

数据分析与算法设计

第1章 绪论

李旻

百人计划研究员

浙江大学 信息与电子工程学院

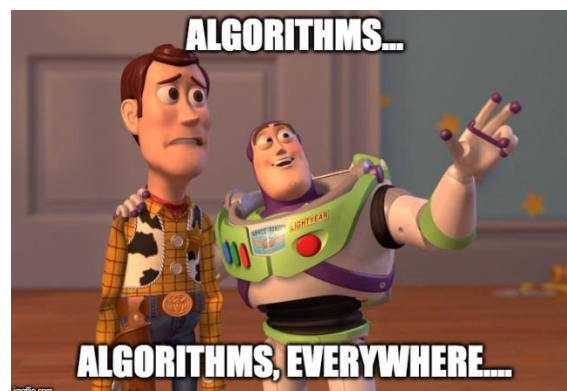
Email: min.li@zju.edu.cn

目录

- 课程简介
 - 课程内容
 - 教学安排
 - 课程考核
- 绪论
 - 什么是算法？
 - 算法问题求解基础
 - 重要的问题类型
 - 基本数据结构

课程简介

- 数据分析与算法设计
 - 数据结构 + 算法 + 数值分析
 - 覆盖的重要问题类型
 - 排序、查找、字符串处理
 - 图问题、组合问题
 - 几何问题、数值问题
- 本课程的重要程度 (★★★★★)
 - 信息和计算机科学的核心课程
 - 信息类专业学生必备技能
 - IT领域求职考试、面试内容



日常生活中的算法应用

淘宝
Taobao

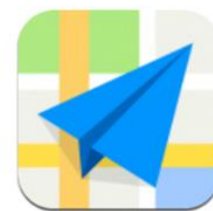
JD.COM

大众点评
dianping.com

互联网应用
(商品推荐、餐馆推荐...)



百度地图



高德地图

GPS系统
(最佳路线计算、公共设施推荐...)



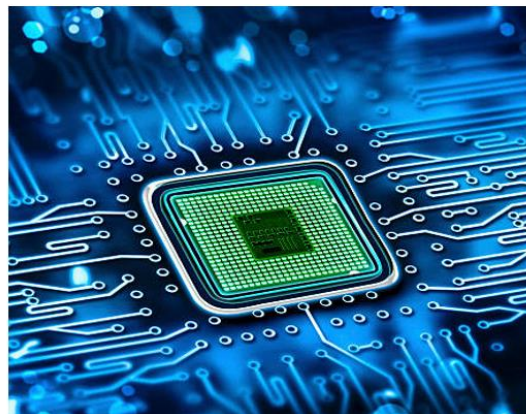
银行系统
(存取交易、在线支付...)

科技前沿中的算法应用



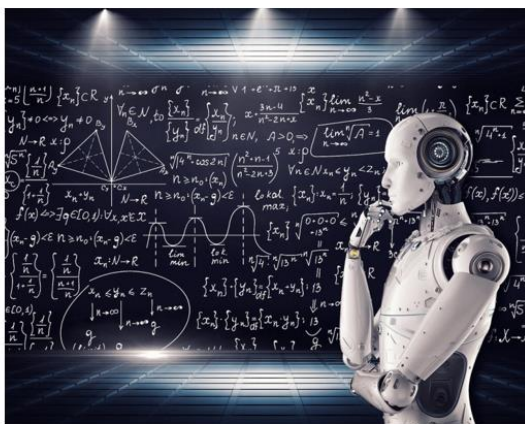
5G/6G无线通信

(数据编码和传输、网络优化、路由选择...)



芯片

(EDA工具优化、存算一体芯片...)



AI

(ChatGPT、AlphaGo、自动驾驶...)



载人航天

(通信导航一体、飞行模拟、热/力学建模...)

关于算法的应用, ChatGPT如何答复?



算法有哪些重要的应用?



算法是一种解决问题的方法,它在各个领域都有广泛的应用。以下是一些算法在不同领域中的重要应用:

1. 计算机科学: 算法是计算机科学中的基础,所有计算机程序都使用算法来实现各种任务,例如排序、搜索、字符串匹配、图形处理等。
2. 人工智能: 人工智能需要大量的算法来实现机器学习、深度学习、自然语言处理、计算机视觉等任务,这些算法的应用包括语音识别、图像识别、推荐系统、智能对话等。
3. 金融: 算法在金融领域中的应用包括股票交易、投资组合优化、风险管理、高频交易等,这些算法能够帮助金融机构做出更好的投资决策和风险管理策略。
4. 医疗保健: 算法在医疗保健领域中的应用包括医学影像分析、医学诊断、药物开发、疾病预测等,这些算法能够帮助医生做出更准确的诊断和治疗决策。
5. 物流与供应链管理: 算法在物流和供应链管理中的应用包括货运路线优化、仓库管理、库存管理等,这些算法能够帮助企业提高物流效率和降低成本。
6. 交通运输: 算法在交通运输领域中的应用包括路况预测、交通流优化、智能驾驶等,这些算法能够帮助提高交通运输效率和安全性。

总的来说,算法是各个领域中的基础和关键技术,对提高效率、降低成本、改善生活等方面都有重要的应用。



You
算法有哪些重要的应用?



ChatGPT
算法在各个领域都有重要的应用,其中一些包括但不限于:

1. **计算机科学和软件工程**: 算法是计算机科学的核心,它们用于解决各种问题,包括数据处理、搜索、排序、图像处理、文本分析、机器学习等。
2. **人工智能和机器学习**: 机器学习算法用于训练模型,使计算机能够从数据中学习并做出预测或决策。这包括监督学习、无监督学习、强化学习等算法。
3. **信息安全**: 加密算法用于保护数据的安全性,如对称加密算法、非对称加密算法、哈希函数等。
4. **网络和图论**: 网络算法用于解决网络优化、路由、流量控制等问题,图论算法用于解决图结构相关的问题,如最短路径、最小生成树、最大流等。
5. **生物信息学**: 算法被用于分析生物数据,如基因序列比对、蛋白质结构预测、基因组组装等。
6. **金融领域**: 算法被用于股票交易、风险管理、投资组合优化等方面。
7. **医疗保健**: 算法被用于医学影像分析、疾病预测、基因组学等方面。
8. **交通和物流**: 算法被用于交通优化、路径规划、货物配送等。
9. **社交网络和推荐系统**: 算法被用于个性化推荐、社交网络分析、信息过滤等。

这只是一小部分领域,算法在现代社会的各个方面都发挥着重要作用,它们帮助解决了许多复杂的问题,并促进了科技和社会的发展。

2023年2月的回答

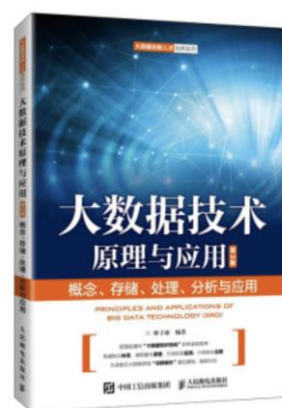
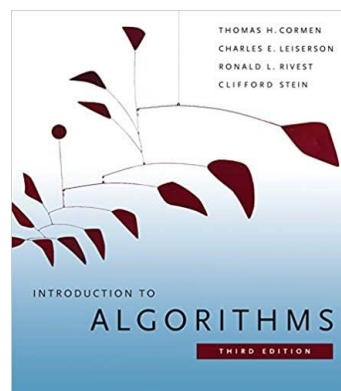
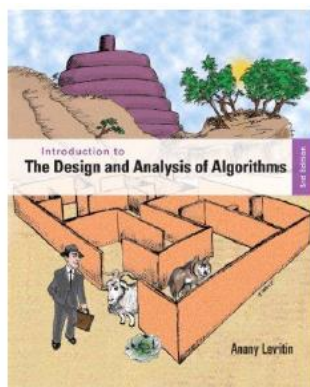
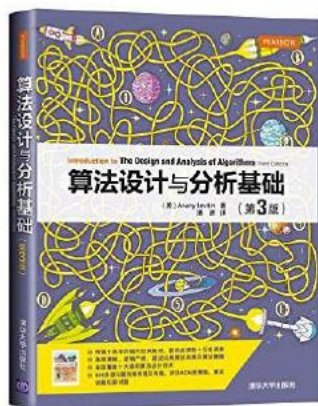


2024年2月的回答

大数据+大模型+强算法

参考教材

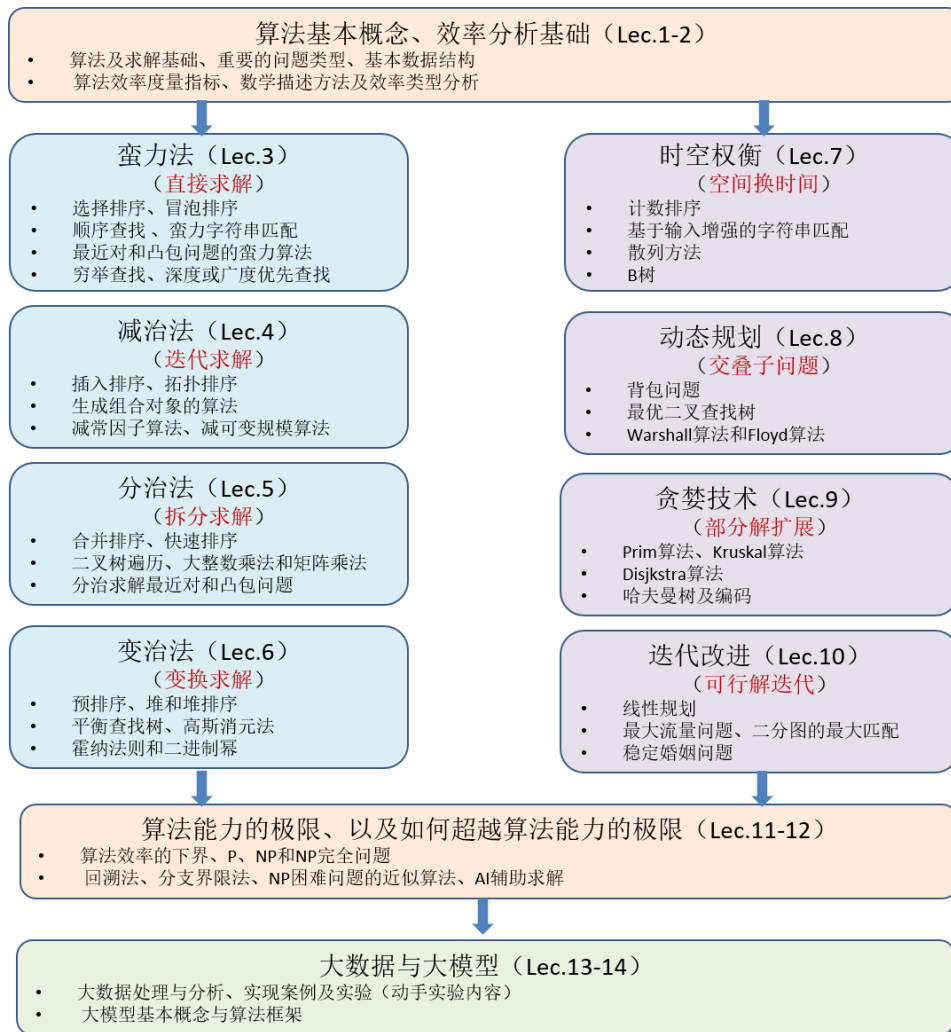
1. Anany Levitin (著), 潘彦 (译). **算法设计与分析基础**, 原书第3版. 清华大学出版社, 2015.
2. Anany Levitin. **Introduction to the Design and Analysis of Algorithms**, 3rd edition. Addison Wesley, 2011.
3. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. **Introduction to Algorithms**, 3rd edition. MIT Press, 2009.
4. 林子雨. **大数据技术原理与应用--概念、存储、处理、分析与应用**, 第3版. 人民邮电出版社, 2021.



教学安排

算法设计技术为主线

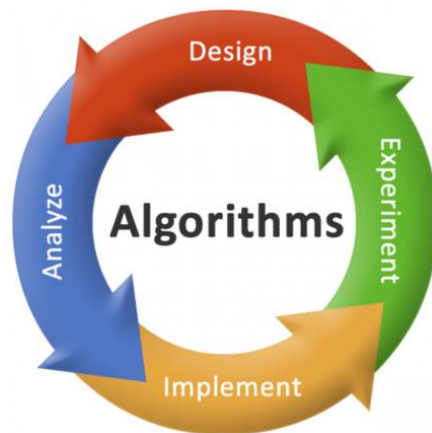
融入人工智能新思想



学习方法

- 预备课程
 - 程序设计基础
 - 概率论与数理统计、离散数学（基本概念）
- 重视**算法思想**，触类旁通
- 重视**编程实践**，熟能生巧

You Are Supposed to Be:
Cool minded
Self motivated
Programming addicted



算法思想的重要性

• 我与算法

- 研究方向：信息论、6G宽带通信、通信感知一体化
- 理论研究也需要大量的“**算法思想**”作为养分

IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, VOL. 35, NO. 7, JULY 2017

1619

Millimeter Wave Beam Alignment: Large Deviations Analysis and Design Insights

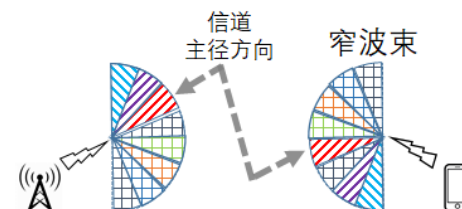
Chunshan Liu, *Member, IEEE*, Min Li, *Member, IEEE*, Stephen V. Hanly, *Fellow, IEEE*,
Iain B. Collings, *Fellow, IEEE*, and Philip Whiting

Abstract—In millimeter wave cellular communication, fast and reliable beam alignment via beam training is crucial to harvest sufficient beamforming gain for the subsequent data transmission. In this paper, we establish fundamental limits in beam-alignment performance under both the exhaustive search and the hierarchical search that adopts multi-resolution beamforming codebooks, accounting for time-domain training overhead. Specifically, we derive lower and upper bounds on the probability of misalignment for an arbitrary level in the hierarchical search, based on a single-path channel model. Using the method of large deviations, we characterize the decay rate functions of both bounds and show that the bounds coincide as the training sequence length goes large. We go on to characterize the asymptotic misalignment probability of both the hierarchical and exhaustive search, and show that the latter asymptotically outperforms the former, subject to the same training overhead and codebook resolution. We show via numerical results that this relative performance behavior holds in the non-asymptotic regime. Moreover, the exhaustive search is shown to achieve significantly higher worst case spectrum efficiency than the hierarchical search, when the pre-beamforming signal-to-noise

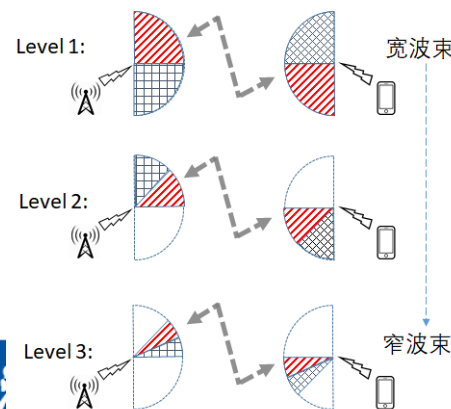
use of large scale antenna arrays and beamforming at transceivers to accommodate directional transmission in mmWave systems [6]–[8].

To achieve large beamforming gain, transmit and receive beams at base station (BS) and user equipment (UE) must be adaptively steered and aligned. Assuming perfect channel knowledge, recent studies [7]–[10] have developed a number of optimized hybrid analog and digital BS/UE beamforming solutions, subject to different hardware resource constraints. However, accurate estimation of the channel (i.e., all entries of the channel matrix) itself is a challenging task in mmWave communications, considering the large scale antenna arrays employed.

Another viable approach for beam alignment at mmWave is beam training, which was first considered in the design of mmWave multi-Gigabit wireless local area network (WLAN) [11], [12] and wireless personal area network (WPAN) [13] and has received significant attention in



“蛮力”与“减治”思想的较量



程学院

College of Information Science & Electronic Engineering, Zhejiang University

课程考核和评分标准

- ① 课堂表现占10%：到课情况、讨论发言等。
- ② 课后习题占20%：布置的课后习题完成质量。
- ③ 线上测试占5%：“学在浙大”的随堂在线测试。
- ④ 编程作业占15%：编程作业的完成质量。
- ⑤ 期末考试占50%：期末闭卷考试。



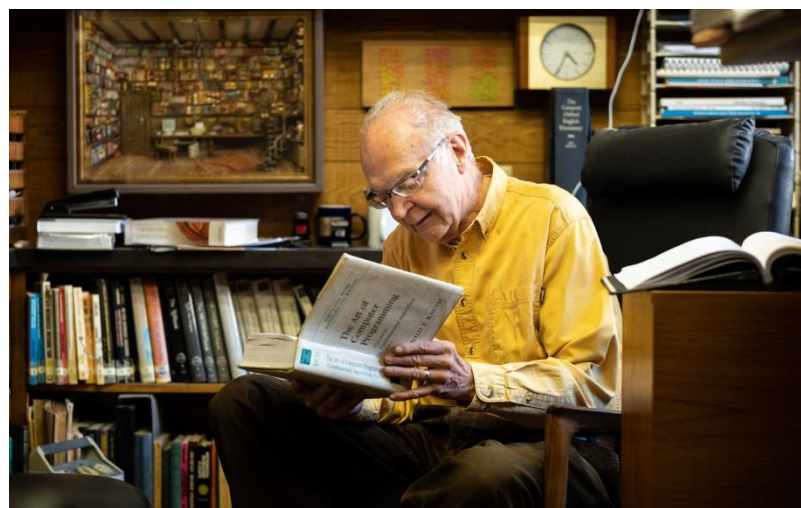
教学辅助信息

- 我的联系方式
 - 手机：18758064038
 - 邮箱：min.li@zju.edu.cn
 - 办公室：玉泉校区教11-241
 - 主页：<https://person.zju.edu.cn/limin>
- 助教：石滨溥（研究生）
 - 手机：13646501257
 - 邮箱：bp.shi@zju.edu.cn
- 线上资源
 - 课件等课程资料通过“学在浙大”平台定期更新
 - 钉钉课程交流群（有问题及时咨询）

目录

- 课程简介
 - 课程内容
 - 教学安排
 - 课程考核
- 绪论
 - 什么是算法?
 - 算法问题求解基础
 - 重要的问题类型
 - 基本数据结构

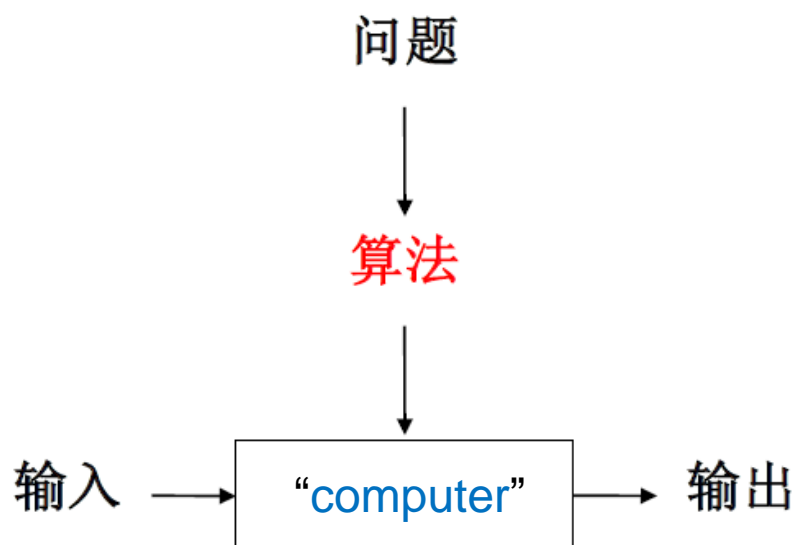
受过良好训练的计算机科学家知道怎样处理算法：如何构造算法、操作算法、理解算法以及分析算法。这些知识远不止为了写出良好的计算机程序而准备的。算法是一种一般性的智能工具，它必定有助于我们对其他学科的理解，不管是化学、语言学或音乐，还是其他学科。为什么算法会有这种作用呢？我们可以这样理解：人们常说，一个人只有把知识教给别人，才能真正掌握它。实际上，一个人只有把知识教给“计算机”，才能“真正”掌握它，也就是说，将知识表述为一种算法……比起简单按照常规去理解事物，尝试用算法将其形式化能使我们的理解更加深刻。([Knu96], p. 9)



Donald Knuth: Computer scientist, mathematician,
and professor emeritus at Stanford University

什么是算法？

- 算法是指解题方案的准确而完整的描述，是一系列解决问题的明确指令，也就是说，对于符合一定规范的输入，能够在有限时间内获得所要求的输出。



➤ “computer”: 理解和执行指令的人或物

算法满足下列性质

- 输入：有零个或多个外部量作为算法的输入。
- 输出：算法产生至少一个量作为输出。
- 确定性：组成算法的每条指令清晰、无歧义。
- 有限性：算法中每条指令的执行次数有限，执行每条指令的时间也有限。



同一问题，不同算法

- 不同算法可能采用不同的时间或空间效率来完成同样的任务
- 范例：求解最大公约数

Problem: Find $\text{gcd}(m,n)$, the greatest common divisor of two nonnegative, not both zero integers m and n

Examples: $\text{gcd}(60,24) = 12$

求解最大公约数：算法一

- 质因数分解法

- 第一步：找出m的所有质因数
- 第二步：找出n的所有质因数
- 第三步：从第一步求得的m的质因数分解式和第二步求得的n的质因数分解式中，找出所有公因数
- 第四步：将第三步找到的公因数相乘，结果即为 $\gcd(m,n)$

例如：求24和60的最大公约数

- 先分解质因数，得 $24=2 \times 2 \times 2 \times 3$ ， $60=2 \times 2 \times 3 \times 5$
- 24与60的全部公有的质因数是2、2、3，它们的积是 $2 \times 2 \times 3=12$
- 所以， $\gcd(24, 60)=12$ 。

- 这是一个真正意义上的算法吗？
- 存在问题：如何求所有的质因数？

埃拉托色尼筛选法 (Sieve of Eratosthenes)

2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
2	3		5		7		9		11		13		15		17		19		21		23		25
2	3		5		7				11		13				17		19				23		25
2	3		5		7				11		13				17		19				23		



1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

埃拉托色尼筛选法

(Sieve of Eratosthenes)

算法 Sieve(n)

//实现“埃拉托色尼筛选法”

//输入：一个正整数 $n > 1$

//输出：包含所有小于等于 n 的质数的数组 L

for $p \leftarrow 2$ to n do $A[p] \leftarrow p$

for $p \leftarrow 2$ to $\lfloor \sqrt{n} \rfloor$ do //参见伪代码前的说明

if $A[p] \neq 0$ // p 没有被前面的步骤消去

$j \leftarrow p * p$

while $j \leq n$ do

$A[j] \leftarrow 0$ //将该元素标记为已经消去

$j \leftarrow j + p$

//将 A 中剩余的元素复制到质数数组 L 中

$i \leftarrow 0$

for $p \leftarrow 2$ to n do

if $A[p] \neq 0$

$L[i] \leftarrow A[p]$

$i \leftarrow i + 1$

return L

2p, ..., (p-1)p 已经
在先前的步骤中消
去，无需重复

求解最大公约数：算法二

- 连续整数检测法
 - 第一步：将 $\min\{m,n\}$ 赋值给 t
 - 第二步： m 除以 t ，如果余数为 0，进入第三步，否则进入第四步
 - 第三步： n 除以 t ，如果余数为 0，返回 t 的值；否则进入第四步
 - 第四步：把 t 的值减1，返回第二步

注：输入为0时，算法错误

求解最大公约数：算法三

- 欧几里得算法（Euclid's Algorithm）
- 最出名、历史最悠久；首次出现于《几何原本》（约公元前300年）

算法

第一步：如果 $n = 0$ ，返回 m 的值作为结果，同时过程结束；否则，进入第二步。

第二步： m 除以 n ，将余数赋给 r 。

第三步：将 n 的值赋给 m ，将 r 的值赋给 n ，返回第一步。

伪代码

ALGORITHM *Euclid*(m, n)

//Computes $\gcd(m, n)$ by Euclid's algorithm

//Input: Two nonnegative, not-both-zero integers m and n

//Output: Greatest common divisor of m and n

while $n \neq 0$ **do**

$r \leftarrow m \bmod n$

$m \leftarrow n$

$n \leftarrow r$

return m

$$\gcd(60, 24) = \gcd(24, 12) = \gcd(12, 0) = 12.$$

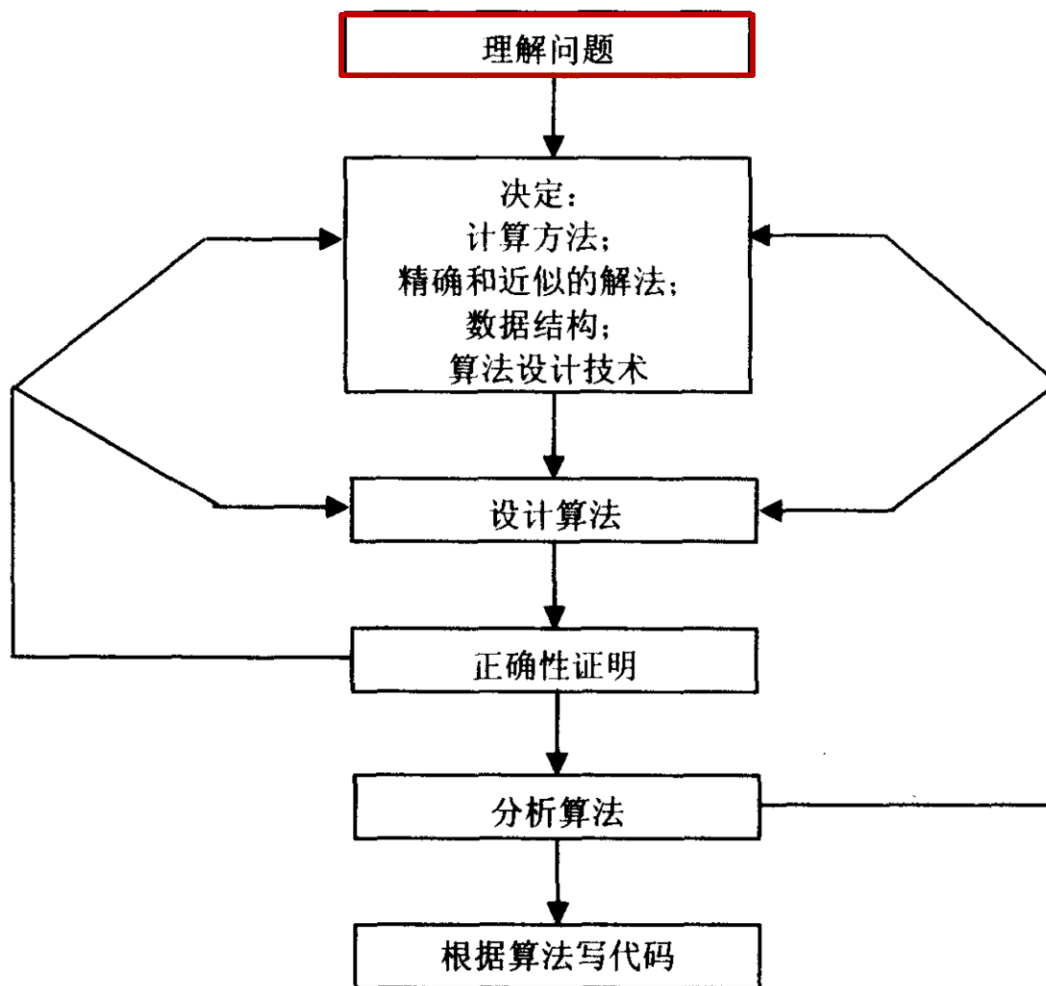


古希腊数学家
欧几里得

目录

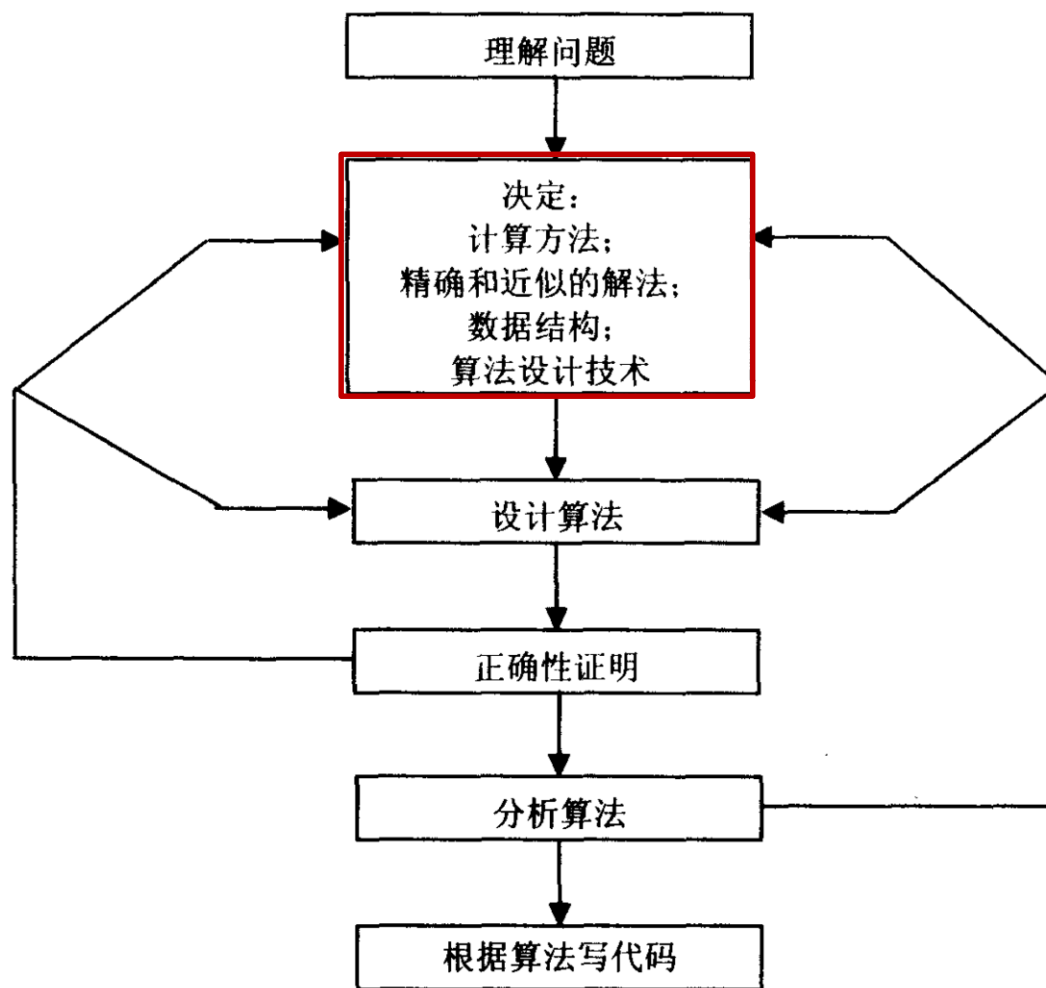
- 课程简介
 - 课程内容
 - 教学安排
 - 课程考核
- 绪论
 - 什么是算法?
 - 算法问题求解基础
 - 重要的问题类型
 - 基本数据结构

算法的设计和分析过程



(1) 理解问题

- 当遇到一个问题时，首先要清楚这个问题的所有内容。如果这个问题已经给出了明显的要求，如对成绩排序，那么只需要看看它是属于哪一类的问题，然后参考相关的资料。
- 了解问题内容这个步骤是十分重要的，因为只有知道了问题具有什么样的输入，需要得到什么样的输出，问题的解决才可能进行下去。**理解问题是问题求解的关键。**



(2) 了解设备的能力

- 在清楚了解问题的内容之后，下一步是确定用于解决问题的设备的能力。目前一般使用的计算机都是**冯诺依曼（von Neumann）体系**架构的。它的一个最重要假设是，程序指令执行是顺序的。针对这一类计算机的算法被称为**串行算法** (sequential algorithms)。
- 与串行算法相区别的是并行计算机以及**并行算法** (parallel algorithms)。指令能够并行执行，效率当然会大大提高，但额外需要考虑的则是指令执行顺序以及同步等问题。
- 并行算法的设计有相应的理论，这里仅考虑串行算法

(3) 精确解法 or 近似解法?

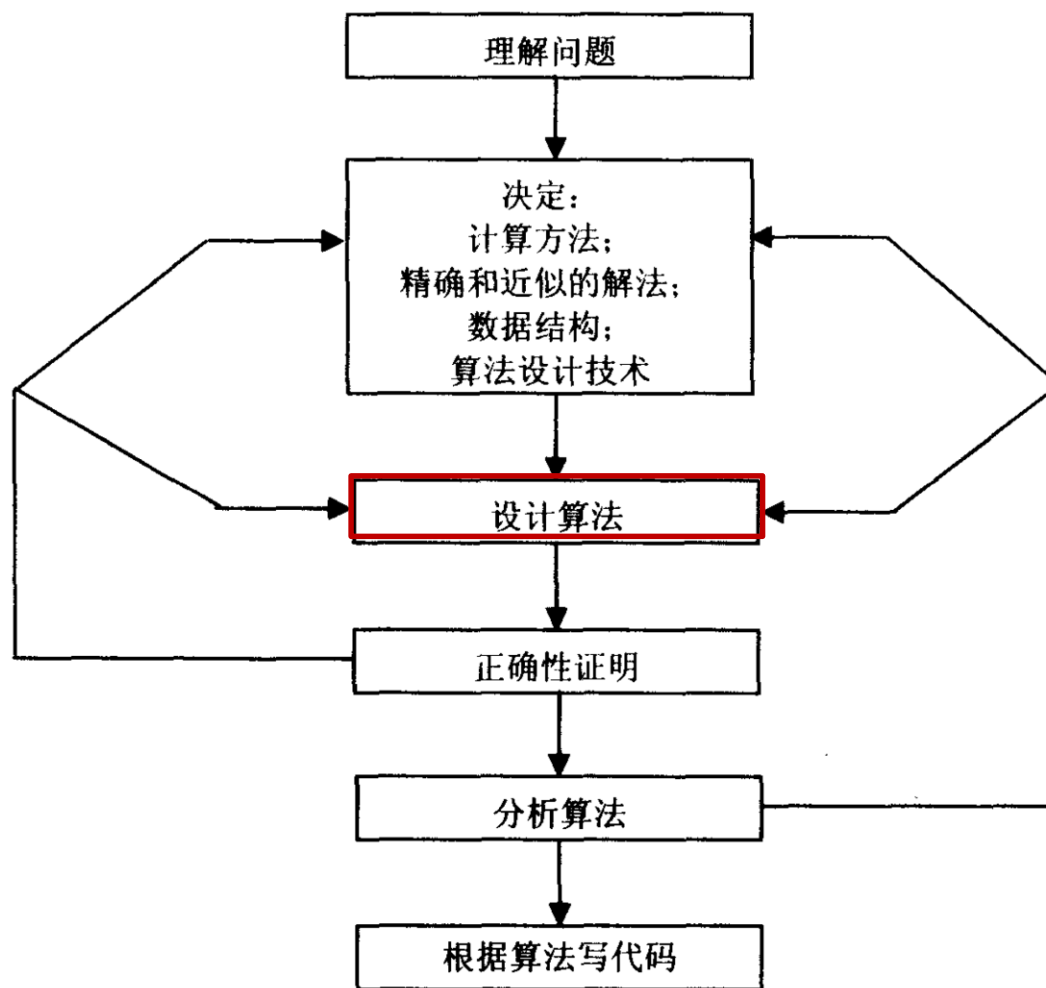
- 并不是每一个可解的问题都有精确的算法，例如求一个数的平方根，求非线性方程的解等。
- 有时候一个问题有精确的解法但是算法的执行效率很差，例如旅行家问题。
- 因此，如果待处理的问题涉及到上述那些方面，则要考虑是选择精确的还是近似的算法。

(4) 确定适当的数据结构

- 许多程序设计的教材都提到：程序设计 = 算法 + 数据结构 (Programs = Algorithms + Data Structures), 由此可以看出数据结构对算法的重要性。
- 例如，在处理搜索问题时，对于仅仅进行搜索的算法只需要用到简单数组即可，如果搜索后伴随着插入删除操作时，那么用链表以及堆等复杂的数据结构算法更加可取。

(5) 选择算法的设计技术

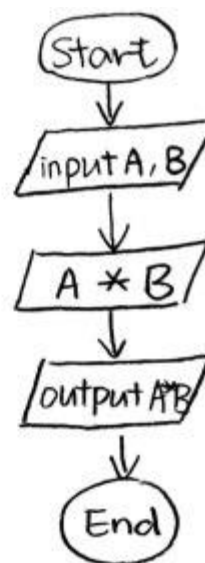
- 算法设计技术（algorithm design technique）或者算法设计策略（strategy）指的是**解决一系列不同问题的通用设计思想**。
- 常用设计技术包括：
 - 分治法（Divide and Conquer）
 - 贪婪法（Greedy Technique）
 - 动态规划（Dynamic Programming）
 - 回溯法（Backtracking）
 - 分支限定法（Branch and Bound）
 - ...

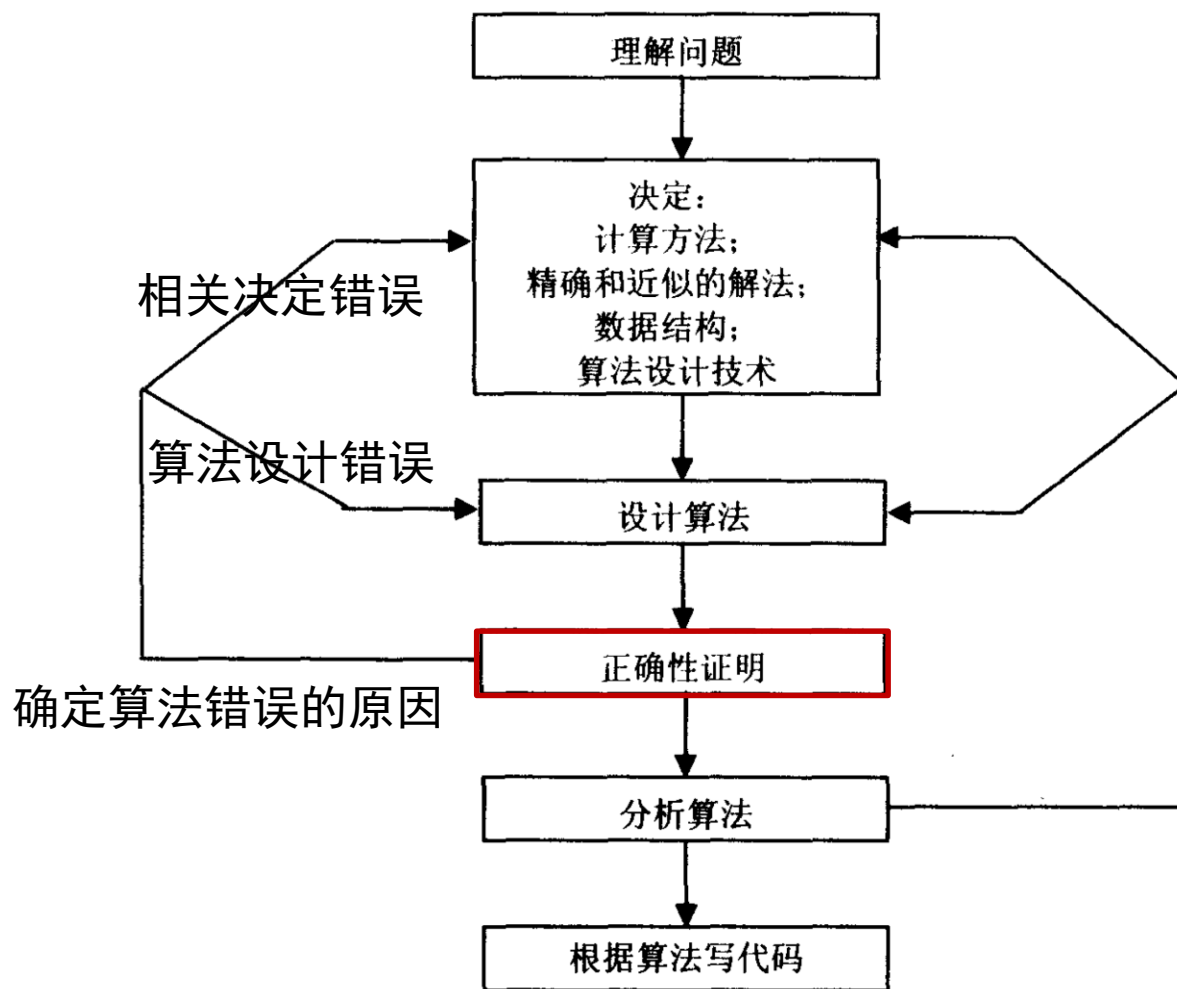


(6) 算法的描述

- 可以用到的工具有自然语言（nature language）、伪代码（ pseudocode ）以及程序流图（flow chart）等
- 当对一个问题有了概要的理解后，下面工作就是把这个问题的想法进行细化。所谓的细化就是把它们表示成算法的步骤。

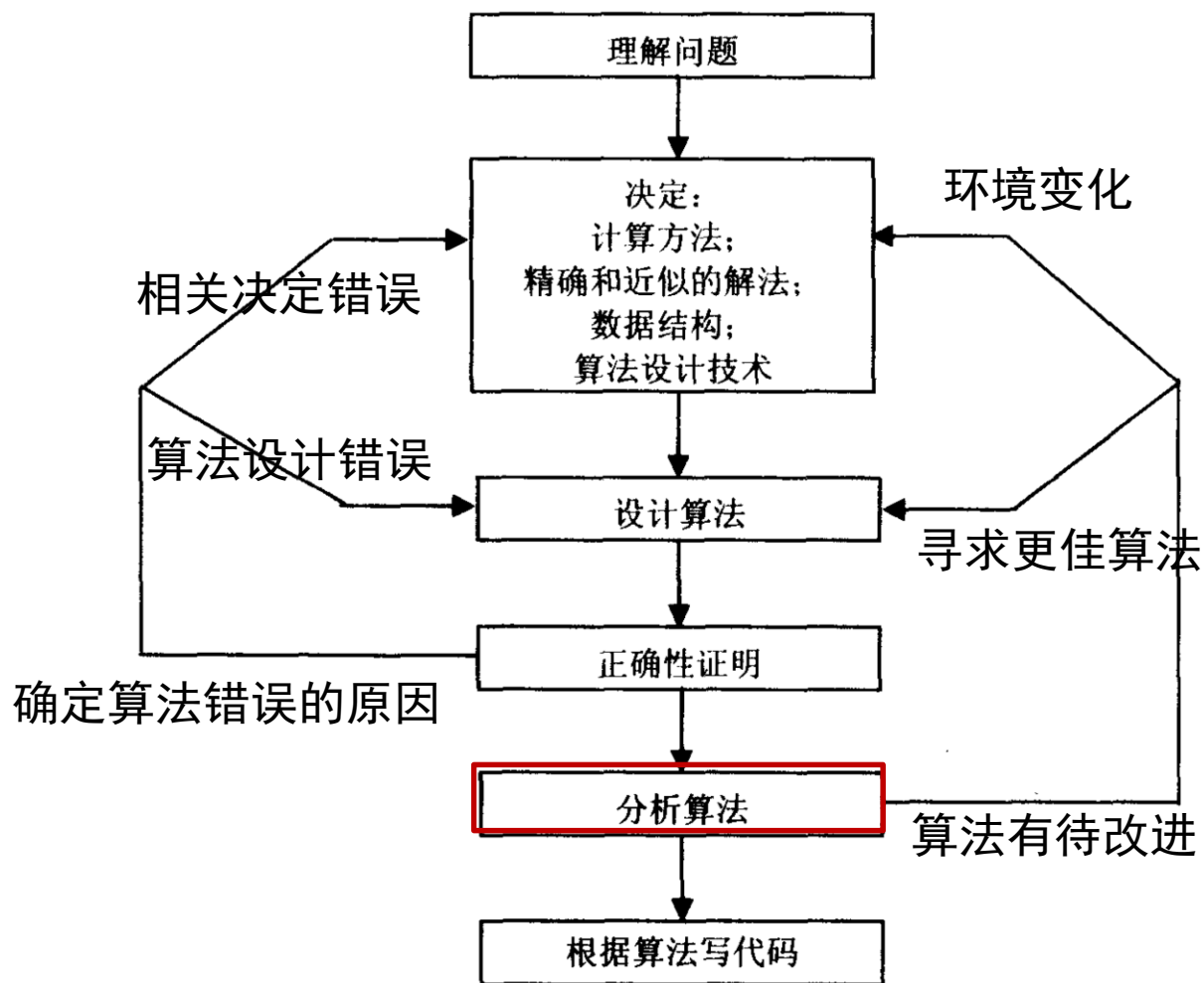
input A
input B
 $ANS = A * B$
output ANS





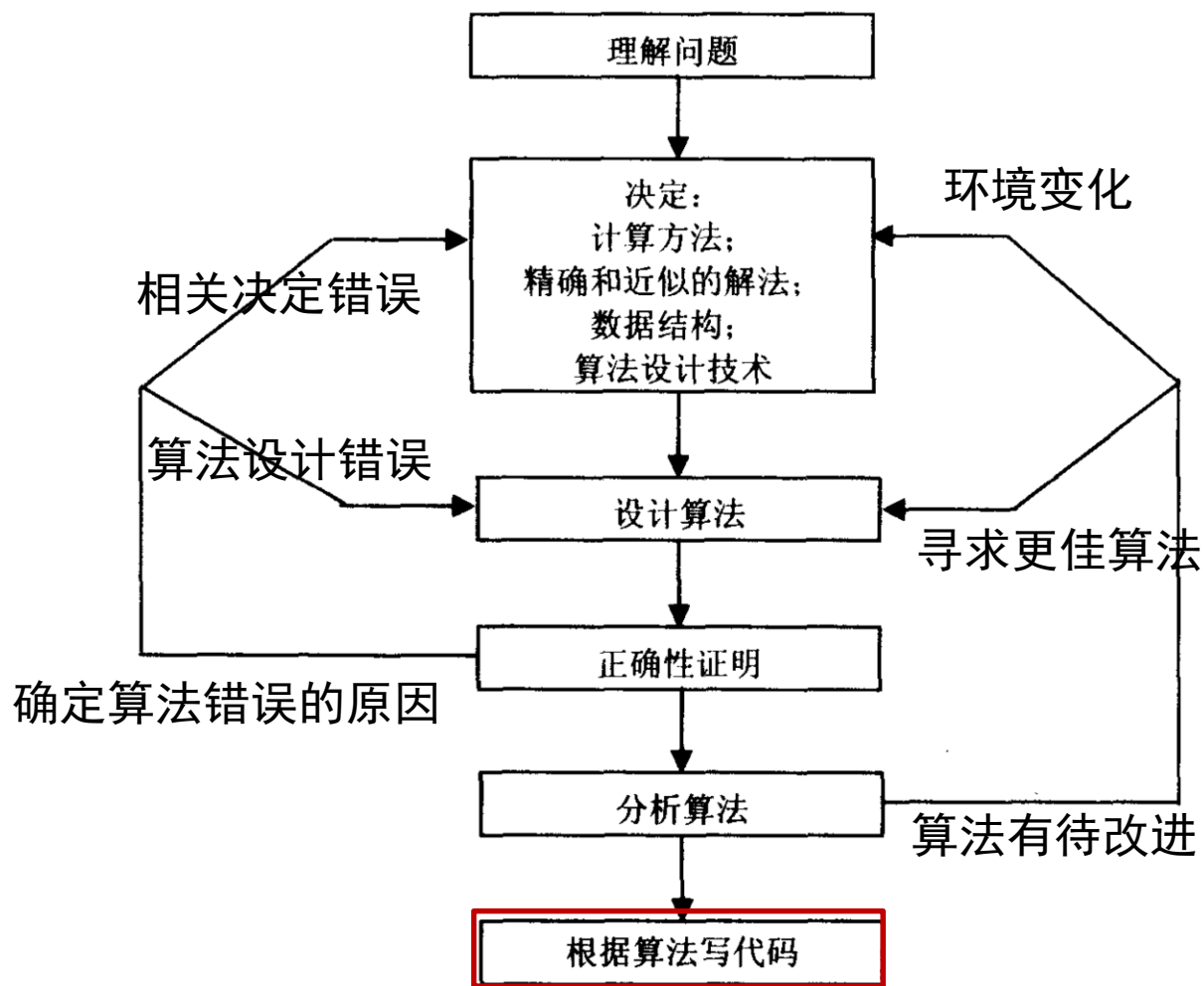
(7) 确认算法的正确性

- 当给算法一个合法输入，而该算法给出一个错误的结果时，可以证明这个算法是错误的。但如果要证明一个算法是正确的，情况就不那样的简单。算法的测试有一套完整理论，这里不作详细叙述。
- 特别需要提出的是，所谓算法正确性指：对于任意合法的输入，在有限时间内都能得到正确的结果。如果算法是一个近似算法，那么，正确性则是指输出结果与理论结果之间的误差在一个可以接受的范围内。



(8) 对算法的分析

- 时空的观点：
 - 时间效率
 - 空间效率
- 交流的观点：算法最容易理解（simplicity）
- 发展的观点：算法适应性（generality）最强
- 设计的观点：算法的设计时间最少



(9) 为算法写代码

- 目前而言，大部分的算法最后还是需要**通过程序语言进行实现**。针对不同的问题，还需要考虑什么样的程序语言才是最适合，这与**个人的知识组成以及程序语言本身特性**等有关。
- 部分的程序语言可能需要对算法进行修改以应这种语言本身。例如广泛用于科学计算的Matlab语言，它是**基于矩阵**的语言，提供了大量数值计算的函数，然而它的循环计算能力相对较弱，所以如果把算法中涉及循环的过程通过**向量的形式**进行，那么最终效率将能得到很大的提高。

算法设计与分析之精髓

“不是在无以复加，而是在无以复减的时候，设计师才知道他已经到达了完美的境界。”——圣·埃克苏佩里



• 安东尼·德·圣·埃克苏佩里

- 圣·埃克苏佩里（1900—1944）是一名法国作家、飞行冒险家，1900年6月29日生于法国里昂，在1944年7月31日晚执行一次飞行任务时失踪。
- 他一生喜欢冒险和自由，是利用飞机将邮件传递到高山和沙漠的先锋。二次大战期间，法国被纳粹占领，他去了美国。1944年他在一次飞行任务中失踪，成为法国文学史上最神秘的一则传奇。
- 除了飞行，用写作探索灵魂深处的寂寞是他的另一终生所爱。



目录

- 课程简介
 - 课程内容
 - 教学安排
 - 课程考核
- 绪论
 - 什么是算法?
 - 算法问题求解基础
 - 重要的问题类型
 - 基本数据结构

重要的问题类型

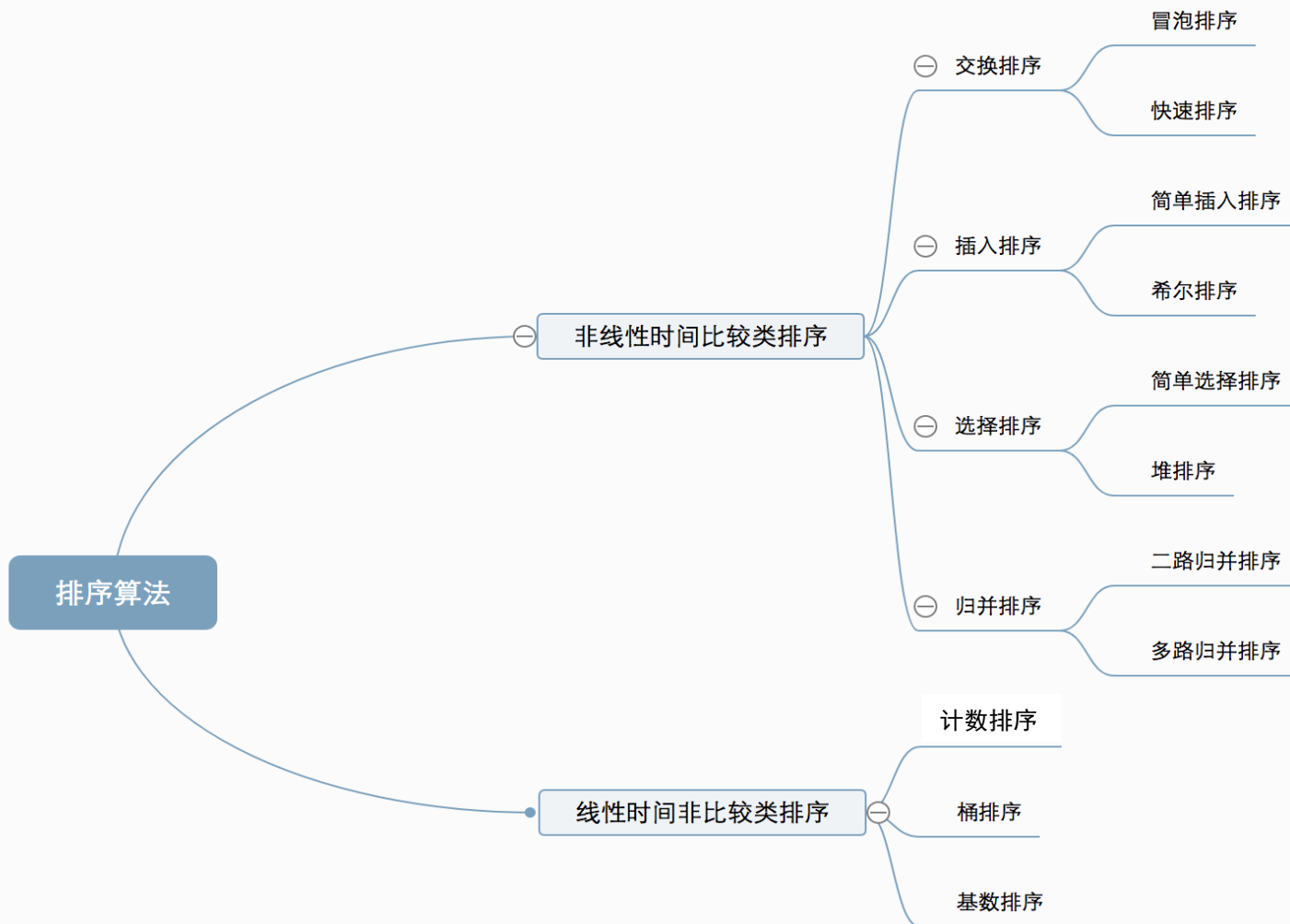
- 排序 (Sorting Problems)
- 查找/搜索 (Searching Problems)
- 字符串处理 (String Processing Problems)
- 图问题 (Graph Problems)
- 组合问题 (Combinatorial Problems)
- 几何问题 (Geometric Problems)
- 数值问题 (Numerical Problems)

(1) 排序问题

- “排序”，目的是把一组杂乱的数据按照一定的规律顺次排列起来。排序是很重要的，因为它可以使得以后的工作更加地容易。
- 计算机科学家发明了几十种排序算法，然而这些排序算法中并不存在一种算法是绝对最优的。
- 与排序相关的重要概念：数据表、关键字、排序的稳定性、内排序与外排序。

sorted by time	sorted by location (not stable)	sorted by location (stable)
Chicago 09:00:00	Chicago 09:25:52	Chicago 09:00:00
Phoenix 09:00:03	Chicago 09:03:13	Chicago 09:00:59
Houston 09:00:13	Chicago 09:21:05	Chicago 09:03:13
Chicago 09:00:59	Chicago 09:19:46	Chicago 09:19:32
Houston 09:01:10	Chicago 09:19:32	Chicago 09:19:46
Chicago 09:03:13	Chicago 09:00:00	Chicago 09:21:05
Seattle 09:10:11	Chicago 09:35:21	Chicago 09:25:52
Seattle 09:10:25	Chicago 09:00:59	Chicago 09:35:21
Phoenix 09:14:25	Houston 09:01:10	Houston 09:00:13
Chicago 09:19:32	Houston 09:00:13	Houston 09:01:10
Chicago 09:19:46	Phoenix 09:37:44	Phoenix 09:00:03
Chicago 09:21:05	Phoenix 09:00:03	Phoenix 09:14:25
Seattle 09:22:43	Phoenix 09:14:25	Phoenix 09:37:44
Seattle 09:22:54	Seattle 09:10:25	Seattle 09:10:11
Chicago 09:25:52	Seattle 09:36:14	Seattle 09:10:25
Chicago 09:35:21	Seattle 09:22:43	Seattle 09:22:43
Seattle 09:36:14	Seattle 09:10:11	Seattle 09:22:54
Phoenix 09:37:44	Seattle 09:22:54	Seattle 09:36:14

Stability when sorting on a second key



(2) 查找/搜索

- 查找/搜索，目的是在数据集中寻找满足某种条件的数据对象。
- 查找的一般形式是先给出一个特定的值，然后在数据集中找出关键字等于该特定值的对象。
- 对于搜索，则通常返回两种可能的结果：一种是简单的搜索成功或失败信息；另一种是返回满足搜索条件的对象所在位置，如果找不到则返回不存在信息。



(3) 字符串处理

- **字符串**：字母表中的符号（字母、数字、特殊符号）所构成的序列。

`strtok()` 字符串分割函数

`strstr()` 字符串查找函数

`strspn()` 字符查找函数

`strrchr()` 定位字符串中最后出现的指定字符

`strpbrk()` 定位字符串中第一个出现的指定字符

`strncpy()` 复制字符串

`strncat()` 字符串连接函数

`strncasecmp()` 字符串比较函数(忽略大小写)

`strlen()` 字符串长度计算函数

`strdup()` 复制字符串

`strcspn()` 查找字符串

`strcpy()` 复制字符串

`strcoll()` 字符串比较函数(按字符排列次序)

`strcmp()` 字符串比较函数(比较字符串)

`strchr()` 字符串查找函数(返回首次出现字符的位置)

`strcat()` 连接字符串

`strcasecmp()` 字符串比较函数(忽略大小写比较字符串)

`rindex()` 字符串查找函数(返回最后一次出现的位置)

`index()` 字符串查找函数(返回首次出现的位置)

`toupper()` 字符串转换函数(小写转大写)

`tolower()` 字符串转换函数(大写转小写)

`toascii()` 将整数转换成合法的ASCII码字符

`strtoul()` 将字符串转换成无符号长整型数

`strtol()` 将字符串转换成长整型数

`strtod()` 将字符串转换成浮点数

`gcvt()` 将浮点型数转换为字符串(四舍五入)

`atol()` 将字符串转换成长整型数

`atoi()` 将字符串转换成整型数

`atof()` 将字符串转换成浮点型数

(4) 图问题

- 图论问题主要是与图或树相关的问题。这里的“图”指的是一些顶点（Vertex）与边（Edge）的集合。
- 图有着非常重要与丰富的实际应用，如交通、通信、社会、经济网络等。
- 主要问题包括图的遍历问题、最短路径计算、有向图的排序、旅行商问题、图着色问题等
 - 旅行商问题：找出访问 n 个城市的最短路径，并保证每个城市只访问一次；
 - 图填色问题：用最少种类的颜色为图中的顶点着色，并保证任何两个邻接顶点的颜色不同

(5) 组合问题

- 组合问题，则要寻找一个能够**满足特定约束或特征的组合对象**，例如一个排列、一个组合或者一个子集。
- 现代组合数学所涉及的领域非常广阔，其中的问题出自抽象代数、拓扑学、基础数学、图论、博弈论、线性规划以及其它许多领域。
- 值得注意的是，这些来源不同的问题**很难在一个统一的理论体系中得到有效解决**，而且随着问题的输入量的增加，问题**规模急剧增长**至计算机处理能力的极限，所以可说组合数学问题是最难的一类问题。**人工智能**对于这类问题的求解可发挥重要作用。

(6) 几何问题

- 几何问题处理类似点、线、多面体这样的几何对象。
- 本课程只讨论两个经典的几何问题：
 - 最近对问题：给定平面上 n 个点，求距离最近的两个点。
 - 凸包问题：找一个能把给定集合中所有点都包含在里面的最小凸多边形。



(A) eggs



(B) frog



(C) aircraft

(7) 数值问题

- 数值问题是**计算机科学与数学**的结合，包括：**方程或方程组求解**、插值及逼近方法、数值微分/积分、常微分方程求解等（“**数值分析方法**”课程67120080）。
- 数值问题通常只能给出一个**近似解**，而不是精确的解析解。这些问题常涉及到具体的数值计算，由于计算机对数字表示精度的限制，还需考虑计算过程中可能带来的误差、以及误差对算法稳定性的影响。

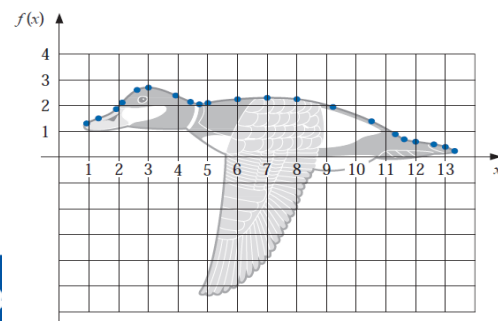
如何求解非线性方程？

$$x = (x + 1)^{\frac{1}{2}} + x^{\frac{1}{3}}$$

如何计算无线通信的传输速率？

$$C = \int_0^{\infty} \log(1 + x) \exp(-x) dx$$

如何基于有限的数据采样点拟合ruddy duck的飞行姿态？



目录

- 课程简介
 - 课程内容
 - 教学安排
 - 课程考核
- 绪论
 - 什么是算法？
 - 算法问题求解基础
 - 重要的问题类型
 - 基本数据结构

数据结构与算法



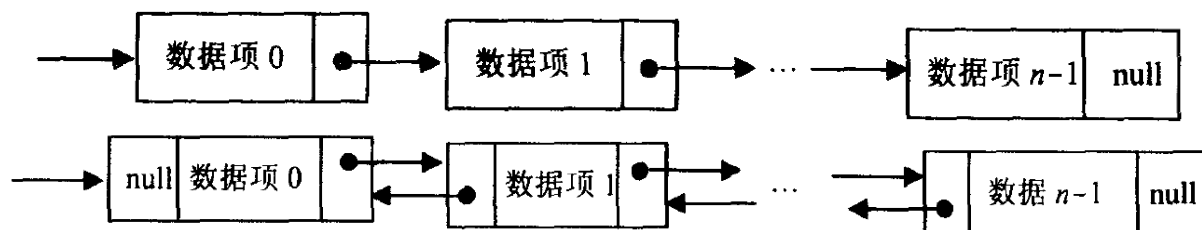
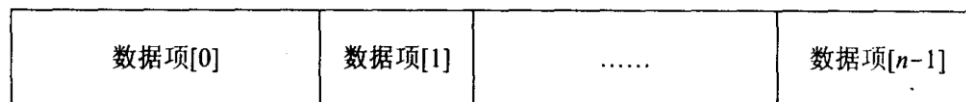
数据结构与算法	拼装积木
输入数据	未拼装的积木
数据结构	积木组织形式，包括形状、大小、连接方式等
算法	把积木拼成目标形态的一系列操作步骤
输出数据	积木模型

数据结构

- 基本的数据结构类型

- 线性列表

- 数组
 - 链表
 - 字符串
 - 栈和队列



栈：后进先出



队列：先进先出

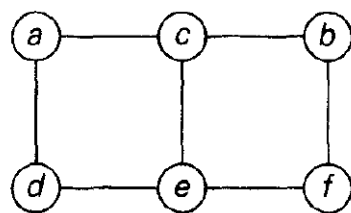


数据结构

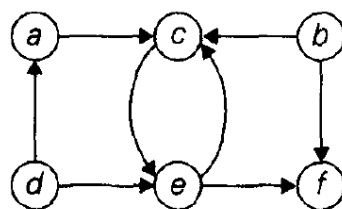
• 基本的数据结构类型

- 图

- 分类：无向图、有向图
- 表示：邻接矩阵、邻接链表
- 特性：稀疏度、连通性、无环性

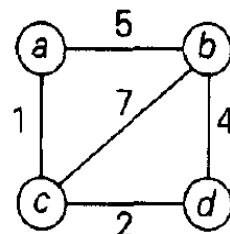


	a	b	c	d	e	f
a	0	0	1	1	0	0
b	0	0	1	0	0	1
c	1	1	0	0	1	0
d	1	0	0	0	1	0
e	0	0	1	1	0	1
f	0	1	0	0	1	0



a	→ c → d
b	→ c → f
c	→ a → b → e
d	→ a → e
e	→ c → d → f
f	→ b → e

加权图



	a	b	c	d
a	∞	5	1	∞
b	5	∞	7	4
c	1	7	∞	2
d	∞	4	2	∞

a	→ b, 5 → c, 1
b	→ a, 5 → c, 7 → d, 4
c	→ a, 1 → b, 7 → d, 2
d	→ b, 4 → c, 2

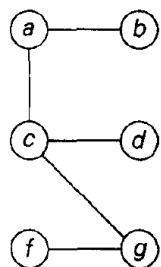
数据结构

- 基本的数据结构类型

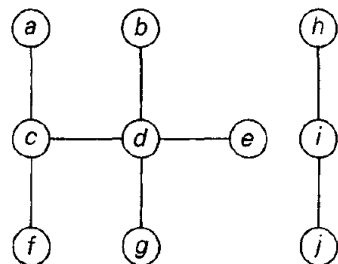
- 树

- 自由树：连通无回路图（边数总比顶点数少1）
 - 有根树：任选自由树中的一个顶点作为根，可将自由树转化为有根树

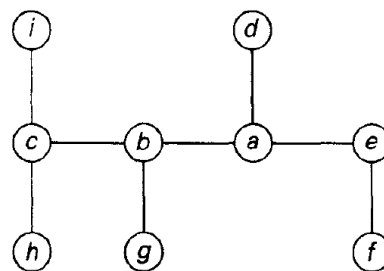
- 定义：祖先/真祖先、父母、子女/兄弟、子孙/真子孙、叶节点、父节点
 - 深度：从根到节点V的路径长度；高度：从根到叶节点的最长路径长度



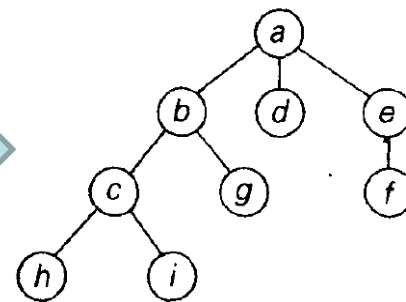
自由树



森林



自由树



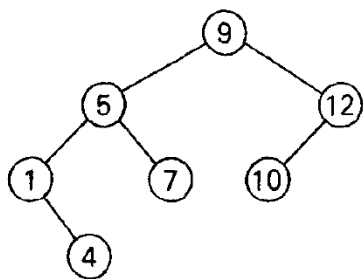
有根树

数据结构

- 基本的数据结构类型

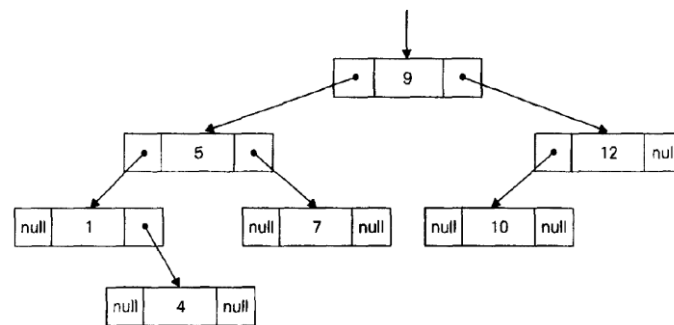
- 树

- 有序树：一棵有根树，且每一顶点的所有子女都是有序的
 - 二叉树：有序树的特例，每个节点的子女数不超过2



二叉查找树

(节点信息+左右子女指针)



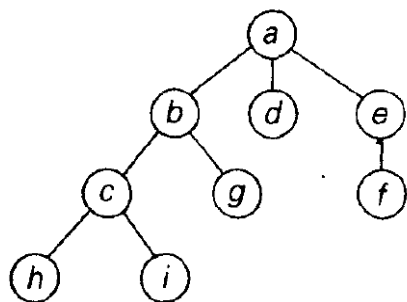
标准的二叉树链表实现

数据结构

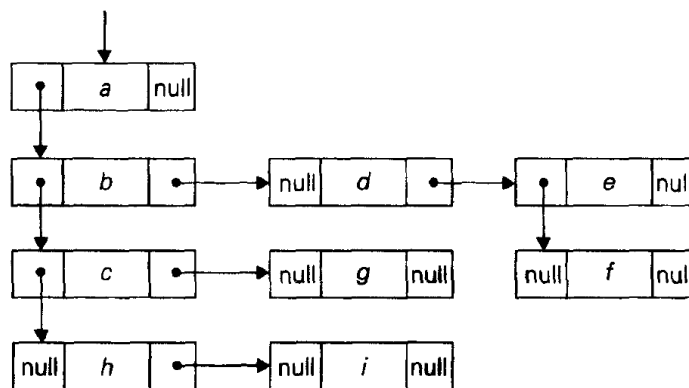
- 基本的数据结构类型

- 树

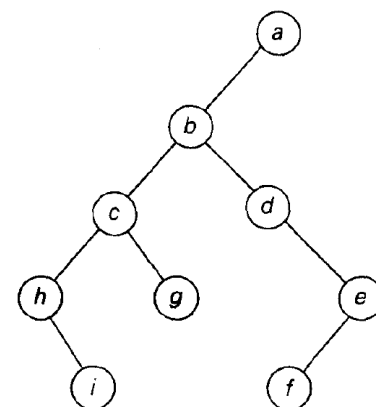
- 有序树：一棵有根树，且每一顶点的所有子女都是有序的
 - 二叉树：有序树的特例，每个节点的子女数不超过2
 - 有序树可改造成成为相应的关联二叉树



原始的有序树



基于先子女后兄弟的链表表示



改造后的关联二叉树

数据结构

- 基本的数据结构类型

- 集合和字典

- 集合

- **位向量**，例如全集 $U = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ ，子集 $S = \{2, 3, 5, 7\}$ 可表示为位串011010100

- 线性列表

- 字典

- 支持在集合中**查找、增加、删除元素**三种操作的数据结构

- 抽象数据类型

- 由一个表示数据项的抽象对象集合和一系列对这些对象所做的操作构成
 - 基于面向对象语言来实现：类(class)

课后作业

章 X	节 X. Y	课后作业题 Z	思考题 Z
1	1.1		11, 12
	1.2	1	2, 9
	1.3	1, 8	4, 5
	1.4		10

注：只需上交“课后作业题”；以“学号姓名_chx. pdf ” 规范命名，提交到“学在浙大”指定文件夹。DDL：2024年3月5日