04-C

栈与队列

进制转换

Hickory, Dickory, Dock

The mouse ran up the clock

wan, twan, tetbera, metbera, pimp,

setbera, letbera, bovera, dovera, dick,

wanadick, twanadick, tetberadick, metberadick, pimpdick,

setberadick, letberadick, boveradick, doveradick, bumfit,

wanabumfit, ...

邓俊辉

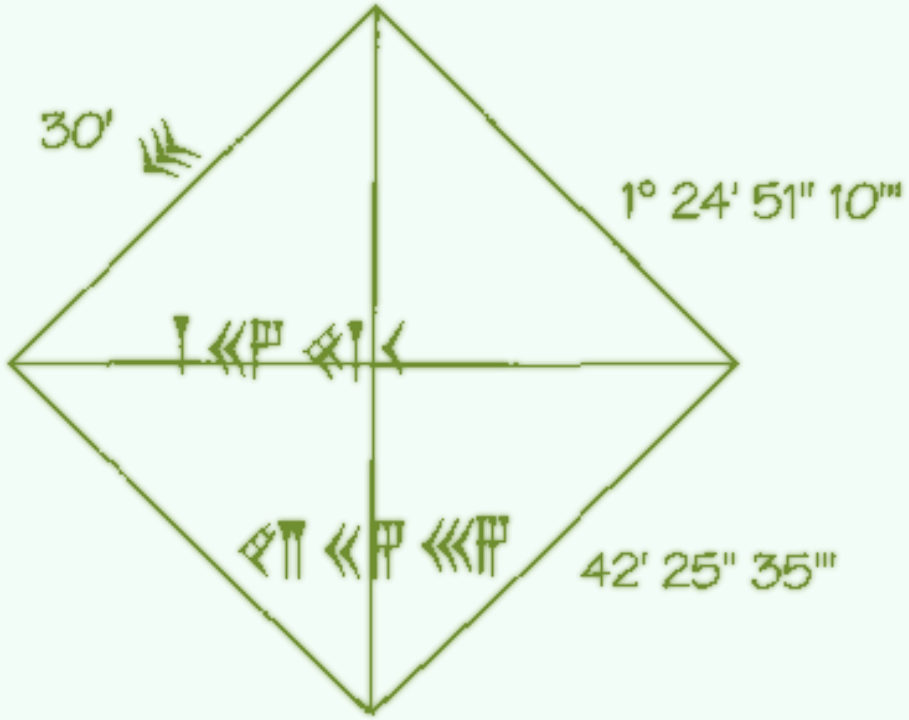
deng@tsinghua.edu.cn

进制转换

❖ 给定任一10进制非负整数，将其转换为λ进制表示形式

- $12345_{(10)} = 30071_{(8)}$
- printf("%d | %I64d | %b | %o | %x" , n);

❖ 巴比伦楔形文字 (Babylonic cuneiform) 中的60进制...



❖ 1°24'51''10'''

= 1 + 24/60 + 51/60^2 + 10/60^3

= 1.41421296296296... //正方形的对角线

❖ 误差 ~ |1°24'51''10''' - sqrt(2)|

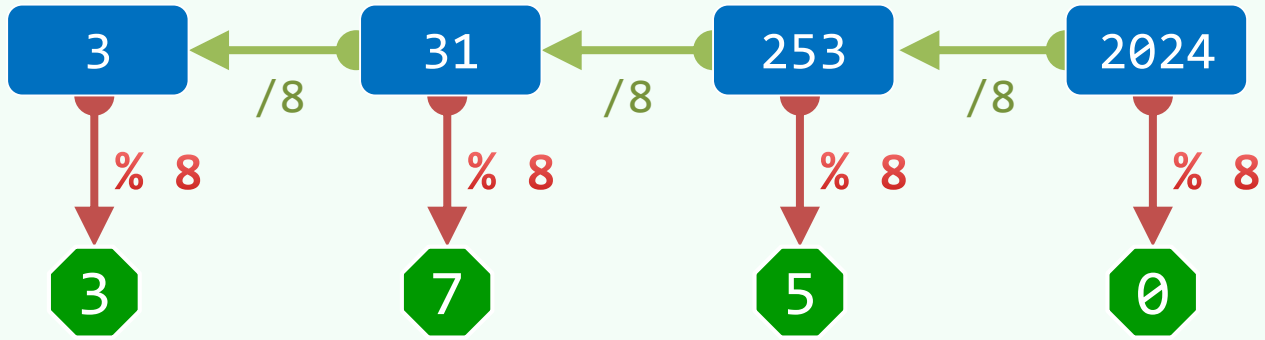
< 0.000,000,6 = 0.6 x 10^{-6} //即便边长为1km，误差亦不足1mm

短除法：整商 + 余数

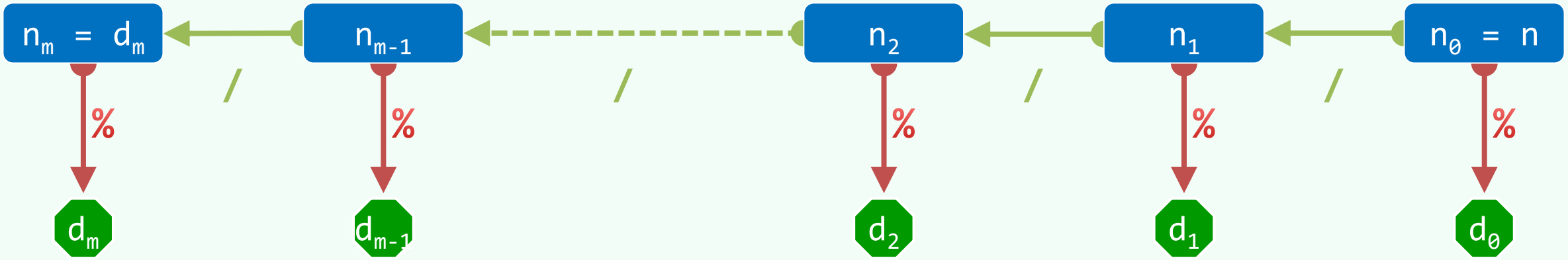
❖ 设：$n = (d_m \dots d_2 d_1 d_0)_\lambda = d_m \cdot \lambda^m + \cdots + d_2 \cdot \lambda^2 + d_1 \cdot \lambda^1 + d_0 \cdot \lambda^0$

令：$n_i = (d_m \dots d_{i+2} d_{i+1} d_i)_\lambda$

❖ 则有：$n_{i+1} = n_i \,/\, \lambda$ 和 $d_i = n_i \,\%\, \lambda$



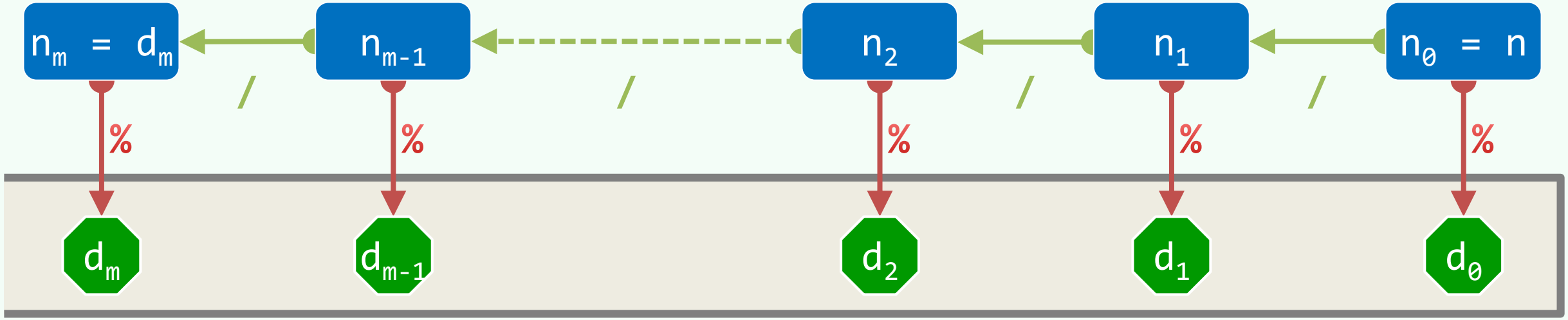
❖ 如此对λ反复整除、留余，即可自低而高得出λ进制的各位



难点 + 算法

❖ 位数 $m+1 = \lfloor \log_\lambda n \rfloor + 1$ 在事先不能简便地确定，辅助空间如何才能既足够，亦不浪费？

❖ 转换后的各数位是自低而高得到的，如何自高而低输出？



❖ 若使用向量，则扩容策略必须得当；若使用列表，则多数接口均被闲置

❖ 使用栈，既可满足以上要求，亦可有效控制计算成本

实现

```
void convert( Stack<char> & S, unsigned long long n, int base ) { //0 < n
   char digit[] = "0123456789ABCDEF"; //2 <= base <= 16，如有必要可扩充
   while ( n > 0 ) //由低到高，逐一计算出新进制下的各数位
      {  S.push( digit[ n % base ] ); n /= base;  } //余数入栈，n更新为除商
} //新进制下由高到低的各数位，自顶而下保存于栈S中

main() {
   /* get n and base */
   Stack<char> S;  (0 < n) ? convert( S, n, base ) : S.push('0');
   while ( ! S.empty() ) S.pop(); //逆序输出
}
```