给定一个字符串 **s**，你可以通过在字符串前面添加字符将其转换为回文串。找到并返回可以用这种方式转换的最短回文串。

示例 1:

输入：`"aacecaaa"`
输出：`"aaacecaaa"`

示例 2:

输入：`"abcd"`
输出：`"dcbabcd"`

思路，该题个人没啥思路，看了别人的代码，模仿其manacher算法写了一个刚好通过的代码。先找串中回文，找到刚好到头的最长回文，平凑后返回

自己代码

```
class Solution {
public:
    string shortestPalindrome(string s) {
        if(s.empty()){
            return s;
        }else if(s.size()==0){
            return "";
        }
        string s1="$";
        int n=s.size();
        for(int i=0,j=1;i<n;i++,j=j+2){
            s1+=s[i];
            s1+='#';
        }
        int wz=0;
        manacher(s1,wz);
        s1=s;
        s.erase(0,wz);
        reverse(s.begin(),s.end());
        return s+s1;
    }
    void manacher(string s,int &wz){
        vector<int> p(s.size());
        int sz=0;
        int j=0;
        int n=s.size();
        for(int i=1;i<n;i++){
            if(i<sz){
                p[i]=min(p[2*j-i],sz-i);
            }else{
                p[i]=1;
            }
```

```
            while(s[i-p[i]]==s[i+p[i]]){
                p[i]++;
            }
            if(sz<i+p[i]){
                j=i;
                sz=i+p[i];
            }
            if(p[i]==i){
                wz=p[i];
            }
        }
    }
};
```

他人方法1：manacher

1.

```
public class Solution {
    public String shortestPalindrome(String s) {
        if (s == null) return null;
        if (s.length() == 0) return "";
        char[] sa = s.toCharArray();
        char[] ma = new char[sa.length*2+1];
        ma[0] = '#';
        for(int i=0, j=1; i<sa.length; i++, j+=2) {
            ma[j] = sa[i];
            ma[j+1] = '#';
        }
        int m = ma.length/2;
        int[] radius = new int[ma.length];
        int rightmost = 1;
        int center = 1;
        int patch = sa.length - 1;
        for(int i=1; i<=ma.length/2; i++) {
            int min = rightmost <= i ? 1 : Math.min(rightmost-i, radius[center - (i -center)]) + 1;
            for(int r=min; i-r>=0; r++) {
                if (ma[i-r] != ma[i+r]) break;
```

```
                radius[i] = r;
                if (i-r==0) patch = sa.length - i;
            }
            if (rightmost < i+radius[i]) {
                rightmost = i + radius[i];
                center = i;
            }
        }
        char[] palindrome = new char[sa.length + patch];
        for(int j=0, k=sa.length-1; j<patch; j++, k--) palindrome[j] = sa[k];
        System.arraycopy(sa, 0, palindrome, patch, sa.length);
        return new String(palindrome);
    }
```

manacher方法2

```
1. public class Solution {
2.     public String shortestPalindrome(String s) {
3.         char[] sa = s.toCharArray();
4.         char[] ma = new char[sa.length * 2 + 1];
5.         Arrays.fill(ma, '#');
6.         for(int i = 0, j = 1; i < sa.length; i++, j += 2) {
7.             ma[j] = sa[i];
8.         }
9.         int[] radius = new int[ma.length];
10.         int center = 0;
11.         int rightmost = 0;
12.         int min = sa.      ;
13.         for(int i = 1; i < ma.length - 1; i++) {
14.             int j = 0;
15.             if (i < rightmost) {
16.                 j = Math.min(rightmost - i, radius[center * 2 - i]);
17.                 radius[i] = j;
18.             }
19.             j++;
```

```
            for(; i - j >= 0 && i + j < ma.length && ma[i - j] == ma[i + j]; j++)
{
                radius[i] = j;
                if (rightmost < i + j) {
                    rightmost = i + j;
                    center = i;
                }
                if (i - j == 0) min = Math.min(min, sa.length - j);
            }
        }
        StringBuilder sb = new StringBuilder(s.substring(sa.length - min));
        sb.reverse();
        sb.append(s);
        return sb.toString();

    }
```

KMP方法

```
1. public class Solution {
2.     public String shortestPalindrome(String s) {
3.         char[] ma = new StringBuilder(s).append("#").append(new
StringBuilder(s).reverse().toString()).toString().toCharArray();
4.         int[] next = new int[ma.length];
5.         for(int i=1; i<ma.length; i++) {
6.             int j=next[i-1];
7.             while (j>0 && ma[j]!=ma[i]) j=next[j-1];
8.             next[i] = j + (ma[j]==ma[i]? 1 : 0);
9.         }
10.         return new StringBuilder(s.substring(next[ma.length-
1])).reverse().toString()+s;
11.     }
```

KMP方法2

```java
public class Solution {
    public String shortestPalindrome(String s) {
        if (s.length() == 0) return s;
        StringBuilder sb = new StringBuilder(s);
        sb.reverse();
        sb.insert(0, "#");
        sb.insert(0, s);
        char[] pa = sb.toString().toCharArray();
        int[] lens = new int[pa.length];
        for(int i = 1; i < lens.length; i++) {
            int j = lens[i - 1];
            while (j > 0 && pa[j] != pa[i]) j = lens[j - 1];
            lens[i] = j + (pa[j] == pa[i] ? 1 : 0);
        }
        StringBuilder pb = new StringBuilder();
        pb.append(s.substring(lens[lens.length - 1]));
        pb.reverse();
        pb.append(s);
        return pb.toString();
    }

}
```