01-B3

1.绪论

计算模型

RAM

There is an infinite set A

that is not too big.

\- J. von Neumann

邓俊辉

deng@tsinghua.edu.cn

Random Access Machine

❖ 寄存器顺序编号，总数没有限制：R[0], R[1], R[2], R[3], ... //但愿如此

❖ 通过编号，可以直接访问任意寄存器 //call-by-rank

❖ 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
```

Random Access Machine

- 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

Floor Division

❖ 对任意 $0 \le c$ 和 $0 < d$，计算

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

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

❖ 例如：

$\lfloor 5/2 \rfloor = 2$ $\lfloor 2015/56 \rfloor = 35$

$\lfloor 6/3 \rfloor = 2$ $\lfloor 2016/36 \rfloor = 56$

$\lfloor 12/5 \rfloor = 2$

❖ 思路： 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

Step	IR	R[0]	R[1]	R[2]	R[3]
0	0	12	5	0	0
		2			

Floor Division

❖ **RAM算法**

```
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

❖ 举例说明：随着问题实例规模增大，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？

❖ 图灵机算法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