算法分析与设计

字符串匹配

主要内容

- Brute Force方法
- Rabin-Karp算法
- Knuth-Morris-Pratt算法

字符串匹配问题

- T[1..n]是一个长度为n的数组
- P[1..m]是一个长度为m的数组，m≤n
- P和T中的元素都是属于有限字母表Σ的字符
- 字符数组T和P通常称为字符串
- 字符串匹配问题可以理解为在T中寻找所有与P匹配的位置

Brute Force方法

- 思想
  - 用一个循环找出所有匹配之处（有效位移），该循环对n-m+1个可能的每一个s值进行检查，看P[1..m]=T[s+1..s+m]是否满足

Brute Force方法

NAIVE-STRING-MATCHER(T, P)

```
n ← length[T]
m ← length[P]
for s ← 0 to n - m
    do if P[1 .. m] = T[s + 1 .. s + m]
          then print "Pattern occurs with shift" s
```

Brute Force方法

- 时间复杂度分析
  - 在最坏情况下，运行时间是$\Theta((n-m+1)m)$
- 实例
  - 文本T=0000100001010001
  - 模式P=0001

Rabin-Karp算法

- 基本思想
  - 将字符串对应于数值，字符串匹配问题即可归结为数值是否相等的问题
  - 字符串如何对应于数值?

Rabin-Karp算法

- 基本思想
  - 为便于说明，假设Σ={0,1,2,...,9}，这样每个字符都是一个十进制数字（可以推广到一般情况，可以假定每个字符都基数为d=| Σ |表示法中的一个数字）
  - 可以用一个长度为k的十进制数表示由k个字符组成的字符串
  - 模式P[1..m]，p为其对应的十进制数的值
  - 对于给定的文本T[1..n]，用t_s表示其长度为m的子串T[s+1..s+m]对应的十进制数的值
  - t_s=p当且仅当T[s+1..s+m]= P[1..m]

Rabin-Karp算法

- 如果可以在Θ(m)时间内计算p的值，并在Θ(n-m+1)时间内计算出所有t_s的值，那就可以通过把p值和每个t_s值进行比较，就可以在Θ(m) +Θ(n-m+1)= Θ(n)的时间内，求出所有的匹配位置（即有效位移，此处不考虑p和t_s 是很大的数的情况）

Rabin-Karp算法

- 可以运用霍纳法则(Horner’s rule)在$\Theta(m)$的时间内计算p的值，类似地，也可以在$\Theta(m)$时间内根据T[1..m]计算t_0的值
- 对于d=10，为在$\Theta(n-m)$时间内计算出$t_1,t_2,\ldots,t_{n-m}$,可在常数时间内根据下式计算
- $t_{s+1}=10(t_s-10^{m-1}T[s+1])+T[s+m+1]$

Rabin-Karp算法

- 存在的问题
  - p和t_s的值很大时怎么办？
  - 取模运算，选择q，对p和t_s进行取模运算，可在在$\Theta(m)$时间内计算出p取模后的值，可在$\Theta(n-m+1)$时间内计算出t_s取模后的值
  - 通常q为一个素数
  - 取模，带来的一个问题就是存在伪命中点，也就是取模后的值相等，但实际上并不相等；可以进行进一步测试，显式检查P[1..m]和T[s+1..s+m]是否相等

Rabin-Karp算法

```
RABIN-KARP-MATCHER(T, P, d, q)
n ← length[T]
m ← length[P]
h ← d^(m-1) mod q
p ← 0
t_0 ← 0
for i ← 1 to m ▹ Preprocessing.
        do p ← (dp + P[i]) mod q
           t_0 ← (dt_0 + T[i]) mod q
for s ← 0 to n - m ▹ Matching.
        do if p = t_s
                then if P[1 .. m] = T [s + 1 .. s + m]
                        then print "Pattern occurs with shift" s
           if s < n - m
                then t_(s+1) ← (d(t_s - T[s + 1]h) + T[s + m + 1]) mod q
```

Rabin-Karp算法

- 复杂度分析
  - 预处理时间为Θ(m)，匹配时间最坏情况下为Θ((n-m+1)m)
  - 在实际应用中，有效位移很少（可能只有常数c个），算法的期望匹配时间为O((n-m+1)+cm)=O(n+m)加上处理伪命中点所需的时间

Knuth-Morris-Pratt算法（KMP算法）

- 基本思想
  - 根据模式P的信息，在某个位置匹配失败时，利用已经得到的“部分匹配”结果将模式向右移动尽可能远的距离

Knuth-Morris-Pratt算法（KMP算法）

- 模式移动的距离由next[]决定
- next[]如何确定？
- next[]只与模式字符串相关，与目标字符串无关

$$next[j]=\begin{cases}0, & j=1\text{时}\\ \text{Max}\{\text{k}|\text{1<k<j}\text{且}'p_1p_2..p_{k-1}'='p_{j-k+1}..p_{j-1}'\}, & j\neq 1\text{时}\\ 1, & j\neq 1\text{时其它情况}\end{cases}$$

Knuth-Morris-Pratt算法（KMP算法）

- 对于如下模式串，求next[]值
  - a b c a b a a
  - a b a a b c a c
  - a b a b a a b a b