期中回顾

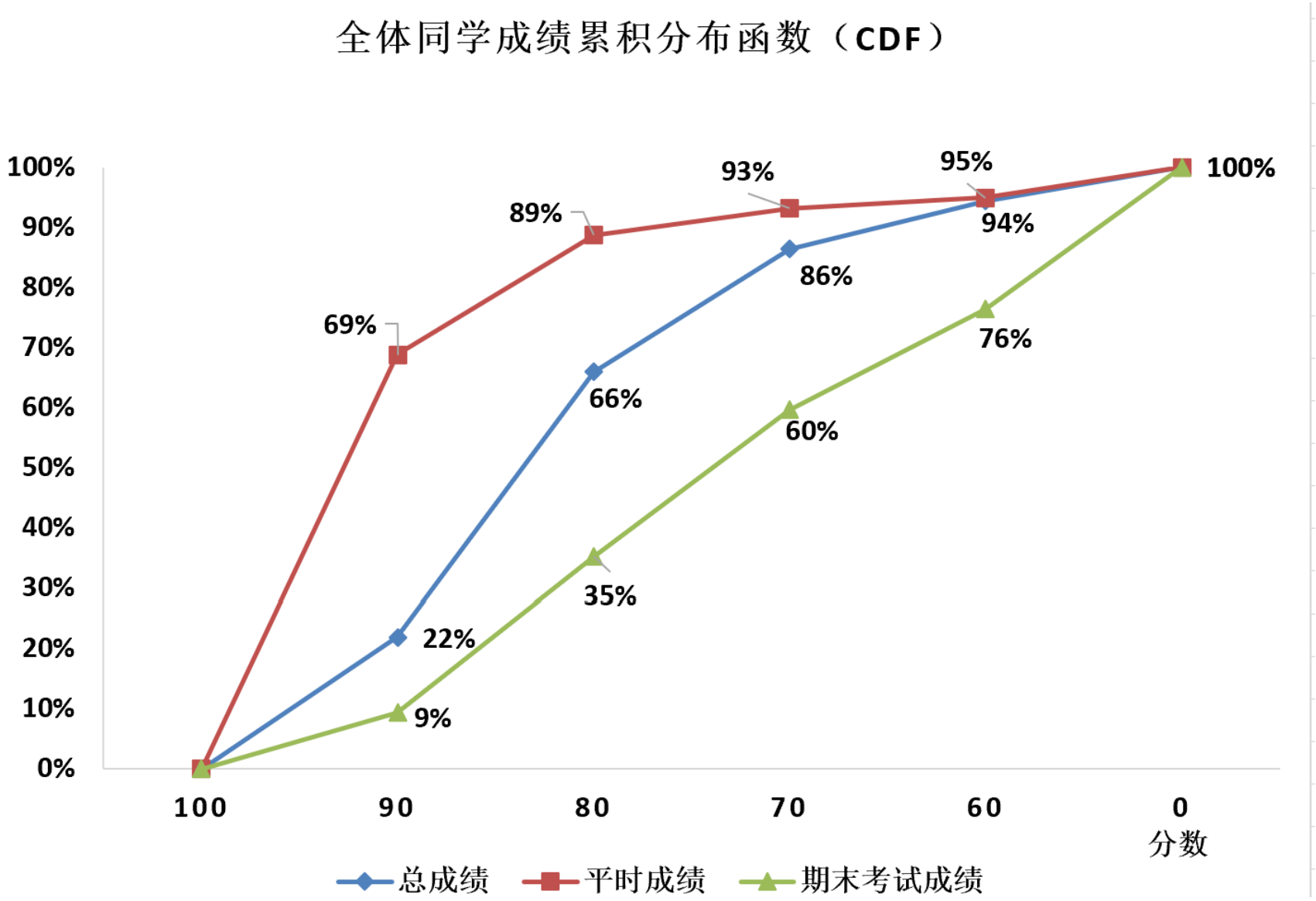
徐志伟 zxu@ict.ac.cn

提纲

- 同学们已经取得了长足的进步
- 逻辑思维回顾
- 图灵机的通用性与局限性
- 什么是构造性？
- 算法思维妙在何处？

1. 同学们已经取得了长足的进步

- 掌握了入门知识
  - 数百个基本词汇
  - 数十个基本概念
  - 十几个基本方法
- 体认了计算思维入门基础
  - 计算思维基本方法：活力法（PEPS）
  - 计算思维十个理解
- 能够阅读编写特定场景的入门程序
  - 十几行的汇编语言程序
  - 数十行的图灵机程序
  - 数十行的Go程序



2022年同学成绩情况

1.1 同学们已经学到了不少知识

比照《计算机科学基础报告》

- “计算机科学是研究计算机以及它们能干什么的一门学科。它研究**抽象计算机**的能力与局限，**真实计算机**的构造与特征，以及用于求解问题的无数**计算机应用**。”
  - 计算机科学涉及**符号**及其操作；
  - 关注多种**抽象**概念的创造和操作；
  - 创造并研究**算法**；
  - 创造各种**人工结构（作品）**，尤其是不受物理定律限制的结构；
  - 利用并应对**指数增长**；
  - 探索计算能力的**基本极限**；
  - 关注与人类**智能**相关的复杂的、分析的、理性的活动。

知道概念，能够举例说明其特点

笔记本电脑

有限状态机、图灵机 、
冯诺依曼模型

- “计算机科学是研究计算机以及它们能干什么的一门学科。它研究**抽象计算机**的能力与局限，**真实计算机**的构造与特征，以及用于求解问题的无数**计算机应用**。”
  - 计算机科学涉及**符号**及其操作；
  - 关注多种**抽象**概念的创造和操作；
  - 创造并研究**算法**；
  - 创造各种**人工结构（作品）**，尤其是不受物理定律限制的结构；
  - 利用并应对**指数增长**；
  - 探索计算能力的**基本极限**；
  - 关注与人类**智能**相关的复杂的、分析的、理性的活动。

快速排序程序，分治

各种编程抽象

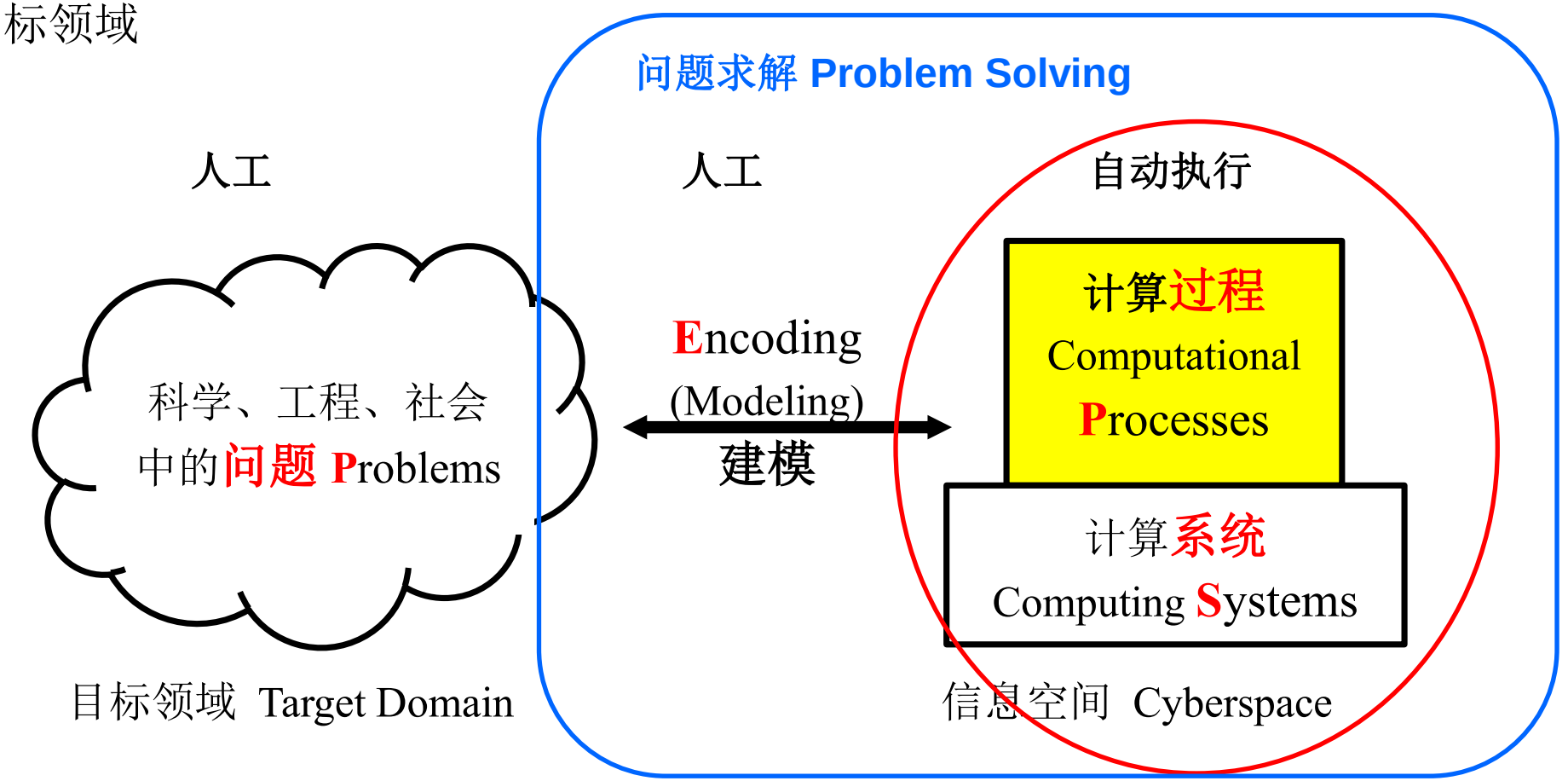
逻辑推理
命题逻辑
谓词逻辑

斐波那契递归程序
P与NP

图灵机不可计算问题
哥德尔不完备定理

1.2 计算思维的基本思路：活力法（PEPS）

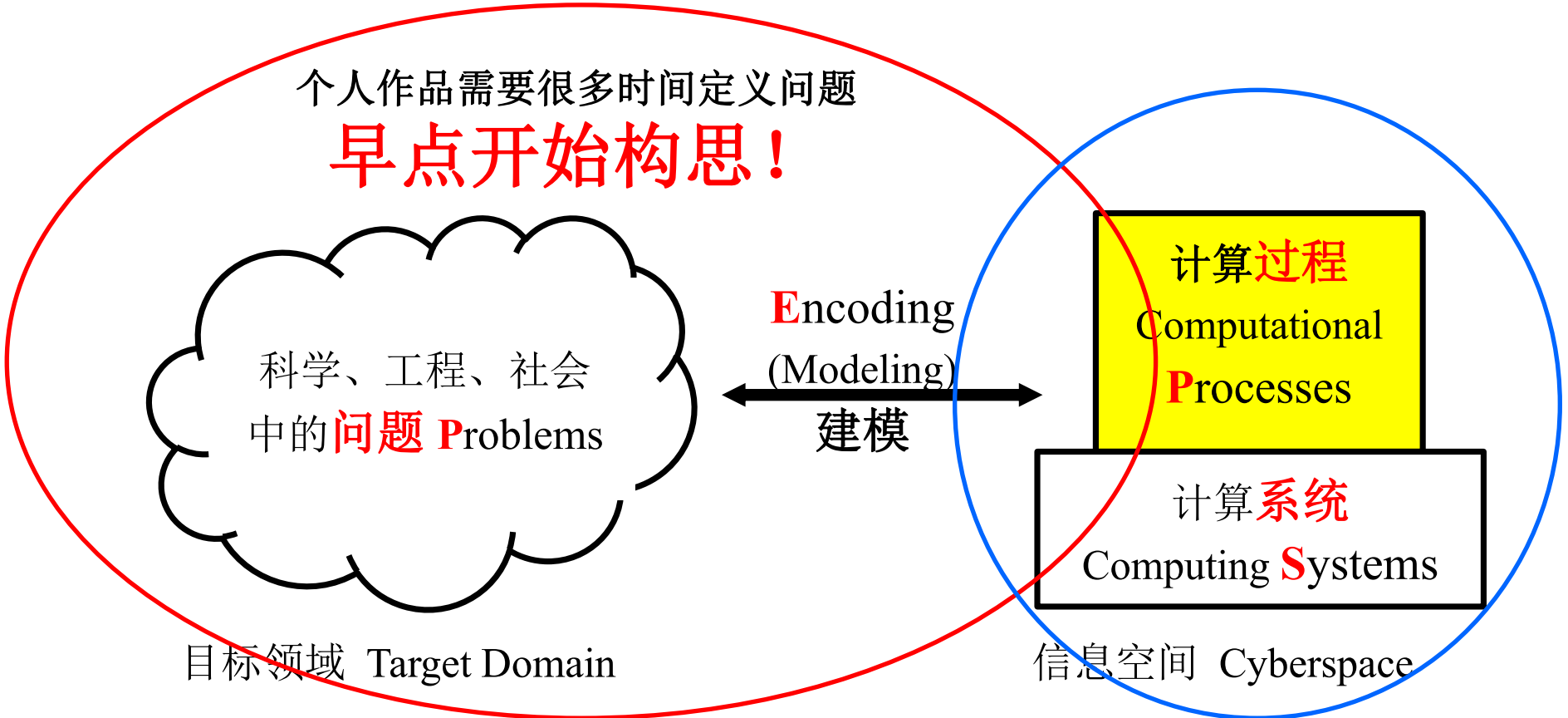
- 定义领域问题（Problem in target domain）
- 建模（Encoding）到信息空间（赛博空间，cyberspace）的计算问题
- 自动执行计算系统（System）上的计算过程（Process），解决问题
  - 计算系统 = 计算机；算法、程序、状态转移表等刻画计算过程
- 映射回到目标领域



实践中的PEPS（问题-建模-计算过程-系统）

- **P**roblem, 目标领域中的待解问题
- **E**ncoding, 建模
- Computational **P**rocess, 计算过程
- Computing **S**ystem, 计算系统

精力占比	P	E	P	S
斐波那契	1%	1%	78%	20%
个人作品	40%	10%	30%	20%



1.3 对计算思维的十个理解： Acu-Exams-CP

期中已经学习了五个理解

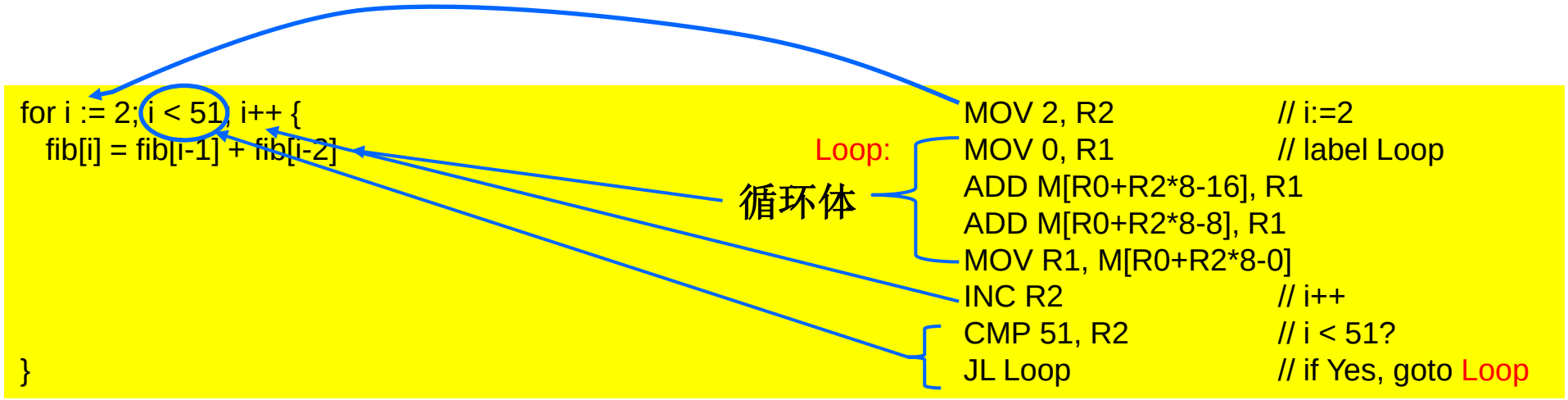
(A): Automatically execution	**自动执行：**计算机能够自动执行由离散步骤组成的计算过程。
(C): Correctness	**正确性：**计算机求解问题的正确性可以比特精准地定义并分析。
(U): Universality	**通用性：**计算机能够求解任意可计算问题。
(E): Effectiveness	**构造性：**人们能够构造出聪明的方法让计算机有效地解决问题。
(X): Complexity	**复杂度：**这些聪明的方法（算法）具备时间/空间复杂度。
(A): Abstraction	**抽象化：**少数精心构造的计算抽象可产生万千应用系统。
(M): Modularity	**模块化：**多个模块有规律地组合成为计算系统。
(S): Seamless Transition	**无缝衔接：**计算过程在计算系统中流畅地执行。
(C): Connectivity	**连接性：**很多问题涉及用户/数据/算法的连接体，而非单体。
(P): Protocol Stack	**协议栈：**连接体的节点之间通过协议栈交互。

1.4 四个实验的分工配合与要点

- 编程入门。在真实计算机上理解应用程序设计与执行
- 加法图灵机。用抽象计算机的“汇编语言”实现循环
  - 此处的汇编语言是状态转移表
- 信息隐藏。将文本文件隐藏到图像文件中的计算机应用
  - 单机应用程序如何定位到某个文件的每个数据的位置，并操作该数据
    - 要点：深入理解base+index*8+offset 寻址模式如何支持循环，举一反三推广到本实验
- 个人作品。创作+实现计算机应用（动态网页）
  - 网络应用程序如何定位到某个网页元素，并操作该元素
    - 要点：构思作品；自学网页文档结构（DOM），超链接，getElementById

多条指令如何支持循环

- 以及数组
  - 在科学计算中，循环和数组往往配套出现
- 注意
  - 汇编代码如何反映了循环之变与不变
  - 数组与循环如何配合



掌握循环与数组

- 多条指令如何支持循环与数组
  - 循环和数组往往配套出现
- 斐波那契计算机例子
  - 汇编代码如何反映了循环之变与不变
  - 数组与循环如何配合
  - 要点：计算机提供了寻址模式
    - Address = Base + Index*8 + Offset

有的同学的方法

- 为理解“循环之变与不变”，将循环展开

fib[2] = fib[1] + fib[0]
fib[3] = fib[2] + fib[1]
fib[4] = fib[3] + fib[2]
…
fib[50] = fib[49] + fib[48]

- 再深入考察头几个迭代的赋值语句的指令对应

基址寄存器R0=24，索引寄存器R2=2，比例因子=8；
通过累加操作（R1用做累加寄存器）

```
0 → R1
R1+fib[0]→R1
R1+fib[1]→R1
R1 → fib[2]
```

将赋值语句fib[2] = fib[1] + fib[0]编译成

```
MOV 0, R1
ADD M[R0+R2*8-16], R1
ADD M[R0+R2*8-8], R1
MOV R1, M[R0+R2*8-0]
```

```
for i := 2; i < 51; i++ {
   fib[i] = fib[i-1] + fib[i-2]
}
```

汇编代码正确执行三行Go代码
但没有忠实地实现for循环（此例不要紧）

```
MOV 2, R2          // i:=2
Loop:
MOV 0, R1          // label Loop
ADD M[R0+R2*8-16], R1
ADD M[R0+R2*8-8], R1
MOV R1, M[R0+R2*8-0]
INC R2             // i++
CMP 51, R2         // i < 51?
JL Loop            // if Yes, goto Loop
```

基址索引偏移量寻址模式　天然适配循环

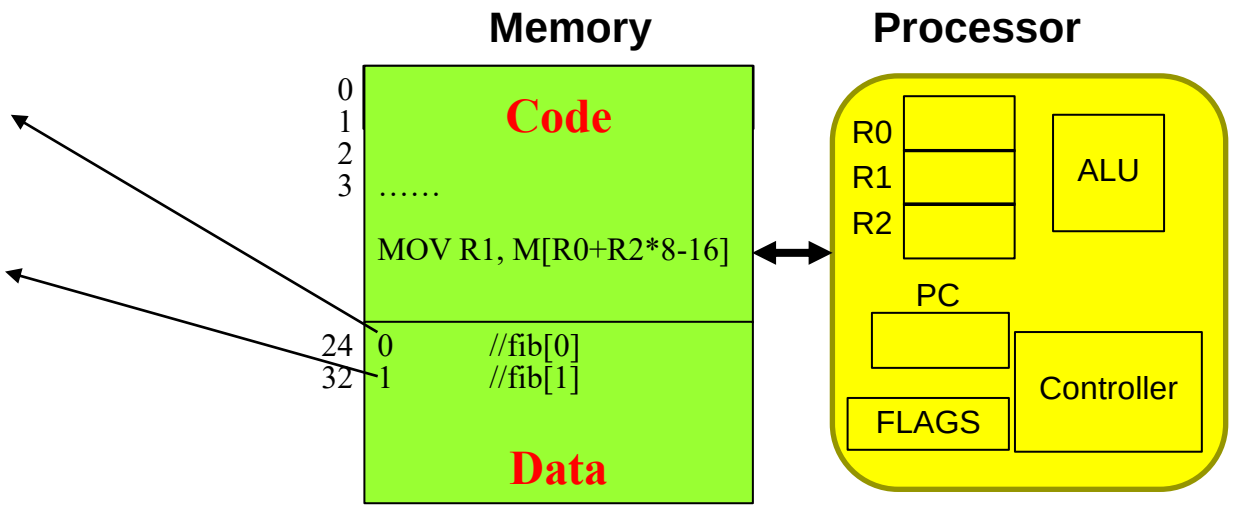
- **address = base + index*8 + offset**
  实际地址 = 基址 + 索引*比例因子 + 偏移量
- 进入for循环，i:=2
  - 基址寄存器R0=24
  - 索引寄存器R2=2
  - 比例因子=8，因为fib[i]是64位整数
  - 赋值语句fib[i] = fib[i-1] + fib[i-2]编译成

    MOV 0, R1
    ADD M[R0+R2*8-16], R1
    ADD M[R0+R2*8-8], R1
    MOV R1, M[R0+R2*8-0]
  - 第一条加法指令实现0+fib[0]

    R1+ M[R0+R2*8-16] → R1，即
    0 + M[24+2*8-16] → R1，即0+fib[0] → R1
  - 第二条加法指令实现0+fib[0]+fib[1]

    R1+ M[R0+R2*8-8] → R1，即
    0 + M[24+2*8-8] → R1，即0+fib[1] → R1

```
fib[0] = 0                      MOV 0, R1
                                MOV R1, M[R0] //R0=12 initially
fib[1] = 1                      MOV 1, R1
                                MOV R1, M[R0+8]
for i := 2; i < 51; i++ {       MOV 2, R2    // i:=2
   fib[i] = fib[i-1] + fib[i-2] MOV 0, R1    // label Loop
                                ADD M[R0+R2*8-16], R1
                                ADD M[R0+R2*8-8], R1
                                MOV R1, M[R0+R2*8-0]
                                INC R2       // i++
                                CMP 51, R2   // i < 51?
}                               JL Loop      // if Yes, goto Loop
```



2. 逻辑思维

- 关注计算过程的比特精准的正确性与通用性

- 课程讨论了
  - 布尔逻辑
  - 有限状态机、图灵机
  - 图灵可计算性、图灵机的通用性与局限
  - 哥德尔不完备定理

- 能够说明每个概念，并给出具体例子

2.1 布尔逻辑

- 比特精准的最基本体现
- 命题逻辑：命题、逻辑连接词、布尔表达式、布尔函数
  - 布尔函数 $f:\{0,1\}^n \to \{0,1\}$，例如$y = f(x_1, \cdots, x_n) = x_1 \oplus \cdots \oplus x_n$
  - 公理系统（从代数角度理解逻辑）
    - 元素：常数0,1、变量$x_1, \cdots, x_n$；布尔表达式
    - 算子：基本操作（与、或、非）
    - 命题公式（布尔逻辑的基本性质），可以看成是布尔代数的公理
    - 推理规则
  - 用真值表辅助
- 谓词逻辑
  - 多了断言和量词（**全称量词∀**和**存在量词∃**）
    - 断言是会传回“真”或“假”的函数，如
      - 任何自然数，要么它是偶数，要么它加1后为偶数
      - $\forall n \in \mathbb{N}\ [\mathrm{Even}(n) \vee \mathrm{Even}(n+1)]$

若干数学集合的标准记号

$\mathbb{N}$：自然数集合
$\mathbb{Z}$：整数集合
$\mathbb{Q}$：有理数集合
$\mathbb{R}$：实数集合
$\mathbb{C}$：复数集合

命题及其连接词，构成布尔表达式

- 基本命题：用命题符号表示
  - P: 今天寒冷。Q: 今天是阴天。R: 今天是寒冷的阴天。
- 复合命题：用命题符号与连接词表示
  - $P \wedge Q$: 今天是寒冷的阴天。
  - 五种连接词的定义
    - 非：$\neg x = 1$ *iff* $x = 0$。也用$\bar{x}$表示$\neg x$
      - $\neg P$：今天不冷
    - 与，合取：$x \wedge y = 1$ *iff* $x = y = 1$
      - $P \wedge Q$: 今天是寒冷的阴天
    - 或，析取：$x \vee y = 0$ *iff* $x = y = 0$
      - $P \vee Q$: 今天冷，或者今天是阴天
    - 异或：$x \oplus y = 1$ *iff* $x \neq y$　　两个命题有且仅有一个为真
      - $P \oplus Q$: 今天不冷且是阴天，或者今天冷且不是阴天
    - 蕴含：$(x \rightarrow y) = 1$ *iff* $x = 0$ or $y = 1$
      - $P \rightarrow Q$: 今天不冷，或者今天是阴天　　如果今天寒冷，则今天是阴天

以真值表为主线，理解命题逻辑的基本概念

- 真值表的基本定义
  - 习题：画出复合命题$P \wedge Q$的真值表
    - 真值表是布尔函数$P \wedge Q$的直接表达
      - $f:\{0,1\}^2 \to \{0,1\}, \quad Y = f(x_1, \cdots, x_n) = f(x_1, x_2) = f(P, Q) = P \wedge Q$
      - $f(P,Q) = \begin{cases} 0, & P=0, Q=0 \\ 0, & P=0, Q=1 \\ 0, & P=1, Q=0 \\ 1, & P=1, Q=1 \end{cases}$
      - 此处$n=2$，输入变量为P, Q，输出变量为Y

P	Q	$P \wedge Q$
0	0	0
0	1	0
1	0	0
1	1	1

以真值表为主线，理解命题逻辑的基本概念

- 真值表的基本定义
  - 习题：画出复合命题$P \wedge Q$的真值表
    - 真值表是布尔函数$P \wedge Q$的直接表达
      - $f:\{0,1\}^2 \to \{0,1\}, \quad Y = f(x_1, \cdots, x_n) = f(x_1, x_2) = f(P, Q) = P \wedge Q$
      - $f(P,Q) = \begin{cases} 0, & P = 0, Q = 0 \\ 0, & P = 0, Q = 1 \\ 0, & P = 1, Q = 0 \\ 1, & P = 1, Q = 1 \end{cases}$
      - 此处$n = 2$，输入变量为P, Q，输出变量为Y

先画表头

$f:\{0,1\}^2 \to \{0,1\}$，即$f:\{0,1\} \times \{0,1\} \to \{0,1\}$

$$Y = f(P, Q) = P \wedge Q$$

P	Q	$P \wedge Q$

以真值表为主线，理解命题逻辑的基本概念

- 真值表的基本定义
  - 习题：画出复合命题$P \wedge Q$的真值表
    - 真值表是布尔函数$P \wedge Q$的直接表达
      - $f:\{0,1\}^2 \to \{0,1\}, \quad Y = f(x_1, \cdots, x_n) = f(x_1, x_2) = f(P, Q) = P \wedge Q$
      - $$f(P,Q) = \begin{cases} 0, & P=0, Q=0 \\ 0, & P=0, Q=1 \\ 0, & P=1, Q=0 \\ 1, & P=1, Q=1 \end{cases}$$
      - 此处$n = 2$，输入变量为P, Q，输出变量为Y

先画表头

P	Q	$P \wedge Q$

再画表身

	P	Q	$P \wedge Q$
0	0	0	
1	0	1	
2	1	0	
2^n-1=3	1	1	

以真值表为主线，理解命题逻辑的基本概念

- 真值表的基本定义
  - 习题：画出复合命题$P \wedge Q$的真值表
    - 真值表是布尔函数$P \wedge Q$的直接表达
      - $f:\{0,1\}^2 \to \{0,1\}, \quad Y = f(x_1, \cdots, x_n) = f(x_1, x_2) = f(P, Q) = P \wedge Q$
      - $$f(P,Q) = \begin{cases} 0, & P=0, Q=0 \\ 0, & P=0, Q=1 \\ 0, & P=1, Q=0 \\ 1, & P=1, Q=1 \end{cases}$$
      - 此处$n=2$，输入变量为P, Q，输出变量为Y

先画表头

P	Q	$P \wedge Q$

再画表身

	P	Q	$P \wedge Q$
0	0	0	
1	0	1	
2	1	0	
2^n-1=3	1	1	

再填真值

P	Q	$P \wedge Q$
0	0	0
0	1	0
1	0	0
1	1	1

以真值表为主线，理解命题逻辑的基本概念

- 真值表的基本定义
  - 习题：画出复合命题$P \wedge Q$的真值表
    - 真值表是布尔函数$P \wedge Q$的直接表达
      - $f:\{0,1\}^2 \to \{0,1\}$，$Y = f(x_1, \cdots, x_n) = f(x_1, x_2) = f(P, Q) = P \wedge Q$
      - $$f(P,Q) = \begin{cases} 0, & P=0, Q=0 \\ 0, & P=0, Q=1 \\ 0, & P=1, Q=0 \\ 1, & P=1, Q=1 \end{cases}$$
  - 布尔函数的两种表示
    - 布尔表达式$P \wedge Q$
    - 真值表

P	Q	$P \wedge Q$
0	0	0
0	1	0
1	0	0
1	1	1

  - 举一反三：$P \vee Q$，$P \to Q$，$P \oplus Q$，$P \wedge Q \wedge P$，$P \wedge Q \wedge R$

以真值表为主线，理解命题逻辑的基本概念

- 真值表的基本定义
  - 习题：从应用题题面，画出复合命题$P \wedge Q$的真值表
    - P: 今天寒冷。
    - Q: 今天是阴天。
    - R: 今天是寒冷的阴天。
      - **注意：$\boldsymbol{R = P \wedge Q}$**

$\boldsymbol{P}$	$\boldsymbol{Q}$	$\boldsymbol{P \wedge Q}$
0	0	0
0	1	0
1	0	0
1	1	1

$\boldsymbol{P}$	$\boldsymbol{Q}$	$\boldsymbol{R}$
0	0	0
0	1	0
1	0	0
1	1	1

以真值表为主线，理解命题逻辑的基本概念

- 真值表的变奏之一：简化

P	Q	$P \wedge Q$
0	0	0
0	1	0
1	0	0
1	1	1

P	Q	$P \oplus Q$
0	0	0
0	1	1
1	0	1
1	1	0

P	Q	$P \rightarrow Q$
0	0	1
0	1	1
1	0	0
1	1	1

  - 将三个真值表画在一起，甚至忽略掉粗线

P	Q	$P \wedge Q$	$P \oplus Q$	$P \rightarrow Q$
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	0	1

以真值表为主线，理解命题逻辑的基本概念

- 真值表的变奏之一：简化

P	Q	$P \wedge Q$
0	0	0
0	1	0
1	0	0
1	1	1

P	Q	$P \oplus Q$
0	0	0
0	1	1
1	0	1
1	1	0

P	Q	$P \rightarrow Q$
0	0	1
0	1	1
1	0	0
1	1	1

- 引入输出变量：$X = P \wedge Q$，$Y = P \oplus Q$，$Z = P \rightarrow Q$

P	Q	$P \wedge Q$	$P \oplus Q$	$P \rightarrow Q$
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	0	1

P	Q	X	Y	Z
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	0	1

以真值表为主线，理解命题逻辑的基本概念

- 真值表的变奏之一：简化

P	Q	$P \wedge Q$
0	0	0
0	1	0
1	0	0
1	1	1

P	Q	$P \oplus Q$
0	0	0
0	1	1
1	0	1
1	1	0

P	Q	$P \rightarrow Q$
0	0	1
0	1	1
1	0	0
1	1	1

引入输出变量：$X = P \wedge Q$，$Y = P \oplus Q$，$Z = P \rightarrow Q$

P	Q	$P \wedge Q$	$P \oplus Q$	$P \rightarrow Q$
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	0	1

下表有几个输入变量?

第2个P是$f(P,Q) = P$

P	Q	**P**	X	Y	Z
0	0	0	0	0	1
0	1	0	0	1	1
1	0	1	0	1	0
1	1	1	1	0	1

以真值表为主线，理解命题逻辑的基本概念

- 真值表的变奏之二：整体理解
  - 纠正下列真值表的所有错误

P	Q	0	1	P	$\neg P$	Q	$\neg Q$	$P \wedge Q$	$P \vee Q$	$P \oplus Q$	$P \rightarrow Q$
0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1

以真值表为主线，理解命题逻辑的基本概念

- 真值表的变奏之二：整体理解
  - 观察下列四个表达式，进一步理解蕴含连接词$P \to Q$ (等价于$\neg P \vee Q$)

原命题：若（两个三角形）是全等三角形，则它们一定是相似三角形　　$\forall x, y\,[P(x,y) \to Q(x,y)]$

Congruent triangles are also similar

逆命题：若是相似三角形，则它们一定是全等三角形　　$\forall x, y\,[Q(x,y) \to P(x,y)]$

否命题：若不是全等三角形，则它们一定不是相似三角形　　$\forall x, y\,[\neg P(x,y) \to \neg Q(x,y)]$

逆否命题：若不是相似三角形，则它们一定不是全等三角形　　$\forall x, y\,[\neg Q(x,y) \to \neg P(x,y)]$

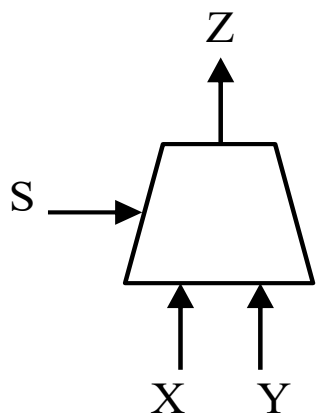
If two triangles are not similar, then they are not congruent triangles

P	Q	原命题 $P \to Q$	逆命题 $Q \to P$	否命题 $\neg P \to \neg Q$	逆否命题 $\neg Q \to \neg P$
0	0	1	1	1	1
0	1	1	0	0	1
1	0	0	1	1	0
1	1	1	1	1	1

以真值表为主线，理解命题逻辑的基本概念

- 真值表的变奏之三：分析与设计
  - 分析：布尔函数描述（题意，或布尔表达式）→ 真值表

- 双路复用器（Multiplexer，MUX）实现一个布尔函数 $f:\{0,1\}^3 \to \{0,1\}$，即$Z = f(S,X,Y)$。它根据控制输入S选择X或Y一路数据输入作为输出Z。
- 它包含4个单比特变量，分别是（1）两个数据输入变量X与Y；（2）一个控制输入变量S；（3）一个结果输出变量Z。当S=0时，Z=X（选择X）；当S=1时，Z=Y（选择Y）。

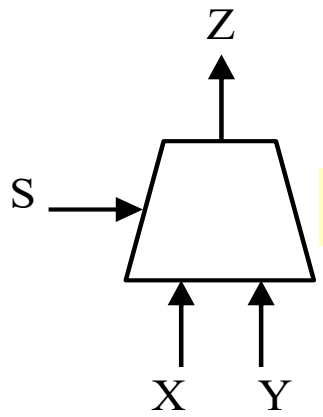


S	X	Y	Z
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

以真值表为主线，理解命题逻辑的基本概念

- 真值表的变奏之三：分析与设计
  - 设计：题意，或真值表 → 布尔表达式

- 双路复用器（Multiplexer，MUX）实现一个布尔函数 $f:\{0,1\}^3 \to \{0,1\}$，即$Z = f(S,X,Y)$。它根据控制输入S选择X或Y一路数据输入作为输出Z。
- 它包含4个单比特变量，分别是（1）两个数据输入变量X与Y；（2）一个控制输入变量S；（3）一个结果输出变量Z。当S=0时，Z=X（选择X）；当S=1时，Z=Y（选择Y）。



$$Z = (S \wedge X) \vee (\neg S \wedge Y)$$

S	X	Y	Z
0	**0**	0	**0**
0	**0**	1	**0**
0	**1**	0	**1**
0	**1**	1	**1**
1	0	**0**	**0**
1	0	**1**	**1**
1	1	**0**	**0**
1	1	**1**	**1**

任意布尔函数有范式（唯一性）

- 合取范式(conjunctive normal form, CNF)
  - $f(x_1,\ldots,x_n)= Q_1 \wedge Q_2 \wedge Q_3 \ldots \wedge Q_m$
  - 其中: $Q_i = l_1 \vee l_2 \vee \ldots \vee l_n$，　$l_j = x_j$ 或 $\neg x_j$
- 析取范式(disjunctive normal form, DNF)
  - $f(x_1,\ldots,x_n)= Q_1 \vee Q_2 \vee Q_3 \ldots \vee Q_m$
  - 其中: $Q_i = l_1 \wedge l_2 \wedge \ldots \wedge l_n$，　$l_j = x_j$ 或 $\neg x_j$
- 写出2个变量的全部函数的析取范式（一共有$2^{2^n}=2^4= 16$个）
  - 【注意：另一套等价的与、或、非记号；计算系统思维常用】
  - $y = 0$ 【注】　　$y = \overline{x_1}\cdot\overline{x_2} + \overline{x_1}\cdot x_2 + x_1\cdot\overline{x_2} + x_1\cdot x_2 = 1$
  - $y = x_1\cdot\overline{x_2} + x_1\cdot x_2 = x_1$　　$y = \overline{x_1}\cdot x_2 + x_1\cdot\overline{x_2} + x_1\cdot x_2 = x_1 + x_2$ ……

x_1	x_2	y
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**0**

x_1	x_2	y
0	0	**1**
0	1	**1**
1	0	**1**
1	1	**1**

x_1	x_2	y
0	0	**0**
0	1	**0**
1	0	**1**
1	1	**1**

x_1	x_2	y
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**1**

以真值表为主线，理解命题逻辑的基本概念

- 真值表的变奏之三：分析与设计
  - 设计：从题意或真值表，得出析取范式
  - 从析取范式获得简化的布尔表达式

S	X	Y	Z
0	0	0	0
0	0	1	0
0	1	0	**1**
0	1	1	**1**
1	0	0	0
1	0	1	**1**
1	1	0	0
1	1	1	**1**

$Z =$ $(\neg S \wedge X \wedge \neg Y)$ $\vee$ $(\neg S \wedge X \wedge Y)$ $\vee$ $(S \wedge \neg X \wedge Y)$ $\vee$ $(S \wedge X \wedge Y)$

$(\neg S \wedge X \wedge \neg Y) \vee (\neg S \wedge X \wedge Y) = (\neg S \wedge X)$

$(S \wedge \neg X \wedge Y) \vee (S \wedge X \wedge Y) = (S \wedge Y)$

$Z = (\neg S \wedge X) \vee (S \wedge Y)$

重新审视布尔函数

- n-输入-1输出的布尔函数是数学函数 $f:\{0,1\}^n \to \{0,1\}$
  - 如，$f(x_1, x_2, \cdots, x_n) = (\vee_{i=1}^{n-1} x_i) \oplus x_n$
- 两个布尔函数（布尔表达式）等价：如果它们有相同的真值表
  - 类似：$f(x,y) = x^2 - y^2$和$g(x,y) = (x+y)(x-y)$是相同的多项式
- 一个变量x的布尔函数 $y = f(x)$
  - 一共四个

布尔函数有两种表示
- 布尔表达式
- 真值表

x	y
0	**0**
1	**0**

$y = 0$

x	y
0	**1**
1	**1**

$y = 1$

x	y
0	**0**
1	**1**

$y = x$

x	y
0	**1**
1	**0**

$y = \overline{x}$

布尔代数是一个五元组 (L, +, •, ¬, 0, 1)　　如何构造出n个变量的L?

L是布尔表达式集合；$x_1, \ldots, x_n$ 是布尔变量；0, 1是布尔常数；或+、与•、非¬是三个算子

- 初始假定：$0,1, x_1, \ldots, x_n \in$ L；递归构造直至达到闭包：If $x, y \in$ L, then $\neg x, x \bullet y, x + y \in$ L.
- 用下列布尔代数性质（公理）合并等价表达式　　（相同元素在集合中只能出现一次）

		布尔代数	中学代数，设x, y, z = 2, 3, 5	
1.	Associativity:	$(x \bullet y) \bullet z = x \bullet (y \bullet z)$, $(x + y) + z = x + (y + z)$	(2•3)•5 = 2•(3•5) (2+3)+5 = 2+(3+5)	结合律
2.	Commutativity:	$x \bullet y = y \bullet x$, $x + y = y + x$	2•3 = 3•2 2+3 = 3+2	交换律
3.	Distributivity:	$(x + y) \bullet z = (x \bullet z) + (y \bullet z)$, $(x \bullet y) + z = (x + z) \bullet (y + z)$	(2+3)•5 = 2•5+3•5 (2•3)+5 ≠ (2+5)•(3+5)	分配率
4.	Identity:	$x + 0 = x$, $x \bullet 1 = x$	2+0=2, 2•1=2	幺元律
5.	Annihilator:	$x \bullet 0 = 0$, $x + 1 = 1$	2•0=0, 2+1≠1	极元律
6.	Idempotence:	$x \bullet x = x$, $x + x = x$	2•2 ≠ 2, 2+2 ≠ 2	幂等律
7.	Absorption:	$(x \bullet y) + x = x$, $(x + y) \bullet x = x$	(2•3)+2≠2, (2+3)•2≠2	吸收律
8.	**Complementation:**	$x + \neg x = 1$, $x \bullet \neg x = 0$		排中律
9.	**Double negation:**	$\bar{\bar{x}} = x$，即$\neg(\neg x) = x$		双重否定律
10.	**De Morgan law:**	$\overline{x \wedge y} = \bar{x} \vee \bar{y}$，$\overline{x \vee y} = \bar{x} \wedge \bar{y}$		德摩根定律
11.	**Implication:**	$x \rightarrow y = \bar{x} \vee y$		蕴含律
12.	**Exclusive-Or:**	$x \oplus y = (\bar{x} \wedge y) + (x \wedge \bar{y})$		异或律

2个变量的布尔表达式

- 第一轮。常数和变量及其非，共6个表达式：

$$L = \{ 0, 1, x_1, x_2, \overline{x_1}, \overline{x_2} \}$$

- 第二轮。应用与或非到第一轮结果，使用公理整理，共8个新表达式：

$$\overline{x_1} + \overline{x_2}, \quad \overline{x_1} + x_2, \quad x_1 + \overline{x_2}, \quad x_1 + x_2$$
$$\overline{x_1} \cdot \overline{x_2}, \quad \overline{x_1} \cdot x_2, \quad x_1 \cdot \overline{x_2}, \quad x_1 \cdot x_2$$

- 第三轮。应用与或非到第二轮表达式，使用公理整理，共2个新表达式：

$$\overline{x_1} \cdot \overline{x_2} + x_1 \cdot x_2, \quad \overline{x_1} \cdot x_2 + x_1 \cdot \overline{x_2}$$

- 第四轮。应用与或非到第三轮表达式，使用公理整理，共0个新表达式。达到闭包。

$$L = \{ \quad 0, 1, x_1, x_2, \overline{x_1}, \overline{x_2},$$
$$\overline{x_1} + \overline{x_2}, \overline{x_1} + x_2, x_1 + \overline{x_2}, x_1 + x_2, \ \overline{x_1} \cdot \overline{x_2}, \ \overline{x_1} \cdot x_2, x_1 \cdot \overline{x_2}, x_1 \cdot x_2,$$
$$\overline{x_1} \cdot \overline{x_2} + x_1 \cdot x_2, \overline{x_1} \cdot x_2 + x_1 \cdot \overline{x_2} \quad \}$$

Kleene Algebra（替换布尔代数的排中律）

- 从克莱因代数 (L, +, •, ¬, 0, 1)获得克莱因表达式
  - L is the set of all Kleene expressions.
    - 0,1, $x_1, \ldots, x_n \in$ L;　　初始假定
    - If $x, y \in$ L, then $\neg x, x \bullet y, x + y \in$ L.　　递归直至达到闭包
      - 且满足如下公理（用下列公理消除等价表达式）

1.	Associativity:	$(x \bullet y) \bullet z = x \bullet (y \bullet z)$, $(x + y) + z = x + (y + z)$	**结合律**
2.	Commutativity:	$x \bullet y = y \bullet x$, $x + y = y + x$	**交换律**
3.	Distributivity:	$(x + y) \bullet z = (x \bullet z) + (y \bullet z)$, $(x \bullet y) + z = (x + z) \bullet (y + z)$	**分配率**
4.	Identity: Annihilator:	$x + 0 = x$, $x \bullet 1 = x$ $x \bullet 0 = 0$, $x + 1 = 1$	**有界律**
5.	Idempotence:	$x \bullet x = x$, $x + x = x$	**幂等律**
6.	Absorption:	$(x \bullet y) + x = x$, $(x + y) \bullet x = x$	**吸收律**
7.	de Morgan:	$\neg(x+y) = \neg x \bullet \neg y$, $\neg(x \bullet y) = \neg x + \neg y$	**摩根律**
8.	Double Negation:	$\neg\neg x = x$	**对合律**
9.	Product:	$p \bullet x_i + p \bullet \neg x_i = p + p \bullet x_i + p \bullet \neg x_i$ where p is a product containing $\neg x_j \bullet x_i$	在布尔代数中， $p \bullet x_i + p \bullet \neg x_i = p$

克莱因表达式 ≠ Kleene函数

- An *n*-variable Kleene function (≠ Kleene expression)
  - $f:\{0,\mathrm{I},1\}^n \to \{0,\mathrm{I},1\}$
- 两个Kleene表达式等价：
  - 能从Kleene公理集推出它们相等
- 一个输入变量*x*的Kleene表达式
  - 一共六个

x	$\overline{x}$	$x \bullet \overline{x}$	$x + \overline{x}$
0	1	0	1
I	I	I	I
1	0	0	1

- 第一轮。常数和变量及其非，共4个表达式：$0, 1, x, \overline{x}$　　(布尔逻辑中，已经达到了闭包)
- 第二轮。应用与或非到第一轮结果，使用公理整理，共2个新表达式：(布尔逻辑中，这两个表达式不是新的)
  $x \cdot \overline{x}$,　　$x + \overline{x}$
- 第三轮。应用与或非到第二轮表达式，使用公理整理，共0个新表达式。达到闭包。

x	y
0	**0**
I	**0**
1	**0**

$y = 0$

x	y
0	**1**
I	**1**
1	**1**

$y = 1$

x	y
0	**0**
I	**I**
1	**1**

$y = x$

x	y
0	**1**
I	**I**
1	**0**

$y = \overline{x}$

x	y
0	**0**
I	**I**
1	**0**

$y = x \cdot \overline{x}$

x	y
0	**1**
I	**I**
1	**1**

$y = x + \overline{x}$

x	y
0	**I**
I	**I**
1	**I**

$y = \mathrm{I}$

克莱因表达式 ≠ Kleene函数

- An *n*-variable Kleene function (≠ Kleene expression)
  - $f:\{0,\mathrm{I},1\}^n \to \{0,\mathrm{I},1\}$
- 两个Kleene表达式相同：
  - 有相同的真值表
  - 能从Kleene公理集推出
- 只有一个变量的Kleene表达式
  - 一共六个

x	$\overline{x}$	$x \bullet \overline{x}$	$x + \overline{x}$
0	1	0	1
I	I	I	I
1	0	0	1

存在无对应表达式的真值表

x	y
0	**0**
I	**0**
1	**0**

$y = 0$

x	y
0	**1**
I	**1**
1	**1**

$y = 1$

x	y
0	**0**
I	**I**
1	**1**

$y = x$

x	y
0	**1**
I	**I**
1	**0**

$y = \overline{x}$

x	y
0	**0**
I	**I**
1	**0**

$y = x \cdot \overline{x}$

x	y
0	**1**
I	**I**
1	**1**

$y = x + \overline{x}$

I	
1	**I**

$y = \mathrm{I}$

计算机科学幸运地选择了布尔逻辑

- n个变量的布尔函数（布尔表达式）有多少个？
  - 例如：¬(¬x)与x是两个布尔表达式，但相互等价
    （①真值表相同）或者（②通过公理导出相等）
  - n=0：2个（0和1）；n=1：4个（1，0，x，¬x）
  - n=n：2^{2^n}
- n个变量的克莱因表达式有多少个？
  - 弱化排中律
  - 不是3^{3^n}；尚未找到闭合公式，但知道 $< 2^{3^n}$

任一布尔函数都有唯一的布尔表达式等价类
布尔表达式等价类=布尔函数=真值表

不存在真值表
无对应表达式

存在真值表
无对应表达式

n	布尔函数个数 2^{2^n}	布尔表达式个数 2^{2^n}	克莱因表达式个数	克莱因函数个数 3^{3^n}
1	$2^{2^1}=4$	$2^{2^1}=4$	6	$3^{3^1}=27$
2	$2^{2^2}=16$	$2^{2^2}=16$	84	$3^{3^2}=3^9$
3	$2^{2^3}=256$	$2^{2^3}=256$	43918	$3^{3^3}=3^{27}$
4	$2^{2^4}=65536$	$2^{2^4}=65536$	160297985276	$3^{3^4}=3^{81}$

2.2 存在很多逻辑论证（arguments）方法

科学需要心胸开阔，切忌鼠目寸光、把现有的科学知识当成教条

下述陈述句集合是不是一个逻辑论证 logical argument?

- 屠呦呦1930年出生于宁波。
- 她的长辈从《诗经》的“呦呦鹿鸣食野之蒿”给她取名“呦呦”。
- “呦呦鹿鸣食野之蒿”之中的“蒿”就是青蒿。
- 屠呦呦长大后研究出了青蒿素，一种治疗疟疾的药物。
- 《诗经》预言成真，冥冥中自有定数。

下述陈述句集合是不是一个逻辑论证 logical argument?

- 屠呦呦1930年出生于宁波。
- 她的长辈从《诗经》的“呦呦鹿鸣食野之蒿”给她取名“呦呦”。
- “呦呦鹿鸣食野之蒿”之中的“蒿”就是青蒿。
- 屠呦呦长大后研究出了青蒿素，一种治疗疟疾的药物。
- 网上有人言：《诗经》预言成真，冥冥中自有定数。这纯属胡说八道。

下述陈述句集合是不是一个逻辑论证 logical argument?

- 屠呦呦1930年出生于宁波。
- 她的长辈从《诗经》的“呦呦鹿鸣食野之蒿”给她取名“呦呦”。
- “呦呦鹿鸣食野之蒿”之中的“蒿”就是青蒿。
- 屠呦呦长大后研究出了青蒿素，一种治疗疟疾的药物。
- ~~网上有人言：《诗经》预言成真，冥冥中自有定数。这纯属胡说八道。~~

- 屠呦呦研究青蒿素，参考了葛洪的《肘后备急方》一书内容。
- 葛洪是炼丹的道士，信奉神仙，其代表作《抱朴子》是讲修仙的书。
- 屠呦呦研究青蒿素的工作，引用了讲迷信的“葛神仙”的著作。
- 屠呦呦研究青蒿素的工作，不科学。

《肘后备急方》

卷三

治寒热诸疟方第十六

治疟病方。鼠妇，豆豉二七枚，合捣令相和。未发时服二丸，欲发时服一丸。

又方：**青蒿一握。以水二升渍，绞取汁。尽服之。**

又方：用独父蒜于白炭上烧之，末。服方寸匕。

又方：五月五日，蒜一片（去皮，中破之，刀割），合容巴豆一枚（去心皮，纳蒜中，令合）。以竹挟以火炙之，取可热，捣为三丸。未发前服一丸。不止，复与一丸。

又方：取蜘蛛一枚，芦管中密塞，管中以绾颈，过发时乃解去也。

又方：日始出时，东向日再拜，毕正长跪，向日叉手，当闭气，以书墨注其管两耳中，各七注，又丹书舌上，言子日死，毕，复再拜，还去勿顾，安卧勿食，过发时断，即瘥。

……

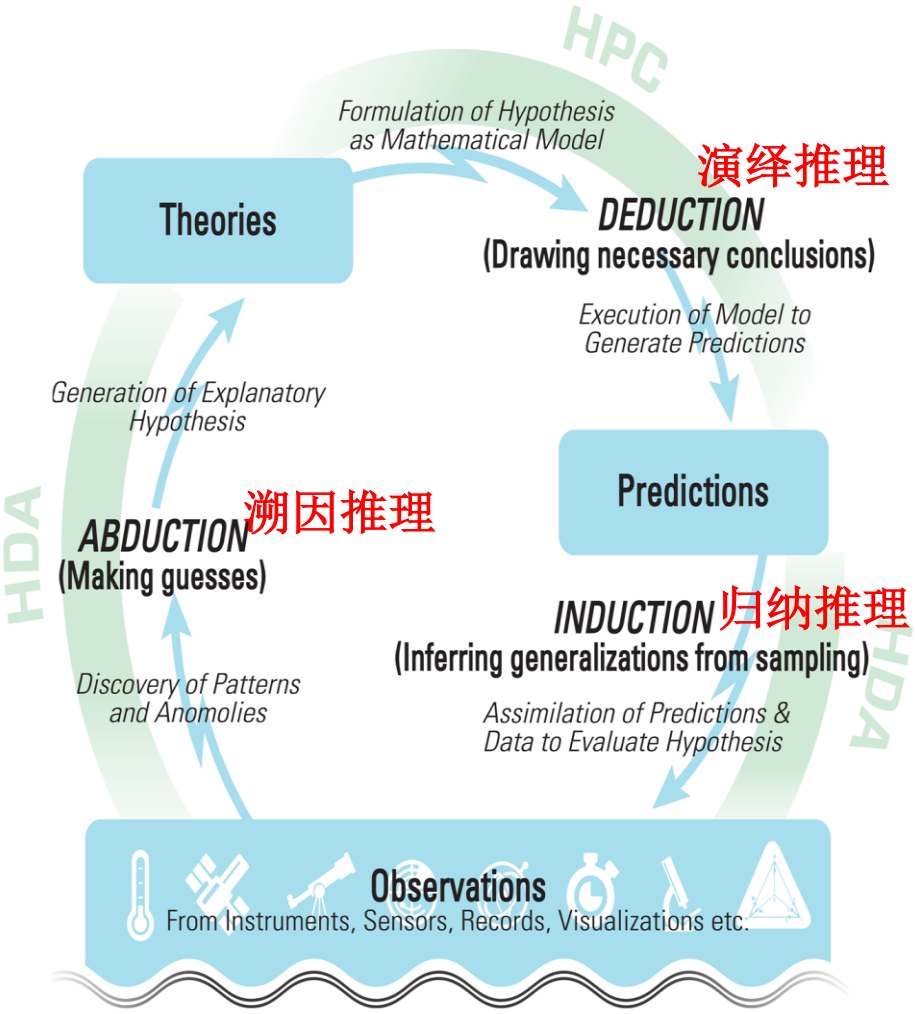
下述陈述句集合是不是一个逻辑论证 logical argument?

- 屠呦呦研究出了青蒿素，一种治疗疟疾的药物。
- 屠呦呦研究青蒿素，参考了葛洪的《肘后备急方》一书内容。
- 葛洪是炼丹的道士，信奉神仙，其代表作《抱朴子》是讲修仙的书。
- 屠呦呦研究青蒿素的工作，引用了讲迷信的“葛神仙”的著作。
- 屠呦呦研究青蒿素的工作，不科学。
- 疟疾患者不应该使用青蒿素。

- The Nobel Prize in Physiology or Medicine 2015 was awarded to William C. Campbell, Satoshi Ōmura and Youyou Tu.
  - 2015年度**诺贝尔生理学或医学奖**被授予三位研究者。奖金的一半授予**威廉·坎贝尔**和**大村智**，以表彰他们在创新蛔虫疗法方面的贡献；另一半授予**屠呦呦**，以表彰她在治疗疟疾方面的贡献。……屠呦呦制成了青蒿素，这种药物显著降低了疟疾患者的死亡率。

存在很多逻辑论证方法，科学中常常融合三种推理（Reasoning）

- Deductive reasoning 演绎推理
  - 从前提（premises，也称为论据）依据推理规则（inference rules）推导出结论（conclusion）
  - 好的演绎推理逻辑论证称为有效的（valid）
  - 不能保证结果是正确的
    - 如果前提都是正确的，结论肯定也是正确的
    - 如果前提有错误，结论可能也是错误的
- Inductive reasoning 归纳推理
  - 从现有样本的结果，推导出泛化结论
- Abductive reasoning 溯因推理
  - 根据理论和观察猜测出最好的假说
- 本课程聚焦可自动执行的演绎推理



ECBD Pathway to Convergence, 2017

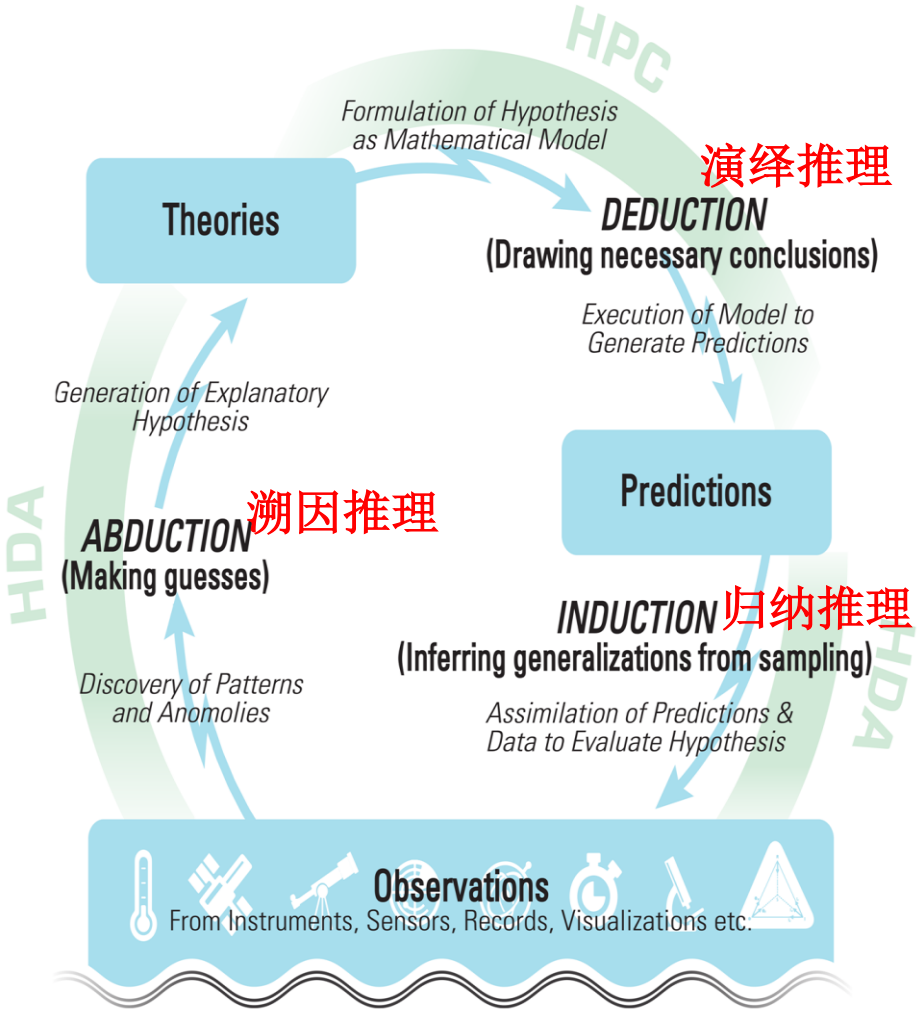
归纳推理（Inductive Reasoning，Induction）

- 从现有样本的结果，推导出泛化结论
  Inductive Generalization

- 观察：人类有史以来，太阳每天早上从东方升起
- 假说：太阳总会每天早上从东方升起
- 预测：太阳明天早上会从东方升起

- 观察：人们看到的天鹅都是白色的
- 假说：天鹅总是白色的
- 预测：明天我去圆明园看到的天鹅也会是白色的

- 数学归纳法属于演绎推理！



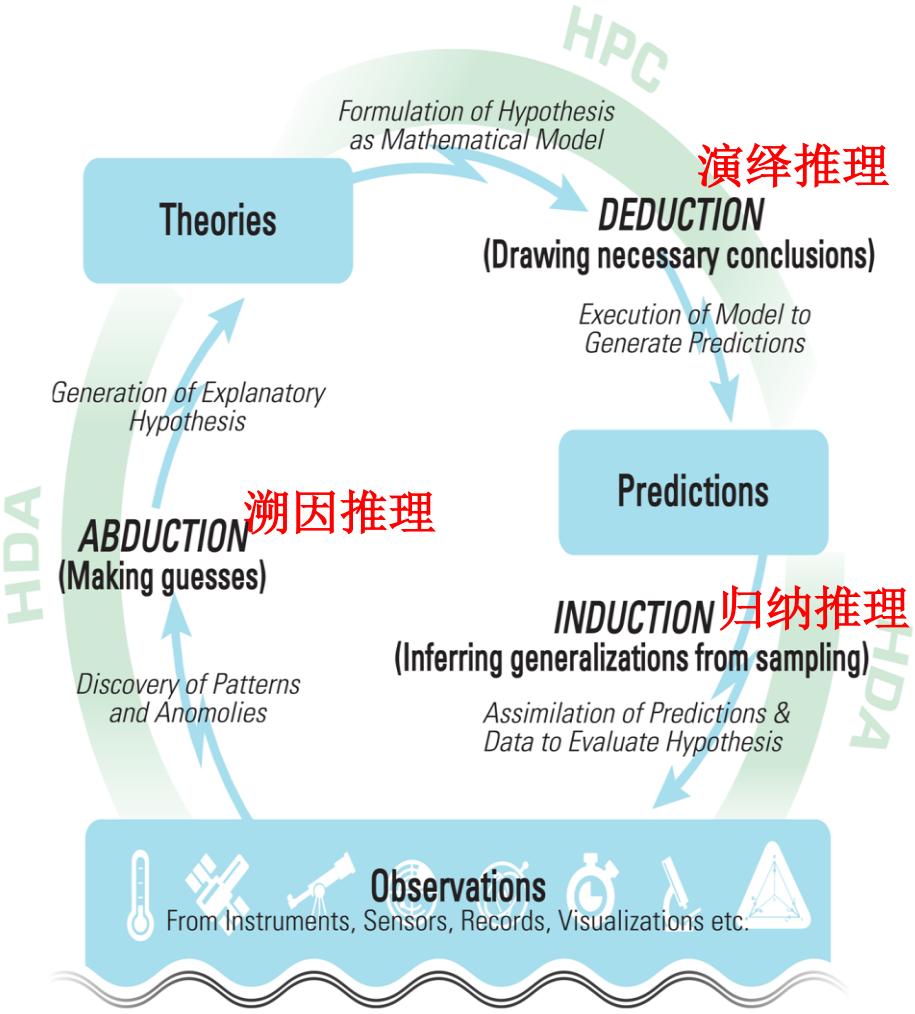
ECBD Pathway to Convergence, 2017

溯因推理（Abductive Reasoning，Abduction）

- 根据理论和观察猜测出最好的假说

- 观察太阳系七大行星，发现天王星公转轨道违反了牛顿万有引力定律
- 假说1：牛顿万有引力定律是错的
- 假说2：太阳系存在第八颗行星

- 随后，科学家根据假说2提供的位置，发现了海王星



ECBD Pathway to Convergence, 2017

课堂小测验

姓名：　　学号：

P∨Q是一个布尔表达式，包含两个布尔变量P、Q。
¬((¬P)∧(¬Q))是另一个布尔表达式，等价于P∨Q。
它们同属于一个布尔表达式等价类。

包含N个布尔变量的布尔表达式等价类一共有多少个？（）

A. $1 \times (2^{2^N})$个

B. $(2 \times 2)^{2^N}$个

C. $2^{(2\times2)^N}$个

D. $2^{2^{(2\times N)}}$个

归纳推理的逻辑论证不保真（not truth preserving）

- 保真（truth preserving），有时也称为真值传递：
  - 所有前提为真，则结论一定为真。
  - 不可能出现“所有前提为真结论为假”的情况。

- 归纳推理例子（从2015年以前的诺贝尔科学奖样本泛化）
    - 诺贝尔科学奖包括诺贝尔物理奖、化学奖、生理学或医学奖，不包括文学奖、和平奖、经济学奖
- 前提
  - 1901年伦琴、范托夫、贝林因科学研究成果获奖
  - …
  - 2014年中村修二、天野浩、赤崎勇、莫尔纳尔、赫尔、贝奇格、莫泽、奥基夫、莫泽因科学研究成果获奖
- 结论：诺贝尔科学奖得主因科学研究成果获奖

演绎推理的逻辑论证保真（truth preserving）！

- 保真（truth preserving）：
  - 所有前提为真，则结论一定为真。
  - 不可能出现“所有前提为真结论为假”的情况。

- P：诺贝尔科学奖得主因科学研究成果获奖
- Q：屠呦呦是诺贝尔科学奖得主
- R：屠呦呦的获奖成果是科学研究成果

- 不可能出现“R为假但是(P∧Q)为真”的情况
  - R为假：屠呦呦的获奖成果不是科学研究成果；同时P为真且Q为真

- 演绎推理保真，只是说论证是有效的（valid）
  - 还要加上前提为真，才能保证结论为真。此时（valid +前提为真），说论证是可靠的（sound）。

如何做到保真呢？

就本课程而言

- 在演绎论证中，只使用布尔逻辑的推理规则连接前提与结论
- 推理规则的一般形式如下

P1　　前提1（布尔表达式）
P2　　前提2（布尔表达式）
…
Pk　　前提k（布尔表达式）

Q　　结论（布尔表达式）

也就是说，

布尔表达式[(P1∧… ∧Pk)→Q]是永真式（tautology）

例如，[(P ∧ P→Q) → Q]恒等于1

P	Q	$P \to Q$	$(P \wedge (P \to Q)) \to Q$
0	0	1	1
0	1	1	1
1	0	0	1
1	1	1	1

演绎推理（本课程的布尔逻辑）的推理规则

替换（Substitution），即利用布尔逻辑的性质，替换相等项。这是最基础的规则。
例如，$P \rightarrow Q = (\neg P) \vee Q$。因此，在推理过程中，
$P \rightarrow Q$（如果我在学编程，那么我用计算机）可替换为
$\neg P \vee Q$（我不在学编程 或者 我用计算机）

肯定前件式（Modus Ponens）

P	我在学编程
$P \rightarrow Q$	我学编程要用计算机
Q	我用计算机

否定后件式（Modus Tollens）

$\neg Q$	我不用计算机
$P \rightarrow Q$	我学编程要用计算机
$\neg P$	我没在学编程

析取三段论（Disjunctive Syllogism）

$P \vee Q$	我在学编程或者我用计算机
$\neg P$	我没在学编程
Q	我用计算机

假言三段式（Hypothetical Syllogism）

$P \rightarrow Q$	我学编程要用计算机
$Q \rightarrow R$	我用计算机要交电费
$P \rightarrow R$	我学编程要交电费

有些推理规则涉及量词

请看英文教科书第104页

2.3 建模需要严谨性

- 给定下列两个论据，能推出结论是成立的吗？
  - 论据1：所有上过计科导课的同学都精通Go语言。
  - 论据2：某些精通Go语言的同学可成为下一年计科导课助教。
  - 结论：有些上过计科导的同学可成为下一年计科导课助教。

建模需要严谨性

- 给定下列两个论据，能推出结论是成立的吗？
  - 论据1：所有上过计科导课的同学都精通Go语言。
  - 论据2：某些精通Go语言的同学可成为下一年计科导课助教。
  - 结论：有些上过计科导的同学可成为下一年计科导课助教。
- 定义下列断言记号
  - $\mathrm{CS101}(x)$：同学x上过计科导
  - $\mathrm{MasterGo}(x)$：同学x精通Go语言
  - $\mathrm{TA}(x)$：同学x可成为下一年计科导课助教

建模需要严谨性

- 给定下列两个论据，能推出结论是成立的吗？
  - 论据1：所有上过计科导课的同学都精通Go语言。
  - 论据2：某些精通Go语言的同学可成为下一年计科导课助教。
  - 结论：有些上过计科导的同学可成为下一年计科导课助教。
- 定义下列断言记号
  - $\mathrm{CS101}(x)$：同学x上过计科导
  - $\mathrm{MasterGo}(x)$：同学x精通Go语言
  - $\mathrm{TA}(x)$：同学x可成为下一年计科导课助教

错误的建模

$\forall x[\mathrm{MasterGo}(x)]$

$\exists x[\mathrm{MasterGo}(x) \rightarrow \mathrm{TA}(x)]$

$\exists x[\mathrm{CS101}(x) \rightarrow \mathrm{TA}(x)]$

论据2的谓词表达式应该是$\exists x[\mathrm{MasterGo}(x) \wedge \mathrm{TA}(x)]$

$\mathrm{MasterGo}(x)$	$\mathrm{TA}(x)$	$\mathrm{MasterGo}(x) \rightarrow \mathrm{TA}(x)$	$\mathrm{MasterGo}(x) \wedge \mathrm{TA}(x)$
0	0	1	0
0	1	1	0
1	0	0	0
1	1	1	1

建模需要严谨性

- 给定下列两个论据，能推出结论是成立的吗？
  - 论据1：所有上过计科导课的同学都精通Go语言。
  - 论据2：某些精通Go语言的同学可成为下一年计科导课助教。
  - 结论：有些上过计科导的同学可成为下一年计科导课助教。

- 严谨地构建谓词表达式、使用推理规则
  - 谓词逻辑往往不关心原初论据本身的正确性。
    即，假设给定论据是正确的。
  - 定义下列断言记号
    - $\mathrm{CS101}(x)$：同学x上过计科导
    - $\mathrm{MasterGo}(x)$：同学x精通Go语言
    - $\mathrm{TA}(x)$：同学x可成为下一年计科导课助教

错误的建模

$\forall x[\mathrm{MasterGo}(x)]$

$\exists x[\mathrm{MasterGo}(x) \rightarrow \mathrm{TA}(x)]$

$\exists x[\mathrm{CS101}(x) \rightarrow \mathrm{TA}(x)]$

论据1：$\forall x[\mathrm{CS101}(x) \rightarrow \mathrm{MasterGo}(x)]$
论据2：$\exists x[\mathrm{MasterGo}(x) \wedge \mathrm{TA}(x)]$
结论：　$\exists x[\mathrm{CS101}(x) \wedge \mathrm{TA}(x)]$

感觉结论不对，试找反例
结论的非：　$\neg \exists x[\mathrm{CS101}(x) \wedge \mathrm{TA}(x)]$
等价于　　$\forall x[\neg \mathrm{CS101}(x) \vee \neg \mathrm{TA}(x)]$
只要找到一个同学集及其映射，
满足论据，且使得每个成员x
　　$\neg \mathrm{CS101}(x)$
或　　$\neg \mathrm{TA}(x)$

建模需要严谨性

- 给定下列两个论据，能推出结论是成立的吗？
  - 论据1：所有上过计科导课的同学都精通Go语言。
  - 论据2：某些精通Go语言的同学可成为下一年计科导课助教。
  - 结论：有些上过计科导的同学可成为下一年计科导课助教。

- 严谨地构建谓词表达式、使用推理规则
  - 谓词逻辑往往不关心原初论据本身的正确性。
    即，假设给定论据是正确的。
  - 定义下列断言记号
    - $\mathrm{CS101}(x)$：同学x上过计科导
    - $\mathrm{MasterGo}(x)$：同学x精通Go语言
    - $\mathrm{TA}(x)$：同学x可成为下一年计科导课助教

- 结论不对，找简单反例
- 设某年级只有两名同学a, b，其映射如下
  - 同学a：$\mathrm{CS101}(a) = \mathrm{MasterGo}(a) = True, \mathrm{TA}(a) = False$
  - 同学b：$\mathrm{CS101}(b) = False$，$\mathrm{MasterGo}(b) = \mathrm{TA}(b) = True$
  - 容易验证论据1和2都成立，但结论不成立

错误的建模

$\forall x[\mathrm{MasterGo}(x)]$

$\exists x[\mathrm{MasterGo}(x) \rightarrow \mathrm{TA}(x)]$

$\exists x[\mathrm{CS101}(x) \rightarrow \mathrm{TA}(x)]$

论据1：$\forall x[\mathrm{CS101}(x) \rightarrow \mathrm{MasterGo}(x)]$
论据2：$\exists x[\mathrm{MasterGo}(x) \wedge \mathrm{TA}(x)]$
结论：$\exists x[\mathrm{CS101}(x) \wedge \mathrm{TA}(x)]$

感觉结论不对，试找反例
结论的非：$\neg \exists x[\mathrm{CS101}(x) \wedge \mathrm{TA}(x)]$
等价于 $\forall x[\neg \mathrm{CS101}(x) \vee \neg \mathrm{TA}(x)]$
只要找到一个同学集及其映射，
满足论据，且使得每个成员x
$\neg \mathrm{CS101}(x)$
或 $\neg \mathrm{TA}(x)$

$\neg \mathrm{TA}(a)$

$\neg \mathrm{CS101}(b)$

3. 图灵机的通用性和局限性

- 图灵机能够求解任意可计算问题
- 邱奇-图灵论题（Church-Turing Hypothesis）
  - 任意抽象计算机与图灵机等价
    - （足够大存储器的）冯诺依曼计算机与图灵机等价
- 存在不可计算问题
  - 例如：停机问题

图灵机的通用性和局限性

- **邱奇-图灵论题（Church-Turing Hypothesis）**
  - 任意抽象计算机与图灵机等价
    - （足够大存储器的）冯诺依曼计算机与图灵机等价

- 此处的“等价”是什么含义？
  - 可计算性意义上等价
  - 实用意义上不等价
  - 冯诺依曼机是桥接模型，图灵机不是
    - 图灵机加法非常繁琐，但Go程序只需要一行：X = 1024 + 512

哥德尔不完备定理的一个实例

- 哥德尔不完备定理：在任何一个包含初等算术的数学系统中，存在真但不可证的命题。
  - 初等算术=皮亚诺算术
  - **哪个真命题在哪个皮亚诺算术系统中不可证？**
- Goodstein Theorem, 1944
  - Every Goodstein sequence approaches zero
- 戈德斯坦定理不能在皮亚诺算术系统中证明
  - Kirby & Paris, 1982
  - Cichon,1983
  - Caicedo, 2007
  - Zankl et al., 2014

皮亚诺算术五公理
(引自大不列颠百科全书)

1. Zero is a natural number.
   零是自然数
2. Every natural number has a successor in the natural numbers.
   任意自然数有一后继自然数
3. Zero is not the successor of any natural number.
   零不是任何自然数的后继
4. If the successor of two natural numbers is the same, then the two original numbers are the same.
   后继相同的两个自然数相等
5. If a set contains zero and the successor of every number is in the set, then the set contains the natural numbers. 归纳公理

Zankl, Harald & Winkler, Sarah & Middeldorp, Aart. (2014). Beyond polynomials and Peano arithmetic—automation of elementary and ordinal interpretations. Journal of Symbolic Computation. Volume 69, 2015, Pages 129-158, https://doi.org/10.1016/j.jsc.2014.09.033.

Goodstein's Theorem

- For any natural number *n*, the **Goodstein sequence** is $(n)_0, (n)_1, \ldots, (n)_{G(n)}, 0, \ldots$
- E.g., for n=266, the Goodstein sequence is

$$
\begin{aligned}
(266)_0 &= 2^{2^{2+1}} + 2^{2+1} + 2 &&= 266 \\
(266)_1 &= 3^{3^{3+1}} + 3^{3+1} + 3 - 1 &&\approx 4.4 \times 10^{38} \\
(266)_2 &= 4^{4^{4+1}} + 4^{4+1} + 2 - 1 &&\approx 3.2 \times 10^{616} \\
(266)_3 &= 5^{5^{5+1}} + 5^{5+1} + 1 - 1 &&\approx 2.5 \times 10^{10921} \\
(266)_4 &= 6^{6^{6+1}} + 6^{6+1} - 1 &&\approx 3.5 \times 10^{217832}
\end{aligned}
$$

……

整个宇宙基本粒子数 < 10^{90}

- 看起来增长很快。但是
  - **Goodstein's Theorem**: For any natural number *n*, its Goodstein sequence $(n)_0, (n)_1, \ldots, (n)_{G(n)}, \ldots$ approaches 0.

- **G(n)** 称为Goodstein function. The first *k* such that $(n)_k = 0$.

Goodstein sequence for n=4

- 2为底：$(4)_0 = 2^2$ $= 4$
- 3为底：$(4)_1 = 3^3 - 1 = 2 \cdot 3^2 + 2 \cdot 3 + 2$ $= 26$
- 4为底：$(4)_2 = 2 \cdot 4^2 + 2 \cdot 4 + 2 - 1$ $= 41$
- 5为底：$(4)_3 = 2 \cdot 5^2 + 2 \cdot 5 + 1 - 1$ $= 60$
- 6为底：$(4)_4 = 2 \cdot 6^2 + 2 \cdot 6 - 1$ $= 83$
- 7为底：$(4)_5 = 2 \cdot 7^2 + 7 + 5 - 1$ $= 109$
- 8为底：$(4)_6 = 2 \cdot 8^2 + 8 + 4 - 1$ $= 139$

……

- $(4)_{G(4)} = 0$
  - $G(4) = 3 \cdot 2^{402653211} - 2 \approx 6.895 \times 10^{121210694}$

整个宇宙基本粒子数 $< 10^{90}$

- 能够枚举所有$(4)_k$吗？

英文教科书勘误（p. 121）

- We discuss a specific statement to make our understanding of Gödel's first incompleteness theorem more concrete. The statement is Goodstein theorem, which is true, but cannot be proven in **any** mathematical system that includes elementary number theory.

"any" 应为 "a"

- We discuss a specific statement to make our understanding of Gödel's first incompleteness theorem more concrete. The statement is Goodstein theorem, which is true, but cannot be proven in **a** mathematical system that includes elementary number theory.

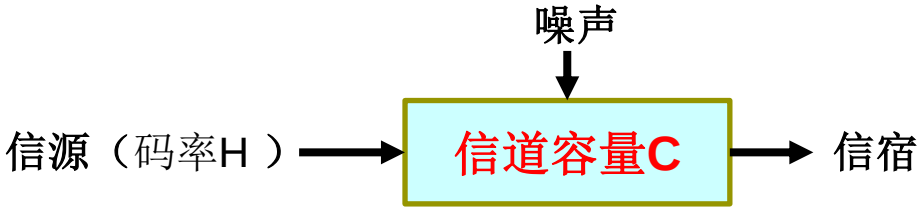
4. 什么是构造性?

Acu-Exams-CP

- Effectiveness = Constructiveness + Finiteness

（广义）构造性 = （狭义）构造性 + 有穷性

- 信息论中的香农第二定理是存在性定理，不是构造性定理



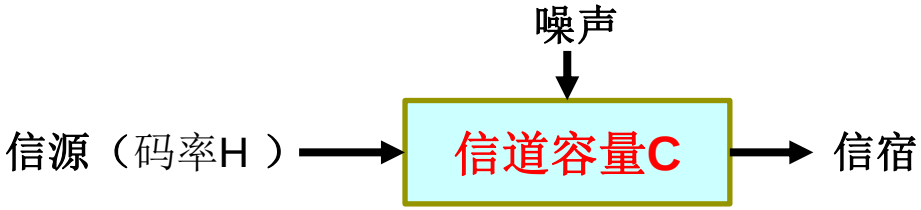
假如H<=C，存在编码，
实现无错消息传递

什么是狭义构造性?

- Effectiveness = Constructiveness + Finiteness

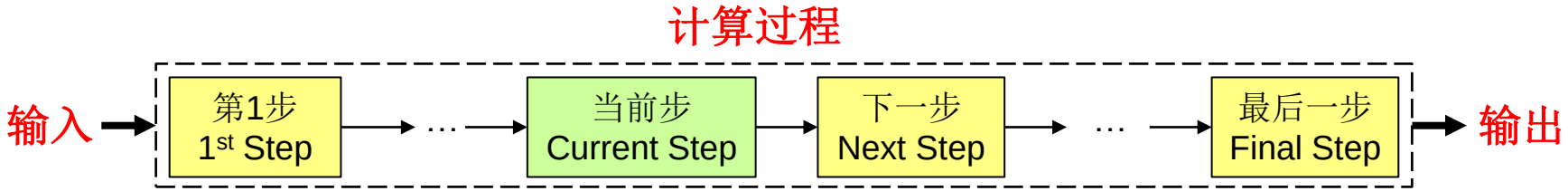
（广义）构造性 = （狭义）构造性 + 有穷性

- 信息论中的香农第二定理是存在性定理，不是构造性定理



假如H<=C，存在编码，
实现无错消息传递

- 计算过程是构造性过程，能够一步一步算出（构造出）结果



计算系统

什么是有穷性?

- Effectiveness = Constructiveness + Finiteness

（广义）构造性 = （狭义）构造性 + 有穷性

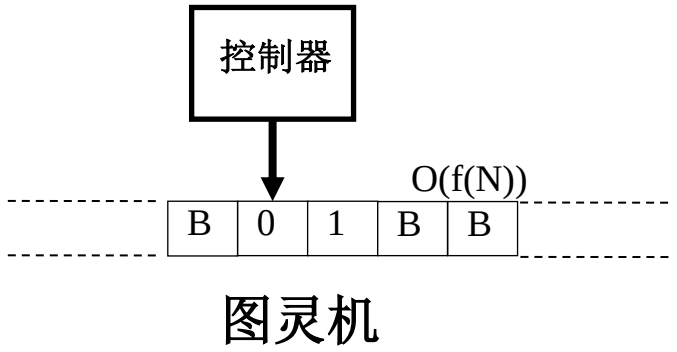
- 两类有穷性
  - 需要的资源量与问题规模N有关，是O(f(N))；
  - 需要的资源量与问题规模N无关，是O(1)
  - 其中，问题规模N可以任意大，但不是无穷大

什么是有穷性?

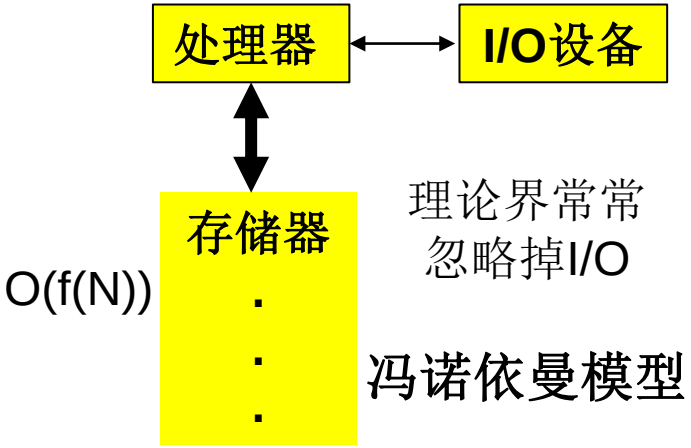
- Effectiveness = Constructiveness + Finiteness

（广义）构造性 = （狭义）构造性 + 有穷性

- 两类有穷性：与问题规模N有关O(f(N))；与问题规模N无关O(1)
  - 存储有穷性：与问题规模有关，空间复杂度 O(f(N))，N任意大
    - 笔记本电脑，与图灵机**不等价**，因为有限存储容量 O(1)
    - （足够大存储容量的）笔记本电脑，与图灵机等价 O(f(N))



图灵机



理论界常常忽略掉I/O

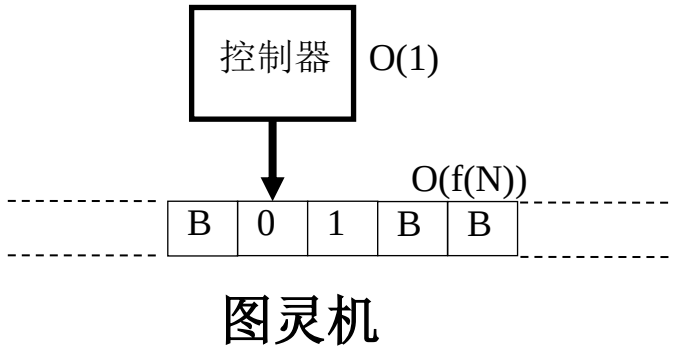
冯诺依曼模型

什么是有穷性?

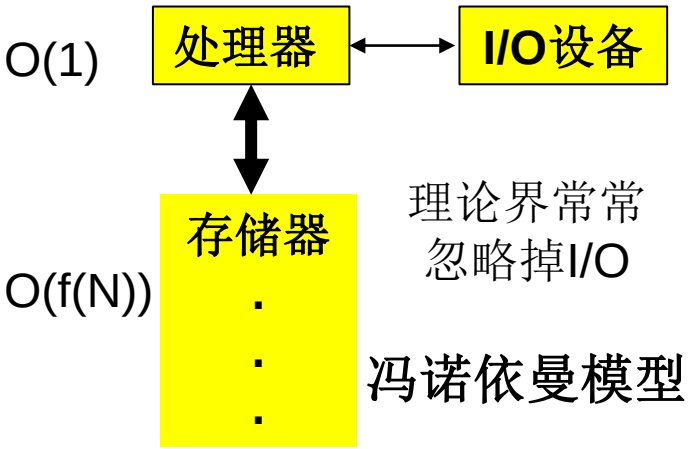
- Effectiveness = Constructiveness + Finiteness

（广义）构造性 = （狭义）构造性 + 有穷性

- 两类有穷性：与问题规模N有关O(f(N))；与问题规模N无关O(1)
  - 存储有穷性：与问题规模有关，空间复杂度　　O(f(N))，N任意大
    - 笔记本电脑，与图灵机**不等价**，因为有限存储容量　　O(1)
    - （足够大存储容量的）笔记本电脑，与图灵机等价　　O(f(N))
  - 程序有穷性：程序行数与问题规模无关　　O(1)
    - 用有限行程序表达计算过程
      - Ada运算流程序→控制流程序
    - 任何图灵机的状态转移表只有有限行
      - 加法图灵机：几十行



图灵机



理论界常常忽略掉I/O

冯诺依曼模型

5. 算法思维妙在何处？

- 算法概念的巧妙定义（高德纳五点定义）
  - 简洁、通用、可检查
  - 加法图灵机设计是算法设计，状态转移表满足高德纳定义

算法思维妙在何处？

- 算法概念的巧妙定义（高德纳五点定义）
  - 简洁、通用、可检查
  - 加法图灵机设计是算法设计
- 巧妙的效率度量（复杂度的o，O，Ω，Θ记号表达）
  - 忽略不重要细节，有利于算法分析
  - 快速排序：最坏复杂度$O(n^2)$，平均复杂度$O(n\log n)$

算法思维妙在何处？

- 算法概念的巧妙定义（高德纳五点定义）
  - 简洁、通用、可检查
  - 加法图灵机设计是算法设计
- 巧妙的效率度量（复杂度的o，O，Ω，Θ记号表达）
  - 忽略不重要细节，有利于算法分析
  - 快速排序：最坏复杂度$O(n^2)$，平均复杂度$O(n\log n)$
- 巧妙的算法设计方法（algorithmic paradigms）
  - 分治、动态规划、贪心、随机

算法思维妙在何处？

- 算法概念的巧妙定义（高德纳五点定义）
  - 简洁、通用、可检查
    - 加法图灵机设计是算法设计
- 巧妙的效率度量（复杂度的o，O，Ω，Θ记号表达）
  - 忽略不重要细节，简化算法分析
    - 快速排序：最坏复杂度$O(n^2)$，平均复杂度$O(n\log n)$
- 巧妙的算法设计方法（algorithmic paradigms）
  - 分治、动态规划、贪心、随机
- 巧妙的变奏，适应问题特色
  - 单因素优选法中的0.618法，而不是二分查找法
    - 复用查询结果
    - 二分法的复杂度是$\log_{\sqrt{2}} n$，$> \log_{1.618} n$

5.1 掌握高德纳的算法定义

一个算法是一组有限条规则，给出求解特定类型问题的操作序列，并具备下列五个特征：

① 有限性：算法在有限步骤之后必然要终止。~n^2

② 确定性：算法的每个步骤都必须精确地（严格地和无歧义地）定义。

③ 输入：算法有零个或多个输入，在算法开始或中途动态地给定。

④ 输出：算法有一个或多个输出，输出是与输入有特定关系的数值。

⑤ 能行性：算法的所有操作必须是充分基本的，原则上一个人能够用笔和纸在有限时间内精确地完成它们。

简洁

通用

功能可检查

性能可分析

冒泡排序算法

- 输入：待排序的数组A，数组A的长度n。
- 输出：排好序的数组A。
- 算法步骤描述：

```
for i = 1 to n − 1
  for j = 1 to n − i
    if A[j]>A[j+1]
      exchange A[j] with A[j+1].
```

掌握高德纳的算法定义

- 什么是/不是算法
- 一个算法是一组有限条规则，给出求解特定类型问题的操作序列，并具备下列五个特征：
  ① 有限性：算法在有限步骤之后必然要终止。
  ② 确定性：算法的每个步骤都必须精确地（严格地和无歧义地）定义。
  ③ 输入：算法有零个或多个输入，在算法开始或中途动态地给定。
  ④ 输出：算法有一个或多个输出，输出是与输入有特定关系的数值。
  ⑤ 能行性：算法的所有操作必须是充分基本的，原则上一个人能够用笔和纸在有限时间内精确地完成它们。

冒泡排序算法

- 输入：待排序的数组A，数组A的长度n。
- 输出：排好序的数组A。
- 算法步骤描述：

for $i = 1$ to $n - 1$
  for $j = 1$ to $n - i$
    if $A[j]>A[j+1]$
      exchange $A[j]$ with $A[j+1]$.

什么不是算法？

重新审视高德纳的算法定义

- 注意三种有穷性

一个算法是一组**有限**条规则，给出求解特定类型问题的操作序列，并具备下列五个特征：

① 有限性：算法在**有限**步骤之后必然要终止。

② 确定性：算法的每个步骤都必须精确地（严格地和无歧义地）定义。

③ 输入：算法有零个或多个输入，在算法开始或中途动态地给定。

④ 输出：算法有一个或多个输出，输出是与输入有特定关系的数值。

⑤ 能行性：算法的所有操作必须是充分基本的，原则上一个人能够用笔和纸在**有限**时间内精确地完成它们。

冒泡排序算法

- 输入：待排序的数组A，数组A的长度n。
- 输出：排好序的数组A。
- 算法步骤描述：

for $i = 1$ to $n - 1$
  for $j = 1$ to $n - i$
    if $A[j]>A[j+1]$
      exchange $A[j]$ with $A[j+1]$.

什么不是算法？

重新审视高德纳的算法定义

- 注意三种有穷性

一个算法是一组**有限**条规则，给出求解特定类型问题的操作序列，并具备下列五个特征：

① 有限性：算法在**有限**步骤之后必然要终止。

② 确定性：算法的每个步骤都必须精确地（严格地和无歧义地）定义。

③ 输入：算法有零个或多个输入，在算法开始或中途动态地给定。

④ 输出：算法有一个或多个输出，输出是与输入有特定关系的数值。

⑤ 能行性：算法的所有操作必须是充分基本的，原则上一个人能够用笔和纸在**有限**时间内精确地完成它们。

冒泡排序算法

- 输入：待排序的数组A，数组A的长度n。
- 输出：排好序的数组A。
- 算法步骤描述：

for $i = 1$ to $n - 1$
  for $j = 1$ to $n - i$
    if $A[j]>A[j+1]$
      exchange $A[j]$ with $A[j+1]$.

什么不是算法？

- Ada的运算流程序
  - 程序行数应与问题规模无关

重新审视高德纳的算法定义

- 注意三种有穷性

一个算法是一组**有限**条规则，给出求解特定类型问题的操作序列，并具备下列五个特征：

① 有限性：算法在**有限**步骤之后必然要终止。

② 确定性：算法的每个步骤都必须精确地（严格地和无歧义地）定义。

③ 输入：算法有零个或多个输入，在算法开始或中途动态地给定。

④ 输出：算法有一个或多个输出，输出是与输入有特定关系的数值。

⑤ 能行性：算法的所有操作必须是充分基本的，原则上一个人能够用笔和纸在**有限**时间内精确地完成它们。

冒泡排序算法

- 输入：待排序的数组A，数组A的长度n。
- 输出：排好序的数组A。
- 算法步骤描述：

for $i = 1$ to $n - 1$
  for $j = 1$ to $n - i$
    if $A[j]>A[j+1]$
      exchange $A[j]$ with $A[j+1]$.

什么不是算法？

- Ada的运算流程序
  - 程序行数应与问题规模无关
- 无穷循环不停机

重新审视高德纳的算法定义

- 注意三种有穷性

一个算法是一组有限条规则，给出求解特定类型问题的操作序列，并具备下列五个特征：

① 有限性：算法在有限步骤之后必然要终止。

② 确定性：算法的每个步骤都必须精确地（严格地和无歧义地）定义。

③ 输入：算法有零个或多个输入，在算法开始或中途动态地给定。

④ 输出：算法有一个或多个输出，输出是与输入有特定关系的数值。

⑤ 能行性：算法的所有操作必须是充分基本的，原则上一个人能够用笔和纸在有限时间内精确地完成它们。

冒泡排序算法

- 输入：待排序的数组A，数组A的长度n。
- 输出：排好序的数组A。
- 算法步骤描述：

for $i = 1$ to $n - 1$
  for $j = 1$ to $n - i$
    if $A[j]>A[j+1]$
      exchange $A[j]$ with $A[j+1]$.

什么不是算法？

- Ada的运算流程序
  - 程序行数应与问题规模无关
- 无穷循环不停机
- 包含下列操作
  - if哥德巴赫猜想为真{…}
    有限时间完不成！

重新审视高德纳的算法定义

- 注意三种有穷性

一个算法是一组**有限**条规则，给出求解特定类型问题的操作序列，并具备下列五个特征：

① 有限性：算法在**有限**步骤之后必然要终止。

② 确定性：算法的每个步骤都必须精确地（严格地和无歧义地）定义。

③ 输入：算法有零个或多个输入，在算法开始或中途动态地给定。

④ 输出：算法有一个或多个输出，输出是与输入有特定关系的数值。

⑤ 能行性：算法的所有操作必须是充分基本的，原则上一个人能够用笔和纸在**有限**时间内精确地完成它们。

冒泡排序算法

- 输入：待排序的数组A，数组A的长度n。
- 输出：排好序的数组A。
- 算法步骤描述：

```
for i = 1 to n − 1
  for j = 1 to n − i
    if A[j]>A[j+1]
      exchange A[j] with A[j+1].
```

规则有穷性

伪代码行数是O(1)，与n无关

步骤有穷性

即算法的时间复杂度

操作能行性

任一操作是充分基本的，所需时间肯定是有限的

重新审视高德纳的算法定义

- 确定性是什么意思？

一个算法是一组有限条规则，给出求解特定类型问题的操作序列，并具备下列五个特征：

① 有限性：算法在有限步骤之后必然要终止。

② 确定性：算法的每个步骤都必须精确地（严格地和无歧义地）定义。

比特精准！

① 输入：算法有零个或多个输入，在算法开始或中途动态地给定。

② 输出：算法有一个或多个输出，输出是与输入有特定关系的数值。

③ 能行性：算法的所有操作必须是充分基本的，原则上一个人能够用笔和纸在有限时间内精确地完成它们。

冒泡排序算法

- 输入：待排序的数组A，数组A的长度n。
- 输出：排好序的数组A。
- 算法步骤描述：

for $i = 1$ to $n - 1$
  for $j = 1$ to $n - i$
    if $A[j]>A[j+1]$
      exchange $A[j]$ with $A[j+1]$.

什么不是算法？

- 包含非比特精准的操作步骤

重新审视高德纳的算法定义

- 确定性是什么意思？

一个算法是一组有限条规则，给出求解特定类型问题的操作序列，并具备下列五个特征：

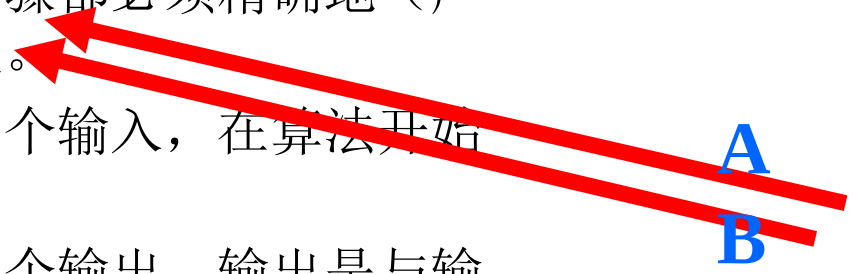
① 有限性：算法在有限步骤之后必然要终止。

② 确定性：算法的每个步骤都必须精确地（严格地和无歧义地）定义。

③ 输入：算法有零个或多个输入，在算法开始或中途动态地给定。

④ 输出：算法有一个或多个输出，输出是与输入有特定关系的数值。

⑤ 能行性：算法的所有操作必须是充分基本的，原则上一个人能够用笔和纸在有限时间内精确地完成它们。



冒泡排序算法

- 输入：待排序的数组A，数组A的长度n。
- 输出：排好序的数组A。
- 算法步骤描述：

for $i = 1$ to $n - 1$
  for $j = 1$ to $n - i$
    if $A[j]>A[j+1]$
      exchange $A[j]$ with $A[j+1]$.

什么不是算法？

- 包含下列操作
- 比较两个物理线段的长度
  If 线段A长度==线段B长度 {…}

5.2 算法的复杂度分析

- 求解斐波那契数列的递归算法GO代码耗时多少？

```
func fibonacci(n int) int {
        if n == 0 || n == 1 {
                return n
        }
        return fibonacci(n-1) + fibonacci(n-2)
}
```

- 复杂度分析（常用求解递推式的分析方法）
  - T(n) = T(n-1) + T(n-2) = T(n-2)+T(n-3)+T(n-3)+T(n-4)
  - 2*T(n-2) < T(n) < 2*T(n-1)
  - T(n) < 2*T(n-1) < 4*T(n-2) < 8*T(n-3) … < 2^n
  - T(n) > 2*T(n-2) > 4*T(n-4) > 8*T(n-6) … > $2^{n/2}$
  - $2^{n/2}$ < T(n) < 2^n
  - **T(n) = $\Omega(2^{n/2})$，T(n) = O(2^n)， 指数复杂度！**

指数复杂度增长非常迅速！

两国战后谈判，A国要求B国赔偿粮食。在10x10棋盘上放满稻米：

第1格放1粒米，第2格放2粒米，第3格放4粒米，…，第k格放2^k粒米。

B国该不该答应？

10x10棋盘

粒	1	2	4	8	16	32			

指数复杂度增长非常迅速！

在10x10棋盘上放满稻米：

第1格放1粒米，第2格放2粒米，第3格放4粒米，…，第k格放2^k粒米。

每粒米质量=0.02克（50粒米=1克）

放满前10个格子，棋盘上稻米才20来克

棋盘上放满稻米总共有$2^{n+1}-1$粒米。

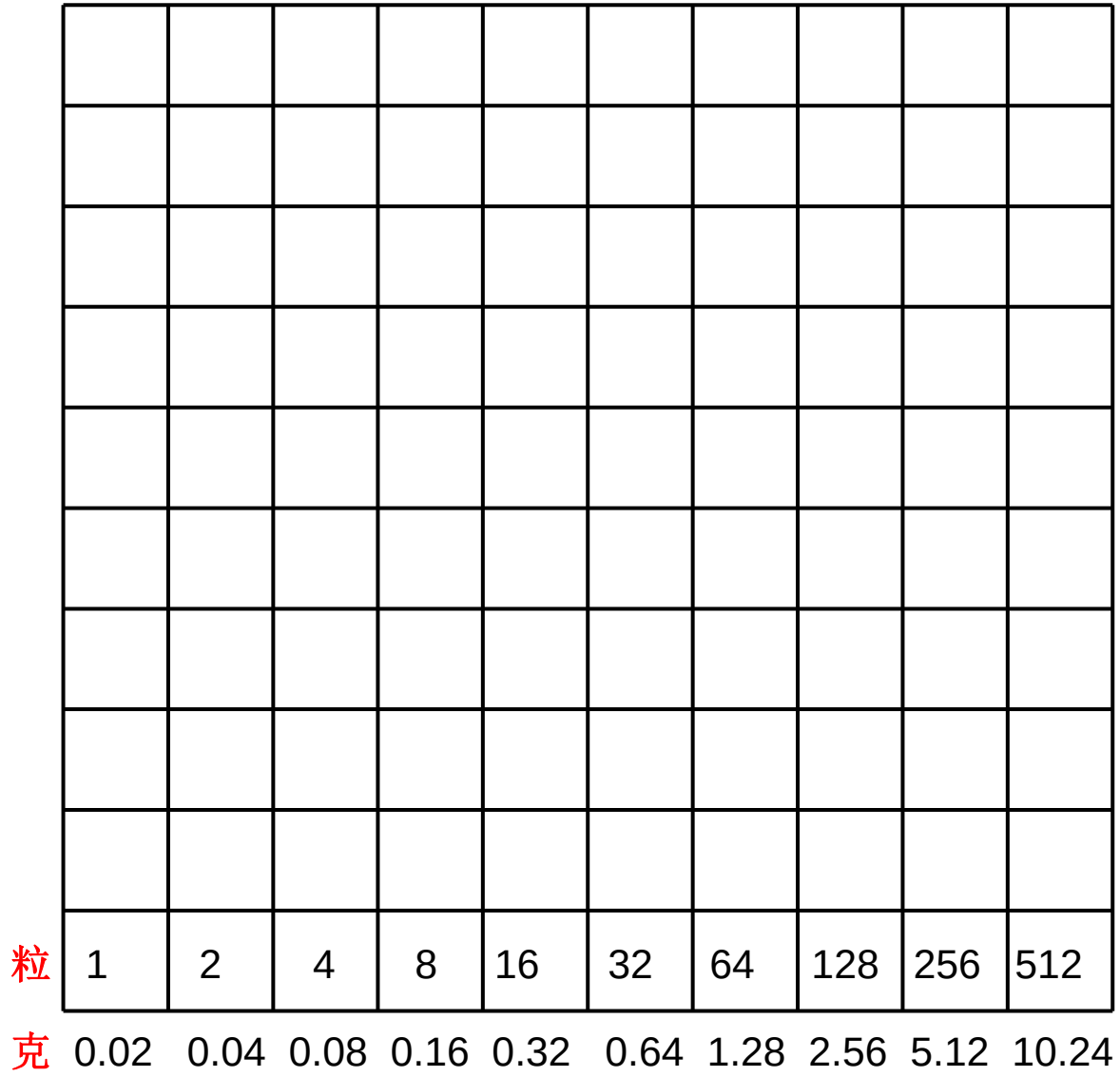
放满100个格子，总共有$2^{101}-1$粒米

地球的质量大约是$6x2^{24}$千克。

B国要赔偿的稻米重量相当于

8个多地球！

10x10棋盘

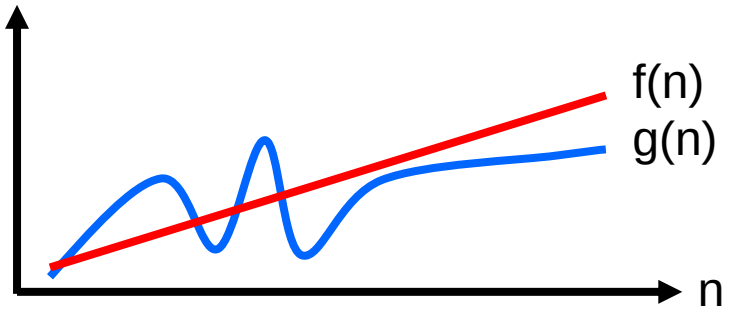


小o，大O，Ω，Θ记号

等号是单向的，不同于数学等号

$n^{1.58} = \mathbf{O}(n^2)$，$n^{1.58} = \mathbf{O}(n^3)$，但$\mathbf{O}(n^2) \neq \mathbf{O}(n^3)$

假设：当n足够大



- $f(n) = \mathrm{o}(g(n))$
  - $n^{1.58} = \mathbf{o}(n^2)$，$n^{1.58} \neq \mathbf{o}(n^{1.58})$，$n^2 \neq \mathbf{o}(n^{1.58})$

$$\lim_{n\to\infty} f(n)/G(n) = 0$$

- $f(n) = \mathrm{O}(g(n))$
  - $n^{1.58} = \mathbf{O}(n^2)$，$n^{1.58} = \mathbf{O}(n^{1.58})$，$n^2 \neq \mathbf{O}(n^{1.58})$

$\exists$常数$c > 0, f(n) \leq cg(n)$

- $f(n) = \Omega(g(n))$
  - $n^{1.58} \neq \mathbf{\Omega}(n^2)$，$n^{1.58} = \mathbf{\Omega}(n^{1.58})$，$n^2 = \mathbf{\Omega}(n^{1.58})$

$\exists$常数$c > 0, f(n) \geq cg(n)$

- $f(n) = \Theta(g(n))$
  - $n^{1.58} \neq \mathbf{\Theta}(n^2)$，$n^{1.58} = \mathbf{\Theta}(n^{1.58})$，$n^2 \neq \mathbf{\Theta}(n^{1.58})$

$f(n) = \mathrm{O}(g(n))$ 并且 $f(n) = \Omega(g(n))$

小o，大O，Ω，Θ记号

共同假设：当n足够大。g是f的严格上阶o 、上阶O 、下阶Ω 、等阶Θ

$$\lim_{n\to\infty}\frac{f(n)}{g(n)}=0$$

- $f(n)=o(g(n))$ g是f的严格上界（阶）
  - $n^{1.58}=\mathbf{o}(n^2)$， $n^{1.58}$ ？ $\mathbf{o}(n^{1.58})$ ， $n^2\neq\mathbf{o}(n^{1.58})$
- $f(n)=O(g(n))$ 上界

  $\exists$常数$c>0, f(n)\leq cg(n)$
  - $n^{1.58}$ ？ $\mathbf{O}(n^2)$， $n^{1.58}=\mathbf{O}(n^{1.58})$， $n^2\neq\mathbf{O}(n^{1.58})$
- $f(n)=\Omega(g(n))$ 下界

  $\exists$常数$c>0, f(n)\geq cg(n)$
  - $n^{1.58}$ ？ $\mathbf{\Omega}(n^2)$， $n^{1.58}=\mathbf{\Omega}(n^{1.58})$， $n^2=\mathbf{\Omega}(n^{1.58})$
- $f(n)=\Theta(g(n))$ 等阶

  $f(n)=O(g(n))$ 并且 $f(n)=\Omega(g(n))$
  - $n^{1.58}$ ？ $\mathbf{\Theta}(n^2)$， $n^{1.58}=\mathbf{\Theta}(n^{1.58})$， $n^2\neq\mathbf{\Theta}(n^{1.58})$

5.4 算法思维方法（Algorithmic Paradigms）

- 分治思想
  - 各种排序算法
  - 单因素优选法，好于二分法
  - 大整数乘法
  - 矩阵乘法

算法思维方法（Algorithmic Paradigms）

- 分治思想
  - 各种排序算法
  - 单因素优选法，好于二分法
  - 大整数乘法
  - 矩阵乘法
  - 复杂度分析的一般递推公式

将规模为n的待解问题
分为a个子问题，
每个子问题规模为n/b；
合并子问题结果的开销为$O(n^d)$。

时间复杂度的递推公式为

$$T(n)=\begin{cases} aT\left(\frac{n}{b}\right)+O(n^d) & n>1 \\ O(1) & n=1 \end{cases}$$

解递推公式，时间复杂度为

$$T(n)=\begin{cases} O(n^d) & d>\log_b a \\ O(n^d \log n) & d=\log_b a \\ O(n^{\log_b a}) & d<\log_b a \end{cases}$$

归并排序：$a=b=2$,
$d=\log_b a=1$,
$T(n)=O(n\log n)$

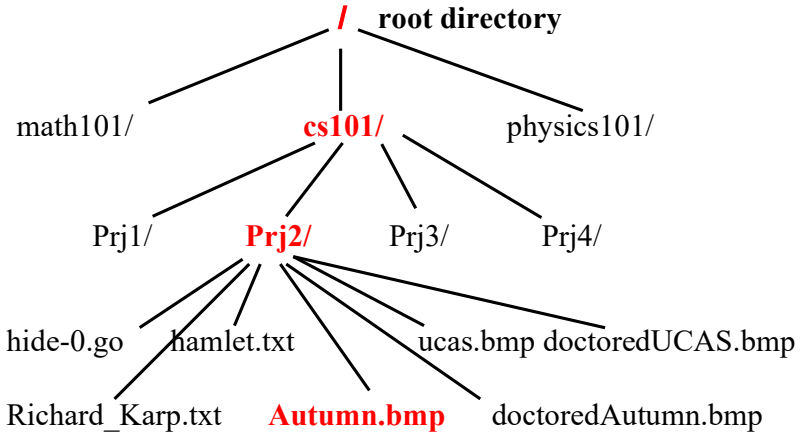
算法思维方法

- 分治思想
  - 各种排序算法
  - 单因素优选法，好于二分法
  - 大整数乘法
  - 矩阵乘法
  - 分析复杂度的一般递推公式
- 其他方法：
  - 动态规划、贪心、随机
  - **快速排序**
    - **fastsort.go vs. quicksort.go**

```
package main  // fastsort.go
import (
   "fmt"
   "math/rand"
   "time"
)
const N = 1 * 1024 * 1024
func main() {
   rand.Seed(time.Now().UnixNano())
   s := make([]int, N)
   for i := 0; i < len(s); i++ {
      s[i] = rand.Int()
   //   s[i] = i
   }
   fastsort(s)
   fmt.Println("s[0]=",s[0],"s[1]=", s[1],"s[",N-1,"]=", s[N-1])
}
func fastsort(A []int) {
   if len(A) < 2 {
      return
   }
   lowerA, upperA := partition(A)
   fastsort(lowerA)
   fastsort(upperA)
}
func partition(A []int) ([]int, []int) {
   // pivotIndex := rand.Intn(len(A))   // randomly select a pivot
   // A[pivotIndex], A[len(A)-1] = A[len(A)-1], A[pivotIndex]
   lower := 0
   for i := 0; i < len(A); i++ {
      //if A[i] < A[pivotIndex] {
      if A[i] < A[len(A)-1] {
         A[lower], A[i] = A[i], A[lower]
         lower++
      }
   }
   A[lower], A[len(A)-1] = A[len(A)-1], A[lower]
   return A[0:lower], A[lower+1 : len(A)]
}
```

系统思维单元
4. The file abstraction　文件与文件系统

- 持久存储数据和程序文件，下电后还存在
- 文件系统是一颗树
  - 叶子节点（文件），内部节点（目录）
  - 文件名（路径）；绝对路径，相对路径
  - 目录；缺省目录，当前目录，父目录，根目录
- 数据与元数据
  - 文件格式，文件大小，访问权限
- 如何读入文件
  - 有利于定位及操作
- 如何支持循环
  - 迭代操作文件元素



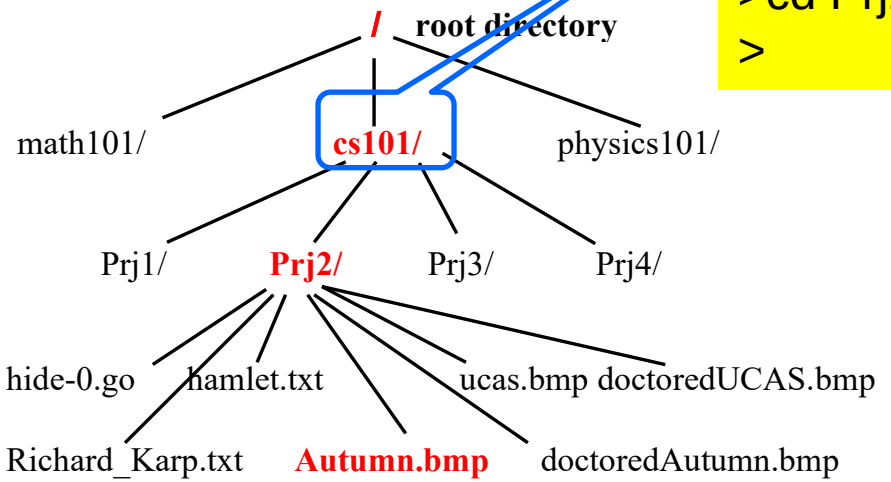
4.1 文件与文件系统

- 文件系统是一颗树
  - Leaf nodes are files 文件; internal nodes are directories 目录 (special files)

- 用文件名（又称为文件路径，简称路径）指明某一文件
  - Absolute path 绝对路径: all the way from the root (/)
    - Absolute path of **Autumn.bmp**: /cs101/Prj2/Autumn.bmp
  - Relative path 相对路径 of **Autumn.bmp**
    - Related to the *current directory* 当前目录，
      i.e., *working directory* 工作目录
    - ./Autumn.bmp if the working directory is /cs101/Prj2/
    - ../Prj2/Autumn.bmp if the working directory is /cs101/Prj2/
  - Home directory 缺省目录
    - The default directory when log in. Assume /cs101 is the home directory

主目录是/cs101/

```
>pwd
/cs101/
>cd Prj2
>
```



Hide text in a picture

- Write a program hide-0.go
  - to hide the text of Shakespeare's *Hamlet*
    - in a picture file Autumn.bmp
    - such that the doctored file shows no visible difference from the original picture
- and another program show-0.go to recover the text

```
HAMLET

DRAMATIS PERSONAE

CLAUDIUS      king of Denmark. (KING CLAUDIUS:)

HAMLET        son to the late, and nephew to the present king.

.....

ACT ISCENE I  Elsinore. A platform before the castle.
.....
PRINCE FORTINBRAS          Let four captains
                  Bear Hamlet, like a soldier, to the stage;
                  For he was likely, had he been put on,
                  To have proved most royally: and, for his passage,
                  The soldiers' music and the rites of war
                  Speak loudly for him.
                  Take up the bodies: such a sight as this
                  Becomes the field, but here shows much amiss.
                  Go, bid the soldiers shoot.
                  [A dead march. Exeunt, bearing off the dead
                  bodies; after which a peal of ordnance is shot off]
.....
] 换行键0x0A
```

Original picture

After careless hiding

After careful hiding

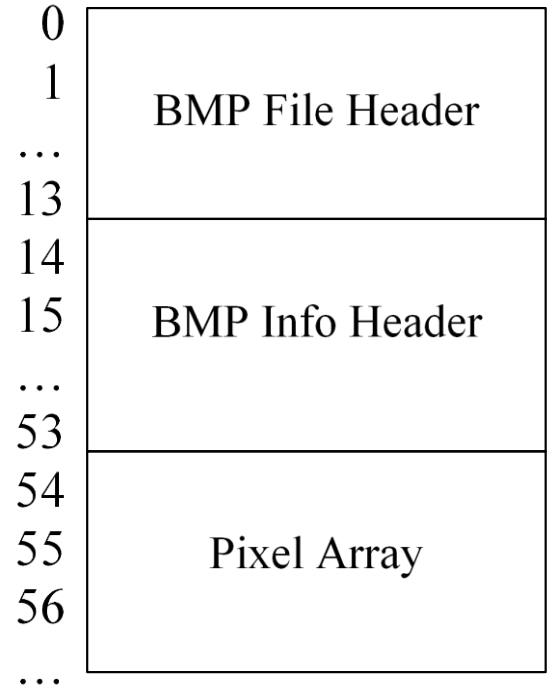
一个动态网页思路

4.2 数据与元数据

- A file is a group of bits ***and*** may be structured
  - 文件是一组比特，可能是有结构的
    - 无结构例子：F(500) = **1394 2322 4561 6978 8013 9724 3828 7040 7283 9500 7025 6587 6973 0726 4108 9629 4832 5571 6228 6329 0691 5576 5887 6222 5212 94125**
- 有结构例子：Data and metadata of file Autumn.bmp
  - Data: bits of the actual picture (Pixel Array)
  - Metadata 元数据 : data about the picture data 文件格式
    - BMP format in the file: File Header, Info Header
    - Other data associated with the file

Addresses 0~53 hold metadata

The pixel array for actual image data starts at address 54



Autumn.bmp The extension (扩展名) bmp says it's a bit map image file

Other types of metadata

- Can be seen by running “ls -l Autumn.bmp”
  - ls -l Autumn.bmp中，l是小写的L，不是大写的I

  > -rw-rw-rw- 1 zxu zxu 9144630 Jul 22 2020 Autumn.bmp
  - The file name, the file size, the time of creation (last modification)
  - Access permissions 三类三种权限
    - Rights to read, write, and execute a file by the owner of the file, by the group the owner belonging to, and by other users
- Example of access permissions
  - ioutil.WriteFile("./doctoredAutumn.bmp", p, 0666)
  - Every user can read and write, but cannot execute
    - -rw-rw-rw-
    - 0666 = 0110110110

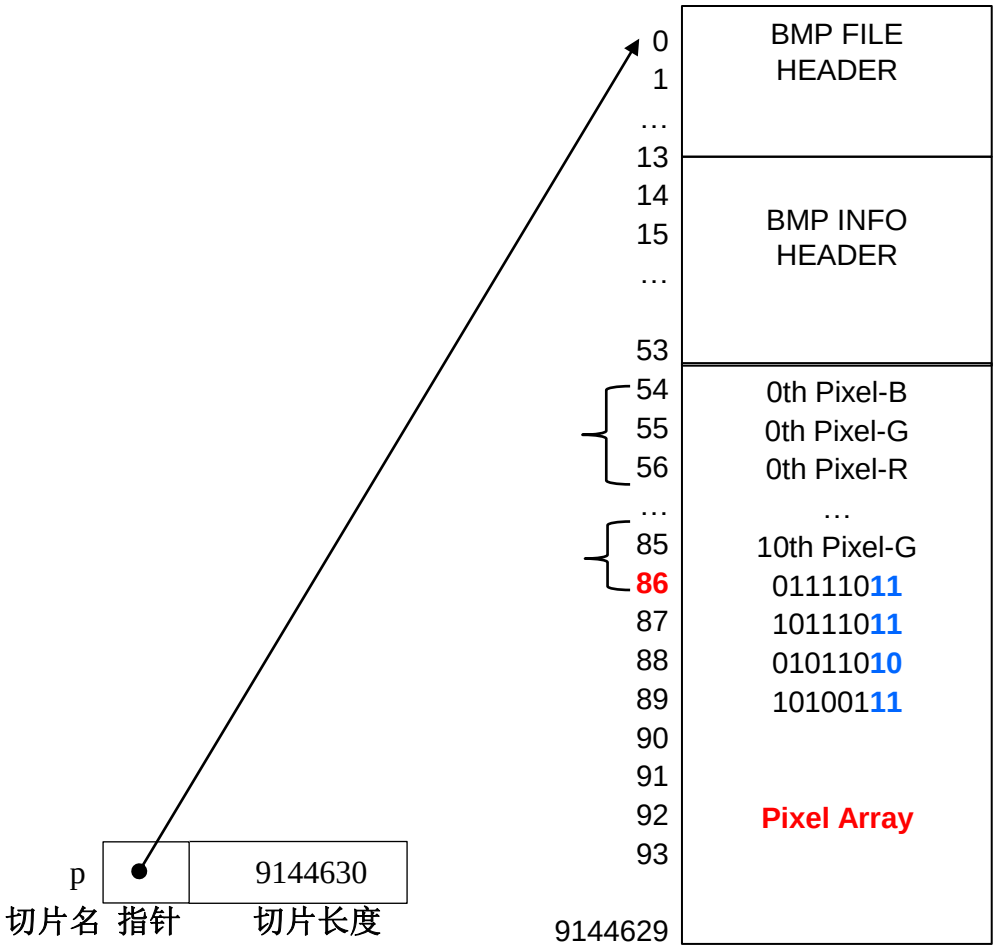
Owner			Group			Others		
r	w	e	r	w	e	r	w	e
1	1	-	1	1	-	1	1	-

0666：拥有者（用户本人）、组、他人可读、可写，不可执行
这是一个数据文件，不是程序文件

0700表示什么权限？

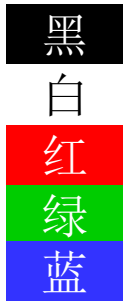
4.3 如何读文件

- 将文件以合适的格式读进内存，便于**定位**
  - 后续操作方便地定位到文件所需区域并处理该区域的数据
- 什么是合适的格式呢？
  - 是**字节切片**（byte slice）
- 使用Go语言提供的库函数
  - p, _ := ioutil.ReadFile("./Autumn.bmp")
  - 将当前目录中的Autumn.bmp图像文件读进内存的字节切片变量p
  - 文件内容被读进内存，形成一个包含9144630个元素的字节数组，成为切片p的底层数组

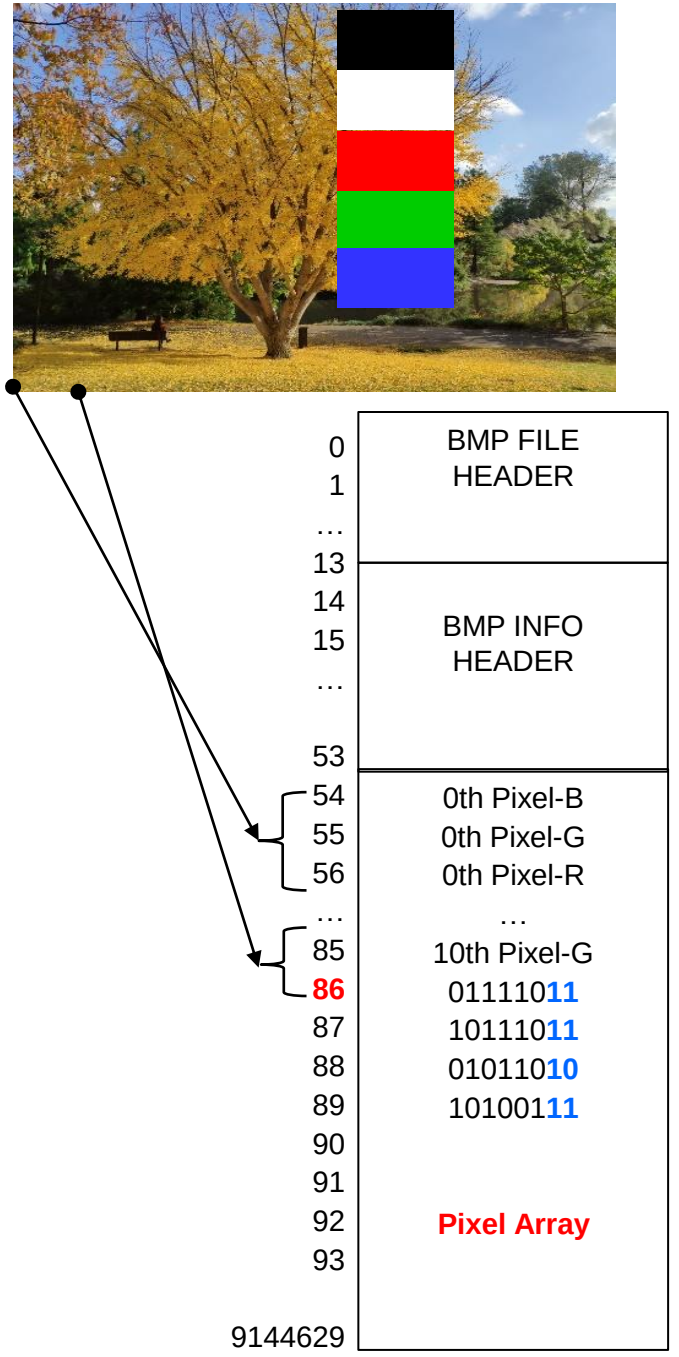


How is the image of Autumn.bmp stored in Pixel Array?

- Pixel = Picture Element
- Pixel Array holds the pixels of the image, starting from the low-left corner going right, and then up, row by row
- Each pixel has three bytes for
  - Color depth values of RGB, i.e., the primary colors of red, green, blue
  - BGR values = (0, 0, 0) → Black
  - BGR values = (255, 255, 255) → White
  - BGR values = (0, 0, 255) → Red
  - BGR values = (0, 255, 0) → Green
  - BGR values = (255, 0, 0) → Blue



- The first pixel is the 0th element of Pixel Array
  - Uses addresses 54, 55, and 56 to store its three BGR color depth values

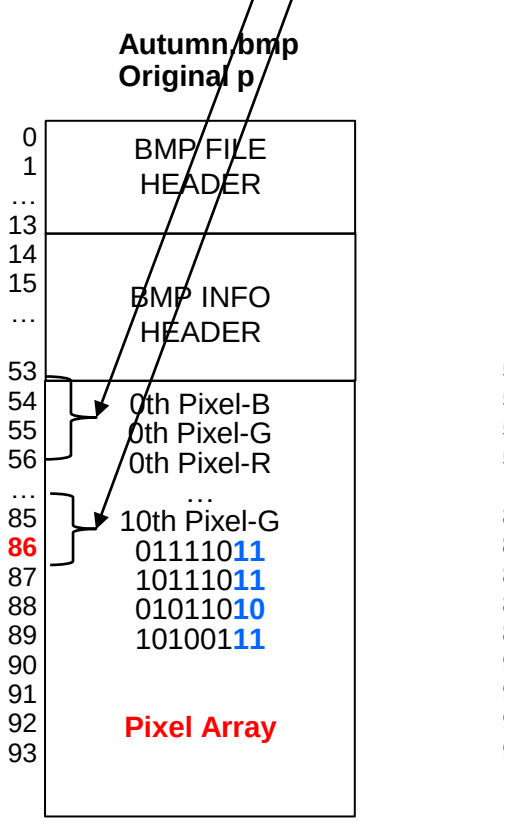


4.4 如何在图片文件中隐藏信息

- 如何在图片文件Autumn.bmp中隐藏文本文件hamlet.txt的长度

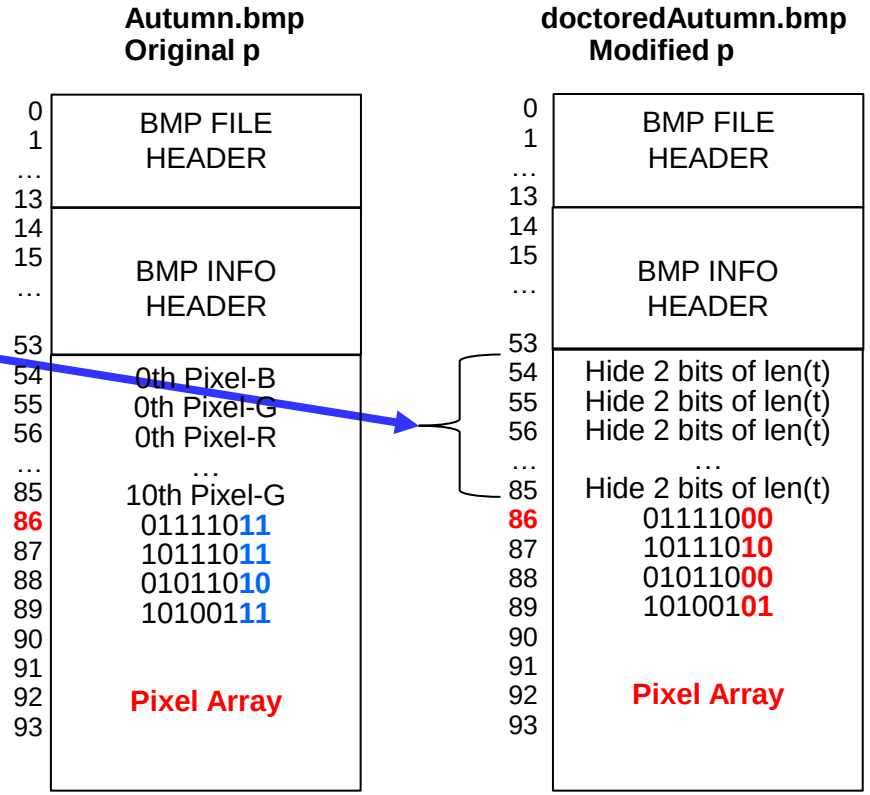
- p, _ := ioutil.ReadFile("./Autumn.bmp") to read the image file into byte slice p

- Recall that a function can return multiple values
  - This function returns two values, the second of which is not needed by this code
  - Use a placeholder symbol '_'
  - Also called the blank identifier



How to hide the length of a text file in a picture?

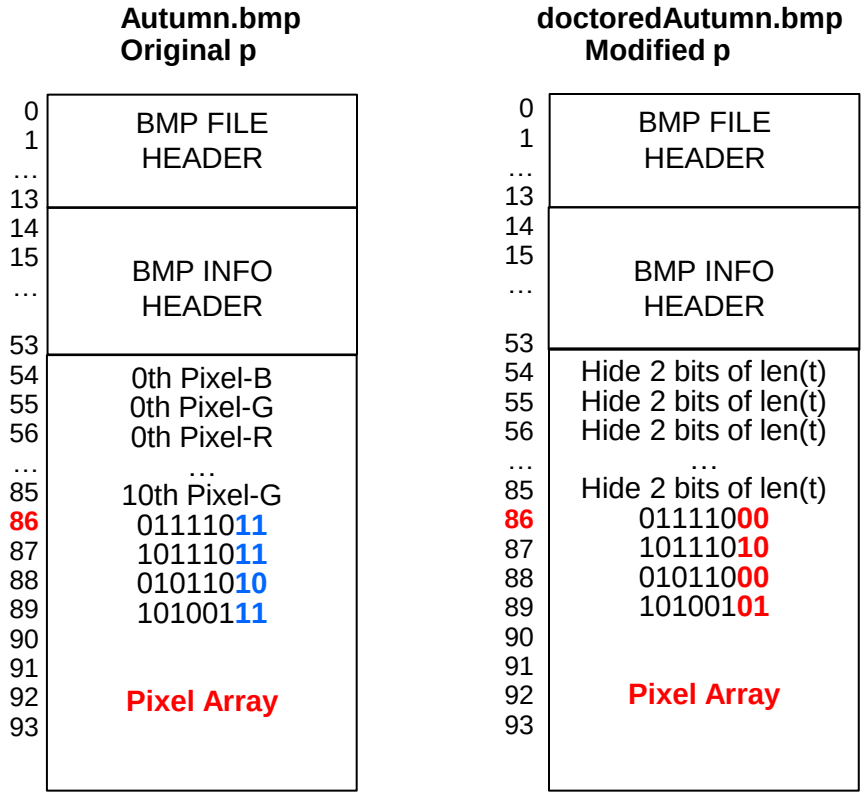
- p, _ := ioutil.ReadFile("./Autumn.bmp")
  to read the image file into byte slice p
- t, _ := ioutil.ReadFile("./hamlet.txt")
  to read the text file into byte slice t
- Length len(t) is a 64-bit integer
  - Hide every 2 bits in a byte of p
  - Need 32 bytes
  - S = 54, T = 32
- modify(len(t), p[S:S+T], T)
  to hide len(t) in p[54:86]



How to hide the length of a text file in a picture?

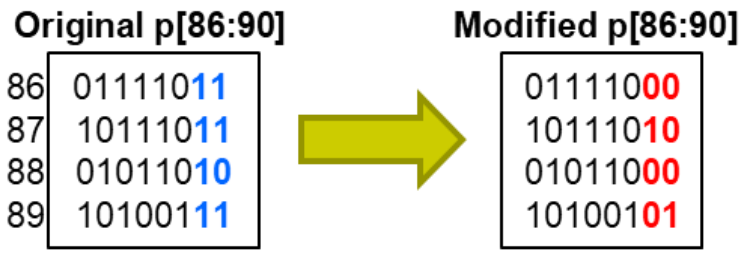
- p, _ := ioutil.ReadFile("./Autumn.bmp")
  to read the image file into byte slice p
- t, _ := ioutil.ReadFile("./hamlet.txt")
  to read the text file into byte slice t
- Length len(t) is a 64-bit integer
  - Hide every 2 bits in a byte of p
  - Need 32 bytes
  - S = 54, T = 32
- modify(len(t), p[S:S+T], T)
  to hide len(t) in p[54:86]

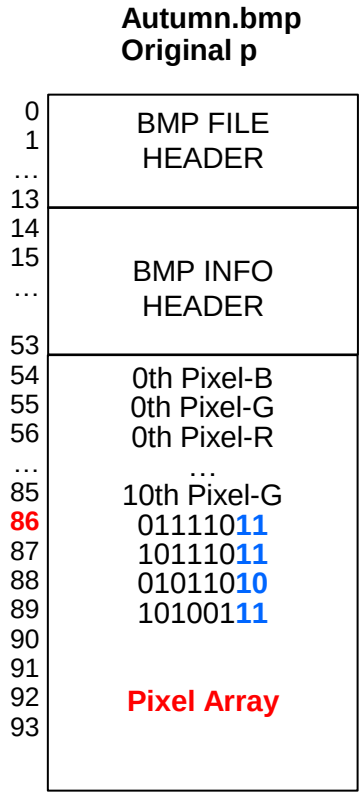
```
func modify(txt int, pix []byte, size int) {
   for i := 0; i < size; i++ {
      replace last 2 bits of pix[i]
           with the last 2 bits of txt
      repeat with the next 2 bits of txt
   }
}
```

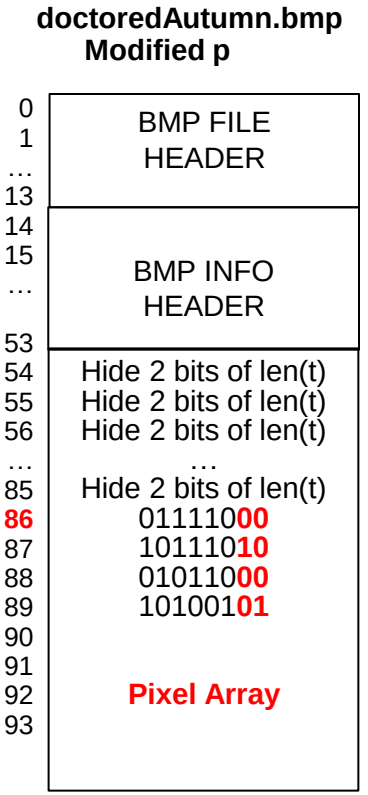


How to hide the contents of a text file in a picture?

- p, _ := ioutil.ReadFile("./Autumn.bmp")
  to read the image file into byte slice p
- t, _ := ioutil.ReadFile("./hamlet.txt")
  to read the text file into byte slice t
- t[0] holds the 1st character ‘H’ = 72
- modify(int(t[0]), p[S+T:S+T+C], C)
  where
  - t[0] is ‘H’ = 72 = **01001000**
  - S = 54, T = 32, C is 4
  - p[S+T:S+T+C] is p[86:90]







How to hide the contents of a text file in a picture?

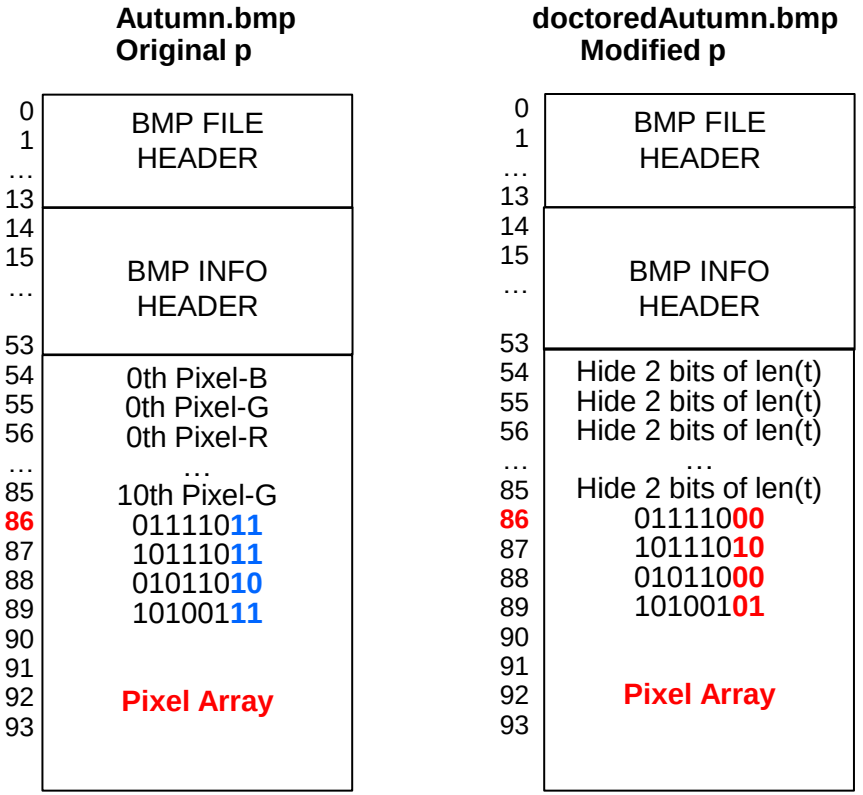
- p, _ := ioutil.ReadFile("./Autumn.bmp")
  to read the image file into byte slice p
- t, _ := ioutil.ReadFile("./hamlet.txt")
  to read the text file into byte slice t
- 使用循环语句隐藏t[i], i = 0~len(t)

```
for i:=0; i<len(t); i++{
   offset := S+T+(i*4)
   modify(int(t[i]), p[offset:offset+C], C)
}
```

比例因子是4，因为每个字符有8位，需要p的4个字节。这4个字节的偏移量分别是0、1、2、3。第0次迭代执行modify(int(t[0]), p[86:90], 4)，修改P[86]， P[87]， P[88]， P[89]

基址+索引*4+偏移量

- 回忆支持循环的基址+索引+偏移量寻址模式



How to hide the contents of a text file in a picture?

- p, _ := ioutil.ReadFile("./Autumn.bmp")
  to read the image file into byte slice p
- t, _ := ioutil.ReadFile("./hamlet.txt")
  to read the text file into byte slice t
- To hide all t[i], i = 0 to len(t)

```
for i:=0; i<len(t); i++{
   offset := S+T+(i*4)
   modify(int(t[i]), p[offset:offset+C], C)
}
```

- Each iteration hides t[i] in
  p[S+T+(i*4):S+T+(i*4)+C]
  - Where S = 54, T = 32, C = 4
- That is, t[i] is hidden in
  p[86+(i*4)], p[86+(i*4)+1],
  p[86+(i*4)+2], p[86+(i*4)+3]
- E.g., t[1]='A', is hidden in p[90:94]

Autumn.bmp
Original p

0 1 … 13	BMP FILE HEADER
14 15 … 53	BMP INFO HEADER
54 55 56 … 85 **86** 87 88 89 90 91 92 93	0th Pixel-B 0th Pixel-G 0th Pixel-R … 10th Pixel-G 01111011 10111011 01011010 10100111 **Pixel Array**

doctoredAutumn.bmp
Modified p

0 1 … 13	BMP FILE HEADER
14 15 … 53	BMP INFO HEADER
54 55 56 … 85 **86** 87 88 89 90 91 92 93	Hide 2 bits of len(t) Hide 2 bits of len(t) Hide 2 bits of len(t) … Hide 2 bits of len(t) 01111**00** 10111**10** 01011**00** 10100**01** **Pixel Array**

Check results

- Write the complete hide-0.go
- Execute and display result

> go run hide-0.go

> display doctoredAutumn.bmp

- The Text Hider project
  - Produce hide.go with good coding practices
  - Also need to write show.go

- Change hide-0.go to hide-1.go
  - by modifying the most significant 2 bits (the rightmost 2 bits) of each byte of Pixel Array

> go run hide-1.go

> display doctoredAutumn.bmp

Original Autumn.bmp

doctoredAutumn.bmp
Modifying rightmost 2 bits

doctoredAutumn.bmp
Modifying leftmost 2 bits