无序数组有序化最小交换次数
1.若只能交换相邻两数，那么最小交换次数为该序列的逆序数

```
//求数组逆序数，归并排序同时计算逆序数
//复杂度O(nlogn)
int cnt;
int Merge(int *A,int *L,int left,int *R,int right)//合并操作
{
    int i,j,k;
    i=0;j=0;k=0;
    while(i<left && j<right){
        if(L[i]<R[j]){
            A[k++]=L[i++];
        } else {
            cnt+=left-i;
            A[k++]=R[j++];
        }
    }
    while(i<left){
        A[k++]=L[i++];
    }
    while(j<right){
        A[k++]=R[j++];
    }
    return cnt;
}
int Mergesort(int *arr,int n)//等分递归
{
    int mid,*L,*R;
    if(n<2) return 0;
    mid=n/2;
    L=new int[mid];
    R=new int[n-mid];
    for(int i=0;i<mid;i++) L[i]=arr[i];
    for(int i=mid;i<n;i++) R[i-mid]=arr[i];
    int n1=Mergesort(L,mid);
    int n2=Mergesort(R,n-mid);
    int n3=Merge(arr,L,mid,R,n-mid);
    delete [] R;
    delete [] L;
    return n1+n2+n3;
}
```

2.交换任意两数
找循环节求出最小交换次数

```cpp
int getMinSwaps(vector<int> &nums){
    //排序
    vector<int> nums1(nums);
    sort(nums1.begin(),nums1.end());
    unordered_map<int,int> m;
    int len = nums.size();
    for (int i = 0; i < len; i++){
        m[nums1[i]] = i;//建立每个元素与其应放位置的映射关系
    }
    int loops = 0;//循环节个数
    vector<bool> flag(len,false);
    //找出循环节的个数
    for (int i = 0; i < len; i++){
        if (!flag[i]){//已经访问过的位置不再访问
            int j = i;
            while (!flag[j]){
                flag[j] = true;
                j = m[nums[j]];//原序列中j位置的元素在有序序列中的位置
            }
            loops++;
        }
    }
    return len - loops;
}
```