# 字符串模式匹配的新算法

## By lccycc 李晔晨

## 北京大学 计算机系08级

这个算法来源于一道匹配题<http://acm.zju.edu.cn/onlinejudge/showProblem.do?problemCode=3298>

众所周知 字符串匹配一般都用KMP算法 KMP算法分两步 第一步是求模板串P的next数组 第二步是用next数组匹配 匹配的结果是 对字符串s[i],得到 能够以i结尾的 模板串的最大匹配

这个值记做 T[i]

比如说 如果T[6]=3 那么就是说 s[4..6]和p[0..2]是匹配的 并且这个T的值是最大的

但是注意到 T[i]表示的是结尾的时候最大 如果我们要求 将模板串的第一个字符放在i 能达到的最大匹配呢

这个要求 KMP算法是无法实现的 下面介绍一个新算法 这个算法复杂度是O(n+m)的 其中n是串s的长度 m是模板串P的长度 并且对s的所有位置 能够求出将P放在这个位置做开头达到的最大匹配长度G 同时可以证明 KMP算法得到的不过只是本算法的一个副产品罢了

算法分两部分

### 第一部分 和KMP相同 对模板串进行预处理

我们求F[i] 表示 P的长度为i的前缀P[0..i-1] 在P中能够达到的最大循环匹配长度

举例说 如果P=ababababac 那么F[1]=1,”a”在P中只能走一次F[2]=F[4]=F[6]=7,因为”ab”可以一直匹配到串尾前一个 F[3]=3同理

可知F[i]>=i 现在 将用O(m)的算法来求F[1]..F[m]

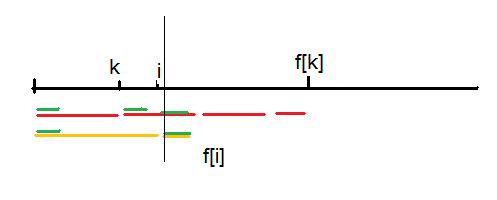
首先 对F[1] 这就是P[0]的连续重复次数 直接线扫一遍

现在假设我们已经知道了F[1..i-1] 求F[i] 设F[1..i-1]中 最大的是F[k],k<i 现在求F[i] 假设i<F[k]

如果P[i]=P[0] 那么我们有P[i]=P[0]=P[k] 于是P[i]=P[k]

如果P[i+1]=P[1] 那么P[i+1]=P[1]=P[k+1]

也就是说 P[i..F[i]-1]=P[0..F[i]-i-1]=P[k..F[i]-i-1+k]



注意 图中绿色的部分是完全相同的 那么 绿色部分的长度+i就等于F[i]

现在要问 绿色部分能到多长呢?注意到上面公式中P[i..F[i]-1]=P[k..F[i]-1-(i-k)]

我们特别关注图中的第二段红线 这个红线上的两条绿线是什么意思呢 意思是 字符串P和P从i-k开始的后缀P+i-k的公共字串的长度!

而 将P的i-k前缀p[0..i-k-1]添加进去 我们得到了P[i-k]的最大循环匹配次数!

也就是说 F[i]=k+F[i-k]

这个公式在什么时候有效呢? 在用F[i-k]的时候 是从k开始匹配的 并不是原串 我们能达到的已知长度是F[k] 因此这个公式只能告诉我们在P[0..F[k]-1]范围内 F[i]能达到的最大值 一旦k+F[i-k]>=F[i] (可能)超过的部分就无能为力了 但是这时候 前i个前缀的F值中 取到最大值的马上由k转到i了 我们令F[i]=F[k] 然后暴力向后继续匹配 就能求出F[i]

另外一种情况:前面假设i<F[k].如果i>=F[k] 那么情况类似 重新开始匹配 但是这种情况当且仅当i=F[k]时出现 所以可以不考虑.

在全局过程中 只有当最大值的取值点发生变化时才向后暴力匹配 这个过程是单调上升的 所以字符串P的每个字符只会暴力到一次 所以总复杂度是O(m)的

代码非常短

void count(int\* p,int\* f,int m)

{

int i,k;

for (f[k=1]=1;f[1]<m&&p[f[1]]==p[0];f[1]++);//初始化

for (i=2;i<=m;i++)

if (f[i-k]+k<f[k]) f[i]=f[i-k]+k;//如果在已知范围内 直接计算

else

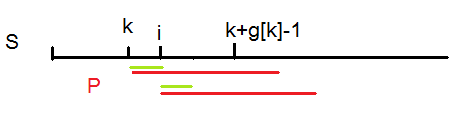
for (f[i]=f[k],k=i;f[i]<m&&p[f[i]]==p[f[i]%i];f[i]++);

//更新k记录的最大取值点 同时向后暴力匹配

}

## 第二部分 利用求出的F 对字符串s进行匹配 求出以任意位置开头能达到的最大匹配G[0..n-1]

同样对G[0]先初始化 假设我们已经求出了G[0..i-1] 并且G[t]+t的最大值在k的时候取到 现在计算G[i] 假设k+G[k]>i,i在已知范围内



如图 如果能看懂第一部分的话 那么这里是非常清楚的 在i位置开头的时候 P最多能匹配多少呢?假设匹配的部分是绿线 那么 图中两条绿线是相同的 也就是说 P最多能匹配F[i-k]那么多 于是G[i]=F[i-k]-(i-k)

同理 如果k+F[i-k]>=k+G[k] 那么超出已知范围 但是此时有F[i-k]>=G[k] 于是 G[i]>=G[k]-i+k 也就是说G[i]+i>=G[k]+k 于是最大取值点k转移到i. 同样令G[i]=G[k]-i+k 然后从G[i]+i的位置开始向后对S和P进行匹配 可以求出G[i]

同时有另外一种情况需要考虑 就是i>=G[k]+k 这时候虽然情况相同 但是G[i]=0然后暴力 否则会出现负数..

由于G[k]+k是递增的 全局的S和P匹配中,S的每个元素恰好被扫到一次 所以是O(n)的

匹配代码:

void pipei(int\* s,int n,int\* p,int m,int\* f,int\* g)

{

int i,k;

for (g[k=0]=0;g[0]<m&&g[0]<n&&p[g[0]]==s[g[0]];g[0]++); //初始化

for (i=1;i<n;i++)

{

g[i]=f[i-k]-(i-k);//直接计算

if (g[i]+i>=g[k]+k||i>=g[k]+k)//如果超了

{

g[i]=g[k]-i+k;//第一种情况

if (g[i]<0) g[i]=0;//第二种情况

while (g[i]+i<n&&g[i]<m&&s[g[i]+i]==p[g[i]]) g[i]++; //向后匹配

k=i; //更新最优取值点

}

}

}

# 应用

1 <http://acm.zju.edu.cn/onlinejudge/showProblem.do?problemCode=3298>

题目给出某串S和模板串P 求P在S中的最大循环匹配 其中匹配起始可以不一定从P的开头开始

使用如上算法 正反做一次计算以任意位置开头结尾的最大匹配 然后两端合并

2

<http://acm.pku.edu.cn/JudgeOnline/problem?id=3461>

标准匹配 求P在S中出现的次数 直接在G中查找多少个=m即可

3 <http://acm.pku.edu.cn/JudgeOnline/problem?id=2406>

求一个串S最多可以是某个字串的重复次数

对S求F 在求F过程中 遇到的第一个F[i]=n且n%i==0的 输出n/i返回

4 <http://acm.pku.edu.cn/JudgeOnline/problem?id=1961>

**这题证明了该算法可以超越KMP**

先对s算F 再用F自匹配算G 再维护单调队列 则得到用KMP求出的next数组

具体方法是:维护单调队列q,q中元素q[i]满足G[q[i]]+q[i]<G[q[i+1]]+q[i+1]

于是 对G线扫 每次扫到i的时候 将G[q[head]]+q[head]<i的全部出队 这时候 如果队非空 那么队头就是满足 将s放在自己的某个位置上 能匹配到s[i-1] 的最小值 这就是我们要求的next数组的定义了 于是有next[i]=i-q[head];

然后 如果队空那么next[i]=0 同时将G[i]插入q中

其中 队列如果只使用两个指针代替 则队列所占空间可省去

关键代码:

count(s,f,n);//计算f

pipei(s,n,s,n,f,g);//自匹配计算g

next[head=tail=0]=-1;//next[0]置-1

for (i=1;i<=n;i++)

{

While (head<tail&&g[q[head]]+q[head]<i) head++; //不符合的出队

if (head==tail) next[i]=0;

else next[i]=i-q[head]; //得到next[i]

if (head>=tail||g[i]+i>g[q[tail-1]]+q[tail-1]) q[tail++]=i; //进队

}

## 总结

于是我们用O(m)的预处理和O(n)的匹配来计算出了两个数组:模式串前缀的最大循环匹配F和字符串S的任意位置开始的最大匹配G 由G线扫+单调队列可以立即得到KMP的所有信息 同时 若G[i]==m 则可知S[i..i+m-1]是完全匹配的 如上证明 该算法可以线性得到KMP算法 但是由KMP却无法反过来得到G 所以在相同复杂度下 这个算法比KMP得到更多的信息 因此更为强大