http://www.cnblogs.com/JohnTsai/p/5606719.html

```
//输入一个字符串，输出它的倒序字符串，注意不能使用系统提供的API
input:  Hello
output: olleH
```

1.使用数组

```
/**
 * 将字符串转换为char数组
 * 遍历循环给char数组赋值
 */
public static String strReverseWithArray(String string){
    if(string==null||string.length()==0)
        return string;
    int length = string.length();
    char [] array = string.toCharArray();
    for(int i = 0;i<length;i++){
        array[i] = string.charAt(length-1-i);
    }
    return new String(array);
}

/**
 * 优化：
 分析上一种解法，循环遍历时，我们不需要循环这么多次。
 每次循环的时候，我们应该直接给前、后位置都赋值。
 */
public static String strReverseWithArray2(String string){
    if(string==null||string.length()==0)return string;
    int length = string.length();
    char [] array = string.toCharArray();
    for(int i = 0;i<length/2;i++){
        array[i] = string.charAt(length-1-i);
        array[length-1-i] = string.charAt(i);
    }
    return new String(array);
}
```

2.使用栈

```
/**
 * 将字符串转换为char数组
 * 将char数组中的字符依次压入栈中
 * 将栈中的字符依次弹出赋值给char数组
 */
public static String strReverseWithStack(String string){
    if(string==null||string.length()==0)
        return string;
    Stack<Character> stringStack = new Stack<Character>();
    char [] array = string.toCharArray();
    for(Character c:array){
        stringStack.push(c);
    }
    int length = string.length();
    for(int i= 0;i<length;i++){
        array[i] = stringStack.pop();
    }
    return new String(array);
}
```

3.逆序遍历

```
/**
 * 逆序遍历字符串中的字符，
 * 并将它依次添加到StringBuilder中
 */
public static String strReverseWithReverseLoop(String string){
    if(string==null||string.length()==0)
        return string;
    StringBuilder sb = new StringBuilder();
    for(int i = string.length()-1;i>=0;i--){
        sb.append(string.charAt(i));
    }
    return sb.toString();
}
```

4.使用位运算

```
/**
 * 计算机的数据流本质上都是0，1二进制数据。字符串也是一样。
```

```
 * 而二进制数据的处理往往是通过位运算来实现的。
 * 位操作有:与，或，非，异或。
 * 使用异或操作能实现交换两个变量的值而不引入第三个变量。
 *  before:"Hello"
 * after: "olleH"
 * index: 0   1   2   3   4
 * char : H   e   l   l   o
 * ASCII: 72  101 108 108 111
 * 以第0个字符和第4个字符交换为例:
 * * 交换前:
 * array[0]=1001000
 * array[4]=1101111
 * * 交换:
 * array[0]=array[0]^array[4]=0100111
 * array[4]=array[4]^array[0]=1001000
 * array[0]=array[0]^array[4]=1101111
 * * 交换后:
 * array[0]=1101111--->111-->o
 * array[4]=1001000--->72--->H
 */
public static String strReverseWithXor(String string){
    if(string==null||string.length()==0)return string;
    char [] array =string.toCharArray();
    int length = string.length()-1;
    for(int i =0;i<length;i++,length--){
        array[i]^=array[length];
        array[length]^=array[i];
        array[i]^=array[length];
    }
    return new String(array);
}
```

5.使用递归

```
/**
 * 找出递归结束的条件
 * 对针对于临界条件的不同的值做出不同的处理
 */
public static String strReverseWithRecursive(String string){
    if(string==null||string.length()==0)
        return string;
    int length = string.length();
```

```
    if(length==1){
        return string;
    }else{
        return  strReverseWithRecursive(string.substring(1))
            + string.charAt(0);
    }
}
```