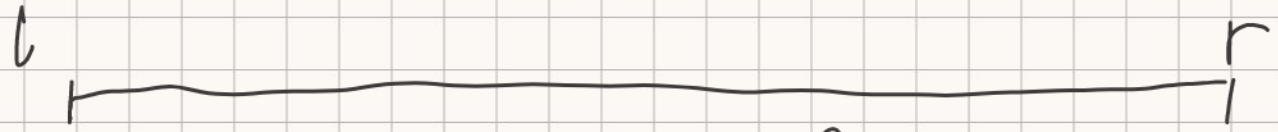
排序 < 快速排序
归并排序

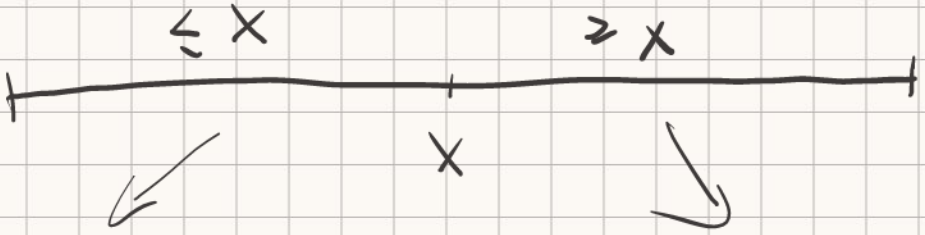
一个 < 整数一个
浮点数一个

快速排序 → 分治



① 确定分界点
- 直接取左边界 q[l]
- 中点值 q[l+r]/2
- 取右边界 q[r]
- 随机

② 调整区间：



左边部分都小于等于 x

右边部分都大于等于 x

③ 递归处理左，右两边

调整区间的简单做法：

1) 开两个额外的数组 a[], b[]

2) 扫描整个区间 q[l~r]

q[i] ≤ x → x → a[]

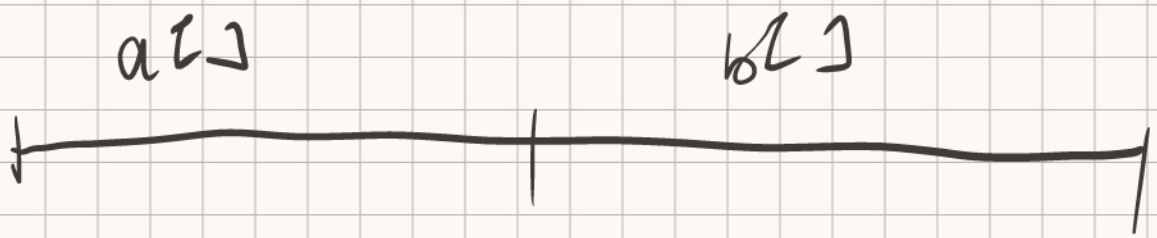
q[i] > x → x → b[]

如果当前数小于等于x，放进 a[]

如果当前数大于x，放进b[]

3) a[] → q[]，b[] → q[]

先把a[]放进q[]，再把b[]放进q[]



（需要两个额外的空间）

O(n)

更好一点的做法：

两个指针：i 和 j

两个指针分别向中间走：

(1) j 先走，如果 i 指向的当前值小于等于 x，j 向后走。

(2) 直到 i 指向的当前值大于等于 x，i 开始移动，如果 j 当前指向的值大于等于 x，j 继续向前移动。

(3) 直到 j 的当前值小于等于 x，此时把 i 指向的数和 j 指向的交换（swap）

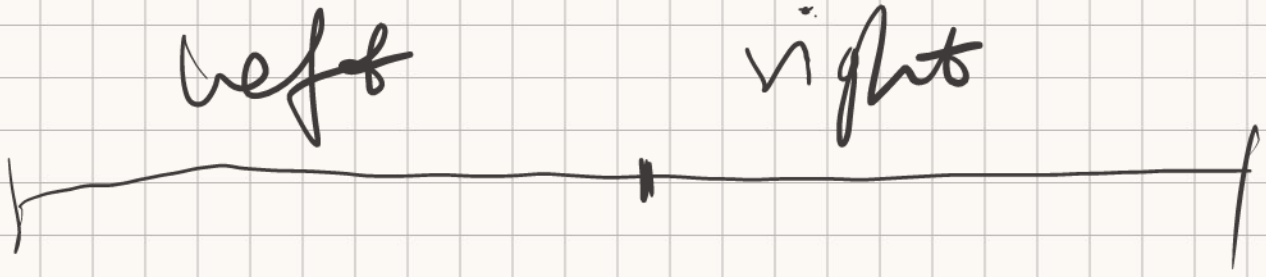
然后 i 和 j 又继续向中间走，直到 i 和 j 相遇

代码实现：


```cpp
#include <iostream>

using namespace std;

const int N = 1e6 + 10;

int n;
int q[N];

void quick_sort(int q[], int l, int r)
{
    if (l >= r) return;

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

    quick_sort(q, l, j);
    quick_sort(q, j + 1, r);
}

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

    quick_sort(q, 0, n - 1);

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

    return 0;
}
```

注意边界问题

归并排序 → 分治



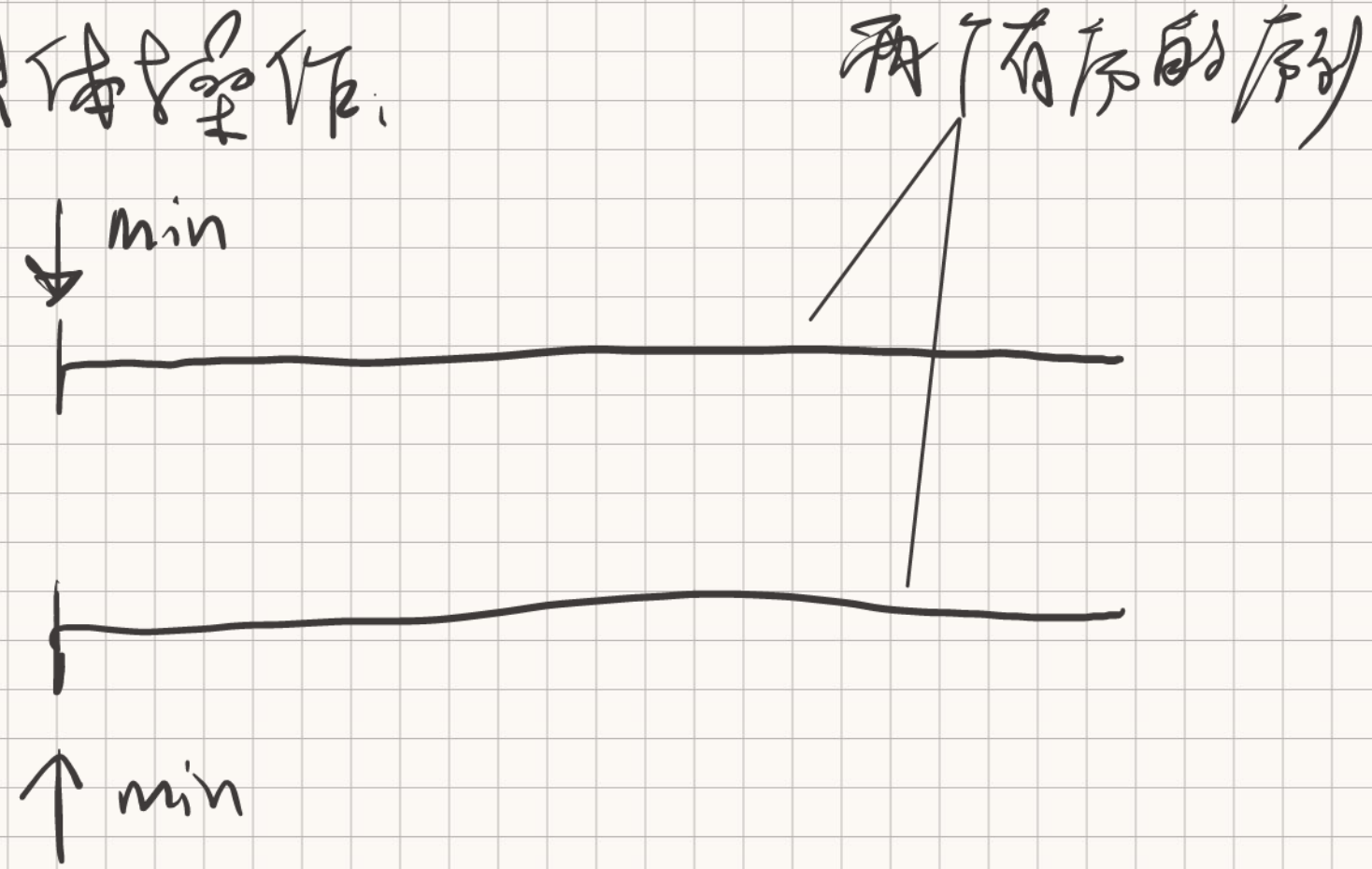
① 确定分界点 mid = (l+r)/2

② 递归排序 (left, right)

③ 归并 → 将两个有序的合并为一个有序的(合二为一)

双指针

具体操作：



(1) 两个指针指向两个有序序列的第一个位置

(2) 定义一个新的数组 res[] 来记录答案

(3) 序列已经排好了序了，指针指向的第一个位置是两个序列的最小值

(4) 将两个最小值比较，小的放入 res

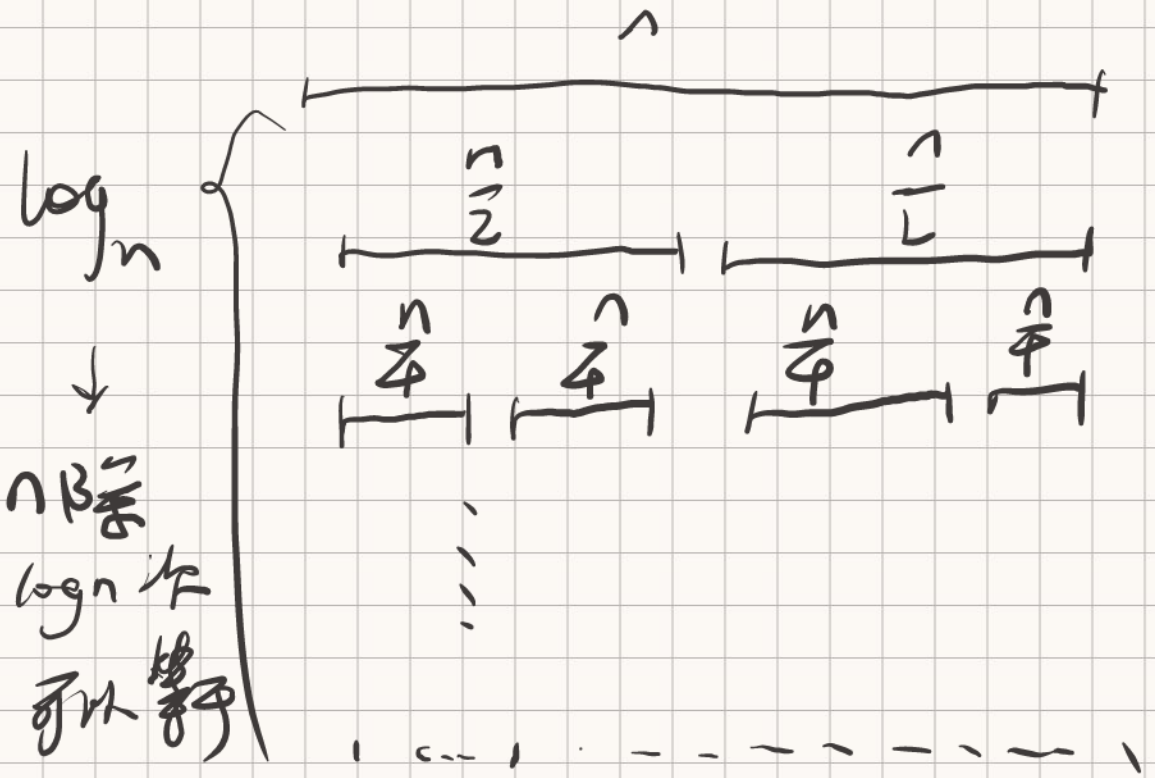
如将小的那个指针向后移1位.

如移动后再比较两个指针的大小

如直到有一个指针到终点为止.

如把多个还剩下的数存进res[]

时间复杂度：nlogn



代码实现：

```
using namespace std;

const int N = 1000010;

int n;
int q[N], tmp[N];

void merge_sort(int q[], int l, int r)
{
    if (l >= r) 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 (i = l, j = 0; i <= r; i ++, j ++ ) q[i] = tmp[j];
}

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]);

    return 0;
}
```