原创 回文树

2019-09-09 15:41:31 _-Y-_-Y-_ 阅读数 12 更多 编辑

回文树

形态

回文树是由 **两棵树** 和 许多**后缀链** 构成。

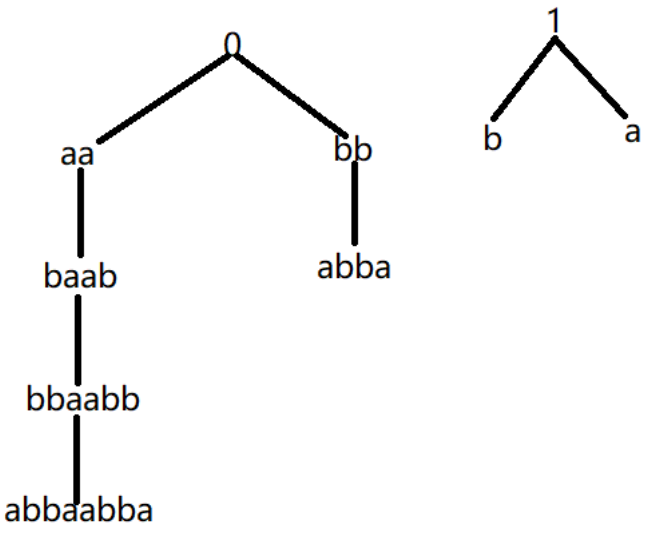
一棵树储存长度为偶数的回文串，另一颗树存储的是长度为奇数的回文串

后缀链：在回文树中假如一个节点a最长可匹配后缀是节点b所对应的字符串，那么就由a向b连一条fail边。

字符串： $abbaabba$ 下的所有回文串

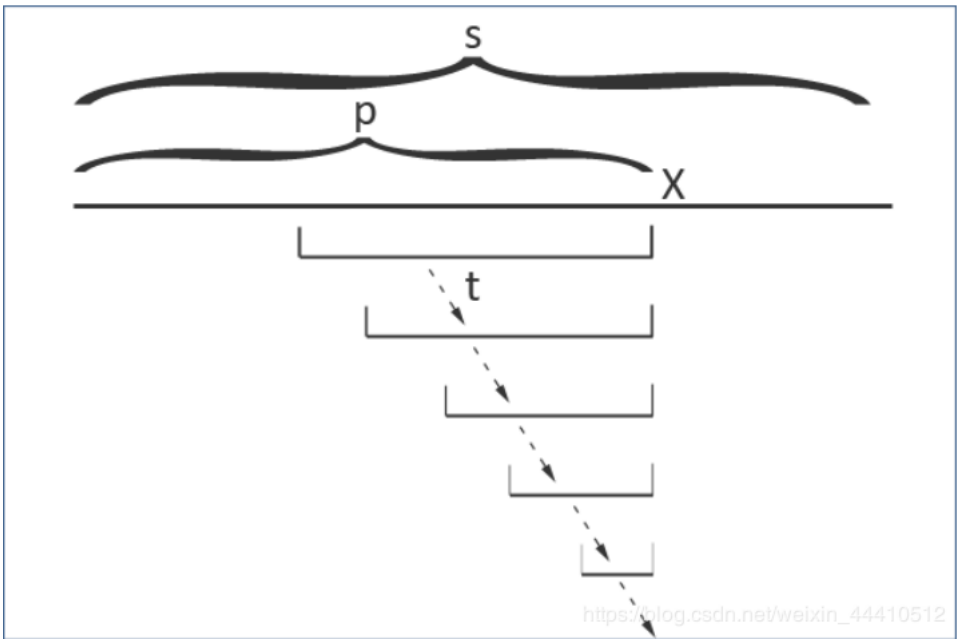
以0为根的树：表示长度为偶数的回文串

以1为根的树：表示长度为奇数的回文串

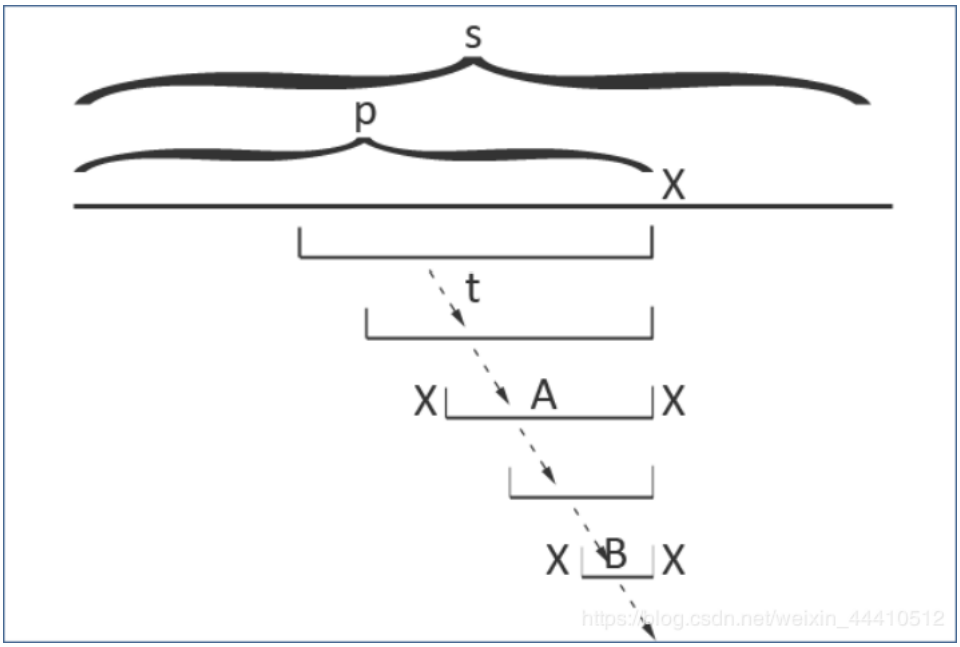


构建

如图，要在加入串P后面加入一个新的字符X构成一个新的加入串P'
t为P的最长回文后缀(回文串的后缀)，图中虚线为后缀链，指向的是加入串P的所有回文后缀



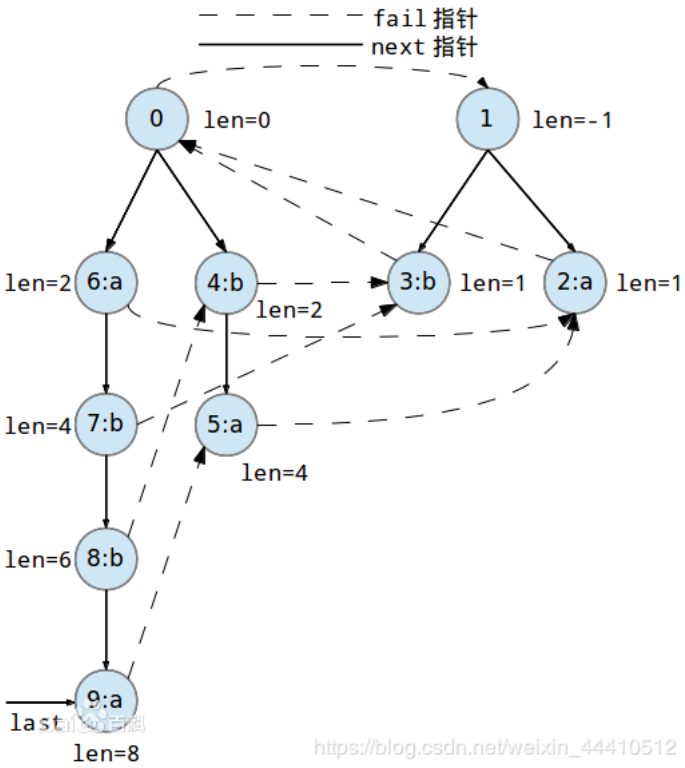
需要找到回文串P的一条回文后缀A，使A的左端点是一个字符X，那么左右个加上一个字符X依然还是一条回文串
A可以是空串(长度为2的回文XX)甚至是-1(长度为1的回文X)
如果已经有一个结点i代表XAX，那么我们就吧t改成i，退出该过程，否则需要新建一个结点
新建结点时需要加入一条后缀链,同样，继续沿着后缀链找，直到找到符合要求的字符串B为止



步骤

先初始化
定义：偶数根 序号为0 长度为0 偶数根 的失配指针指向奇数根
定义：奇数根 序号为1 长度为-1 （方便计算，不用特判）

```
void Init_Tr()//初始化很重要
{
    top = 1, last = 0;
    a[0].len = 0, a[1].len = -1;
    a[0].fail = 1;
}
```



开始建树
和构建一样，开始找 回文后缀 A
第1个字符 ‘a’ 序号为2 A = NULL 所以插在 1 的下面
最长可匹配后缀为 ‘\0’ fail 指向 0

第2个字符 ‘b’ 序号为3 A = NULL 所以插在 1 的下面b
最长可匹配后缀为 ‘\0’ fail 指向 0

第3个字符 ‘b’ 序号为4 A = “b” 由"b"衍生下来，长度为(2)偶数，所以在0的下面
最长可匹配后缀为 b fail 指向 序号 3

第4个字符 ‘a’ 序号为5 A = “bb” 由"bb"衍生下来，长度为(4)偶数，所以在4的下面
最长可匹配后缀为 a fail 指向 序号 2

第5个字符 ‘a’ 序号为6 A = “a” 由"a"衍生下来，长度为(2)偶数，所以在0的下面
最长可匹配后缀为 a fail 指向 序号 2

第6个字符 ‘b’ 序号为7 A = “aa” 由"aa"衍生下来，长度为(4)偶数，所以在6的下面
最长可匹配后缀为 b fail 指向 序号 3

第7个字符 ‘b’ 序号为8 A = “baab” 由"baab"衍生下来，长度为(6)偶数，所以在7的下面
最长可匹配后缀为 bb fail 指向 序号 4

第8个字符 ‘a’ 序号为9 A = “bbaabb” 由"bbaabb"衍生下来，长度为(8)偶数，所以在8的下面
最长可匹配后缀为 abba fail 指向 序号 5

code

代码连接：https://blog.csdn.net/bbbbswbq/article/details/82628928
题目链接：https://cn.vjudge.net/problem/HYSBZ-3676

```
/*
回文串
HYSBZ - 3676
https://cn.vjudge.net/problem/HYSBZ-3676
题意：给你一个字符串 s，求回文子串长度 * 该回文串子出现次数的最大值。
解法：回文树+拓扑序
*/
#include<bits/stdc++.h>
using namespace std;
#define ll long long
```

```
const int N = 3e5+100;
struct node
{
    int len, cnt;//该回文串的长度，该回文串出现的次数
    int next[26];//只有26个小写字母
    int fail;//指向等于最长后缀回文的前缀点
}a[N];
int top, last;
char s[N];
void Init_Tr()//初始化很重要
{
    top = 1, last = 0;
    a[0].len = 0, a[1].len = -1;
    a[0].fail = 1;
}
int i;//减少传参可以优化很大的时间复杂度
int get_id(int now)
{
    while(s[i] != s[i-a[now].len-1]) now = a[now].fail;//判断是否满足回文
    return now;
}
void Insert()
{
    int len = strlen(s+1);
    for(i = 1; i <= len; i++) {
        int t = s[i]-'a';
        int id = get_id(last);
        if(!a[id].next[t]) {
            a[++top].len = a[id].len + 2;//每次前后各增加一个点
            a[top].fail = a[get_id(a[id].fail)].next[t];
            a[id].next[t] = top;
        }
        last = a[id].next[t];//
        a[last].cnt++;
    }
}
ll solve()
{
    ll ans = 0;
    for(int i = top; i >= 2; i--) {//从后往前遍历相当于拓扑序
        a[a[i].fail].cnt += a[i].cnt;
        ans = max(ans, 1LL*a[i].cnt*a[i].len);
    }
    return ans;
}
int main()
{
    scanf("%s", s+1);
    Init_Tr();
    Insert();
    printf("%lld\n", solve());
    return 0;
}

```

有 0 个人打赏

文章最后发布于: 2019-09-09 17:55:52