串串 解题报告

福州第三中学 钟子谦

1 题意简述

定义一个串 x 为双回文串当且仅当存在非空回文串 a,b 满足 $x=ab$。

求对于一个给定的串 s，有多少个本质不同的子串为双回文串。

2 数据范围

对于所有数据，$|s|\le 5\times 10^5$ ，s 由小写字母组成。

子任务编号	$\|s\|\le$	其他约定
1	300	-
2	1500	-
3	40000	-
4	150000	-
5	500000	s 每一位都在 $\{a,b\}$ 中等概率生成
6	500000	-

3 约定与记号

记字符串 s 的长度为 $|s|$，字符串下标从 0 开始，即 $s=s_0s_1...s_{|s|-1}$。

记 $s[l,r]$ 为 $s_ls_{l+1}...s_r$，若 $l>r$ 则为空串。记 ab 为 a、b 两个串拼接而成的结果，记 x^a 为 a 个 x 串拼接而成的结果。记 x^R 为将 x 的各个字符逆序排列所得的串，例如 $abc^R=cba$。

我们称串 x 有周期 t 当且仅当 $\forall i\in[0,|x|-t)$，$x_i=x_{i+t}$，称串 x 有整周期 t 当且仅当 $|x|$ 是 t 的倍数且 x 有周期 t。

我们称一个串 x 为整周期串当且仅当 x 存在整周期 $t<|x|$，称一个串 x 为本原串当且仅当 x 不是整周期串。

我们称一个串 x 为平方串当且仅当 x 存在整周期 $|x|/2$（$|x|$ 为偶数），称一个串 x 为本原平方串当且仅当 x 为平方串且 $x[0,|x|/2-1]$ 为本原串。称一个串 x 的本原根为 x 的一个前缀，它的长度为 x 的最小整周期。

我们称一个串 x 为双回文串当且仅当存在非空回文串 a,b 满足 $x=ab$，称任意这样的 (a,b) 为 x 的一组双回文拆分。

我们称一个串 x 为弱双回文串当且仅当存在可空回文串 a 和非空回文串 b 满足 $x=ab$，称任意这样的 (a,b) 为 x 的一组弱双回文拆分。

4 部分分算法

4.1 算法一

枚举所有子串 x，我们可以 $O(|x|)$ 判断 x 是否是回文串。

接下来枚举所有子串 x 判断是否是双回文串。因为我们已经预处理出了每个子串是否是回文串，枚举 $a\in[1,|x|)$，然后用之前处理好的信息判断 x 长度为 a 和 $|x|-a$ 的前后缀是否是回文串，将找到的双回文串排序去重即可。

时间复杂度 $O(|s|^3)$，可以通过第一个子任务。

如果采用 manacher 等算法判断是否为回文串，并且枚举 a 时使用压位技巧（例如使用 std::bitset），可以将时间复杂度降到 $O(|s|^3/w)$，其中 w 是计算机的字长，可以通过第二个子任务。

4.2 算法二

对于 s 每一位都在 $\{a,b\}$ 中等概率生成的子任务，一个长度为 k 的 $\{a,b\}$ 随机串为回文串的概率为 $2^{-\lfloor k/2\rfloor}$（确定了左半边之后右半边每一个字符都要与左半边对应字符匹配），所以可以先用 manacher 或 hash 求出所有回文串，对于 $s=ab$ 枚举 a 的右端点（即 b 的左端点 -1）遍历所有以它为右端点的回文串，再遍历所有以 b 左端点为左端点的回文串，hash 后去重。由于以某个点开头的回文串期望不超过 $\sum_{i=1}^{\infty}2^{-\lfloor i/2\rfloor}=3$ 个，所以期望复杂度为 $O(|s|\log(|s|))$ 或 $O(|s|)$（使用哈希表去重），可以通过第五个子任务。

4.3 算法三

为了获得低于 $O(|s|^{3-\epsilon})$ 的复杂度，我们需要一些双回文串的性质。

引理 4.1. *若 $s = ab$，a 和 s 为回文串，则 $|b|$ 为 s 的周期。*

Proof. $\forall i \in [0, |a| = |s| - |b|)$，$s_i = s_{|a|-1-i} = s_{|s|-1-(|a|-1-i)} = s_{i+|b|}$。

□

引理 4.2. *如果一个字符串 $s = x_1x_2 = y_1y_2 = z_1z_2$, $|x_1| < |y_1| < |z_1|$，x_2, y_1, y_2, z_1 均为非空回文串，那么 x_1 和 z_2 也为回文串。*

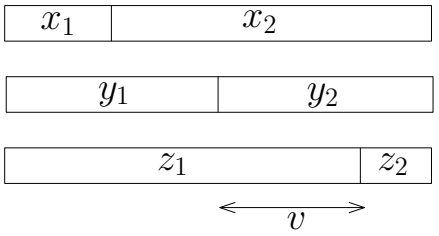
Proof. 设 $z_1 = y_1v$。

因为 v^R 是 y_2 的后缀，即 x_2 的后缀，所以 v 是 x_2 的前缀。那么 x_1v 就是 z_1 的前缀。

因为 y_1 是回文串 z_1 的回文前缀且 $z_1 = y_1v$，$|v|$ 是 z_1 的周期，那么 $|v|$ 也是 x_1v 的周期，所以 x_1 是 v^∞ 的后缀。

因为 v 是 z_1 的后缀，那么 v^R 是 z_1 的前缀，又 $|v|$ 是 z_1 的周期，那么 x_1 是 $(v^R)^\infty$ 的前缀。

由以上两点，可知 x_1 是回文串。类似地，z_2 也是回文串。



□

引理 4.3. *如果一个字符串 s 是双回文串，那么存在一种双回文拆分 $s = ab$ 使得 a 是 s 的最长回文前缀，或者 b 是 s 的最长回文后缀。*

Proof. 假设不成立，设 (c, d) 为 s 的某一双回文串拆分，设 $s = ab = cd = ef$，其中 b 是 s 的最长回文后缀，e 是 s 的最长回文前缀。由上述引理，a 和 f 也为回文串，所以 (a, b) 和 (e, f) 也为双回文串拆分。 □

由上一引理，判断一个串是否为双回文串，我们只需要考虑最长回文前缀和最长回文后缀。使用 manacher 或 hash 求出每个区间是否是回文串，使用适当的顺序枚举区间，即可求出每个子串的最长回文前缀和最长回文后缀并判断，最后排序去重。复杂度 $O(|s|^2 \log(|s|))$，可以通过前两个子任务。

5 标准解法

关于双回文串，我们还需要挖掘更多的性质。为了辅助证明，我们先引入一个引理。

引理 5.1. *若 p 和 q 均为字符串 s 的周期，$p+q \le |s|$，那么 $gcd(p,q)$ 也为 s 的周期。*

Proof. 设 $p < q, d = q - p$。

若 $|s| - q \le i < |s| - d$，$s_i = s_{i-p} = s_{i-p+q} = s_{i+d}$。

若 $0 \le i < |s| - q$，$s_i = s_{i+p} = s_{i+q-p} = s_{i+d}$。

那么 $d = q - p$ 也为 s 的周期。由欧几里得算法可知 $gcd(p,q)$ 为 s 的周期。 □

引理 5.2. *若 s 有两种（或以上）不同的弱双回文拆分，则 s 是一个整周期串。*

Proof. 设 $s = p_1q_1 = p_2q_2$（$|p_1| < |p_2|, |q_2| \ne 0$，$p_1, q_1, p_2, q_2$ 为回文串），$t = |p_2| - |p_1|$，我们证明 $gcd(t, |s|)$ 为 s 的一个整周期。

由于 p_1 是回文串 p_2 的回文前缀，$t = |p_2| - |p_1|$ 为 p_2 的周期，类似地 t 为 q_1 的周期。由于它们有长为 t 的公共部分，所以 t 也是 s 的周期。

$\forall i \in [0, t)$，$s_i = s_{|p_2|-1-i} = s_{|s|-1+|p_1|-(|p_2|-1-i)} = s_{i+|s|-t}$，所以 $|s| - t$ 也是 s 的周期。

由上述引理，$gcd(t, |s| - t) = gcd(t, |s|)$ 为 s 的一个（整）周期。 □

这个性质揭示了（弱）双回文串和整周期串的联系。事实上，我们还可以发现一个更强的性质。

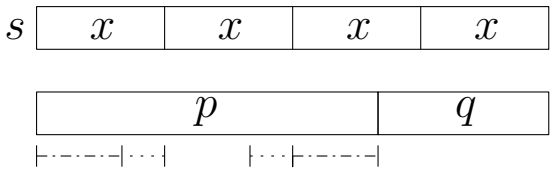
引理 5.3. *若 s 为弱双回文串，设它的最小整周期为 t，则 s 恰有 $|s|/t$ 种不同的弱双回文拆分。*

Proof. 若 $t = |s|$ 由上一引理显然成立，以下假设 $|s|/t \geq 2$。

设 $x = s[0, t-1]$，x 显然不为整周期串（否则 x 的最小整周期即为 s 更小的整周期），故由上一引理有不超过一种弱双回文拆分。

设 $s = pq$ 为某一弱双回文拆分，那么 $max(|p|, |q|) \geq t$。若 $|p| \geq t$，$x[0, |p| \bmod t-1] = p[0, |p| \bmod t-1] = p^R[0, |p| \bmod t-1] = x^R[0, |p| \bmod t-1]$，所以 $x[0, |p| \bmod t-1]$ 为回文串，类似地 $x[|p| \bmod t, t-1]$ 也为回文串。$|q| \geq t$ 时类似也可推出 $x[0, |p| \bmod t-1]$ 和 $x[|p| \bmod t, t-1]$ 均为回文串。所以 $x[0, |p| \bmod t-1]$ 和 $x[|p| \bmod t, t-1]$ 为 x 的一种弱双回文拆分。

由于 s 存在弱双回文拆分，x 也存在弱双回文拆分。设 $x[0, g-1]$ 和 $x[g, t-1]$ 为 x 的唯一一种弱双回文拆分，则 s 的所有弱双回文拆分 (p, q) 均需满足 $|p| \bmod t = g$，那么因为 $0 \leq |p|/t < |s|/t$ 至多有 $|s|/t$ 种不同的弱双回文拆分。同时不难发现这 $|s|/t$ 种弱双回文拆分均可取到。



□

有了以上性质，我们不难得到一个初步思路：尝试计数所有子串本质不同的双回文串拆分（两个双回文串拆分 (a_1, b_1) (a_2, b_2) 本质不同当且仅当 $a_1 \neq a_2$ 或 $b_1 \neq b_2$）。如果一个双回文串有多种不同双回文串拆分，那么它一定是一个整周期串，并且根据最小整周期的循环次数我们还可以算出它的不同双回文串拆分个数（双回文串拆分个数显然是弱双回文拆分个数或弱双回文拆分个数减一，需要减一当且仅当最小整周期串是回文串），那么我们可以尝试找到所有不同的整周期子串并且扣掉重复数的。

5.1 计数子串本质不同的双回文串拆分

考虑算法二的思路，对于 s 的一个子串 $s[l, r]$ 的一种双回文串拆分 $(s[l, m], s[m+1, r])$，我们枚举 m，那么就是要将 s 中任意以 s_m 结尾的回文子串和以 s_{m+1} 开头的回文子串计入拆分数量。

考虑使用回文树进行维护。回文树的 fail 指针结构是一种树形结构，有 $O(|s|)$ 个节点，每个节点表示一个回文串，一个节点的父节点表示的回文串为这

个节点表示的回文串（除本身外）的最长回文前（后）缀。此外，我们还可以方便地获取到字符串每个前缀的最长回文后缀所对应的节点。

我们对原串和反串分别建出回文树[1]，那么问题就被转换为一个数据结构问题：有两棵大小 $O(|s|)$ 的树，$O(|s|)$ 次操作，每次给定第一棵树上的某个点 x 和第二棵树上某个点 y，将所有 x 在第一棵树上的祖先和所有 y 在第二棵树上的祖先形成的二维点标记，所有操作结束后问有多少个被标记过的二维点。

我们枚举点的第一维，那么就相当于第一棵树每个点上有一些第二棵树上到根的链，问第一棵树每个点子树里的链并集大小。

5.1.1 $O(|s|\log^3(|s|))$ 的做法

我们可以在第一棵树上使用树上启发式合并（dsu on tree）[2]，使用树链剖分和支持区间加、查询最小值和出现次数的线段树维护第二棵树，即在加入和删除链的时候将对应点点权 $+1$，查询点权为零的点个数。时间复杂度 $O(|s|\log^3(|s|))$。

5.1.2 $O(|s|\log(|s|))$ 或 $O(|s|\log^2(|s|))$ 的做法

先考虑如何计算一些第二棵树上到根的链的并集大小。可以发现将所有链按下端点的 dfs 序排序，把所有链长度之和减去相邻两条链交集长度之和即为并集大小。这是因为到 x,y 两点到根的链的交必然是 $lca(x,y)$ 到根的链，那么如果将这些链一条一条加入并集，原先在链并上的点一定是这条链的下端点与原来某条链的下端点的 lca 中，深度最深的一个到根的链。考虑欧拉序求 lca 的算法，两点的 lca 为欧拉序中两点第一次出现位置中间的点中深度最浅的点。欧拉序第一次出现位置顺序即 dfs 序，所以我们只要考虑前一条链，因为与更前面的链的 lca 只会更浅。

回到原问题，在第一棵树上使用树上启发式合并（dsu on tree），用平衡树（例如 std::set）维护在集合中的链下端点 dfs 序从小到大的顺序，每次插入新链，查询相邻处的 lca 并维护答案，即可做到 $O(|s|\log^2(|s|))$ 的时间复杂度。

实际上我们可以使用线段树合并。具体地，用动态开点线段树维护每个子树在集合中的链下端点 dfs 序从小到大的顺序，将两棵子树合并时，直接使用

[1]实际上也可以只建出原串的回文树，因为原串和反串回文树 fail 指针结构是一样的

[2]即先递归轻子树，回溯时清空，再递归重子树，回溯时保留，接着加入轻子树中的点

线段树合并（即只合并两棵线段树的公共节点）。在线段树每个节点上维护下属叶子节点 dfs 序的 min 和 max，那么递归时就可以得到一些极长的来自同一棵树的段，记录每一个段的 min 和 max，在边界处重新计算贡献即可。由于对于一个修改只会新开 $O(\log(|s|))$ 个线段树节点并且合并的复杂度与回收的节点数同阶，所以线段树合并一部分的复杂度是 $O(|s|\log(|s|))$ 的。如果查询 lca 的复杂度是 $O(1)$ 的，这个做法整体的复杂度就也是 $O(|s|\log(|s|))$ 的。

5.2 求出整周期子串并统计答案

注意到一个整周期串必然有一个前缀为本原平方串，我们不妨从本原平方串着手研究。我们首先估计本原平方串的个数。

引理 5.4. *若非空字符串 a,b,s 满足 $s=ab=ba$，则 s 为整周期串。*

若非空字符串 p,x 满足 px 为 xx 的前缀且 $|p|<|x|$，则 x 为整周期串。

Proof. $\forall i\in[0,|b|)$，$s_i=b_i=s_{i+|a|}$。$\forall i\in[0,|a|)$，$s_i=a_i=s_{i+|b|}$。

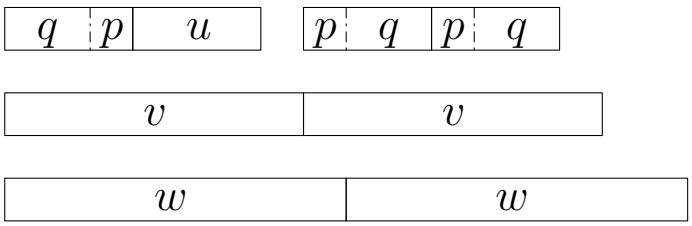
所以 $|a|$、$|b|$ 均为 s 的周期，由引理 5.1，$gcd(|a|,|b|)=gcd(|a|,|s|)$ 为 s 的一个整周期。

同理可得 $gcd(|p|,|x|)$ 为 x 的一个周期。 □

引理 5.5. *若 u,v,w 为三个不同非空串，u 为本原串，uu 是 vv 的前缀，vv 是 ww 的前缀，那么 $|u|+|v|\le|w|$。*

Proof. 假设命题不成立，则 $|v|<|w|<|vu|$。

若 $2|u|\le|v|$，即 uu 为 v 的前缀。设 $w=vp$，由于 w 是 vu 的前缀，可设 $wq=vu$，即 $pq=u$。注意到（第二个）v 为 pw 的前缀，那么 uu 为 pw 的前缀，所以 qp 为 w 的前缀，而 $u=pq$ 也为 w 的前缀，所以 $u=pq=qp$，所以由上述引理 u 不是本原串。



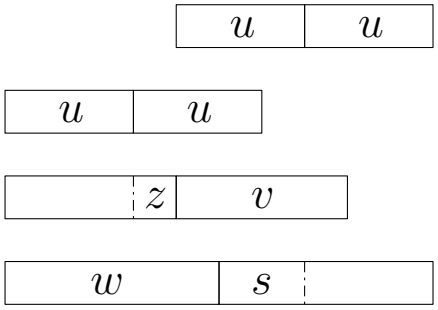
否则 $|v| < 2|u|$。设 $v = uz$，z 的本原根为 x。因为 u 是 zu 的前缀，$|z|$ 是 u 的周期，则 $|x|$ 也是 u 的周期。那么可设 $z = x^{\beta}, u = x^{\alpha}r$（$\alpha \geq \beta \geq 1$），$r$ 为 x 的一个非空前缀。

设 $ws = vu$，那么 s 为 u 的非空后缀，sz 是 w 的前缀，此外 $v = uz$ 也是 w 的前缀。所以 s 也是 u 的前缀。因为 s 为 u 的前缀，可设 $s = x^{\gamma}t$（$0 \leq |t| < |x|, \gamma \geq 0$）。

若 $|s| \geq |x|$，由于 xt 为 xxr 的后缀，若 $|t| > |r|$，设 $t = ar$，xa 为 xx 的后缀，则 x 为整周期串，矛盾。若 $|t| < |r|$，设 $r = at$，x 为 xxa 的后缀，由于 a 为 x 的前缀，那么 $ax = xa$，那么 x 亦为整周期串，矛盾。所以只能 $t = r$，那么 $\alpha > \gamma \geq 1$。那么 $sx = x^{\gamma}rx$，$u = x^{\alpha}r$，则 sx 为 u 的前缀，即 rx 为 $x^{\alpha-\gamma}r$ 的前缀，所以 rx 为 $x^{\alpha-\gamma+1}$ 的前缀，rx 为 xx 的前缀，x 为整周期串，矛盾。

若 $|s| < |x|, |s| \leq |x^{\alpha-1}r|$，那么 sx 为 $u = x^{\alpha}r$ 的前缀，sx 为 xx 的前缀，矛盾。

若 $|x^{\alpha-1}r| < |s| < |x|$，那么 $\alpha = \beta = 1, \gamma = 0, |r| < |s|$。那么 $u = xr, v = xrx, s = t$，设 $pt = xr$，则 $w = xrxp$，那么 $vv = xrxxrx$ 为 $ww = xrxpxrxp$ 的前缀，所以 pxr 为 xrx 的前缀，那么 pxr 为 $xrxr$ 的前缀，即 pu 为 uu 的前缀，所以 u 为整周期串，矛盾。



□

引理 5.6. *一个长度为 n 的串为本原平方串的前缀数量为 $O(\log(n))$ 个。*

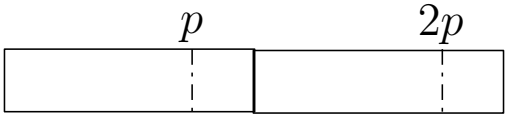
Proof. 设长度第 i 小的本原平方串前缀长度为 a_i，则由上一引理 $a_i \geq a_{i-1} + a_{i-2}$（$i \geq 3$），所以 $a_i \geq 2F_{i+1}$，其中 F 为斐波那契数列，所以 $a_i \sim (\frac{\sqrt{5}-1}{2})^i$。 □

所以一个串的本原平方串子串个数为 $O(n\log(n))$。

先考虑如何求出一个串的所有平方串子串。[3]我们枚举它的长度的一半 p，

[3]可以参见 NOI2016《优秀的拆分》

将为 p 的倍数的下标称为关键点，那么一个长度为 $2p$ 的平方串一定会经过两个相邻的关键点，我们只要将相邻两个关键点求出以它们为左端点的最长公共后缀和以它们为右端点的最长公共前缀，即可找到一组开头连续的平方串。两个后缀的最长公共前缀和前缀的最长公共后缀可以使用后缀数组 $O(1)$ 求出，所以复杂度由调和级数即为 $O(n\log(n))$。



找到了所有的平方串之后，我们称找到的长度相同的开头连续的极长一段为一组。注意到如果一个平方串不是本原平方串，那么同一组的串也都不是本原平方串。这是因为对于平方串 $s[a,a+2p-1]$，如果 $s[a,a+p-1]$ 有整周期 t，那么由于 $s_a=s_{a+p}$，所以 $s[a+1,a+p]$ 也有整周期 t。所以如果 $[l,r]$ 开头的长度为 $2p$ 的一组非本原平方串，那么一定存在 $q|p,q<p$，使得有 $[l,r+2p-2q]$ 开头的长度为 $2q$ 的本原平方串，反之亦然。所以我们只需要从小到大枚举 p，对于每一个长度为 p 的组 $[l,r]$ 判断 $s[l,l+2p-1]$ 是否是本原平方串。如果不是则跳过这组，否则枚举满足 $l\le r+2p-2q$ 的 p 的倍数 q，将 $s[l,l+2q-1]$ 标记为非本原平方串即可。由于所有本原平方串组 $r-l+1$ 之和即为本原平方串子串子串个数，所以复杂度为 $O(n\log(n))$。

求出了所有的本原平方串组后，注意到一个开头在 $[l,r]$ 之间，长度为 $2p$ 的本原平方串组满足 $s_i=s_{i+p}\ (i\in[l,r+p-1])$，$s_{r+p}\neq s_{r+2p}$，而一个子串 $s[a,b]$ 有周期 p 即 $s_i=s_{i+p}\ (i\in[a,b-p])$。所以对一个开头为 $i\in[l,r]$ ，周期为 p 的整周期串，它的右端点 j 必须满足 $j-p<r+p,p|j-i+1$，我们只需要枚举 i，求出最大的合法 j，将本原串 $s[i,i+p-1]$ 连续重复的最大次数用 $(j-i+1)/p$ 更新。求出 $s[i,i+p-1]$ 的 hash 值，使用哈希表更新即可。这部分的期望复杂度也为 $O(n\log(n))$。

为了求出答案，我们还需要对于每个 $s[i,i+p-1]$ 判断是否为回文串和弱双回文串。由引理 4.3，判断一个串是否为双回文串，我们只需要考虑最长回文前缀和最长回文后缀。在回文树上求一个区间 $[l,r]$ 的最长回文后缀，只需先找到 r 这一前缀的最长回文后缀对应节点，然后往 fail 指针跳直到对应回文串长度 $\le r-l+1$。优化这一过程可以在回文树上维护倍增数组（binary lifting），即对于每个点 i 维护 $up_{i,j}$ 表示 i 往 fail 指针跳 2^j 步后的节点，设 i 为 r 这一前缀

的最长回文后缀对应节点，如果它的长度 $> r-l+1$，询问时从大到小枚举 j，若 $up_{i,j}$ 的长度 $> r-l+1$ 就跳，最后再多跳一步。最长回文前缀只需对反串进行这一过程。判断回文串可以用 manacher 或 hash，当然也可以直接沿用这一倍增过程。判断单个串的复杂度为 $O(\log(n))$。由于有 $O(n\log(n))$ 个本原回文串子串，所以复杂度是 $O(n\log^2(n))$ 的。

事实上，我们可以发现一个串的本质不同的本原平方串子串个数并不多。

引理 5.7. 一个串 s 本质不同的本原平方串子串个数不超过 $2|s|$。[4]

Proof. 记 a_i 为 $s[i,n-1]$ 的前缀有多少个本原平方串不为任意一个 $j>i$ 的 $s[j,n-1]$ 的前缀。以下证明 $a_i \le 2$。

若 $a_i > 3$，设 $s[i,n-1]$ 有三个本原平方串前缀 uu, vv, ww（$|u|<|v|<|w|$）。若 $|w| \ge 2|u|$，则 uu 为 w 的前缀，所以在 $i+|w|$ 位置处再次出现。否则，$|w| < 2|u| < |u|+|v|$，与引理 5.5 矛盾。 □

所以我们只判断本质不同的 $s[i,i+p-1]$，复杂度就是 $O(n\log(n))$ 的。

综合以上两节，我们得到了一个 $O(n\log(n))$ 的解法。

6 总结

本题的难度较高，使用标准解法通过需要对字符串组合性质有着较深入的了解，并且也要对后缀数组和回文树有一定的了解。本题放置部分分较为困难，命制此题的主要目的即是希望普及一些字符串组合性质。

由于考场策略等因素，预计有 90% 左右的选手能获得 10 分或以上，70% 左右的选手能获得 20 分或以上，50% 左右的选手能获得 30 分或以上，30% 左右的选手能获得 40 分或以上，15% 左右的选手能获得 70 分或以上，5% 以下的选手能获得满分。

[4]事实上可以证明一个更强的结论：一个串 s 本质不同的平方串子串个数不超过 $2|s|$，可参见参考文献 [5]

参考文献

[1] Lyndon, Roger C., and Marcel-Paul Schützenberger. "The equation $a^M = b^N c^P$ in a free group."

[2] Galil, Zvi, and Joel Seiferas. "A Linear-Time On-Line Recognition Algorithm for"Palstar"."

[3] Kemp, R. "On the number of words in the language $\{w \in \Sigma^* | w = w^R\}^2$"

[4] Rubinchik, Mikhail, and Arseny M. Shur. "EERTREE: an efficient data structure for processing palindromes in strings."

[5] Fraenkel, Aviezri S., and Jamie Simpson. "How many squares can a string contain?."

[6] Lothaire, Monsieur. *Algebraic combinatorics on words*.