Manacher 算法

描述

给定一个长度为 n 的字符串 s，请找到所有对 (i,j) 使得子串 $s[i\ldots j]$ 为一个回文串。当 $t=t_{rev}$ 时，字符串 t 是一个回文串（t_{rev} 是 t 的反转字符串）。

更进一步的描述

显然在最坏情况下可能有 $O(n^2)$ 个回文串，因此似乎一眼看过去该问题并没有线性算法。

但是关于回文串的信息可用 **一种更紧凑的方式** 表达：对于每个位置 $i=0\ldots n-1$，我们找出值 $d_1[i]$ 和 $d_2[i]$（二者分别表示以位置 i 为中心的长度为奇数和长度为偶数的回文串个数。

举例来说，字符串 $s=\mathtt{abababc}$ 以 $s[3]=b$ 为中心有三个奇数长度的回文串，也即 $d_1[3]=3$： $$ a\ \overbrace{b\ a\ \underset{s_3}{b}\ a\ b}^{d_1[3]=3}\ c $$ 字符串 $s=\mathtt{cbaabd}$ 以 $s[3]=a$ 为中心有两个偶数长度的回文串，也即 $d_2[3]=2$： $$ c\ \overbrace{b\ a\ \underset{s_3}{a}\ b}^{d_2[3]=2}\ d $$ 因此关键思路是，如果以某个位置 i 为中心，我们有一个长度为 l 的回文串，那么我们有以 i 为中心的长度为 $l-2$、$l-4$ 等等的回文串。所以 $d_1[i]$ 和 $d_2[i]$ 两个数组已经足够表示字符串中所有子回文串的信息。

一个令人惊讶的事实是，存在一个复杂度为线性并且足够简单的算法计算上述两个“回文性质数组” $d_1[]$ 和 $d_2[]$。在这篇文章中我们将详细地描述该算法。

解法

总的来说，该问题具有多种解法：应用字符串哈希，该问题可在 $O(n\log n)$ 时间内解决，而使用后缀数组和快速 LCA 该问题可在 $O(n)$ 时间内解决。

但是这里描述的算法 **压倒性** 地简单，并且在时间和空间复杂度上具有更小的常数。该算法由 **Glenn K. Manacher** 在 1975 年提出。

朴素算法

为了避免在之后的叙述中出现歧义，这里我们指出什么是“朴素算法”。

该算法通过下述方式工作，对每个中心位置 i，在比较一对对应字符后，只要可能，该算法便尝试将答案加 1。

该算法是比较慢的：它只能在 $O(n^2)$ 的时间内计算答案。

该朴素算法的实现如下：

```
vector<int> d1(n), d2(n);
```

```
for (int i = 0; i < n; i++) {
  d1[i] = 1;
  while (0 <= i - d1[i] && i + d1[i] < n && s[i - d1[i]] == s[i + d1[i]]) {
    d1[i]++;
  }

  d2[i] = 0;
  while (0 <= i - d2[i] - 1 && i + d2[i] < n &&
         s[i - d2[i] - 1] == s[i + d2[i]]) {
    d2[i]++;
  }
}
```