

11. 串

(f2) Karp-Rabin算法：散列

邓俊辉

deng@tsinghua.edu.cn

数位溢出

❖ 如果 $|\Sigma|$ 很大，模式串 P 较长，其对应的**指纹**将**很大**

比如，若将 P 视作 $|P|$ 位的 $|\Sigma|$ 进制**自然数**，并将其作为指纹...

❖ 仍以ASCII字符集为例 // $|\Sigma| = 128 = 2^7$

只要 $|P| > 9$ ，则指纹的长度将至少是： $7 \times 10 = 70$ bits

❖ 然而，目前的字长一般也不过**64**位 //存储不便

❖ 而更重要地，指纹的**计算**及**比对**，将不能在 $O(1)$ 时间内完成 //RAM!?

准确地说，需要 $O(|P| / 64) = O(m)$ 时间；总体需要 $O(n * m)$ 时间 //与蛮力算法相当

❖ 有何高招...

散列压缩

❖ 基本构思：通过对比压缩后的指纹，确定匹配位置 //更接近于人类的指纹

❖ 关键技巧：借助散列，将指纹压缩至存储器支持的范围

比如，采用模余函数： $\text{hash}(\text{key}) = \text{key} \% M$ //若取 $M = 97 \dots$

❖ $P = [8\ 2\ 8\ 1\ 8]$ // $\text{hash}(P) = \text{hash}(82818) = [77]$

$T = 2\ 7\ 1\ [8\ 2\ 8\ 1\ 8]\ 2\ 8\ 4\ 5\ 9\ 0\ 4\ 5\ 2\ 3\ 5\ 3\ 6$

$[2\ 7\ 1\ 8\ 2]$ //22

$[7\ 1\ 8\ 2\ 8]$ //48

$[1\ 8\ 2\ 8\ 1]$ //45

$[8\ 2\ 8\ 1\ 8]$ //[77]

散列冲突

❖ 注意：hash()值相等，并非匹配的充分条件... //好在必要

因此，通过hash()筛选之后，还须经过严格比对，方可最终确定是否匹配...

❖ P = 1 8 2 8 4 //仍取M = 97，则hash(18284) = 48

T = 2 7 1 8 2 8 1 8 2 8 4 5 9 0 4 5 2 3 5 3 6

2 7 1 8 2 //22

7 1 8 2 8 //48

. . .

1 8 2 8 4 //48

❖ 既然是通过散列压缩，指纹冲突就在所难免

适当选取散列函数，可以大大降低冲突的可能

快速指纹计算

❖ $\text{hash}()$ 的计算，似乎每次均需 $O(|P|)$ 时间

有可能加速吗？



$\text{suffix}(T_{i-1})$

❖ 回忆一下，进制转换算法...

$\text{prefix}(T_i)$



❖ 观察 相邻的两次计算之间，存在着某种相关性

相邻的两个指纹之间，也有着某种相关性

❖ 利用上述性质，即可在 $O(1)$ 时间内，由上一指纹得到下一指纹...