算法基础HW15

PB18000203 汪洪韬

2021 年 1 月 3 日

32.3-3 如果由$P_k \supseteq P_q$导出$k = 0$或$k = q$，则称模式P是**不可重叠的**。试描述与不可重叠模式相应的字符串匹配自动机的状态转换图。

solution:

状态转换图中，若状态返回，只会返回到第一个点（若当前字母不为模式第一个字母），或者返回到第二个点（当前字母为模式的第一个字母）。

32.4-2 给出关于q的函数 $\pi^*[q]$ 的规模的上界。举例说明所给出的上界是严格的。

solution:

上界为模式P的长度。举例：对于一个字符全相同的模式P，如$P = aa\cdots a$，长度为m，由定义 $\pi^*[q]$，则 $\pi^*[q] = \{0, 1, 2, \cdots, m-1\}$，规模为m。

32.4-6 试说明如何通过以下方式对过程KMP-MATCHER进行改进：把第七行（不是第12行中）出现的π替换为π'，其中对于$q = 1, 2, \cdots, m-1$，π'递归定义如下：

$$\pi'[q] = \begin{cases} 0, & if \quad \pi[q] = 0 \\ \pi'[\pi[q]], & if \quad \pi[q] \neq 0 \&\& P[\pi[q]+1] = P[q+1] \\ \pi[q], & if \quad \pi[q] \neq 0 \&\& P[\pi[q]+1] \neq P[q+1] \end{cases}$$

试说明为什么修改后的算法是正确的，并说明在何种意义上，这一修改是对原算法的改进。

solution:

只需证明在结束6，7行的循环后，q的值与修改前循环后得到的值不变即可。第6,7行while循环的作用是：当匹配$P[q+1] \neq T[i]$时，寻找最大的k满足：$k < q$ && P_k是P_q的后缀，同时，要求$P[k+1] == T[i]$。而π'与π不同的是，它仅仅选择寻找最大的k，满足$k < q$ && P_k是P_q的后缀，$P[k+1] \neq P[q+1]$。所以$\pi' * [q]$是$\pi^*[q]$的子集，其去除了其中$P[k+1] == P[q+1]$的部分。所以$\pi' * [q]$的遍历直接跳过了$P[k+1] == P[q+1]$的k值，这样就对算法进行了优化。

EX 已知：

$T[1\cdots30]$=ACGCTDAGAAGDCAGADGTDAGCDGDAGC.

$P[1\cdots10]$=DAGCDGDAGC.

1.计算Shift Or 算法中的S_c数组($S[T(i)])t$和R数组($state$)变换的情况（给出表格），

2.给出QS算法中的$Q_s - B_c$数组，计算QS算法找到第1个成功匹配所需的字符比较次数；

solution:

1.

$S_A[10\cdots1] =$	$(1101111101111011111100101111111)$
$S_C[10\cdots1] =$	$(0111110111111111101111111110101)$
$S_D[10\cdots1] =$	$(1110101110110111011111011111111)$
$S_G[10\cdots1] =$	$(1011011011101101110110111111011)$
$S_T[10\cdots1] =$	$(1111111111011111111111101111111)$

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
		A	C	G	C	T	D	A	G	A	A	G	D	C	A	G	A	D	G	T	D	A	G	C	D	G	D	A	G	C
1	D	1	1	1	1	1	0	1	1	1	1	1	0	1	1	1	1	0	1	1	0	1	1	1	0	1	0	1	1	1
2	A	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	0	1	1
3	G	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	0	1
4	C	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	0
5	D	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
6	G	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
7	D	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
8	A	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
9	G	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
10	C	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

2.

P=DAGCDGDAGC

	A	C	D	G	T
$Q_s - B_c$	3	1	4	2	11

compare 1 characters, $Q_s - B_c[G]$, shitf by 2;

compare 1 characters, $Q_s - B_c[C]$, shift by 1;

compare 1 characters, $Q_s - B_c[A]$, shitf by 3;

compare 1 characters, $Q_s - B_c[D]$, shift by 4;

compare 1 characters, $Q_s - B_c[A]$, shitf by 3;

compare 1 characters, $Q_s - B_c[D]$, shift by 4;

compare 1 characters, $Q_s - B_c[A]$, shitf by 3;

compare 10 characters, find the match;

the algorithm end, performs 17 charater comparasions.