

# Design and Analysis of Computer Algorithms

高宏  
计算机科学与技术学院

## Who is Who



- Instructor: 高宏 (海量数据计算研究中心)
- Office: 综合楼809
- Telephone: 13945092842
- Email: honggao@hit.edu.cn
- 研究方向: 海量数据管理与分析、物联网
- 学术兼职: ACM China常务理事  
中国计算机学会(CCF)理事  
CCF数据库专委会 副主任  
黑龙江省计算机学会理事  
数据科学与大数据技术专委会主任

## Who is Who



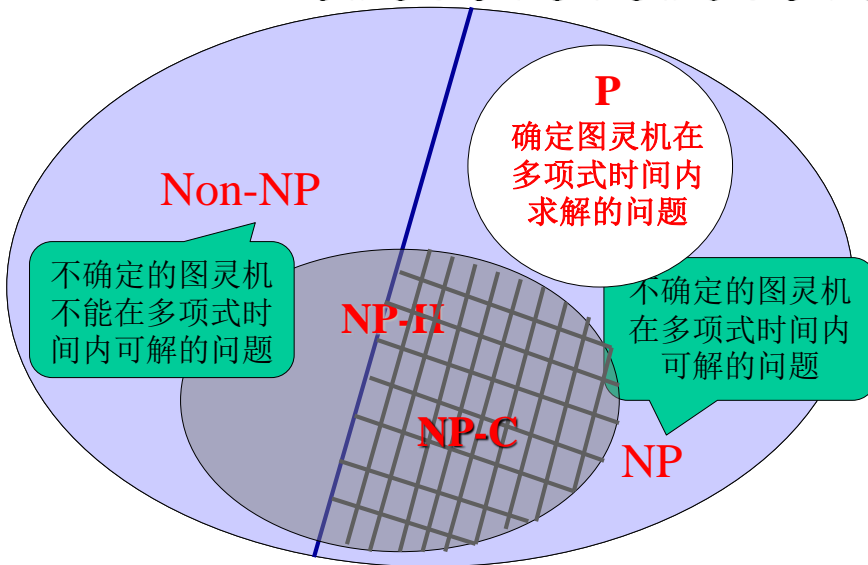
- TA: 郭环宇(18804620324)
- Office: 综合楼, 815房间
- Email: DBLB\_2019algorithm@163.com  
psw: DBLB12356
- QQ: 699932154

## Information about Class



- Class News and Homework
  - Will be posted in QQ
- Homework
  - Must be submitted by emails on time
- Grading
  - Homework and Reading: 40%
  - Final Exam (Written Test): 60%

# What do we Learn?



## Outline of the course

- Overview
- Mathematical Foundations
- Divide-and-Conquer Algorithms
- Dynamic Programming
- Greedy Algorithms
- Searching Strategies
- Amortized Analysis
- Network and Graph Algorithms
- String Matching Algorithms
- Other Algorithms

## References



1. Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*, The MIT Press, Third Edition, 2012.
2. R.C.T.Lee, S.S.Tseng, R.C.Chang, and Y.T.Tsai, *Introduction to the Design and Analysis of Algorithms*, McGraw-Hill, 2007.
3. D. E. Knuth, *Art of the Computer Programming*, Vol. 3, Addison-Wesley, 1973.
4. A.V.Aho, J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.

## Important Conferences



- 计算机各个研究领域的Rank One会议
  - STOC、FOCS
  - SIGMOD/PODS、VLDB、ICDE
  - AAAI、IJCAI、KDD、ICCV
  - SIGCOMM、INFOCOM、MOBICOM、USENIX
  - SIGIR、SIGMETRICS
  - .....

## Important Journals



- 计算机各个研究领域的Rank One期刊
  - Journal of the ACM、Algorithmica
  - ACM TODS、TKDE、VLDB J.
  - IEEE/ACM Transactions on Networking、JSAC、TMC
  - ACM Transactions on Computer Systems
  - IEEE Transactions on Computers
  - IEEE Transactions on Parallel and Distributed Systems
  - IEEE Trans on Pattern Analysis and Machine Intl
  - IEEE Trans on Image Processing
  - .....

## 第一章 绪论



- 1.1 Role of Algorithms in Computer Science**
- 1.2 Concepts of Algorithms**
- 1.3 Designing Algorithms**
- 1.4 Analyzing Algorithms**

## 1.1 Role of Algorithms in Computer Science

- 算法是计算机科学的重要主题
- 影响算法领域的重要学者
- 计算机科学体系及算法设计分析的地位

## 算法是计算机科学的重要主题

- 70年代前

- 计算机科学基础的主题没有被清楚地认清

- 70年代

- Knuth出版了《The Art of Computer Programming》

- 1968, 1969, 1973

- 以算法研究为主线确立了算法为计算机科学基础的重要主题

- 因具有划时代意义的这本书，1974年获得图灵奖

- 70年代后

- 算法作为计算机科学核心推动了计算机科学技术飞速发展

## 影响算法世界的重要学者



美国Texas大学  
计算机和  
数学教授

1972年获得图灵奖  
for ALGOL60编译器

Edsger W. Dijkstra



斯坦福大学教授

获1974年图灵奖 for  
《The Art of Computer  
Programming》  
and algorithm  
design & analysis

Donald Ervin Knuth (高德纳)



Toronto大学教授  
1982年获得图灵奖  
for a famous paper:  
“The Complexity of  
Theorem Proving  
Procedures” 奠定了  
NP-完全理论基础

Stephen Arthur Cook



UC Berkely(“三栖学者”)

1985年获得图灵奖 for  
在算法理论特别是  
NP-completeness理论  
方面连续不断的贡献

Richard M. Karp



斯坦福大学教授

著名公式“算法+数据结构=程序”  
获1984年图灵奖 for  
开发了一系列计算机语言

**Niklaus Wirth**



Cornell University

1986年获得图灵奖  
在算法及数据结构设计和分析方面的基础性成就，

**John Hopcroft**



Cornell University

1986年获得图灵奖  
在算法及数据结构设计和分析方面的基础性成就，

**Robert E. Tarjan**

Cornell University



University at Albany  
(奥尔巴尼)

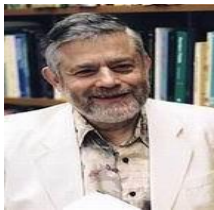


1993年获得图灵奖  
for a the paper:  
“On the Computational Complexity of Algorithms” published in 1965  
奠定了算法计算复杂性的理论基础

**Juris Hartmanis**

**Richard Edwin Stearns**

**计算复杂性理论的主要奠基人**



UC Berkely,  
卡内基梅隆

1995年获得图灵奖  
for在计算复杂性理论及其在密码学方面的应用。

**Manuel Blum**



普林斯顿大学，  
清华大学

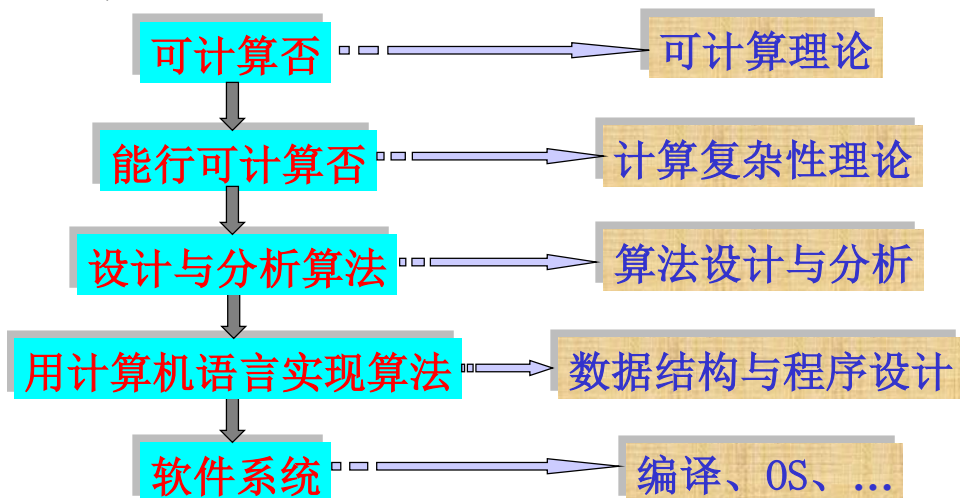
2000年获得图灵奖  
for在计算理论方面的贡献。

**Andrew Yao (姚期智)**



# 计算机科学技术的体系

- 解决一个计算问题的过程



- 可计算理论

- 计算模型

- Turing机、递归函数、 $\lambda$ 演算、正则算法、POST系统

- 可计算问题/不可计算问题

- 计算模型的等价性--图灵/Church命题

- 所有合理的计算模型，其计算能力是等价的

- 计算复杂性理论

- 在给定的计算模型下研究问题的复杂性

- 固有复杂性
  - 复杂性上、下界
  - 平均复杂性
  - 复杂性问题的分类:  $P=NP?$
  - 公理复杂性理论: 抽象复杂性研究

- 算法设计和分析
  - 解决可计算问题的算法的设计与分析
  - 设计算法的理论、方法和技术
  - 分析算法的理论、方法和技术
- 计算机软件
  - 系统软件
  - 工具软件
  - 应用软件

## 1.2 Concepts of Algorithms

- 算法的定义
- 问题的定义
- 算法的实例

## 算法的定义

**定义1(计算)** 可由一个给定**计算模型**  
**机械地执行的规则或计算步骤序列**称为该  
计算模型的一个**计算**。

### - 注意

- 一个计算机程序是一个计算（计算模型是计算机）
- 计算可能永远不停止——不是算法

**定义2(算法)** 算法是一个满足下列条件的**计算**：

- 终止性**：有限步内必须停止（有穷性），
- 确定性**：每步都是严格定义和确定的动作，
- 能行性**：每个动作都能被精确地机械执行，
- 输入**：具有满足给定约束条件的输入，
- 输出**：产生满足给定约束条件的结果。

**算法的目的是求解问题。**  
**什么是问题？**

**定义3 (问题)** 设 $Input$ 和 $Output$ 是两个集合. 一个问题是一个关系 $P \subseteq Input \times Output$ ,  $Input$ 称为问题 $P$ 的输入集合,  $Input$ 的每个元素称为 $P$ 的一个输入,  $Output$ 称为问题 $P$ 的输出或结果集合,  $Output$ 的每个元素称为 $P$ 的一个结果.

### — 注意

- 问题定义了输入和输出的关系

**例.** SORT问题定义如下

输入:  $Input = \{ \langle a_1, \dots, a_n \rangle \mid a_i \text{ 是整数} \}$

输出:  $Output = \{ \langle b_1, \dots, b_n \rangle \mid b_i \text{ 是整数, } b_1 \leq \dots \leq b_n \}$

问题:  $SORT = \{ (\langle a_1, \dots, a_n \rangle, \langle b_1, \dots, b_n \rangle) \mid$

$\langle a_1, \dots, a_n \rangle \in Input,$

$\langle b_1, \dots, b_n \rangle \in Output,$

$\{a_1, \dots, a_n\} = \{b_1, \dots, b_n\} \}$

$SORT = \{ (\langle 3, 2, 1, 5, 4, 6 \rangle, \langle 1, 2, 3, 4, 5, 6 \rangle), (\langle 6, 8, 4 \rangle, \langle 4, 6, 8 \rangle),$   
 $(\langle 9, 6, 7, 3 \rangle, \langle 3, 6, 7, 9 \rangle), \dots \}$

$SORT = \{(\langle 3, 2, 1, 5, 4, 6 \rangle, \langle 1, 2, 3, 4, 5, 6 \rangle), (\langle 6, 8, 4 \rangle, \langle 4, 6, 8 \rangle),$   
 $(\langle 9, 6, 7, 3 \rangle, \langle 3, 6, 7, 9 \rangle), \dots\}$

定义4(问题实例). 问题 $P$ 的一个实例是  
 $P$ 的一个二元组.

二元组 $(\langle 9, 6, 7, 3 \rangle, \langle 3, 6, 7, 9 \rangle)$ 是SORT问题的一个实例

#### — 注意

- 问题是一个二元组集合，问题实例是一个二元组.
- 一个算法求解一个完整的问题，而不是仅求解一个问题中的一个或几个实例.

## 算法示例



### • 问题定义

- $Input = \{\langle a_1, \dots, a_n \rangle \mid a_i \text{ 是整数}\}$
- $output = \{\langle b_1, \dots, b_n \rangle \mid b_i \text{ 是整数, 且 } b_1 \leq \dots \leq b_n\}$
- $P = \{(\langle a_1, \dots, a_n \rangle, \langle b_1, \dots, b_n \rangle) \mid \langle a_1, \dots, a_n \rangle \in Input,$   
 $\langle b_1, \dots, b_n \rangle \in output, \{a_1, \dots, a_n\} = \{b_1, \dots, b_n\}\}$

### • 算法的思想

- 扑克牌游戏

抓牌过程:



- 算法描述

Insertion-sort(A)

Input:  $A[1, \dots, n]$  =  $n$  个数

output:  $A[1, \dots, n]$  =  $n$  个sorted数

FOR  $j=2$  To  $n$  Do

$key \leftarrow A[j]$ ;

    //将 $A[j]$ 插入到已排序的 $A[1 \dots j-1]$ 中

$i \leftarrow j-1$

        WHILE  $i > 0$  AND  $A[i] > key$  Do

$A[i+1] \leftarrow A[i]$ ;

$i \leftarrow i-1$ ;

$A[i+1] \leftarrow key$ ;

$$\begin{array}{ccccccc}
 & & & j & & & \\
 \hline
 2 & 4 & 5 & 6 & 1 & 3 & 8 \\
 \hline
 & & & i & & & 
 \end{array}$$

Key= $A[4]=6$

## 1.3 Analyzing Algorithms

- 算法的正确性分析
- 算法的复杂性分析

### 算法的正确性分析

**定义5 (算法正确性).** 一个算法是正确的, 如果它对于每一个输入都**最终停止**, 而且**产生正确的输出**.

#### — 什么算法是不正确算法?

- ① 在一个或多个输入上不停止
- ② 对所有输入都停止, 但对一个或多个输入产生不正确结果

#### — 什么是近似算法/随机算法的正确性?

- ① 对所有输入都停止
- ② 产生近似正确的解/产生不多的不正确解

- 为什么要进行正确性证明?
  - 调试程序=程序正确性证明?  
程序调试只能证明程序有错!  
不能证明程序无错误!!
- 如何证明算法的正确性?
  - 证明算法对所有输入都停止
  - 证明对每个输入都产生正确结果

## 算法的复杂性分析



- 目的
  - 分析算法对不同输入所需资源量
- 复杂性测度:
  - 时间、空间、I/O等
  - 是输入大小的函数
- 用途:
  - 为求解一个问题选择最佳算法、最佳设备
- 需要的数学基础
  - 离散数学、组合数学、概率论、代数等
- 需要的数学能力
  - 建立算法复杂性的数学模型
  - 数学模型化简



定义6 (输入的大小). 设 $Input$ 是问题 $P$ 的输入集合,  $P$ 的输入大小是一个函数  $F: Input \rightarrow N$ ,  $N$ 是正整数集合.

–  $size(\alpha)$ 表示输入 $\alpha \in Input$ 的大小

– 示例:

- 排序问题的输入大小?
- 矩阵问题的输入大小?
- 图论问题的输入大小?

定义7 (实例时间复杂性). 一个算法对特定输入的时间复杂性是该算法对该输入产生结果需要的原子操作数.

– 注意

- 时间复杂性是输入大小的函数 $T(n)$
- 我们假设每一步的执行需要常数时间, 实际上每步需要的时间量可能不同.

定义8 (实例空间复杂性). 一个算法对特定输入的空间复杂性是该算法对该输入产生结果所需要的存储空间大小.

— 注意

- 空间复杂性是输入大小的函数  $S(n)$

定义9 (算法最坏复杂性). 设  $Input$  是问题  $P$  的输入集合,  $Complexity(Size(x))$  是求解  $P$  的实例  $x$  的算法  $A$  的复杂性函数,  $A$  的最坏复杂性是

$$\text{Max}\{Complexity(size(x)) \mid x \in Input\}$$

定义10 (算法最小复杂性).

$$\text{Min}\{Complexity(size(x)) \mid x \in Input\}$$

定义11 (算法平均复杂性). 设  $x \in Input$ ,  $x$  作为算法  $A$  的输入出现的概率是  $p_x$ ,  $A$  的平均复杂性为

$$\sum_{x \in Input} p_x \times \text{complexity}(\text{size}(x))$$

## 算法分析模型

- 随机访问模型 (Random-Access-Model, RAM)
  - 单处理机, 串行执行, 无并发
  - 基本数据类型: integer、floating point
  - 基本操作(每个操作常数时间)
- 并行多处理机模型(PRAM)

## 算法复杂性分析

- 算法执行时间的计算
  - 所有原子操作执行时间的和
  - 每个原子操作假设为常数时间

例如: 计算 $n$ 个数的平均值

$S=0;$	$c_1$	} $n(c_2+c_3) + c_1+c_4+c_5$
for $i=1$ to $n$		
输入 $a$ 值;	$c_2$	
$S=S+a;$	$c_3$	
$S=S/n;$	$c_4$	
输出 $S$ 值;	$c_5$	

## 算法复杂性分析



### • 直接插入排序算法的复杂性分析

输入序列：1, 2, 3, 4, 5, 6

具有最小时间复杂性：最少比较次数  
为： $n$

输入序列：6, 5, 4, 3, 2, 1

具有最坏时间复杂性：最多比较次数

$$\sum_{k=1}^n k = \frac{n(n+1)}{2}$$

## 算法复杂性分析



### • 直接插入排序算法的复杂性分析

输入序列：5, 2, 4, 6, 1, 3

排序过程：

5

2, 5

2, 4, 5

2, 4, 5, 6

1, 2, 4, 5, 6

1, 2, 3, 4, 5, 6

当读入第 $k$ 个数 $x_k$ 时：

(1). 将 $x_k$ 与前面的数比较

(2). 若 $x_k$ 为第 $j$ 大, 将 $x_k$ 插入第 $k-j+1$ 的位置

$x_k$ 为第 $j$ 大的概率为 $1/k$ .

$x_k$ 插入后形成有序序列而需要的比较次数：

$$\frac{1}{K} \sum_{j=1}^K (K-j+1) = \frac{1}{K} \left[ K^2 - \frac{K(K+1)}{2} + K \right] = \frac{K+1}{2}$$

总的平均  
比较次数：

$$\sum_{k=2}^N \left( \frac{K+1}{2} \right) = \sum_{k=1}^{N-1} \left( \frac{K+2}{2} \right) = \frac{1}{2} \left[ \frac{(N+1)(N+2)}{2} \right] = \frac{1}{4} N^2 + \frac{3}{4} N - 1$$

## 1.4 Designing Algorithms

- 算法的设计方法
- 算法的分析方法

### 算法设计方法

- Divide-and-Conquer
- Dynamic Programming
- Greedy Algorithms
- Tree Searching Strategies
- Genetic Algorithms
- Annealing
- Approximation Algorithms
- Randomized Algorithms
- On-Line Algorithms
- Parellel Algorithms

## 算法的分析方法



- 不同的设计方法有不同的分析方法