计算机科学导论

第5章
算法与复杂性

杨老师邮箱: fdteachers@163.com

第5章 算法与复杂性

学习目标

- 了解算法的概念和特性
- 算法的描述工具、评价
- 掌握几种经典算法的基本思想:递归算法。
- 算法设计策略: 分治策略　动态规划　　贪心算法
- 分布式算法、可计算性理论基础、NP问题、自动机理论、加密算法、几何算法、并行算法等。

- 一个好的算法是程序设计的关键
- 本章首先介绍算法的基本知识、常用算法及算法评价的基础知识
- 然后介绍几种常用的算法
- 为今后进一步学习算法及其复杂性打好基础。

5.1 算法分析

5.1.1 算法

“算法”即演算法，中文名称出自《周髀算经》，英文名称Algorithm 来自于9世纪波斯数学家al-Khwarizmi，因为他在数学上提出了算法这个概念。“算法”原为"algorism"，意思是阿拉伯数字的运算法则，在18世纪演变为"algorithm"。

5.1 算法分析

5.1.1 算法

- 大卫·希尔伯特(David Hilbert，1862~1943)
  1900年，在巴黎举行的第2届国际数学家大会上，他作了题为《数学问题》的著名演讲，提出了新世纪所面临的23个问题。
- 库尔特·哥德尔、赫尔布兰德和斯蒂芬·科尔·克莱尼分别于 1930年、1934年和1935年提出的递归函数，阿隆佐·邱奇于1936年提出的λ演算，1936年Emil Leon Post的Formulation 1和艾伦·图灵1937年提出的图灵机。

5.1.1 算法

- 定义
- 算法（Algorithm）是一组明确的、可以执行步骤的有序集合，在有限的时间内终止并产生结果。
- 算法和数据结构之间存在密切联系，数据结构是算法的基础，数据结构不同，通常采用的算法也不同。

5.1.2 算法的特性

算法反映了求解问题的方法和步骤，不同的问题需要用不同的算法来解决，同一个问题也可能有多种不同的算法。

一个算法必须具有以下特性：

有穷性(可终止性)

一个算法必须在有限的操作步骤内以及合理的时间内执行完成。

确定性

算法中的每一个操作步骤都必须有明确的含义，不允许存在二义性。

5.1.2 算法的特性

3. 有效性(可行性)

包括以下两个方面：

① 算法中每一个步骤必须能够实现，如在算法中不允许出现分母为0的情况。

② 算法执行的结果要能够达到预期的目的，实现预定的功能。

4. 输入数据与输出数据的要求

一个算法应该有0个或多个输入数据、有1个或多个输出数据。

5.2 常用算法

1. 递归算法

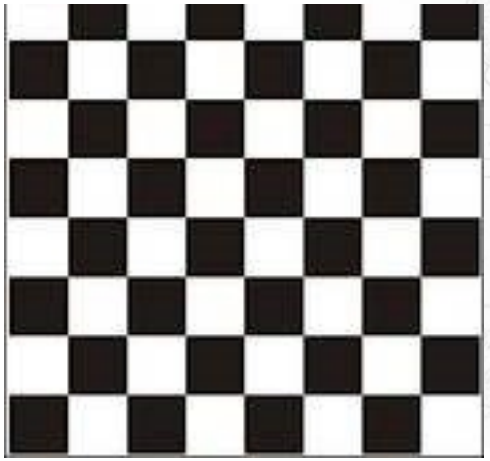
如果一个过程（函数、子程序）直接或间接地调用它本身，则称该过程（函数、子程序）是递归的。递归是设计和构造计算机算法的一种基本方法，递归过程必须存在一个递归终止条件，即存在一个“递归出口”，无条件的递归是毫无意义的。

递归算法()

递归(recursion)算法的一个典型例子：印度舍罕王打算奖赏国际象棋的发明人 - 宰相西萨·班·达依尔。国王问他想要什么，他对国王说：“陛下，请您在这张棋盘的第1个小格里赏给我一粒麦子，在第2个小格里给2粒，第3个小格给4粒，以后每一小格都比前一小格加一倍。请您把这样摆满棋盘上所有64格的麦粒，都赏给您的仆人吧！”国王觉得这个要求太容易满足了，就命令给他这些麦粒。当人们把一袋一袋的麦子搬来开始计数时，国王才发现：就是把全印度甚至全世界的麦粒全拿来，也满足不了那位宰相的要求。

那么，宰相要求得到的麦粒到底有多少呢？总数为：$1+2+2^2+ \ldots +2^{63}=2^{64}-1$

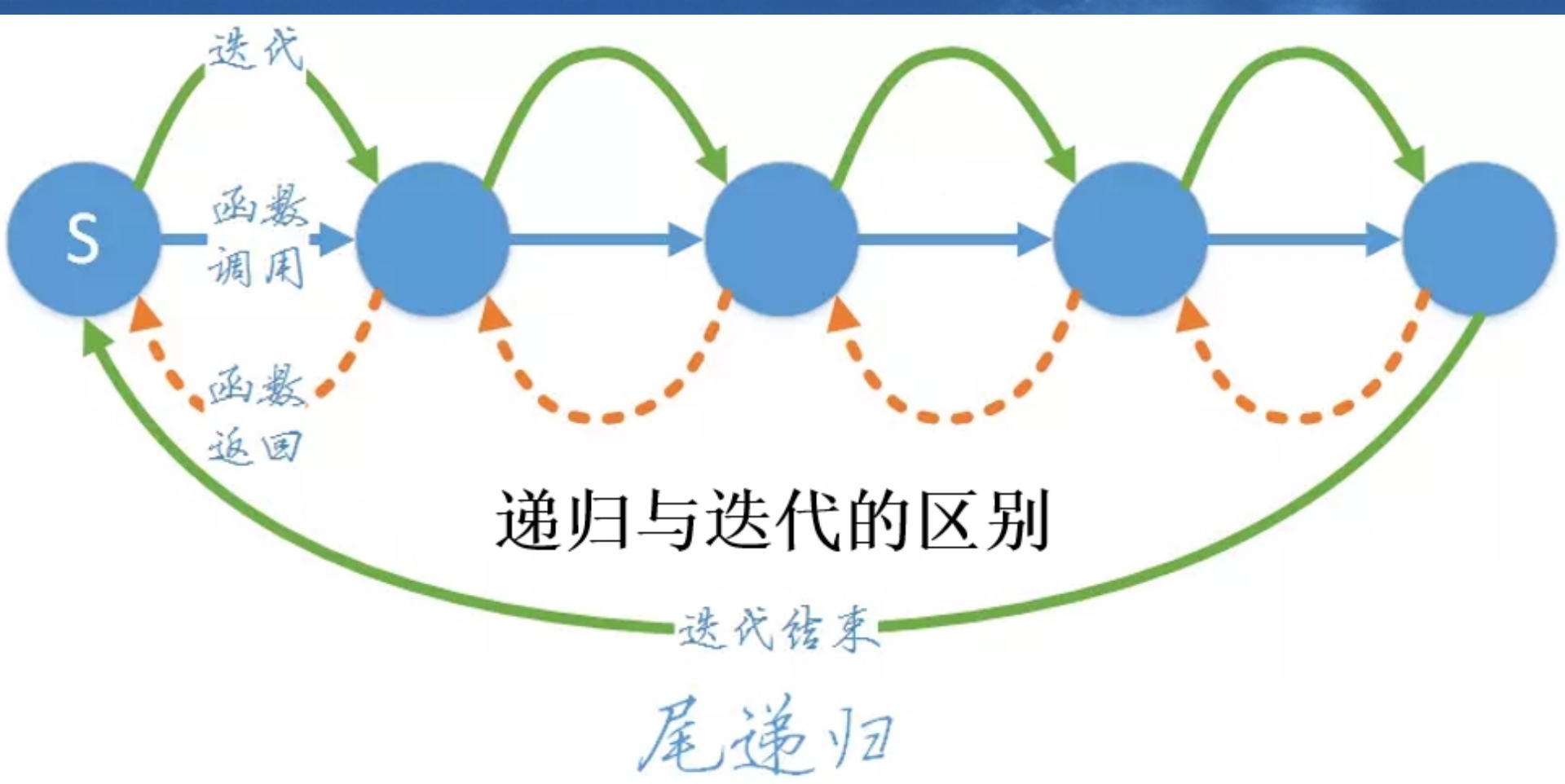
注意:理论上,递归算法都可以转换为非递归算法



5.2 常用算法

2. 迭代算法(iteration)

迭代是指重复执行一组指令或操作步骤，在每次执行这组指令时，都在原来的解的基础上推出一个新的解，新的解比原来的解值更加接近真实的解。这个过程不断重复，直到最后计算得到的解与真实解的误差满足实际要求。



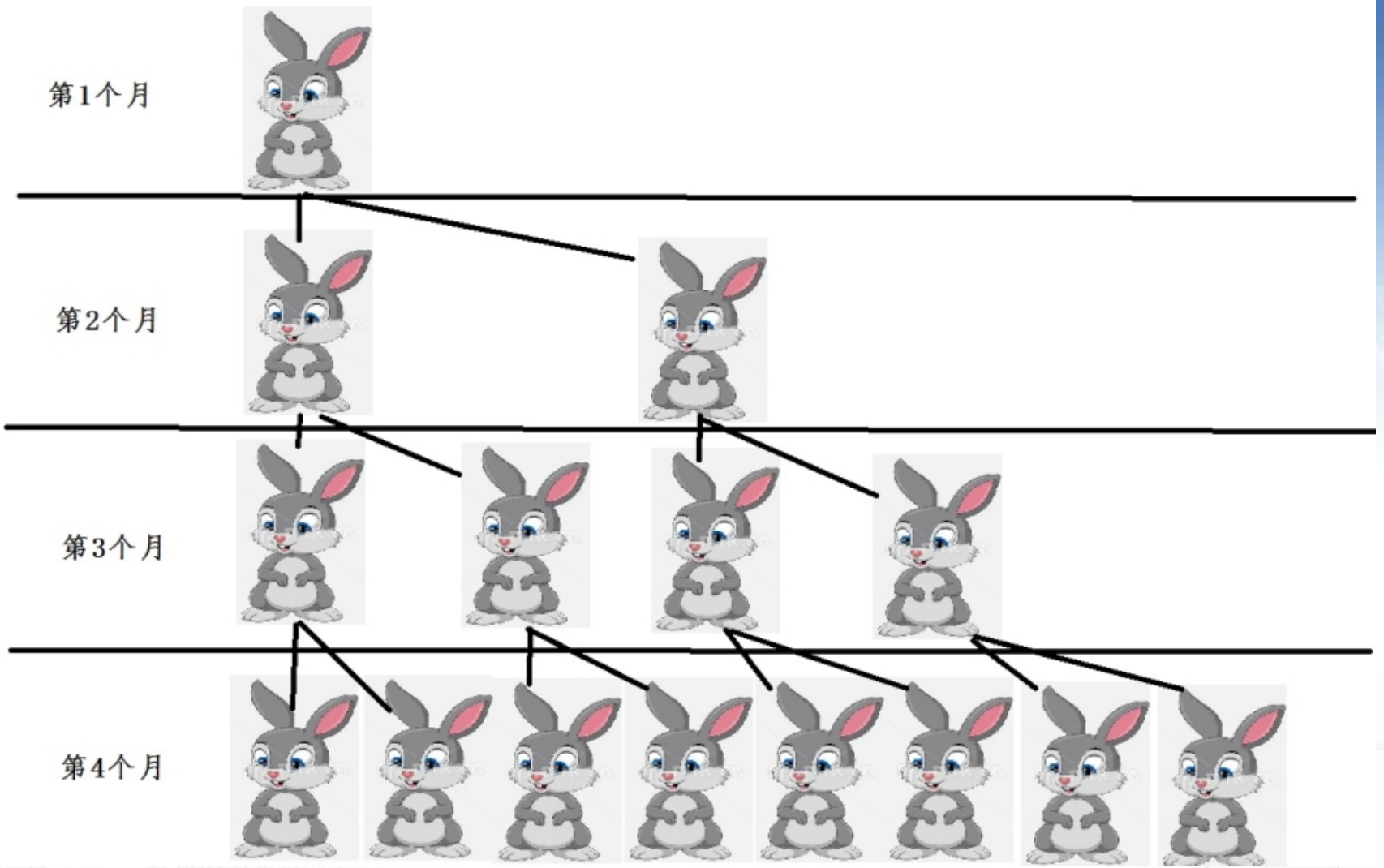
递归与迭代的区别

理论上递归和迭代可以相互转换，但实际从算法结构来说，递归声明的结构并不总能转换为迭代结构（原因有待研究）。迭代可以转换为递归，但递归不一定能转换为迭代

迭代算法

- 一个饲养场引进一只刚出生的新品种兔子，这种兔子从出生的下一个月开始，每月新生一只兔子，新生的兔子也如此繁殖。如果所有的兔子都不死去，问到第 12 个月时，该饲养场共有兔子多少只?
- 分析：这是一个典型的递推问题。不妨假设第1个月时兔子的只数为 u_1，第2个月时兔子的只数为u_2，第3个月时兔子的只数为u_3，……根据题意，“这种兔子从出生的下一个月开始，每月新生一只兔子”，则有

$$u_1 = 1,\ u_2 = u_1 + u_1 \times 1 = 2,\ u_3 = u_2 + u_2 \times 1 = 4, \cdots\cdots$$



问到第 12 个月时，该饲养场共有兔子多少只?

迭代算法

- 根据这个规律，可以归纳出下面的递推公式：

$$u_n = u_{(n-1)} \times 2 (n \geq 2)$$

- 对应 u_n和u_{n-1}，定义两个迭代变量 y 和 x，可将上面的递推公式转换成如下迭代关系：

```
y=x*2
x=y
```

- 初始化x=1,让计算机对这个迭代关系重复执行 11 次，就可以算出第 12 个月时的兔子数,最终结果保存在变量x里面。

```
for(int i=1;i<12;i++)
{   y=2*x;
    x=y;
}
```

5.2 常用算法

3. 穷举算法

穷举算法亦称枚举法，该算法首先根据问题的部分条件确定问题解的大致范围，然后在此范围内对所有可能的情况逐一进行验证，直到全部情况验证完毕。若某个情况使验证结果符合题目的条件，则为本题的一个答案；若全部情况验证完后均不符合题目的条件，则判定该问题无解。

- 判定素数算法
- a+b+c=1000

  $a^2+b^2=c^2$ 求自然数a,b,c

5.2 常用算法

4. 贪婪算法

贪婪算法也称贪心算法，是通过一系列的选择，最终得到问题的解。算法做出的每一个选择都是在当前状态下的最优选择。

贪婪算法(贪心算法)

- 贪婪算法通常具有贪婪选择性和最优子结构性。
- 贪婪选择性指的是所求解问题的整体最优解可以通过一系列局部最优的选择。贪婪算法所做的贪婪选择可以依赖以往所做过的选择，但不依赖于将来的选择，也不依赖于子问题的求解，通常采取自上向下的方法，以迭代方式做出贪婪选择，每一次选择都将问题简化为更小的子问题。
- 最优子结构性指的是一个问题的最优解往往包含着它的子问题的最优解。贪婪算法一般可以快速得到满意的解，因为它省去了为求最优解要穷尽所有可能而必须耗费的大量时间。

贪婪算法

假设顾客希望找回总额为16的硬币。同时假设银行发行的硬币面额是分别为1、5和10，那么按照贪婪算法，首先应该选取一枚面额为10的硬币，然后选择1枚面额为5的硬币，最后选择1枚面额为1的硬币。

Kruskal算法

是一种用来寻找最小生成树的算法,是贪心算法

.算法简单描述

1).记Graph中有v个顶点，e个边

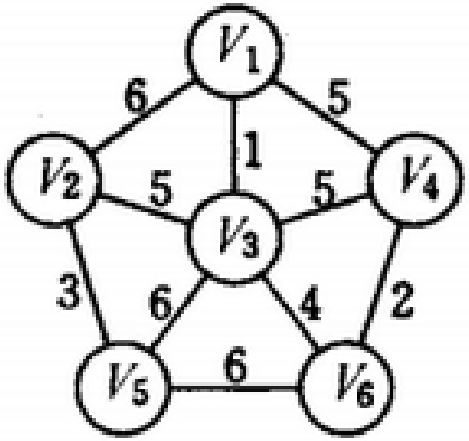
2).新建图$Graph_{new}$，$Graph_{new}$中拥有原图中相同的e个顶点，但没有边

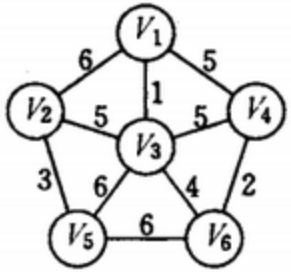
3).将原图Graph中所有e个边按权值从小到大排序

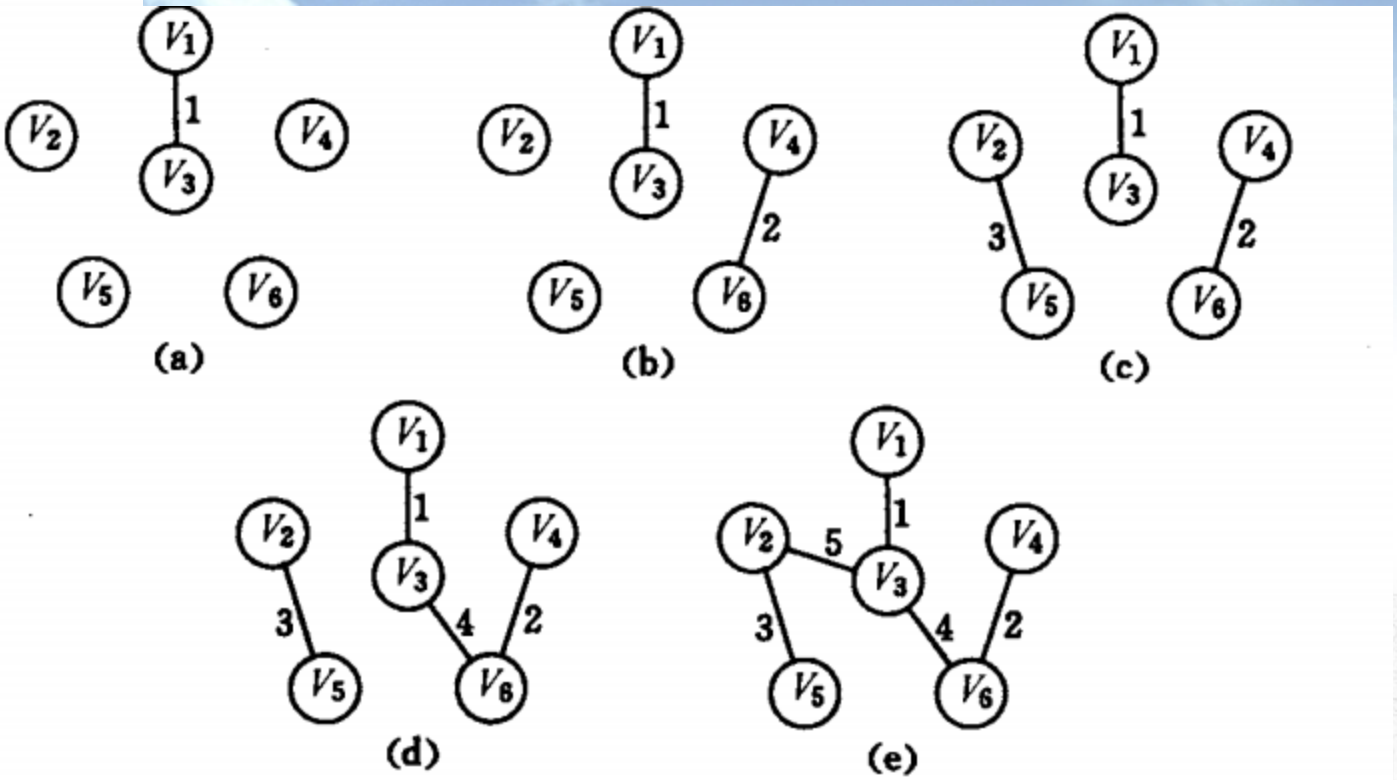
4).循环：从权值最小的边开始遍历每条边 直至图Graph中所有的节点都在同一个连通分量中

if 这条边连接的两个节点于图$Graph_{new}$中不在同一个连通分量中

添加这条边到图$Graph_{new}$中







克鲁斯卡尔算法构造最小生成树的过程

5.3 算法描述工具

算法是要通过程序才能加以实现的。常用的算法描述方式：

1. 自然语言

自然语言就是人们日常使用的语言，可以是中文、英文等。

例如，求3个数中最大者的问题，可以描述为：

① 比较前两个数。

② 将①中较大的数与第三个数进行比较。

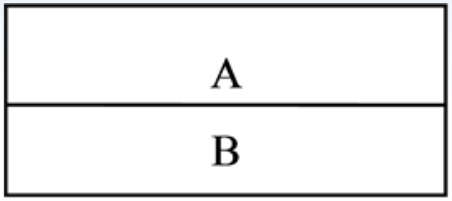
③ 步骤②中较大的数即为所求。

5.3 算法描述工具

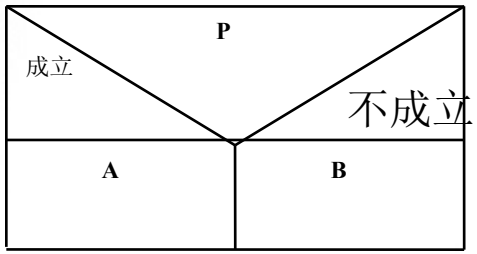
2. 流程图

流程图是用规定的一组图形符号、流程线和文字说明来描述算法的一种表示方法。

(1) 顺序结构。程序执行完A语句后接着执行B语句，如图所示。

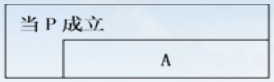


(2) 选择结构。当条件P成立时，则执行A语句，否则执行B语句，如图所示。

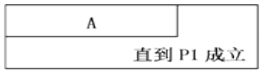


5.3 算法描述工具

(3) 当型循环结构。当条件P成立时，则循环执行A语句，如图所示



(4) 直到型循环结构。循环执行A语句，直到条件P1成立为止，如图所示。



5.3 算法描述工具

3. 伪代码

伪代码是用一种介于自然语言与计算机语言之间的文字和符号来描述算法，它比计算机语言形式灵活、格式紧凑，没有严格的语法。

例如，求两个数的较大者，用伪代码描述算法如下：

```
Find the bigger
Input: two number s:a,b
1.  if (the first number a is greater than or
equal to the  second number b)
    then
    1.1  return a
else
    1.2  return b
end if
    end
```

5.4 算法的评价

对于一个算法的评价，通常要从正确性、可理解性、健壮性、时间复杂度(Time Complexity)及空间复杂度(Space Complexity)等多个方面加以衡量。

1．算法的时间复杂度

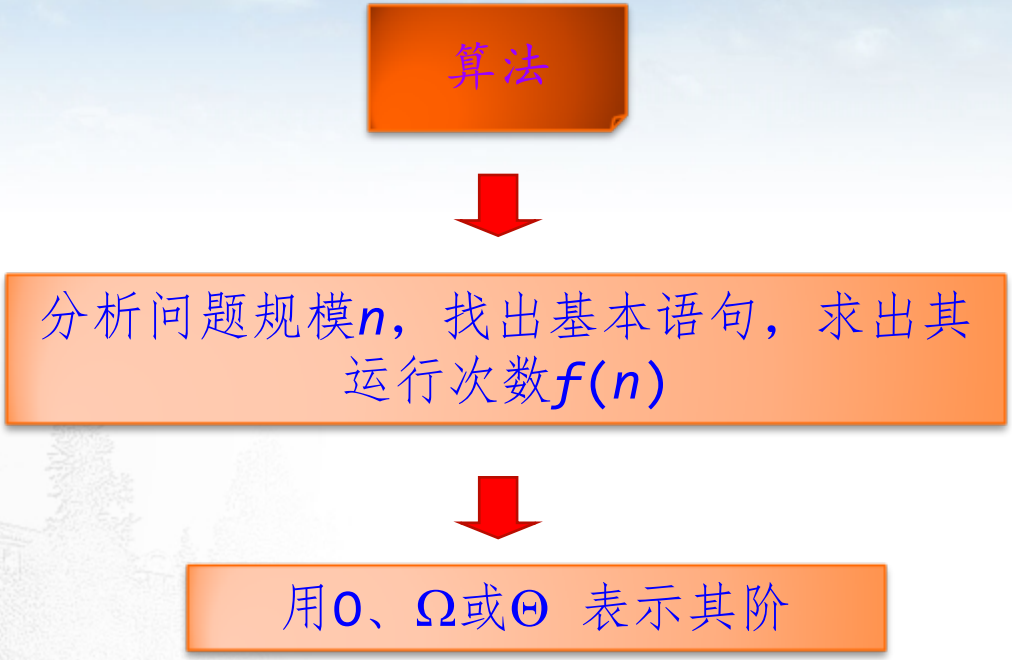
时间复杂度是度量时间的复杂性，即算法的时间效率的指标。

2．算法的空间复杂度

算法的空间复杂度是度量空间的复杂性，即执行算法的程序在计算机中运行时所占用空间的大小。

设n为算法中的问题规模，通常用大O、大Ω和Θ等三种渐进符号表示算法的执行时间与n之间的一种增长关系。

分析算法时间复杂度的一般步骤：



2．渐进符号（O、Ω和Θ）

定义1（大O符号），$f(n)=O(g(n))$（读作“$f(n)$是$g(n)$的大O”）当且仅当存在正常量c和n_0，使当$n \geq n_0$时，$f(n) \leq cg(n)$，即$g(n)$为$f(n)$的上界。

如$3n+2=O(n)$，因为当$n \geq 2$时，$3n+2 \leq 4n$。

$10n^2+4n+2=O(n^4)$，因为当$n \geq 2$时，$10n^2+4n+2 \leq 10n^4$。

大O符号用来描述增长率的上界，表示$f(n)$的增长最多像$g(n)$增长的那样快，也就是说，当输入规模为n时，算法消耗时间的最大值。这个上界的阶越低，结果就越有价值，所以，对于$10n^2+4n+2$，$O(n^2)$比$O(n^4)$ 有价值。

一个算法的时间用大O符号表示时，总是采用最有价值的$g(n)$表示，称之为“紧凑上界”或“紧确上界”。

一般地，如果$f(n)=a_m n^m+a_{m-1}n^{m-1}+\ldots+a_1 n+a_0$，有$f(n)=O(n^m)$。

定义2（大Ω符号），$f(n)=\Omega(g(n))$（读作“$f(n)$是$g(n)$的大Ω”）当且仅当存在正常量c和n_0，使当$n\geq n_0$时，$f(n)\geq cg(n)$，即$g(n)$为$f(n)$的下界。

如$3n+2=\Omega(n)$，因为当$n\geq 1$时，$3n+2\geq 3n$。

$10n^2+4n+2=\Omega(n^2)$，因为当$n\geq 1$时，$10n^2+4n+2\geq n^2$。

大Ω符号用来描述增长率的下界，表示$f(n)$的增长最少像$g(n)$ 增长的那样快，也就是说，当输入规模为n时，算法消耗时间的最小值。

与大O符号对称，这个下界的阶越高，结果就越有价值，所以，对于$10n^2+4n+2$，$\Omega(n^2)$比$\Omega(n)$ 有价值。一个算法的时间用大Ω符号表示时，总是采用最有价值的$g(n)$表示，称之为“紧凑下界”或“紧确下界”。

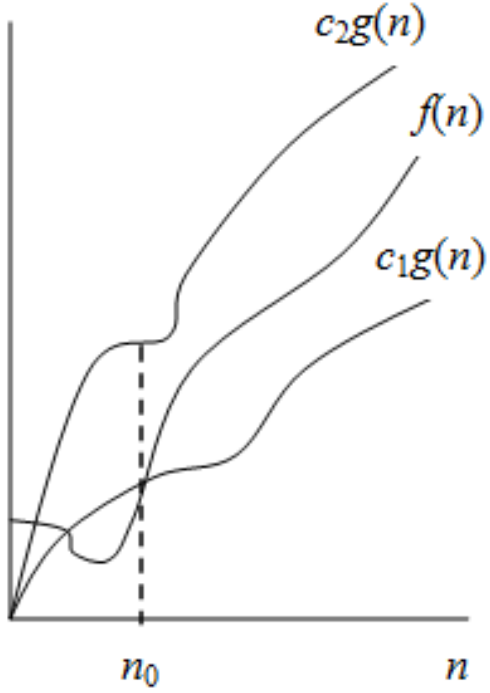
一般地，如果$f(n)=a_m n^m+a_{m-1}n^{m-1}+\ldots+a_1 n+a_0$，有$f(n)=\Omega(n^m)$。

定义3（大Θ符号），$f(n)=\Theta(g(n))$（读作“$f(n)$是$g(n)$的大Θ”）当且仅当存在正常量c_1、c_2和n_0，使当$n\geq n_0$时，有$c_1g(n)\leq f(n)\leq c_2g(n)$，即$g(n)$与$f(n)$的同阶。

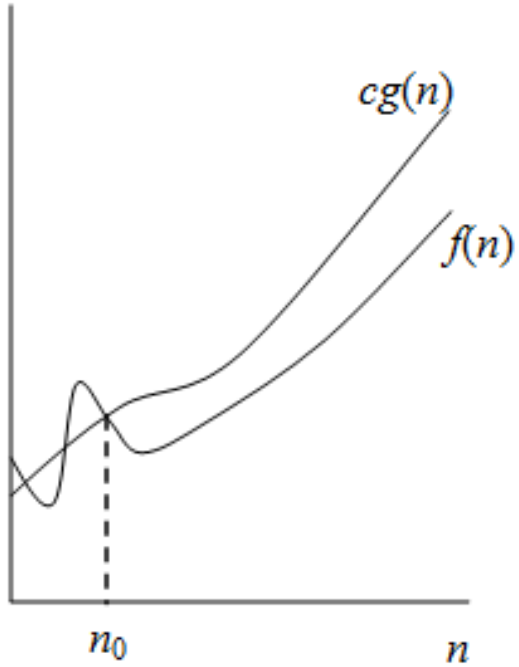
如$3n+2=\Theta(n)$，$10n^2+4n+2=\Theta(n^2)$。

一般地，如果$f(n)=a_mn^m+a_{m-1}n^{m-1}+\ldots+a_1n+a_0$，有$f(n)=\Theta(n^m)$。

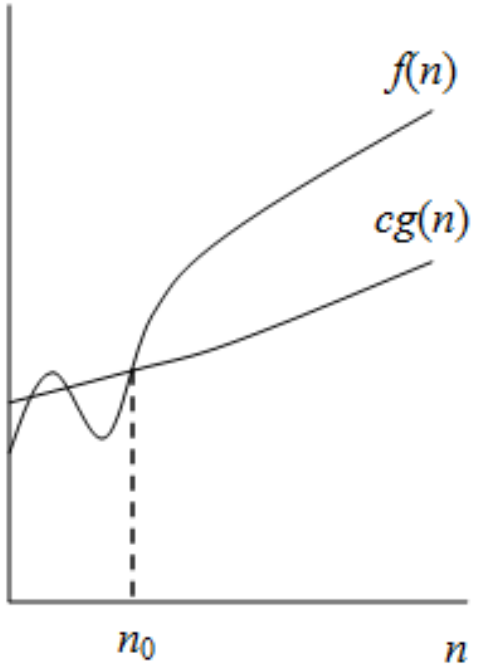
大Θ符号比大O符号和大Ω符号都精确，$f(n)=\Theta(g(n))$，当且仅当$g(n)$既是$f(n)$的上界又是$f(n)$的下界。



(a) $f(n)=\Theta(g(n))$



(b) $f(n)=O(g(n))$



(c) $f(n)=\Omega(g(n))$

【例】分析以下算法的时间复杂度：

```
void fun(int n)
{  int s=0,i,j,k;
   for (i=0;i<=n;i++)
       for (j=0;j<=i;j++)
            for (k=0;k<j;k++)
                s++;
}
```

解：该算法的基本语句是s++，所以有：

$$f(n)=\sum_{i=0}^{n}\sum_{j=0}^{i}\sum_{k=0}^{j-1}1=\sum_{i=0}^{n}\sum_{j=0}^{i}(j-1-0+1)=\sum_{i=0}^{n}\sum_{j=0}^{i}j$$

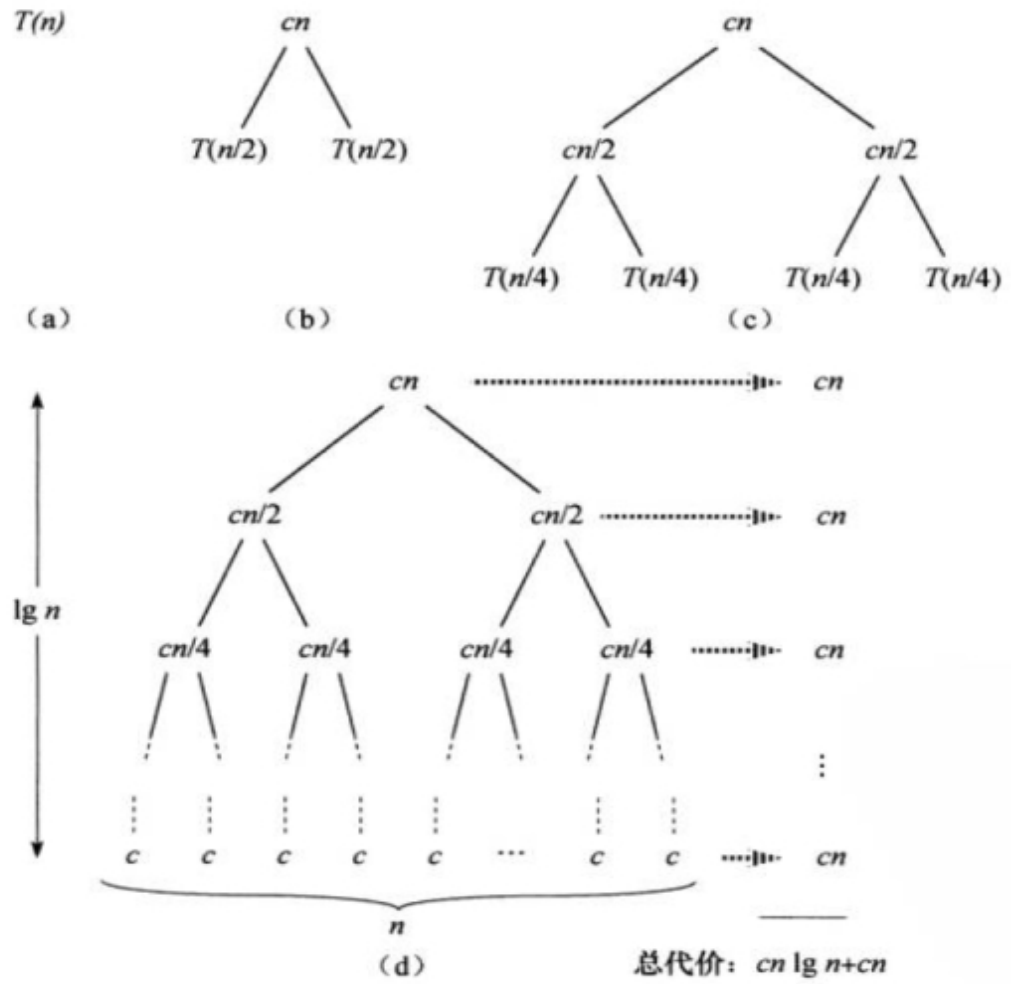
$$=\sum_{i=0}^{n}\frac{i(i+1)}{2}=\frac{1}{2}(\sum_{i=0}^{n}i^2+\sum_{i=0}^{n}i)=\frac{2n^3+6n^2+4n}{12}=O(n^3)$$

则该算法的时间复杂度为$O(n^3)$。

如果得到某一算法的递推公式

$$T(n)=\begin{cases} c & \text{若 } n=1 \\ 2T(n/2)+cn & \text{若 } n>1 \end{cases}$$

则可以按如下推导



5.5 算法设计策略

- 一个优秀的算法可以运行在比较慢的计算机上，但一个劣质的算法在一台性能很强的计算机上也不一定能满足应用的需要，因此，在计算机程序设计中，算法设计往往处于核心地位。
- 要想充分理解算法并有效地应用于实际问题中，关键是对算法的分析。通常可以利用实验对比分析、数学方法来分析算法。

分治策略

这种算法设计策略叫做分治法。 如果原问题可分割成k个子问题，1<k≤n，且这些子问题都可解，并可利用这些子问题的解求出原问题的解，那么这种分治法就是可行的。由分治法产生的子问题往往是原问题的较小模式，这就为使用递归技术提供了方便。在这种情况下，反复应用分治手段，可以使子问题与原问题类型一致而其规模却不断缩小，最终使子问题缩小到很容易直接求出其解。这自然导致递归过程的产生。分治与递归像一对孪生兄弟，经常同时应用在算法设计之中，并由此产生许多高效算法。 分治法所能解决的问题一般具有以下几个特征：

- 该问题的规模缩小到一定的程度就可以容易地解决；
- 该问题可以分解为若干个规模较小的相同问题，即该问题具有最优子结构性质。
- 利用该问题分解出的子问题的解可以合并为该问题的解；
- 该问题所分解出的各个子问题是相互独立的，即子问题之间不包含公共的子子问题
- 例如归并排序 它使用了分治策略

动态规划

动态规划程序设计是对解最优化问题的一种途径、一种方法，而不是一种特殊算法。不像搜索或数值计算那样，具有一个标准的数学表达式和明确清晰的解题方法。动态规划程序设计往往是针对一种最优化问题，由于各种问题的性质不同，确定最优解的条件也互不相同，因而动态规划的设计方法对不同的问题，有各具特色的解题方法，而不存在一种万能的动态规划算法，可以解决各类最优化问题。因此读者在学习时，除了要对基本概念和方法正确理解外，必须具体问题具体分析处理，以丰富的想象力去建立模型，用创造性的技巧去求解。我们也可以通过对若干有代表性的问题的动态规划算法进行分析、讨论，逐渐学会并掌握这一设计方法。

基本思想

- 动态规划算法通常用于求解具有某种最优性质的问题。在这类问题中，可能会有许多可行解。每一个解都对应于一个值，我们希望找到具有最优值的解。
- 动态规划算法与分治法类似，其基本思想也是将待求解问题分解成若干个子问题，先求解子问题，然后从这些子问题的解得到原问题的解。
- 与分治法不同的是，适合于用动态规划求解的问题，经分解得到子问题往往不是互相独立的。若用分治法来解这类问题，则分解得到的子问题数目太多，有些子问题被重复计算了很多次。
- 如果我们能够保存已解决的子问题的答案，而在需要时再找出已求得的答案，这样就可以避免大量的重复计算，节省时间。我们可以用一个表来记录所有已解的子问题的答案。不管该子问题以后是否被用到，只要它被计算过，就将其结果填入表中。这就是动态规划法的基本思路。具体的动态规划算法多种多样，但它们具有相同的填表格式。

用动态规划解决爬楼梯问题

问题描述:有一座高度是9级台阶的楼梯，从下往上走，每跨一步只能向上1级或者2级台阶。求出一共有多少种走法

台阶数	1	2	3	4	5	6	7	8	9
走法数	1	2							

台阶数	1	2	3	4	5	6	7	8	9
走法数	1	2	3						

台阶数	1	2	3	4	5	6	7	8	9
走法数	1	2	3	5					

5.6 分布式算法

- 分布式算法是用于解决多个互连处理器运行问题的算法。分布式算法的各部分并发和独立地运行，每一部分只承载有限的信息。即使处理器和通信信道以不同的速度运作，或即使某些构件出了故障，这些算法仍然能工作正常。
- 应用：如今天的电话系统、航班订票系统、银行系统、全球信息系统、天气预报系统以及飞机和核电站控制系统都依赖于分布式算法。

5.7 可计算性理论基础

- 研究计算的可行性和函数算法的理论，又称算法理论，是算法设计与分析的基础，也是计算机科学的理论基础。可计算性是函数的一个特性。

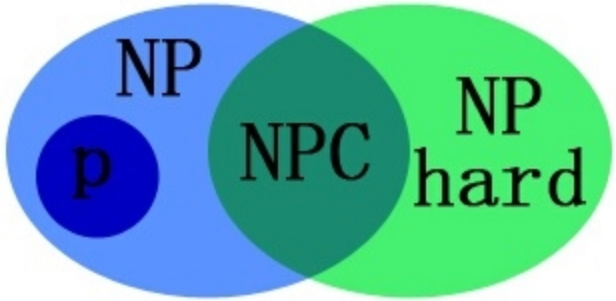
5.8 NP问题

NP(Non-deterministic Polynomial)问题是非确定性多项式问题，是指算法无法直接计算出结果，只能通过进行一些有选择的“猜算”来得到结果。

NP问题的研究结果有两种可能：

- 一种是找到了求解问题的算法；
- 另一种就是求解问题的算法是不存在的，那么就要从数学理论上证明它为什么不存在。

- P问题(**Polynomial**)：能在多项式时间内解决的问题；
- NP问题：（Nondeterministic Polynomial time Problem）不能在多项式时间内解决或不确定能不能在多项式时间内解决，但能在多项式时间内验证的问题；
- NPC问题(NP Complete)NP完全问题，所有NP问题在多项式时间内都能规约（Reducibility）到它的NP问题，即解决了此NPC问题，所有NP问题也都能得到解决；NPC包含了NP中最难的问题
- NP hard问题：NP难问题，所有NP问题在多项式时间内都能规约（Reducibility）到它的问题，但不一定是NP问题

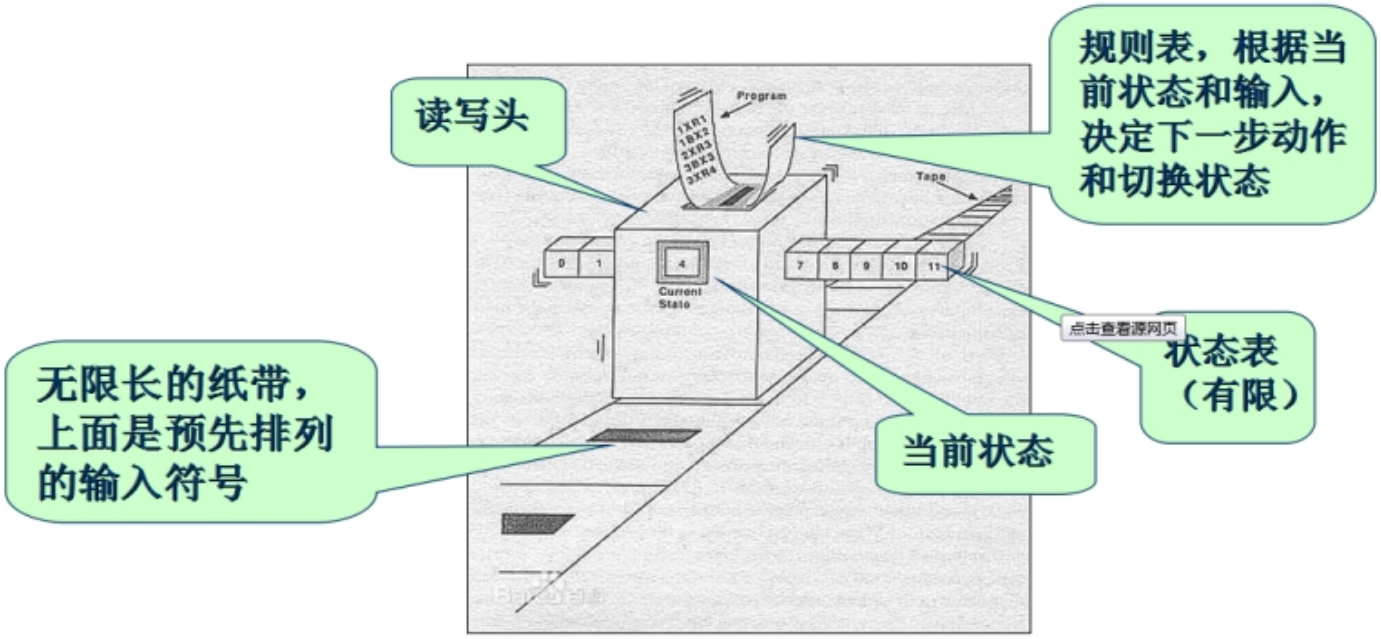


5.9 自动机理论

- 数理语言学中研究抽象自动机的理论。抽象自动机是一种能够识别语言的抽象装置，它不是具有物理实体的机器，而是表示计算机运算方式的抽象的逻辑关系系统，这样的抽象自动机可以用来检验输入的符号串是不是语言中合格的句子，如果是合格的句子，自动机就接收它，如果不是就不接收它。
- 自动机可分为有限自动机、后进先出自动机、线性有界自动机、图灵机等几种。它们对语言的识别能力各不相同。

图灵机，又称图灵计算、图灵计算机，是由数学家艾伦·麦席森·图灵（1912～1954）提出的一种抽象计算模型，即将人们使用纸笔进行数学运算的过程进行抽象，由一个虚拟的机器替代人们进行数学运算。
所谓的图灵机就是指一个抽象的机器，它有一条无限长的纸带，纸带分成了一个一个的小方格，每个方格有不同的颜色。有一个机器头在纸带上移来移去。机器头有一组内部状态，还有一些固定的程序。在每个时刻，机器头都要从当前纸带上读入一个方格信息，然后结合自己的内部状态查找程序表，根据程序输出信息到纸带方格上，并转换自己的内部状态，然后进行移动

邱奇，图灵和哥德尔提出了著名的邱奇-图灵论题：一切直觉上能行可计算的函数都可用图灵机计算，反之亦然



5.10 加密算法

- 加密一直伴着人们左右，想一下小时候智力测试的题目：“007收到遇害同事的字条，上面写着4FEFKKILJK81IP，根据事先约定，已知c=3，q=H，问该同事要传递什么信息给007？”该题放到信息技术方面来看待，就是一条经过简单的替换字符算法加密的字符串，它把原字符改为使用相对应的数字排序来替代，从而产生了一组“没有意义”的字符组合，但是因为知道c=3等条件，所以稍加排序即可得到原句：“Do not trust Hary”，这个过程也可称为“解密”（Decrypt）。

5.10 加密算法

- 数据加密的基本过程就是对原来为明文的文件或数据按某种算法进行处理，使其成为不可读的一段代码，通常称为“密文”。
- 该过程的逆过程为解密，即将该编码信息转化为其原来数据的过程。
- 加密技术通常分为“对称式”和“非对称式”两大类。
- 对称式加密就是加密和解密使用同一个密钥，
- 非对称式加密就是加密和解密所使用的不是同一个密钥，通常有两个密钥，称为“公钥”和“私钥”，它们两个必须配对使用，否则不能打开加密文件。

5.10 加密算法

常见加密算法有如下：

(1)DES(Data Encryption Standard)：数据加密标准，速度较快，适用于加密大量数据的场合。

(2)3DES(Triple DES)：是基于DES，对一块数据用3个不同的密钥进行3次加密，强度更高；

(3)RC2和RC4：用变长密钥对大量数据进行加密，比DES快。

(4)IDEA(International Data Encryption Algorithm)国际数据加密算法，使用128位密钥提供非常强的安全性。

(5)RSA：由RSA公司发明，是一个支持变长密钥的公共密钥算法，需要加密的文件块的长度也是可变的。

(6)DSA(Digital Signature Algorithm)：数字签名算法，是一种标准的DSS(数字签名标准)。

5.10 加密算法

常见加密算法有如下：

(7) MD5：Message Digest Algorithm MD5（中文名为消息摘要算法第五版）为计算机安全领域广泛使用的一种散列函数，用以提供消息的完整性保护。以512位分组来处理输入的信息，且每一分组又被划分为16个32位子分组，经过了一系列的处理后，算法的输出由4个32位分组组成，将这4个32位分组级联后将生成一个128位散列值。

2004年8月17日的美国加州圣巴巴拉的国际密码学会议（Crypto’ 2004）上，来自中国山东大学的王小云教授做了破译MD5、HAVAL-128、 MD4和RIPEMD算法的报告，公布了MD系列算法的破解结果。

5.10 加密算法

- 2004年8月17日的美国加州圣巴巴拉的国际密码学会议（Crypto’ 2004）上，来自中国山东大学的王小云教授做了破译MD5、HAVAL-128、MD4和RIPEMD算法的报告，公布了MD系列算法的破解结果。
- 2005年8月，王小云、姚期智，以及姚期智妻子姚储枫（即为Knuth起名高德纳的人）联手于国际密码讨论年会尾声部份提出SHA-1杂凑函数杂凑冲撞演算法的改良版。此改良版使破解SHA-1时间缩短。
- 2009年，冯登国、谢涛二人利用差分攻击，将MD5的碰撞算法复杂度从王小云的2^{42}进一步降低到2^{21}，极端情况下甚至可以降低至2^{10}。

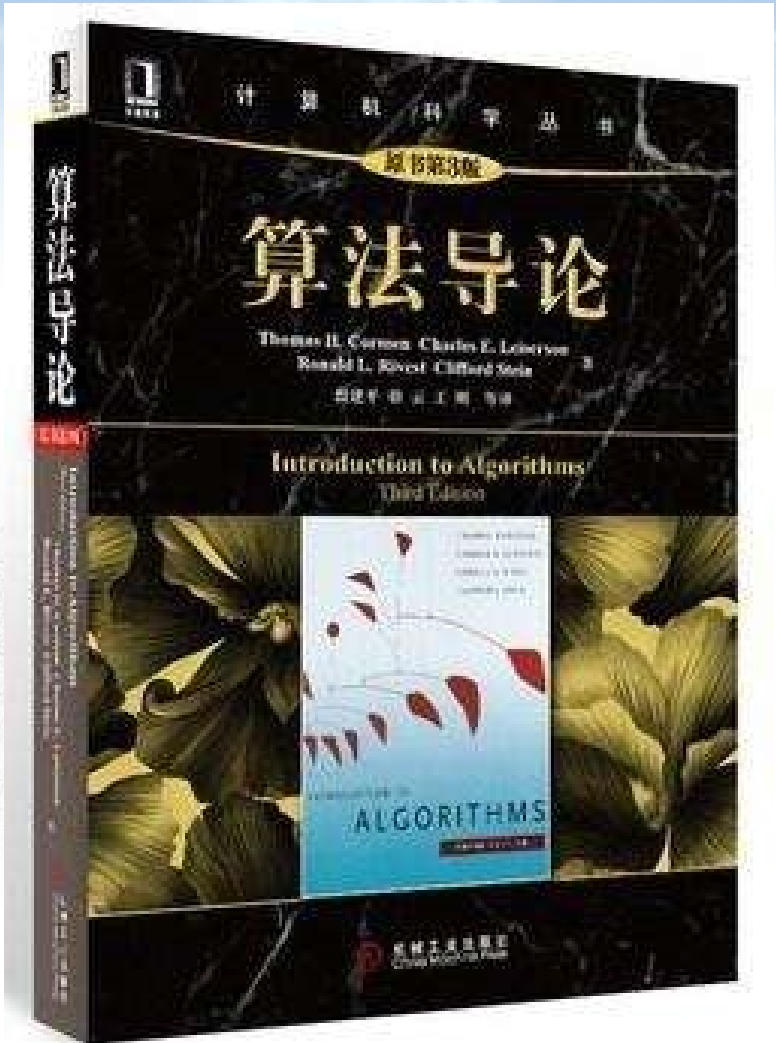
5.11 几何算法

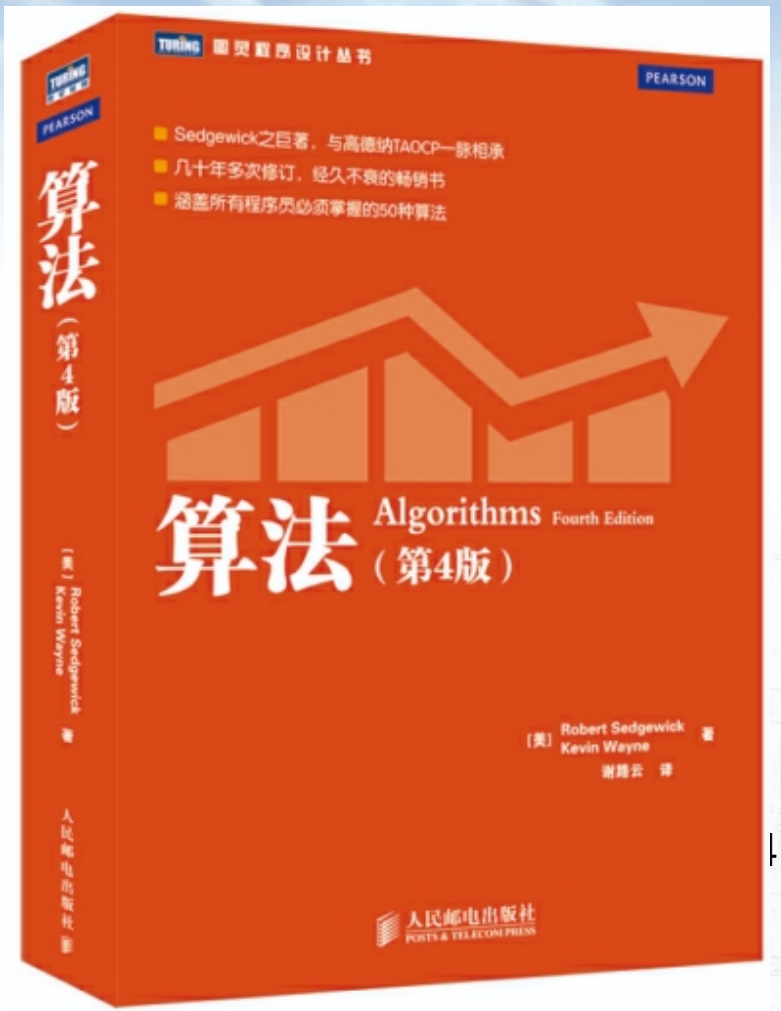
- 几何算法的内容不属于欧几里德的几何证明公理化范畴，而是属于欧几里德几何构造，即由算法和复杂性构成。
- 在一个几何构造过程中，执行原始运算的总次数称为该过程的复杂性度量，这个概念对应于算法的时间复杂度。同样，还有对应于算法的空间复杂度的概念。这是欧几里德几何构造过程复杂性的定量测度。

5.12 并行算法

- 并行算法是在给定并行模型下的一种具体、明确的计算方法和步骤，其有不同的分类方法。
- 根据并行计算任务的大小分类，可以分为粗粒度并行算法、中粒度并行算法和细粒度并行算法3类。
- 根据并行计算的基本对象可分为数值并行计算和非数值并行计算。
- 根据并行计算进程间的依赖关系，可以分为同步并行算法和异步并行算法。
- 一个高效的并行算法设计过程比较复杂。一般编程设计过程可以分为任务划分、通信分析、任务组合和处理器映射4步。

算法经典教材


计 算 机 科 学 丛 书
原书第3版
算法导论
Introduction to Algorithms
Third Edition
ALGORITHMS


TURING 图灵程序设计丛书
PEARSON
Sedgewick之巨著，与高德纳TAOCP一脉相承
几十年多次修订，经久不衰的畅销书
涵盖所有程序员必须掌握的50种算法
算法（第4版）
Algorithms Fourth Edition
算法（第4版）
[美] Robert Sedgewick
Kevin Wayne
著
谢路云 译
人民邮电出版社
POSTS & TELECOM PRESS

本章小结

- 本章介绍了算法的概念和特性，常用算法的应用，算法描述的工具和评价方式，算法设计的策略以及分布式算法、NP问题引入、自动机理论，加密、几何和并行三种算法的具体概念。
- 通过本章的学习，读者应该了解算法的3种表示方式，即自然语言、流程图和伪代码，熟悉递归、迭代、穷举和贪婪四种常用算法的实现过程，能够自然的描述出算法，对算法在时间和空间复杂度进行计算和评价，对加密、几何和并行算法有初步的认识。

谢谢,同学们再见!