回文串 解题报告

长沙市一中 张天扬

1 试题来源

APIO2014 第一题。

http://www.lydsy.com/JudgeOnline/problem.php?id=3676

2 试题大意

给定一个字符串S。定义S的一个字串T的“出现价值”为T在S中的出现次数乘以T的长度。

求所有S的回文字串中最大的“出现价值”。

$|S| \leqslant 3 \times 10^5$

3 一个暴力算法

首先，有一个经典结论：一个字符串本质不同的回文串个数是$O(n)$的。

那么我们考虑使用manacher算法直接找出所有本质不同的回文串。

然后我们只需要算出每一个回文串的出现次数即可。这是一个经典问题，使用后缀数组和RMQ即可解决。

复杂度$O(n \log n)$。实际比赛时，有许多人使用这个算法卡常数通过本题。

4 命题人给出的标准算法

和上一个算法一样，我们同样考虑如何算出所有回文串的出现次数。

我们把每一个回文串看做一个节点，并令这个回文串删掉第一个和最后一个字符构成的串（一定是回文的）为它的父节点。那么就会构成一棵树。

我们只考虑每一个极长回文串（如果左右都延长1就不再是回文串）的出现次数。这是容易通过manacher算法统计的。之后，每一个回文串的出现次数就等于它的子树和。将整棵树从下往上递推一遍就可以得出每一个回文串的出现次数了。

注意到我们对于一个极长回文串需要快速找到它对应的节点，这需要使用hash来实现。

我们认为hash是$O(1)$的。因此算法复杂度为$O(n)$。

5 一个使用后缀自动机的算法

我们构建出原串的后缀自动机，并计算出每个点right集合的大小以及right集合中的最大值（分别记为*rcnt*和*rmax*）。

之后，我们将原串的反串放在自动机上运行。不妨设当前运行到了原串的第i个字符（即反串的第$n-i+1$个字符）。设当前在的节点为*now*。当前存在于后缀自动机上的串长度为l。

我们发现：如果有$i \leqslant rmax_{now} < i+l$，那么$[i, rmax_{now}]$这一个字串是一个回文字串。我们分两种情况考虑它：

1.如果这个回文字串位于*now*这个节点上，那么我们可以直接用它的长度乘以$rcnt_{now}$来更新答案。

2.如果这个回文字串位于*now*的祖先节点上，注意到我们需要沿父指针走直到找到它所在的节点。但是我们不妨先不考虑这种情况。

接下来，还有另一个可能性：我们当前考虑的长度是l，有可能会导致$rmax_{now} < i$。但考虑*now*的某一个祖先节点fa，如果$i \leqslant rmax_{fa} < i + len_{fa}$，那么我们同样可以用它来更新答案。但是如果每次都沿父指针暴力走一遍，显然复杂度会退化。

实际上，有一个结论：每一个节点在沿父指针走时只需要考虑第一次！这是因为，我们每个点记录的是right集合的最大值，当我们第一次从子节点沿父指针走到这个节点的时候，i一定是最大的（和之后再到这个节点相比）。那么如果这一个节点在第一次没能更新成功，在之后一定都不会更新成功！而从子节点走上来的时候，串的长度是不变的（都是这个节点的len值），那么对答案的更新必然是不变的。因此，我们只需要记录一下每一个节点是否从子节点到过，如果到过就直接不作考虑即可。

对这种情况的考虑同时也包含了上面的第二种情况。因此，这个算法是正确的。

整个算法的复杂度是$O(n)$的。由于常数的原因，相比起使用hash的算法要快上不少。

6 一个奇怪的算法

网上的一篇博文（http://victorwonder.blog.uoj.ac/blog/146）中提到，有一种叫做回文树的数据结构能够用于解决本题。其速度非常快，且代码简短。但由于其用处不广泛，在此只作提及。有兴趣的话可以自行浏览。