



Introduction to

# *Algorithm Design and Analysis*

## [1] Model of Computation



*Yu Huang*

<http://cs.nju.edu.cn/yuhuang>  
Institute of Computer Software  
Nanjing University



# Course Information

- Syllabus
- Textbook
- Website



# Syllabus



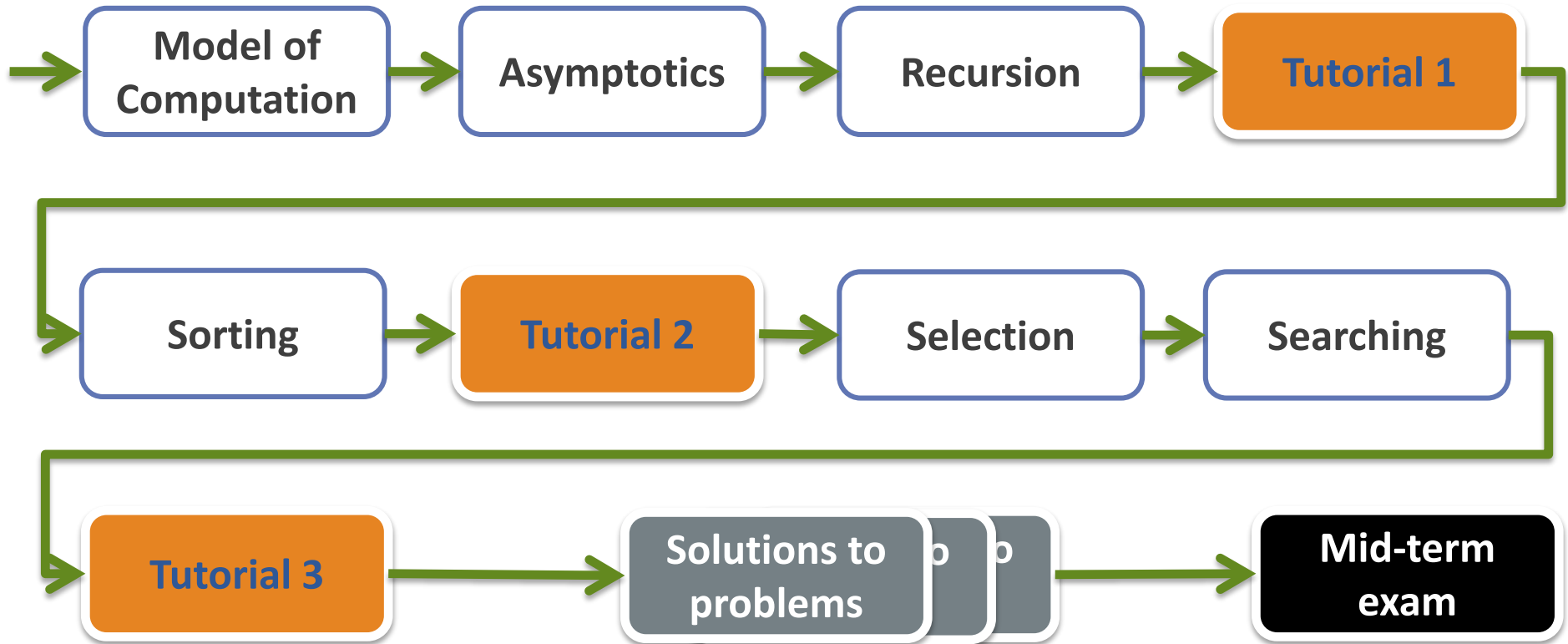
Model of  
Computation

Algorithm  
Design &  
Analysis  
Techniques

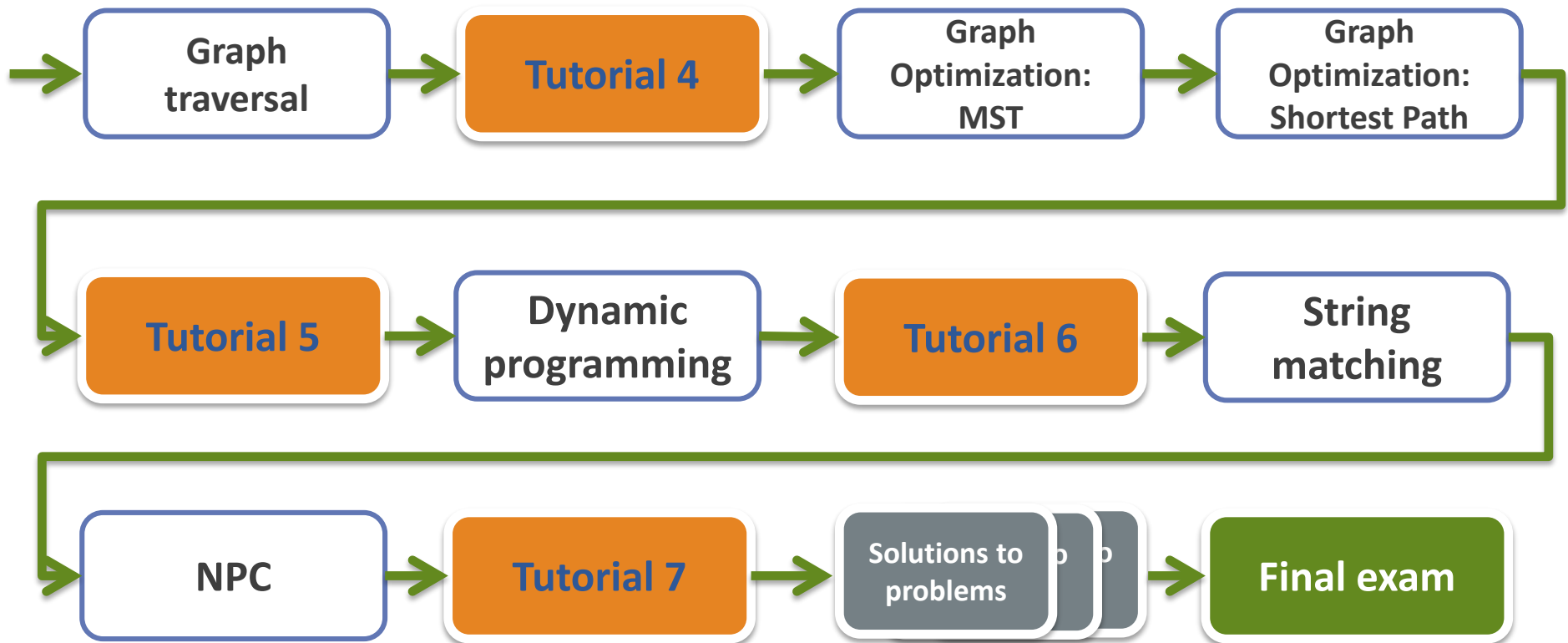
Computation  
Complexity



# Syllabus



# Syllabus



# Syllabus

**Strategies**

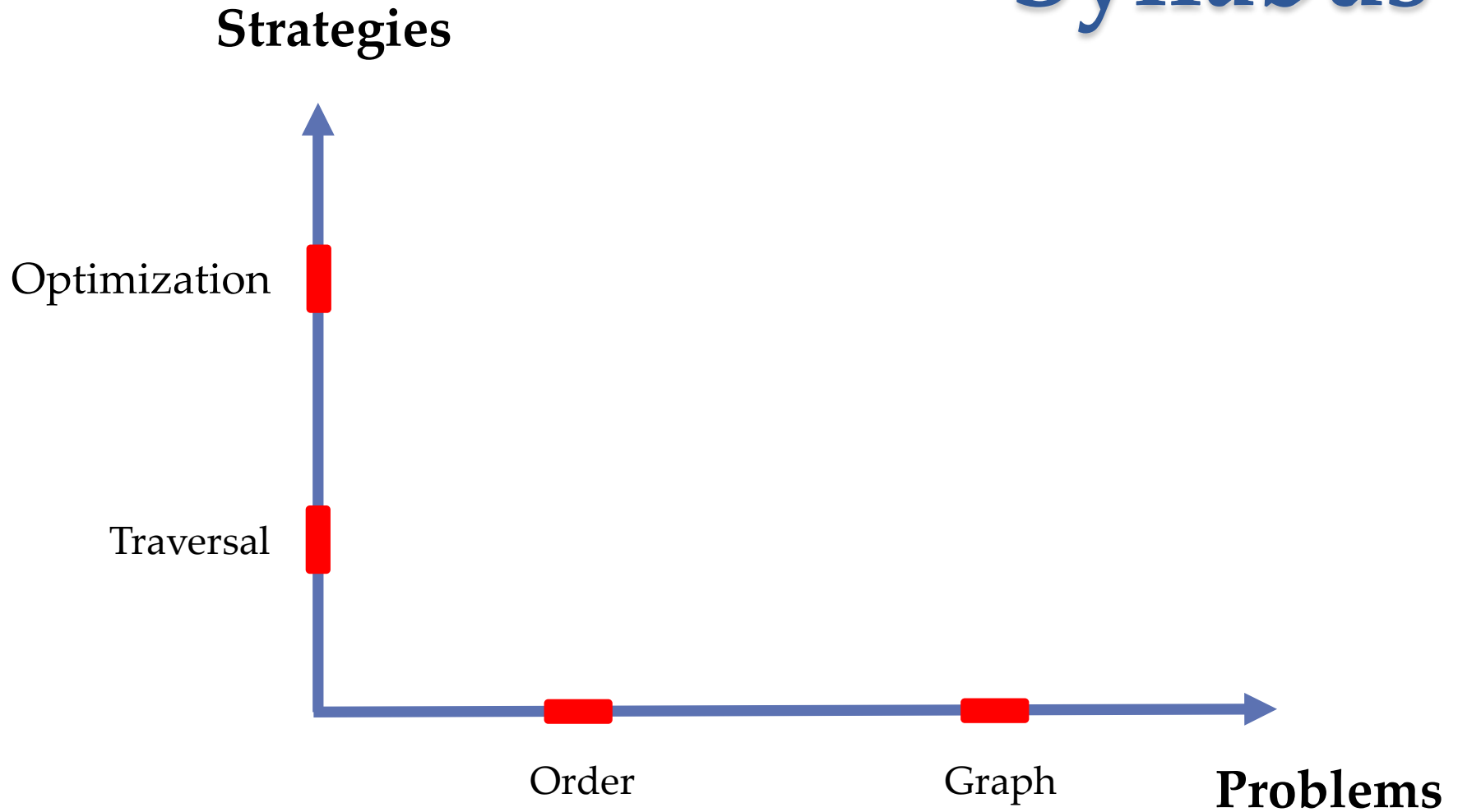


Algorithm  
Design & Analysis

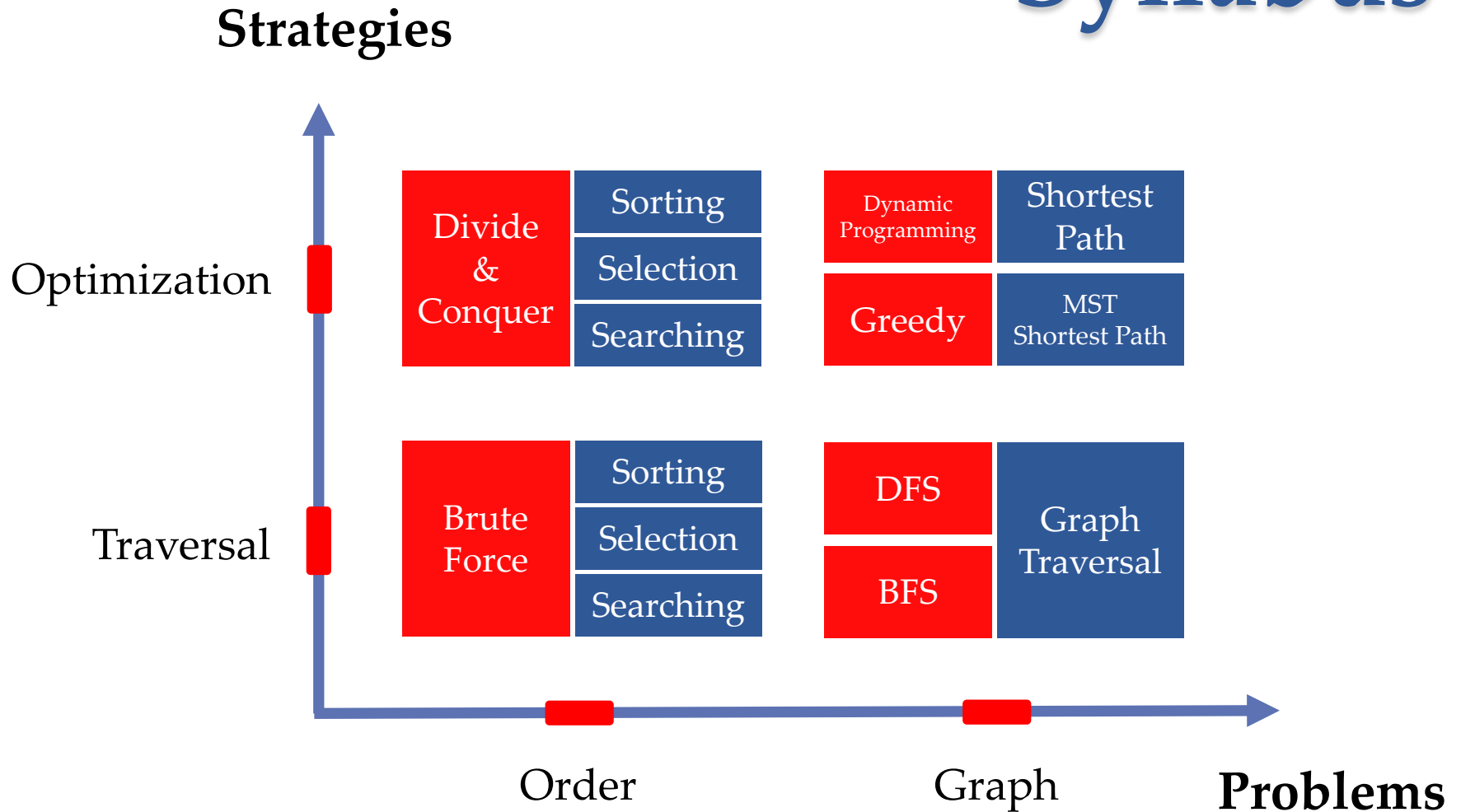
**Problems**



# Syllabus



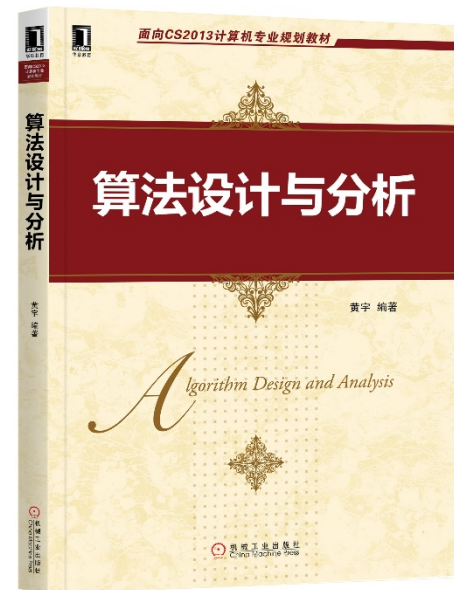
# Syllabus





# Textbooks

- **Course outline: LADA**
  - Lectures on **A**lgorithm **D**esign & **A**nalysis (slides)
- **Course contents**
  - Algorithm Design and Analysis

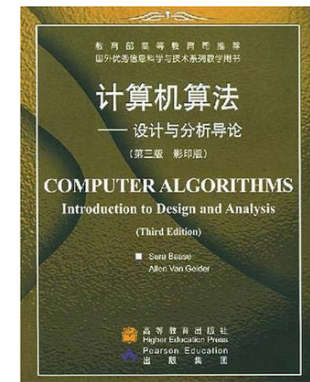
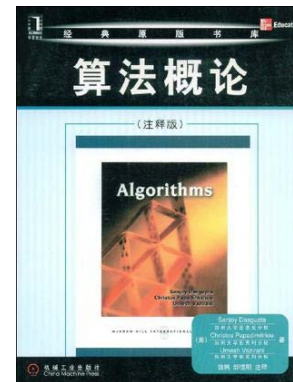
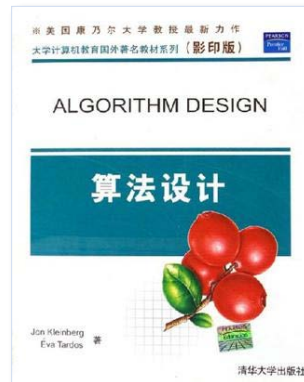
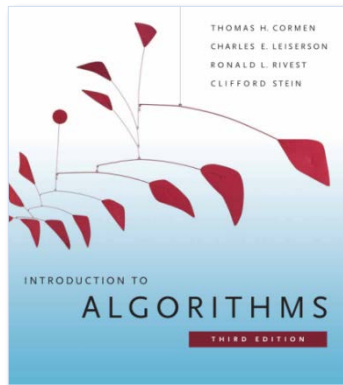


More info about the book:  
<https://zhuanlan.zhihu.com/p/24150569>

# Textbooks

- Further reading
  - Introduction to Algorithms
  - Algorithm Design
  - Algorithms
  - Computer Algorithms\*

See the “douban list” for more info:  
<http://book.douban.com/doulist/1155824/>



# Algorithm – Design & Analysis

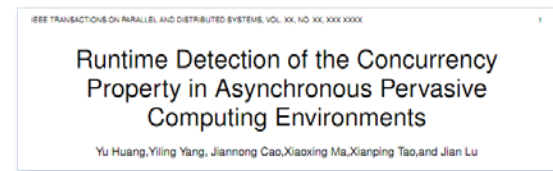
- **Algorithm - the spirit of computing**
  - Model of computation
- **Algorithm by example**
  - Greatest common divisor
  - Sequential search
- **Algorithm design & analysis**
  - Correctness
  - Worst-case / average-case cost analysis



# Computer and Computing

- Problem 1

- Why the computer **seems to** be able to do anything?
  - Scientific computing, document processing, computer games, ebooks, movies, computer games, ...

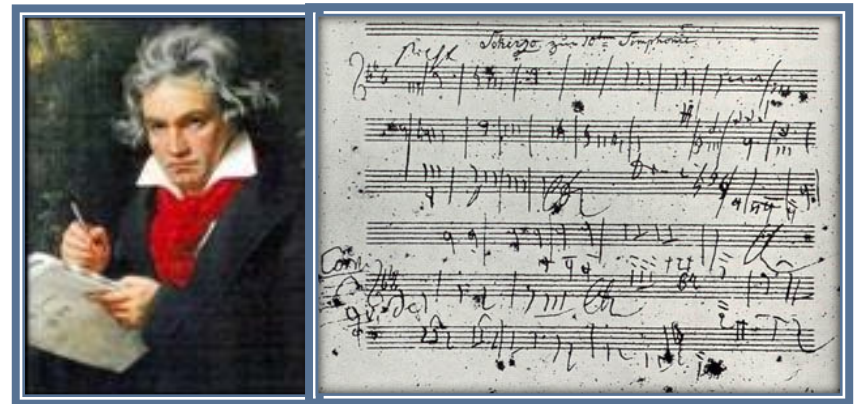


# Computer and Computing

## • Problem 2

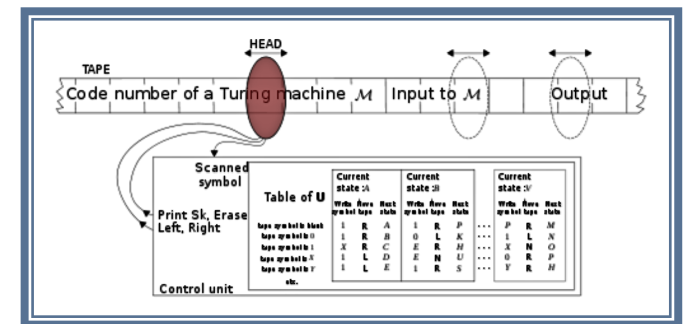
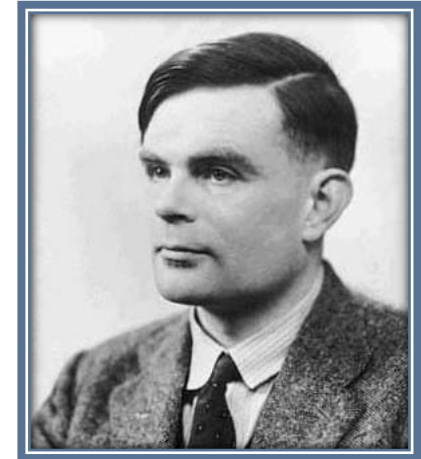
- What can / cannot be efficiently done by a computer?
  - Manage millions of songs vs. music composition

新歌TOP100		歌曲TOP500	
1	▶ 只是太爱你 🏆 张敬轩	1	▶ 没那么简单 🏆 黄小琥
2	▶ 绿旋风 凤凰传奇	2	▶ 走天涯 隆央卓玛
3	▶ 好朋友只是... 郁可唯	3	▶ 漂亮的姑娘... 龙梅子
4	▶ 配角 sara	4	▶ 小三 冷漠
5	▶ 有时候 张靓颖	5	▶ 无法原谅 李佳璐
6	▶ 曾经太年轻 蓝又时	6	▶ 都要好好的 小沈阳
7	▶ 你和我时... 张惠妹	7	▶ 老男孩 筷子兄弟
8	▶ 爱情掉在哪... 井柏然	8	▶ 我相信 杨培安
9	▶ 一瞬之光 a-lin	9	▶ 爱的供养 杨幂
10	▶ 狂想曲 萧敬腾	10	▶ 春天里 汪峰
<a href="#">更多&gt;&gt;</a> <a href="#">试听全部</a>		<a href="#">更多&gt;&gt;</a> <a href="#">试听全部</a>	



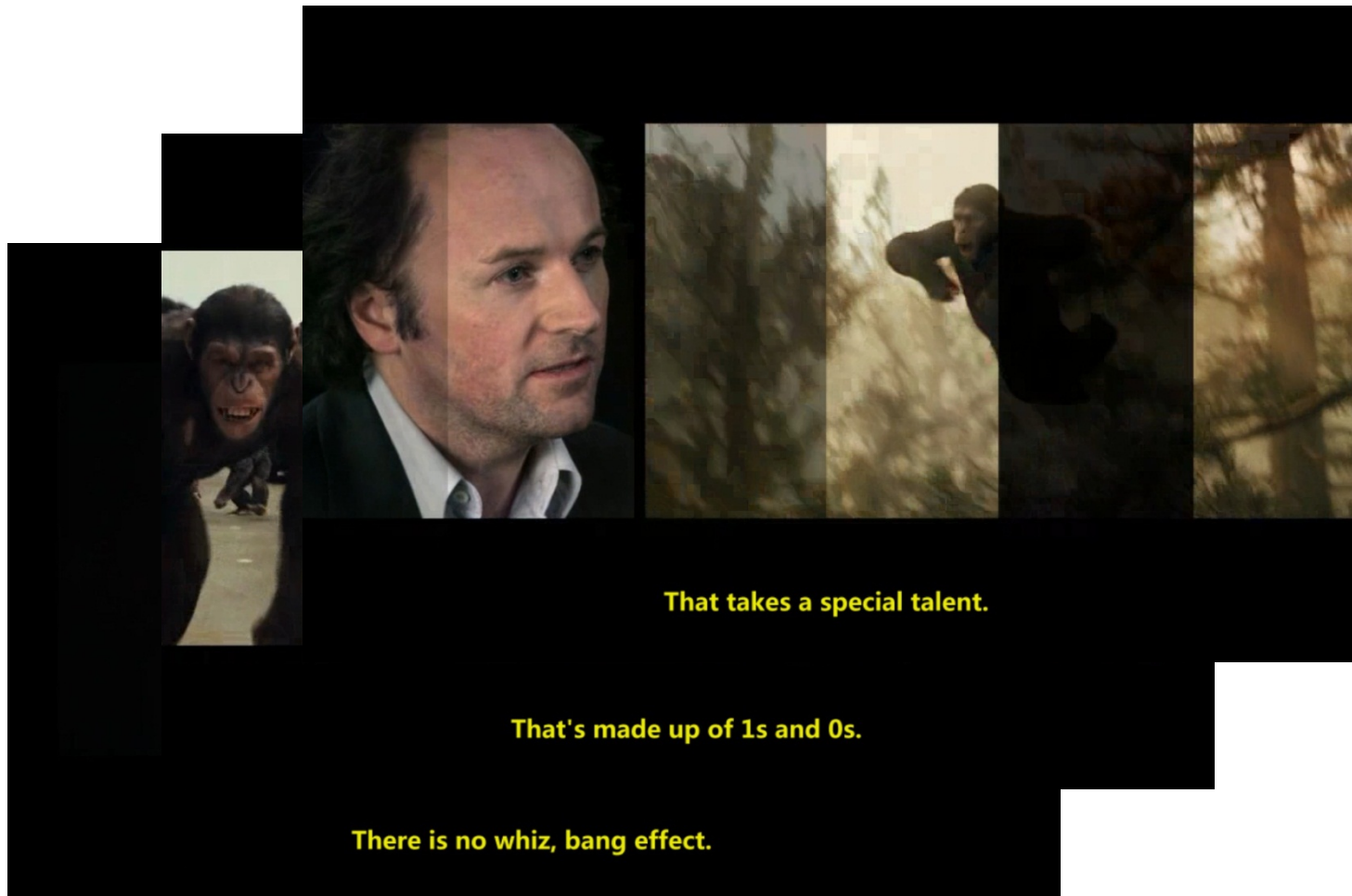
# Computer and Computing

- **Computing**
  - Encoding everything into '0's and '1's
  - Operations over '1's and '0's
  - Decoding the '1's and '0's
- **Turing machine**
  - An abstract/logical computer





# Computing in Everyday Life



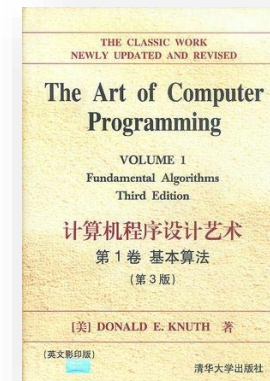
# Algorithm





# Algorithm

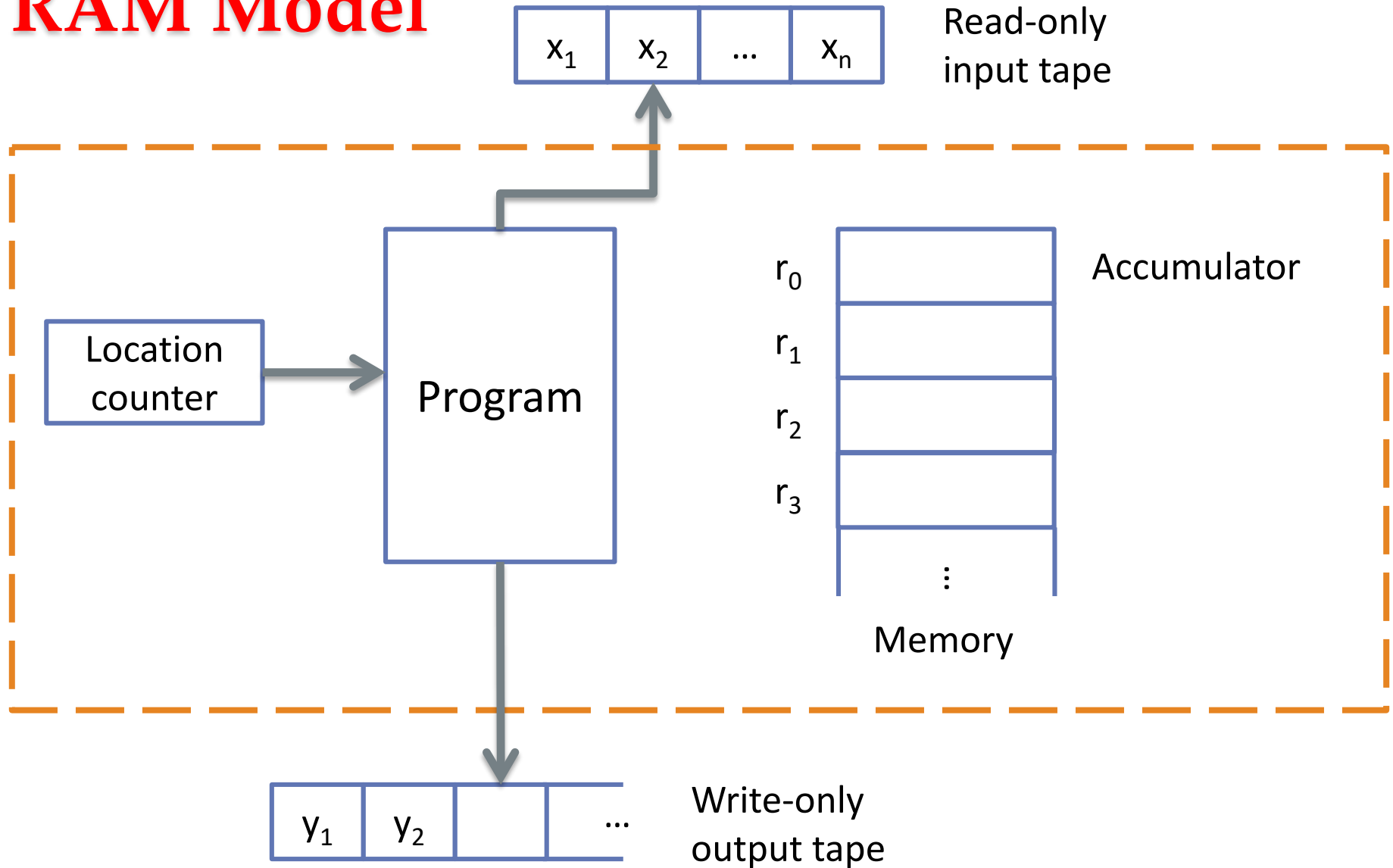
- **Algorithm is the spirit of computing**
  - To solve a specific problem (so called an *algorithmic problem*)
  - Combination of basic operations
    - in a precise and elegant way
- **Essential issues**
  - Model of computation
  - Algorithm design
  - Algorithm analysis



# Model of Computation

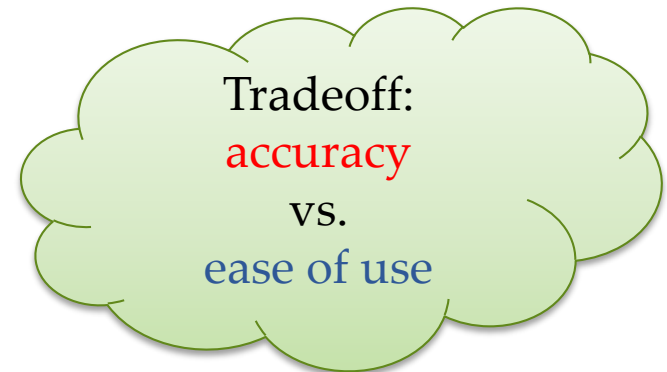
- **Problems**
  - Why the algorithms we learn can run almost everywhere?
  - Why the algorithms we learn can be implemented in any language?
- **Machine- and language- independent algorithms, running on an **abstract** machine**
  - Turing machine: over-qualify
  - RAM model: simple but powerful

# RAM Model



# The RAM Model of Computation

- Each *simple operation* takes one time step
  - E.g., key comparison, +/-, memory access, ...
- Non-simple operations should be decomposed
  - Loop
  - Subroutine
- Memory
  - Memory access is a simple operation
  - Unlimited memory



# Further Reading

“哼，你让他们成楔形攻击队形不就行了？”秦始皇轻蔑地看着冯·诺伊曼。牛顿不知从什么地方掏出六面小旗。三白三黑，冯·诺伊曼接过来分给三名士兵，每人一白一黑，说：“白色代表0，黑色代表1。好，现在听我说，出，你转身看着入1和入2，如果他们都举黑旗，你就举黑旗，其他的情况你都举白旗，这种情况有三种：入1白，入2黑；入1黑，入2白；入1、入2都是白。”

“不需要，我们组建一千万个这样的门部件，再将这些部件组合成一个系统，这个系统就能进行我们所需要的运算，解出那些预测太阳运行的微分方程。这个系统，我们把它叫做……嗯，叫做……”

“计算机。”汪淼说。

“啊——好！”冯·诺伊曼对汪淼竖起一根指头，“计算机，这个名字好，整个系统实际上就是一部庞大的机器，是有史以来最复杂的机器！”

刘慈欣，《三体、牛顿、冯·诺依曼、秦始皇、三日连珠》，《三体》第一部



# To Create an Algorithm

- **Algorithm design**
  - Composition of simple operations, to solve an algorithmic problem
- **Algorithm analysis**
  - Amount of work done / memory used
    - In the worst/average case
  - Advanced issues
    - Optimality, approximation ratio, ...

# Algorithm by Example

- **Algorithmic Problem 1**
  - Find the greatest common divisor of two non-negative integers  $m$  and  $n$
- **Algorithmic Problem 2**
  - Is a specific key  $K$  stored in array  $E[1..n]$ ?



# Probably the Oldest Algorithm

- Euclid Algorithm

## Problem

- Find the greatest common divisor of two non-negative integers  $m$  and  $n$

## Specification

Input: non-negative integer  $m, n$   
Output:  $\text{gcd}(m, n)$

## Euclid algorithm

[E1]  $n$  divides  $m$ , the remainder  $\rightarrow r$   
[E2] if  $r = 0$  then return  $n$   
[E3]  $n \rightarrow m; r \rightarrow n$ ; goto E1

## Euclid algorithm – recursive version

Euclid( $m, n$ )  
[E1] if  $n=0$  then return  $m$   
[E2] else return Euclid( $n, m \bmod n$ )





# Sequential Search

## Problem

- Search an array for a specific key

## Specification

Input: K, E[1..n]

Output: Location of K (1,2,...,n; -1: K is not in E[])

## Sequential searchEuclid algorithm

```
Int seqSearch(int[] E, int n, int K)
    int ans, index;
    ans=-1;
    for (index=1; index<=n; index++)
        if (K==E[index])
            ans=index;
            break;
    Return ans;
```

# Algorithm Design

- **Criteria**

- Defining correctness

## Specification

Input: non-negative integer  $m, n$   
Output:  $\text{gcd}(m, n)$

- **Main challenge**

- For proving correctness

## Main challenge

- The output is **always** correct, for **any** legal input.
- Infinite possible inputs

- **Our strategy**

- Mathematical induction
- ...

## Mathematical induction

- Weak principle
- Strong principle



# For Your Reference

- Mathematical induction

## The **Weak** Principle of Mathematical Induction

- If the statement  $p(b)$  is true and the statement  $p(n-1) \Rightarrow p(n)$  is true for all  $n > b$ , then  $p(n)$  is true for all integers  $n \geq b$ .

## The **Strong** Principle of Mathematical Induction

- If the statement  $p(b)$  is true, and the statement  $\{p(b) \text{ and } p(b+1) \text{ and } \dots \text{ and } p(n-1) \Rightarrow p(n)\}$  is true, for all  $n > b$ , then  $p(n)$  is true for all integers  $n \geq b$ .

# Correctness of the Euclid Algorithm

- Induction on  $n$

- Base case

- $n = 0$ : for any  $m$ ,  $\text{Euclid}(m, 0) = m$ ;
    - $n = 1$ : for any  $m$ ,  $\text{Euclid}(m, 1) = 1$ ;
    - $n = 2$ : ...

- Assumption

- For any  $n \leq N_0$ ,  $\text{Euclid}(m, n)$  is correct;

- Induction

- $\text{Euclid}(m, N_0+1) = \text{Euclid}(N_0+1, m \bmod (N_0+1))$ ;



$$\gcd(m, N_0+1) = \gcd(N_0+1, m \bmod (N_0+1))$$

# Notes on Induction

“Notes on Structured Programming”, E.W. Dijkstra

I have mentioned **mathematical induction** explicitly, because it is the only pattern of reasoning that I am aware of, that eventually enables us to cope with loops and recursive procedures.

# Algorithm Analysis

- **Criteria**
  - Performance metrics
- **Worst case**
  - Best case?
- **Average case**
  - Average cost?
- **Advanced topics**
  - Lower bound, optimality, ...

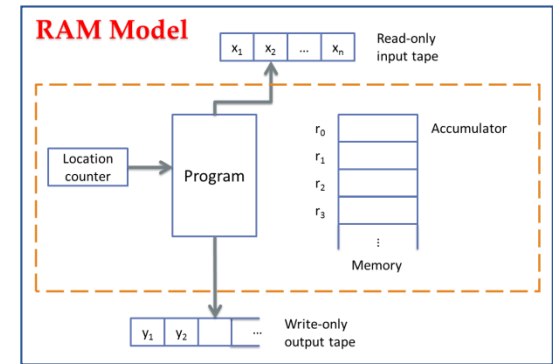


# Algorithm Analysis

- **How to measure**
  - Not too general
    - Giving essential indication in comparison of algorithms
  - Not too precise
    - Machine independent
    - Language independent
    - Programming paradigm independent
    - Implementation independent

# Algorithm Analysis

- **Criteria**
  - Critical operation
  - How many critical operation are conducted
- **For example**



Algorithmic problem	Critical operation
Sorting, selection, searching String matching	Comparison (of keys)
Graph traversal	Processing a node/edge
Matrix multiplication	Multiplication



# Algorithm Analysis

- **Amount of work done**
  - usually depends on size of the input
  - usually does not depend on size of the input only



# Worst-case Complexity

- $W(n)$ 
  - Upper bound of cost
    - For any possible input
  - $W(n) = \max_{I \in D_n} f(I)$

# Average-case Complexity

- **A(n)**
  - Weighted average
  - $A(n) = \sum_{I \in D(n)} \Pr(I) f(I)$
- **A special case**
  - Average cost
    - Total cost of all inputs, averaged over the input size
  - $Average(n) = \frac{1}{|D(n)|} \sum_{I \in D(n)} f(I)$

# Average-case Cost of *SeqSearch*

- **Case 1: K is in E[]**

- Assumptions:

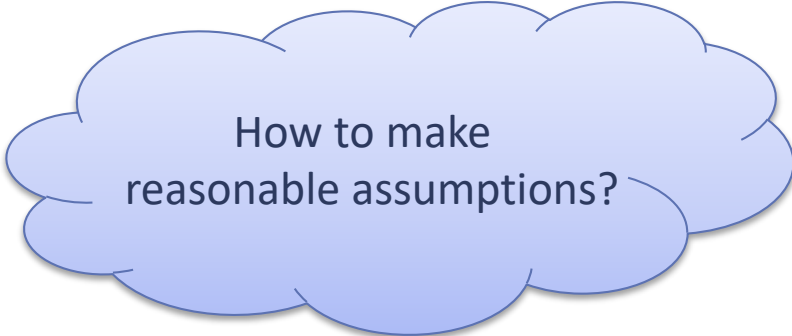
1. Assuming that K is in E[]
2. Assuming no same entries in E[]
3. Each possible input appears with equality (thus, K in the  $i^{th}$  location with probability  $\frac{1}{n}$ )

- $$\begin{aligned} A_{succ}(n) &= \sum_{i=0}^{n-1} \Pr(I_i | succ) t(I_i) \\ &= \sum_{i=0}^{n-1} \frac{1}{n} (i + 1) \\ &= \frac{n+1}{2} \end{aligned}$$



# Average-case Cost of *SeqSearch*

- **Case 2: K may (or may not) be in E[]**
  - Assume that K is in E[] with probability  $q$
  - $A(n) = \Pr(succ) A_{succ}(n) + \Pr(fail) A_{fail}(n)$ 
$$= q \frac{n+1}{2} + (1 - q)n$$



How to make  
reasonable assumptions?

# Algorithm Analysis

- **Advanced topics**
  - Lower bound (Selection)
  - Optimality (Greedy, DP)
  - Computation complexity
  - Approximate / online / randomized algorithms

*Thank you!*

*Q & A*

*Yu Huang*

<http://cs.nju.edu.cn/yuhuang>



# Appendix




也许计算的设  
备会变得天翻  
地覆，但是算  
法终将是算法

— 黄宇 《算法设计  
与分析讲义》





# Appendix


 Question  

Introduction to Algorithms (book)

Computer Programming

## How many people have read all of "Introduction to Algorithms"?

Many people suggest this book when asked about algorithms or programming in general. I have this book and it is BIG. I'm wondering how many people have actually read this cover to cover.

Following 45 Downvote Comment 

Write Answer

8 ANSWERS 44 FOLLOWERS 24,557 VIEWS LOG

 Thomas Cormen  
Professor of Computer Science at Dartmouth ...  
[Upvote](#) • 1.6k upvotes by Barak Cohen, Hilawi W. Belachew, (more)  
I have. And I mean truly cover to cover. I've even read the index.  
Written Fri.

[Upvote](#) 1.6k Downvote Comments 11+ 