Manacher算法

主讲人：邓哲也

回文串

对于一个长度为 n 的字符串 str，如果它正着读和反着读一样，即 str[i] = str[n - i + 1] (1 <= i < n - i + 1)

如 aba，acbbca 就是回文串，abc，abab 就不是回文串。

Manacher 算法

Manacher 算法可以求出以每个位置为中心，向两边能扩展的最长回文子串长度 p[i]，它的时间复杂度是 O(n) 的。

注意到回文子串的长度可能是偶数， 如 abba，中心不是某个字符（中心是两个 b 之间的空隙），所以先要在相邻的字符中插入一个标识符，例如 #

这样例如 #a#b#b#a# 的中心就是 # 了。

Manacher 算法

我们用 abbabcba 来举例。

先插入 # 得到 #a#b#b#a#b#c#b#a#。

然后用 Manacher 可以得到如下的 p 数组。

对于每个 p[i]，一定有 str[i + j] == str[i - j] (1 ⩽ j < p[i])

str	#	a	#	b	#	b	#	a	#	b	#	c	#	b	#	a	#
p[i]	1	2	1	2	5	2	1	4	1	2	1	6	1	2	1	2	1

Manacher 算法

类比 Z 算法，我们也维护一个 mx 和 id，表示对于当前计算的所有 i，i + p[i] 的最大值是 mx，mx 对应的 i 记为 id。

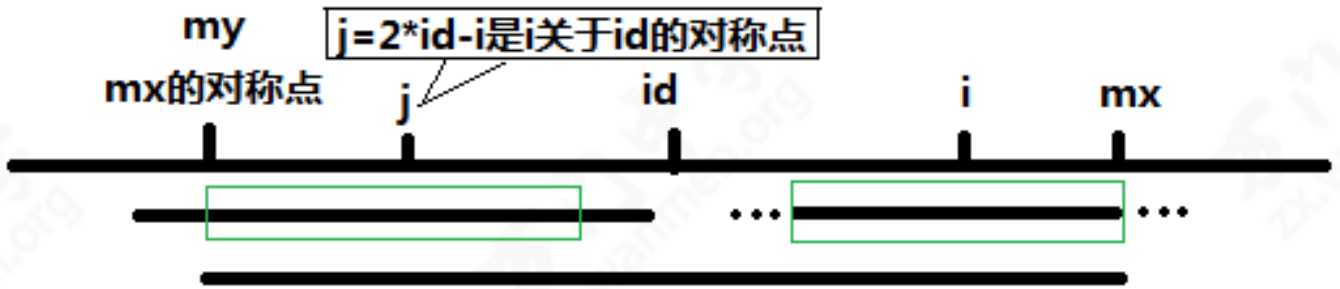
当你现在开始计算 p[i] 时，默认 p[1..i-1] 都已经算出。

如果 mx > i，那么 p[i] >= min(p[2 * id - i], mx - i)

Manacher 算法

当 mx - i > p[j]，那么 p[i] = p[j]

否则 p[i] = mx - i，且需要进一步判断



Manacher 算法代码实现

```
int id, mx = 0;
for (int i = 1;i < n;i ++){
    if (mx > i) p[i] = min(p[2 * id - i], mx - i);
    else p[i] = 1;
    while (str[i + p[i]] == str[i - p[i]]) p[i] ++;
    if (i + p[i] > mx) mx = i + p[i], id = i;
}
```

Manacher 算法

Manacher 算法的时间复杂度是 O(n) 的。

考虑 mx 的移动。

mx 最多从 0 移到 n。

而一次字符串比较会让 mx 右移一次。

因此最多只会比较 n 次字符相等。

时间复杂度 O(n)

下节课再见