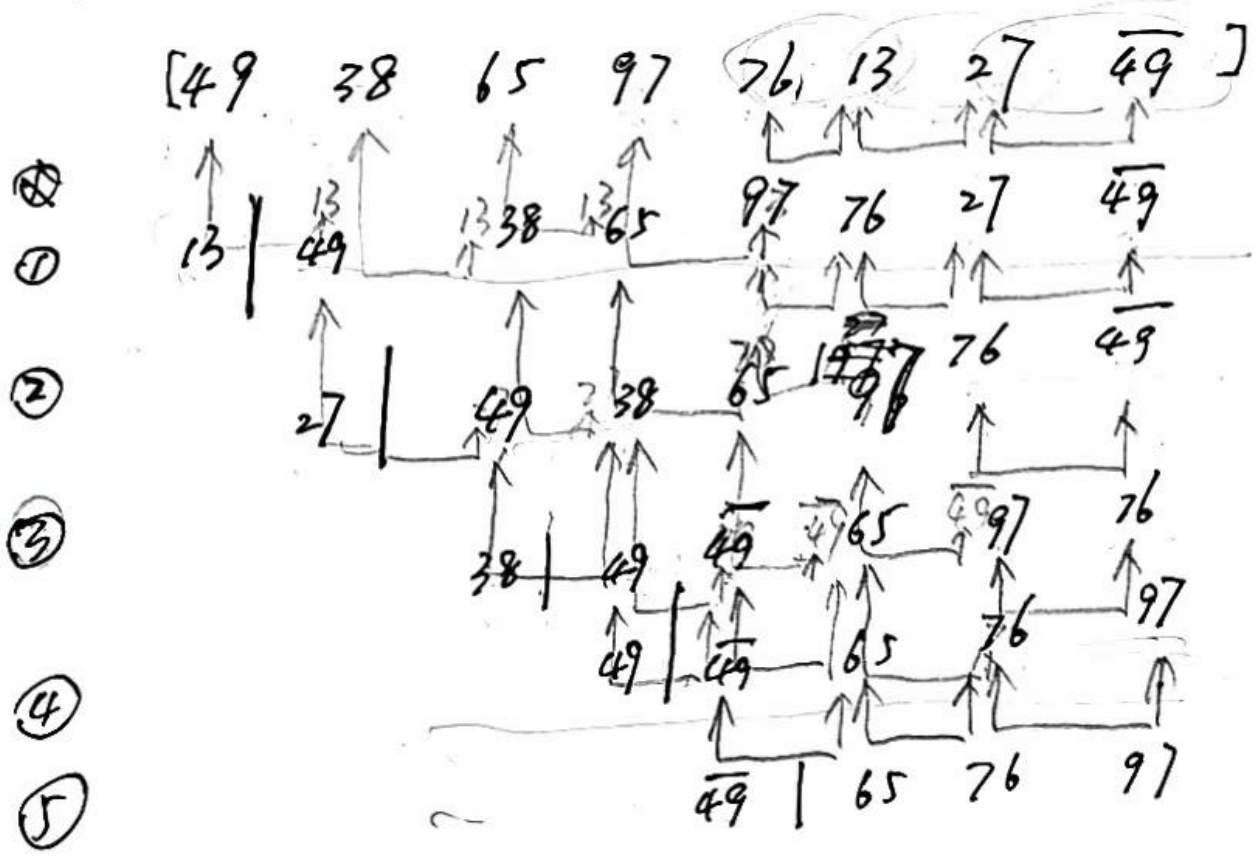
交换类排序

1. 冒泡排序法



结束：

若某趟排序未经过交换，

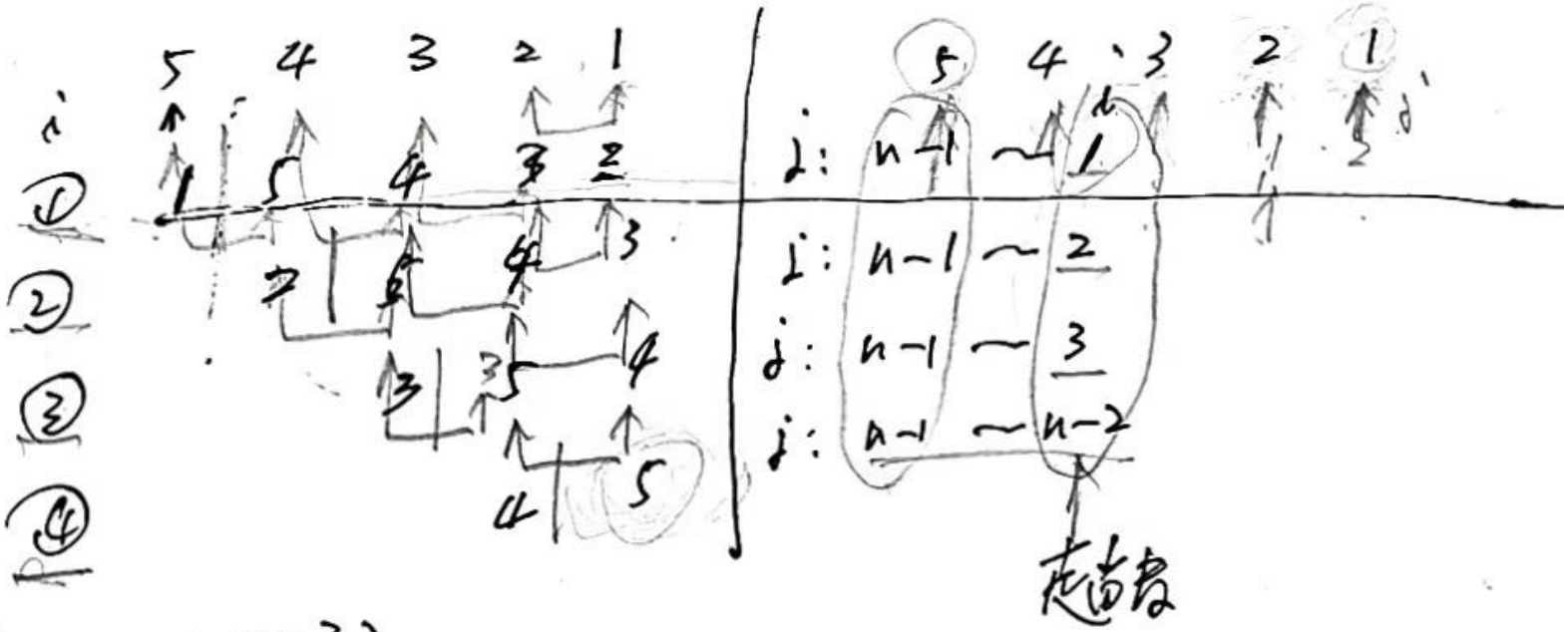
实现：设标志

flag=0;

进行一趟排序

if(!flag) break;

总结：最多需要 $n-1$ 趟排序 $\Rightarrow$ 最多比较 $\frac{n(n-1)}{2}$



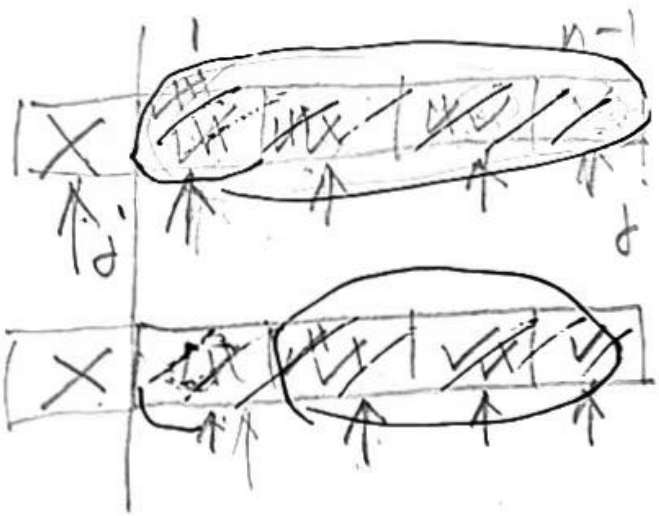
时间：$O(n^2)$

稳定性：稳定

实现：因为最多可能要进行 n-1 趟比较，

所以要搭建一个 n-1 趟的循环。

某一趟：



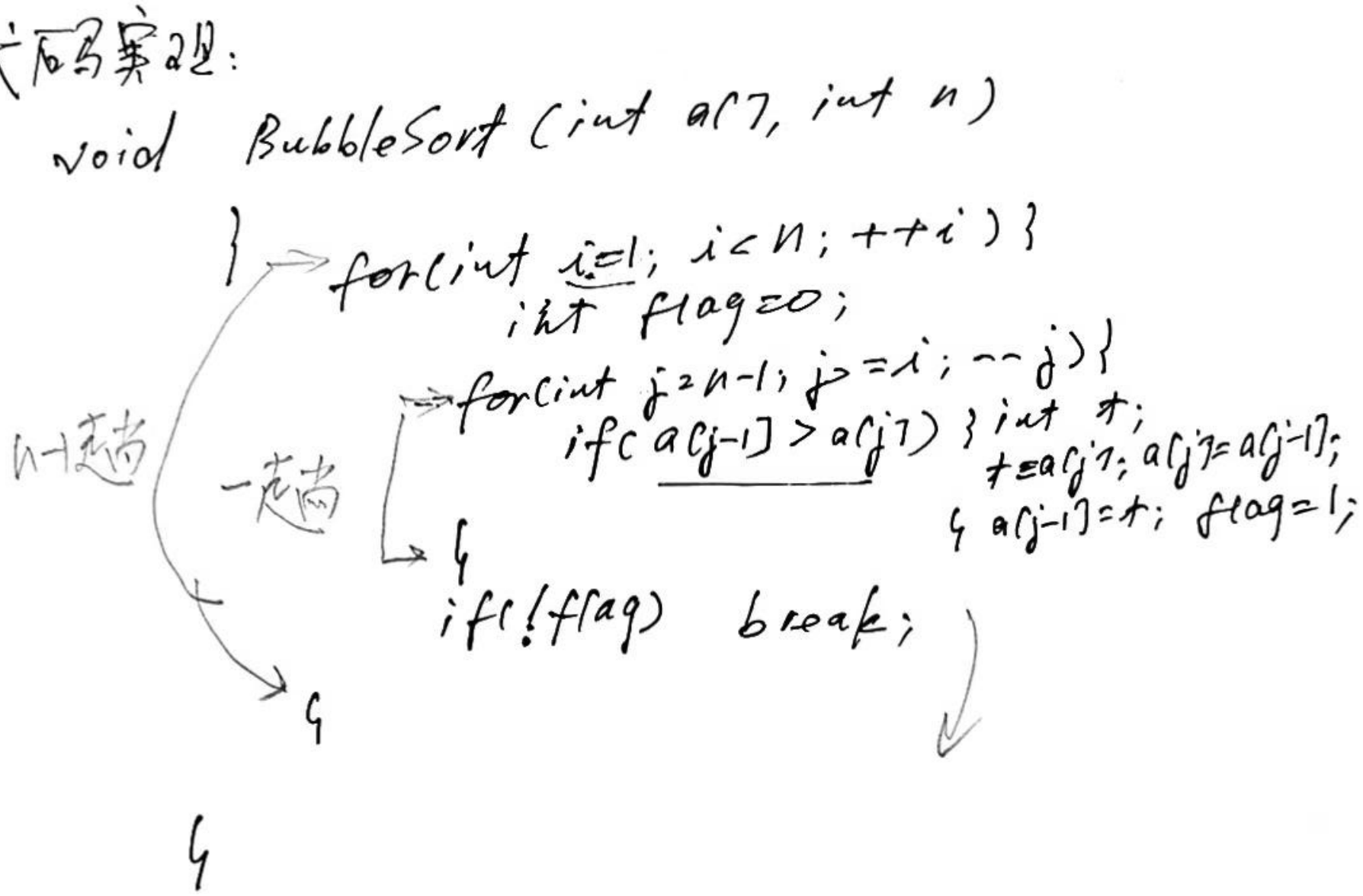
j: n-1 ~ ①

j: n-1 — ②

```
flag=0;
for(j=n-1; j>=i; --j){
    if(a[j]<a[j-1]){
        a[j]⇔a[j-1]
        flag=1;
    }
}
if(!flag) break;
```

③

代码实现：


void BubbleSort(int a[], int n)
{
for(int i=1; i<n; ++i){
int flag=0;
for(int j=n-1; j>=i; --j){
if(a[j-1]>a[j]){int t;
t=a[j]; a[j]=a[j-1];
a[j-1]=t; flag=1;}
}
if(!flag) break;
}
}
n-1趟
一趟

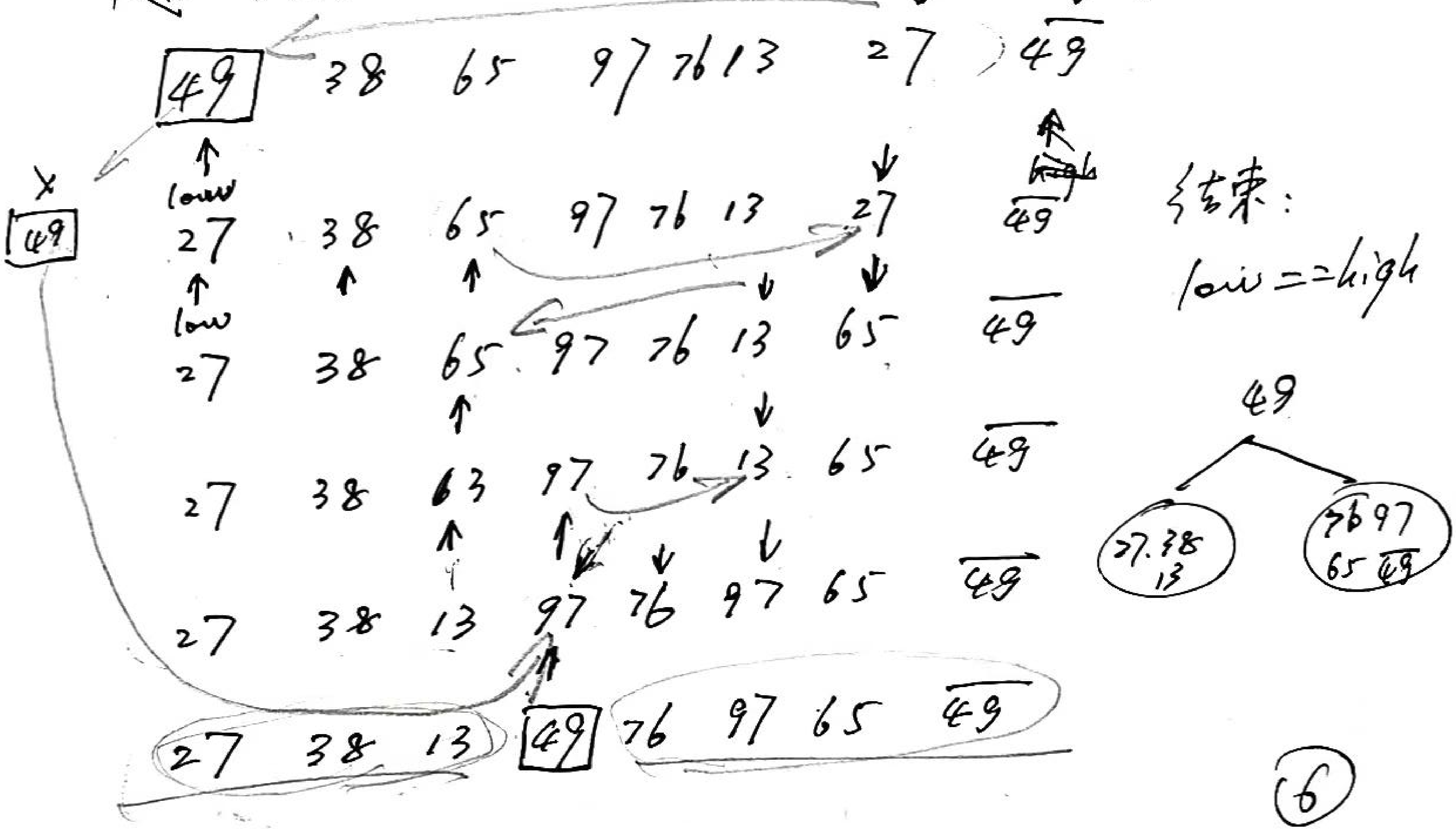
☆ 快速排序（QuickSort）

原因：① 冒泡排序时间复杂度：$O(n^2)$。
—— 线性比较。
② 每次交换两个元素。

改进：数据结构：效率提升：$O(n^2) \Rightarrow O(n\log n)$
② 不交换 ⇒ 赋值

思想：设置一个支点，不断移动数据元素，
使支点左边存储合适的元素、使得支点左边
的值都小于支点，支点右边的值都大于支点

一趟的过程 —— 设置支点：待排区间的第一个元素.



一趟：

x = a[low]

while(low < high){

① high 从后向前找比 x 小的 (low < high)

a[low] = a[high] (low < high)

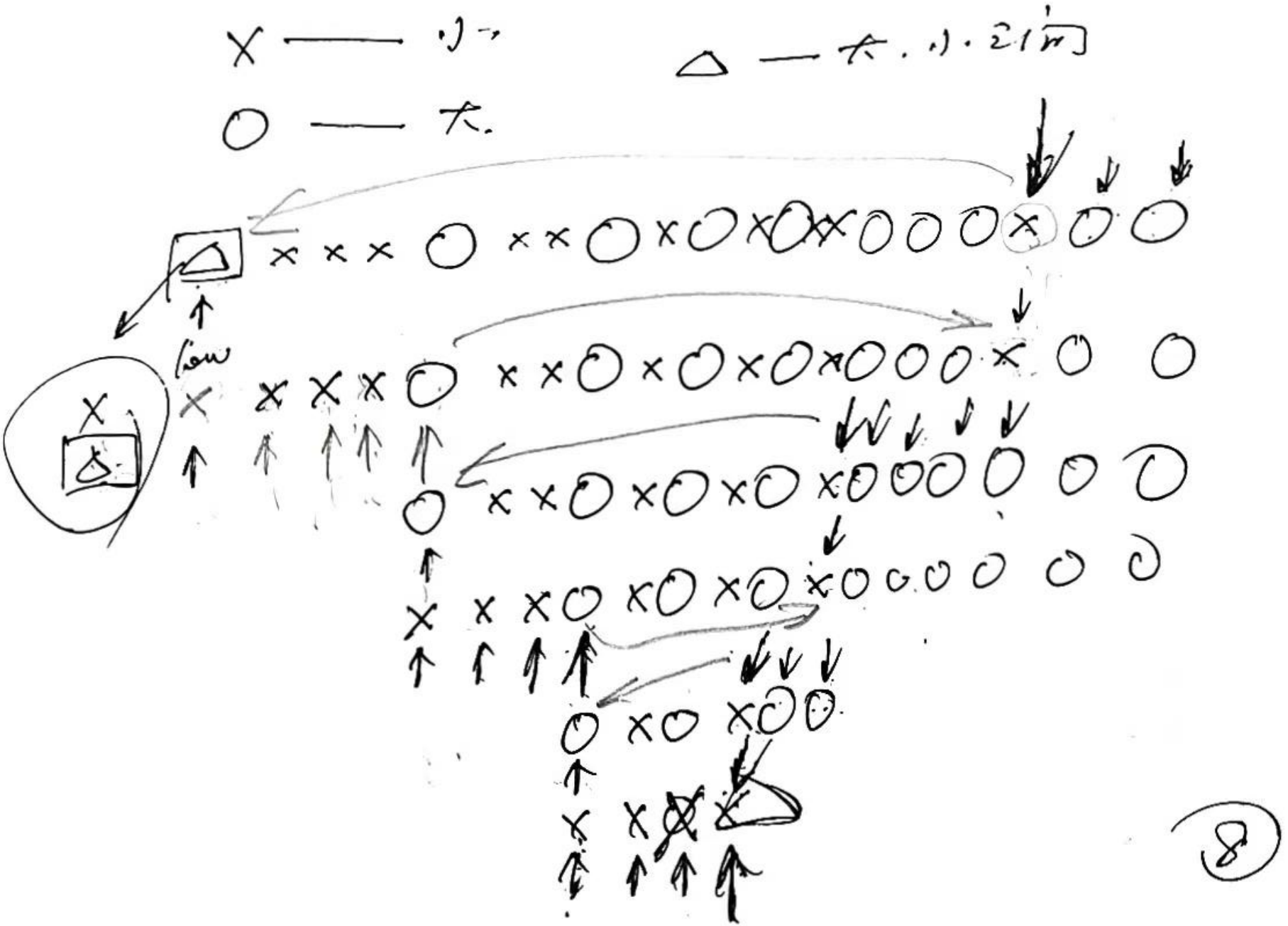
② low 从前向后找比 x 大的 (low < high)

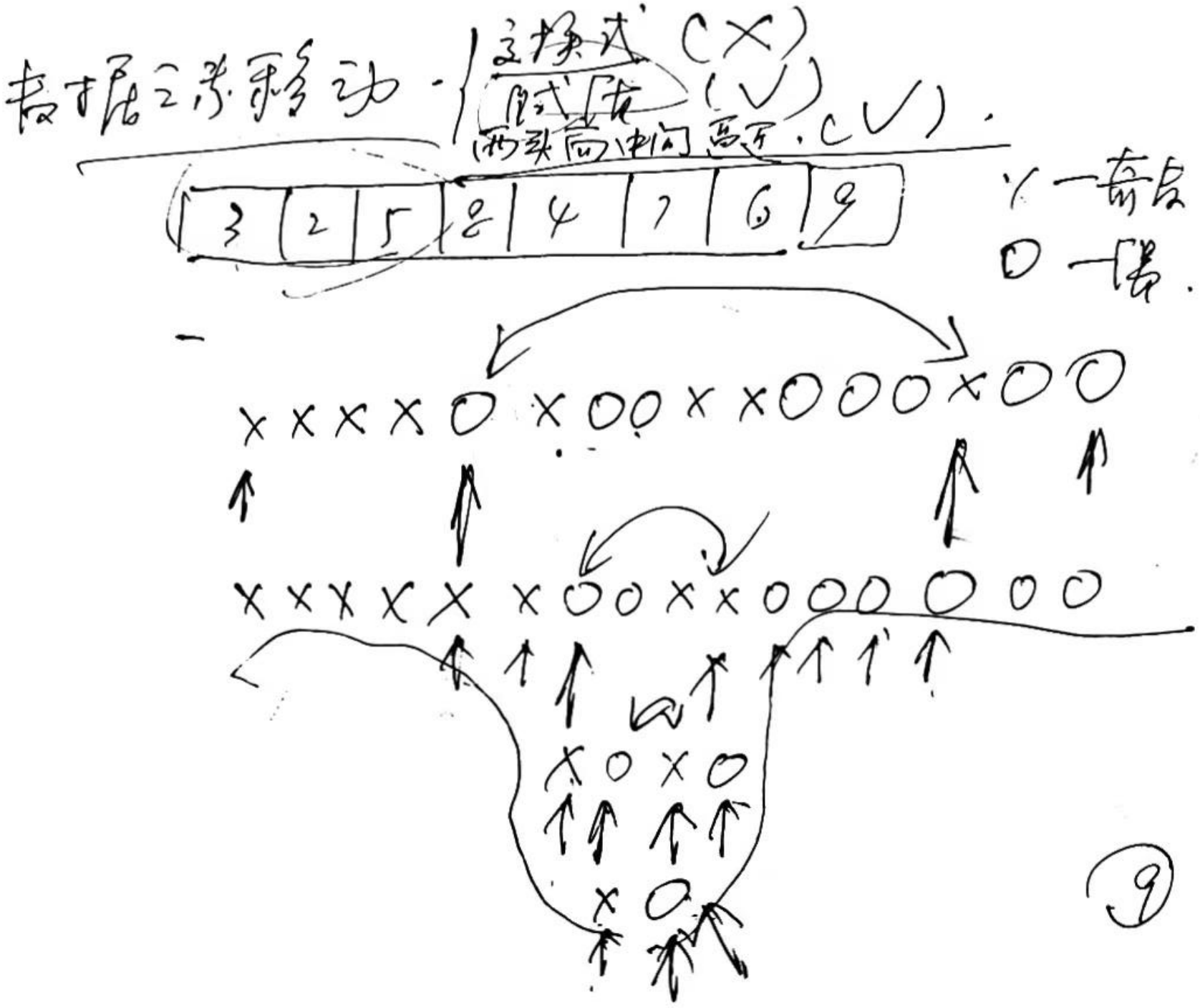
a[high] = a[low] (low < high)

}

a[low] = x;

high


X —— 小
O —— 大
△ —— 大、小、21个
low
8


数据元素移动
3 2 5 8 4 7 6 9
× 一奇点
○ 一偶.
9

一趟的算法代码

```
int Partition (int a[], int low, int high)
{
    int x;
    x = a[low];
    while (low < high) {
        while (low < high && a[high] >= x) --high;
        if (low < high) a[low] = a[high];
        while (low < high && a[low] <= x) ++low;
        if (low < high) a[high] = a[low];
    }
    a[low] = x;
    return low;
}
```

总结：① 稳定性：不

② 时间：$O(n\log_2 n)$

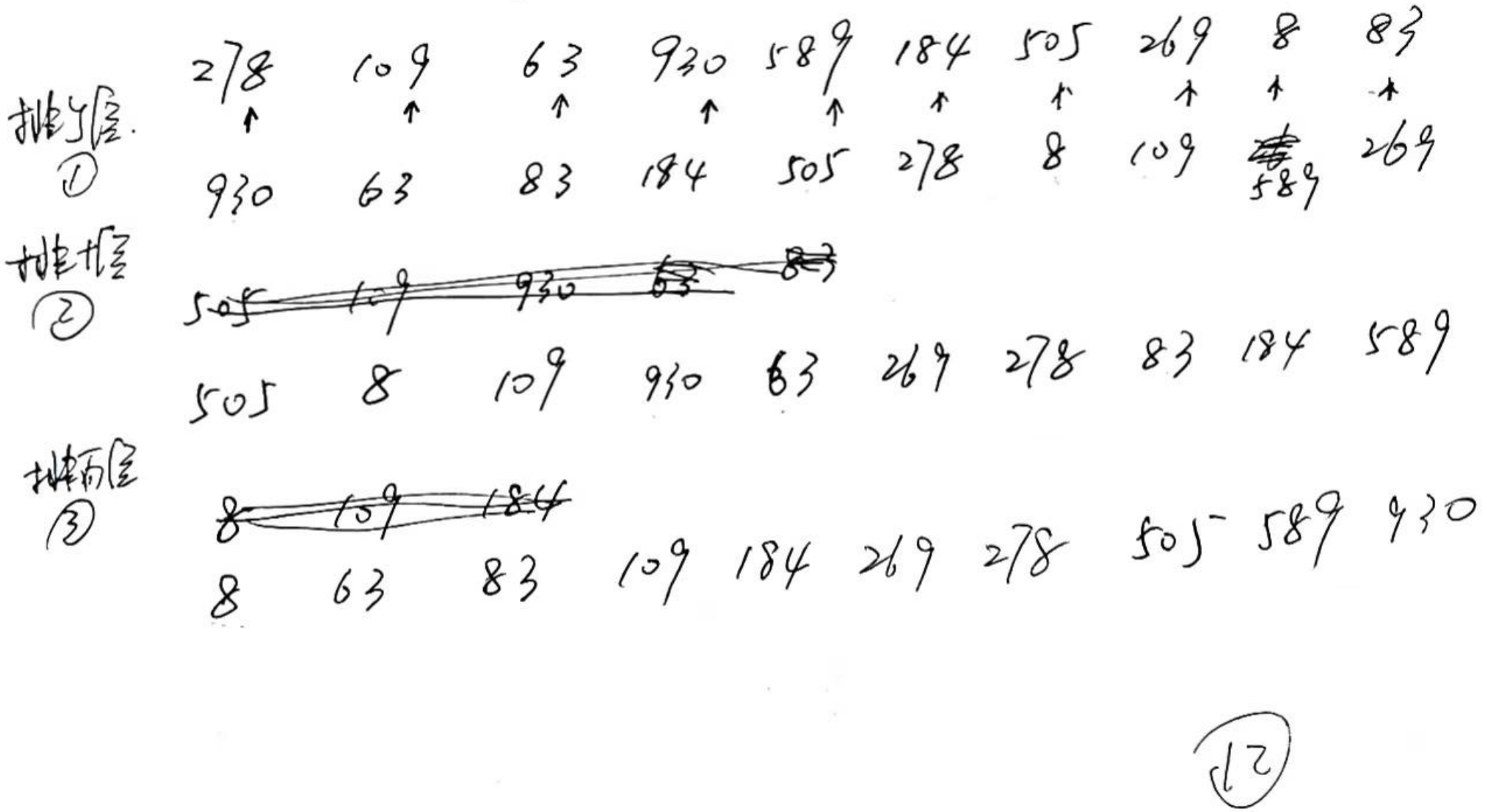
（最坏：$O(n^2)$）.

☆ 目前从时间上最好的排序算法

```
void QuickSort(int a[], int low, int high)
{ // 对顺序表a中的 [low, high] 之间元素排序
  int pivoloc;
  if(low<high){
    pivoloc=Partition(a, low, high);
    QuickSort(a, low, pivoloc-1);
    QuickSort(a, pivoloc+1, high);
  }
}
```

对n个元素的表a调用：QuickSort(a, 0, n-1);

基数排序


278 109 63 930 589 184 505 269 8 83
排序后 ①
930 63 83 184 505 278 8 109 589 269
排序后 ②
505 8 109 930 63 269 278 83 184 589
排序后 ③
8 63 83 109 184 269 278 505 589 930

时间复杂度：$O(d(n+rd))$.

空间：$O(rd)$

排序算法的稳定性和复杂性

类别	排序方法	时间复杂度		空间复杂度	稳定性
		平均时间	最坏打算	辅助存储	
插入排序	直接插入	$O(n^2)$	$O(n^2)$	$O(1)$	稳定
	Shell	$O(n^{1.3})$	$O(n^2)$	$O(1)$	不稳定
选择排序	直接选择	$O(n^2)$	$O(n^2)$	$O(1)$	不稳定
	堆排序	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(1)$	不稳定
交换排序	冒泡排序	$O(n^2)$	$O(n^2)$	$O(1)$	稳定
	快速排序	$O(n\log_2 n)$	$O(n^2)$	$O(n\log_2 n)$	不稳定
归并排序		$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n)$	稳定
基数排序		$O(d(r+n))$	$O(d(r+n))$	$O(rd+n)$	稳定

☆ 在排序过程中，每趟都可确定一个元素在其
最终位置：冒泡、简单选择、堆、快排
全局有序序列 枢轴

☆ 平均性能：快排（√）
↳最坏：不如堆、
不如归并

时间复杂度：归并时间 < 堆排序
↳空间辅助空间大