快速排序

基于分治

① 确定分界点 1° q[l] 2° q[(l+r)/2] 3° q[r] 4° 随机

② 调整区间 | ≤x | ≥x | 划分成两个区间 ☆

③ 递归处理左右两端

关于②的思路

开两个空间 a[] b[] 遍历 q, ≤x 放入 a, >x 放入 b, 最终 a、b 写回 q

双指针

i → 走到一个 ≥x 的数, j ← 走到一个 ≤x 的数

当 i、j 停下来的时候, swap 一下

任意时刻, i 左边的数一定 ≤x, j 右边的数一定 ≥x , i、j 相遇的时候, 停下来划分

```cpp
#include <stdio.h>
#include <iostream>
using namespace std;

const int N = 1e6 + 10;
int q[N];
int n;

void qsort(int *q, int l, int r) {
    if(l >= r) return ;

    int x = q[(l + r) / 2];
    int i = l - 1, j = r + 1;   // 先放在边界的两侧
    while(i < j) {
        do i ++; while(q[i] < x);  // do while,无论什么情况，都要让两个指针先移动
        do j --; while(q[j] > x);  // 这样可以避免卡死
        if(i < j) swap(q[i], q[j]);  // 只有没相遇的时候才需要交换
    }
    qsort(q, l, j); qsort(q, j + 1, r);  // 这里注意，以j为划分后的边界指针
    // 如果是用i的话，就应该是 i - 1,  i为分界
    // 同时x的选定也要调整，可以是(l + r + 1) / 2,向上取整，一定不能取到l为边界，会死循环
    // 1 2的例子，1为分界，结束后i不会移动，停在0，之后 [i, r]还是[0, 1]，就死循环了

    //同理，选用j为分界的时候，  j, j + 1，x就不能取到q[r]
    // 1 2   2为分界，结束后j不移动，[l, j]还是[0, 1]，死循环了
    // 同一个区间被无限递归
    return ;
}

int main() {
    scanf("%d", &n);
    for(int i = 0; i < n; i++) scanf("%d", &q[i]);
    qsort(q, 0, n - 1);
    for(int i = 0; i < n; i++) printf("%d ", q[i]);
    return 0;
}
```

786. 第k个数

题目 | 提交记录 | 讨论 | 题解 | 视频讲解

给定一个长度为 n 的整数数列，以及一个整数 k，请用快速选择算法求出数列从小到大排序后的第 k 个数。

输入格式

第一行包含两个整数 n 和 k。

第二行包含 n 个整数（所有整数均在 $1 \sim 10^9$ 范围内），表示整数数列。

输出格式

输出一个整数，表示数列的第 k 小数。

数据范围

$1 \le n \le 100000$,
$1 \le k \le n$

输入样例：

```
5 3
2 4 1 5 3
```

输出样例：

```
3
```

难度：简单
时/空限制：1s / 64MB
总通过数：204084
总尝试数：347337
来源：模板题
算法标签

```cpp
#include <stdio.h>
#include <iostream>
using namespace std;

const int N = 1e6 + 10;
int q[N];
int n, k;

void qsort(int *q, int l, int r) {
    if (l >= r)
        return ;
    int x = q[l + (r - l) / 2];
    int 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]);
    }
    qsort(q, l, j);
    qsort(q, j + 1, r);
    return ;
}

int main() {
    scanf("%d %d", &n, &k);
    for (int i = 0; i < n; i++)
        scanf("%d", &q[i]);
    qsort(q, 0, n - 1);
    printf("%d", q[k - 1]);
    return 0;
}
```

排序完打印即可

r0tten