

算法设计与分析

第一章 绪论

哈尔滨工业大学（深圳）

李穆

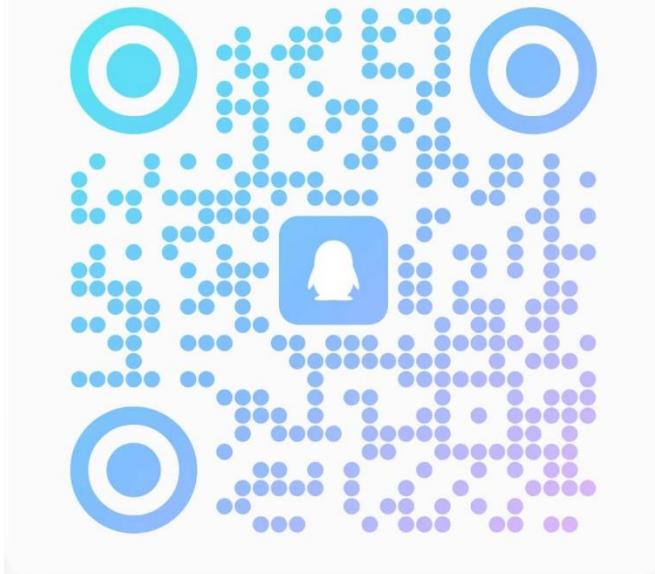
limu2022@hit.edu.cn

课程信息

- 授课教师: 李穆, 助理教授
- 办公室: 信息楼1512
- 邮箱: limu2022@hit.edu.cn
- 助教: 程裕龙, 周天鸣
- 课程QQ群: 扫码加入



2023秋-算法设计与...
群号: 893100723



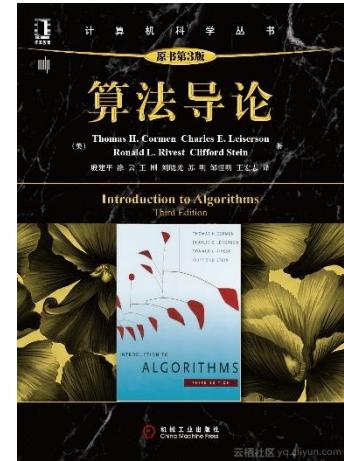
以班级-姓名（如：5班-xxx）格式实名

课程信息

课程教材：《算法导论》第三版，殷建平，徐云，王刚等译，机械工业出版社，2012.12.

学习资源：<https://leetcode-cn.com/problemset/all/>

<http://oi-wiki.com/contest/roadmap/>



课程考核方法

考核环节	所占分值	考核与评价细则
平时作业及考核	40	学生们上课考勤，对提问的回答等，占5分；课程测试与作业，每次单独评分，最后合计成绩占35分。
期末考试	60	卷面成绩100分，以卷面成绩按比例折算成实际得分；考试命题以大纲中的应知应会内容为主，并保证逐年有所变化。

数学中的算法

$$\begin{cases} x_1+x_2+x_3=1 \\ 8x_1+3x_2+10x_3=15 \\ 13x_1+5x_2+6x_3=-2 \end{cases}$$
$$1 \frac{b-c^3 \cdot \cos a}{a}$$
$$\frac{\Delta f}{\Delta x} \frac{5x^2+6y^3-1}{(1+x+y+z)^2}$$

$$\int f(x,y,z) dz$$
$$x \ln x \operatorname{Sa}$$
$$\iiint \frac{dxdydz}{(1+x+y+z)^2}$$
$$\sqrt{2} \cdot \sin 2x$$
$$5x^2 + 14xy + 2y^2 = -18$$
$$A \cos x \cdot \arctan y$$
$$2 \sin^3 52^\circ = 1$$

designed by freepik

生活中的算法

衣



食



大众点评
dianping.com

住



行



生活中的算法



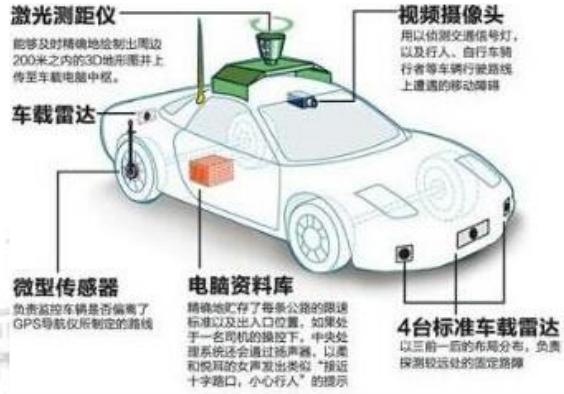
人脸识别



人机对战



视频监控



图像恢复

无人驾驶

车辆识别

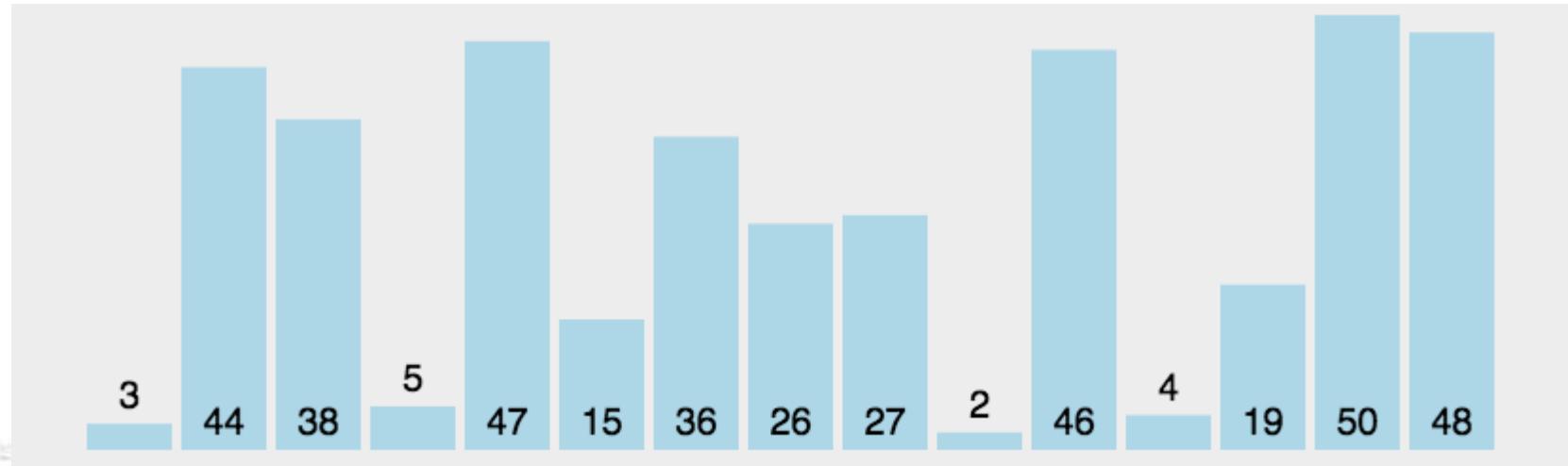
课程中的算法

请输入行数:8

```
*  
***  
*****  
*****  
*****  
*****  
*****  
*****  
*****
```

| 原码，反码，补码和移码

	1	-1
原码	0000 0001	1000 0001
反码	0000 0001	1111 1110
补码	0000 0001	1111 1111
移码	1000 0001	0111 1111



本讲内容

1.1 什么是算法？

1.2 计算机科学中算法的位置

1.3 算法分析引论

1.4 算法设计引论

什么是算法？

- 在数学和计算机科学之中，算法/算则法（Algorithm）为一个计算的具体步骤，常用于计算、数据处理和自动推理。（Wikipedia）



计算的定义

- 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。

算法的定义

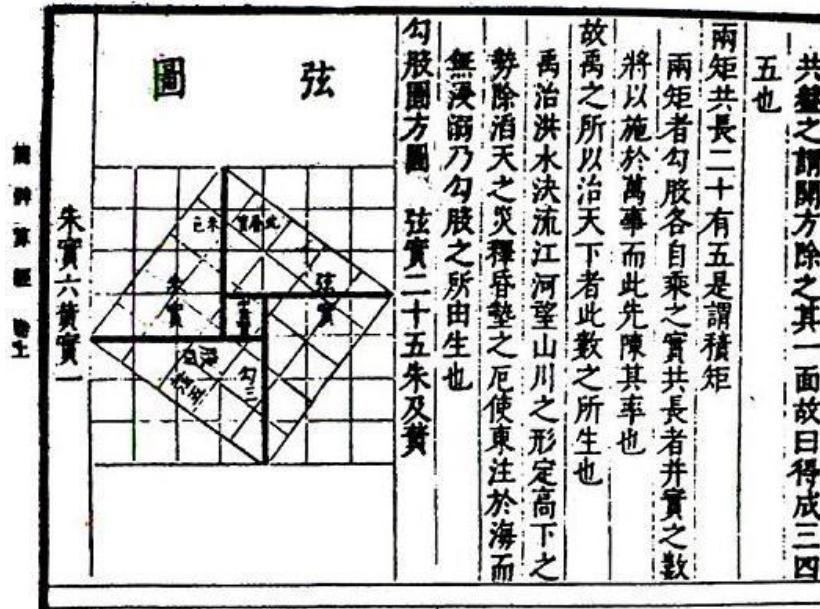
算法是一个满足下列条件的计算：

- 有穷性/终止性：有限步内必须停止，
- 确定性：每一步都是严格定义和确定的动作，
- 能行性：每一个动作都能够被精确地机械执行，
- 输入：有一个满足给定约束条件的输入，
- 输出：满足给定约束条件的结果。

算法的由来

➤ 名称由来

- 中文名称“算法”出自约成书于公元前1世纪的《周髀算经》



介绍了勾股定理：“以日下为勾，日高为股，勾股各自乘，并而开方除之”

算法的由来

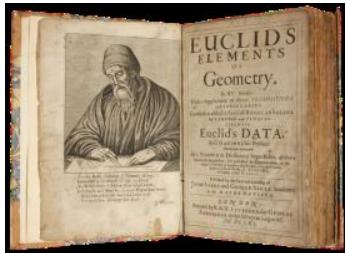
➤ 名称由来

- 波斯著名的数学家、天文学家、地理学家阿尔·花拉子米 (Al-Khwarizmi) 在**公元825年**写成《印度数字算术》一书，对于印度-阿拉伯数字系统在中东及欧洲的传播起到了重要作用
- 该书被翻译成拉丁语“Algoritmi de numero Indorum”，花拉子米的拉丁文音译即为“算法” (Algorithm) 一词的由来



苏联在**1983**年发行邮票纪念花拉子米**1200**岁生辰

算法的由来



公元前300年
《几何原本》辗转相除法

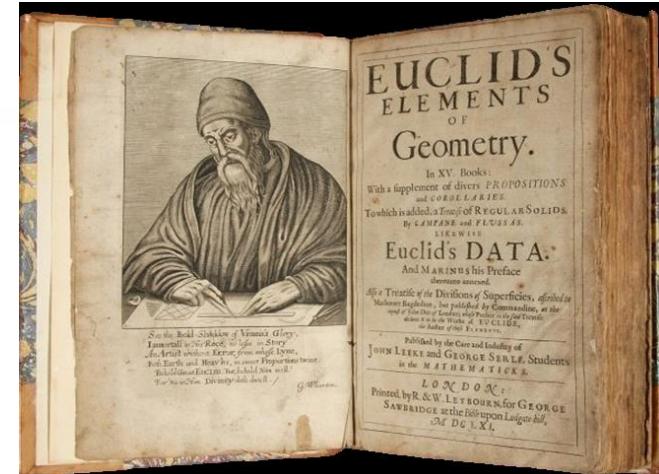


算法的由来

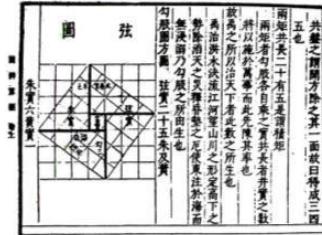
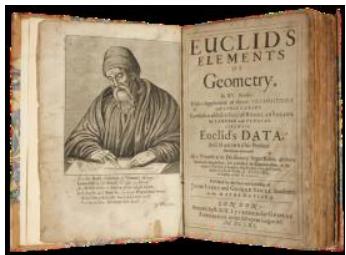
➤ 辗转相除法

- 约公元前300年由欧几里德提出
- 用于计算两个整数的最大公约数
- 定理：两个整数的最大公约数等于其中较小的那个数和两数相除余数的最大公约数
- //注意：这里不用考虑m和n的大小问题。

```
int euclid(int m, int n)
{
    int r;
    do{  r = m % n;
        m = n;
        n = r;
    } while(r!=0);
    return m;
}
```



算法的由来

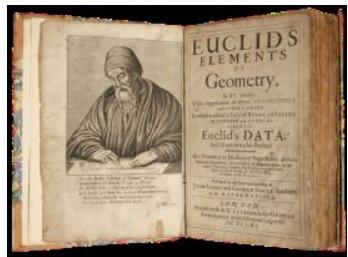


公元前300年
《几何原本》辗转相除法

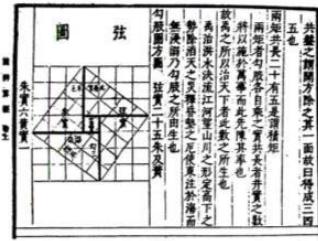
公元前1世纪
算法《周髀算经》



算法的由来



公元前300年
《几何原本》辗转相除法



公元前1世纪
算法《周髀算经》



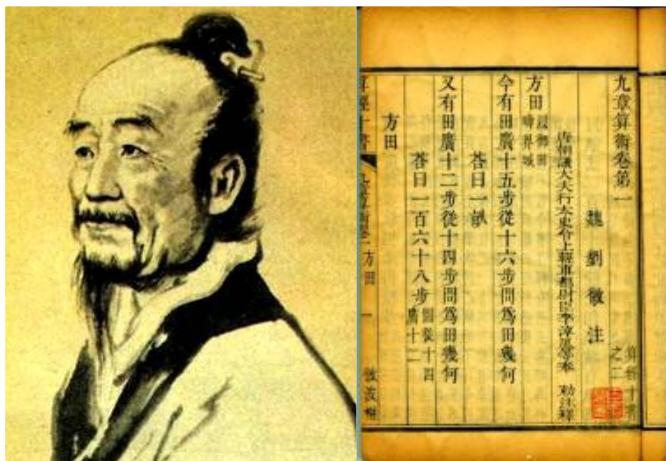
魏晋时期
割圆术《九章算术注》



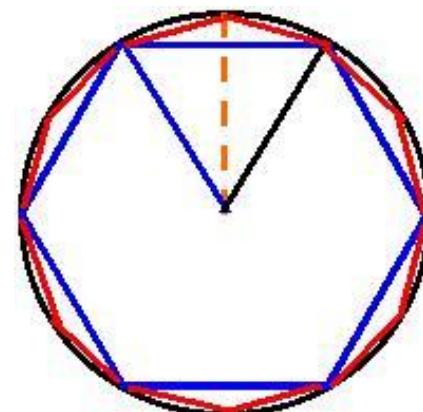
算法的由来

➤ 割圆术

- 内接正多边形去无限逼近圆，并以此求取圆周率的方法
- 魏晋时期的数学家刘徽在《九章算术注》中首创

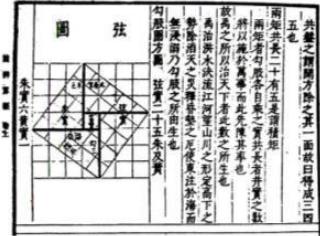
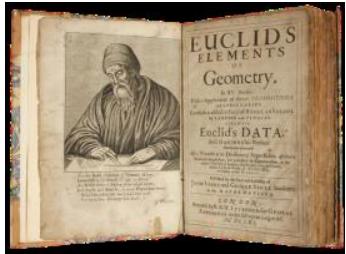


刘徽与《九章算术注》



割圆术示意图

算法的由来



公元前300年
《几何原本》辗转相除法

公元前1世纪
算法《周髀算经》

魏晋时期
割圆术《九章算术注》

7世纪
《算经十书》



算法的由来

➤ 算经十书

- 唐高宗显庆元年（公元656年），规定将十部汉、唐一千多年间的十部著名数学著作作为国家最高学府的算学教科书，用以进行数学教育和考试，后世通称为《算经十书》



唐高宗李治



周髀算经



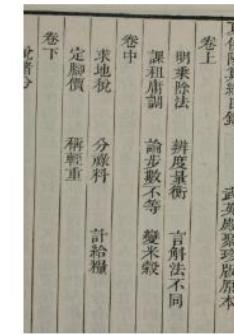
九章算术



海岛算经



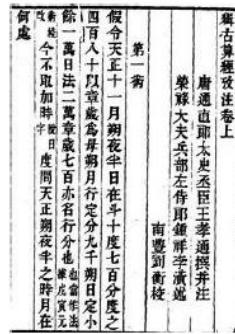
张丘建算经



夏侯阳算经



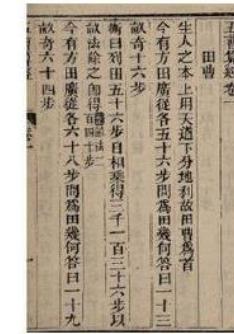
五经算术



缀术



五曹算经



孙子算经

夏侯阳算经目錄
卷上
明乘除法
辨度量衡
論步數不等
課租調
定腳價
求地稅
分祿符
糴米穀

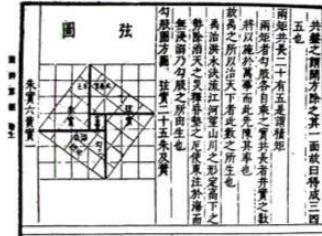
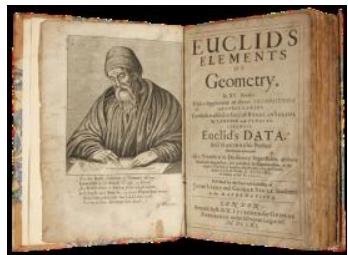
武英殿聚珍版原本
卷中
計船糧
稱輕重
卷下
丈尺分

孫子算經卷上
唐高宗顯慶元年正月廿二日
夏侯陽著於京師
卷一
人所用田地分利故用苗爲首

引五十尺爲一塊四十尺爲一疋六十尺爲一
步一百四十尺爲一畝三百步爲一里
一畝爲十步爲一頃三十畝爲一鄰二
十四鄰爲一兩十六兩爲一斤三十斤爲一鈞

度所起於勿訛知勿訛知勿訛知勿訛
爲一絲十絲爲一毫四十毫爲一釐七十釐爲一分
一百四十釐爲一毫三十毫爲一厘六厘爲一
步一百四十步爲一畝三十步爲一里
一畝爲十步爲一頃三十畝爲一鄰二
十四鄰爲一兩十六兩爲一斤三十斤爲一鈞

算法的由来



公元前300年
《几何原本》辗转相除法

公元前1世纪
算法《周髀算经》

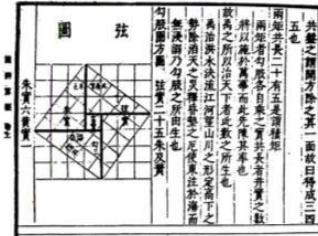
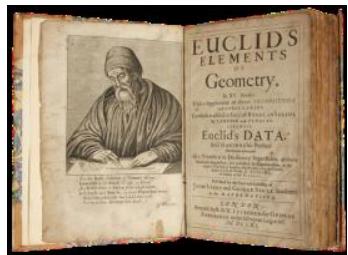
魏晋时期
割圆术《九章算术注》

7世纪
《算经十书》

Algorithm
9世纪 花拉子米



算法的由来



公元前300年
《几何原本》辗转相除法

公元前1世纪
算法《周髀算经》

魏晋时期
割圆术《九章算术注》

7世纪
《算经十书》

Algorithm
9世纪 花拉子米

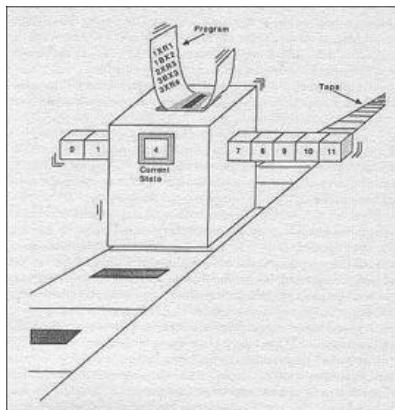
20世纪30年代~40年代
现代计算机萌芽



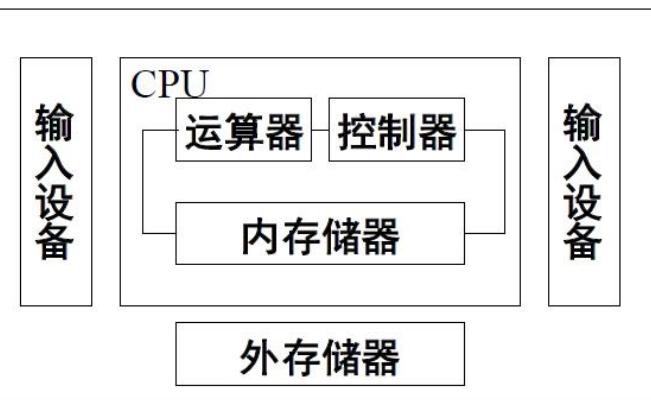
算法的由来

➤ 算法与计算机的结合

- 1936年，艾伦·图灵提出图灵机，通过建立通用计算机模型，刻画计算机的计算行为
- 1946年，冯·诺依曼提出存储程序原理



理论计算机科学与人工智能之父
艾伦·图灵
Alan Turing



现代计算机之父
约翰·冯·诺伊曼
John von Neumann

图灵奖

- 1966年由计算机协会(Association for Computing Machinery, ACM)设立，奖励对计算机事业做出突出贡献的个人



艾伦·图灵
Alan Turing



计算机界的“诺贝尔奖”

问题的定义

- 算法的目的是求解问题。什么是问题？
- 问题
 - 设Input和Output是两个集合。一个问题是一个关系 $R \subseteq Input \times Output$ ， Input称为问题R的输入集合， Input的每个元素称为R的一个输入， Output称为问题R的输出或结果集合， Output的每个元素称为R的一个结果。
 - 注意
 - 问题定义了输入和输出的关系。

问题的例子

SORT问题定义如下：

- 输入集合 $\text{Input} = \{\langle a_1, \dots, a_n \rangle \mid a_i \text{是整数}\}$
- 输出集合 $\text{Output} = \{\langle b_1, \dots, b_n \rangle \mid b_i \text{是整数}, b_1 \leq \dots \leq b_n\}$
- 问题 $\text{SORT} = \{(\langle a_1, \dots, a_n \rangle, \langle b_1, \dots, b_n \rangle) \mid \langle a_1, \dots, a_n \rangle \in \text{Input}, \langle b_1, \dots, b_n \rangle \in \text{Output}, \{a_1, \dots, a_n\} = \{b_1, \dots, b_n\}\}$

• 问题实例

-问题P的一个实例是P的一个二元组。

-注意

- 一个算法面向一个问题，而不是仅求解一个问题的一个或几个实例。

算法示例

- 问题定义

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

- 算法的思想—扑克牌游戏

算法演示

$A[1, \dots, n] = 5, 2, 4, 6, 1, 3$

$A[1, \dots, n] = 5, 2, 4, 6, 1, 3$

$A[1, \dots, n] = 2, 5, 4, 6, 1, 3$

$A[1, \dots, n] = 2, 4, 5, 6, 1, 3$

$A[1, \dots, n] = 2, 4, 5, 6, 1, 3$

$A[1, \dots, n] = 1, 2, 4, 5, 6, 3$

$A[1, \dots, n] = 1, 2, 3, 4, 5, 6$

算法描述

Insertion-sort(A)

Input: A[1, , n]=n个数

output: A[1, , n]=n个sorted数

FOR j=2 To n Do

 key←A[j];

 i←j-1

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

 A[i+1]←A[i];

 i←i-1;

 A[i+1]←key;

- 实例: A[1, , n]=5, 2, 4, 6, 1, 3

本讲内容

1.1 什么是算法？

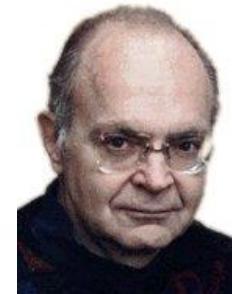
1.2 计算机科学中算法的位置

1.3 算法分析引论

1.4 算法设计引论

算法是计算机科学基础的重要主题

- 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》
 - 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
 - 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



算法与计算复杂性领域图灵奖得主



Donald E. Knuth
1974, USA
算法分析之父



Michael O. Rabin
1976, Israeli
非确定自动机
素数判定随机算法



Dana S. Scott
1976, USA
非确定自动机



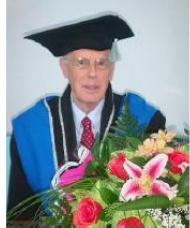
Robert W. Floyd
1978, USA
最短路径Floyd算法



Stephen A. Cook
1982, USA
NP完全性



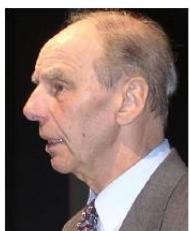
Richard M. Karp
1985, USA
NP完全性与
网络流算法



John Hopcroft
1986, USA
最差情况分析
数据结构与算法



Robert Tarjan
1986, USA
数据结构与图算法



Juris Hartmanis
1993, Latvia
计算复杂性理论



Richard E. Stearns
1993, USA
计算复杂性理论



Manuel Blum
1995, Venezuela
计算复杂性理论



Andrew Yao
2000, China
伪随机数生成
与通信复杂性



Leslie G. Valiant
2010, Hungarian
#P完全性与
计算学习理论

学术研究中的算法

Algorithm 1 Dataset Distillation

Input: $p(\theta_0)$: distribution of initial weights; M : the number of distilled data
Input: α : step size; n : batch size; T : the number of optimization iterations; $\tilde{\eta}_0$: initial value for $\tilde{\eta}$

- 1: Initialize $\tilde{\mathbf{x}} = \{\tilde{x}_i\}_{i=1}^M$ randomly, $\tilde{\eta} \leftarrow \tilde{\eta}_0$
- 2: **for each** training step $t = 1$ to T **do**
- 3: Get a minibatch of real training data $\mathbf{x}_t = \{x_{t,j}\}_{j=1}^n$
- 4: Sample a batch of initial weights $\theta_0^{(j)} \sim p(\theta_0)$
- 5: **for each** sampled $\theta_0^{(j)}$ **do**
- 6: Compute updated parameter with GD: $\theta_1^{(j)} = \theta_0^{(j)} - \tilde{\eta} \nabla_{\theta_0^{(j)}} \ell(\tilde{\mathbf{x}}, \theta_0^{(j)})$
- 7: Evaluate the objective function on real training data: $\mathcal{L}^{(j)} = \ell(\mathbf{x}_t, \theta_1^{(j)})$
- 8: **end for**
- 9: Update $\tilde{\mathbf{x}} \leftarrow \tilde{\mathbf{x}} - \alpha \nabla_{\tilde{\mathbf{x}}} \sum_j \mathcal{L}^{(j)}$, and $\tilde{\eta} \leftarrow \tilde{\eta} - \alpha \nabla_{\tilde{\eta}} \sum_j \mathcal{L}^{(j)}$
- 10: **end for**

Output: distilled data $\tilde{\mathbf{x}}$ and optimized learning rate $\tilde{\eta}$

Algorithm 1 Greedy Estimation of the Pseudocylindrical Representation Parameters

Input: ERP image set $\mathcal{D} = \{\mathbf{x}^{(0)}, \dots, \mathbf{x}^{(|\mathcal{D}|-1)}\}$, and the quantized width set $\{\bar{W}_0, \dots, \bar{W}_{L-1}\}$.
Output: The optimized parameter set $\{W_t^*\}$.

- 1: **for** $t \leftarrow 0$ to $T - 1$ **do**
- 2: $W_t^* \leftarrow \bar{W}_{L-1}$; ▷ Initialization
- 3: **end for**
- 4: **for** $t \leftarrow 0$ to T_{half} **do**
- 5: $V_{\text{best}} \leftarrow \infty$, $W_{\text{best}} \leftarrow 0$;
- 6: **if** $t = 0$ **then**
- 7: start_width $\leftarrow \bar{W}_0$;
- 8: **else**
- 9: start_width $\leftarrow W_{t-1}^*$;
- 10: **end if**
- 11: **for** $W_t^* \leftarrow \text{start_width}$ to \bar{W}_{L-1} **do**
- 12: $W_{T-t-1}^* \leftarrow W_t^*$;
- 13: $V_{\text{temp}} \leftarrow \mathbb{E}_{\mathbf{x} \in \mathcal{D}} \text{RD}(\mathbf{x}, \text{compress}_{\alpha}(\mathbf{x}); \{W_t^*\})$;
- 14: **if** $V_{\text{temp}} < V_{\text{best}}$ **then**
- 15: $V_{\text{best}} \leftarrow V_{\text{temp}}$, $W_{\text{best}} \leftarrow W_t$;
- 16: **end if**
- 17: **end for**
- 18: $W_t^* \leftarrow W_{\text{best}}$, $W_{T-t-1}^* \leftarrow W_{\text{best}}$;
- 19: **end for**

Algorithm 1 ∞ -AE model training

Input: User set \mathcal{U} ; dataset $\mathbf{X} \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{I}|}$; NTK $\mathbb{K} : \mathbb{R}^{|\mathcal{I}|} \times \mathbb{R}^{|\mathcal{I}|} \mapsto \mathbb{R}$; regularization const. $\lambda \in \mathbb{R}$
Output: Dual parameters $\alpha \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{I}|}$

- 1: **procedure** FIT($\mathcal{U}, \mathbf{X}, \mathbb{K}$)
- 2: $\mathbf{K} \leftarrow [0]_{|\mathcal{U}| \times |\mathcal{U}|}$ ▷ Zero Initialization
- 3: $\mathbf{K}_{u,v} \leftarrow \mathbb{K}(\mathbf{X}_u, \mathbf{X}_v) \quad \forall u \in \mathcal{U}, v \in \mathcal{U}$
- 4: $\alpha \leftarrow (\mathbf{K} + \lambda I)^{-1} \cdot \mathbf{X}$
- 5: **return** α

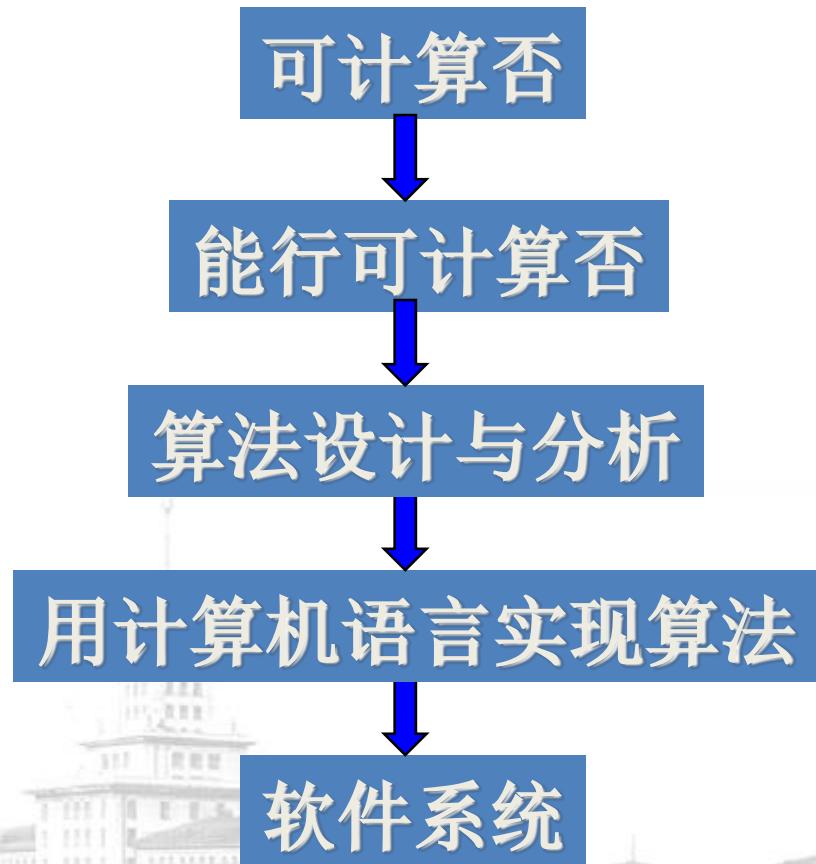
Algorithm 2 ∞ -AE inference

Input: User set \mathcal{U} ; dataset $\mathbf{X} \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{I}|}$; NTK $\mathbb{K} : \mathbb{R}^{|\mathcal{I}|} \times \mathbb{R}^{|\mathcal{I}|} \mapsto \mathbb{R}$; dual params. $\alpha \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{I}|}$; inference user history $\hat{\mathbf{X}}_u \in \mathbb{R}^{|\mathcal{I}|}$
Output: Prediction $\hat{y} \in \mathbb{R}^{|\mathcal{I}|}$

- 1: **procedure** PREDICT($\mathcal{U}, \mathbf{X}, \hat{\mathbf{X}}_u, \mathbb{K}, \alpha$)
- 2: $\mathbf{K} \leftarrow [0]_{|\mathcal{U}|}$ ▷ Zero Initialization
- 3: $\mathbf{K}_v \leftarrow \mathbb{K}(\hat{\mathbf{X}}_u, \mathbf{X}_v) \quad \forall v \in \mathcal{U}$
- 4: $\hat{y} \leftarrow \text{softmax}(\mathbf{K} \cdot \alpha)$
- 5: **return** \hat{y}

计算机科学的体系

- 解决一个计算问题的过程



- 可计算理论
 - 计算模型
 - 可计算问题/不可计算问题
 - 计算模型的等价性--图灵/Church命题
- 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
 - 上界
 - 下界
 - 平均
 - 复杂性问题的分类: $P=NP?$
 - 抽象复杂性研究

- 算法设计和分析
 - 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



本讲内容

1.1 什么是算法？

1.2 计算机科学中算法的位置

1.3 算法分析引论

1.4 算法设计引论



算法的正确性分析

- 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。
 - 不正确算法：
 - ①不停止(在某个输入上)
 - ②对所有输入都停止，但对某输入产生不正确结果
 - 近似算法
 - ①对所有输入都停止
 - ②产生近似正确的解或产生不多的不正确解

- 算法正确性证明
 - 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
 - 调试程序 \neq 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！



插入排序的正确性

- 循环不变量
 - 在每次循环的开始，子数组 $A[1..j-1]$ 包含原来数组中 $A[1..j-1]$ 但是已经有序
- 证明
 - 初始化 : $j=2, A[1..j-1]=A[1..1]=A[1]$, 已经有序.
 - 维护：每一层循环维护循环不变量.
 - 终止: $j=n+1, \text{ so } A[1..j-1]=A[1..n]$ 有序.



算法的复杂性分析

- 目的：
 - 预测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

算法复杂性分析的度量

- 输入的大小
 - 设Input是问题R的输入集合， R的输入大小是一个函数 $F: \text{Input} \rightarrow \mathbb{N}$ ， \mathbb{N} 是正整数集合。

示例：

- 矩阵问题的输入大小=矩阵的维数
- 图论问题的输入大小=图的边数/结点数

算法复杂性分析的度量

- **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或“步”数
 - 注意
 - 时间复杂性是输入大小的函数
 - 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。

算法复杂性分析的度量

- 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。

算法复杂性分析的度量

- **最坏复杂性**

- 设Input是问题R的输入集合，Complexity(X)是求解R的算法A的复杂性函数，Size(y)是确定R中输入大小的函数，A的最坏复杂性是

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

- **最小复杂性**

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

- **平均复杂性**

- 设 $y \in \text{Input}$, y 作为算法A的输入出现的概率是 p_y , A的平均复杂性为

$$\sum_{y \in \text{Input}} p_y \times \text{Complexity}(\text{size}(y))$$

算法分析的模型

- 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
 - 基本操作(每个操作常数时间)
- 并行多处理机模型(*PRAM*)

插入排序的分析

Insertion-sort(A)

Input: A[1,⋯,n] = n个数

Output: A[1,⋯,n] = n个sorted数

```
1. For j = 2 To n Do  
2.   key ← A[j];  
3.   %Insert A[j] into the sorted sequence A[1,⋯,j-1]%  
4.   i ← j-1;  
5.   While i > 0 and A[i] > key Do  
6.     A[i+1] ← A[i];  
7.     i ← i-1;  
8.   A[i+1] ← key;
```

代价	次数
C_1	n
C_2	$n - 1$
0	$n - 1$
C_4	$n - 1$
C_5	$\sum_{j=2}^n t_j$
C_6	$\sum_{j=2}^n (t_j - 1)$
C_7	$\sum_{j=2}^n (t_j - 1)$
C_8	$n - 1$

t_j 表示对下标j第5行执行while循环测试的次数

总时间代价 $T(n) = \text{代价} * \text{次数之和}$

$$= C_1 n + C_2(n - 1) + C_4(n - 1) + C_5 \sum_{j=2}^n t_j + C_6 \sum_{j=2}^n (t_j - 1) + C_7 \sum_{j=2}^n (t_j - 1) + C_8(n - 1)$$

插入排序的分析(续)

- 最好代价: 有序的数组
 - $t_j=1$, 且6和7行执行0次
 - $T(n) = c_1n + c_2(n-1) + c_4(n-1) + c_5(n-1) + c_8(n-1)$
 $= (c_1 + c_2 + c_4 + c_5 + c_8)n - (c_2 + c_4 + c_5 + c_8) = cn + c'$
- 最坏代价: 逆序数组
 - $t_j=j$,
 - $\sum_{j=2}^n t_j = \sum_{j=2}^n j = n(n+1)/2-1$, 且 $\sum_{j=2}^n (t_j-1) = \sum_{j=2}^n (j-1) = n(n-1)/2$
 - $T(n) = c_1n + c_2(n-1) + c_4(n-1) + c_5(n(n+1)/2 - 1) + c_6(n(n-1)/2) + c_7(n(n-1)/2) + c_8(n-1) = ((c_5 + c_6 + c_7)/2)n^2 + (c_1 + c_2 + c_4 + c_5/2 - c_6/2 - c_7/2 + c_8)n - (c_2 + c_4 + c_5 + c_8) = an^2 + bn + c$
- 平均代价: 随机数
 - 平均来看, $t_j = j/2$. $T(n)$ 与 n^2 同阶, 和最坏情况相同.

排序 (续)

- 插入排序
 - 插入排序：总时间代价 $T(n)$ 与 n^2 同阶
- 有没有更快速的排序算法
 - 归并排序：总时间代价 $T(n)$ 与 $n * \lg n$ 同阶
 - 快速排序：总时间代价 $T(n)$ 与 $n * \lg n$ 同阶



本讲内容

1.1 什么是算法？

1.2 计算机科学中算法的位置

1.3 算法分析引论

1.4 算法设计引论

算法设计模式

- 暴力搜索
- 分治法
- 图搜索与枚举
 - 分支界限
 - 回溯
- 随机化方法



算法实现方法

- 递归与迭代
- 顺序、并行与分布式
- 确定性与非确定性
- 近似求解与精确求解
- 量子算法



最优化算法设计方法

- 线性规划
- 动态规划
- 贪心法
- 启发式方法

