快速排序

快速排序是一种基于分治思想的排序，平均算法时间复杂度为 **nlogn** 。

它没有并归排序那么稳定。

具体实现是先确定一个数x（基准），然后用双指针将这个数组扫一遍，把左边比x大的数和右边比x小 的数进行交换，直到扫完这个数组。然后将这个数组分两个部分 **l - j** 和 **i - r**，因为while判断条件为 **i<=j** 所以 出循环时**i，j**大小关系一定为 **i = j + 1** ，然后将这两个区间进行递归。

```
void quick_sort(int q[],int l,int r)
{
  if(l >= r) //区间只有一个数或者没有
    return;
  int x = q[(l+r)/2], i = l  , j = r;
  //x这个值可以任取
  while(i <= j)
  {
    while(q[i] < x)  i++; //左边指针的数小于x符合条件 指针移动
    while(q[j] > x)  j--; //右边指针的数大于x符合排序条件 指针移动
    if(i <= j)  //找到不符合条件的即左边的指向的数大于x，右边小于x
    {
      int t;
      t = q[i];  //交换
      q[i] = q[j];
      q[j] = t;
      i++,j--; //换完之后一定要移动指针！！
    }
  }

  quick_sort(q,l,j);
  quick_sort(q,i,r);
}
```

归并排序

归并排序也是一种基于分治思想的排序，它比快速排序**稳定**，时间复杂度为 **nlogn** ，但是它的空间复杂度要比快排高，为什么呢，因为他需要一个辅助排序的数组。

归并排序和快速排序不一样，归并排序是先进行递归，而快排是先排再递归的。归并排序递归的过程是一颗二叉树。

先将数组从中间划分成两部分，一直递归下去，直到区间长度为 1 或 0 结束递归。然后用两个指针分别指向这个区间的最左侧和中间往右挪一位的位置，比较大小，小的先放进临时数组 **tmp** 中，进行合并，最后将**tmp** 复制进原数组。

```cpp
#include<cstdio>
using namespace std;
const int N = 1000010;
int n,q[N],tmp[N]; //辅助数组tmp


void merge_sort(int q[],int l,int r)
{
  if(l >= r) //区间长度为 1 || 0
    return ;

  int mid = l + r >> 1; //等价于（l + r）/2

  merge_sort(q,l,mid);  //进行递归
  merge_sort(q,mid + 1,r);

  int k = 0, i = l, j = mid + 1; //初始化指针
  while(i<=mid && j<=r) //合并
  {
    if(q[i]<=q[j])  tmp[k++] = q[i++]; //按大小放进数组中
    else  tmp[k++] = q[j++];
  }
  while(i <= mid)  //把残余的放进去，此时左边剩下的数都大于tmp数组已有的数
    tmp[k++] = q[i++];
  while(j <= r)
    tmp[k++] = q[j++];

  for(i = l,j = 0; i <= r; i++,j++)
    q[i] = tmp[j]; //复制进原数组
}


int main()
{
  scanf("%d",&n);
  for(int i=0;i<n;i++)
    scanf("%d",&q[i]);

  merge_sort(q,0,n - 1);

  for(int i=0;i < n;i++)
    printf("%d ",q[i]);
}
```