# 旋转数组

题目描述：给定一个整数数组 *nums*，将数组中的元素向右轮转 *k* 个位置，其中 *k* 是非负数。

## 代码头部文件和主函数

#include<bits/stdc++.h>   
using namespace std;   
int main()   
{   
 Solution s;   
 vector<int> nums={1,2,3,4,5,6,7};   
 s.rotate(nums,3);   
 for(auto i:nums)   
 {   
 cout<<i<<" ";   
 }   
 return 0;   
}

## 方法一临时数组

思路：设一个新的数组，将这个数组的后两位存储于一个temp里面，接下来将这个数组的前面所有的向后轮转k个位置，最后把temp数组写入数组中去

代码实现：

class Solution {   
public:   
 void rotate(vector<int>& nums, int k) {   
 vector<int> temp;   
 int n=nums.size();   
 k=k%n;   
 for(int i=n-k;i<n;i++)   
 {   
 temp.push\_back(nums[i]);   
 }//将后k个元素放入temp中   
 for(int i=n-1;i>=k;i--)   
 {   
 nums[i]=nums[i-k];   
 }//将前n-k个元素向后移动k位   
 for(int i=0;i<k;i++)   
 {   
 nums[i]=temp[i];   
 }//将temp中的元素放入前k位   
 }   
};

运行结果如下所示:



## 方法二按部就班移动法

思路：将整个数组按照顺序，向右移动，设一个temp变量用来存储nums[n-1]，并且将这个变量重新赋值给nums[0]，这个算作第一次翻转。共计翻转k次

代码如下：

class Solution {   
public:   
 void rotate(vector<int>& nums, int k) {   
 for(int i=0;i<k;i++)   
 {   
 int temp=nums[nums.size()-1];//存储最后一个元素   
 for(int j=nums.size()-1;j>0;j--)//将前面的元素向后移动   
 {   
 nums[j]=nums[j-1];   
 }   
 //将最后一个元素放到第一个位置   
 nums[0]=temp;   
 }   
 }   
};

运行结果如下所示：



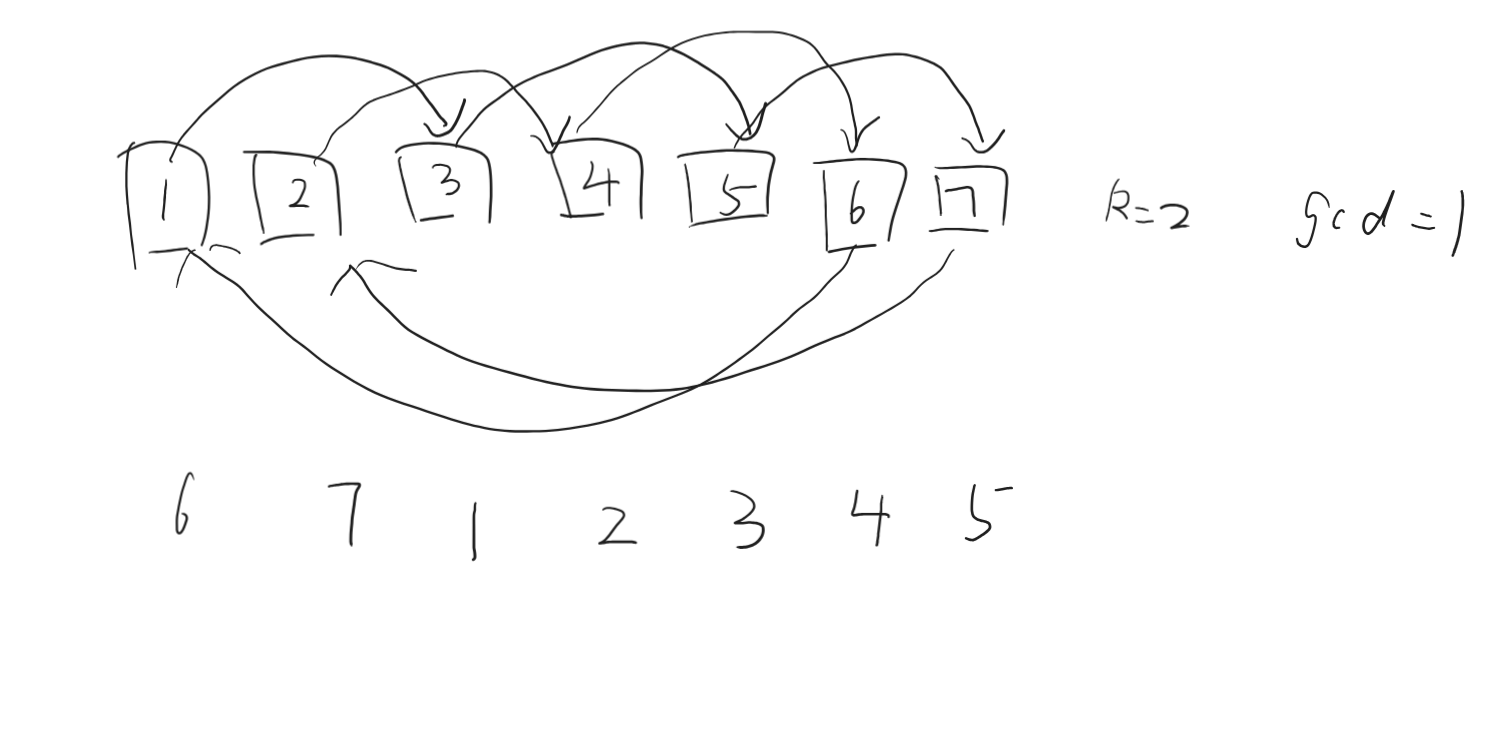
时间复杂度：O(k\*n)

空间复杂度：O(1)

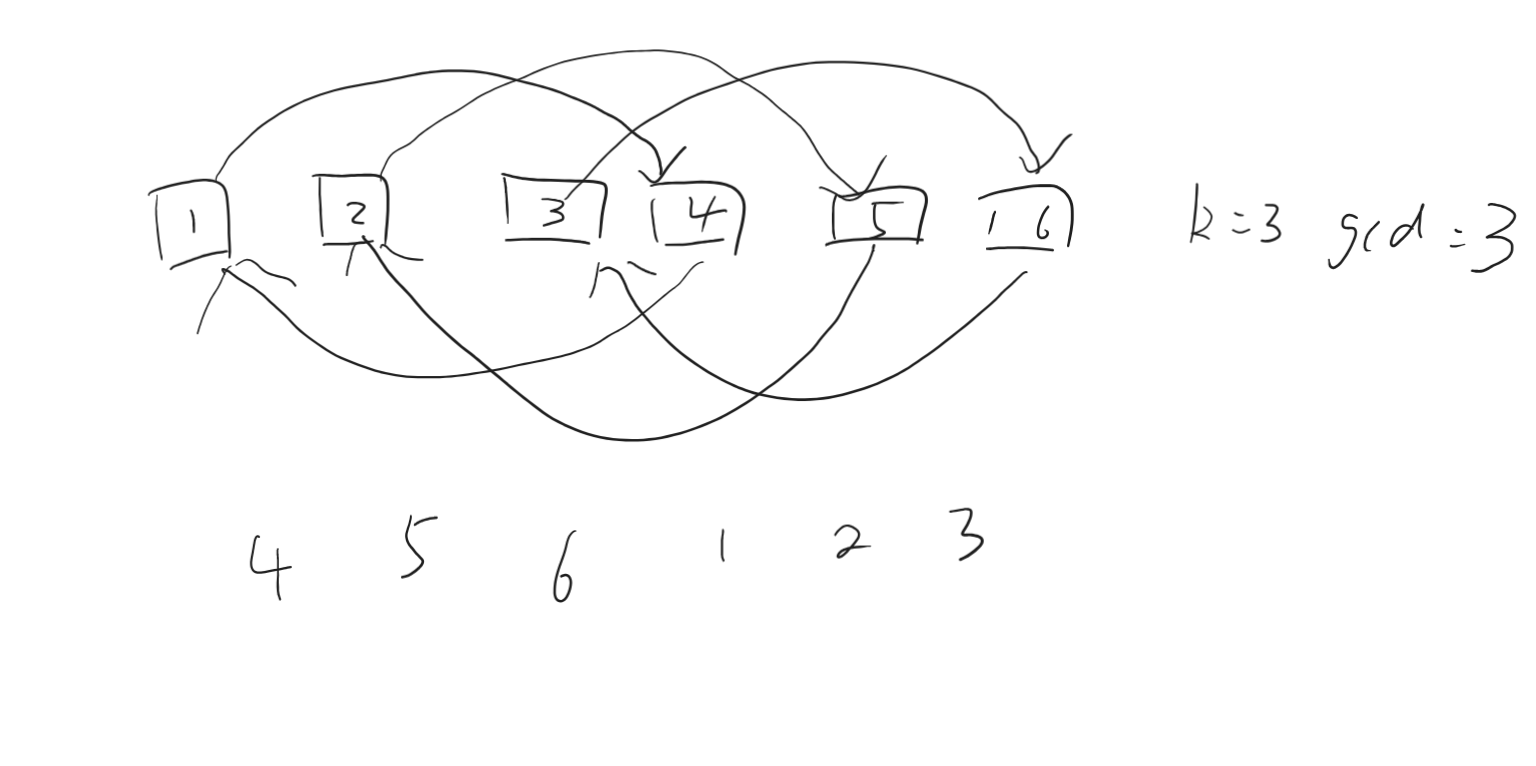
## 方法三最大公约数法

思路：循环替换这个数组，将数组的首位依次往下，直到回归到本位，如果这个个数和长度能被整除，也就是说有除1外的公约数的时候，就应该确定公约数次位置，详情如下图举例说明

假设[1,2,3,4,5,6,7] k=2；



假设[1,2,3,4,5,6] k=3;

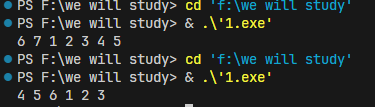


思路大体就是将每个数字向后放k个位置，其因为只会占用一个temp用来储存值，所以他只会占用常量空间，但是他每个数字都会重新确定位置，所以他的时间复杂度为O(n)

代码实现：

class Solution {   
public:   
 void rotate(vector<int>& nums, int k) {   
 int t=\_\_gcd((int)nums.size(),k);   
 k=k%nums.size();   
 for(int i=0;i<t;i++)   
 {   
 int j=i;   
 int temp=nums[i];   
 while(1)   
 {   
 int next=(j+k)%nums.size();   
 if(next==i)   
 {   
 break;   
 }   
 swap(temp,nums[next]);   
 j=next;   
 }   
 nums[i]=temp;   
 }   
 }   
};

运行结果（运行和示例图的结果相符）;



## 方法四分块法

分块的方法就是将数组分成两个块，单独设置一个数组，存储size-k后的数组，之后，把向后移动k次，之后把right数组赋值回去，代码t如下：

class Solution {   
public:   
 void rotate(vector<int>& nums, int k) {   
 //块交换   
 vector<int> arrright(nums.end()-k,nums.end());   
 for(int i=nums.size()-1;i>=k;i--)   
 {   
 nums[i]=nums[i-k];   
 }   
 for(int i=0;i<k;i++)   
 {   
 nums[i]=arrright[i];   
 }   
 }   
};

结果如下：



## 方法五三次旋转法

以n-k为分界点，先旋转全部，再旋转对旋转后的左侧旋转，再右侧旋转，3次旋转即可

代码如下:

class Solution {   
public:   
 void rotate(vector<int>& nums, int k) {   
 int n=nums.size();   
 k=k%n;   
 reverse(nums.begin(),nums.end());   
 reverse(nums.begin(),nums.begin()+k);   
 reverse(nums.begin()+k,nums.end());   
 }   
};

