[9] 排序

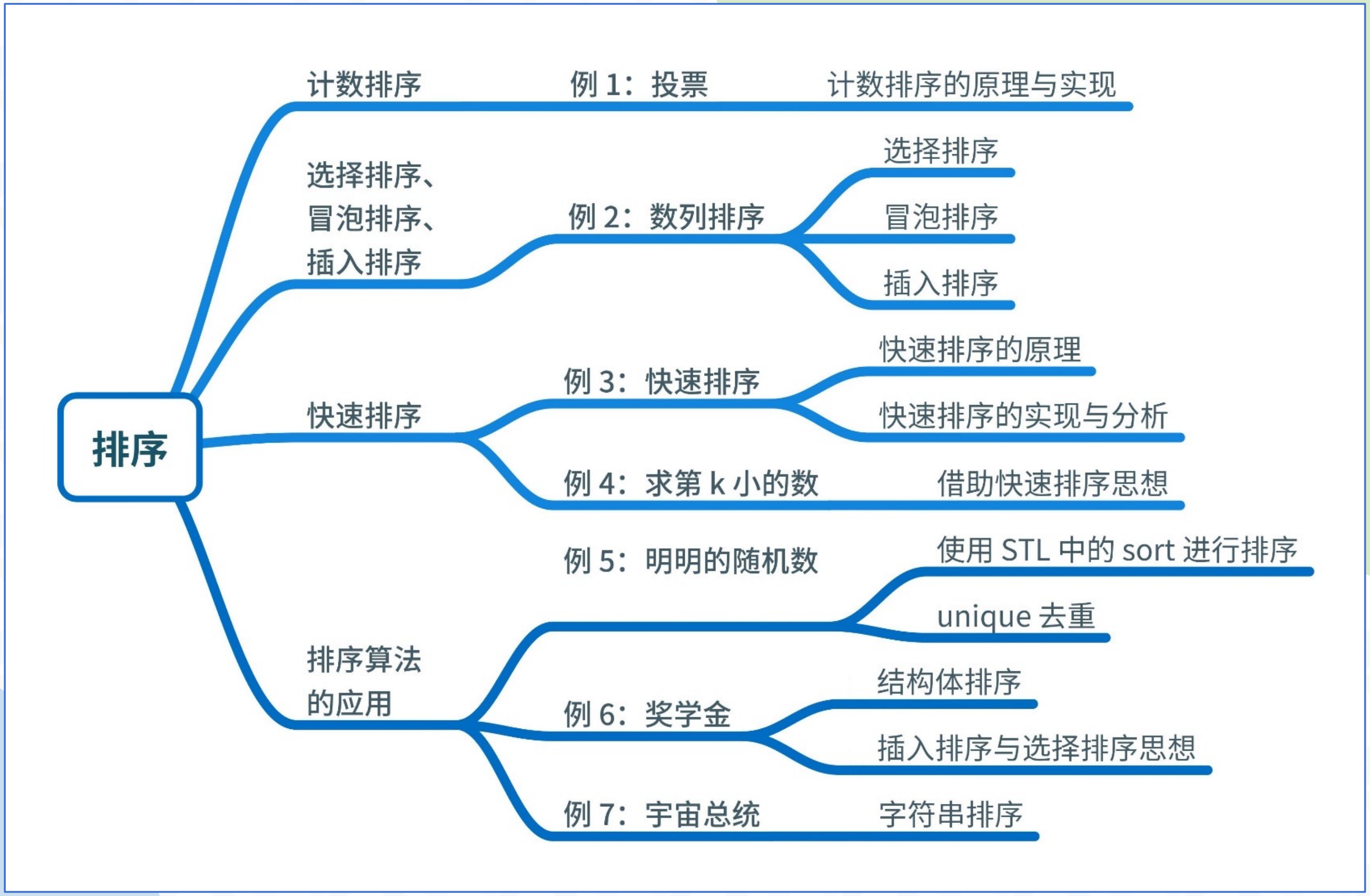
深入浅出程序设计竞赛
第 2 部分 – 初涉算法
V 2021-02

洛谷
www.luogu.com.cn

版权声明

- 其它《深基》配套资源、购买本书等请参阅：https://www.luogu.com.cn/blog/kkksc03/IPC-resources
- 如果课件有任何错误，请在这里反馈 https://www.luogu.com.cn/discuss/show/296741

本章知识导图


排序
计数排序
例 1：投票
计数排序的原理与实现
选择排序、
冒泡排序、
插入排序
例 2：数列排序
选择排序
冒泡排序
插入排序
快速排序
例 3：快速排序
快速排序的原理
快速排序的实现与分析
例 4：求第 k 小的数
借助快速排序思想
排序算法
的应用
例 5：明明的随机数
使用 STL 中的 sort 进行排序
unique 去重
例 6：奖学金
结构体排序
插入排序与选择排序思想
例 7：宇宙总统
字符串排序

第 9 章 排序

- 计数排序
- 选择排序、冒泡排序、插入排序
- 快速排序
- 排序算法的应用
- 课后习题与实验

排序问题

在现实生活中，经常需要对事物或信息进行排序。

具体而言，我们收集到的数据大多是无序的，而我们需要有序的结果作为产出或作为后续步骤的输入。

有序性是一个很好的规律。例如，若一个序列是有序的，我们可以在O(1)时间内查找第k小的元素，但对于无序序列就很困难。

将无序的序列转变为有序，即是排序问题。

本章将介绍若干种排序算法。我们将会看到，不同算法均有各自适应的应用场景，并非只需掌握一种即可。

计数排序

“身高为150的站一排，151的站一排…… 189的站一排，190的站一排。”

“然后从前到后合成一条队，对，这样就排好了。”

请翻至课本 P129

选举学生会

例 9.1 （洛谷 P1271）

学校正在选举学生会成员，有 $n(n \le 999)$ 名候选人，每名候选人编号分别从 1 到 n，现在收集到了 $m(m < 2000000)$ 张选票，每张选票都写了一个候选人编号。

现在想把这些堆积如山的选票按照投票数字从小到大排序。

输入 n 和 m 以及 m 个选票上的数字，求出排序后的选票编号。

样例输入：

```
2 5 2 2 5 2 2 2 1 2
```

样例输出：

```
1 2 2 2 2 2 2 2 5 5
```

选举学生会

显然，题意即是经典的排序问题。输入无序序列，输出有序序列。

我们可以考虑现实中如何（匿名）计票：

1、取出一张票（可以按照任意顺序，一般按照收集顺序即可）。

2、将其放入与其编号相同的票箱，且该堆的计数+1。

例如样例：2 5 2 2 5 2 2 2 1 2

票箱#1	票箱#2	票箱#3	票箱#4	票箱#5
1	2			5
	2			5
	2			
	2			
	2			
	2			
	2			
1	7	0	0	2

选举学生会

因为是匿名计票，所以选票本身的特殊性并不重要；表格中实际有效的只有总计数值！

票箱#1	票箱#2	票箱#3	票箱#4	票箱#5
1	7	0	0	2

为了得到有序序列，我们只需要按照票箱号从小到大从票箱中拿出选票即可；此时我们有1张#1、7张#2和2张#5，因此序列为：

1 | 2222222 | | | 55

即为所求。

注意中间的隔断“|”：我们依然可以“尝试”从3、4号票箱取票；但因为是空的，所以没有取出内容。

选举学生会

使用数组模拟票箱。根据刚才的结论，我们只需要记载票箱内的选票数即可。

```
int a[1010] = {0};
```

接下来，我们按输入顺序读取每一张票，并置入票箱：

```
for (int i = 0; i < m; i++) {
    cin >> tmp;
    a[tmp]++;
}
```

最后，再按照票箱编号顺序取出所有选票*：

```
for(int j = 1; j <= n; j++)
    for(int i = 0; i < a[j]; i++)
        cout << j << ' ';
```

为了表述清晰，我们用循环变量 i 代表选票，j 代表票箱。

计数排序

计数排序 顾名思义，即统计每一个元素出现的次数，再按照顺序依次排列。数列中的元素就是“票”，而一个与元素取值范围相符的数组就是“票箱”。

记n为数列长度，m为取值范围。

需要$O(n)$的时间统计每一数值出现次数。之后再用$O(n+m)$的时间构造出结果数列，总时间$O(n+m)$。

另外，需要$O(m)$的额外空间作为票箱。

优点：当m较小时，时间复杂度近似于$O(n)$，性能强大。

缺点：当m远大于n时，时空复杂度均取决于m，得不偿失。

取值范围为非整数时，无法实现。

计数排序变种

离散化计数排序

若能将不可表示的数据范围（双向）映射到较小的整数集合上，则可以在映射后使用计数排序。这一映射过程称为离散化。

桶排序

将数列按数值区间（而非具体数值）划分为若干个桶。桶内采用其他排序算法。

基数排序

从低到高对每一个（X进制）位进行一次计数排序。这样，当高位有序时，所有低位均已有序。可以保证只使用X个桶。

此处不要求掌握。

选择/冒泡/插入排序

现在我们开始玩牌。你要如何整理你的手牌呢？

请翻至课本 P131

基于交换的排序方法

我都是把相同的数字牌扣在桌面上来排序的！
这不还是一种计数排序吗？

没错，但是这样你的对手就可以通过观看牌桌上倒扣的牌堆推测你的牌型。所以你决定只通过交换的方式来排序。

逆序对：假设排序应当是从小到大，如果有一对牌，前面的那张牌比后面的那张牌数字大，那么就是逆序对。

基于交换的排序方法

所谓交换，即数列元素两两比较；若顺序反了则对换二者位置。

由于每一次交换逆序对数必然减少，持续交换必然可以完成排序，且具有不用到其他变量而直接在原数列中操作、额外空间复杂度$O(1)$的优点。

考虑到元素两两间均需要互相比较，可以大致估计出，此类方法最坏时间复杂度均约为$O(n^2)$。

选择排序

思路：n次大循环，第i次大循环中，用小循环寻找数列中第i小的元素。如果发现更小的数字，则交换至第i个位置。

```
for (int i = 0; i < n - 1; i++)
    for (int j = i + 1; j < n; j++)
        if (a[j] < a[i])
            swap(a[i], a[j]);
```

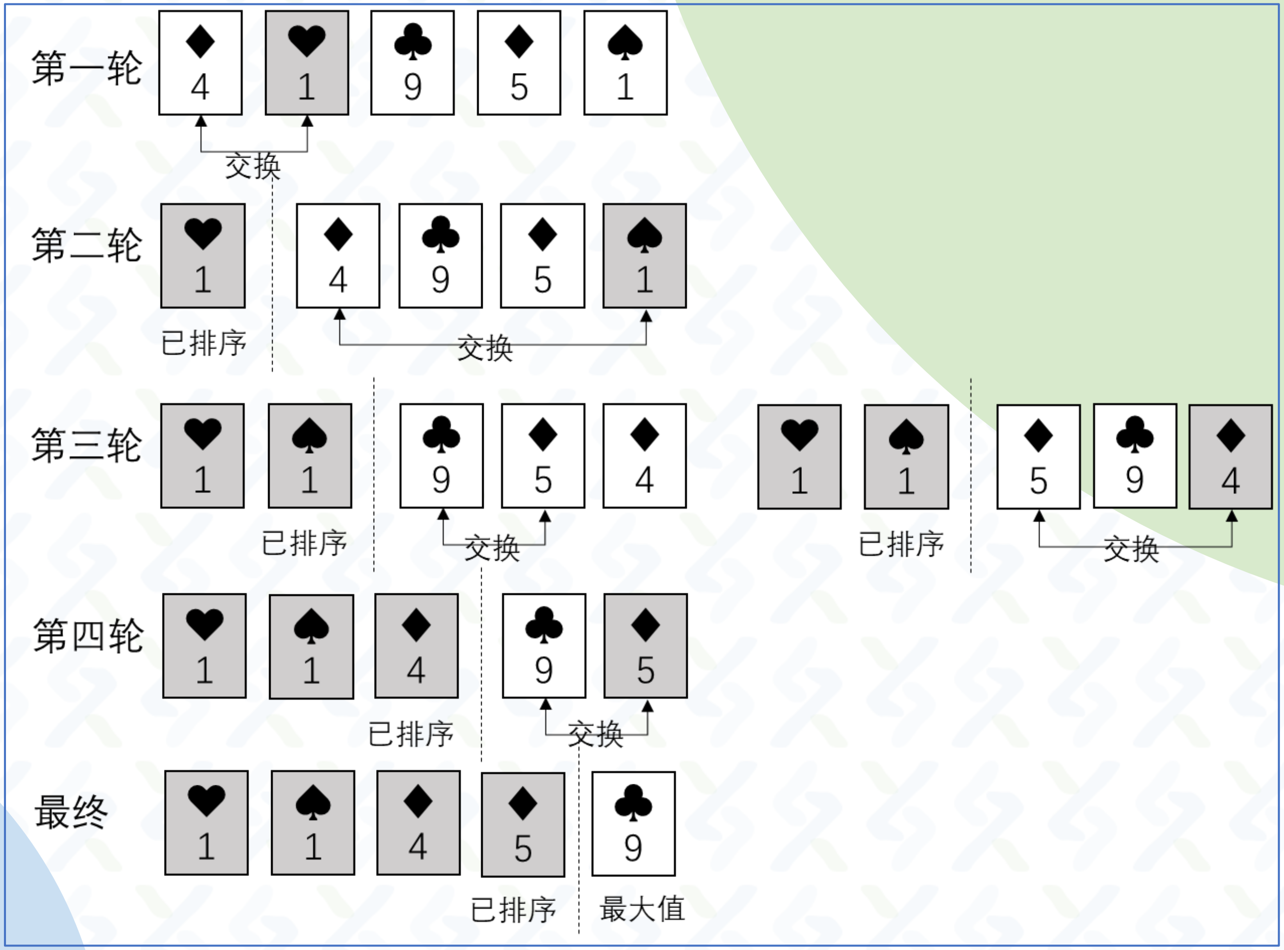
事实上，由于前i-1项已经排序完毕，第i小的元素等价于第i至第n项中的最小元素。

特点：思路简单，实现更简单。

每次迭代都能保证至少一个（最小）元素的位置被确定。

前i个元素有序，且为最小的i个元素。

选择排序



第一轮
4
1
9
5
1
交换
第二轮
1
4
9
5
1
已排序
交换
第三轮
1
1
9
5
4
已排序
交换
1
1
5
9
4
已排序
交换
第四轮
1
1
4
9
5
已排序
交换
最终
1
1
4
5
9
已排序
最大值

冒泡排序

思路：不超过n次大循环；每次大循环，按照顺序比较相邻元素并交换，直到序列有序。

```
for (int i = 0; i < n - 1; i++)
    for (int j = 0; j < n - i - 1; j++)
        if (a[j] > a[j + 1])
            swap(a[j], a[j + 1]);
```

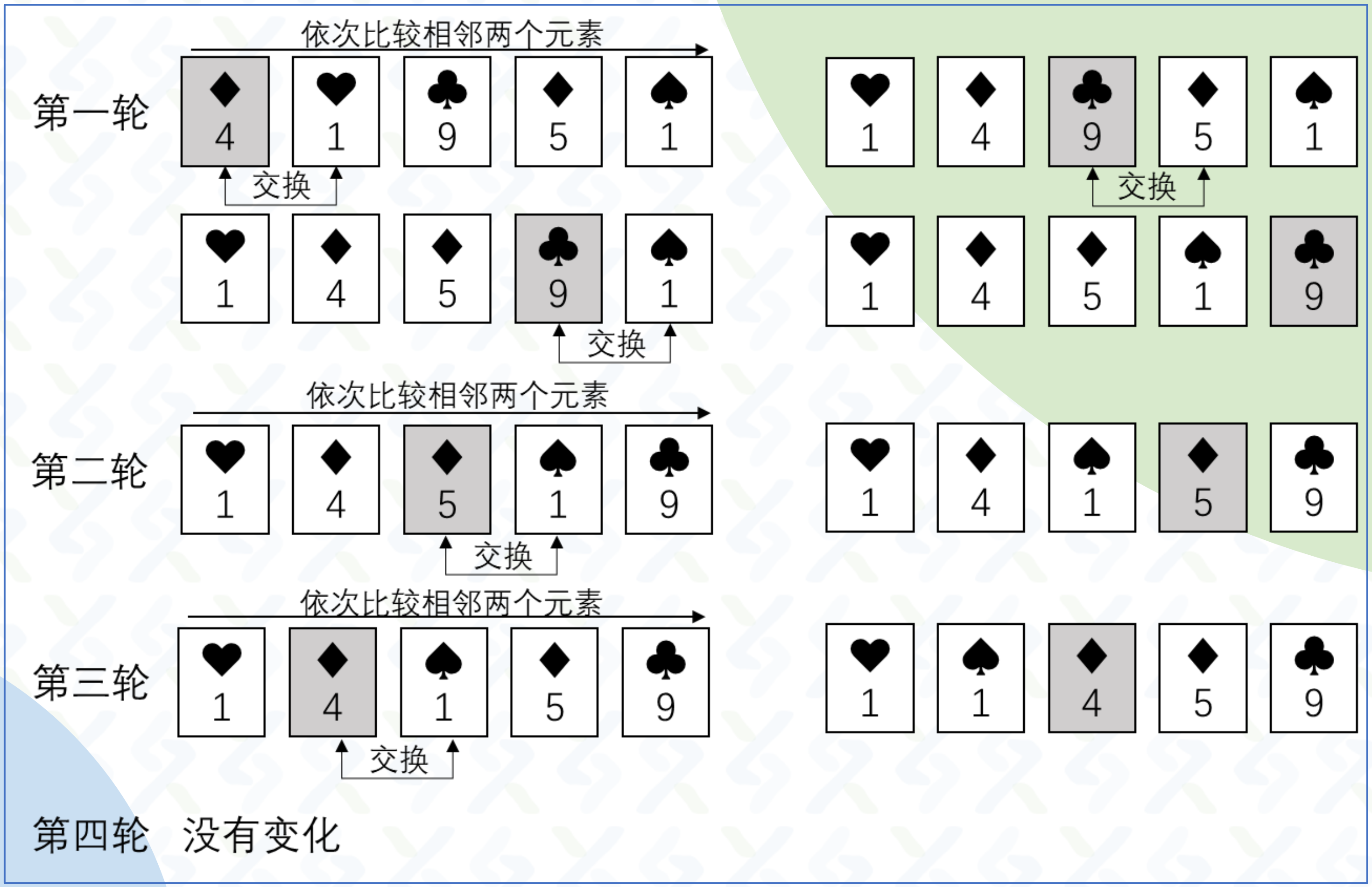
特点：

总交换次数恰为逆序对数。

每次迭代都能保证至少一个（最大）元素的位置被确定。

后i个元素有序，且为最大的i个元素。

冒泡排序


依次比较相邻两个元素
第一轮
4
1
9
5
1
交换
1
4
9
5
1
交换
1
4
5
9
1
交换
1
4
5
1
9
依次比较相邻两个元素
第二轮
1
4
5
1
9
交换
1
4
1
5
9
依次比较相邻两个元素
第三轮
1
4
1
5
9
交换
1
1
4
5
9
第四轮　没有变化

插入排序

思路：n次大循环；第i次大循环将第i个元素向前交换，直至左侧元素不大于它，或抵达数列首部。

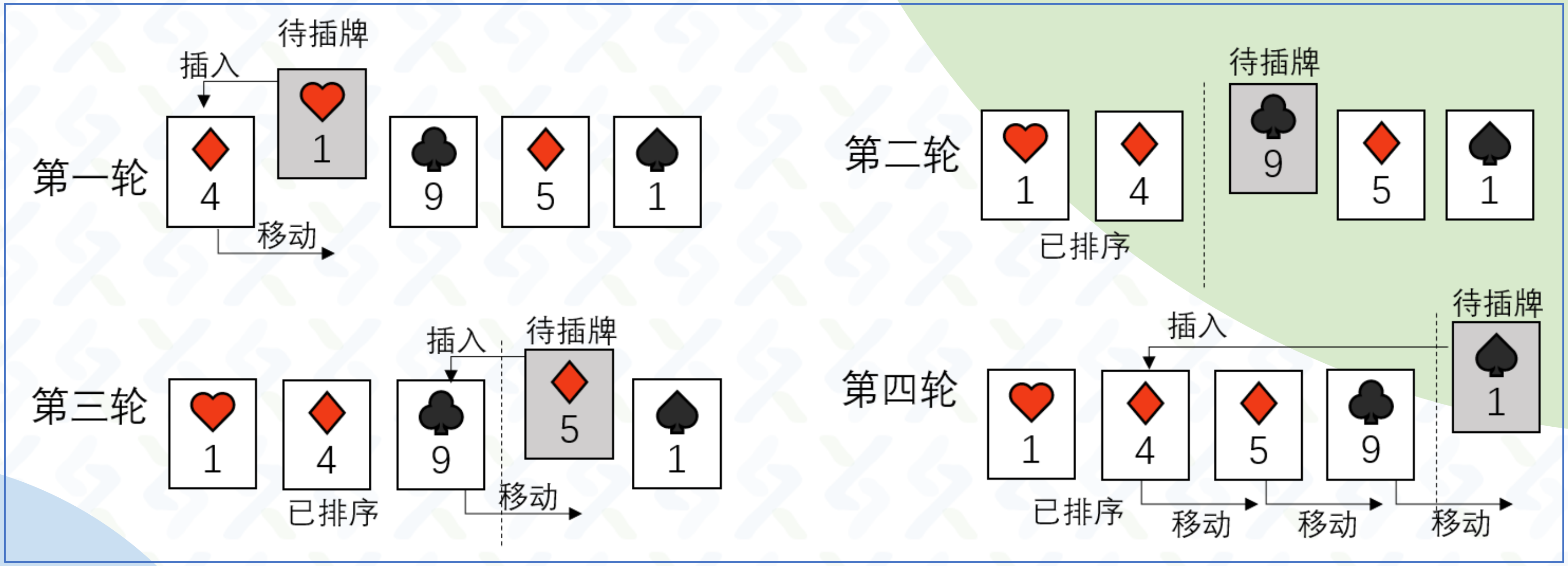
```
for (int i = 1; i < n; i++) {
    int now = a[i], j; // 记录一下待插牌，等下还要放回去
    for (int j = i - 1; j >= 0; j--)
        if (a[j] > now)
            a[j + 1] = a[j];
        else break;
    a[j + 1] = now;
}
```

特点：前i个元素有序。但是直到排序完成，不能保证任何一个元素的最终位置被确定（设想最小元素在数列尾部）。

可以用来动态维护前k小元素，单次插入时间复杂度$O(k)$。在此场景下，第$k+1$小的元素将不会右移，而是被直接丢弃。

~~可以用来理牌。~~

插入排序



第一轮
插入
待插牌
4
1
9
5
1
移动
第二轮
1
4
已排序
待插牌
9
5
1
第三轮
1
4
已排序
插入
9
待插牌
5
1
移动
第四轮
插入
1
4
5
9
待插牌
1
已排序
移动
移动
移动

三种排序：应用场景

是否存在性能比 $O(n^2)$ 更优的排序算法呢？

答案是肯定的（将在下一节介绍），因此在实际应用中，很少使用以上三种算法进行单纯的排序需求。

但这并不意味着它们是无用的，整理一下应用场景：

选择排序：k 很小时快速求解前 k 小/大元素

冒泡排序：模拟逆序对相关的问题

插入排序：k 很小时动态维护前 k 小/大元素

快速排序

为什么基于交换的排序需要$O(n^2)$时间？因为理论上任意两个不同元素都需要比较，共$\frac{n(n-1)}{2}$次。是否可以减少必须的比较次数呢？

请翻至课本 P134

快速排序

选择排序，每一迭代确定一个元素的位置。

但是所有已确定元素均在左侧，对右侧没有指导意义。

插入排序，每一迭代只需要与大于自身的值比较，期望只需比较一半的元素。

但是没有元素确定位置；新元素的插入导致所有更大的元素右移，带来额外开销。

考虑结合二者优点。以这个数列为例：

```
3 8 4 10 6 7 2 5 9 1
```

快速排序

尝试随机选取一个元素确认位置，称为哨兵数。

3 8 4 10 6 7 2 5 9 1

将序列中所有比哨兵数小的数字都移动到哨兵数的左边，所有比哨兵数大的数字都移动到哨兵数的右边。

3 4 2 5 1 6 8 10 7 9

显然哨兵数左右两侧的元素不再需要任何比较。因此，对两侧的子数列分别采用相同的做法，直到数列不可再分（长度为0或1）。

3	4	2	5	1	6	8	10	7	9
1	2	3	4	5	6	8	7	9	10
1	2	3	4	5	6	8	7	9	10
1	2	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10

快速排序

那么这样做到底有多“快”呢？

在最优情况下，每次选择的哨兵数均将数列对半分开。则$O(\log_2(n))$次划分后数列将不可再分。每次划分的复杂度为$O(n)$，总复杂度$O(n\log_2(n))$。

在最坏情况下，每次选择的哨兵数均在数列一端。则退化为选择排序算法，总复杂度$O(n^2)$。

小贴士：什么是对数（log）

刚刚我们提到了$\log_2 n$。数学上称为对数函数。它是什么意思？

形式定义：若$a^x = b$，则$\log_a b = x$。

形象理解：我们可以估计对数函数的上界

<table>
<tr><td>0</td><td colspan="10">10 $(1010)_2$</td></tr>
<tr><td>1</td><td colspan="5">5 $(101)_2$</td><td colspan="5">5 $(101)_2$</td></tr>
<tr><td>2</td><td colspan="3">3 $(11)_2$</td><td colspan="2">2 $(10)_2$</td><td colspan="3">3 $(11)_2$</td><td colspan="2">2 $(10)_2$</td></tr>
<tr><td>3</td><td>1</td><td colspan="2">2 $(10)_2$</td><td>1</td><td>1</td><td>1</td><td colspan="2">2 $(10)_2$</td><td>1</td><td>1</td></tr>
<tr><td>4</td><td></td><td>1</td><td>1</td><td colspan="3"></td><td>1</td><td>1</td><td colspan="2"></td></tr>
</table>

将长度为10的序列折半划分需要4折（10的2进制表示有4位），所以$\log_2 10 < 4$。

$2147483648 = 2^{31}$，因此$\log_2(2147483648) = 31$。可见对数增长十分缓慢，相比$O(n)$是一个非常优秀的复杂度。

小贴士：常见复杂度与数据范围

复杂度	常见范围	冒险范围
$O(n)$	10^7	10^8
$O(n\log n)$	3×10^5	10^6
$O(n\log^2 n)$	10^5	10^6
$O(n^2)$	5×10^3	10^4
$O(n^2\log n)$	10^3	3×10^3
$O(n^3)$	100	400
$O(2^n)$	20	25
$O(n2^n)$	15	20
$O(n!)$	10	10

快速排序

此处给出一种实现。并不存在一种在所有情况下都最优的实现，建议自行了解，根据应用场景选择合适的实现。

```
void qsort(int a[], int l, int r) { // 引入数组的地址
    int i = l, j = r, flag = a[(l + r) / 2], tmp;  // flag=哨兵
    do {
        while (a[i] < flag) i++; // 从左找比哨兵大的数
        while (a[j] > flag) j--; // 从右找比哨兵小的数
        if (i <= j) { // 交换
            swap(a[i], a[j]);
            i++; j--;
        }
    } while (i <= j);
    // 习惯上，上面用于分段的过程一般称作partition
    if (l < j) qsort(a, l, j);
    if (i < r) qsort(a, i, r);
}
```

快速排序

快排是一种基于分治法的排序算法。

大多数基于分治法的算法均具有O(nlogn)的时间复杂度。

其余具有此复杂度的排序算法还有：

基数排序，基于按位处理

归并排序，基于分治法

堆排序，基于数据结构

求第k小的数

例9.4（洛谷 P1923）

输入 $n(n < 5000000$ 且 n 为奇数) 个数字 $a_i(a_i < 10^9)$，输出这些数字的第 k 小的数。最小的数是第 0 小。

样例输入：

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

样例输出：

```
2
```

求第k小的数

如果 k 很小（常数级别），那么可以用选择排序直接把第 k 大的数选出来。然而并没有这个保证。

如果直接排序，我们已经知道可以在O($n\ln n$)的时间复杂度内使用快速排序求出。然而会超时

提示：先给出一个大家都知道的结论：

$$1+\frac{1}{2}+\frac{1}{4}+\frac{1}{8}+\cdots=2$$

求第k小的数

借用快速排序中partition的思想。

每次迭代，随机选取一个哨兵，将序列一分为二：左侧比哨兵小，右侧比哨兵大。此时哨兵数的位置就是它排序后的顺位。

若这个序位等于k，那就是它了。

若这个序位大于k，就在左侧寻找。

若这个序位小于k，则在右侧寻找。

注意当选定一侧后，另一侧已不再可能成为答案，可以直接丢弃。

最优复杂度：$O\left(n+\frac{n}{2}+\frac{n}{4}+\cdots\right)=O(n)$

均摊复杂度：$O\left(n+\frac{n}{2}+\frac{n}{4}+\cdots\right)=O(n)$（想一想，为什么？）

排序算法的应用

养兵千日，用兵一时。排序作为重要的算法工具，可以将无序变有序，使问题更好的得到处理。

请翻至课本 P136

STL中的排序算法

algorithm（算法）库包含很多常用的算法，包括排序。

包含头文件：

```
#include <algorithm>
```

使用排序功能：

```
sort(a.begin(), a.end());          // O(nlogn)
sort(a.begin(), a.end(), cmp);     // O(nlogn)
```

begin、end分别表示需要排序位置的首末。

cmp是一个可选参数，可以自定义排序的比较方法。

排序之后，可以使用以下算法：

去重：

```
unique(a.begin(), a.end());      // O(n)
```

这个函数会返回去重后的序列末尾地址（序列长度可能会变短）。

查找：

```
find(a.begin(), a.end(), val);   // O(logn)
```

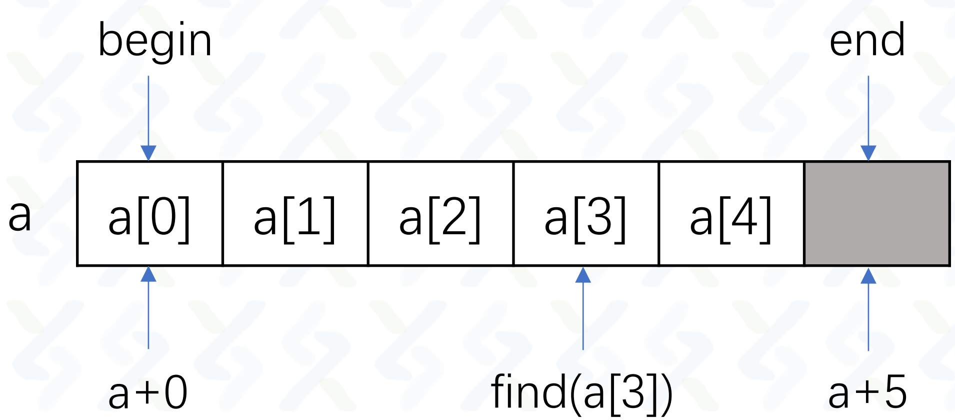
若元素存在则返回元素地址，否则返回末尾地址（end）。

STL中的排序算法

对于数组和 vector（暂时没教） 的数组坐标：

- a+0 = begin
- a+n = end
- a+(n-1) = 最后一个元素
- end – begin = n
- find(a[x]) – a = x

```
sort(a.begin(), a.end());
// 对一个 vector 进行排序
sort(a.begin(), a.end(), cmp);
// 对一个 vector 进行【自定义】排序
sort(a, a+n);
// 对一个长度为 n 的数组 a 排序
sort(a, a+n, cmp);
// 对长度为 n 的数组 a【自定义】排序
```



明明的随机数

例 9.5（洛谷 P1059，NOIP2006 普及组）

给出 $N(N \leq 100)$ 个 1 到 1000 的数字，输出去重后剩余数字的个数，以及去重排序后的序列。

样例输入：

```
10
20 40 32 67 40 20 89 300 400 15
```

样例输出：

```
8
15 20 32 40 67 89 300 400
```

明明的随机数

解法 1：

注意到取值范围很小，可以使用计数排序。

解法 2：

直接使用STL中的排序、去重函数。

```
sort(a, a + n);
cnt = unique(a, a + n) - a;
cout << cnt << endl;
for (int i = 0; i < cnt; i++)
    cout << a[i] << ' ';
```

自定义排序

如果我想排序，但不是从小到大排序，怎么办？
或者，能对结构体元素进行排序吗？

sort 函数可以增加第三个参数 cmp，也就是自定义排序的基准。

cmp 函数需要两个待排序的元素类型 a、b 作为参数。

返回一个 bool 值，表示 x 是否严格小于 y。

例如：实现整数从大到小排序。

```
sort(a, a + n, cmp);
```

```
bool cmp(int x, int y){
    return x > y;
}
```

cmp 函数名称可以任取，保持上下一致即可。

奖学金

例 9.6（洛谷 P1093，NOIP2007 普及组）

给出 n(n≤300) 名学生的语文、数学、英语成绩，这些学生的学号依次是从 1 到 n。需要对这些学生进行排序。

如果总分相同，则语文分数高者名次靠前；如果语文成绩还是一样的，学号小者靠前。输出排名前 5 的学生学号和总分。

```
6
90 67 80
87 66 91
78 89 91
88 99 77
67 89 64
78 89 98
```

```
6 265
4 264
3 258
2 244
1 237
```

奖学金

解法 1：使用 STL 进行排序。

构造一个结构体 student 用户存储学生的各项有用的信息（数学和英语并不重要，可以不用存下来）。注意使用 cmp 来进行比较，当两个学生比较时，排名比较高的学生返回 true。

```cpp
#include <algorithm>
#include <iostream>
using namespace std;
int const MAXN = 310;
int n;
struct student {
    int id, chinese, total;
}a[MAXN];
int cmp(student a, student b) {
    if(a.total != b.total) // 总分先定胜负
        return a.total > b.total;
    if(a.chinese != b.chinese) // 然后比语文
        return a.chinese > b.chinese;
    return a.id < b.id; // 最后比学号
}
// 本书2020年第1到2次印刷中，cmp函数的
// 代码有误，这里已更正，在此致歉
```

```cpp
int main() {
    cin >> n;
    for (int i = 0; i < n; i++) {
      int math, english;
      cin >> a[i].chinese>>math>>english;
      a[i].total=a[i].chinese+math+english;
      a[i].id = i + 1;
    }
    sort(a, a + n, cmp);
    for (int i = 0; i < 5; i++)
      cout<<a[i].id<<" "<<a[i].total<<endl;
    return 0;
}
```

还有参照 选择排序 和 插入排序 思路的其它解法，参见课本 P139。

课后习题与实验

学而时习之，不亦说乎。学而不思则罔，思而不学则殆。——孔子

请翻至课本 P139

复习

排序 输入无序序列，处理后变成有序序列

$O(m)$ 计数排序

$O(n^2)$ 选择排序 冒泡排序 插入排序

$O(n\log n)$ 快速排序 基数排序

对数函数 $\log_a b$，近似为a进制表示下b的位数。

STL中的函数 sort unique find 可以简化编程

作业

实验题

1. 使用计数排序完成例 9.5，并且思考如果数字值域到一亿，还可以使用计数排序吗？
2. 例 9.6 中，参照 P138 解法 2 和解法 3 的思路，分别使用选择排序和插入排序完成这个例题。

习题 9.1 超级书架（洛谷 P2676）

有 $N(N \leq 20000)$ 头牛，都有确定的身高 $h_i(h_i \leq 10000)$。书架高度是 $B(B \leq 2\times10^9)$。

现在要选出最少头牛，使它们的身高之和不小于书架高度。

作业

习题 9.2 车厢重组（洛谷 P1116）

$N(N \leq 10000)$ 节车厢，给出初始的车厢顺序；每次可以交换相邻两节车厢，计算最少用多少步就能将车厢排序。

习题 9.6 生日（洛谷 P1104）

给出 $n(n \leq 100)$位同学的姓名（长度不超过 20 的字符串）和他们的生日（年、月、日），请将这些同学按照年龄从大到小进行排序。

习题 9.7 拼数（洛谷 P1012，NOIP1998）

将n个整数拼成一个多位整数，使其数值尽可能大。

参考阅读材料

以下的内容限于课件篇幅未能详细阐述。如果学有余力，可自行翻阅课本作为扩展学习。

- P139 例 9.6：解法 2、解法 3，或用选择排序和插入排序
- P139 例 9.7：如何去对一组高精度数字进行排序
- 习题 9.3、9.4、9.5。