快速排序

思路

• 1. 确定分界点的值 `x`

三种选择：取左边界 `q[l]` ，取中间值 `q[(r + 1)/2]` ，取右边界 `q[r]`

• 2. 调整区间

把该序列分为两部分，使得所有小于等于 `x` 的在左边部分，大于等于 `x` 的在右边部分

- 实现方法1-略暴力

1. 定义两个数组 `a,b`
2. 扫描数组 `q` ，把小于 `x` 的放入 `a` 数组，把大于等于 `x` 的放入 `b` 数组
3. 把 `a` ， `b` 赋值给 `q`

- 实现方法2-推荐

1. 定义两个指针 `i` ， `j` 分别指向 `q` 的左右两端
2. `i` 向左移动，直到遇到 `q[i] > x` 暂停
3. 随后， `j` 向右移动，直到遇到 `q[j] < x` 暂停
4. 交换 `q[i]` 和 `q[j]` 的值，然后 `i` 向左移动一次， `j` 向右移动一次
5. 当 `i < j` 时，重复2~4步

• 3. 递归处理

对左右两部分递归执行1，2两步

- **时间复杂度 $\theta(nlogn)$**

期望上，快速排序平均分为 $logn$ 层（因为分界值的选择不同而产生差异），每层的扫描交换复杂度之和是 $O(n)$ 所以平均时间复杂度 $\theta(nlogn)$

模板

```
void quick_sort(int q[], int l, int r)
{
    if(r <= l)  return;
    int x = q[(l + r) >> 1], i = l - 1, j = r + 1;
    while(i < j)
    {
        do i++; while(q[i] < x);
        do j--; while(x < q[j]);
        if(i < j) swap(q[i], q[j]);
    }
    quick_sort(q, l, j), quick_sort(q, j + 1, r);
}
```

练手

AcWing 785. 快速排序

AcWing 786. 第k个数

归并排序

思路

- **1. 确定分界点**

使用数组的中间点为分界点，即 `mid = (left + right) / 2`

- **2. 递归对左右两部分进行排序**

- **3. 将左右两部分合二为一**

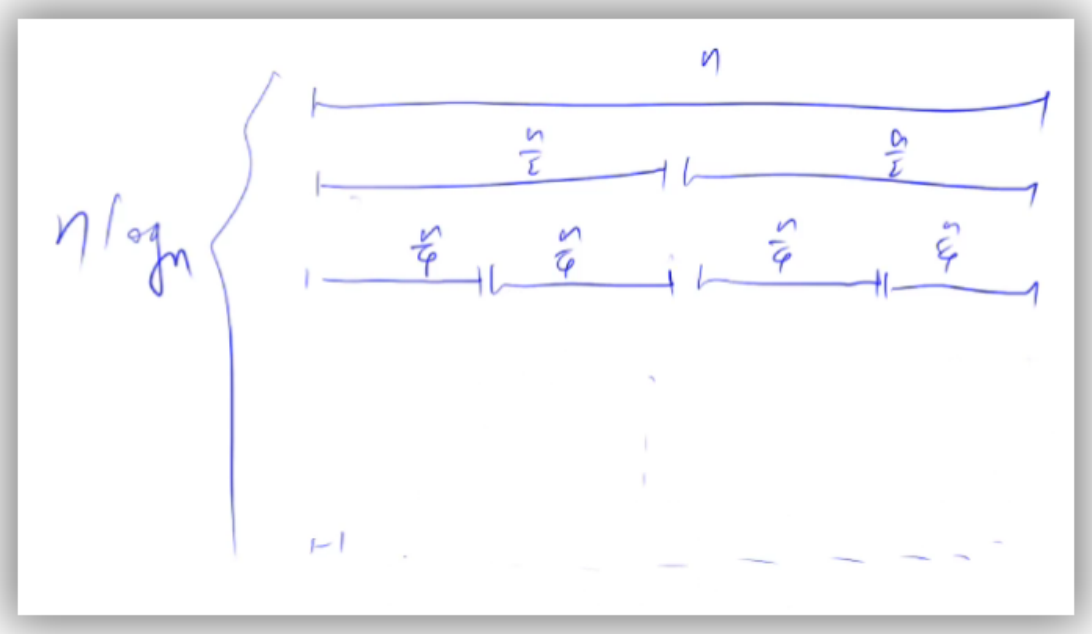
大致思路是使用两个指针分别指向两部分的开头，再定义一个数组 `res` 存放两部分有序合并的结果

比较两个指针指向的值，将较小的存入 `res` ，对应的指针向后移动，较大的不动，按照这个规则重复直至两部分都存入了 `res` 即可

- **时间复杂度 `O(nlogn)`**

递归地进行二分，一直到左右两部分各一个元素，一共可以分为 `logn` 层，然后进行合并

每层无论被范围多少部分，加起来都是 `n` 个元素，即每层合二为一的复杂度是 `O(n)` ，得到整个算法的复杂度是 `O(nlogn)`



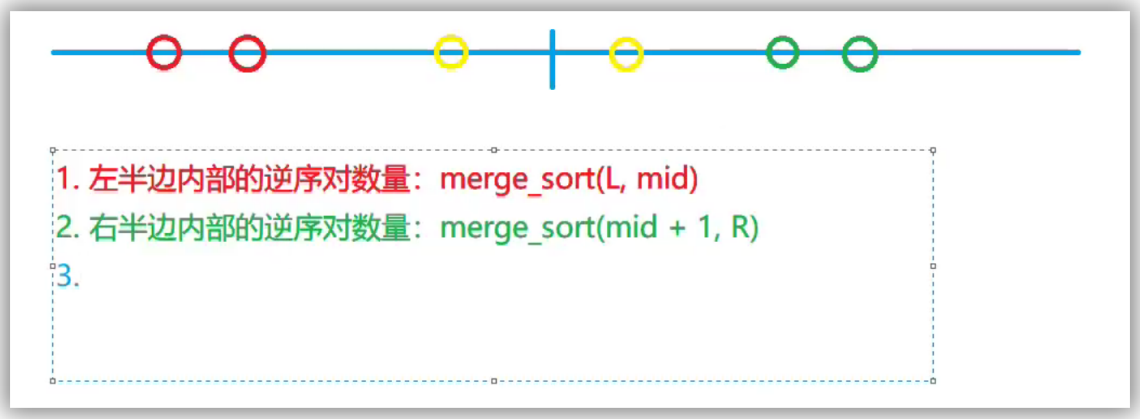
模板

```
int tmp[N];
void merge_sort(int q[], int l, int r)
{
    if(r <= l) return;
    int mid = (l + r) >> 1;
    merge_sort(q, l, mid), merge_sort(q, mid + 1, r);
    int k = 0, i = l, j = mid + 1;
    while(i <= mid && j <= r)
        if(q[i] <= q[j])tmp[k++] = q[i++];
        else            tmp[k++] = q[j++];
    while(i <= mid)     tmp[k++] = q[i++];
    while(j <= r)   tmp[k++] = q[j++];
    for(int i = l, j = 0; i <= r; ++i, ++j) q[i] = tmp[j];
}
```

练手

AcWing 787. 归并排序

AcWing 788. 逆序对的数量



我们求逆序对总数实际上就是上面三种情况的逆序对数量之和，逆序对两元素都在左半部分和都在右半部分交给递归处理即可，主要需要处理的就是跨越左半部分和右半部分的情况

这种情况下，我们在一旦在左半部分发现 `q[i]` 大于右半部分的 `q[j]` ，那么在左半部分中 `q[i]` 及它后面的 `mid - i` 个元素都和 `q[j]` 构成逆序对，跨越情况的逆序对 `+ (mid - i + 1)`

> “
>
> 注意当数组有 `n` 个元素时，逆序对的数量最大的情况出现在 `n - 1, n - 2, ..., 0` 这种序列的情况下，逆序对数量为 `(n - 1) + (n - 2) + ... + 1 + 0` ，数量级在$O(n^2)$，这种情况下注意有时需要将逆序对的计数变量定义为 `long long`

```
#include<iostream>
using namespace std;
typedef long long ll;
const int N = 1e5 + 10;
int q[N], tmp[N];
int n;

ll merge_sort(int q[], int l, int r)
{
    if(r <= l)  return 0;
    int mid = (l + r) >> 1;
    ll res = merge_sort(q, l, mid) + merge_sort(q, mid + 1, r);
    int k = 0, i = l, j = mid + 1;
    while(i <= mid && j <= r)
    if(q[i] <= q[j]) tmp[k++] = q[i++];
    else
    {
        res += mid - i + 1;
        tmp[k++] = q[j++];
    }
    while(i <= mid) tmp[k++] = q[i++];
    while(j <= r)   tmp[k++] = q[j++];
    for(int i = l, j = 0; i <= r; ++i, ++j) q[i] = tmp[j];
    return res;
}

int main()
{
```

```
29      cin >> n;
30      for(int i = 0; i < n; ++i)  scanf("%d", &q[i]);
31      cout << merge_sort(q, 0, n - 1) << endl;
32      return 0;
33  }
```