数据结构与算法（Python版）

算法和计算复杂性

陈斌 北京大学 gischen@pku.edu.cn

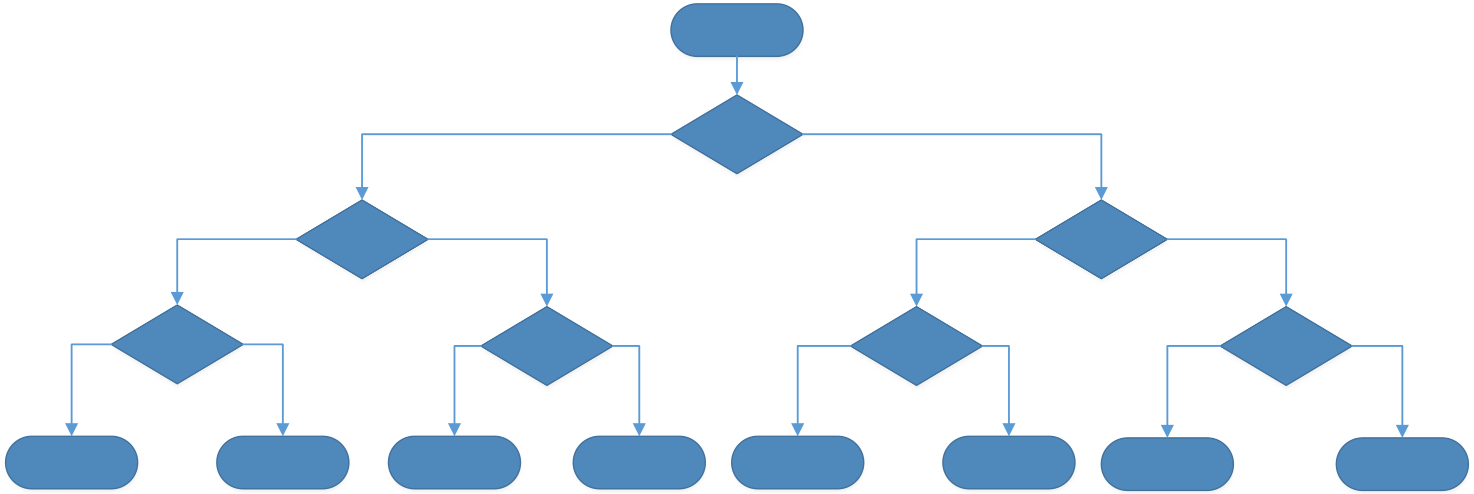
问题的分类

- **What：是什么？**
  面向判断与分类的问题；
- **Why：为什么？**
  面向求因与证明的问题；
- **How：怎么做？**
  面向过程与构建的问题。

可以通过"计算"解决的问题

- 用任何一个"有限能行方法"下的计算模型可以解决的问题，都算是"可计算"的
- What：分类问题

可以通过树状的判定分支解决



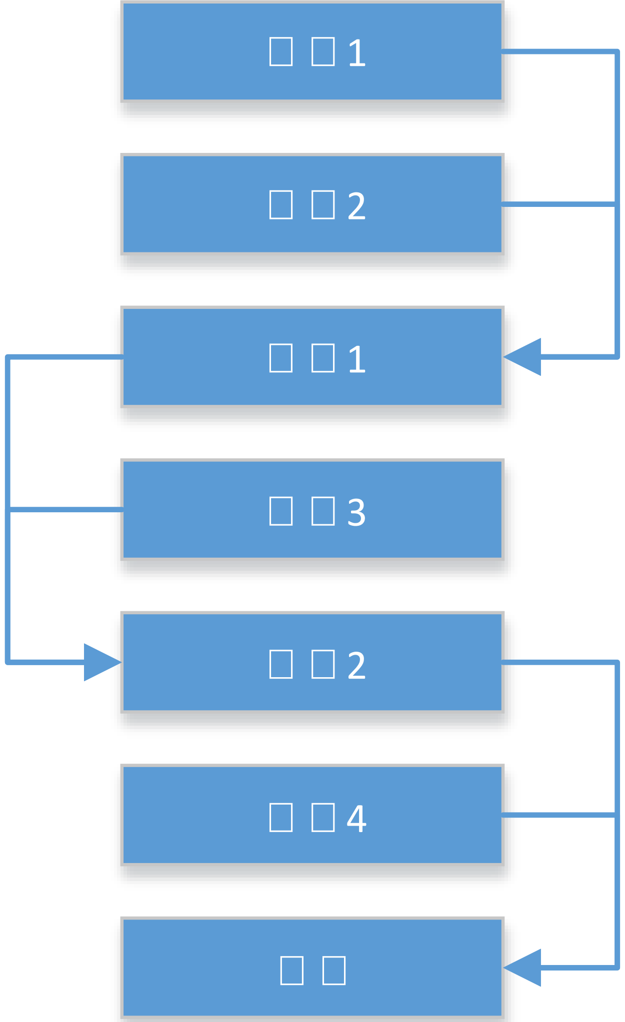
可以通过“计算”解决的问题

❖ Why：证明问题

可以通过有限的公式序列来解决

数学定理证明从不证自明的公理出发，一步步推理得出最后待证明的定理

我们在以往学习过的定理证明即为此类解决方法

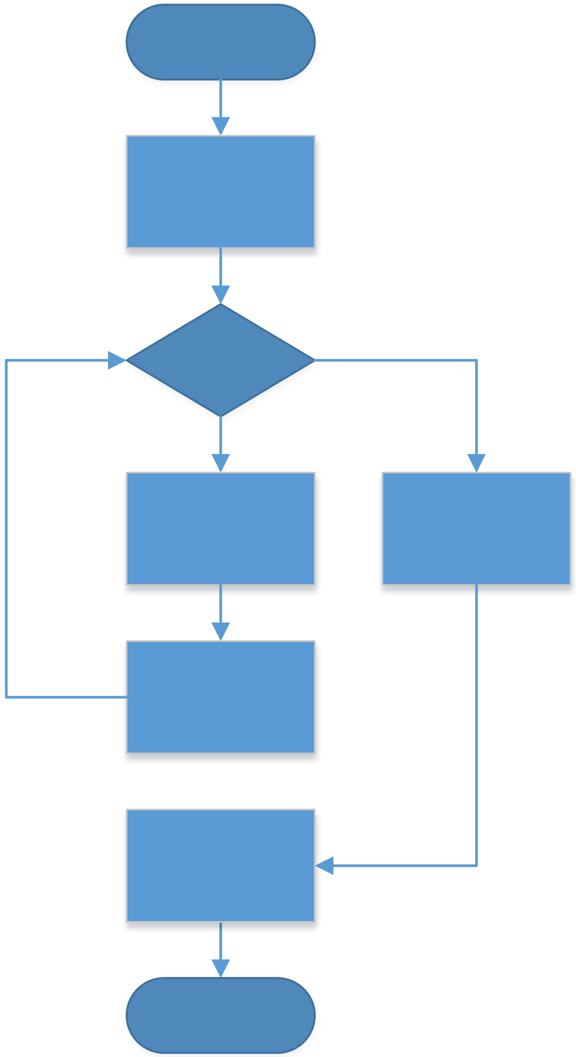


可以通过“计算”解决的问题

❖ **How：过程问题**

可以通过算法流程来解决

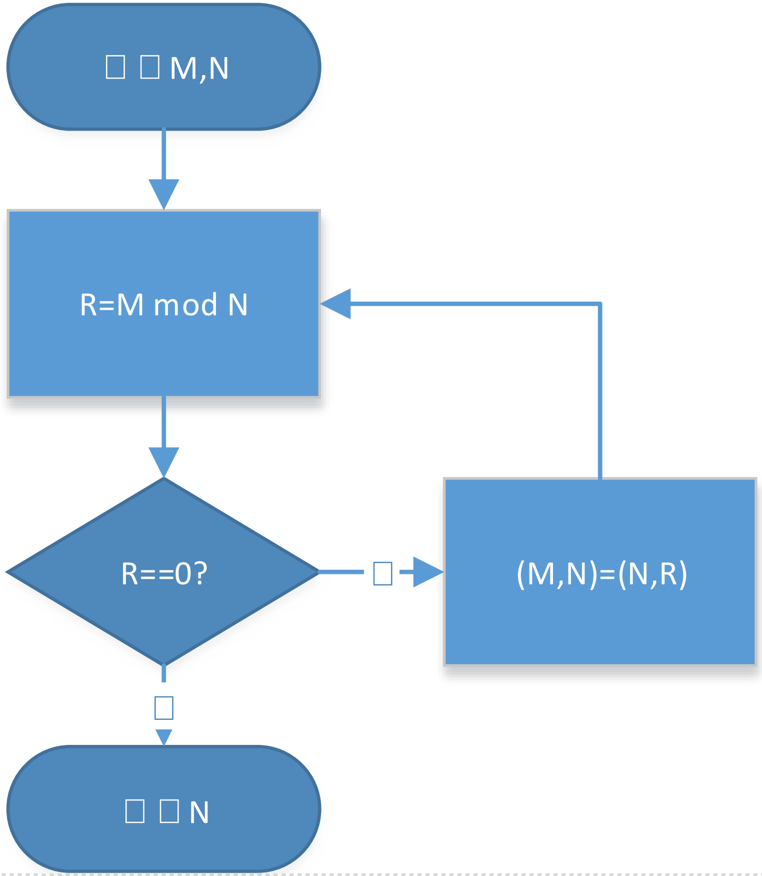
解决问题的过程：算法和相应数据结构的研究，即为本课主要内容



世界上最早的算法：欧几里德算法

- 公元前3世纪，记载于《几何原本》

辗转相除法求最大公约数



世界上最早的算法：欧几里德算法

- ❖ **辗转相除法处理大数时非常高效**
- ❖ **它需要的步骤不会超过较小数位数的5倍**

加百利·拉梅(Gabriel Lame)于1844年证明了这个结论

并开创了计算复杂性理论。

计算复杂性

- ❖ “基于有穷观点的能行方法”的“可计算”概念

仅仅涉及到问题的解决是否能在有限资源内（时间/空间）完成

并不关心具体要花费多少计算步骤或多少存储空间

计算复杂性

- 由于资源（时间/空间）相当有限，对于问题的解决需要考虑其**可行性**如何
- 人们发现各种不同的可计算问题，其难易程度是不一样的

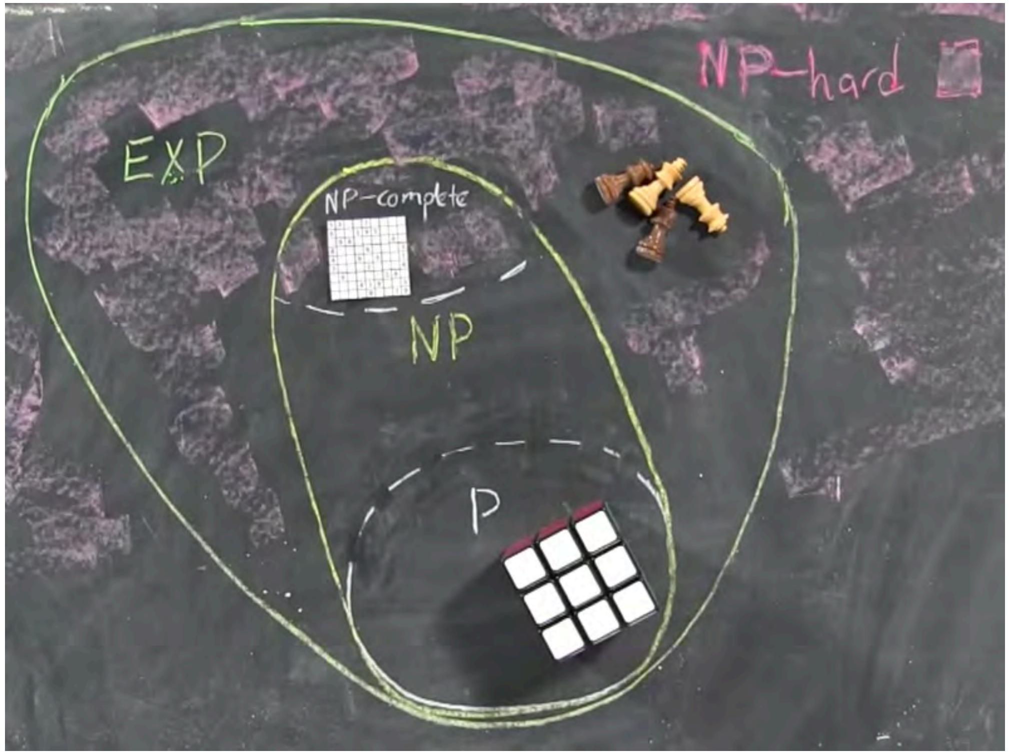
有些问题**非常容易解决**，如基本数值计算；

有些问题的解决程度**尚能令人满意**，如表达式求值、排序等；

有些问题的解决会爆炸性地吞噬资源，虽有解法，但**没什么可行性**，如哈密顿回路、货郎担问题等

计算复杂性

❖ 定义一些衡量指标，对问题的难易程度（所需的执行步骤数/存储空间大小）进行**分类**，是计算复杂性理论的研究范围



计算复杂性

- 但对于同一个问题，也会有不同的解决方案，其解决效率上也是千差万别
- 如排序问题，以n张扑克牌作为排序对象

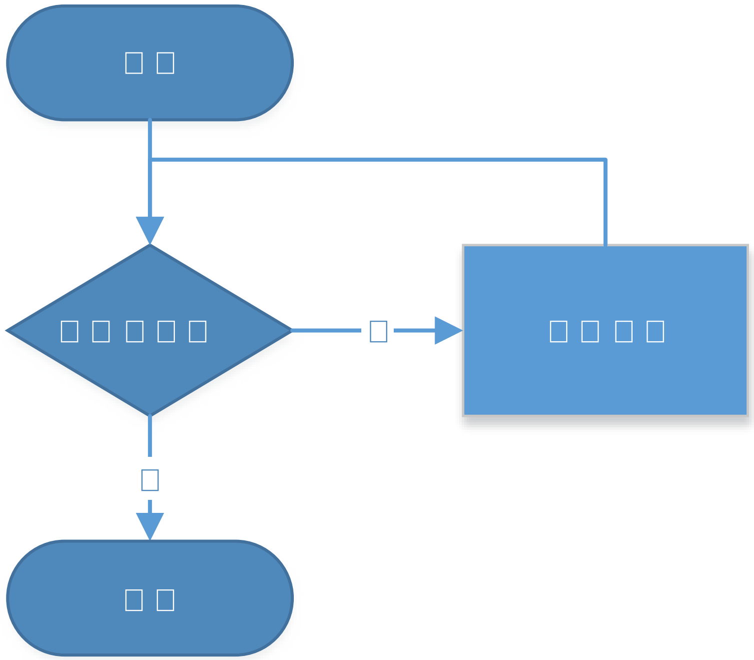
  一般人们会想到的是“冒泡”排序，即每次从牌堆里选出一张最小的牌，这样全部排完大概会需要n^2量级的比较次数

计算复杂性

❖ 另一种有趣的“Bogo”排序方法 😜

洗一次牌，看是否排好序，没有的话，接着洗牌，直到排序成功！

这样全部排完，平均需要n*n!量级的比较次数



计算复杂性与算法

- 计算复杂性理论研究问题的本质，将各种问题按照其难易程度**分类**，研究各类问题的难度**级别**，并不关心解决问题的具体方案
- 而算法则研究问题在不同现实**资源约束**情况下的不同解决方案，致力于找到**效率**最高的方案

  不同硬件配置（手持设备、PC设备、超级计算机）

  不同运行环境（单机、多机环境、网络环境、小内存）

  不同应用领域（消费、工业控制、医疗系统、航天领域）

  甚至不同使用状况（正常状况、省电状况）

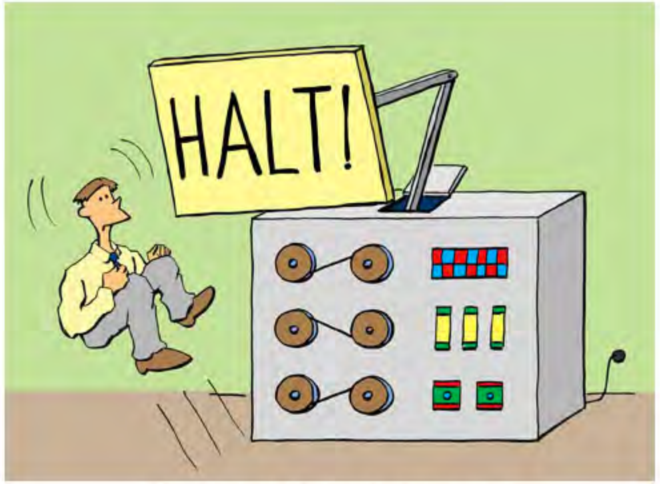
- 如何对具体的算法进行分析，并用衡量指标评价其复杂度，我们在后面的课程里还会详细介绍

不可计算问题

- **有不少定义清晰，但无法解决的问题**

  并不是尚未找到解，而是在“基于有穷观点的能行方法”的条件下，已被证明并不存在解决方案

- **停机问题：判定任何一个程序在任何一个输入情况下是否能够停机**



Alan designed the perfect computer

不可计算问题

- 不可计算数：几乎所有的无理数，都无法通过算法来确定其任意一位是什么数字

  可计算数很少：如圆周率Pi，自然对数的底e

- 似乎计算之道解决问题存在边界和极限？

$$e=\sum_{n=0}^{\infty}\frac{1}{n!}=\frac{1}{0!}+\frac{1}{1!}+\frac{1}{2!}+\frac{1}{3!}+\frac{1}{4!}+\cdots$$

$$\pi=\frac{1}{2^6}\sum_{n=0}^{\infty}\frac{(-1)^n}{2^{10n}}\left(-\frac{2^5}{4n+1}-\frac{1}{4n+3}+\frac{2^8}{10n+1}-\frac{2^6}{10n+3}-\frac{2^2}{10n+5}-\frac{2^2}{10n+7}+\frac{1}{10n+9}\right)$$