回文子串问题

问题：

给定一个字符串 s，返回其最长回文子串

分析：

法 1：

采用两边扩展法，即以一个点为中心向左右两边扩展，求最长回文子串，但要注意会出现奇数长度和偶数长度的子串，所以应该分别求解。

法 2：

采用动态规划算法，建立数组 $P(i, j)$ 表示 S 中从 i 位置到 j 位置能不能构成回文子串，则有

$$P(i, j) = \begin{cases} true, & \text{如果子串} S_i, \cdots S_j \text{是回文子串} \\ false, & \text{其他情况} \end{cases}$$

容易给出状态转移方程

$$P(i, j) = P(i+1, j-1) \&\& (S_i == S_j)$$

法 3：

采用 Manacher 算法

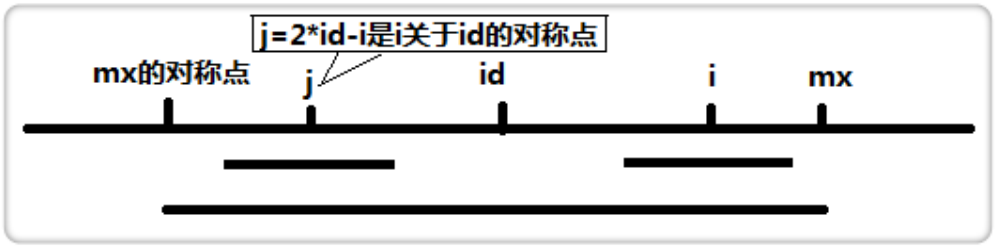
考虑到奇串和偶串的问题，想到向 S 添加字符的方法，得到

$$new\_S = \$\#S_1\#S_2\#, \cdots, \#S_n\#$$

这样处理后得到的 new_S 总是一个奇串了，定义数组 $P(i)$，表示以 $new\_S(i)$ 为中心可以得到的回文串的最长半径，则源串的最长回文子串的长度为 $P(i)-1$，例如：

i	0	1	2	3	4	5	6	7	8	9	10	11
new_S	$	#	a	#	b	#	a	#	b	#	a	#
P		1	2	1	4	1	6	1	4	1	2	1

接下来就是怎么求解 $P(i)$，看下面的图：



设 id 是已经遍历过的位置，mx 是以 id 为中心的最长回文串的右边界，即

$mx = id + P(id)$，现在来求$P(i)$，如果$i < mx$，那么有

$$P(i) = \min(P(2*id - i), mx - i)\ ,\quad if\ (i < mx)$$

$2*id - i$即是i的对称点j，我们就可以根据之前求过的$P(j)$来加快查找。

下面给出算法的C++实现：

```
1.  string longestPalindrome(string s)
2.  {
3.          //Manacher 算法
4.          if(s.size() == 0 || s.size() == 1)
5.              return s;
6.          //现将 s 变为长度为奇数的串
7.          string news = "&#";
8.          for(int i=0;i<s.size();i++)
9.          {
10.             news += s[i];
11.             news += '#';
12.         }
13.         news += '\0';
14.         int len = news.size();   //新串的长度
15.         int maxlen = -1;  //记录最长回文子串的长度
16.         int maxid = 0; //记录最长回文子串的中心
17.         vector<int> p(len);  //p[i]记录以 i 为中心的最长回文子串的半径，则
    maxlen = p[i] - 1
18.         int id = 0; //当前确定回文中心
19.         int mx = 0;  //当前回文中心能到达的右边界下标
20.         for(int i=1;i<len;i++)
21.         {
22.             if(i < mx)  //i 在当前的回文子串中，确定以 i 为中心的构成的最长的回文
    半径
23.                 p[i] = min(mx - i, p[2 * id -i]);  //i 关于 id 的对称下标为
    2*id-1，p[2*id-1]的最长半径已经确定，根据对称，p[i]应该取较小的一个
24.             else
25.                 p[i] = 1;  //以他本身为回文子串
26.             while(news[i - p[i]] == news[i + p[i]])  //确定以 i 为中心的最长回
    文半径
27.                 p[i]++;
28.             if(i + p[i] > mx)  //超出了 mx 的范围，应该更新 mx 和对称中心
29.             {
30.                 id = i;
31.                 mx = i + p[i];
32.             }
33.             if(maxlen < p[i] - 1)
```

```
            {
                maxlen = p[i] - 1;
                maxid = i;
            }
        }
        string result = "";
        for(int i = maxid - maxlen;i<=maxid + maxlen;i++)
            if(news[i] != '#')
                result += news[i];
        return result;
    }
```