1.绪论

(b) 计算模型

To measure is to know.

If you can not measure it,
you can not improve it.

\- Lord Kelvin

邓俊辉

deng@tsinghua.edu.cn

算法分析

❖ 两个主要方面　$\boxed{\text{正确}}$：　算法功能与问题要求一致？

数学证明？可不那么简单...

$\boxed{\text{成本}}$：　运行时间 + 所需存储空间

如何度量？如何比较？

❖ 考察： $T_A(P)$ = 算法A求解问题实例P的计算成本

意义不大，毕竟...可能出现的问题实例太多

如何归纳概括？

❖ 观察： 问题实例的$\boxed{\text{规模}}$，往往是决定计算成本的主要因素

❖ 通常： 规模接近，计算成本也接近

规模扩大，计算成本亦上升

特定算法 + 不同实例

- 令$T_A(n)$ = 用算法A求解某一问题规模为n的实例，所需的计算成本

  讨论特定算法A（及其对应的问题）时，简记作T(n)

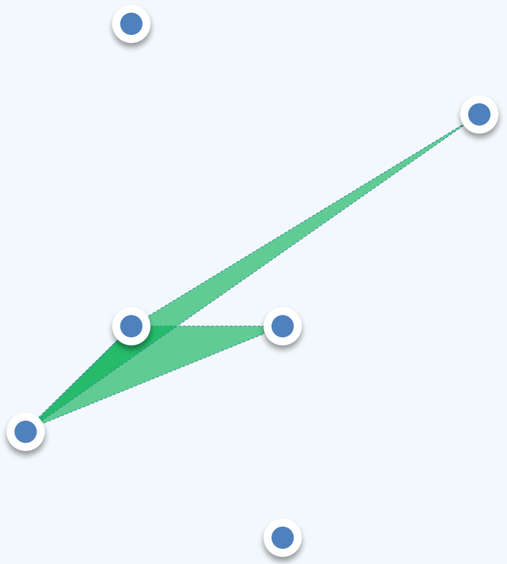
- 然而，这一定义仍有问题...

- 观察：同一问题等规模的不同实例，计算成本不尽相同，甚至有实质差别

- 例如：在平面上的n个点中，找到所成三角形面积最小的三个点

  以蛮力算法为例，最坏情况下需枚举所有C(n,3)种组合

  但运气好的话...



- 既然如此，又该如何定义T(n)呢？

- 稳妥起见，取$T(n) = \boxed{\boxed{\max}\ \{\ T(P)\ |\ |P| = n\ \}}$

  亦即，在规模同为n的所有实例中，只关注**最坏**（成本最高）者

特定问题 + 不同算法

- 同一问题通常有多种算法，如何评判其优劣？
- 实验统计是最直接的方法，但足以准确反映算法的真正效率？
- 不足够！
  - 不同的算法，可能更适应于不同**规模**的输入
  - 不同的算法，可能更适应于不同**类型**的输入
  - 同一算法，可能由不同**程序员**、用不同程序**语言**、经不同**编译器**实现
  - 同一算法，可能实现并运行于不同的**体系结构**、**操作系统**...
- 为给出**客观**的评判，需要抽象出一个**理想**的平台或模型
  - 不再依赖于上述种种具体的因素
  - 从而直接而准确地描述、测量并评价算法

TM: Turing Machine

❖ Tape　依次均匀地划分为单元格

　　各注有某一字符，默认为'#'

❖ Alphabet 字符的种类有限

❖ Head　总是对准某一单元格，并可读取和改写其中的字符

　　每经过一个节拍，可转向左侧或右侧的邻格

❖ State　TM总是处于有限种状态中的某一种

　　每经过一个节拍，可（按照规则）转向另一种状态

❖ Transition Function：(q, c; d, L/R, p)

若当前状态为q且当前字符为c，则将当前字符**改写**为d；**转向**左侧/右侧的邻格；**转入**p状态

一旦转入特定的状态'h'，则**停机**

TM：Increase

❖ 功能：

将二进制非负整数加一

❖ 算法：

全'1'的后缀翻转为全'0'

原最低位的'0'或'#'翻转为'1'

❖ (<, 1, 0, L, <) //左行，1->0

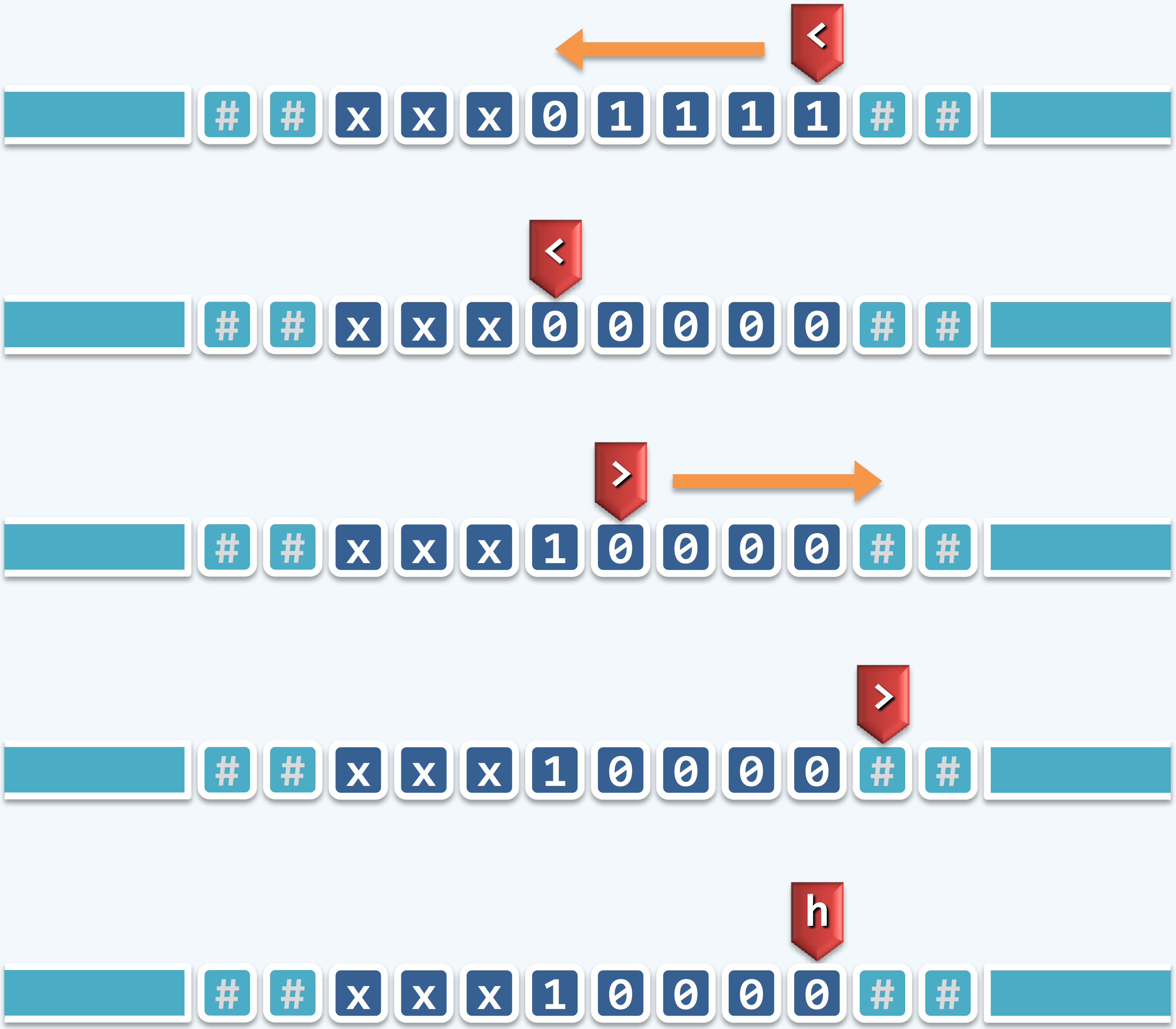
(<, 0, 1, R, >) //掉头，0->1

(<, #, 1, R, >) //？

(>, 0, 0, R, >) //右行

(>, #, #, L, h) //复位

❖ 规范 ～ 接口



RAM: Random Access Machine

- 寄存器顺序编号，总数没有限制 //但愿如此

  R[0], R[1], R[2], R[3], ...

- 每一基本操作仅需常数时间 //循环及子程序本身非基本操作

R[i] <- c	R[i] <- R[R[j]]	R[i] <- R[j] + R[k]
R[i] <- R[j]	R[R[i]] <- R[j]	R[i] <- R[j] - R[k]
IF R[i] = 0 GOTO l	IF R[i] > 0 GOTO l	GOTO l STOP

- 与TM模型一样，RAM模型也是一般计算工具的简化与抽象

  使我们可以独立于具体的平台，对算法的效率做出可信的比较与评判

- 在这些模型中

  算法的运行时间 $\propto$ 算法需要执行的基本操作次数

  T(n) = 算法为求解规模为n的问题，所需执行的基本操作次数

RAM: Floor

❖ 功能：向下取整的除法，$0 <= c$, $0 < d$

$$\lfloor c/d \rfloor = \max\{ x \mid d \cdot x <= c \}$$

$$= \max\{ x \mid d \cdot x < 1 + c \}$$

❖ 算法： 反复地从R[0] = 1 + c中减去R[1] = d

统计在下溢之前，所做减法的次数x

```
0    R[3] <- 1             //increment
1    R[0] <- R[0] + R[3]   //c++
2    R[0] <- R[0] - R[1]   //c -= d
3    R[2] <- R[2] + R[3]   //x++
4    IF R[0] > 0 GOTO 2    //if c > 0 goto 2
5    R[0] <- R[2] - R[3]   //else x-- and
6    STOP    //return R[0] = x = ⌊c/d⌋
```

Step	IR	R[0]	R[1]	R[2]	R[3]
0	0	12	5	0	0
1	1	^	^	^	1
2	2	13	^	^	^
3	3	8	^	^	^
4	4	^	^	1	^
5	2	^	^	^	^
6	3	3	^	^	^
7	4	^	^	2	^
8	2	^	^	^	^
9	3	0	^	^	^
10	4	^	^	3	^
11	5	^	^	^	^
12	6	2	^	^	^

课后

- ❖ 举例：随问题实例规模增大，同一算法的求解时间可能波动甚至下降
- ❖ 在哪些方面，现代电子计算机仍未达到RAM模型的要求？
- ❖ 在TM、RAM等模型中衡量算法效率，为何通常只需考察运行时间？
- ❖ 图灵机Increase中，以下这条指令可否省略：

  <0, #, 1, R, 1>

- ❖ 设计一个图灵机，实现对正整数的减一（Decrease）功能