

01-B-2 图灵机 (TM, Turing machine)

#数据结构邓神

同一个问题通常有多个算法，如何评价优劣？

实验统计是最直接的方法，但是足以准确反映算法的真正效率吗？

❖ 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**...

所以为了给出客观的评价，往往需要抽象出一个理想的平台或者是模型

不再依赖于上述的种种具体的因素

从而准确的描述测量和评价算法

TM : Turing machine （图灵机）

图灵机模型具有以下要件：

Tape ：纸带或者磁带 均匀划分为单元格 各自标记为一个特殊的符号默认为 "#"

Alphabet（字母）：字符的种类有限

Head ：总是对准某一个单元格，并且可以读取和改写其中的字符（读写头）

每经过一个节拍（或者可以叫做一个单位时间），可以转向左侧或者右侧的临格

State ：TM 总是处于有限种状态中的一种，每经过一个节拍，可（按照规则）转向另外一种状态

规则如下：

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

q ：当前状态 c：读写头当前所对的单元内存储的字符 可以理解为当前的状态

d ：在当前修改为的字符，L/R 指挥读写头运动

p ：下一个新的状态

如果当前状态为 q 且字符为 c，则将字母改写为 d；转向左侧/右侧的临格；转入 p 状态，一旦转入特定状态 "h"，则停机

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**

图灵机具体实例

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

The diagram illustrates the execution of the 'Increase' Turing Machine on a binary tape. The tape initially contains 'x's followed by a sequence of '1's. The head (red arrow) moves left, then right, flipping '1's to '0's. When it finds a '0' or '#', it flips it to '1'. If it reaches the end of the tape (indicated by '#'), it resets to the start.

规范 ~ 接口