Algorithm Design and Analysis

Introduction

How was your break!?

Today: Jixu Bailan

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
- TAs
  - Programming
    - 杨宗翰
  - Writing
    - 王文茜
    - 赵佳豪
    - 徐遥
    - 毛康睿

张宇昊

杨宗翰


赵佳豪

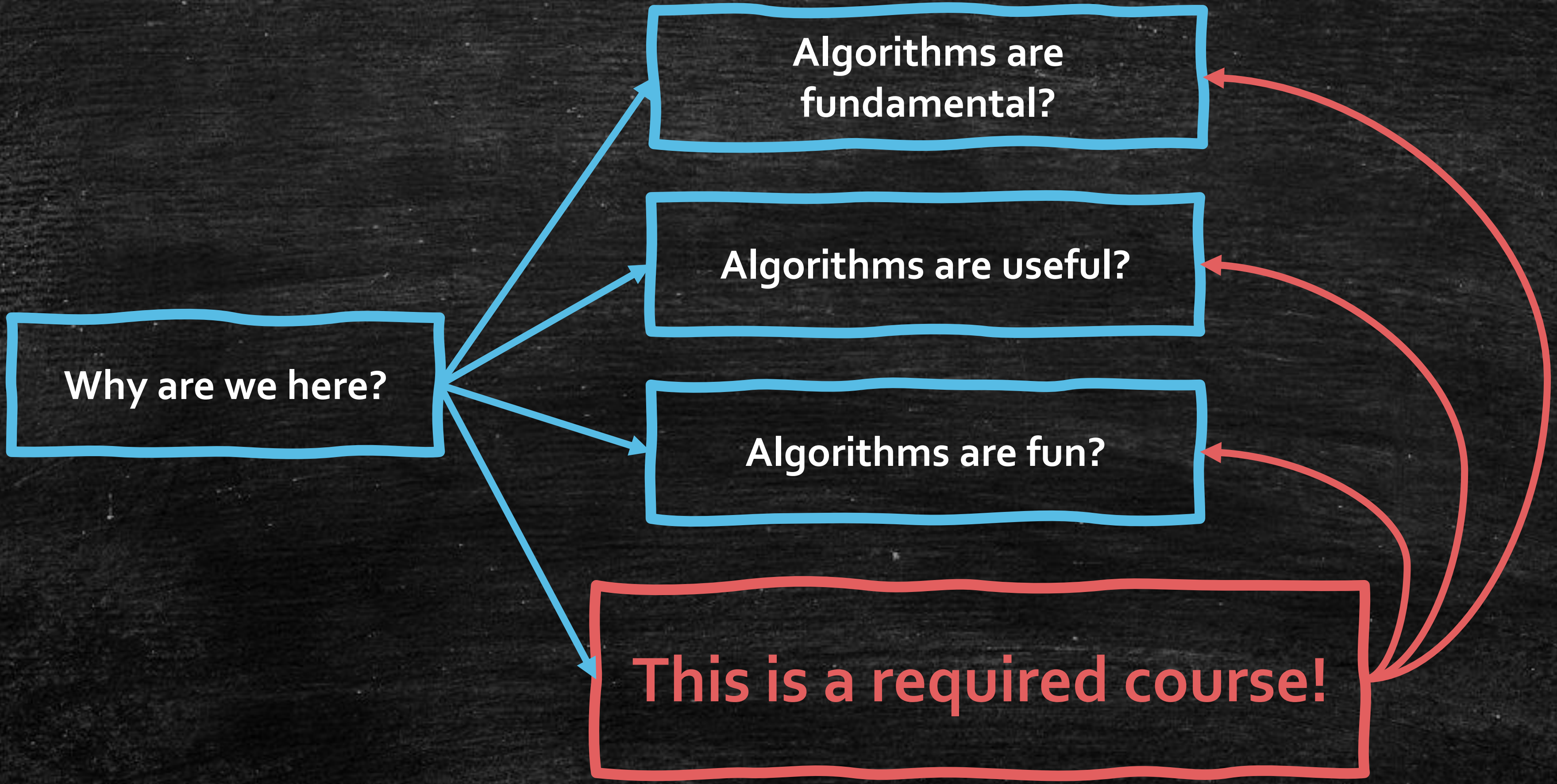

王文茜


毛康睿

徐遥

Why are we here?

Algorithm!

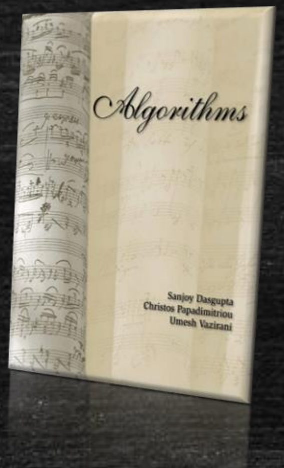

Why are we here?
Algorithms are fundamental?
Algorithms are useful?
Algorithms are fun?
This is a required course!

What is going on?

About the course?

References (optional)

- **Algorithms by Dasgupta, Papadimitriou, Vazirani**



- **You can download this book online.**
- **Refer to many formal proofs on the book.**

Homework

- Homework: **70%**
  - 6 writing homework: $\geq 30\%$
  - 4 programming homework: $\geq 20\%$
  - 1 midterm (in-class): 20%
- Final exam: **30%**
- 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 am stuck here…
- Office hours
  - Any time on wechat
  - Regular Office Hour: TBD
- Wechat group
  - Check **CANVAS**

Policy: Writing

- 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

Plagiarism Policy of Programming

- We have **NO TOLERANCE** on it
- Unless instructed, **you can not submit copied code from any source**
  - Protecting your solution from copying is **your responsibility**
  - If you help others debugging, **do not submit using your own account**
- If the anti-cheating result is confident enough, you will have:
  - For the first time, **NO credit on that homework**
  - Otherwise, **NO credit on that homework** and **ALL upsolving disabled**
  - Trying to avoid anti-cheating by tampering code will count **twice**
  - **We will review code with you before making decision**
- We will do our best to fulfill our declaration here.

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

What is an algorithm?

Informal(or General) Definition

- "In mathematics and computer science, an **algorithm** is a finite **sequence of rigorous instructions**, typically used to solve a class of specific **problems** or to perform a **computation**."
- Is that what we want?
- What is included
  - A computer program.
  - Cook-book recipe.

Why we need a formal definition?

Entscheidungsproblem (Decision Problem)

- David Hilbert and Wilhelm Ackermann (1928)
- The problem asks for an **algorithm(machine)** that considers, as input, a **statement** and **answers "Yes" or "No"** according to whether the statement is universally valid, i.e., valid in every structure satisfying the axioms.
- It asks
  - Can AI be a mathematicians?
  - More general: can machine do any problems we want?

A formal definition of Problems.

- A **Computing Problem**
  - $f: \{0,1\}^* \to \{0,1\}^*$
- Specific: **Decision Problem**
  - $f: \{0,1\}^* \to \{0,1\}$
- Entscheidungsproblem
  - Encode a **statement** to $\{0,1\}^*$.
  - Encode “yes” to 1.
  - Encode “no” to 0.
- Additive
  - Encode two natural numbers to $\{0,1\}^*$.
  - Encode the output natural number to to $\{0,1\}^*$.

Formal Definitions of Algorithms

- **Paper-and-Pencil methods.** (Before 1930)
  - "sequence of rigorous instructions"
  - it can be done by a human without any aids except writing materials.
  - Its instructions need only to be followed **rigorously** to succeed. In other words, it requires no **ingenuity** to succeed.

Formal Definitions of Algorithms (1930s)

- In 1933, **Kurt Gödel**, with **Jacques Herbrand:**
  - **General recursive functions**
  - He tries to define a set of **computable** functions.

Kurt Gödel

Jacques Herbrand

Formal Definitions of Algorithms (1930s)

- In 1936, **Alonzo Church**
  - **λ-calculus**
  - He define “algorithm” to be a sequence of λ-calculus.
  - Programming: functional programming.

Formal Definitions of Algorithms (1930s)

- Also in 1936, **Alan Turing**
  - **Turing machines**
  - He define “algorithm” to be a Turing machine.
  - Programming: imperative programming.

What is a Turing Machine

- Turing machine can formally describe **Paper-and-Pencil methods.**
- A Turing Machine is a Triple (Γ, Q, δ) (partial definition)
  - Γ: **symbols** you can write.
  - Q: The set of **states**.
  - $\delta: \Gamma \times Q \to \Gamma \times Q \times \{L, R\}$.

Church–Turing Thesis

- A function on the natural numbers can be calculated by **an effective method** if and only if it is computable by a **Turing machine**.
  - Turing machine = λ-calculus = C++ program= quantum algorithms
- A computing problem $f:\{0,1\}^* \to \{0,1\}^*$ is computable
  - $\leftrightarrow$ There is a **Turing machine** can correctly give the answer.
  - $\leftrightarrow$ There is a **λ-calculus** can correctly give the answer.
  - $\leftrightarrow$ There is a **C++ program** can correctly give the answer.
  - $\leftrightarrow$ There is a **quantum algorithm** can correctly give the answer.

A Small Question

What is our computer?

Answer Entscheidungsproblem

Are all decision problems solvable?

How to think

- Turing Machine $\rightarrow \{0,1\}^*$
  - We can encode a Turing machine.
  - How many decision problems exist?
  - The number of integers.
- Decision Problem: $f: \{0,1\}^* \rightarrow \{0,1\}$
  - How many decision problems exist?
  - The number of real numbers (binary)
- The number of decision problems is much more then the number of Turing machines.

Halt Problem

- $Halt(TM, y)$ has two input,
  - TM is an encoded Turing machine.
  - y is an input of TM.
- $Halt(TM, y) = \begin{cases} 1 & \text{if } TM(y) \text{ halt} \\ 0 & \text{if } TM(y) \text{ does not halt} \end{cases}$

Does $Halt(TM, y)$ exist?

- Let $A(TM, y) = Halt$.
- Let us define a new $TM \rightarrow B(x)$.

```
Function B(x)
    if A(x, x) = 0 then
        return 1;
    else
        for (;;);
```

Find a contradiction

- Input B to $B(x)$
  - What is $B(B)$?

Function $B(x)$
    if $A(x, x) = 0$ then
        return 1;
    else
        for (;;);

Find a contradiction

- Input B to $B(x)$
  - What is $B(B)$?

```
Function B(x)
    if A(B, B) = 0 then
        return 1;
    else
        for (;;);
```

Find a contradiction

- Input B to $B(x)$
  - What is $B(B)$?
- If $B(B)$ does not halt → It halt (return 1).
- If $B(B)$ halt → It does not halt.

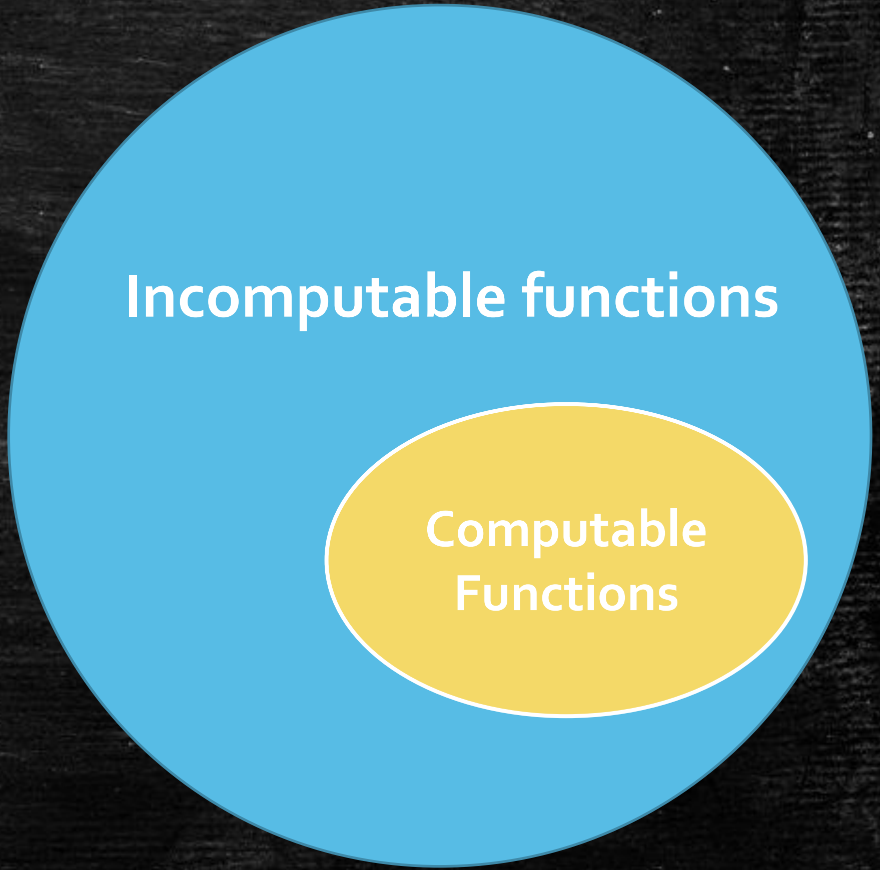
```
Function B(x)
    if A(B,B) = 0 then
        return 1;
    else
        for (;;);
```

Answer Entscheidungsproblem

Not all Entscheidungsproblem are solvable, e.g., Halt.

Computing Problems: Overview

- Research about Incomputable functions.
  - Computability Theory
- Research about computable functions.
  - Algorithms and Complexity



Our course

Computable Functions

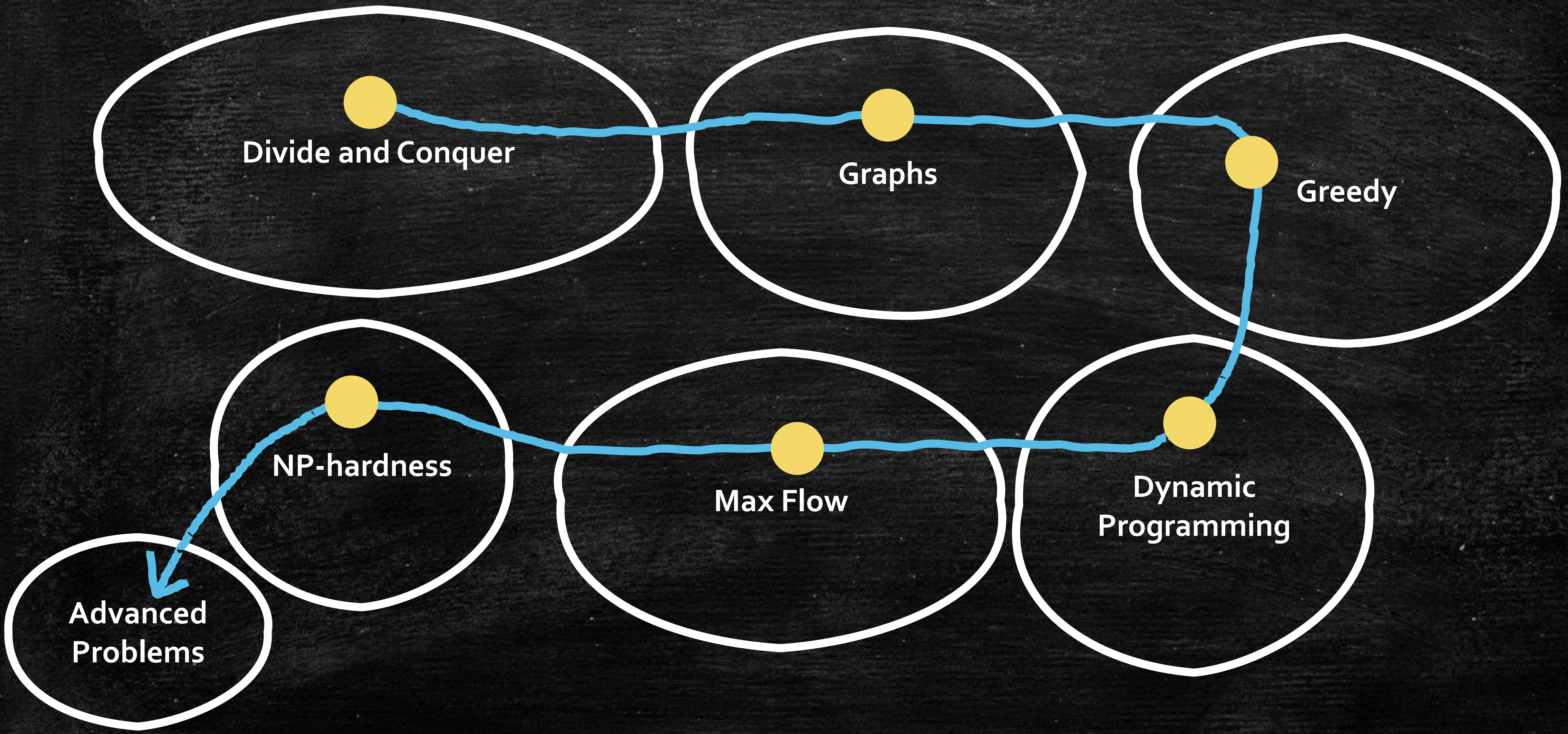
Algorithms for computable functions

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


Divide and Conquer
Graphs
Greedy
Dynamic Programming
Max Flow
NP-hardness
Advanced Problems

How to analyze algorithms?

Guide questions

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

How to think in most of this course?

- Does the algorithm **work**?
  - Optimal or correct answer
  - **Exact Algorithms**
- Is it **fast**?
  - Time complexity
  - Worst case
- Can I do **better**?
  - More efficient
  - Better time complexity

Most Important Thing

Questions & Discussion

Aside the course.

- 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

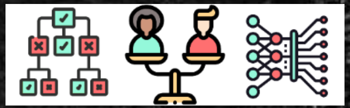
$(x \text{ OR } y \text{ OR } z) \text{ AND } (x \text{ OR } \bar{y} \text{ OR } z) \text{ AND }$
$(x \text{ OR } y \text{ OR } \bar{z}) \text{ AND } (x \text{ OR } \bar{y} \text{ OR } \bar{z}) \text{ AND }$
$(\bar{x} \text{ OR } y \text{ OR } z) \text{ AND } (\bar{x} \text{ OR } \bar{y} \text{ OR } \bar{z})$

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 $\Gamma-$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 $\Gamma -$competitive algorithm.