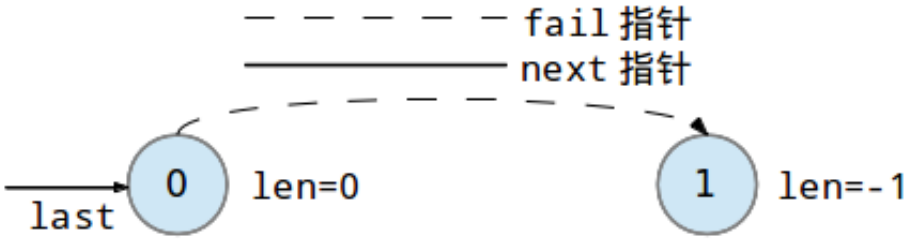
Palindromic Tree——回文树【处理一类回文串问题的强力工具】

分类： 回文树【Palindromic Tree】 2014-12-23 16:14 3427人阅读 评论(8) 收藏 举报

算法学习

今天我们来学习一个神奇的数据结构：Palindromic Tree。中译过来就是——回文树。
那么这个回文树有何功能？
假设我们有一个串S，S下标从0开始，则回文树能做到如下几点：
1.求串S前缀0~i内本质不同回文串的个数（两个串长度不同或者长度相同且至少有一个字符不同便是本质不同）
2.求串S内每一个本质不同回文串出现的次数
3.求串S内回文串的个数（其实就是1和2结合起来）
4.求以下标i结尾的回文串的个数

那么我们该如何构造回文树？
首先我们定义一些变量。
1.len[i]表示编号为i的节点表示的回文串的长度（一个节点表示一个回文串）
2.next[i][c]表示编号为i的节点表示的回文串在两边添加字符c以后变成的回文串的编号（和字典树类似）。
3.fail[i]表示节点i失配以后跳转不等于自身的节点i表示的回文串的最长后缀回文串（和AC自动机类似）。
4.cnt[i]表示节点i表示的本质不同的串的个数（建树时求出的不是完全的，最后count()函数跑一遍以后才是正确的）
5.num[i]表示以节点i表示的最长回文串的最右端点为回文串结尾的回文串个数。
6.last指向新添加一个字母后所形成的最长回文串表示的节点。
7.S[i]表示第i次添加的字符（一开始设S[0] = -1（可以是任意一个在串S中不会出现的字符））。
8.p表示添加的节点个数。
9.n表示添加的字符个数。
一开始回文树有两个节点，0表示偶数长度串的根和1表示奇数长度串的根，且len[0] = 0，len[1] = -1，last = 0，S[0] = -1，n = 0，p = 2（添加了节点0、1）。



编号	0	1	2	3	4	5	6	7	8
S	-1								

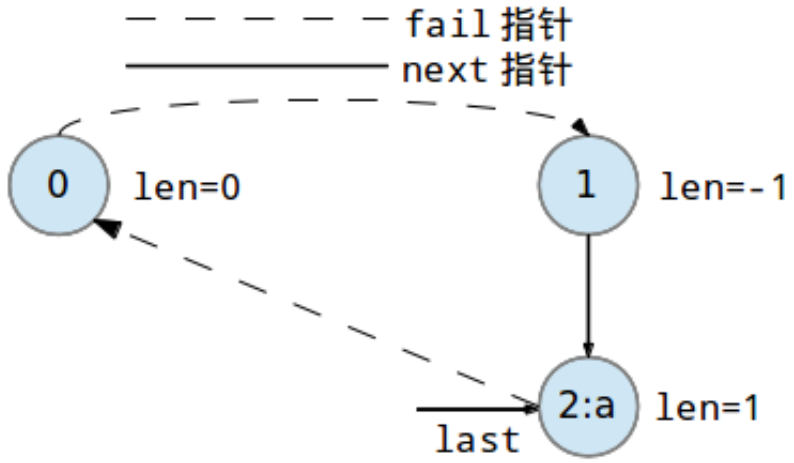
假设现在我们有串S = abbaabba。

首先我们添加第一个字符'a'，S[++ n] = 'a'，然后判断此时S[n - len[last] - 1]是否等于S[n]，即上一个串-1的位置和新添加的位置是否相同，相同则说明构成回文。否则，last = fail[last]。此时last = 0，我们发现S[1 - 0 - 1] != S[1]，所以last = fail[last] = 1，然后我们发现S[1 - (-1) - 1] == S[1]（即自己等于自己，所以我们让len[1]等于-1可以让这一步更加方便）。

令cur等于此时的last（即cur = last = 1），判断此时next[cur]['a']是否已经有后继，如果next[cur]['a']没有后继，我们就进行如下的步骤：新建节点（节点数p++，且之后p = 3），并让now等于新节点的编号（now = 2），则len[now] = len[cur] + 2（每一个回文串的长度总是在其最长子回文串的基础上在两边加上两个相同的字符构成的，所以是+2，同时体现出我们让len[1] = -1的优势，一个字符自成一个奇回文串时回文串的长度为(-1) + 2 = 1）。然后我们让fail[now] = next[get_fail (fail[cur])]['a']，即得到fail[now]（此时为fail[2] = 0），其中的get_fail函数就是让找到第一个使得S[n - len[last] - 1] == S[n]的last。然后next[cur]['a'] = now。

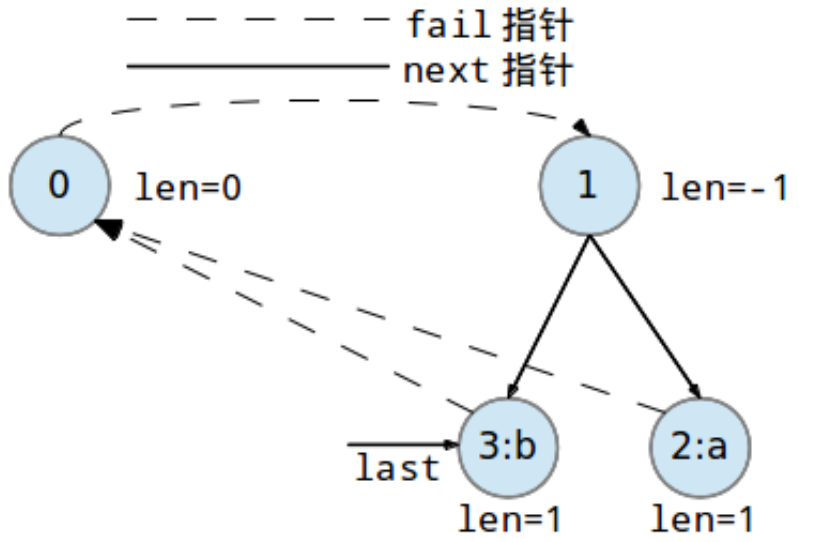
当上面步骤完成后我们让last = next[cur][c]（不管next[cur]['a']是否有后继），然后cnt[last] ++。

此时回文树为下图状态：



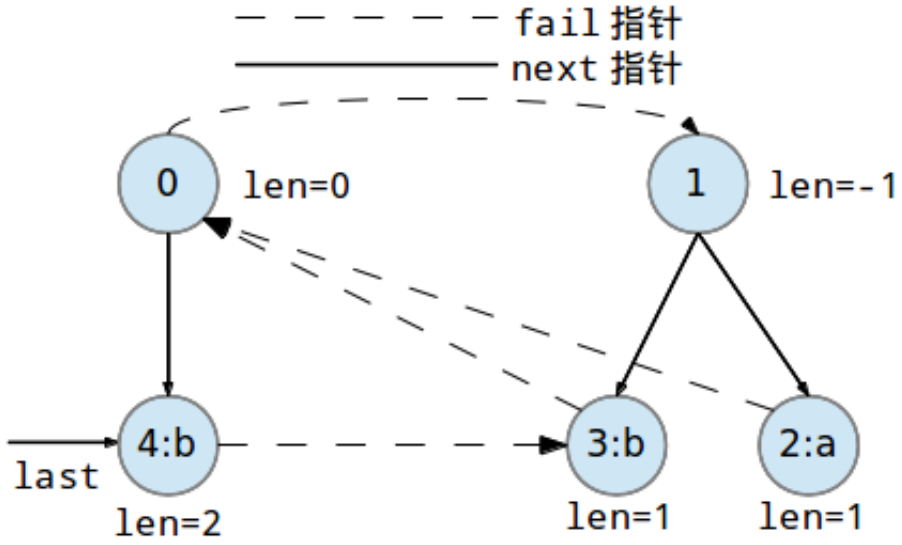
编号	0	1	2	3	4	5	6	7	8
S	-1	a							

现在我们添加第二个字符字符'b'到回文树中：



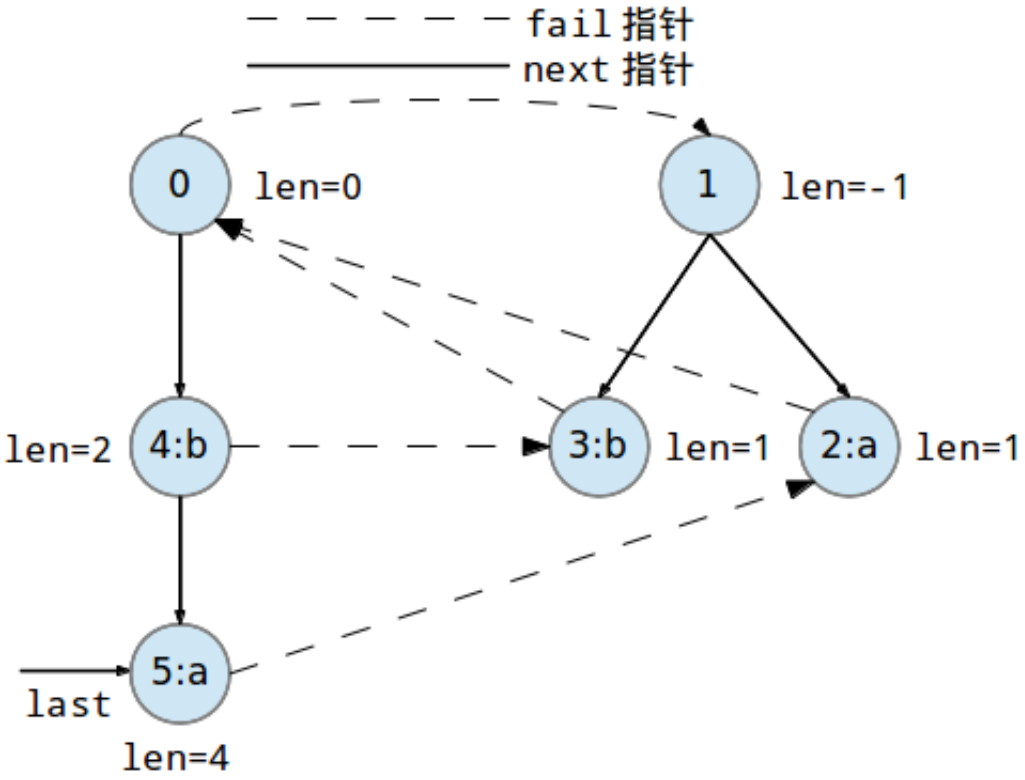
编号	0	1	2	3	4	5	6	7	8
S	-1	a	b						

继续添加第三个字符'b'到回文树中：



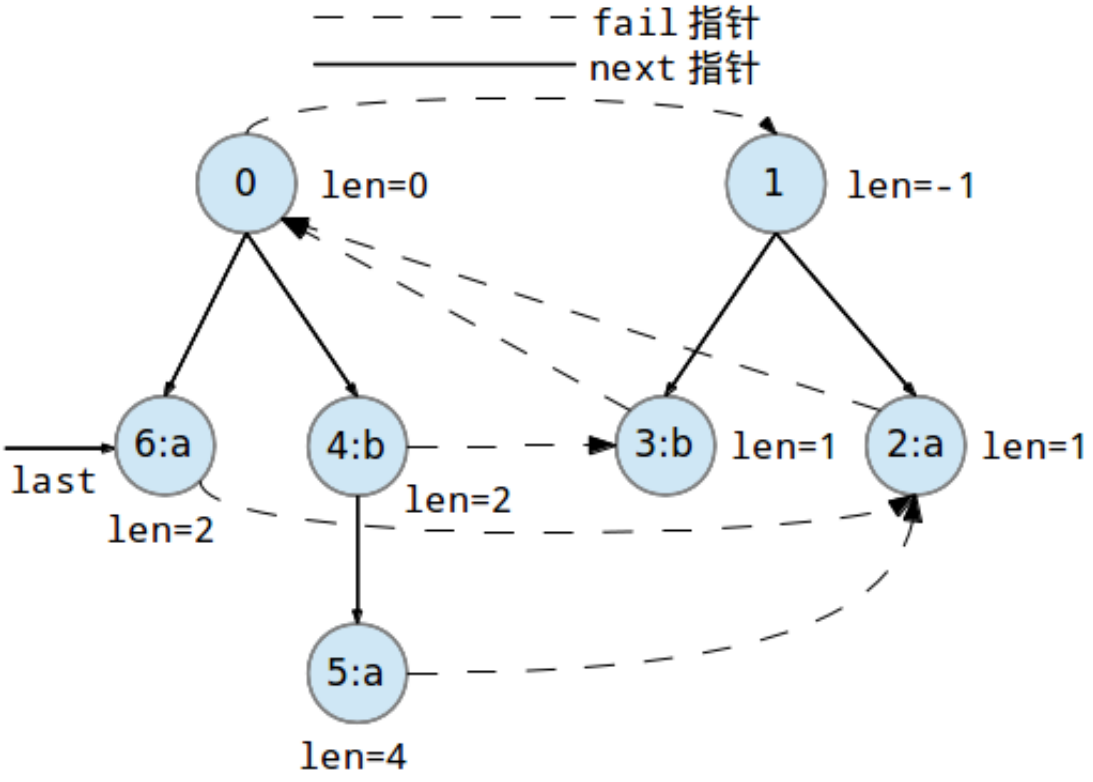
编号	0	1	2	3	4	5	6	7	8
S	-1	a	b	b					

继续添加第四个字符'a'到回文树中：



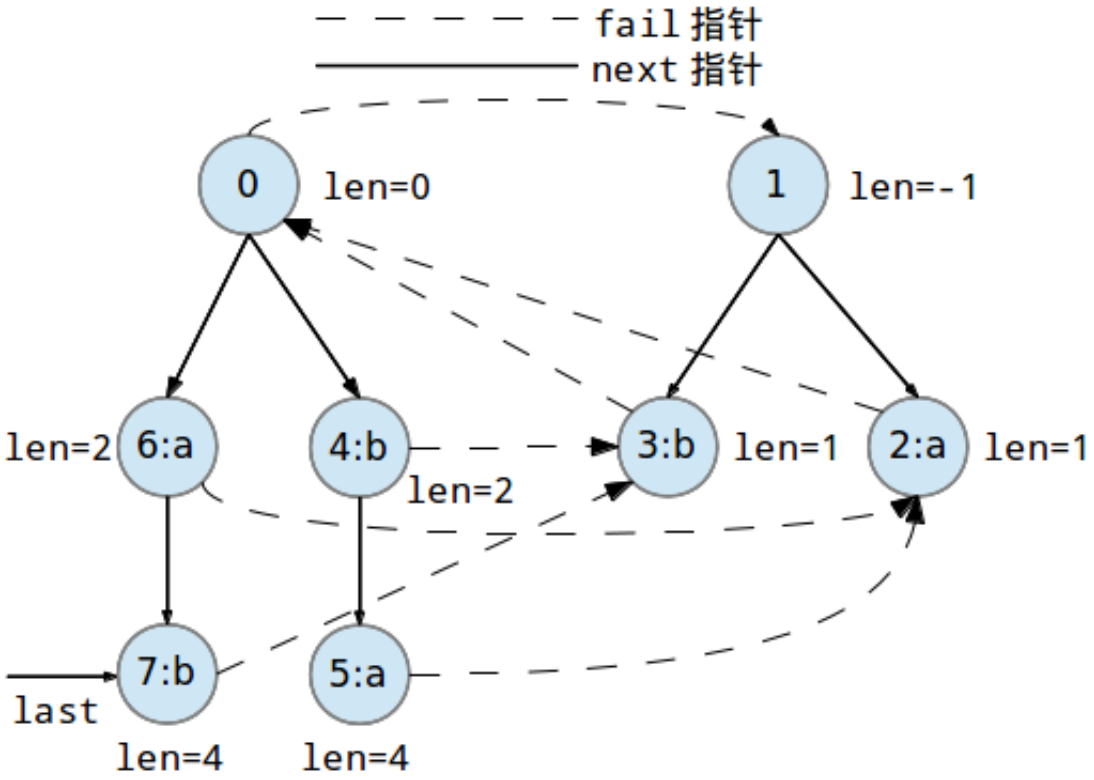
编号	0	1	2	3	4	5	6	7	8
S	-1	a	b	b	a				

继续添加第五个字符'a'到回文树中：



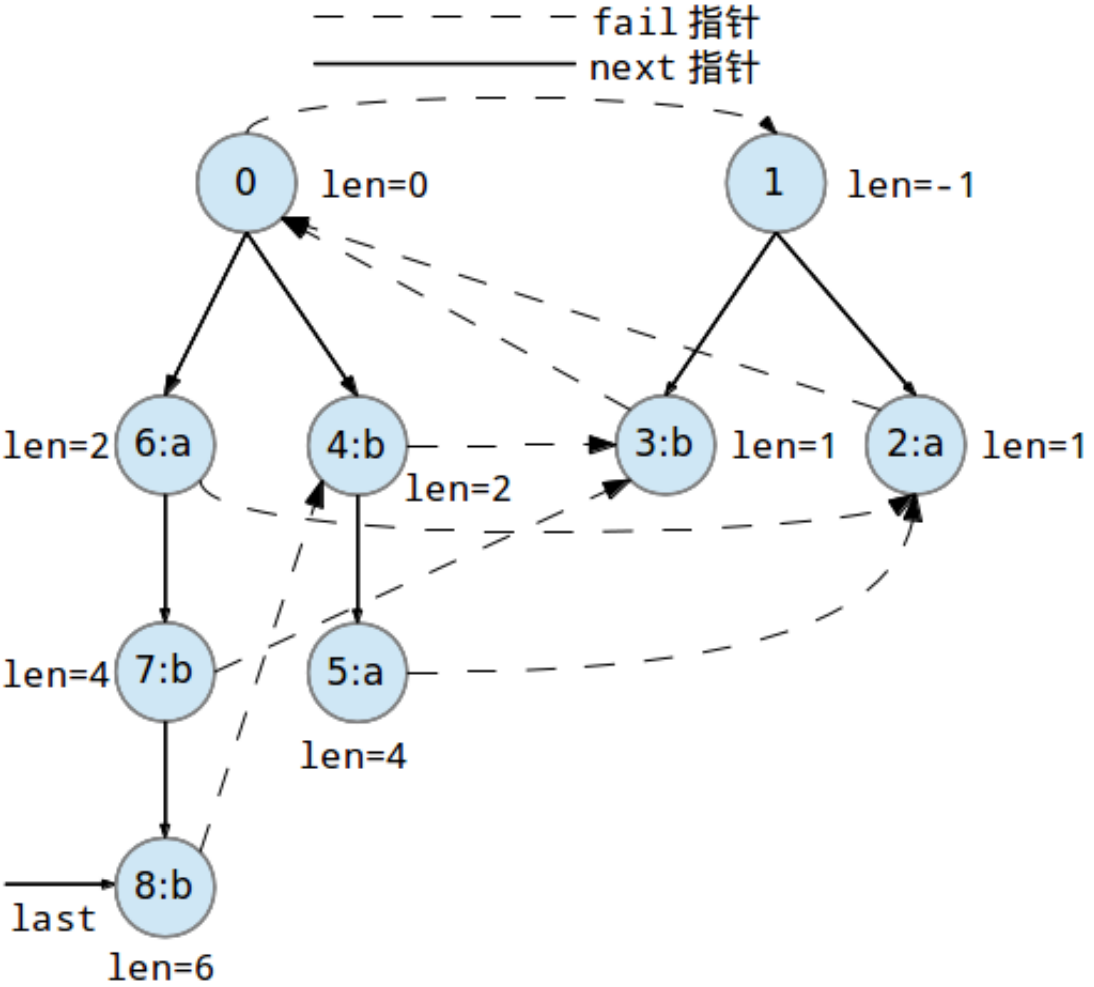
编号	0	1	2	3	4	5	6	7	8
S	-1	a	b	b	a	a			

继续添加第六个字符'b'到回文树中：



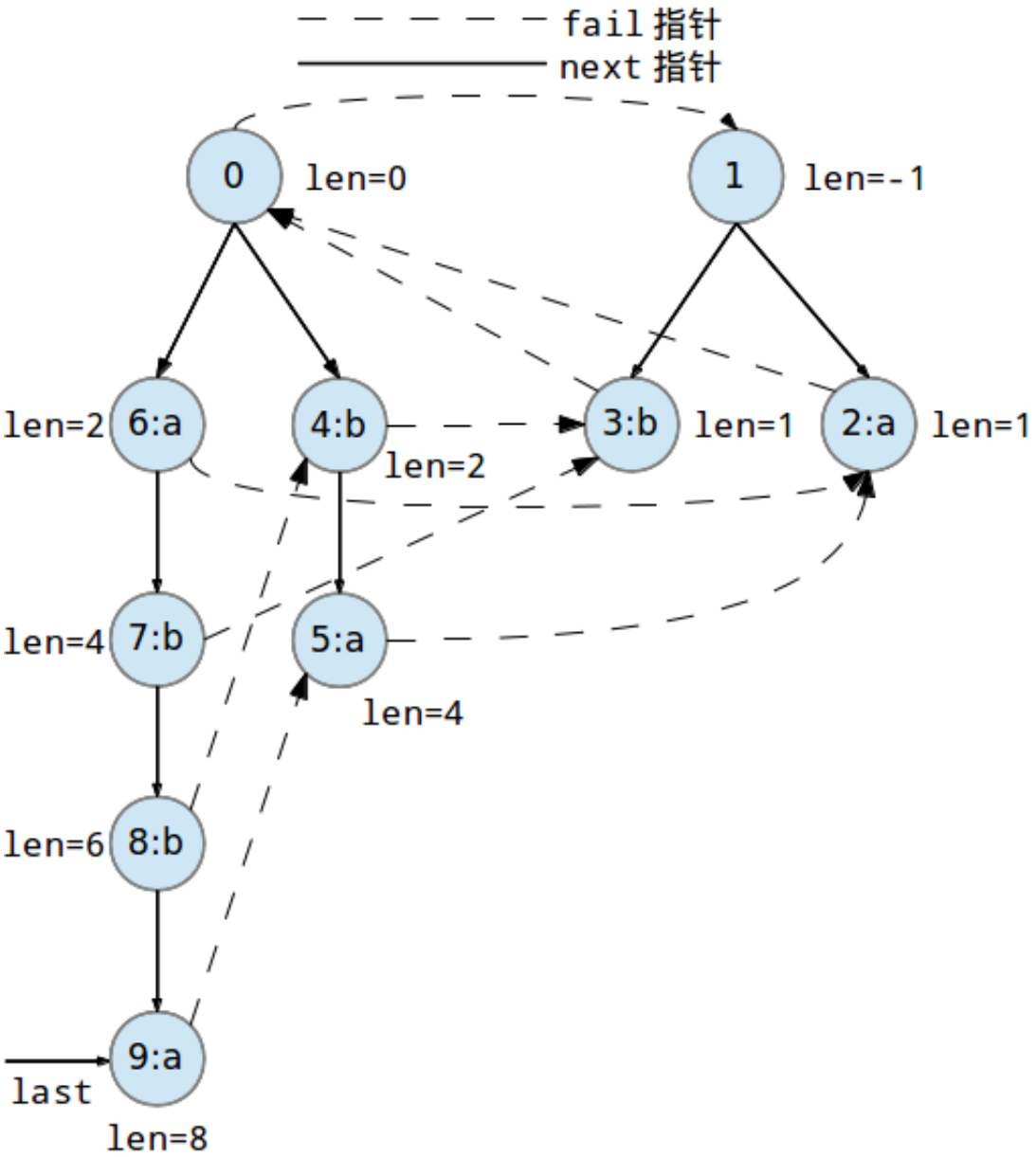
编号	0	1	2	3	4	5	6	7	8
S	-1	a	b	b	a	a	b		

继续添加第七个字符'b'到回文树中：



编号	0	1	2	3	4	5	6	7	8
S	-1	a	b	b	a	a	b	b	

继续添加第八个字符'a'到回文树中：



编号	0	1	2	3	4	5	6	7	8
S	-1	a	b	b	a	a	b	b	a

到此，串S已经完全插入到回文树中了，现在所有的数据如下：

节点	0	1	2	3	4	5	6	7	8	9
char	无	无	a	b	b	a	a	b	b	a
fail	1	无	0	0	3	2	2	3	4	5
len	0	-1	1	1	2	4	2	4	6	8
cnt	无	无	1	1	1	1	1	1	1	1

然后我们将节点x在fail指针树中将自己的cnt累加给父亲，从叶子开始倒着加，最后就能得到串S中出现的每一个本质不同回文串的个数。

构造回文树需要的空间复杂度为O（N*字符集大小），时间复杂度为O（N*log（字符集大小）），这个时间复杂度比较神奇。如果空间需求太大，可以改成邻接表的

形式存储，不过相应的要牺牲一些时间。
总的来说，这是一个很好的算法~

下面给上我的code：

[cpp] view plain copy print ?

```cpp
const int MAXN = 100005 ;
const int N = 26 ;

struct Palindromic_Tree {
    int next[MAXN][N] ;//next指针，next指针和字典树类似，指向的串为当前串两端加上同一个字符构成
    int fail[MAXN] ;//fail指针，失配后跳转到fail指针指向的节点
    int cnt[MAXN] ;
    int num[MAXN] ;
    int len[MAXN] ;//len[i]表示节点i表示的回文串的长度
    int S[MAXN] ;//存放添加的字符
    int last ;//指向上一个字符所在的节点，方便下一次add
    int n ;//字符数组指针
    int p ;//节点指针

    int newnode ( int l ) {//新建节点
        for ( int i = 0 ; i < N ; ++ i ) next[p][i] = 0 ;
        cnt[p] = 0 ;
        num[p] = 0 ;
        len[p] = l ;
        return p ++ ;
    }

    void init () {//初始化
        p = 0 ;
        newnode (  0 ) ;
        newnode ( -1 ) ;
        last = 0 ;
        n = 0 ;
        S[n] = -1 ;//开头放一个字符集中没有的字符，减少特判
        fail[0] = 1 ;
    }

    int get_fail ( int x ) {//和KMP一样，失配后找一个尽量最长的
        while ( S[n - len[x] - 1] != S[n] ) x = fail[x] ;
        return x ;
    }

    void add ( int c ) {
        c -= 'a' ;
        S[++ n] = c ;
        int cur = get_fail ( last ) ;//通过上一个回文串找这个回文串的匹配位置
        if ( !next[cur][c] ) {//如果这个回文串没有出现过，说明出现了一个新的本质不同的回文串
            int now = newnode ( len[cur] + 2 ) ;//新建节点
            fail[now] = next[get_fail ( fail[cur] )][c] ;//和AC自动机一样建立fail指
```

```
针，以便失配后跳转
            next[cur][c] = now ;
            num[now] = num[fail[now]] + 1 ;
        }
        last = next[cur][c] ;
        cnt[last] ++ ;
    }

    void count () {
        for ( int i = p - 1 ; i >= 0 ; -- i ) cnt[fail[i]] += cnt[i] ;
        //父亲累加儿子的cnt，因为如果fail[v]=u，则u一定是v的子回文串！
    }
} ;
```

看到这大家应该对回文树有所了解了吧？

下面是一些题目，感兴趣的话可以做一下~

1.ural1960. Palindromes and Super Abilities

2.TsinsenA1280. 最长双回文串

3.TsinsenA1255. 拉拉队排练

4.TsinsenA1393. Palisection

5.2014-2015 ACM-ICPC, Asia Xian Regional Contest G The Problem to Slow Down You