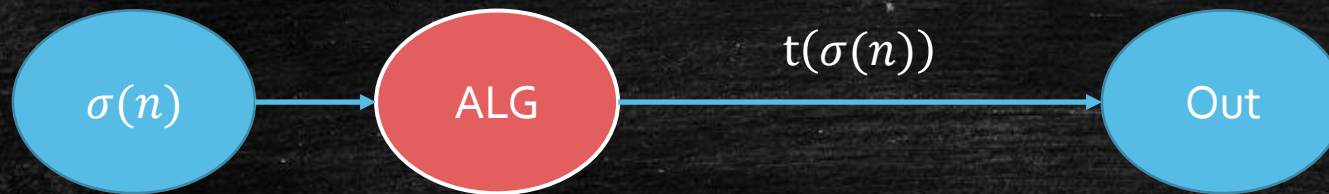
	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$	$i = 6$...	$i = 2^w$
$a[i]$	$\{0,1\}$	$\{0,1\}$	$\{0,1\}$	$\{0,1\}$	$\{0,1\}$	$\{0,1\}$...	$\{0,1\}$

Sample Program

1: int sum=0,n;	+3
2: input(n);	+1
3: int a[n];	+ n
4: input(a);	+ n
5: for i=0 to n	+ n
6: sum+=a[i];	+ n
7: output(sum);	+1

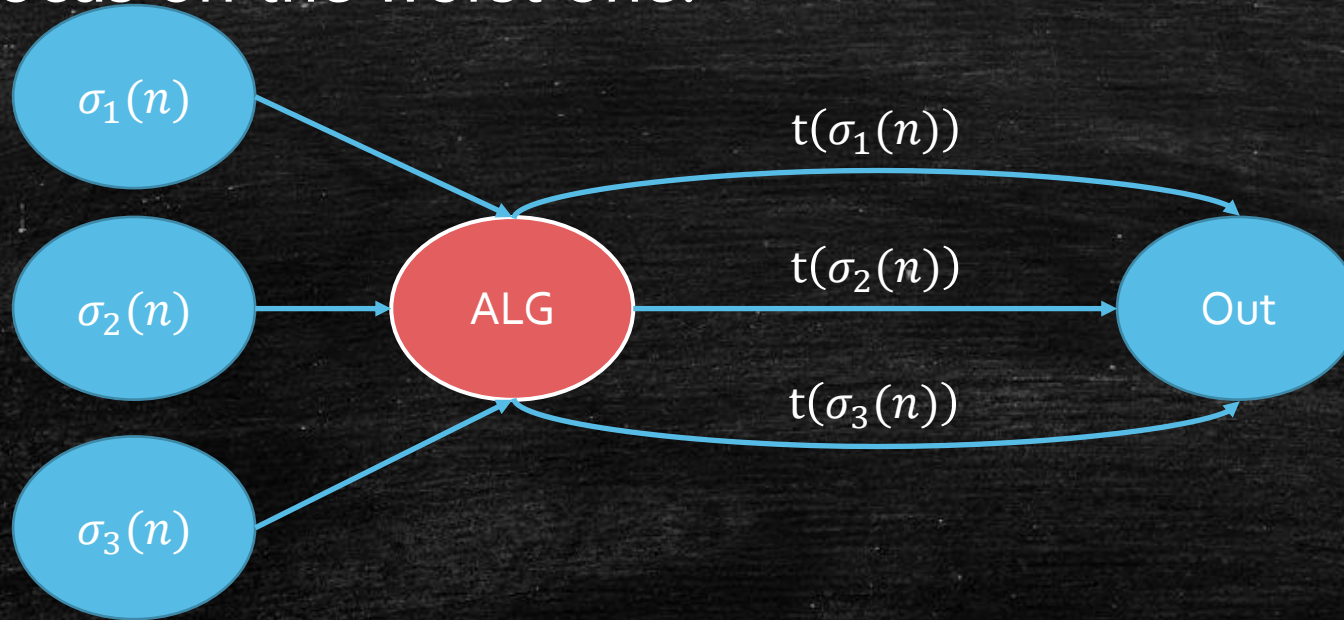
Time Complexity

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

- $T(n) = 103n^2 + 5n + 101 \rightarrow O(n^3)$
- 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)

- $T(n) = O(g(n))$

- $\exists C, n_0, \text{ s.t. } \forall n > n_0, T(n) \leq C \cdot g(n)$

- Big Ω Notation (Lower Bound)

- $T(n) = \Omega(g(n))$

- $\exists C, n_0, \text{ s.t. } \forall n > n_0, T(n) \geq C \cdot g(n)$

- Big Θ Notation (Exact!)

- $T(n) = O(g(n))$

- $T(n) = \Omega(g(n))$

Discussion

- Show n^2 is not $O(n)$.
- Let $p(n) = a_k n^k + a_{k-1} n^{k-1} + a_{k-2} n^{k-2} \dots + a_1 n + a_0$
 - $a_k \geq 0$,
- 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?

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. $c \leftarrow y[1:\frac{n}{2}], d \leftarrow y[\frac{n}{2} + 1:n]$
4. $ac \leftarrow \text{karatsuba}(\frac{n}{2}, a, c)$
5. $bd \leftarrow \text{karatsuba}(\frac{n}{2}, b, d)$
6. $z \leftarrow \text{karatsuba}(\frac{n}{2}, (a - b), (d - c))$
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($n, x[n], y[n]$):

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. $c \leftarrow y[1:\frac{n}{2}], d \leftarrow y[\frac{n}{2} + 1:n]$

4. $ac \leftarrow \text{karatsuba}(\frac{n}{2}, a, c)$

5. $bd \leftarrow \text{karatsuba}(\frac{n}{2}, b, d)$

6. $z \leftarrow \text{karatsuba}(\frac{n}{2}, (a - b), (d - c))$

7. $xy \leftarrow ac \cdot 10^n + (z + ac + bd) \cdot 10^{\frac{n}{2}} + bd$

9. **return** xy

$T(n)$

$O(1)$

$O(\frac{n}{2})$

$O(\frac{n}{2})$

$T(\frac{n}{2})$

$T(\frac{n}{2})$

$T(\frac{n}{2})$

$O(n)$

$O(n)$

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: $+O(1.5^2n)$
 - 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!
