循环左移数组

问题描述：给定一个数组，如 a[a, b, c, d, e, f ,g] 循环左移 k位
a[d,e,f,g,a,b,c]；要求时间复杂度为O(N)。

问题分析：1．输入：一个数组，[a,b,c,d,e,f,g]
2．特征：①循环左移
②元素的相对位置不变[d,e,f,g],[a,b,c]
3．约束：①时间复杂度为O(n)
4．输出：[d,e,f,g,a,b,c]。

解决思路一：

1．新建一个数组b[k]，用于存放前k个元素：b[a,b,c]

2．将原数组的后n-k个元素移动到前面：a[d,e,f,g]

3．将b[k]中的元素放入原数组的后面：a[d,e,f,g,a,b,c]

C++代码实现：

```
#include <stdio.h>
#include<iostream>
using namespace std;
int main()
{
    const int n=7,k=3;
    char a[n]={'a','b','c','d','e','f','g'};
    char b[k];
    int i;
    for(i=0;i<k;i++)b[i]=a[i];          //将前k个放到新的数组里
    for(;i<n;i++)a[i-k]=a[i];           //将后边的元素前移
    for(i=0;i<k;i++)a[n-k+i]=b[i];      //将前k个元素放到末尾
    for(i=0;i<n;i++)cout<<a[i]<<" ";
    cout<<endl;
}
```

算法的时间复杂度：

1．时间复杂度：可以看到算法只需进行线性的扫描即可，故时间复杂度为O(n)

2．空间复杂度：由于用到一个临时数组b[k]，故空间复杂度为O(k)

解决思路二：

1．假设前k个元素为A，后n-k个元素为B，原数组为AB

2．新数组为$BA=((BA)^{-1})^{-1}=(A^{-1}B^{-1})^{-1}$

3．先对A求逆,再对B求逆,再对$A^{-1}B^{-1}$求逆

4．时间复杂度：算法只是线性扫描为O(n)

5．空间复杂度：算法只用原数组的空间，故为O(1)

C++代码实现：

```
#include <stdio.h>
#include<iostream>
using namespace std;
void Reverse(char *a,int left,int right)
{
    int mid = (left+right)/2;
    char temp;
    for(int i=0;i<=(mid-left);i++)
    {
        temp = a[left+i];
        a[left+i] = a[right-i];
        a[right-i]= temp;
    }
}
int main()
{
    const int n=7,k=3;
    char a[n]={'a','b','c','d','e','f','g'};
    Reverse(a,0,n-1-k); //对A进行逆置
    Reverse(a,n-k,n-1); //对B进行逆置
    Reverse(a,0,n-1);   //对A^-1 B^-1进行逆置
    for(int i=0;i<n;i++)cout<<a[i]<<" ";
    cout<<endl;
}
```