字符串学习笔记(4) 马拉车(manacher)

严格鸽
柚子厨/萝莉控/ACM银牌

马拉车经常被用来解决这样的问题，求字符串中的最长的回文子串。复杂度为 $O(n)$

我们先考虑如何用其它方法来求。

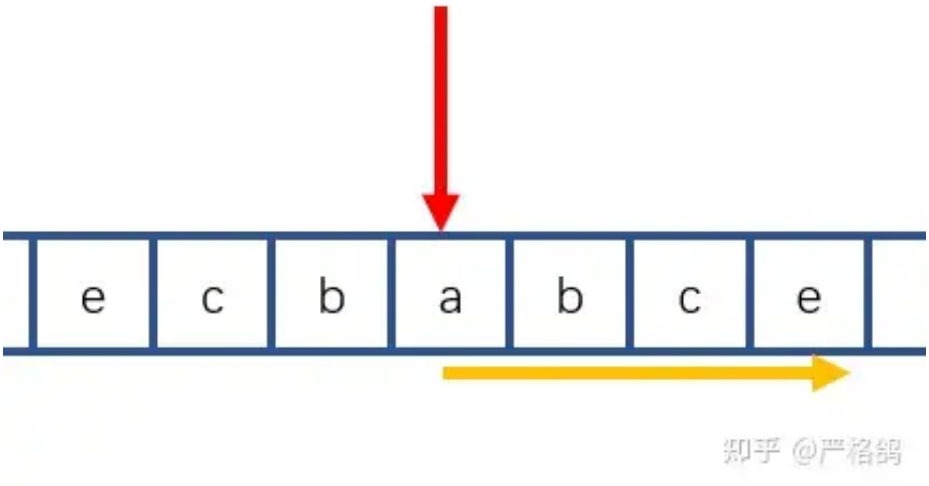
我们可以使用字符串哈希的方法在 $O(1)$ 的时间内，判断子串是否为回文。

```
typedef unsigned long long ull;//自然溢出
namespace str_hash {
    //从1开始计数
    const int maxn = 2e6 + 7;
    ull base = 100071;//经验值
    ull p[maxn];
    char s[maxn];
    unordered_map<ull, ull>rv;
    unordered_map<ull, ull>hs;
    int n;
    void init(string &str) {
        n = str.length();
        for (int i = 1; i <= n; i++)s[i] = str[i - 1];
        p[0] = 1;
        for (int i = 1; i <= n; i++)
        {
            hs[i] = hs[i - 1] * base + s[i];
            p[i] = p[i - 1] * base;
        }
        for (int i = n; i; --i)
```

```
        return hs[r] - hs[l - 1] * p[r - l + 1];
    }
    ull rev(ull l, ull r) {
        return rv[l] - rv[r + 1] * p[r - l + 1];
    }
// rev(L,R) = que(L,R) 说明是回文
}
```

如何找到最长的回文呢?

我们可以用回文中心，和回文半径来唯一确定一个回文 (i, len)

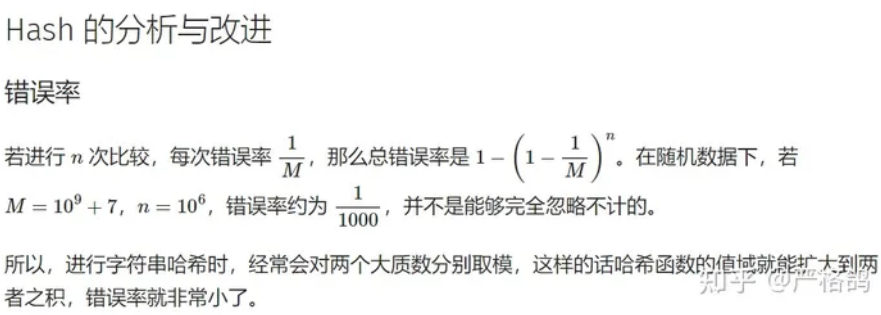


图中的回文半径为4

那么我们有，当 i 是固定的话， len 是具有二分性的。如果 (i, len_1) 不是回文的话，$(i, len_1 + 1), (i, len_1 + 2), (i, len_1 + 3) \ldots$ 肯定也不是回文。

所以我们可以使用字符串哈希加上二分来跑出答案。

不过这样，我们需要调用 $nlogn$ 次字符串哈希。当调用次数比较多的时候，比较容易出现冲突。


Hash 的分析与改进

错误率

若进行 n 次比较，每次错误率 $\frac{1}{M}$，那么总错误率是 $1-\left(1-\frac{1}{M}\right)^n$。在随机数据下，若 $M = 10^9 + 7$，$n = 10^6$，错误率约为 $\frac{1}{1000}$，并不是能够完全忽略不计的。

所以，进行字符串哈希时，经常会对两个大质数分别取模，这样的话哈希函数的值域就能扩大到两者之积，错误率就非常小了。

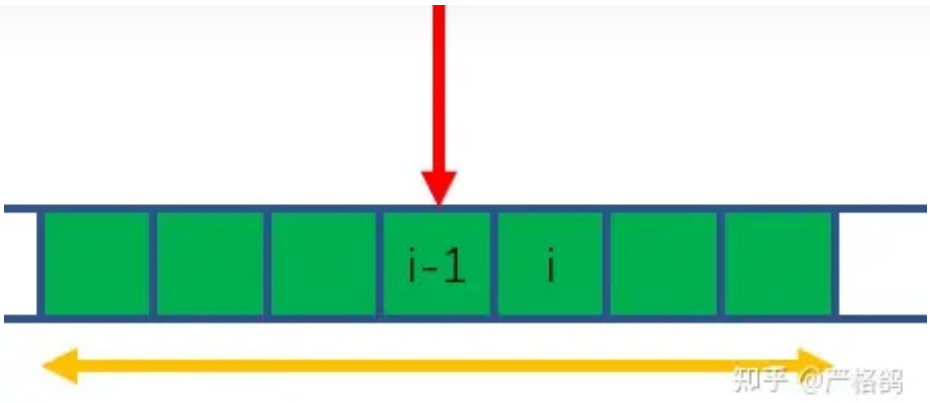
知乎 @严格鸽

（据说是比较次数大于 $\sqrt{mod}$ 的时候比较危险qwq

但是实际上，我们可以在 $O(n)$ 的复杂度下，使用字符串哈希获得最长回文串。

我们记录下当前获得的最长回文的半径。

比如第 $i-1$ 的位置上，我们暴力扩展匹配的长度为 4，我们记录为 $maxlen$



那么对于 i 这个位置，我不二分，我直接从4这个半径开始匹配（每一次都继承之前的 $maxlen$

这样可以发现，我们的复杂度只和 $maxlen$ 的增加有关，复杂度为 $O(n)$ 。

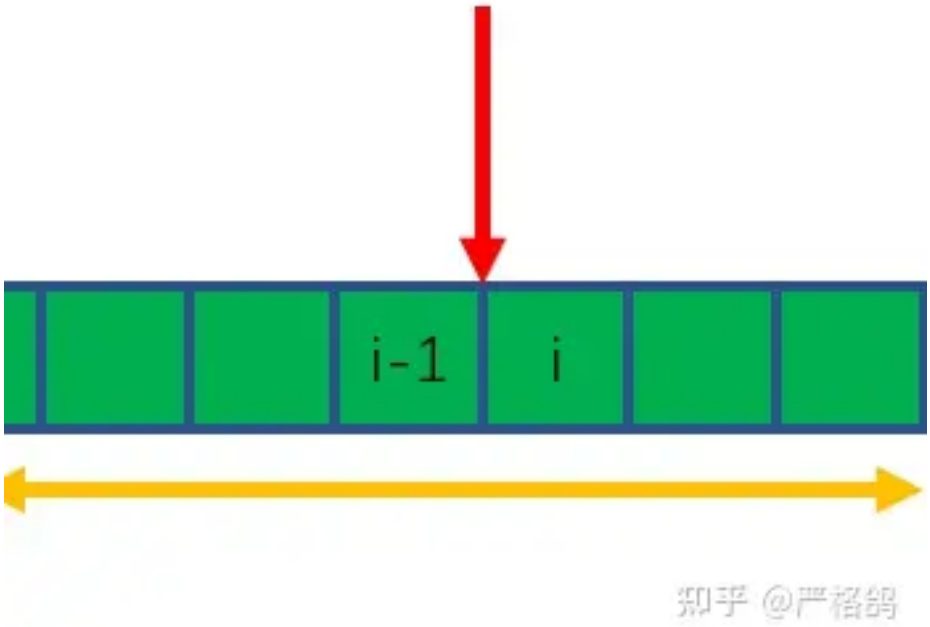
这样我们就解决最长回文子串了。你可能会问，那马拉车是干嘛吃的（qwq

如果我们需要计算，一共有多少个回文子串呢？我们可以通过计算每个点的最大回文半径，然后获得回文子串的数量。

但是这样就不能哈希 $O(n)$ 求了，因为，可能 $i-1$ 的最大半径为4， i 的最大半径为2，但是每次都是让 $maxlen$ 增大的，所以求不出来。

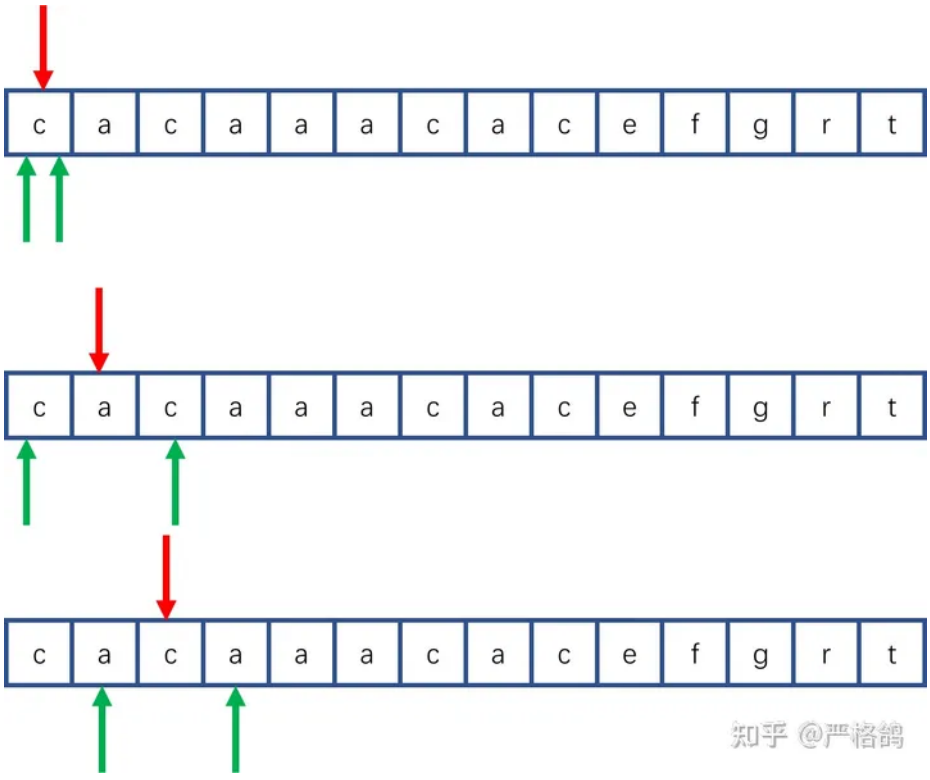
那么马拉车就是可以 $O(n)$ 求出 $len[i]$ (以 i 为中心的最大回文半径)的算法。

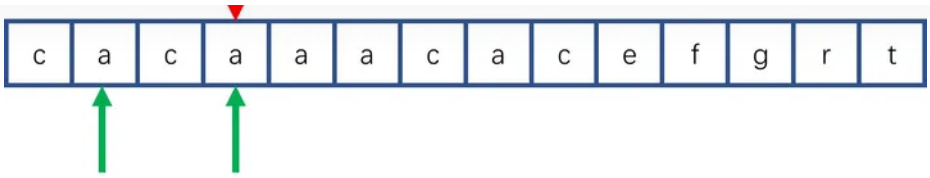
对于回文串有个常见的处理，就是因为回文串分为奇数长度和偶数长度。所以可能回文中心落在两个点中间。

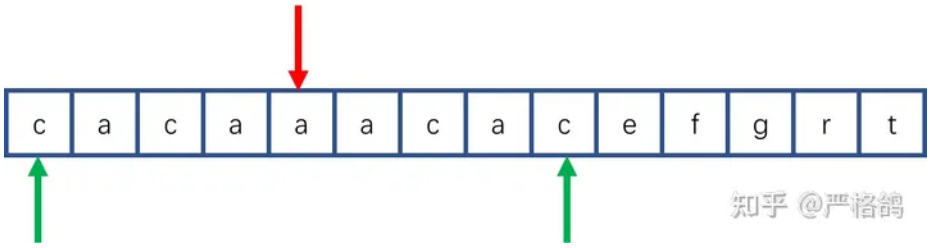


这样偶数回文会被扩展成奇数回文，我们就保证了中心点一定在点上。

马拉车的过程是，从左到右，去跑 $len[i]$，并且维护一个，右端点 R 最右的回文串（用 (L,R) 表示

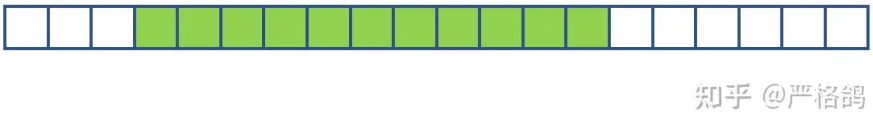






你可能会问，这样有啥用啊。我知道你很急，但是现在还不能急。

如果我们现在需要求 $len[i]$ （我们用绿色表示右端点最右的回文子串）。



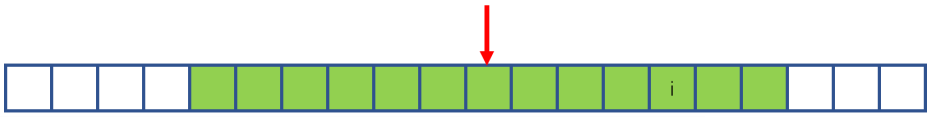
然后就是一个分类讨论了。

如果 $i > R$

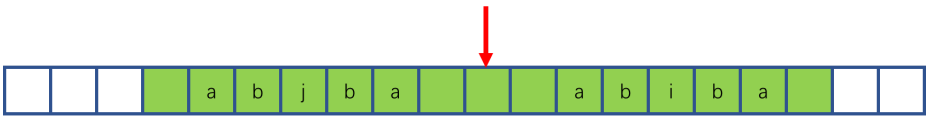
无论如何， R 都会到 i 的右边去（最差就是 $s[i]$ 自己是一个回文串

无论如何 R 都要变大，所以我们直接 $len[i]$ 从1开始**暴力扩展**。

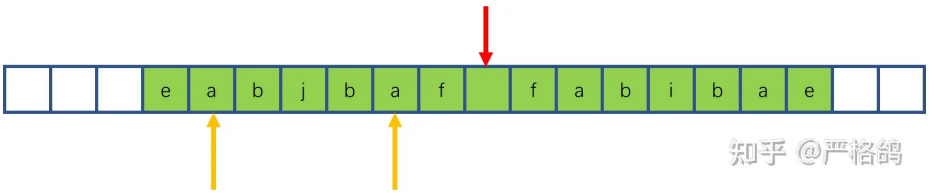
如果 i 落在 R 里面。



根据回文串的性质， $len[i] \geq len[j]$

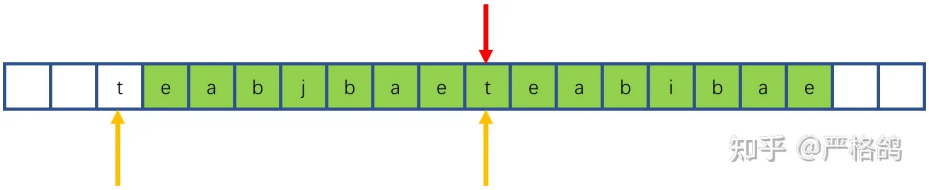


如果说， j 的回文半径落在 (L, R) 里面。

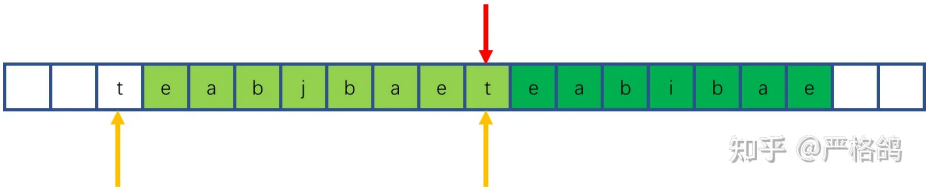


那么我们直接 $len[i] = len[j]$ 即可。

如果 j 的半径超过了 (L, R)



那么我们我们只能获得， i 到 R 这一段是个回文。



然后我们**暴力扩展** $len[i]$ ，顺带扩展 R 。

这样我们来分析下复杂度。我们的每次暴力扩展 $len[i]$ 都会带着 R 的增加，所以复杂度为线性的。

（可以类似上面的， 哈希$O(n)$ 求最长回文子串）

这样我们来看看前几天的牛客多校。

题意：

给定一个字符串，问有多少个以 k, f, c 结尾的回文子串。

做法：

我们可以先求出 $len[i]$ （这个 $len[i]$ 可以是扩展了#的字符串，不会影响计数的）

对于 $len[i]$ ，我们可以“告诉” $[i, i + len[i] - 1]$ 的字符，你们是某个回文的结尾。

这个我们只需要一个区间加法即可，最后获得 k, f, c 的总和即可。

```
vector<int> manacher(string &a)
{
    string b = "$|";
    for (auto i : a)
    {
        b += i;
        b += '|';
    }
   int len = b.length();
    vector<int> hw(b.length());
    int maxright = 1, mid = 1;
    for (int i = 1; i < len; i++)
    {
        if (i < maxright)
            hw[i] = min(hw[mid * 2 - i], hw[mid] + mid - i);
        else
            hw[i] = 1;
        while (b[i - hw[i]] == b[i + hw[i]])
            hw[i]++;
        if (i + hw[i] > maxright)
        {
            maxright = i + hw[i];
            mid = i;
        }
    }
    a = b;
    return hw;
}

int d[MAXN];
int n; string s;
int cnt[128];
void slove() {
    cin >> n;cin >> s;
    auto len = manacher(s);
    n = s.length() - 1;
    for (int i = 1; i <= n; i++) {
        int L = i, R = i + len[i] - 1;
        d[L]++;
        d[R + 1]--;
    }
    int pre = 0;
    for (int i = 1; i <= n; i++) {
        pre += d[i];
        cnt[s[i]] += pre;
    }
    cout << cnt['k'] << " " << cnt['f'] << " " << cnt['c'] << endl;
}
```

发布于 2022-08-03 10:04

字符串　ACM 竞赛　OI（信息学奥林匹克）

发布一条带图评论吧

7 条评论　默认　最新

让我remake吧

这个base经验值没看懂🤔

回复　喜欢

2022-08-16 · IP 属地广东 回复 喜欢

Christophe
大爱严格鸽
2022-08-06 · IP 属地江苏 回复 喜欢

Christophe
蹲严格鸽的 SAM 和 PAM
2022-08-06 · IP 属地江苏 回复 喜欢

唐门
这是信息竞赛
2022-08-04 · IP 属地浙江 回复 喜欢

hinlole
%%%，不过测了一下cnt数组应该得开long long，最大值可能会达到十几次方来着
2022-08-04 · IP 属地广东 回复 喜欢

夏水
写得好棒！
2022-08-03 · IP 属地重庆 回复 喜欢

文章被以下专栏收录

字符串学习笔记
为了不在卡字符串的题目，勇敢勇敢我的朋友

推荐阅读

求解最长回文子串：马拉车(manacher)算法详解
这个算法是在做一道“求最长回文子串”的题目时查到的一个算法，在此之前的方法是遍历所给字符串，使用中心展开的方法求的最长的子串 def getExpandLength(left, right, s): '''...
渐渐弃坑

方程式赛车悬挂设计 转载
尽管数字计算工具发展迅速，悬挂设计仍然不为大众所知。特别是对于公路汽车，悬挂设计方面的资料非常少，因为悬挂设计需要考虑的因素特别多，而且这些资料往往是保密的。对于赛车悬挂设计来...
wwwoooqq

什么是马拉车算法?
噜噜呀

未来概念-足!
pursu...