

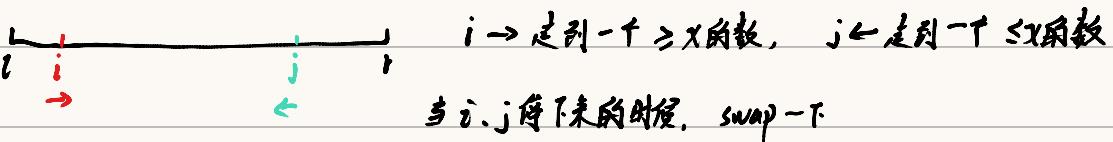
快速排序

基于分治

- ① 确定分界点 $q[l]$ $q[\frac{l+r}{2}]$ $q[r]$ 随机
- ② 调整区间  划分成两个区间 \star
- ③ 递归处理左右两端

关于②的思路

小孩分开两个空间 $a[]$ $b[]$ 遍历 q , $\leq x$ 放入 a , $> x$ 放入 b , 最终 a, b 合回 q

用双指针 

注意时刻, i左边的数一定 $\leq x$, j右边的数一定 $\geq x$. i, j相遇的时候, 原来划分

```

1 #include <stdio.h>
2 #include <iostream>
3 using namespace std;
4
5 const int N = 1e6 + 10;
6 int q[N];
7 int n;
8
9 void qsort(int *q, int l, int r) {
10    if(l >= r) return ;
11
12    int x = q[(l + r) / 2];
13    int i = l - 1, j = r + 1; // 先放在边界的两侧
14    while(i < j) {
15        do i++; while(q[i] < x); // do while, 无论什么情况, 都要让两个指针先移动
16        do j--; while(q[j] > x); // 这样可以避免卡死
17        if(i < j) swap(q[i], q[j]); // 只有没相遇的时候才需要交换
18    }
19    qsort(q, l, j); qsort(q, j + 1, r); // 这里注意, 以j为划分后的边界指针
20    // 如果是用i的话, 就应该是 i - 1, i为分界
21    // 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
22    // 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
23
24    // 同理, 选用j为分界的时候, j, j + 1, x就不能取到q[r]
25    // 1 2 2为分界, 结束后j不移动, [l, j]还是[0, 1], 死循环了
26    // 同一个区间被无限递归
27    return ;
28}
29
30 int main() {
31    scanf("%d", &n);
32    for(int i = 0; i < n; i++) scanf("%d", &q[i]);
33    qsort(q, 0, n - 1);
34    for(int i = 0; i < n; i++) printf("%d ", q[i]);
35    return 0;
36}

```

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

$1 \leq n \leq 100000$,

$1 \leq k \leq n$

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▼

```
1 #include <stdio.h>  
2 #include <iostream>  
3 using namespace std;  
4  
5 const int N = 1e6 + 10;  
6 int q[N];  
7 int n, k;  
8  
9 void qsort(int *q, int l, int r) {  
10    if (l >= r)  
11        return;  
12    int x = q[l + (r - l) / 2];  
13    int i = l - 1, j = r + 1;  
14    while (i < j) {  
15        do  
16            i++;  
17        while (q[i] < x);  
18        do  
19            j--;  
20        while (q[j] > x);  
21        if (i < j)  
22            swap(q[i], q[j]);  
23    }  
24    qsort(q, l, j);  
25    qsort(q, j + 1, r);  
26    return;  
27}  
28  
29 int main() {  
30    scanf("%d %d", &n, &k);  
31    for (int i = 0; i < n; i++)  
32        scanf("%d", &q[i]);  
33    qsort(q, 0, n - 1);  
34    printf("%d", q[k - 1]);  
35    return 0;  
36}
```

排序完打印即可

