参考博客：

<http://blog.csdn.net/dextrad_ihacker/article/details/51121731>

<http://blog.csdn.net/ggggiqnypgjg/article/details/6645824/>

<https://www.zhihu.com/question/30226229>

一.首先明确Manacher算法是干嘛的

这个算法是用来求一个字符串中最长的回文子串。

什么是回文串？

就是对一个字符串正向读和反向读是一样的，例如ababa

我们使用枚举回文串中点的暴力方法来求最长的回文子串，复杂度是0(n^2)，而我们使用Manacher算法在0(n)的时间内就可以求出来。

二.枚举回文子串中点的暴力方法

首先我们考虑一下枚举回文子串中点的暴力方法，用一个数组len[i]记录以i为中点，回文串最右端p到i的长度p-i+1，这里我们要对回文串长度的奇偶性进行考虑。

1.为偶数：while(st[i] == st[i+1]) i--;

2.为奇数：while(st[i-1] == st[i+1]) i++;中点为i;

然后取两个最长回文子串的最大值。

是不是很麻烦。

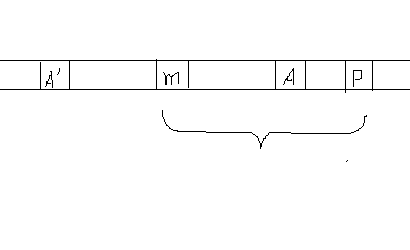
这里我们可以通过插入特殊字符来实现只计算一次。

例如：#A#B#C#

这里我们插入len(原字符串长度)+1个’#’,对于任一个长度为slen的回文子串st，都扩充为一个2\*slen+1长度的奇回文子串(因为插入的都是’#’,所以不影响原串对称性)，这样我们只要计算一次奇回文串的长度即可。最大长度为max{len[i]-1}，想一下就知道了吧。

三.Manacher算法实现

同样像KMP算法一样考虑，在上述的暴力算法中，当我们枚举某个回文子串中点时，前面的回文串信息我们都是知道的。



如图：

我们现在求以st[A]为中点的回文串长度，len[i]和上面定义一样。其中m为已知的某个回文串中点，p为其所能延伸到的最右边的位置。

St[A’]为st[A]关于m对称的对称点。

这里我们分两种情况讨论：

1.st[P]在st[A]的右边：

若len[m]-A > len[A’]，即m的回文串包含A’的回文串，根据对称性，回文串A和A’是一样的，所以len[A] = len[A’]。

否则，len[A] = len[m]-A。还是由对称性，若len[A]>len[m]-A，则len[m]也肯定比原来要长，画个图看看。

2.st[P]在st[A]的左边：

得好好将A两侧字符进行比较。

while(st[A--] == st[A++])

在manacher算法中的m我们取A左边的回文串中能够到达最右端的，毫无疑问这样更高效。

情况1的复杂度是0(1)，情况2的复杂度理论上是0(n),但

m的长度是不断递增的，假如在字符串n/2,n/4,n/8…..处更新m，即我们总的比较次数为num = (n/2+n/4+n/8…..) = 2\*n,

总的复杂度为0(n)。