软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2011,22(3):353−365 [doi: 10.3724/SP.J.1001.2011.03867]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

串的快速连续弱哈希及其应用*

徐泽明[1,2+], 侯紫峰[1,3]

[1](中国科学院 计算技术研究所,北京 100190)

[2](中国科学院 研究生院,北京 100049)

[3](联想研究院,北京 100085)

Fast Continuous Weak Hashes in Strings and Its Applications

XU Ze-Ming[1,2+], HOU Zi-Feng[1,3]

[1](Institute of Computing Technology, The Chinese Academy of Sciences, Beijing 100190, China)

[2](Graduate University, The Chinese Academy of Sciences, Beijing 100049, China)

[3](Lenovo Corporate R&D, Beijing 100085, China)

+ Corresponding author: E-mail: zeming.xu@gmail.com

Xu ZM, Hou ZF. Fast continuous weak Hashes in strings and its applications. *Journal of Software*, 2011,22(3): 353−365. http://www.jos.org.cn/1000-9825/3867.htm

Abstract: In this paper, the fast continuous weak Hash (FCWH) in strings is proposed and its theoretic and practical applications are investigated. First, FCWH is conceptualized and a uniform construction framework for FCWH is formulized from an algebraic viewpoint. Secondly, the theoretical and experimental collision probabilities of FCWH are analyzed, and the related work by Michael O. Rabin is generalized and strengthened. Finally, by generalizing the Karp-Rabin algorithm for string-matching problem, FCWH is applied to solve the problem of sequential extraction of common substrings (SECS), and, based on SECS, the express synchronization (X-Sync) protocol is designed to address the issue of real-time backup and the retrieval of multiple versions of a given document in the current environment of broadband communication network and cloud computing.

Key words: fast continuous weak Hash (FCWH); string-matching; sequential extraction of common substrings (SECS); express synchronization (X-Sync); finite group; finite ring; finite field

摘 要: 提出串的快速连续弱哈希(fast continuous weak Hash,简称 FCWH),并研究它在理论和工程上的应用.首先提出 FCWH 的概念,从代数结构角度统一规划该类哈希的构造框架;然后对哈希冲突概率进行理论分析和实验数据分析,推广并加强了 Rabin 的相关工作;最后,通过推广串匹配的 Karp-Rabin 算法,应用 FCWH 解决顺序抽取公共子串问题(sequential extraction of common substrings,简称 SECS),并据此设计快速同步协议 X-Sync 来解决当今宽带网络和云计算环境下文档多版本内容的实时备份检索.

关键词: 快速连续弱哈希(FCWH);串匹配;顺序抽取公共子串(SECS);快速同步(X-Sync);有限群;有限环;有限域

中图法分类号: TP301 **文献标识码**: A

本研究由 Knuth[1],Rabin[2]和 Karp[3]关于哈希和串匹配的工作所触发.哈希是算法设计中的一种重要技术.

* 收稿时间: 2010-01-31; 修改时间: 2010-04-28; 定稿时间: 2010-05-12

一个好的哈希函数应满足两个条件[1]:

1) 该函数的计算应该迅速,即其的计算必须有效;

2) 它应该尽量减少冲突,即其产生的指纹相当清晰.

这两个条件在许多哈希设计中得到了体现.串匹配的 Karp-Rabin 算法[3−5]中的哈希满足上述的要求条件[5].串匹配问题是计算机科学(形式语言自动机)、计算机技术(编译、数据压缩)、工程应用(计算分子生物学)等诸多领域多年来密切关注的一类经典问题,有着广泛的应用背景[5].一般串匹配问题是这样描述的:已知长度为 n 位(bit)的模式串 X 和长度为 $m\geqslant n$ 位的文本串 Y,要求找出 X 在 Y 中最先出现的位置.Karp 和 Rabin 首先利用哈希处理这个问题[3].Karp-Rabin 串匹配算法的背后思想是这样的:当需要确定串 x 和串 y 是否相同时,首先检查哈希 $hash(x)$和 $hash(y)$;如果 $hash(x)\neq hash(y)$,那么 $x\neq y$,否则作进一步检查.

然而这两个条件,哈希计算的有效性和哈希指纹的清晰度并不总是一致的,因为计算有效性是机器相关的,而指纹清晰度是数据相关的[1].在许多情况下是优先考虑计算有效性.高效的计算通过简单的运算,甚至是直接的机器指令来获得.然而,运算的简单可能导致指纹不清晰.从理论上来讲,弱哈希(定义 2.1)所产生的指纹最不清晰.那么,在算法设计中,弱哈希是否毫无价值呢?

单独一个弱哈希可能没有什么用处,但是弱哈希序列另当别论.为了显示弱哈希序列的价值,本文提出快速连续弱哈希(fast continuous weak Hash,简称 FCWH),利用它设计实时算法来解决一种特殊的串匹配问题:顺序抽取公共子串(sequential extraction of common substrings,简称 SECS),并由此来设计快速同步协议(express synchronization,简称 X-Sync),以解决当今宽带网络和云计算环境下文档多历史版本的实时备份和检索.

本文第 1 节约定一些符号和机器指令.第 2 节形成和提出 FCWH 和 SECS 的概念.第 3 节~第 5 节从代数结构(有限域、有限群、有限环)统一规划 FCWH 的构造框架,构造现实可行的 FCWH.第 6 节对 FCWH 的冲突概率进行理论和实验分析.第 7 节利用 FCWH 解决 SECS.第 8 节是 X-Sync 协议.最后是总结.

1 符号和机器指令约定

1.1 符号约定

本文中,块专指一定长度的二进制位串.长度为 h 位的块称为基本块,所有的基本块所成的集合记为 u_h.文中 h,d,k,l,m,n 是 6 个自然数;其中,h 是基本块的长度,d 是基本块的并置重复次数.串 s 和串 t 的并置记为 st 或 $concat(s,t)$.对于$\forall b_0b_1\ldots b_{n-1}\in u_h$,定义符号函数 $sign$:$sign(b_0b_1\ldots b_{n-1})=b_0$.对于$\forall x\in u_h$,定义追尾函数 $tail_k$:$tail_k(x)=concat(x,0_k)$.对于$\forall b_0b_1\ldots b_{k-1}\in u_k,k>h$,定义截尾函数 $curtail_h$:$curtail_h(b_0b_1\ldots b_k)=b_{k-h+1}b_{k-h+2}\ldots b_k$.每个位都为 0 的基本块记为 0_h.对于$\forall x\in u_h$,引入算子$\circ$:$0\circ x=0_h$;$1\circ x=x$.

本文使用类 C 的程序设计语言描述算法.遵循 C 的惯例,数组(包括字符串)的下标从 0 开始计数.算法描述时,像 C 语言的无符号整数那样处理 u_h,记为 $uint_h$.

1.2 机器指令约定

在 u_h 上引入一些算术逻辑运算指令,它们是现代数字电子计算机所提供的[6]:加法($+_h$)、减法($-_h$)、乘法($\times_h$)、异或($\oplus_h$)、与($\wedge_h$)、或($\vee_h$)、逻辑左移($\langle\langle_h$)、逻辑右移($\rangle\rangle_h$).其中,$+_h$,$-_h$,$\times_h$ 是模 2^h 运算,即对于$\forall x,y\in u_h$,$x+_hy=(x+y) \bmod 2^h$,$x-_hy=(x-y) \bmod 2^h$,$x\times_hy=(x\times y) \bmod 2^h$;而$\oplus_h$,$\wedge_h$,$+_h$,$\langle\langle_h$,$\rangle\rangle_h$ 是按位运算,即对于$\forall a_0a_1\ldots a_{n-1},b_0b_1\ldots b_{n-1}\in u_h$,

$$a_0a_1\ldots a_{h-1}\oplus_hb_0b_1\ldots b_{h-1}=(a_0\oplus b_0)(a_1\oplus b_1)\ldots(a_{h-1}\oplus b_{h-1})$$

$$a_0a_1\ldots a_{h-1}\wedge_hb_0b_1\ldots b_{h-1}=(a_0\wedge b_0)(a_1\wedge b_1)\ldots(a_{h-1}\wedge b_{h-1})$$

$$a_0a_1\ldots a_{h-1}\rangle\rangle_h1=0a_0a_1\ldots a_{h-2}$$

$$a_0a_1\ldots a_{h-1}\vee_hb_0b_1\ldots b_{h-1}=(a_0\vee b_0)(a_1\vee b_1)\ldots(a_{h-1}\vee b_{h-1})$$

$$a_0a_1\ldots a_{h-1}\langle\langle_h1=a_1\ldots a_{h-1}0$$

2 快速连续弱哈希(FCWH)和顺序抽取公共子串(SECS)的概念

2.1 快速连续弱哈希的概念

以下定义中,$\forall x_0,x_1,\ldots,x_{d-1},x_d\in u_h$,$hash_d(x_0x_1\ldots x_{d-1})$是定义在 $u_{d\times h}$ 上的任意一个哈希函数.

定义 2.1. 如果存在算法 A,可以在时间复杂度 $O(d\times h)$下找到哈希冲突,即找到 $y_0,y_1,\ldots,y_{d-1}\in u_h$,使得 $x_0,x_1,\ldots,x_{d-1}\neq y_0,y_1,\ldots,y_{d-1}$,但是 $hash_d(x_0x_1\ldots x_{d-1})=hash_d(y_0y_1\ldots y_{d-1})$,那么称 $hash_d$ 是 $u_{d\times h}$ 上的一个弱哈希.

定义 2.2. 如果存在算法 A,使得 $hash_d(x_1x_2\ldots x_d)$可以由 $hash_d(x_0x_1\ldots x_{d-1})$在时间复杂度 $O(h)$下得到,那么称 $hash_d$ 是 $u_{d\times h}$ 上的一个连续哈希.

定义 2.3. 如果存在算法 A,可以在时间复杂度 $O(d\times h)$下得到 $hash_d(x_0x_1\ldots x_{d-1})$,那么称 $hash_d$ 是 $u_{d\times h}$ 上的一个快速哈希.称 $hash_d$ 是 $u_{d\times h}$ 上的一个快速连续弱哈希(FCWH),如果它同时是弱,连续和快速的.

2.2 顺序抽取公共子串的概念

一般串匹配问题涉及到两个串:模式串 X 和文本串 Y.如果模式串 X 在文本串 Y 的位置 pos 开始匹配,那么称 X 是 Y 从 pos 开始的子串,记为 $X=substr(Y,pos,n)$.其中,pos 是串 X 的起始位置,n 是串 X 的长度.

顺序抽取公共子串问题也涉及两个串:长为 $r\times h$ 位的参考串 REF 和长为 $c\times h$ 位的对比串 CMP.串 REF 或 CMP 的子串称为正则子串,如果子串的起始位置和长度都是 h 的整数倍.记一些特别的正则子串如下:

- $ref[pos_r,d]=substr(REF,pos_r\times h,d\times h)$,$cmp[pos_c,d]=substr(CMP,pos_c\times h,d\times h)$;
- $ref[pos_r]=substr(REF,pos_r\times h,h)$,$cmp[pos_c]=substr(CMP,pos_c\times h,h)$.

$ref[pos_r,d]$和 $cmp[pos_c,d]$称为基本正则子串.

如果 $substr(CMP,pos_c\times h,len\times h)$,$len\geq d$ 是 REF 的正则子串,而 $substr(CMP,pos_c\times h,(len+1)\times h)$不是 REF 的正则子串,那么称 $substr(CMP,pos_c\times h,len\times h)$是 REF 和 CMP 的一个当选公共子串.此时,向量(pos_c,len,pos_r)称为参考串 REF 和对比串 CMP 关于基本块长度 h 和重复次数 d 的一个当选指标;在该当选指标中,分别称 pos_c 为位置指标,len 为长度指标,pos_r 为参考位置指标.

定义 2.4. 参考串 REF 和对比串 CMP 关于基本块长度 h 和重复次数 d 的顺序抽取公共子串(SECS)问题,就是确定这样的当选指标集 $SS=\{(pos_{c,k},len_k,pos_{r,k})|0\leqslant k\leqslant l\}$,使得:(1) 对于 $0\leqslant k\leqslant l-1$,$pos_{c,k}+len_k\leqslant pos_{c,k+1}$;(2) 不存在小于 $pos_{c,0}$ 的位置指标;不存在大于 $pos_{c,1}+len_1$ 的位置指标;不存在介于 $pos_{c,k}+len_k$ 和 $pos_{c,k+1}$ 中间的位置指标.即不存在位置指标 pos,使得 $pos_{c,k}+len_k<pos<pos_{c,k+1}$.

本质上讲,SECS 问题的结果就是利用蛮力算法进行一般串匹配,寻找尽可能早出现的互不相交的所有当选指标.因此,SECS 问题是一般串匹配问题的推广.但是在一般串匹配问题中,模式串的长度和内容是预先确定的,是唯一的;而在 SECS 问题中,模式串的长度和内容不是预先确定的,也不一定唯一.

3 通过有限域构造 FCWH

3.1 FCWH的构造

对于质数幂 q,寻找 q 元有限域 $GF(q)$上的不可约多项式是有限域理论中的重要研究课题,已有多种构造算法[7],这里只讨论 $GF(2)=(u_1,+_1,\times_1)$.记 $GF(2)$上以 t 为不定元的 n 次多项式集合为 $P_n[t]$.

定义 3.1. 定义 u_n 和 $P_n[t]$之间的一一映射:

- 对于$\forall b_0b_1\ldots b_{n-1}\in u_n$, $G(b_0b_1..b_{n-1})=\sum_{j=0}^{n-1}b_jt^{n-j-1}$;
- 对于 $\forall\sum_{j=0}^{n-1}b_jt^{n-j-1}\in P_n[t]$, $G^{-1}\left(\sum_{j=0}^{n-1}b_jt^{n-j-1}\right)=b_0b_1..b_{n-1}$.

选定 $GF(2)$上一个以 t 为不定元的 h 次不可约多项式 $I_h(t)=\sum_{j=0}^{h}i_jt^{h-j}$,记 $G^{-1}\left(\sum_{j=0}^{h}i_jt^{h-j}\right)=i_0i_1..i_h$.

定义 3.2. 对于$\forall x_0,x_1,\ldots,x_{d-1}\in u_h$,定义哈希函数 $hash_{d,ff}(x_0x_1\ldots x_{d-1})=G^{-1}(G(x_0x_1\ldots x_{d-1}) \bmod I_h(t))$.

命题 3.1. $hash_{d,ff}$是 $u_{d\times h}$上的 FCWH.

证明:对于$\forall x_0,x_1,\ldots,x_{d-1},x_d\in u_h$,记 $x_i=b_{i\times h}b_{i\times h+1}\ldots b_{(i+1)\times h-1},0\leqslant i\leqslant d$.记 $x=x_0x_1\ldots x_{d-1},y=x\oplus_{d\times h}(i_0,i_1,\ldots,i_h0_{h+1}0_{h+2}\ldots 0_{d\times h})\neq x$,有 $hash_{d,ff}(x)=hash_{d,ff}(y)$.所以 $hash_{d,ff}$是弱的.下面证明 $hash_{d,ff}$是连续的.计算:

$$\begin{aligned}hash_{d,ff}(x_0x_1...x_{d-1}) &= G^{-1}(G(x_0x_1...x_{d-1}) \bmod I_h(t)) = G^{-1}((G(x_0)t^{(d-1)h}+G(x_1...x_{d-1})) \bmod I_h(t))\\ &= G^{-1}((G(x_0)t^{(d-1)h} \bmod I_h(t))+(G(x_1...x_{d-1}) \bmod I_h(t))) \quad (1)\\ &= G^{-1}(G(x_0)t^{(d-1)h} \bmod I_h(t))\oplus_h G^{-1}(G(x_1...x_{d-1}) \bmod I_h(t))\end{aligned}$$

$$\begin{aligned}hash_{d,ff}(x_1x_2...x_d) &= G^{-1}(G(x_1x_2...x_d) \bmod I_h(t)) = G^{-1}(G(x_1...x_{d-1})t^h+(G(x_d)) \bmod I_h(t))\\ &= G^{-1}((G(x_1...x_{d-1})t^h \bmod I_h(t))+(G(x_d) \bmod I_h(t))) \quad (2)\\ &= G^{-1}(G(x_1...x_{d-1})t^h \bmod I_h(t))\oplus_h G^{-1}(G(x_d)\bmod I_h(t)) = G^{-1}(G(x_1...x_{d-1})t^h \bmod I_h(t))\oplus_h x_d\end{aligned}$$

由以上公式(1)可知,$G^{-1}(G(x_1\ldots x_{d-1}) \bmod I_h(t))=G^{-1}(G(x_0)t^{(d-1)h} \bmod I_h(t))\oplus_h hash_{d,ff}(x_0x_1\ldots x_{d-1})$.其中,

$$G^{-1}(G(x_0)t^{(d-1)h} \bmod I_h(t)) = \bigoplus_{j=0}^{h-1}(b_j\circ G^{-1}(t^{dh-1-j} \bmod I_h(t)))$$

提前预处理 $G^{-1}(t^{dh-1-j} \bmod I_h(t))$,$0\leqslant j\leqslant h-1$.所以,$G^{-1}(G(x_1\ldots x_{d-1}) \bmod I_h(t))$可以由 $hash_{d,ff}(x_0x_1\ldots x_{d-1})$和 x_0在 $O(h)$的时间复杂度内得到.

而 $G^{-1}(G(x_1\ldots x_{d-1})t^h \bmod I_h(t))$可以由 $G^{-1}(G(x_1\ldots x_{d-1}) \bmod I_h(t))$在时间复杂度 $O(h)$得到,算法描述见左侧.所以由以上公式(2)可知,$hash_d(x_1x_2\ldots x_d)$可以由 $hash_{d,ff}(x_0x_1\ldots x_{d-1})$在 $O(h)$的时间复杂度下得到.

$hash_{d,ff}$是 $u_{d\times h}$上的一个快速哈希.因为对于 $x_0x_1\ldots x_{d-1}=b_0b_1\ldots b_{h-1}b_hb_{h+1}\ldots b_{2h-1}\ldots b_{dh-h}b_{dh-h+1}\ldots b_{dh-1}$,可以在时间复杂度 $O(d\times h)$内得到 $hash_{d,ff}(x_0x_1\ldots x_{d-1})$,算法描述见右侧.

0. //以下第 6 行输出 $G^{-1}(G(x_1\ldots x_{d-1})t^h \bmod I_h(t))$
1. $uint_{h+1}val=tail_1(G^{-1}(G(x_1\ldots x_{d-1}) \bmod I_h(t)))$;
2. for (j=0; j<h; j++) {
3. if ($sign(val)$==1)$val=val\oplus_{h+1}i_0i_1\ldots i_h$;
4. $val=val\langle\langle_{h+1}1$;
5. }
6. output ($curtail_h(val)$);

0. //以下第 6 行输出 $hash_{d,ff}(x_0x_1\ldots x_{d-1})$.
1. $uint_{h+1}val=b_0b_1\ldots b_{h-1}b_h$;
2. for (j=h+1; j<dh; j++) {
3. if ($sign(val)$==1)$val=val\oplus_{h+1}i_0i_1\ldots i_h$;
4. $val=concat(curtail_h(val),b_j)$;
5. }
6. output ($curtail_h(val)$); □

4 通过有限群构造 FCWH

u_h中的所有串赋予不同的运算,可以构成不同的2^h阶有限群,如 u_h中的串相对于$+_h$构成了一个循环群;对于$\oplus_h$构成了一个布尔群,即对于$\forall x\in u_h,x\oplus_h x=0_h,x\oplus_h 0_h=x$.有限群构造是代数和符号计算中的迷人领域[8−12].本文所关心的是 2^h阶有限群的快速实现.本节首先实现两类 2^h阶有限群:2^h阶阿贝尔群 G_a和一类特殊的一般线性群(GL)G_b.G_b利用 $GF(2)$上的矩阵构造得到.本节最后基于 G_a和 G_b构造 FCWH.

4.1 G_a的实现

命题 4.1 是有限阿贝尔群的结构定理[11,12].作为命题 4.1 显而易见的推论,命题 4.2 刻画了 G_a的结构.

命题 4.1. $n=p_1^{e_1}\ldots p_r^{e_r}$阶阿贝尔群是 Sylow 子群 $S(p_1),\ldots,S(p_r)$的直积,这里:质数 $p_1,\ldots,p_r$两两互质,并且 $S(p_i)$的阶数为 $p_i^{e_i}$;而 $S(p_i)$是阶数分别为 $p_i^{e_{i_1}},\ldots,p_i^{e_{i_s}}$ 的循环群的直积,这里,$e_{i_1}+\ldots+e_{i_s}=e_i$.

命题 4.2. 相对于 h 的整数分解 $h=h_1+\ldots+h_s$,2^h阶阿贝尔群 G_a是阶数分别为 $2^{h_1},\ldots,2^{h_s}$ 的循环群的直积.

因为 u_{h_i} 中的串对于 $+_{h_i}$ 构成了一个 2^{h_i} 阶的循环群,记为 $C_{h_i}=(u_{h_i},+_{h_i})$.这样,相对于整数分解 $h=h_1+\ldots+h_s$ 的 G_a 可以表示为 $G_{a,(h_1,\ldots,h_s)}=(u_h,+_{(h_1,\ldots,h_s)})=\mathop{\times}_{i=1}^{s}C_{h_i}$.因此,群 $G_{a,(h_1,\ldots,h_s)}$ 的群加法运算和逆元运算相对于机器指令$+_h$,$-_h$的时间复杂度为 $O(s)=O(h)$.

4.2 G_b的实现

定义 4.1. 整数 h 的 s 分解指的是一个非负整数向量 $p=(h_1,\ldots,h_s,h_{s+1})$,只要它满足:(1) $h=h_1+\ldots+h_s+h_{s+1}$;(2) $h_1>0$;(3) 对于$\forall 2\leqslant i\leqslant s+1, 0\leqslant h_i\leqslant s+2-i$.对于整数 h 的 s 分解 $p=(h_1,\ldots,h_s,h_{s+1})$,定义 $GF(2)$上的矩阵类 M_p,M_p 由所有满足以下条件的$(s+2)\times(s+2)$矩阵组成:主对角线上的元素为 1,主对角线和显著位置之外的任何位置的元素为 0.这里的显著位置指的是第 i 行的最后 h_i 个位置,$1\leqslant i\leqslant s+1$.由整数 h 的 s 分解 $p=(h_1,\ldots,h_s,h_{s+1})$,建立了 u_h 和 M_p 之间的一一映射:对于$\forall x=b_0b_1\ldots b_{h-1}\in u_h$,将 $b_0b_1\ldots b_{h-1}$ 依次填入矩阵的 h 个显著位置得到 M_p 类型矩阵.也用 M_p, M_p^{-1} 分别表示 u_h 到矩阵类 M_p 的映射和逆映射.

由 M_p 类型矩阵所对应的行列式为 1 以及矩阵乘法的非交换性,可以证明以下命题:

命题 4.3. 对于整数 h 的 s 分解 p,矩阵类 M_p 对于 $GF(2)$上的矩阵乘法$\bullet$构成一个非阿贝尔群.记为$(M_p,\bullet)$.因此可以定义 u_h 上的非阿贝尔群 $G_{b,p}=(u_h,\bullet_p)$:对于$\forall x,y\in u_h$, $x\bullet_p y=M_p^{-1}(M_p(x)\bullet M_p(y))$.

取 h 两个具体的 s 分解: $s_c=(s+1,s,\underbrace{0,\ldots,0}_{s-1})$, $s_f=(s+1,\underbrace{1,\ldots,1}_{s})$,(此时,$h=2s+1$ 是奇数;当位串长度为偶数时,可在串首添加一个 0,命题 4.3 仍然成立).此时,对于$\forall x=x_0x_1\ldots x_{h-1}, y=y_0y_1\ldots y_{h-1}\in u_h$,有:

G_{b,s_c} 的群运算:记 $x\bullet_{s_c} y=z_0z_1\ldots z_{h-1}$,那么对于 $i=0,s+1,\ldots,h-1$,有 $z_i=x_i+_1y_i$;对于 $i=1,\ldots,s$,有 $z_i=x_i+_1y_i+_1(x_0\times_1 y_{s+i})$.设 x 在 G_{b,s_c} 中的逆元 $x^{-1}=v_0v_1\ldots v_{h-1}$.由 $x\bullet_{s_c} x^{-1}=0_h$ 解得:对于 $i=0,s+1,\ldots,h-1$,有 $v_i=x_i$;对于 $i=1,\ldots,s$,有 $z_i=x_i+_1(x_0\times_1 y_{s+i})$. $x\bullet_{s_c} y$,x^{-1} 可由指令$\oplus_h$在时间复杂度 $O(1)=O(h)$实现.

G_{b,s_f} 的群运算:记 $x\bullet_{s_f} y=z_0z_1\ldots z_{h-1}$,那么对于 $i=s$,有 $z_i=x_i+_1y_i+_1\sum_{j=0}^{s-1}(x_j\times_1 y_{s+j+1})$;对于 $i\neq s$,有 $z_i=x_i+_1y_i$.设 x 在 G_{b,s_f} 中的逆元 $x^{-1}=v_0v_1\ldots v_{h-1}$.由 $x\bullet_{s_f} x^{-1}=0_h$ 解得:对于 $i\neq s$,有 $v_i=x_i$;对于 $i=s$,有 $v_i=x_i+_1\sum_{j=0}^{s-1}(x_j\times_1 y_{s+j+1})$.$x\bullet_{s_f} y$, x^{-1} 可指令$\oplus_h,\wedge_s$借助奇偶标志位 PF[13]在时间复杂度 $O(h)$内实现.

4.3 通过有限群构造FCWH

定义 4.2. 用 $G=(u_h,\otimes)$表示以上所构造的 $G_{a,(h_1,\ldots,h_s)}$,G_{b,s_c} 或 G_{b,s_f} 等 2^h 阶有限群,对于$\forall x_0,x_1,\ldots,x_{d-1}\in u_h$,定义哈希函数 $hash_{d,fg}(x_0x_1\ldots x_{d-1})=\bigotimes_{j=0}^{d-1}x_j=x_0\otimes x_1\otimes\ldots\otimes x_{d-1}$.

命题 4.4. $hash_{d,fg}$ 是 $u_{d\times h}$ 上的 FCWH.

证明:对于$\forall x_0,x_1,\ldots,x_{d-1},x_d\in u_h$,对于$\forall x\neq x_0$,让 $z=x^{-1}\otimes x_0\otimes x_1$ 则

$hash_{d,fg}(x_0x_1x_2\ldots x_{d-1})=hash_{d,fg}(xzx_2\ldots x_{d-1})$

$hash_{d,fg}(x_1\ldots x_{d-1}x_d)=x_1\otimes\ldots\otimes x_{d-1}\otimes x_d=x_0^{-1}\otimes(.x_0\otimes.x_1\otimes\ldots\otimes x_{d-1})\otimes x_d=x_0^{-1}\otimes.hash_{d,fg}(x_0x_1\ldots x_{d-1})\otimes x_d$

$hash_{d,fg}(x_0x_1\ldots x_{d-1})=x_0\otimes x_1\otimes\ldots\otimes x_{d-1}$

相应的运算均可在要求的时间复杂度内完成. □

5 通过有限环构造 FCWH

u_h 对于运算$+_h,\times_h$构成一个有限环,记为 $R_h=(u_h,+_h,\times_h)$.对于$\forall x\in u_h,\forall x_0,x_1,\ldots,x_{d-1},x_d\in u_h$,考察以下算法过程:

```
0. //以下第 9 行输出 hash_{d,fr}(x_0x_1…x_{d-1})
1. uint_h S_1,S_2,…,S_{k-1},S_k,i,j;
2. S_1=S_2=…=S_{k-1}=S_k=0;
3. for (j=0; j<d; ++j) {
4.     S_1=S_1+_h x_j;
5.     for (i=2; i<k+1; ++i) {
6.         S_i=S_i+_h S_{i-1};
```

7. }
8. }
9. output (S_k);
10. for (i=1; i<k+1; ++i) {
11. $S_i = S_i -_h \binom{d+i-2}{i-1}_h \times_h x_0$;
12. }
13. $S_1=S_1+_h x_d$;
14. for (i=2; i<k+1; ++i) {
15. $S_i=S_i+_h S_{i-1}$;
16. }
17. output (S_k);
18. //以上第 17 行输出 $hash_{d,fr}(x_1\ldots x_{d-1}x_d)$

算法第 11 行的组合数定义为 $\binom{i}{j}_h = \frac{i!}{j!(i-j)!} \bmod 2^h$,有组合数递推关系 $\binom{i}{j}_h = \binom{i-1}{j}_h +_h \binom{i-1}{j-1}_h$ 成立.将以上算法第 9 行的输出定义为 $x_0x_1\ldots x_{d-1}$ 的哈希 $hash_{d,fr}(x_0x_1\ldots x_{d-1})$.当算法运行到第 9 行时,对于$\forall 1\leqslant i\leqslant k$,$S_i$ 可以表示为 $x_0,x_1,\ldots,x_{d-1}$ 在环 $R_h=(u_h,+_h,\times_h)$上的一个线性组合,即$\exists c_{i,j},0\leqslant j\leqslant d-1$,使得 $S_i=c_{i,0}\times_h x_0+_h c_{i,1}\times_h x_1+_h\ldots+_h c_{i,d-1}\times_h x_{d-1}$.根据组合数的递推关系,可证 $c_{i,j}=\binom{d+i-j-2}{i-1}_h$, $1\leqslant i\leqslant k$, $0\leqslant j\leqslant d-1$.

同理可知,以上算法第 17 行输出 $hash_{d,fr}(x_1\ldots x_{d-1}x_d)$.至此证明了 $hash_{d,fr}$ 是 $u_{d\times h}$ 上的快速哈希和连续哈希.

至于 $hash_{d,fr}$ 是 $u_{d\times h}$ 上的弱哈希,这是因为 $c_{k,d-1}=\binom{k-1}{k-1}_h=1$ 和 $c_{k,d-2}=\binom{k}{k-1}_h=k$,由此对于$\forall x\neq x_{d-2}\in u_h$, $\exists y\in u_h$, 使得 $c_{k,d-2}\times_h x_{d-2}+_h c_{k,d-1}\times_h x_{d-1}=c_{k,d-2}\times_h x+_h c_{k,d-1}\times_h y$,$hash_{d,fr}(x_0\ldots x_{d-3}.x_{d-2}x_{d-1})=hash_{d,fr}(x_0\ldots x_{d-3}.xy)$.$k$=2 时,$hash_{d,fr}$ 就是 Adler 校验和.Adler 校验和曾经为 IETF 的下一代传输层协议 SCTP(流控传输协议)所采纳[14,15].

基于有限环 $R_h=(u_h,+_h,\times_h)$,还可以构造 $u_{d\times h}$ 上的其他 FCWH.例如,利用秦九韶-Horner 算法可以证明 $p^{d-1}\times_h x_0+_h p^{d-2}\times_h x_1+_h\ldots+_h p\times_h x_{d-2}+_h x_{d-1}$ 是 $x_0x_1\ldots x_{d-1}$ 的一个 FCWH.这里,p 是一个小于 2^h 的质数.一般串匹配问题的 Karp-Rabin 算法就采取了该 FCWH.

6 FCWH 的冲突概率

以上从多个途径构造了 FCWH,同时显示了 FCWH 普遍存在于计算机科学理论和工程实践中.对于上文中所构造的 $u_{d\times h}$ 上具体的 FCWH:$hash_{d,ff}$,$hash_{d,fg}$,$hash_{d,fr}$,笼统地记为 $hash_d$.下面分析 $hash_d$ 所产生的冲突概率.首先,针对 $hash_{d,ff}$ 推广 Rabin 的一些相应工作.

6.1 对Rabin相应工作的推广

对于质数幂 q,q 元有限域 $GF(q)$上的 h 次不可约多项式的数目记为 $N_q(h)$.命题 6.1[16,17]是有限域论中的不可约多项式的计数定理.这里用它来证明:对于任意正整数 h,都有 $N_q(h)>\frac{1}{h}\left(q^{\frac{h}{2}}-1\right)^2$ (见命题 6.2、命题 6.3).

命题 6.1. 那么, $N_q(h)=\frac{1}{h}\sum_{x\cdot y=h}\mu(x)q^y$.这里,$\mu(x)$是默比乌斯函数,即 x 为 1 时,$\mu(x)$=1;x 为 k 个互不相同的质数之积时,$\mu(x)=(-1)^k$;在其他的情况下,$\mu(x)$=0.

命题 6.2. 如果 h 有多于一个的质因子,那么 $N_q(h)>\frac{1}{h}\left(q^{\frac{h}{2}}-1\right)^2$.

证明:设 $h=p_1^{e_1}p_2^{e_2}...p_k^{e_k}$,这里,质数 $p_1<p_2<\ldots<p_k$.显然,$p_1\geqslant 2$,$p_2\geqslant 3$,$h\geqslant 6$.由命题 6.1 计算

$$\sum_{x\cdot y=h}\mu(x)q^y=q^h-\sum_{1\leqslant i_1\leqslant k}q^{\frac{h}{p_{i_1}}}+\sum_{1\leqslant i_1<i_2\leqslant k}q^{\frac{h}{p_{i_1}p_{i_2}}}+\ldots+(-1)^j\sum_{1\leqslant i_1<i_2<\ldots<i_j\leqslant k}q^{\frac{h}{p_{i_1}p_{i_2}\cdots p_{i_j}}}+\ldots+(-1)^kq^{\frac{h}{p_1p_2\cdots p_k}}>$$

$$q^h-q^{\frac{h}{p_{i_1}}}-\left\{q^{\frac{h}{p_{i_2}}}+q^{\left(\frac{h}{p_{i_2}}-1\right)}+\ldots+q+1\right\}=q^h-q^{\frac{h}{p_{i_1}}}-\frac{q^{\left(\frac{h}{p_{i_2}}+1\right)}-1}{q-1}\geqslant q^h-q^{\frac{h}{p_{i_1}}}-(q^{\left(\frac{h}{p_{i_2}}+1\right)}-1)\geqslant$$

$$q^h-q^{\frac{h}{2}}-q^{\left(\frac{h}{3}+1\right)}+1\geqslant q^h-q^{\frac{h}{2}}-q^{\frac{h}{2}}+1=\left(q^{\frac{h}{2}}-1\right)^2\quad\left(\text{因为}h\geqslant 6,\ \frac{h}{2}\geqslant\frac{h}{3}+1\right).$$ □

命题 6.3. 如果 h 只有一个的质因子 p,那么 $N_q(h)=\frac{1}{h}\left(q^h-q^{\frac{h}{p}}\right)>\frac{1}{h}\left(q^{\frac{h}{2}}-1\right)^2$.

证明:由命题 6.1 计算,$\sum_{x\cdot y=h}\mu(x)q^y=\left(q^h-q^{\frac{h}{p}}\right)\geqslant\left(q^h-q^{\frac{h}{2}}\right)>\left(q^h-2q^{\frac{h}{2}}+1\right)=(q^h-1)^2$. □

当 $q=2$,$h=p$ 时,命题 6.3 是 Rabin[2]的一个引理.下面是 $hash_{d,ff}$ 的冲突概率的一个上界.

命题 6.4. 在 $u_{d\times h}$ 中随机放回抽样 n 个串 $x_1,x_2,\ldots,x_n$,那么$\exists 1\leqslant i<j\leqslant n$,使得 $hash_{d,ff}(x_i)=hash_{d,ff}(x_j)$的概率

$$Pr_{d,ff}<\frac{n^2dh}{2\left(2^{\frac{h}{2}}-1\right)^2}.$$

特别地,当 $h\geqslant 4$ 时有 $Pr_{d,ff}<\frac{n^2dh}{2^h}$.

证明:记多项式 $\pi(t)=\prod_{1\leqslant i<j\leqslant n}(G(x_i)-G(x_j))$,则$\pi(t)$的度数满足不等式:

$$\partial\pi(t)=\sum_{1\leqslant i<j\leqslant n}\partial(G(x_i)-G(x_j))\leqslant\sum_{1\leqslant i<j\leqslant n}\max(\partial G(x_i),\partial G(x_j))\leqslant\sum_{1\leqslant i<j\leqslant n}(dh-1)=\binom{n}{2}(dh-1).$$

将$\pi(t)$的 h 次不可约因式 $f(t)$的重复因子定义为自然数 m,使得 $f^m(t)$是$\pi(t)$的因式,但 $f^{m+1}(t)$不是$\pi(t)$的因式.记$\pi(t)$所有 h 次不可约因式的重复因子之和为 M,则有 $M\leqslant\frac{1}{h}\binom{n}{2}(dh-1)$.由命题 6.2、命题 6.3,记

$$N_2(h)=N>\frac{1}{h}\left(2^{\frac{h}{2}}-1\right)^2.$$

re 表示随机事件:放回抽样一个 h 次不可约多项式,*RE* 是 *re* 重复 M 次并且抽中已选定的 $I_h(t)$至少一次的随机事件,RE_i,$1\leqslant i\leqslant M$ 是在第 i 次抽样中首次抽中 $I_h(t)$的随机事件.随机事件 RE_i,$1\leqslant i\leqslant M$ 两两互斥,$hash_{d,ff}$的冲突概率就是 *RE* 的概率.所以,

$$Pr_{d,ff}=Pr\{RE\}=Pr\left\{\bigcup_{i=1}^{M}RE_i\right\}=\sum_{i=1}^{M}Pr\{RE_i\}=\sum_{i=1}^{M}\left(\frac{N-1}{N}\right)^{i-1}\frac{1}{N}<\sum_{i=1}^{M}\frac{1}{N}=\frac{M}{N}\leqslant$$

$$\frac{\frac{1}{h}\binom{n}{2}(dh-1)}{\frac{1}{h}\left(2^{\frac{h}{2}}-1\right)^2}=\frac{\binom{n}{2}(dh-1)}{\left(2^{\frac{h}{2}}-1\right)^2}<\frac{n^2(dh-1)}{2\left(2^{\frac{h}{2}}-1\right)^2}.$$

当 $h\geqslant 4$ 时, $h\geqslant \frac{h}{2}+2$, $2^h\geqslant 2^{\frac{h}{2}+2}=4\times 2^{\frac{h}{2}}$.所以, $2\left(2^{\frac{h}{2}}-1\right)^2=2^h+\left(2^{\frac{h}{2}}-4\right)\cdot 2^{\frac{h}{2}}+2>2^h$, $Pr_{d,ff}<\frac{n^2dh}{2^h}$. □

以上讨论 h 为任意正整数时 $hash_{d,ff}$ 的冲突概率.Rabin[2]只讨论了 h 为质数的情形,工程实践经常引用该结果.下面将从另外的角度给出该冲突概率的一个更优的上界(见命题 6.7).

6.2 $hash_d$的哈希冲突概率

命题 6.5. 对于$\forall v\in u_h$,$|\{x\in u_{d\times h}|hash_d(x)=v\}|=2^{(d-1)\times h}$;因此,$|\{hash_d(x)|\forall x\in u_{d\times h}\}|=2^h$.

证明:可证

$$\{x\in u_{d\times h}|hash_{d,ff}(x)=v\}=\{G^{-1}(G(y)\times I_h(t)+G(v))|y\in u_{(d-1)\times h}\};$$

$$\left\{\forall x_0,x_1,...,x_{d-1}\in u_h\mid hash_{d,fg}(x_0x_1...x_{d-1})=\overset{d-1}{\underset{j=0}{\otimes}}x_j=v\right\}=\left\{\forall x_0,x_1,...,x_{d-2}\in u_h,x_{d-1}=\left(\overset{d-2}{\underset{j=0}{\otimes}}x_j\right)^{-1}\otimes v\in u_h\right\};$$

$$\{\forall x_0,x_1,...,x_{d-1}\in u_h\mid hash_{d,fr}(x_0x_1...x_{d-1})=v\}=\left\{\forall x_0,x_1,...,x_{d-2}\in u_h,x_{d-1}=v-_h\sum_{j=0}^{d-2}\left(\binom{d+k-2-j}{k-1}\times_h x_j\right)\in u_h\right\}.$$ □

在命题 6.5 中,$hash_d$ 的值在 $u_{d\times h}$ 中的分布是均匀的.称这种取值均匀分布的 FCWH 为匀值 FCWH (homogeneous FCWH,简称 FCWH^2).根据命题 6.5,简单计算有:

命题 6.6. 在 $u_{d\times h}$ 中随机放回抽样两个串 x_1,x_2,事件 $x_1=x_2$ 的概率为 $2^{-(d\times h)}$;事件 $hash_d(x_1)=hash_d(x_2)$的概率为 2^{-h}(与 d 无关);事件 $hash_d(x_1)=hash_d(x_2)$,$x_1\neq x_2$ 的概率为 $2^{-h}-2^{-(d\times h)}$(与 d 有关).

基于以上讨论,下面给出 $hash_d$(匀值 FCWH)哈希冲突概率的一个公共上界.

命题 6.7. $u_{d\times h}$ 中放回抽样 n 个串 $x_1,\ldots,x_n$,那么$\exists 1\leqslant i<j\leqslant n$ 使得 $hash_d(x_i)=hash_d(x_j)$的概率 $Pr_d<\frac{n^2}{2^h}$.

证明:$n>2^h$ 时,命题自然成立.以下只考虑 $n\leqslant 2^h$ 的情形.随机事件 E_1 表示在 $u_{d\times h}$ 中放回抽样到任意串 x_1, $Pr\{E_1\}=1$.对于$\forall 2\leqslant i\leqslant n$,事件 E_i 表示在 $u_{d\times h}$ 中放回抽样到任意串 x_i 满足 $hash_d(x_i)\neq hash_d(x_j)$,$1\leqslant j<i$.由命题 6.5 知, $Pr\{E_i\mid E_1E_2...E_{i-1}\}=\frac{2^h-i+1}{2^h}$.由 Bayes 公式有

$$Pr\{E_1E_2...E_n\}=Pr\{E_1\}\prod_{i=2}^{n}Pr\{E_i\mid E_1E_2...E_{i-1}\}=\prod_{i=2}^{n}\frac{2^h-i+1}{2^h}>\prod_{i=2}^{n}\frac{2^h-n}{2^h}=\left(\frac{2^h-n}{2^h}\right)^{n-1}>\left(\frac{2^h-n}{2^h}\right)^n.$$

所以, $Pr_d=1-Pr\{E_1E_2...E_n\}<1-\left(\frac{2^h-n}{2^h}\right)^n=\frac{2^{h^n}-(2^h-n)^n}{2^{h^n}}$.

而 $2^{h^n}-(2^h-n)^n=(2^h-2^h+n)\sum_{i=0}^{n-1}2^{h^i}(2^h-n)^{n-1-i}<n\sum_{i=0}^{n-1}2^{h^i}(2^h)^{n-1-i}=n\sum_{i=0}^{n-1}(2^h)^{n-1}=n^2(2^h)^{n-1}$.命题成立. □

6.3 加强$hash_d$指纹清晰度的方法

从现在开始约定 d 是 2 的幂,即$\exists k\geqslant 1$ 使得 $d=2^k$;对于分块的细化程度 i,$\forall 0\leqslant i\leqslant k$,由第 2 节和第 6 节平行定义 $cmp[pos_c,d/2^i]$,$ref[pos_r,d/2^i]$和 $hash_{d/2^i}$.记

$$hcmp_i(pos_c)=hash_{d/2^i}(cmp[pos_c,d/2^i]),\ href_i(pos_r)=hash_{d/2^i}(ref[pos_r,d/2^i]).$$

定义基本正则子串的哈希向量分别为如下的 2^i 维向量:

$$vcmp_i[pos_c]=(hcmp_i(pos_c),hcmp_i(pos_c+d/2^i),...,hcmp_i(pos_c+(2^i-1)d/2^i)),$$

$$vref_i[pos_r]=(href_i(pos_r),href_i(pos_r+d/2^i),...,href_i(pos_r+(2^i-1)d/2^i)).$$

要判断 $cmp[pos_c,d]=ref[pos_r,d]$,可先判断 $vcmp_i[pos_c]=vref_i[pos_r]$.

如果在 $u_{d\times h}$ 中放回抽样得到 $cmp[pos_c,d]$,$ref[pos_r,d]$,那么由命题 6.6 可知,随机事件 $vcmp_i[pos_c]=vref_i[pos_r]$ 的概率为 $(2^{-h})^{(2^i)}=2^{-2^ih}$.因此,误判事件($cmp[pos_c,d]\neq ref[pos_r,d]$和 $vcmp_i[pos_c]=vref_i[pos_r]$)的概率小于 2^{-2^ih}.因此,尽管 $hash_d$ 是弱哈希,但是对于$\forall\varepsilon>0$,通过增大 i 可以使误判的概率小于ε.显然,$i=k$ 时,误判是不可能事件.

一般情况下,将长度为 $d\times h$ 位的块 x 分割成 2^i 个长度为 $d/2^i\times h$ 位前后相继的子块 $x_0,x_1,...,x_{2^i-1}$.相应子块的 $hash_{d/2^i}$ 所成向量记为 $vhash_i(x)=(hash_{d/2^i}(x_0),hash_{d/2^i}(x_1),...,hash_{d/2^i}(x_{2^i-1}))$.重复命题 6.7 的证明过程有:$u_{d\times h}$ 中放回抽样 n 个串 $x_1,\ldots,x_n$,那么$\exists 1\leqslant i_1<i_2\leqslant n$,使得 $vhash_i(x_{i_1})=vhash_i(x_{i_2})$ 的概率

$$Pr_{(d,2^i)}<\frac{n^2}{2^{i}h}.$$

对于$\forall\varepsilon>0$, $Pr_{(d,2^i)}<\dfrac{n^2}{2^{i}h}<\varepsilon$ 等价于 $2^{i}h<\dfrac{n^2}{\varepsilon}$, $i<\log_2(2\log_2^n-\log_2^\varepsilon)-\log_2^h$,即 $i<\log_2(2\log_2^n-\log_2^\varepsilon)-\log_2^h$.记

$$i(n,h,\varepsilon)=\left\lceil \log_2(2\log_2^n-\log_2^\varepsilon)-\log_2^h\right\rceil.$$

当 h=8 时,取在 $u_{d\times h}$ 中放回抽样的次数 $n=2^s$,s=0,1,2,…;冲突概率上限$\varepsilon=2^{-t}$,t=0,1,2,…,有

$$i(n,h,\varepsilon)=\lceil \log_2(2s+t)-3\rceil.$$

当 $15\leqslant s\leqslant 40$,$35\leqslant t\leqslant 40$ 时,恒有$\lceil \log_2(2s+t)-3\rceil=4$.实际应用中,$n$ 为在一个文件中抽取定长字节流的次数.因此,n 小于文件大小.可将文件大小的上限设置为 1T(2^{40})字节,$n\leqslant 2^{40}$.因此,在冲突概率上限 $2^{-35}\approx 2.9104e^{-011}$ 的情况下,可将分块细化程度 i 的初始值取为 4.

6.4 $hash_d$哈希冲突的实验数据的分析

在作者维护的开源项目 X-Sync[18]中,用 Google 随机搜索到一篇关于西方古典绘画的文献目录[19](大小为 43 008 字节的 Microsoft Word 文档),对它进行了实验.其中,所采取的 $hash_d$ 是由 u_h 中的串对$\oplus_h$ 构成了一个布尔群所产生.h 取为 8,d 取为 1 024;细化程度取遍所有可能情形:$0\leqslant i\leqslant 10$.实验数据证实了以上的理论分析.

借助 FCWH 可以确定该文档中只有 l_r=35973 个 1K 字节长度的互异子串.在细化程度 i 下,对 l_r 个哈希向量统计如下:重复出现正好 x 次的哈希向量的总数在不重复计数,重复计数情况下分别记为 $root(i,x)$,$leaf(i,x)=x\times root(i,x)$.记 $discr(i)=leaf(i,1)/l_r$.$discr(i)$在一定程度上反映了哈希指纹的清晰程度.满足 $root(i,x)>0$ 的 x 的个数记为 $stem(i)$,满足 $root(i,x)>0$ 的 x 的最大值记为 $M(i)$.$stem(i)$和 $M(i)$在和一定程度上反映了哈希指纹的模糊程度.将 $discr(i)$,$stem(i)$,$M(i)$分别表示在图 1 中.

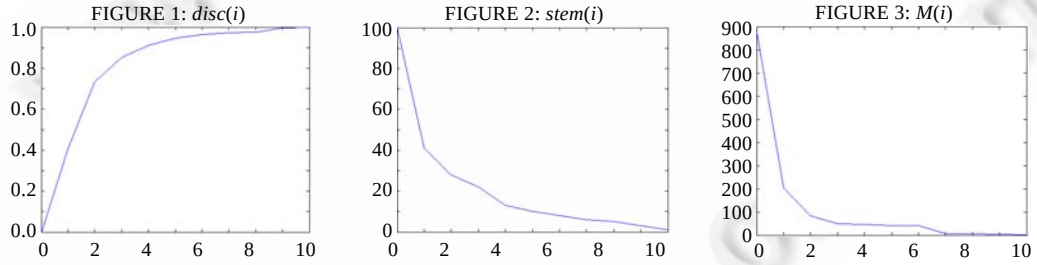

Fig.1

图 1

可见,随着 i 的增加,$discr(i)$呈指数增加,而 $stem(i)$和 $M(i)$呈指数减少.i=4 时,$discr(4)\approx 0.9103$,$stem(4)=13$,相应地,$(x,root(4,x))$分别为(1,32748),(2,840),(3,254),(4,132),(5,5),(6,5),(7,3),(8,8),(14,1),(16,1),(18,1),(23,1),(45,1).i=7 时,$discr(7)\approx 0.9726$,$stem(7)=6$,相应地,$(x,root(7,x))$分别为(1,34990),(2,345),(3,26),(4,51),(5,1),(7,1);即,i=7 时,l_r=35973 哈希值向量中,正好出现了 1 次的有 34 990 个,2 次的有 345 个,3 次的有 26 个,4 次的有 51 个,5 次和 7 次的各 1 个.

7 用 FCWH 解决 SECS

沿用 Karp-Rabin 算法的框架,利用 FCWH 来解决 SECS 分为两步:第 1 步,处理参考串 *REF*,整理参考串 *REF* 的所有基本正则子串的哈希向量;第 2 步,处理对比串 *CMP*,由前向后扫描对比串 *CMP*,执行选择-计算-比较-平

移方案.即:选择是在 Y 中选择与 X 等长度的子串 S;计算是计算子串 S 的哈希向量;比较是比较 X 的哈希向量和 S 的哈希向量;平移是在文本串 Y 中向前平移寻找下一个需要比较的子串的起始位置.

7.1 处理参考串*REF*:整理连续弱哈希

为了利用 *REF* 的所有基本正则子串的哈希向量 $vhash_i(x)$,计算 *REF* 的所有 $ref[pos_r,d/2^i]$相应的 $href_i(pos_r)$,共$(r-d/2^i+1)=L_r$ 个;这些哈希值存放在长 $L_r\times h$ 的位串 rh 中.用 $rh[j]$表示 $substr(rh,j\times h,h)$,$0\leqslant j<L_r$.$rh[j]$中存放的是 $href_i(j)$.为了加速在 rh 中的查找 $vcmp_i[pos_c]$,对 $rh[j]$,$0\leqslant j<L_r$ 的下标进行排序,排序结果存放在一个长 $L_r\times H_r$ 的位串 sh.这里,$H_r=\left\lceil \log_2^{L_r} \right\rceil$.用 $sh[j]$表示 $substr(sh,j\times H_r,H_r)$,$0\leqslant j<L_r$.$sh[j]$的初始化值为 j.根据字典顺序$(vref_i[sh[j]],sh[j])$对 $sh[j]$,$0\leqslant j<H_r$ 进行排序.一般的排序算法并不保证排序的稳定性[1].通过将初始下标值作为排序比较时最后的一个键值,保证了排序的稳定性.*REF* 中互不相同的基本正则子串的数目记为 l_r,l_r 与 i 的选择无关.对串 sh 作进一步的处理,删除其中非首次出现的基本正则子串的下标;这样,只有基本正则子串首次出现的下标被保存在 $sh[j]$,$0\leqslant j<l_r$ 中.另外,还需要一个长 $l_r\times H_r$ 的位串 th;用 $th[j]$表示 $substr(th,j\times H_r,H_r)$,$0\leqslant j<l_r$.$th[j]$,$0\leqslant j<l_r$ 是对 $vref_i[j]$出现次第所作的统计.$vref_i[j]$首次出现时对应的 $th[j]$为 0.

通过串 rh、串 sh 和串 th,保证了通过 FCWH 解决 SECS 所得到的结果与蛮力算法所得到的结果一致.由 th 可知 $vref_i$ 产生的指纹是否足够清晰.当指纹不够清晰时,可以提高分块细化程度 i(初始值 4).简单计算可知:串 rh,sh,th 的空间复杂度为 $O(2r\log(r)+rh)$;处理参考串 *REF* 总的时间复杂度为 $O(r\times h+r\log(r)+2r)$.

7.2 处理对比串*CMP*:选择-计算-比较-平移方案

计算对比串 *CMP* 在 pos_c 的哈希向量 $vcmp_i[pos_c]$之后,通过在 sh 中折半查找,确定是否存在参考串 *REF* 的下标 pos_r,使得 $vcmp_i[pos_c]=vref_i[pos_r]$.如果找不到,那么 pos_c 不可能成为当选指标的位置指标;如果找到,通过 th 确定 $vcmp_i[pos_c]$首次出现的位置和所有依次出现的位置(基本正则子串重复出现时,只考虑首次出现).依次比对确定是否存在 pos_r,使得 $cmp[pos_c,d]=ref[pos_r,d]$.如果不存在,$pos_c$ 不可能成为位置指标;否则,就找到了一个位置指标 pos_c.此时,根据 SECS 的定义还要沿着 *REF* 和 *CMP* 依次向后比对,以确定长度指标 len,这样就得到了一个当选指标(pos_r,pos_c,len).下标 pos_c 沿着范围 0~$(r-d)$从小到大:如果 pos_c 对应到一个当选指标(pos_r,pos_c,len),下标 pos_c 平移到 pos_c+len 开始新一轮循环;否则,下标 pos_c 平移到 pos_c+1 开始新一轮循环.循环结束,得到当选指标集 $SS=\{(pos_{c,k},len_k,pos_{r,k})|0\leqslant k\leqslant l\}$.

简单计算可知,本处理过程的时间复杂度为 $O(c\times(\log(r)+d+h))$.在比较 2^i 维向量 $vcmp_i[pos_r]$和 $vcmp_i[pos_c]$时,可以对向量的分量由后向前比较.这映射到程序实现上,每生成一个分量就查找相应的分量在已找到分量基础上是否存在.这样处理,加速了平移过程,特别当 FCWH 是通过阿贝尔群产生时尤其如此.

8 文件多版本快速同步协议 X-Sync

8.1 新的网络环境和计算模式下的文件同步场景

文件同步是常见的计算机应用.目前,UNIX 系统下的标准工具是 rsync[20].rsync 是 Tridgell 和 Mackerras[21,22] 于 1996 年开发的,其应用场景为:机器α(拥有文件 A)和机器β(拥有文件 B)通过低带宽通信链路连接,文件 A 和文件 B 很相似,现在要让机器β也拥有文件 A.rsync 的解决算法为:

1. 机器β将 B 分割成相继的不重叠的块,每一块长度都为 S 字节(最后一块可能小于 S 字节);
2. β为所有块计算两个校验和:32-位的 Adler 校验和 128-位的 MD4 校验和;
3. β将这些校验和发送给α;
4. α从文件 A 开始位置计算长度为 S 字节的块的 Adler 校验和;
   4.1. 在β发过来的 Adler 校验和中查找该 Adler 校验和;
   4.2.
      4.2.1. 如果找不到,就认为该块在文件 B 中不存在,于是将该块的第 1 个字节发送给β,接着从下一

字节位置计算 Adler 校验和;

4.2.2. 如果找到,就计算该块的 MD4 校验和,比对β发送过来的相应的 MD4 校验和;

4.2.2.1. 如果两个 MD4 校验和不一致,就执行步骤 4.2.1 所描述的操作;

4.2.2.2. 如果两个 MD4 校验和相同,那么就认为该块就是文件 *B* 中的相应块,发送命令让β在文件 *B* 中拷贝相应块;接着在 *S* 字节之后的位置计算 Adler 校验和.

重复步骤 4.1 和步骤 4.2,直至遍历完文件 *A*.

理论上看,rsync 算法存在着缺陷:首先,在步骤 4.2.2.2 中,两个块的 MD4 校验和相同,不能由此断定它们完全一样;其次,在步骤 4.2.1 和步骤 4.2.2.1 中找不到 Adler 或 MD4 校验和,不能由此判断该块在文件 *B* 中不存在.

从实践上看,近年来宽带网络的普及和云计算的兴起,在某些应用领域中对文件同步的需求发生了改变.Miller[23]描述了云计算下日常文档的处理模式.用户所面临的文件同步场景常常是这样的:文件编辑的客户端只拥有当前版本和为数不多的里程碑版本;而服务器端则是云计算的存储集群,拥有文件的所有里程碑版本和版本增量.客户端以一定的编辑周期向服务器端提交里程碑版本或版本增量.客户端可在任何时候向服务器端请求内容查询,实时检索任何历史版本的任何位置的内容.该应用场景反映的是文件多版本内容的实时备份好实时检索,rsync 算法不适用于该问题.下面基于 SECS 设计 X-Sync 同步协议解决以上问题.

8.2 X-Sync同步协议

X-Sync 同步协议是应用层协议,X-Sync 报文通过流传输(TCP 或 SCTP)交换.X-Sync 报文有命令报文和版本报文两种.前者有 NEW(创建命令报文,类型代码 0),DELETE(销毁命令报文,类型代码 1),OPEN(打开命令报文,类型代码 2),CLOSE(关闭命令报文,类型代码 3);后者有 BASELINE(基准版本报文,类型代码 4),DELTA(版本增量报文,类型代码 5),REQUEST(请求版本报文,类型代码 6),RESPOND(响应版本报文,类型代码 7).

里程碑版本在客户端称为基准版本,以强调对版本增量的参照作用.里程碑版本和版本增量通过版本的编辑时间区间来标记.编辑时间区间包括开始和终止两个时间戳(时间戳采用 UNIX 时间戳).当前版本编辑时间区间的终止时间戳加上 1s,就是下一个版本编辑时间区间的起始时间戳.

每一个文档的历史版本序列对应唯一的项目 ID,通过创建命令报文客户端和服务器端协商项目 ID.其他所有报文基于项目 ID 进行交互.X-Sync 同步协议交互的第 1 次会话是协商项目 ID.客户端发送项目 ID 为 0 的创建命令报文;服务器端以一个创建命令报文回复,内含一个未使用的项目 ID.最后一次会话是客户端发送销毁命令报文,服务器端以销毁命令报文应答,同时销毁该文档项目,原项目 ID 可以重新分配.从创建到销毁构成了文件备份检索的生命期.期间,客户端发送关闭命令报文和打开命令报文分别暂停和重启项目.

客户端发送的基准版本报文和版本增量报文在服务器端被原封不动地保存(包括报文首部).它们构成了版本实时备份和检索的实体内容.出于对文件内容实时检索的需要,在协议实现时,建议为所有编辑时间区间建立一个二叉查找树(时间戳索引树);为每一个历史版本的文件内容建立一个二叉查找树(文件内容位置索引树).

基准版本报文的数据部分是一个完整的里程碑版本.版本增量报文的数据部分是一系列版本增量数据块.客户端以基准版本为参考串,以当前版本为对比串,基于 SECS 生成版本增量报文.版本增量报文依据在当前版本中的先后顺序排列.版本增量数据块有两种:共同数据块和特有数据块.共同数据块包含两方面的内容:所含数据的长度和实际数据在基准版本中出现的起始位置.特有数据块也包含两方面的内容:所含数据的长度和实际数据.SECS 得到的当选指标集 *SS* 依位置指标依次排列,由当选指标集从当前版本依次产生所有特有数据块.

请求版本报文包含一个时间戳(请求时间戳),检索文件内容的起始位置和长度.如果时间戳对应不到任何文件版本,那么响应版本报文中的返回数据长度字段为 0.一次成功的文件版本的内容检索过程大致为:服务器端收到客户端发来的请求版本报文,沿着时间戳索引树确定要查找的文件版本所对应的文件内容位置索引树.然后,沿着文件内容位置索引树确定相应内容在里程碑版本或版本增量出现的起止位置(注意,一次内容检索可能涉及多个共同数据块和特有数据块),再通过响应版本报文将检索到的内容返回给给客户端.

根据以上客户端和服务器的交互,设计各报文的具体格式.命令报文只有报文首部,版本报文此外还有数据部分.所有报文首部的前 5 个八位组是一样的:第 1 个八位组的高 4 位是 X-Sync 协议版本号,低 4 位是报文类型

代码.第 2~第 5 这 4 个八位组是项目 ID.命令报文的首部只有这 5 个八位组,版本报文的首部还有 4 个八位组来用来表示报文数据的长度.不同类型的版本报文,其报文数据的格式也不一样.

基准版本报文的数据部分:第 1~第 4 这 4 个八位组是版本起始时间戳;第 5~第 8 这 4 个八位组是版本终止时间戳;第 9~第 12 这 4 个八位组是里程碑版本的文件长度;其后,该长度的数据是完整的里程碑文件.

版本增量报文的数据部分:第 1~第 4 这 4 个八位组是增量起始时间戳,第 5~第 8 这 4 个八位组是增量终止时间戳,第 9~第 12 这 4 个八位组是基准起始时间戳,第 13~第 16 这 4 个八位组是基准终止时间戳,第 17~第 20 这 4 个八位组是版本增量数据块的总长度,其后是该长度的数据是一系列的版本增量数据块.每个增量数据块的第 1 个八位组表明该增量数据块的类型,0 为共同数据块,1 为特有数据块.共同数据块的第 2~第 5 这 4 个八位组是该共同数据块在基准版本中的起始位置,第 6~第 9 这 4 个八位组是共同数据的长度.特有数据块的第 2~第 5 这 4 个八位组是该特有数据块中版本特有数据的长度,其后就是该长度的版本特有数据.

请求版本报文的数据部分:第 1~第 4 这 4 个八位组是请求时间戳,第 5~第 8 这 4 个八位组是检索文件内容的起始位置,第 9~第 12 这 4 个八位组是检索文件内容的和长度.

响应版本报文的数据部分:第 1~第 4 这 4 个八位组是返回数据长度,其后是该长度的数据.

作者维护的开源项目 X-Sync[18]提供了版本增量提取和版本恢复的演示程序和测试文档版本.

9 结束语

本文针对计算机科学理论和工程技术中广泛存在的快速连续弱哈希(FCWH)现象,抽取共性,提出概念;在涵盖已有现象的基础上,从代数结构角度统一规划了该类哈希的构造框架;从理论上分析了该类哈希冲突的共同上界,改进加强了现有理论成果;并应用它们设计出高效富有竞争性的算法来解决现实具体问题.这有助于人们加深对快速连续弱哈希现象的认识,从而设计出更多快速连续弱哈希函数来解决实际工程问题.

作为 FCWH 的理论应用,本文解决了一种特殊的串匹配问题,顺序抽取公共子串(SECS)问题.通过比对细化分块的 FCWH 指纹向量,提取参考串的所有定长互异子串,进而顺序抽取参考串和对比串的公共子串.这种借助 FCWH 来提取互异子串和公共子串的方法对众多串匹配应用有借鉴意义,特别是计算分子生物学的基因序列比对[24].这是因为当前的基因序列比对都是借助高性能并行计算机进行的,当参考串和对比串都很大时,对它们进行分块并行处理不仅可以提高单个计算节点上的处理速度,而且可以降低哈希冲突概率.

作为 FCWH 的工程应用,本文设计了快速同步协议 X-Sync 以解决宽带网络和云计算环境下的文档多历史版本内容的实时备份和实时检索.

鉴于 FCWH 现象在计算机科学理论和工程技术中如此广泛存在,希望本研究能够引起编码论和量子计算等领域研究者的注意.当今数字电子计算机的指令系统设计中,没有充分开发其代数性质.而群论和量子力学具有天然的血缘关系[25,26],如果能在指令层次上利用代数结构,由此产生的 FCWH 不仅计算高效,而且指纹清晰.

References:

[1] Knuth DE. The Art of Computer Programming. Vol.3, 2nd ed., Reading: Addison-Wesley, 1998. 144−148.

[2] Rabin MO. Fingerprinting by random polynomials. Technical Report, TR-15-81, Center for Research in Computing Technology, Harvard University, 1981.

[3] Karp RM, Rabin MO. Efficient randomized pattern-matching algorithms. IBM Journal of Research and Development, 1987,31:249−260. [doi: 10.1147/rd.312.0249]

[4] Cormen TH, Leiserson CE, Rivest RL, Stein C. Introduction to Algorithms. 2nd ed., MIT Press, 2001. 911.

[5] Crochemore M, Rytter W. Jewels of Stringology. World Scientific Publishing Co., 2002, 271, 6−7.

[6] Patterson DA, Hennessy JL. Computer Architecture: A Quantitative Approach. 4th ed., San Francisco: Morgan Kaufmann Publishers, 2007.

[7] Shparlinski IE. Finite Fields: Theory and Computation. Kluwer Academic Publishers, 1999. 45−52.

[8] Besche HU, Eick B. Construction of finite groups. Journal of Symbolic Computation, 1999,27(4):387−404. [doi: 10.1006/jsco.1998.0258]

[9] Besche HU, Eick B, O'Brien EA. A millennium project: Constructing small groups. Int'l Journal of Algebra and Computation, 2002,12(6):623−644.

[10] Seress A. An introduction to computational group theory. Notices of the AMS, 1997,44(6):671−679.

[11] Hall M Jr. The Theory of Groups. 2nd ed., New York: The Macmillan Company, 1976.36.

[12] Zhang YD, Construction of Finite Groups. Beijing: Science Press, 1984 (in Chinese).

[13] AMD Publication No. 24592. AMD64 Architecture Programmer's Manual. Vol. 1, Revision 3.14, 2007. 34.

[14] RFC 2960. Stream control transmission protocol. 2000.

[15] RFC 3309. Stream control transmission protocol (SCTP) checksum change. 2002.

[16] Roman S, Field theory, GTM 158. 2nd ed., Berlin: Springer-Verlag, 1995. 236.

[17] Lidl R, Niederreiter H. Finite Fields. 2nd ed., Cambridge University Press, 1997. 91.

[18] Homepage of X-sync. http://code.google.com/p/x-sync/

[19] http://www.projekte.kunstgeschichte.uni-muenchen.de/dt_frz_malerei/41-dt-franz-malerei/studieneinheiten/doc/bibliographie.doc

[20] Homepage of rsync. http://rsync.samba.org/

[21] Tridgell A, Mackerras P. The rsync algorithm. Technical Report, TR-CS-96-05, Australian National University, 1996.

[22] Tridgell A. Efficient algorithms for sorting and synchronization [Ph.D. Thesis]. Australian National University, 1999. 49.

[23] Miller M. Cloud Computing. Que. 2008.

[24] Pevzner PA. Computational Molecular Biology, An Algorithmic Approach. MIT Press, 2000.

[25] Weyl H. The Theory of Groups and Quantum Mechanics. Dover, 1931.

[26] Tinkham M. Group Theory and Quantum Mechanics. Dover, 2003.

附中文参考文献:

[12] 张远达.有限群构造.现代数学基础丛书.北京:科学出版社,1984.

徐泽明(1973－),男,湖北黄石人,博士,主要研究领域为计算机体系结构.

侯紫峰(1955－),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为计算机体系结构,数字化技术,计算机软件工程,计算机辅助设计,人工智能,数字信号处理,计算机图形学,图像处理.