# *Algorithm Design*

Department of Computer Science, Tsinghua University

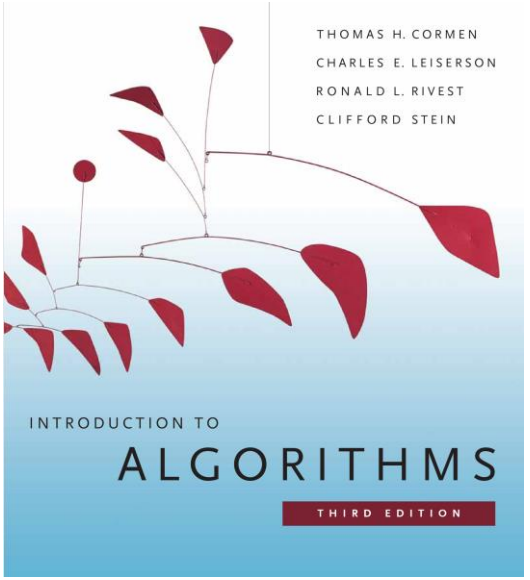
# Welcome to Algorithms Design, Fall 2020

- Instructor: Dr. Ying Zhao
  - Office: east main building 8-204
  - Office hours: Wed 9:00-10:00am
  - Email: yingz@tsinghua.edu.cn
  - Or via WeChat

- TA: Haojia Zuo
  - Office: east main building 8-207
  - Office hours: via WeChat
  - Email: zuohj19@mails.tsinghua.edu.cn

# Course Information

- About textbook:
  - "Introduction to algorithms"
  - by MIT press, 3rd edition




- About homework:
  - due every Thursday.
  - Late homework will be accepted *only if you provide a good reason before its deadline*.
  - Solutions should be well justified. If it requires formal proofs, showing by examples does not count!
  - *Copying from the Internet is not acceptable!*

# Course Resources

**ZHUMU On-line Meeting:**

Meeting ID: 185-708-3755

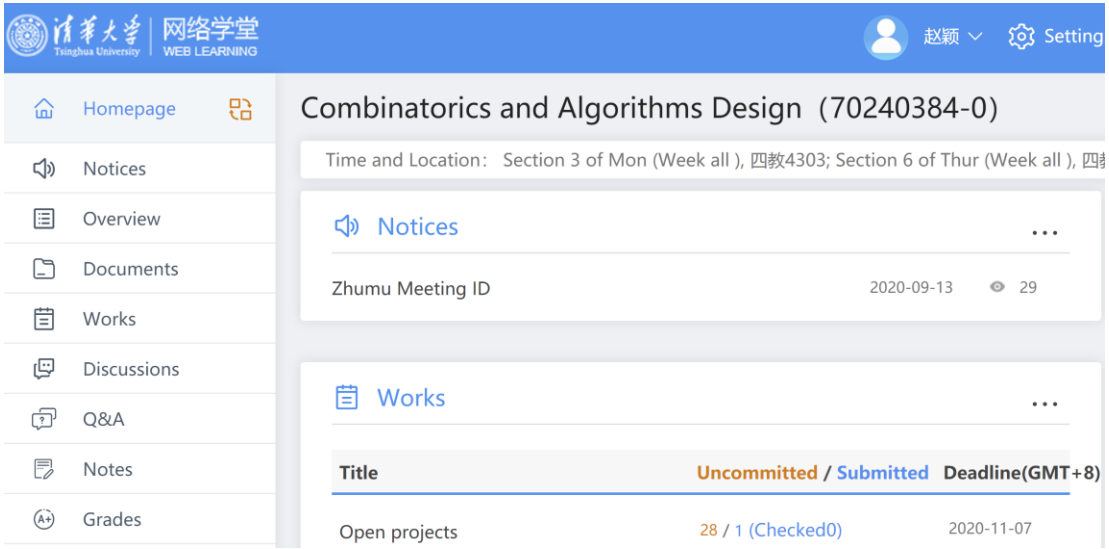
Password: algo2020

**Course WeChat Group:**

In-class exercises, Q&A

**Web Learning Course Page:**

Announcements, Lecture notes, HWs



# Evaluation

- **Grades:** 50-point system
  - Homework & projects: 15 points
  - Final Exam: 35 points
  - Extra points: in-class exercises, up to 3 points

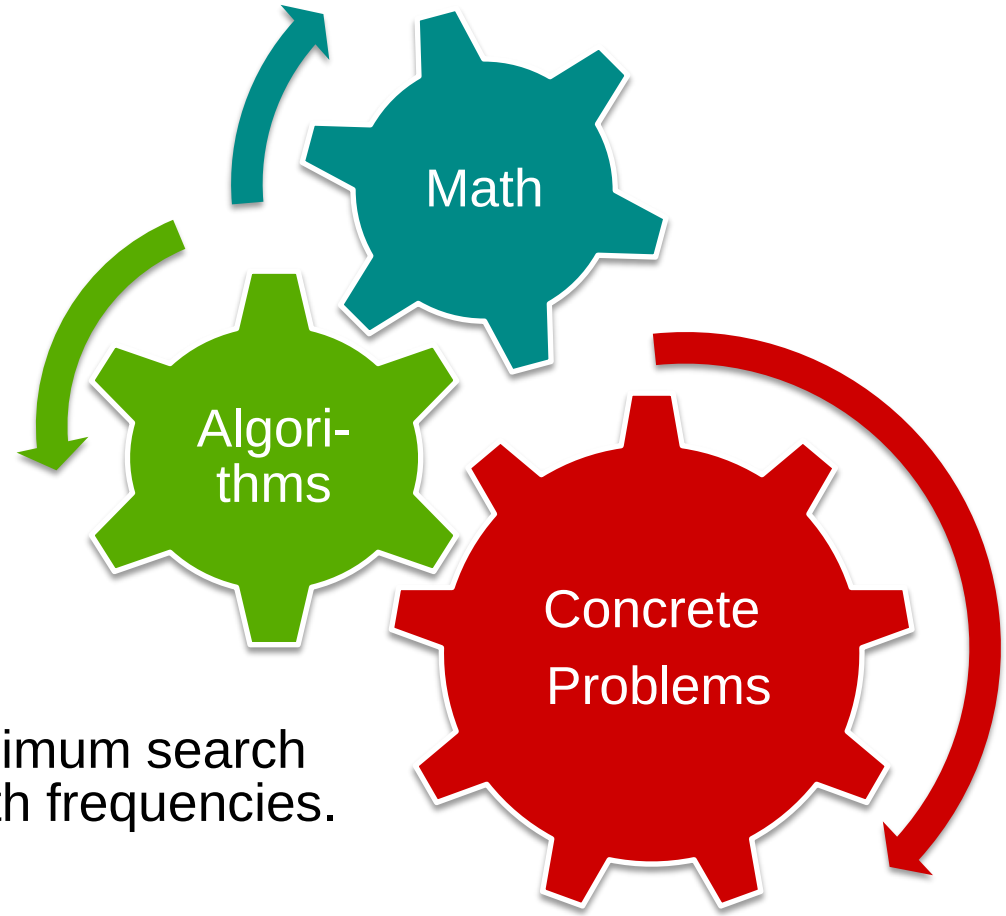
- If you are going to be absent on the day of an examination, you must provide a university-approved justification for your absence before the day of the examination.

# Big Picture

Catalan numbers

Dynamic programming (optimal binary search tree)

Build a search tree of minimum search time for a set of words with frequencies.




# What is Algorithm?

- A well-defined computational procedure for solving a (*math*) problem.
- Computational steps that solve the problem transfer input to output.
- Well-defined:
  - Deterministic: same input → same output
  - Correct for any input
  - Randomized algorithms also depend on a random number generator
- An *instance* of a problem is an input satisfying the constraints of the problem.

# Course Content

- Prerequisite:
  - basic knowledge on programming, data structures, probability, combinatorics…
- Algorithm design strategies:
  - Incremental Algorithm
  - Divide & Conquer (D&C)
  - Randomized Algorithm
  - Dynamic Programming (DP)
- Algorithm analysis methods:
  - Solve recurrences
  - Probabilistic analysis
  - Correctness prove
- Basic Algorithms:
  - Sorting, Order Statistics…

# Schedule

Algorithm Design, Fall 2020, Monday 13:30-15:05 and Thursday 19:20-20:55 @Building #4 4303
Dr. Ying Zhao, East Main Building 8-204, yingz@tsinghua.edu.cn Office Hours: Wed. 9:00-10:00am
Textbook: Introduction to Algorithm 3rd edition, CRLS

| | Monday | Thursday | Readings | Pre-Requisite Self-Checking |
|---|---|---|---|---|
| 11/2-11/6 | **Introduction** | **Midterm：Combinatorics** | *Ch1.1* | |
| 11/9-11/13 | **Complexity**:<br>Running time<br>Growth of function<br>Reduction | **Incremental Algorithm:**<br>Insertion sort<br>Loop invariant<br>Dutch National Flag | *Ch 2.1, 2.2, Ch 3* | *Counting the number of executions of statements in a pseudo-code* |
| 11/16-11/20 | **Divide & Conquer-1:**<br>DC fundamentals<br>Merge sort<br>Order statistics: select | **Divide & Conquer-2**:<br>Solving Recurrences:<br>Substitution, Recursion tree | *Ch 2.3, Ch 9.3*<br>*Ch 4.3, 4.4* | *Merge sort (ch 6.4)*<br>*Quick sort (ch 7.1)*<br>*Summation Formulas (Appendix A.1)* |
| 11/23-11/27 | **Divide & Conquer-3**:<br>Solving Recurrences:<br>Master theorem | **Randomized Algorithms-1:**<br>Probability analysis<br>Balls and bins | *Ch 4.5, 4.6*<br>*Appendix C.3 C.4*<br>*Ch 5.4.2* | *Appendix C.1 C.2* |
| 11/30-12/4 | **Randomized Algorithms-2:**<br>Indicator Random Variable<br>Hire assistant<br>On-line hire assistant | **Randomized Algorithms-3:**<br>RA fundamentals<br>Randomized select | *Ch 5.1-5.3, 5.4.4*<br>*Ch 8.3*<br>*Ch 9.2* | |

# Schedule

| | | | | |
|---|---|---|---|---|
| 12/7-12/11 | **Dynamic Programming-1:**<br>Maximum Subarray | **Dynamic Programming-2:**<br>DP fundamentals<br>Matrix Chain Multiplication | *Ch 4.1*<br>*Ch 15.2, 15.3* | |
| 12/14-12/18 | **Dynamic Programming-3:**<br>All-pairs-shortest-paths | **Greedy Algorithm-1:**<br>Activity selection<br>GA fundamentals | *Ch 25.2*<br>*Ch 16.1, 16.2* | |
| 12/21-12/25 | **Greedy Algorithm-2:**<br>Coin change<br>**Review & QA** | **Final：Algorithms** | | |

# Majority Element

- Input: Given an input array $A[1..n]$ of $n$ numbers
- Output: the majority element (i.e., the element repeatly appears for more than half of the array) if it exists

- Algorithm 1 $O(n \lg n)$:
  - sort the entire array, pick the median and check if it is the majority element

- Algorithm 2 $O(n \lg n)$:
  - Divide and conquer: search in two equal sized subarrays, and then check if the results are the majority element

# Majority Element

- Algorithm 3 $O(n)$:
  - Apply SELECT (an $O(n)$ algorithm) to find the median and check if it is the majority element.

- Algorithm 4 $O(n)$:
  - Incremental Algorithm by Moore and Boyer, 1991


# Majority Element

```
1   candidate = NIL
2   count = 0
3   for i = 1 to n
4       if count == 0
5           candidate = A[i]
6       if candidate == A[i]
7           count = count + 1
8       else count = count − 1
```

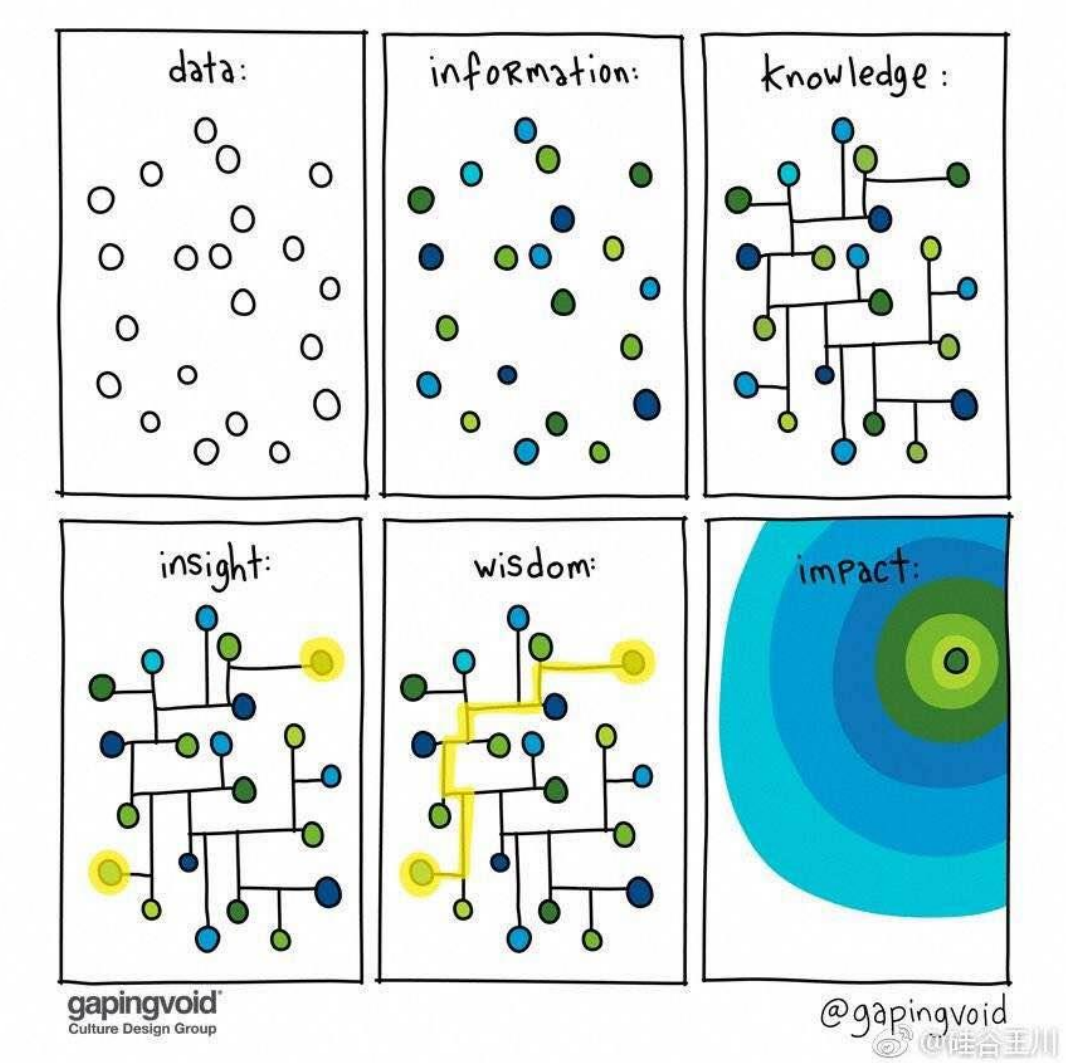
- Loop invariant:
  - Remove *count candidate* from $A[1..i-1]$, the remaining array does not contain any majority element.

# Analysis of Algorithms

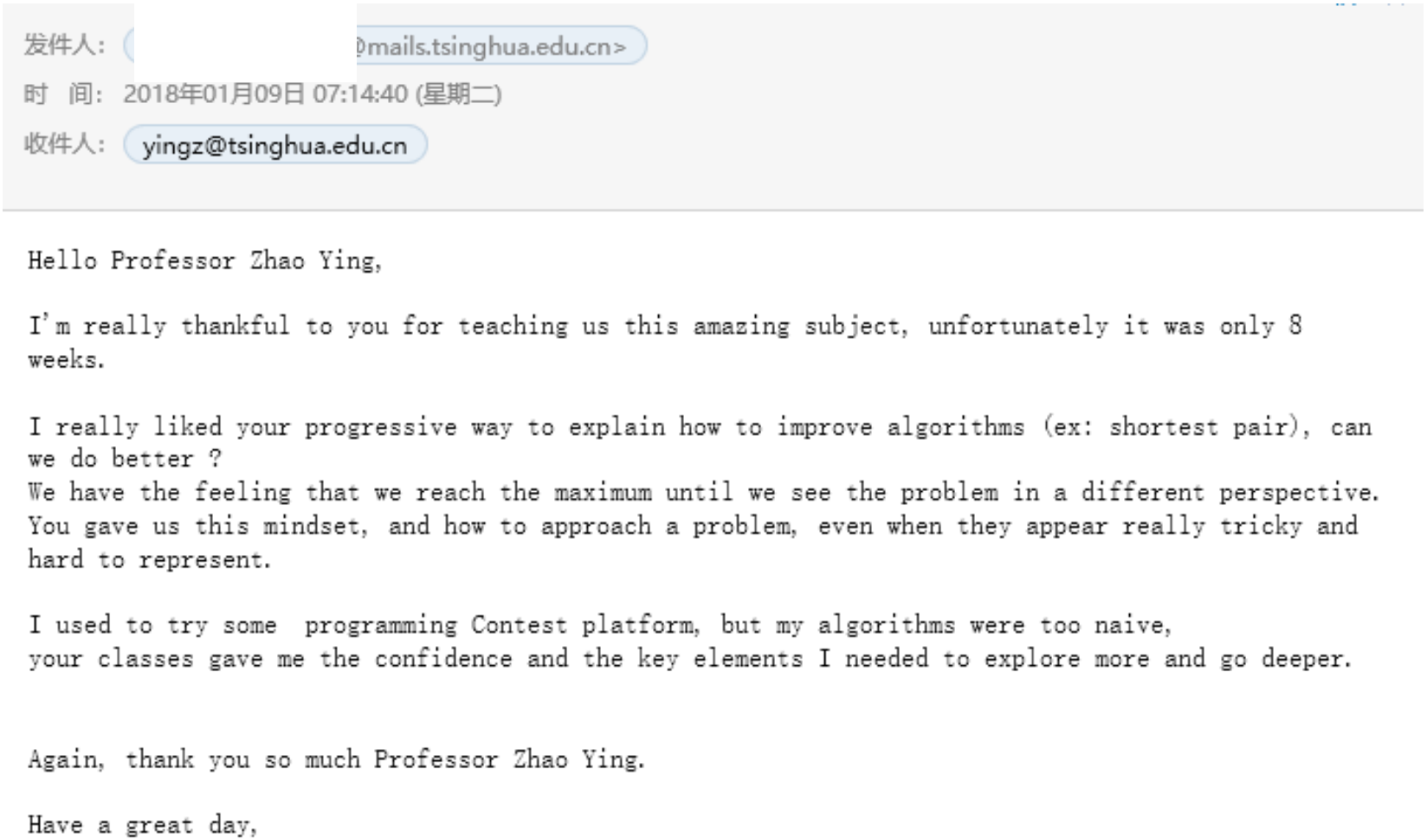
- Why study algorithms and correctness?
  - Program verification is hard.
  - Correctness of an algorithm can be proved.
  - Often verified by ***mathematical induction***.

- Why study algorithms and performance?
  - help us to understand ***scalability***
  - performance often draws the line between what is ***feasible*** and what is ***impossible***.
  - the lessons of program performance generalize to other computing resources.

# How to study Algorithms



# Feedbacks



发件人: @mails.tsinghua.edu.cn>
时 间: 2018年01月09日 07:14:40 (星期二)
收件人: yingz@tsinghua.edu.cn

Hello Professor Zhao Ying,

I'm really thankful to you for teaching us this amazing subject, unfortunately it was only 8 weeks.

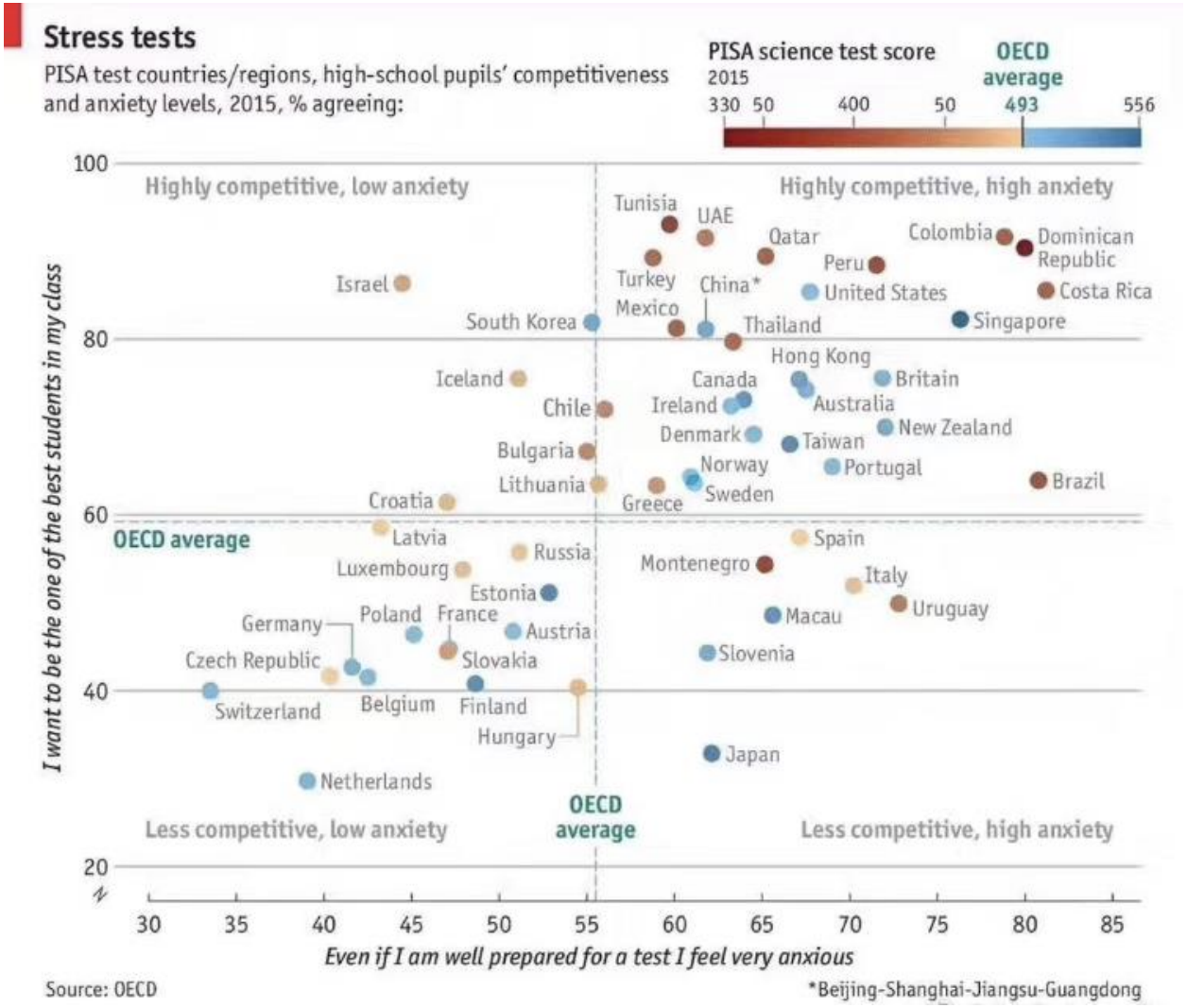
I really liked your progressive way to explain how to improve algorithms (ex: shortest pair), can we do better ?
We have the feeling that we reach the maximum until we see the problem in a different perspective. You gave us this mindset, and how to approach a problem, even when they appear really tricky and hard to represent.

I used to try some programming Contest platform, but my algorithms were too naive,
your classes gave me the confidence and the key elements I needed to explore more and go deeper.

Again, thank you so much Professor Zhao Ying.

Have a great day,

# Difficulties?



# Group Exercise-1

- Form groups of 3-4 students
- For each group, you need to
  - First of all, say hi to your group members;
  - Have a group discussion and find out one common hobby/celebrity/movie…that all your group members share;
  - Have one student present the common hobby/celebrity/movie… (maybe show a picture)