浅谈回文子串问题

江苏省常州高级中学 徐毅

摘 要

本文首先对用后缀数组求回文半径的算法进行了简单回顾，紧接着对Manacher算法进行了详细介绍，在此基础上对几种类型的回文子串问题分别举例分析，最后对解题的一般思路进行了总结。

1 引言

近年来，随着处理字符串的算法和数据结构在国内的不断推广，与字符串相关的问题越来越多地进入大家的视线。

回文子串问题，是与字符串相关的诸多问题中比较热门的问题之一。由于回文子串有许多优美的性质，此类问题往往灵活性较强。

为此，本文对此类问题进行了一定的分析和总结，希望能起到抛砖引玉的作用。

2 基本概念

- 字符的非空有限集，称为**字母表（alphabet）**。
- 字母表中字符的有限序列，称为**字符串（string）**。
- 字符串中字符的个数，称为该字符串的**长度（length）**。
- 字符串中连续的一段，称为该字符串的**子串（substring）**。
- 字符串中反转后与反转前相同的子串，称为该字符串的**回文子串（palindromic substring）**。

- 字符串中长度最大的回文子串，称为该字符串的**最长回文子串（longest palindromic substring）**。
- 以字符串第i位为中心的回文子串的最大长度的一半，称为该字符串第i位的**回文半径（radius of palindrome）**。

3 常用算法

在处理回文子串问题时，一般需要求出字符串每一位的回文半径。

为了避免奇偶讨论和边界问题，从而降低代码复杂度，我们在字符串每一位两侧都添加同一个特殊字符，在字符串的首位前添加一个不同的特殊字符，在字符串的末位后再添加一个不同的特殊字符，如将abbabcba变为$#a#b#b#a#b#c#b#a#@。

由此我们得到了一个长度为n的新字符串$S[1..n]$，而要求出该字符串每一位的回文半径$R[1..n]$，通常可以采用以下两种算法。

Table 1: S和R的对应关系

S	#	a	#	b	#	b	#	a	#	b	#	c	#	b	#	a	#
R	1	2	1	2	5	2	1	4	1	2	1	6	1	2	1	2	1

3.1 后缀数组

后缀数组的构造大家应该都很熟悉，在这里就不赘述了。而利用后缀数组来求字符串每一位的回文半径，前人也有过介绍，我们再简单回顾一下。

$R[i]$显然等于$S[i..n]$和反转后的$S[1..i]$的最长公共前缀。因此，我们将反转后的原字符串写在原字符串后面，中间用一个特殊字符隔开，这样就把问题变为了求这个新字符串某两个后缀的最长公共前缀，可以很方便地用后缀数组来完成。

- 若使用倍增算法来构造后缀数组，RMQ问题用$O(n\log n)$的时间复杂度来预处理，则总时间复杂度为$O(n\log n)$。
- 若使用DC3算法来构造后缀数组，RMQ问题用$O(n)$的时间复杂度来预处理，则总时间复杂度为$O(n)$。

3.2 Manacher算法

要想顺利解决回文子串问题，通常需要用$O(n)$的时间复杂度来求$R[1..n]$，而利用后缀数组来实现的代码复杂度较高。因此，我们应当利用问题的特殊性，熟练使用下述短小精悍的Manacher算法。

3.2.1 算法流程

在Manacher算法中，我们添加两个辅助变量mx和p，分别表示已有回文半径覆盖到的最右边界和对应中心的位置。

计算$R[i]$时，我们先给它一个下界，令i关于p的对称点$j = 2p - i$，分以下三种情况讨论（示意图中，上方线段描述了字符串，中间线段描述了$R[p]$，左下方线段描述了$R[j]$，右下方线段描述了$R[i]$的下界）：

- $mx < i$时，只有$R[i] \geq 1$；
- $mx - i > R[j]$时，以第j位为中心的回文子串包含于以第p位为中心的回文子串，由于i和j关于p对称，以第i位为中心的回文子串必然也包含于以第p位为中心的回文子串，故有$R[i] = R[j]$；
- $mx - i \leq R[j]$时，以第j位为中心的回文子串不一定包含于以第p位为中心的回文子串，但由于i和j关于p对称，以第i位为中心的回文子串向右至少会扩展到mx的位置，故有$R[i] \geq mx - i$。

Figure 1: $mx < i$的情况

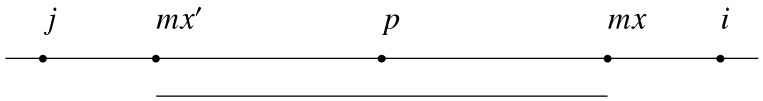


Figure 2: $mx - i > R[j]$的情况

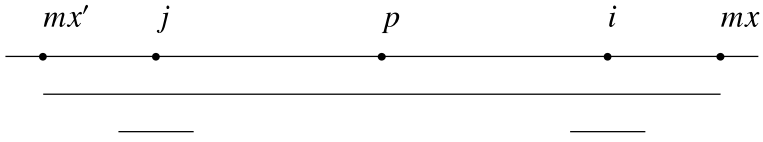
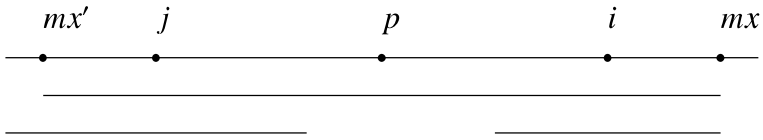


Figure 3: $mx - i \leq R[j]$的情况



对于超过下界的部分，我们只要逐位匹配就可以了。

由于逐位匹配成功必然导致mx右移，而mx右移不超过n次，故Manacher算法的时间复杂度为$O(n)$。

3.2.2 伪代码

MANACHER'S-ALGORITHM(S)

```
1  p ← 0
2  mx ← 0
3  for i ← 1 to n
4       do if mx > i
5             then R[i] ← MIN(R[2p − i], mx − i)
6             else  R[i] ← 1
7          while S[i + R[i]] = S[i − R[i]]
8                do R[i] ← R[i] + 1
9          if i + R[i] > mx
10            then p ← i
11                 mx ← i + R[i]
12 return R
```

4 例题分析

4.1 例题一：拉拉队排练[1]

4.1.1 题目大意

给出一个长度为$n(1 \le n \le 10^6)$的字符串，求前$k(1 \le k \le 10^{12})$长的长度为奇数的回文子串长度的乘积除以19930726的余数或判断长度为奇数的回文子串不足k个。

4.1.2 算法分析

首先，我们使用Manacher算法求出该字符串每一位的回文半径。

显然，回文子串的长度不超过n种，因此我们可以想办法统计出每种长度回文子串的个数$cnt[1..n]$。设以第i位为中心的回文子串的可能长度区间为$[l_i, r_i]$，则我们要实现对$cnt[l_i..r_i]+1$，这个子问题非常经典，差分后每次只要对$cnt[l_i]+1$，对$cnt[r_i+1]-1$，最后再求一遍前缀和就可以了。

下面，我们只要按回文子串长度从大到小依次处理（当然只处理长度为奇数的），每次利用快速幂将相应长度的乘积累乘到答案中，直到长度为奇数的回文子串总个数达到k。

时间复杂度为$O(n \log n)$。

4.2 例题二：Palisection[2]

4.2.1 题目大意

给出一个长度为$n(1 \le n \le 2 \times 10^6)$的字符串，求相交的回文子串对数。

4.2.2 算法分析

首先，我们使用Manacher算法求出该字符串每一位的回文半径，同时求出回文子串的总个数，进而得到回文子串的总对数。

[1]题目来源：2011年国家集训队互测

[2]题目来源：Codeforces Beta Round #17

利用补集转化思想，我们可以将问题转化求不相交的回文子串对数，显然用回文子串的总对数减去不相交的回文子串对数就得到了相交的回文子串对数。

令$S_1[i]$表示以第i位结尾的回文子串个数，$S_2[i]$表示以第$j(1 \le j \le i)$位结尾的回文子串个数和，显然$S_2[i] = S_2[i-1] + S_1[i]$。设当前处理到以第$i$位为中心的回文子串与左侧回文子串的不相交对数，我们要求的就是

$$\sum_{j=i-R[i]}^{i-1} S_2[j]$$

。为了快速得到上式的值，我们再求前缀和$S_3[i] = S_3[i-1] + S_2[i]$，则上式就为$S_3[i-1] - S_3[i-R[i]-1]$。至此，单次处理就能在$O(1)$的时间复杂度内完成。

时间复杂度为$O(n)$。

4.3 小结一

对于例题一和例题二这样与回文子串相关的简单计数问题，我们通常可以在使用Manacher算法求出字符串每一位回文半径的基础上，利用前缀和等常用手段将核心问题在$O(n)$的时间复杂度内解决。

4.4 例题三：最长双回文串[3]

4.4.1 题目大意

给出一个长度为$n(2 \le n \le 10^5)$的字符串，求最长双回文子串T（双回文指可将T分为非空连续两部分X, Y，且X和Y都是回文串）的长度。

4.4.2 算法分析

首先，我们使用Manacher算法求出该字符串每一位的回文半径。

考虑枚举分界点i，我们要使得以第i位为右边界和左边界的回文子串都最长，也就是要分别找到在第i位左右两侧回文半径可覆盖至第i位的离第i位最远的位置。

[3]题目来源：2012年国家集训队清华集训

那么，以找左侧最优位置为例，我们从左往右扫描，同时维护一个当前处理到的右边界。扫描到第i位时，我们只要更新从当前右边界的后一位到$i + R[i]$这段范围里每一位的左侧最优位置，同时更新右边界为$i + R[i]$。

其正确性是显然的，因为如果左右两个位置同时覆盖到一个位置，那么选左边那个必然更优。找右侧点同理。

得到了左侧和右侧的最优位置后，我们也就得到了以第i位为分界点的最长双回文子串长度，取最大值即为最后的答案。

时间复杂度为$O(n)$。

4.5 例题四：双倍回文[4]

4.5.1 题目大意

给出一个长度为$n(1 \le n \le 5 \times 10^5)$的字符串，求最长双倍回文子串$T$（双倍回文指$T$是长度为4的倍数的回文串，且$T$的前一半和后一半也是回文串）的长度。

4.5.2 算法分析

首先，我们使用Manacher算法求出该字符串每一位的回文半径。

考虑从左往右枚举中心i，我们需要找到最小的子中心$j(j < i)$，使得第j位的回文半径能覆盖到第i位，并且第i位到第j位的距离的2倍不能超过$R[i]$。

我们可以很容易得到满足“第i位到第j位的距离的2倍不超过$R[i]$”的j的左边界$k = i - \lfloor \frac{R[i]}{2} \rfloor$。那么，我们要找的就是在第$k$位右侧离第$k$位最近的回文半径能覆盖到第$i$位的位置$j$。

我们维护一个可能的最优位置表，在表中找到第k位右侧离第k位最近的未被删除的位置后，若该位置的回文半径不能覆盖到第i位，就把该位置从表中删除（因为该位置不可能再被右侧的中心所用），继续重复该操作，直到找到的位置的回文半径能覆盖到第i位或已经找到第i位为止。

对于一个表，我们要支持删除操作和查询某个位置右侧最近的未被删除的位置，这个就很容易用并查集来维护了。

对以每一位为中心的最长双倍回文子串长度取最大值即为最后的答案。

[4]**题目来源**：Shanghai Team Selection Contest 2011

时间复杂度为$O(n\alpha(n))$。

4.6 例题五：Tricky and Clever Password[5]

4.6.1 题目大意

一个长度为l（l为奇数）的回文串T可以被加密为$A+T[1..x]+B+T[x+1..l-x]+C+T[l-x+1..l]$（+表示字符串连接，$x$为满足$2x<l$的任意非负整数，$A,B,C$为可以为空的任意字符串）。给出一个长度为$n(1\le n\le 10^5)$的密文$S$，求$T$的最大可能长度。

4.6.2 算法分析

首先，我们使用Manacher算法求出该字符串每一位的回文半径。

我们注意到，$T[x+1..l-x]$也是一个长度为奇数的回文串，且其中心就是T的中心。

考虑枚举中心i，将$T[x+1..l-x]$设为$S[i-R[i]+1..i+R[i]-1]$必然是最优的，这很容易通过讨论B,C是否为空来证明。

接下来，我们要最大化x，也即要求反转后的$S[i+R[i]..n]$在$S[1..i-R[i]]$中的最大匹配长度。令反转后的S为S'，我们可以先用KMP算法预处理出$S_1[i]$表示满足$S[i-k+1..i]=S'[1..k]$的最大的k，再求前缀最大值$S_2[i]=\max\{S_2[i-1],S_1[i]\}$。这样，以第$i$位为中心的$T$的最大可能长度就为$2(R[i]+\min\{S_2[i-R[i]],n-i-R[i]+1\})-1$。

对以每一位为中心的T的最大可能长度取最大值即为最后的答案。

时间复杂度为$O(n)$。

4.7 小结二

对于例题三、例题四和例题五这样与回文子串相关的求最优字符串问题，我们通常可以在使用Manacher算法求出字符串每一位回文半径的基础上，用$O(n)$的时间复杂度枚举所求字符串的中心（或分界点），分析题中的约束条件后结合一些数据结构或是辅助算法来加速单次处理。

[5]**题目来源：**Codeforces Beta Round #30

4.8 例题六：Palindromic Substring[6]

4.8.1 题目大意

给出一个长度为$n(1 \le n \le 10^5)$的小写字母字符串，进行$m(1 \le m \le 20)$次询问，第i次询问给每个字母一个$[0,25]$的权值，求权值第k_i小的回文子串（保证存在）的权值（回文子串的权值指将该回文子串的前一半提取出来，每一位用字母对应的权值替代后形成的二十六进制数除以777777777的余数）。

4.8.2 算法分析

定理1. *一个字符串只有$O(n)$个**本质不同**的回文子串。*

Proof. 我们注意到，在Manacher算法中，只有mx右移时才会产生新的回文子串（否则一定存在对称的回文子串），而mx右移不超过n次，故一个字符串只有$O(n)$个本质不同的回文子串。 □

据此，我们可以用Manacher算法在$O(n)$的时间复杂度内得到所有本质不同的回文子串的位置。

对于每次询问，我们都可以用类似字符串Hash的计算方法来得到每种回文子串的权值。

而为了求出第k小的，我们需要得到每种回文子串在该字符串中出现的次数，这个子问题可以通过构造后缀数组后二分来解决。

接下来，我们只要将每种回文子串的权值从小到大排序，依次累加出现次数直到不小于k，则对应的权值就是答案。

时间复杂度为$O(mn \log n)$。

4.9 例题七：Palindromic Equivalence[7]

4.9.1 题目大意

给出一个长度为$n(1 \le n \le 10^6)$的小写字母字符串S，求和S回文性质相同的字符串T（回文性质相同指$T[i..j]$是回文串当且仅当$S[i..j]$是回文串）的个数。

[6]**题目来源：** Asia Changchun Regional Contest 2012
[7]**题目来源：** 11th POI Training Camp

4.9.2 算法分析

我们发现，S和T的回文性质可以用一些相等和不等关系来表示。对于以第i位为中心的回文子串，显然有$T[i-k] = T[i+k](1 \le k < R[i])$，以及$T[i-R[i]] \neq T[i+R[i]]$。

为了进行统计，我们肯定要得到这些相等和不等关系。

定理2. *一个字符串的回文性质可以用$O(n)$个相等和不等关系来表示。*

Proof. 我们注意到，在Manacher算法中，只有mx右移时才会产生新的相等关系（否则一定存在对称的相等关系），而mx右移不超过n次，故相等关系的数量是$O(n)$的。

由于以每一位为中心都只会产生一个不等关系，故不等关系的数量同样是$O(n)$的。

综上，一个字符串的回文性质可以用$O(n)$个相等和不等关系来表示。 □

据此，我们可以用Manacher算法在$O(n)$的时间复杂度内完成将相等的位置合并只保留最前的一个，并在不等的位置间连边，相当于形成了一个新字符串，边代表不等关系。

下面的问题相当于给出一个无向图，要求用26种颜色给点染色，使得每条边的两个点颜色不同，求方案数。

然而，对于一般的无向图，这个问题是没有有效算法的，因此我们还要进一步挖掘这个图的特殊性。

定理3. *将一个字符串T的相等位置合并只保留最前的一个后得到的新字符串T'，若存在不等关系$T'[i] \neq T'[j](j < i), T'[i] \neq T'[k](k < i)$，则一定有不等关系$T'[j] \neq T'[k]$。*

Proof. 我们仍然从Manacher算法的流程来思考，计算$R[i]$时，令i关于p的对称点$j = 2p - i$，分以下情况讨论：

- $mx < i$时，mx发生右移，相当于向T'末尾添加字符$T[mx]$，使之成为当前活跃字符，并有初始不等关系$T[i - R[i]] \neq T[mx]$；
- $mx - i > R[j]$时，所得的不等关系$T[i - R[i]] \neq T[i + R[i]]$是冗余的（一定存在对称的不等关系），故无需考虑；

- $mx - i \leq R[j]$时，若mx发生右移，则同$mx < i$的情况；若mx不发生右移，则我们使得当前活跃字符又多了一个不等关系$T[i - R[i]] \neq T[mx]$。

设当前活跃字符为$T[i]$，存在不等关系$T[i] \neq T[j](j < i), T[i] \neq T[k](k < i)$，不妨设$k < j$，可得$T[k + 1..i - 1]$和$T[j + 1..i - 1]$均为回文子串，由对称进一步得到$T[j] = T[k + i - j]$，$T[k + 1..k + i - j - 1]$是回文子串。

那么，在Manacher算法中一定会比较$T[k]$与$T[k + i - j]$，又因为$T[j] = T[k + i - j]$，故$T[j]$与$T[k]$要么一定相等，要么一定不等。而一定相等时，$T[k]$与$T[j]$在T'中是同一个位置。

综上，若存在不等关系$T'[i] \neq T'[j](j < i), T'[i] \neq T'[k](k < i)$，则一定有不等关系$T'[j] \neq T'[k]$。 □

得到该性质后，$T'[i]$的可能字母数就是26减去不等关系$T'[i] \neq T'[j](j < i)$的个数，故可以很方便地完成计算。

时间复杂度为$O(n)$。

4.10 小结三

对于例题六和例题七这样与回文子串相关却又很难直接下手的问题，我们往往需要分析问题的瓶颈，利用Manacher算法的流程来证明某种对象的数量级（通常是$O(n)$的），并得到一些有用的性质从而进一步简化问题。

5 总结

对于回文子串问题，我们首先要能熟练掌握求回文半径的基本算法，尤其是深入理解Manacher算法，这对分析回文子串的诸多性质来说是必不可少的。

在此基础上，我们要抓住题中的条件仔细分析，善于利用回文子串的性质来简化问题，并能够结合多种算法和数据结构解决问题。

致谢

- 感谢CCF提供的学习和交流平台。
- 感谢学校的支持和曹文老师的栽培。

- 感谢许多同学在论文完成过程中对我的帮助。
- 最后，感谢父母的养育之恩。

参考文献

[1] Maxime Crochemore, Wojciech Rytter Mokhtar, “Text Algorithms”, Oxford University Press.

[2] 罗穗骞,《后缀数组——处理字符串的有力工具》, 2009年国家集训队论文。