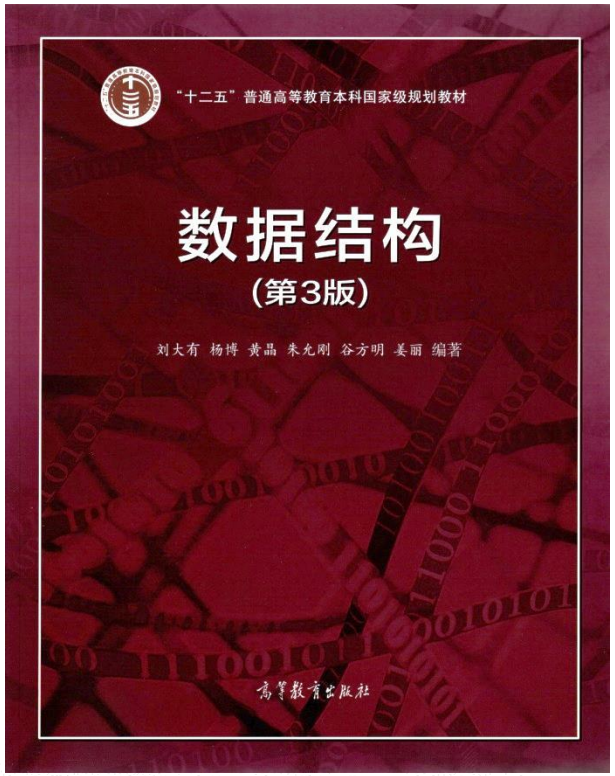
计算机学院王湘浩班

2024级



数据之法

结构之美

算法之道

平方阶排序算法

- 排序概念
- 回顾直接插入排序
- 回顾冒泡排序
- 回顾直接选择排序

zhuyungang@jlu.edu.cn

Last updated on 2025.5

钱易

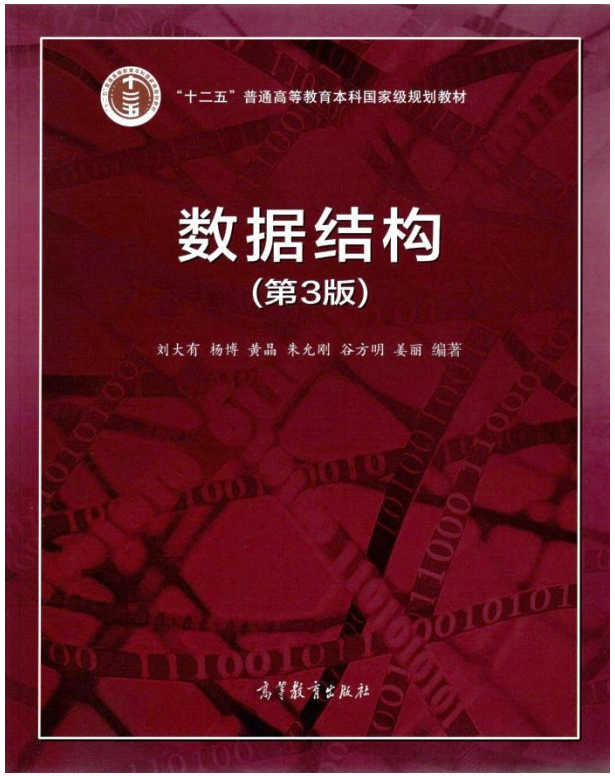
北京大学22级本科生

2020年NOI全国中学生信息学奥赛决赛金牌

2021年IOI世界中学生信息学奥赛亚军

对于一个数据结构，我会思考：它能干什么事情？它能在什么时候使用？什么时候不能使用？为什么不能使用？这些思考对我实力提升非常大。

要多做题。做的题越多，就很容易从之前某一道题的做法中来联想到新的题目怎么做。当做题正确率低时不要灰心，可以通过练习慢慢提高。



平方阶排序算法

- **排序概念**
- 回顾直接插入排序
- 回顾冒泡排序
- 回顾直接选择排序

数据之法
结构之美
算法之道

zhuyungang@jlu.edu.cn

Last updated on 2025.5


淘宝网
书包
搜 索
所有分类
找到相关宝贝 647157件
所有宝贝
人气
天猫
二手
1/100
常用：海外商品 货到付款 消费者保障 7天退换 正品保障 旺旺在线
默认排序
销量
信用
价格
总价
所在地
列表
大图
销量从高到低
信用从高到低
价格从低到高
价格从高到低
总价从低到高
总价从高到低
owbone双肩包 韩版背包男休闲旅行
电脑双肩背包运动包女新款书包
天猫 TMALL.COM
owbone唯然专卖店 和我联系
¥ 108.77
运费：0.01
信用卡
折扣
广东 广州
货到付款
月售5058笔
13735条评价
消费者保障
七天退换
正品保障
木村耀司 双肩包学生包书包旅行包双
肩背包电脑包男女韩版潮包
天猫 TMALL.COM
木村耀司旗舰店 和我联系
¥ 54.00
运费：0.00
信用卡
折扣
上海
代购
月售3704笔
8481条评价
消费者保障
七天退换
正品保障
韩版包袋日韩新款糖果色背包笔记本包
海绵隔层双肩糖果包书包男女
特价折扣包 和我联系
¥ 56.00
运费：12.00
信用卡
折扣
上海
月售140笔
368条评价
消费者保障
七天退换
24小时发货
2012迷彩耐克双肩包 nike背包 男女学
生书包 情侣包 日韩流行背包
¥ 75.60
运费：0.01
北京
月售81笔
21条评价
消费者保障

排序问题的基本概念

A

- 文件：给定待排序的 n 个数据对象，R_1，R_2，…，R_n，这些数据对象为记录，并称这 n 个记录的集合为一个文件；
- 关键词：通常数据对象包括多个属性域，可将其中的一个属性域作为排序的依据，称其为关键词域。上述 n个记录的关键词分别是 K_1，K_2，…，K_n；
- 排序：按规定的顺序，以关键词为依据，对一个文件中的诸记录进行排列的过程。

```
struct student {
    char Name[10];
    int Gender;
    int ID;
}
```

排序算法的度量指标

A

- **时间复杂度：** 衡量排序算法好坏的最重要标准。关键运算为算法执行中**关键词的比较次数**与**数据的移动次数**来衡量。
- **空间复杂度：** 主要考察排序过程占用存储空间的大小。
- **排序算法的稳定性：** 如果两个对象R_i和R_j，其关键词相同，且在排序前R_i在R_j的前面，若排序后R_i仍在R_j的前面，则称该排序算法是稳定的，否则称这个排序算法是不稳定的。

班级	学号	姓名
7班	21230705	张无忌
7班	17230102	周芷若
5班	23230106	赵敏
6班	17230101	杨逍
5班	55230107	范瑶
5班	21230503	张三丰

按学号排序

班级	学号	姓名
6班	17230101	杨逍
7班	17230102	周芷若
5班	21230503	张三丰
7班	21230705	张无忌
5班	23230106	赵敏
5班	55230107	范瑶

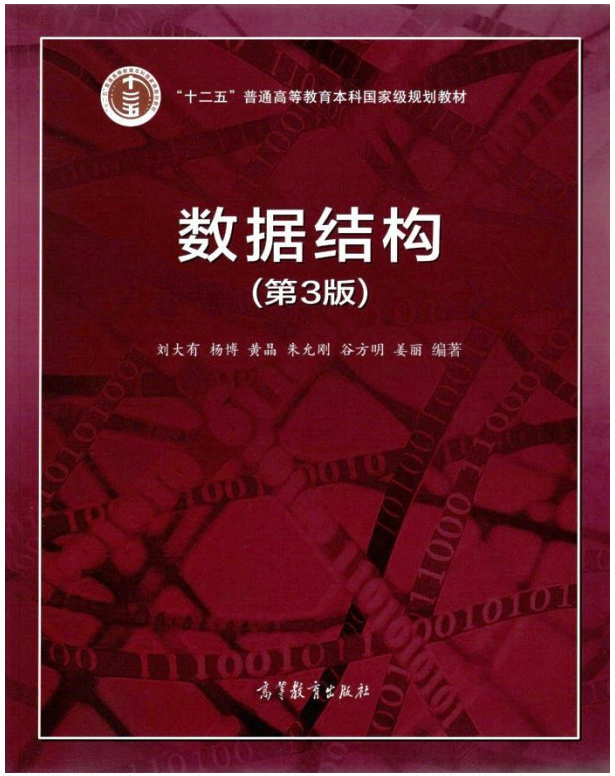
按班级排序

班级	学号	姓名
5班	21230503	张三丰
5班	23230106	赵敏
5班	55230107	范瑶
6班	17230101	杨逍
7班	17230102	周芷若
7班	21230705	张无忌

排序算法的分类

A

- 从存储设备角度：
  - ✓ 内排序：在排序过程中所有数据元素都在内存中；
  - ✓ 外排序：当待排序元素所占空间大到内存存不下时，排序必需借助外存来完成。
- 按对关键词的操作：
  - ✓ 基于关键词比较的排序：基于关键词比较；
  - ✓ 分布排序：基于元素的分布规律。
- 按时间复杂度：
  - ✓ 平方阶算法：算法简单易于实现，平均时间复杂度 $O(n^2)$；
  - ✓ 线性对数阶算法：相对复杂，平均时间复杂度 $O(n\log n)$；
  - ✓ 线性算法：不依赖关键词比较，需要已知元素的分布规律。



平方阶排序算法

- 排序概念
- **回顾直接插入排序**
- 回顾冒泡排序
- 回顾直接选择排序

数据之法
结构之美
算法之道

zhuyungang@jlu.edu.cn

Last updated on 2025.5

直接插入排序

A

不断将一个元素插入到其左侧已排好序的有序表中，从而得到一个新的有序表。

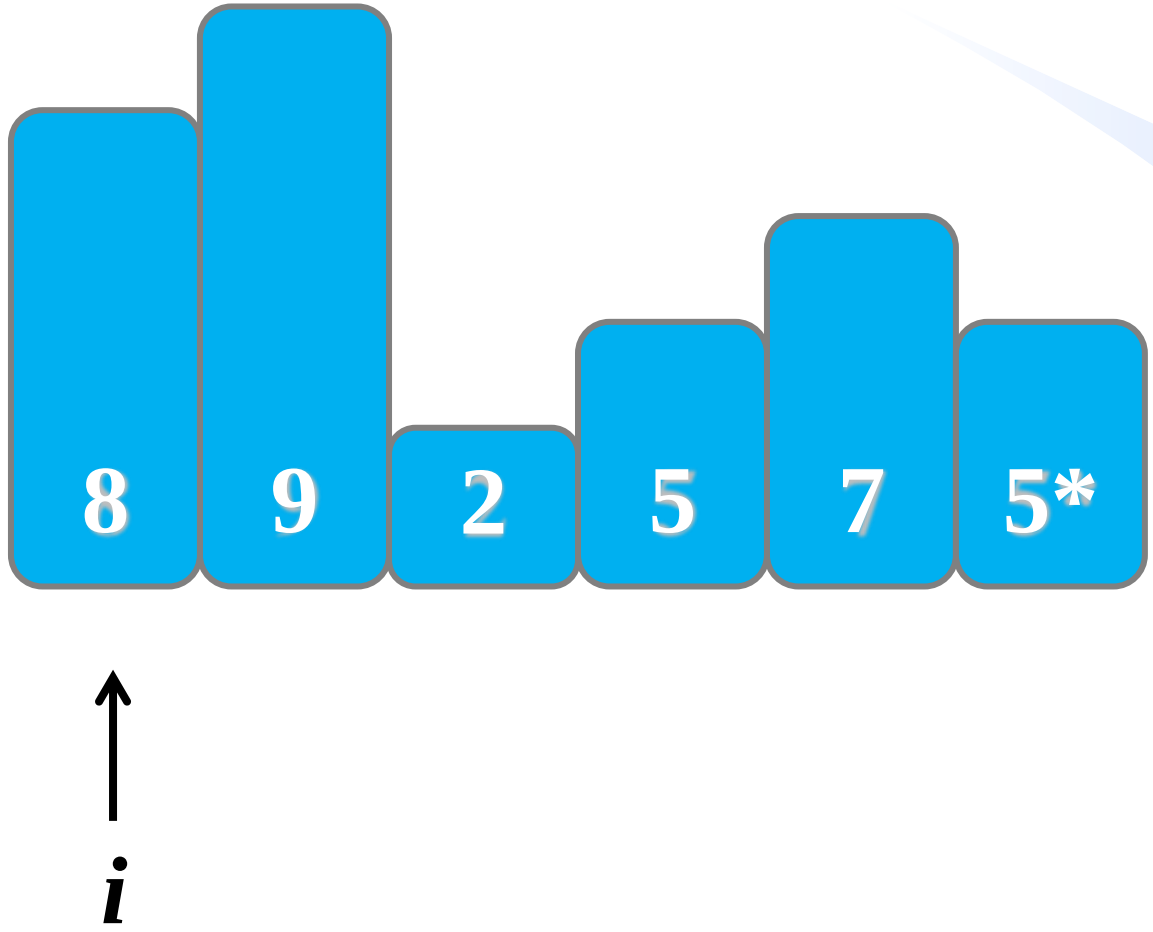
例：

✓原有序表：(9　15　23　28　37) 20

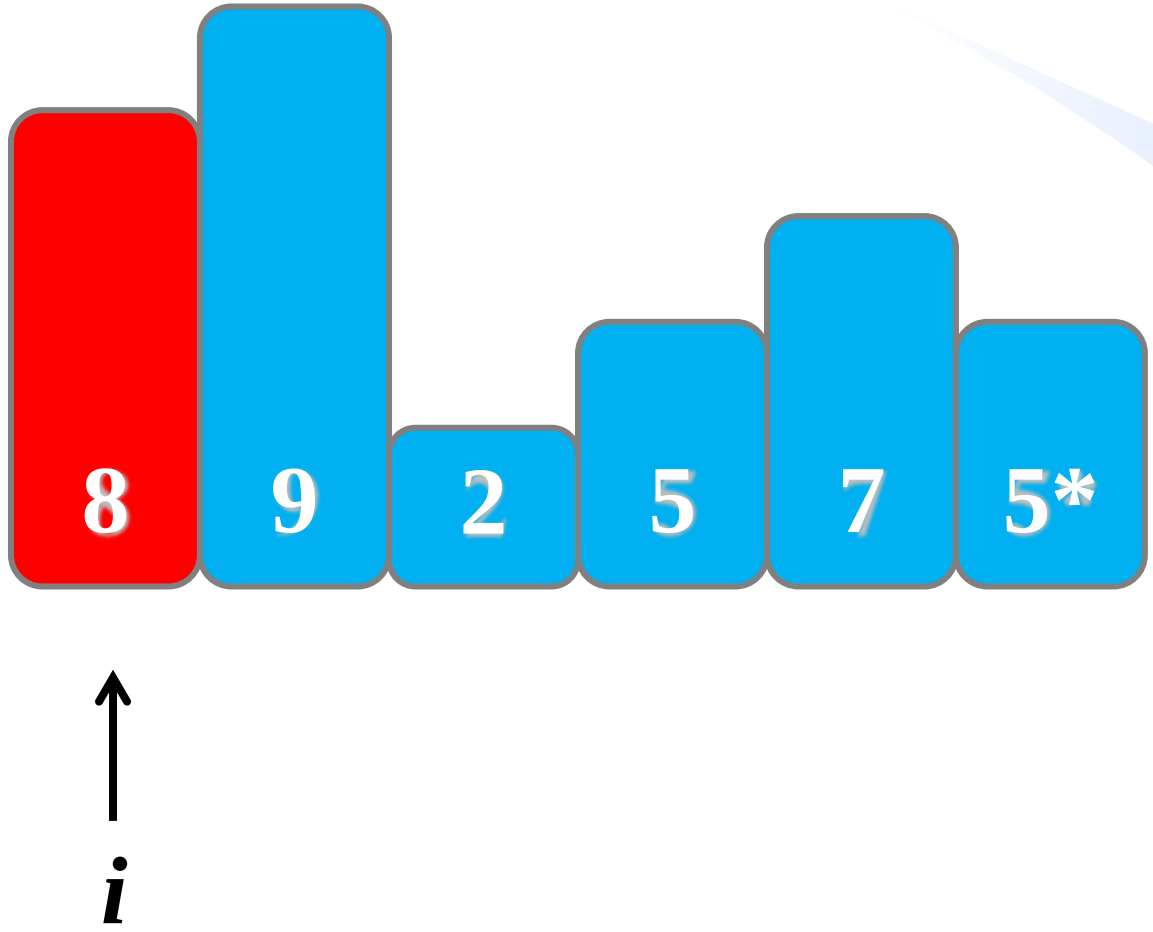
✓找插入位置：(9　15 ↑ 23　28　37) 20

✓新有序表：(9　15　20　23　28　37)

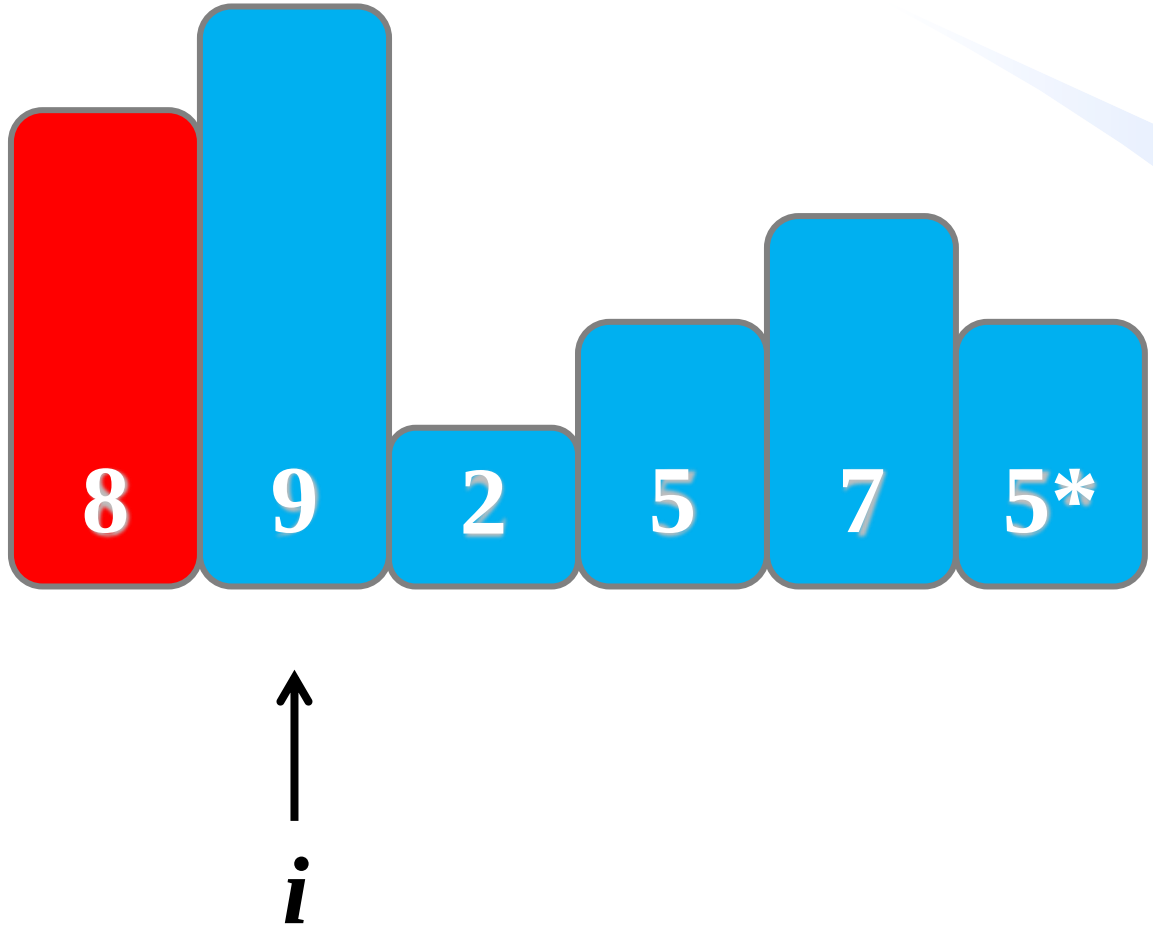
直接插入排序——示例



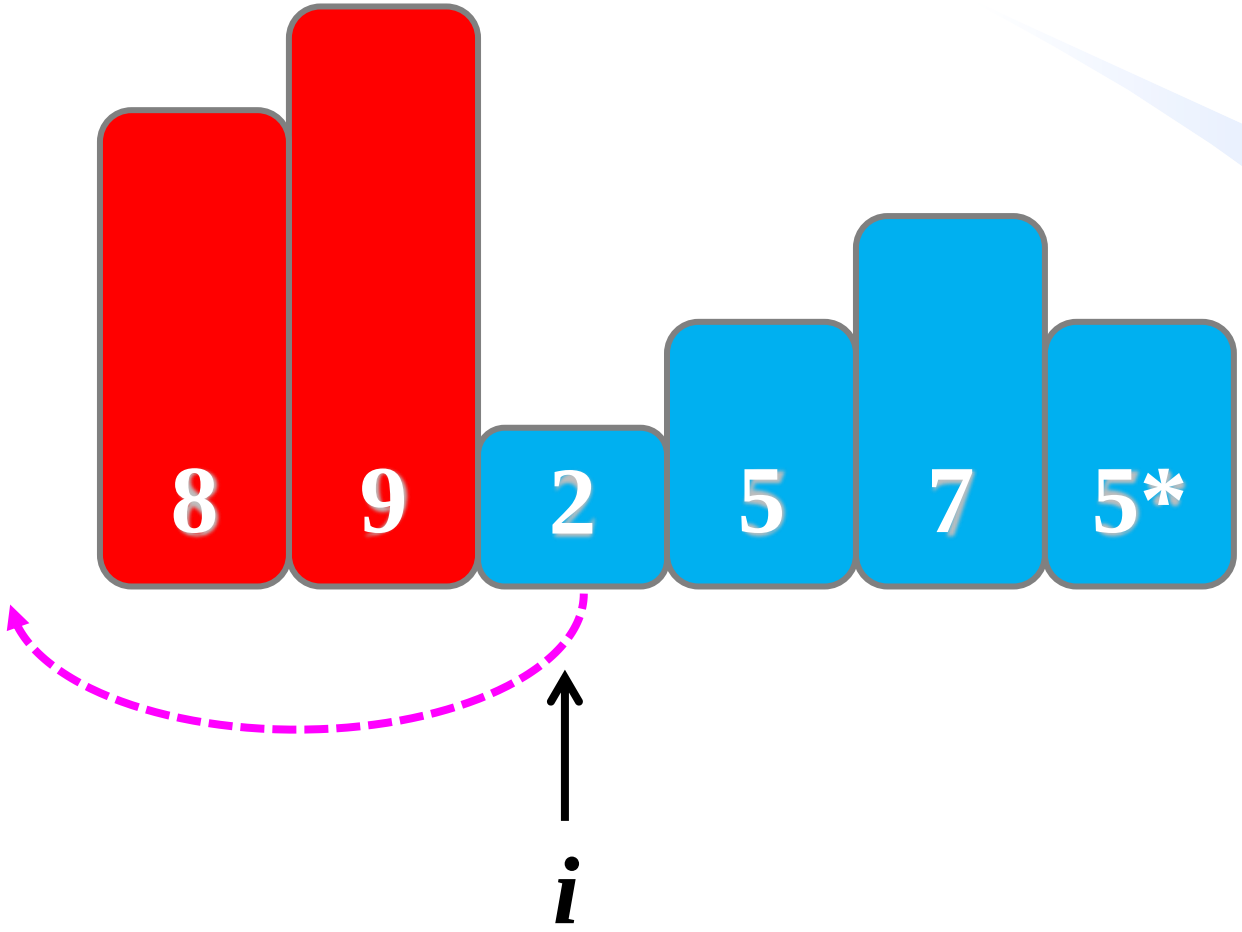
直接插入排序——示例



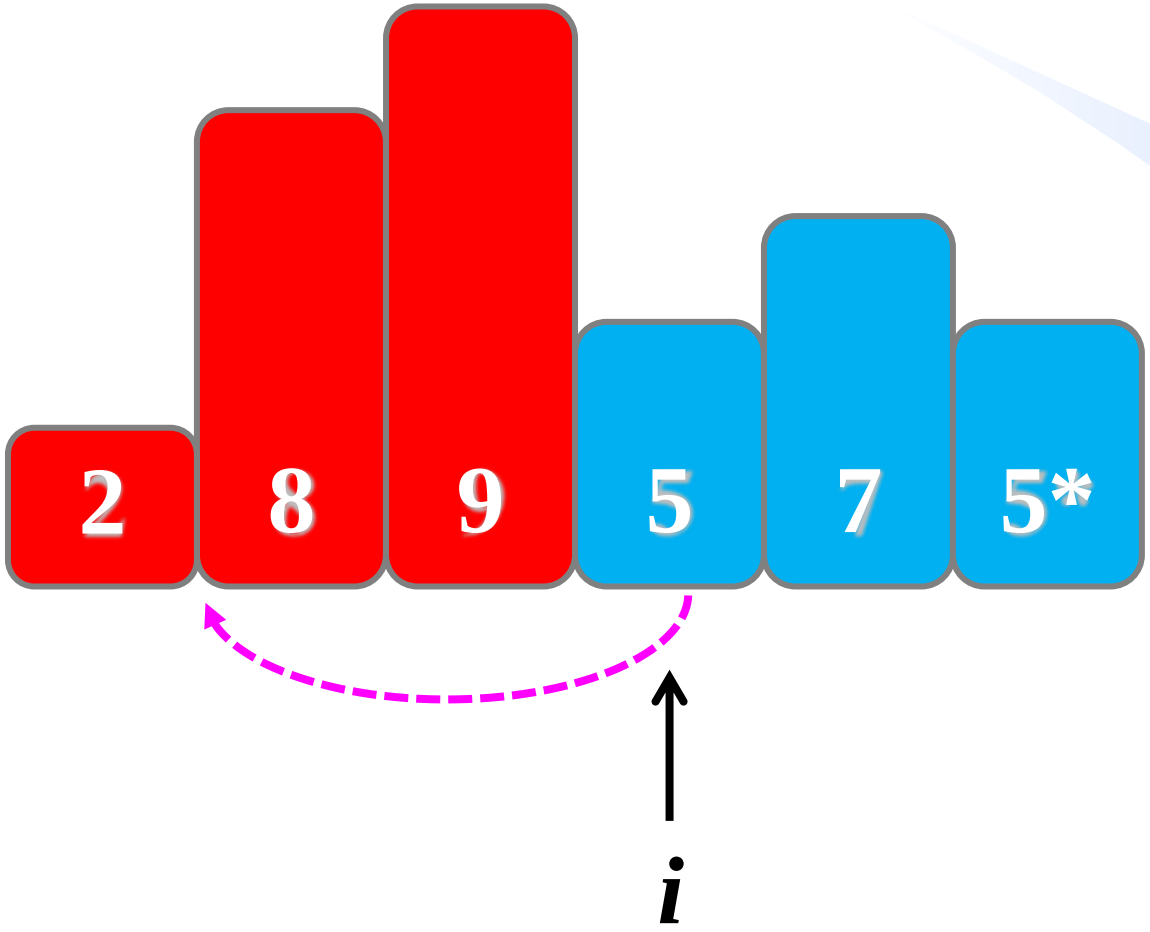
直接插入排序——示例



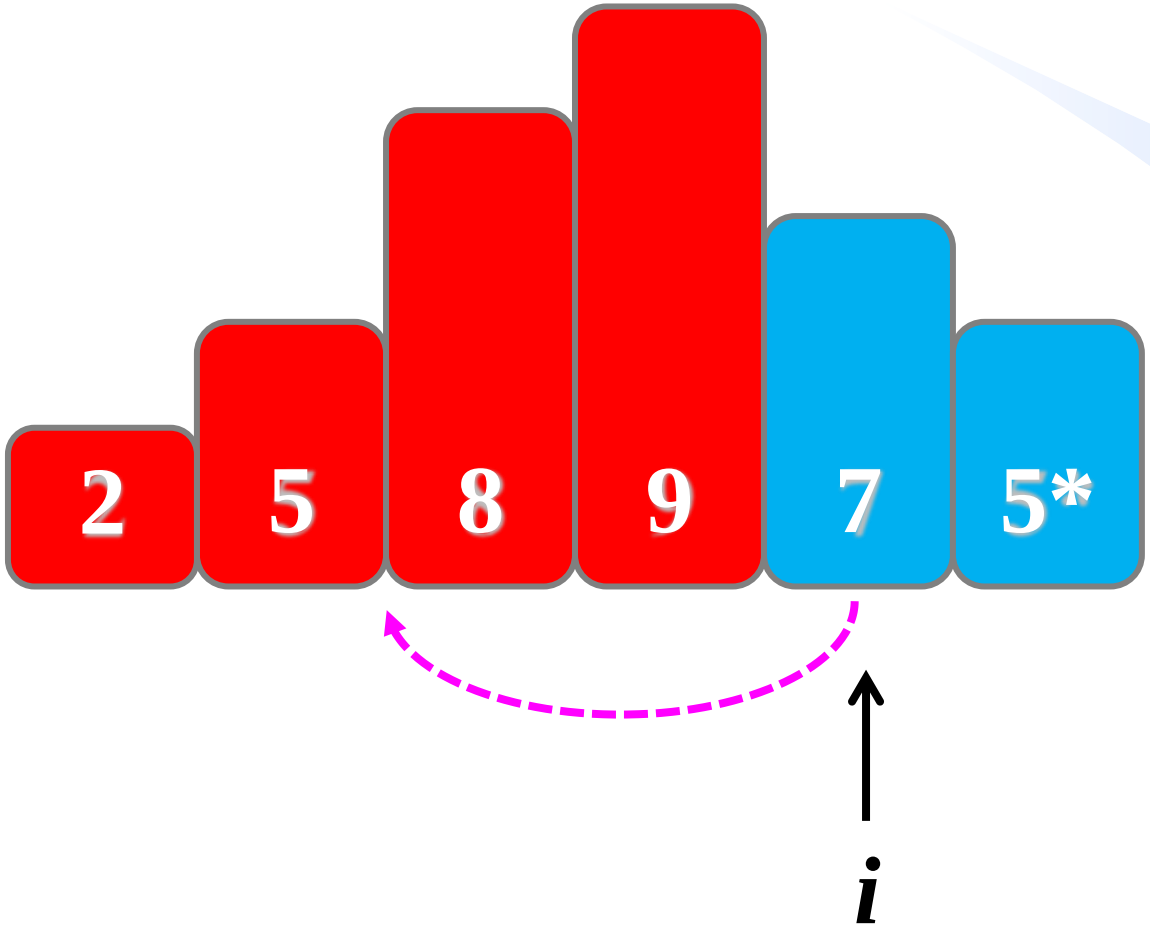
直接插入排序——示例



直接插入排序——示例

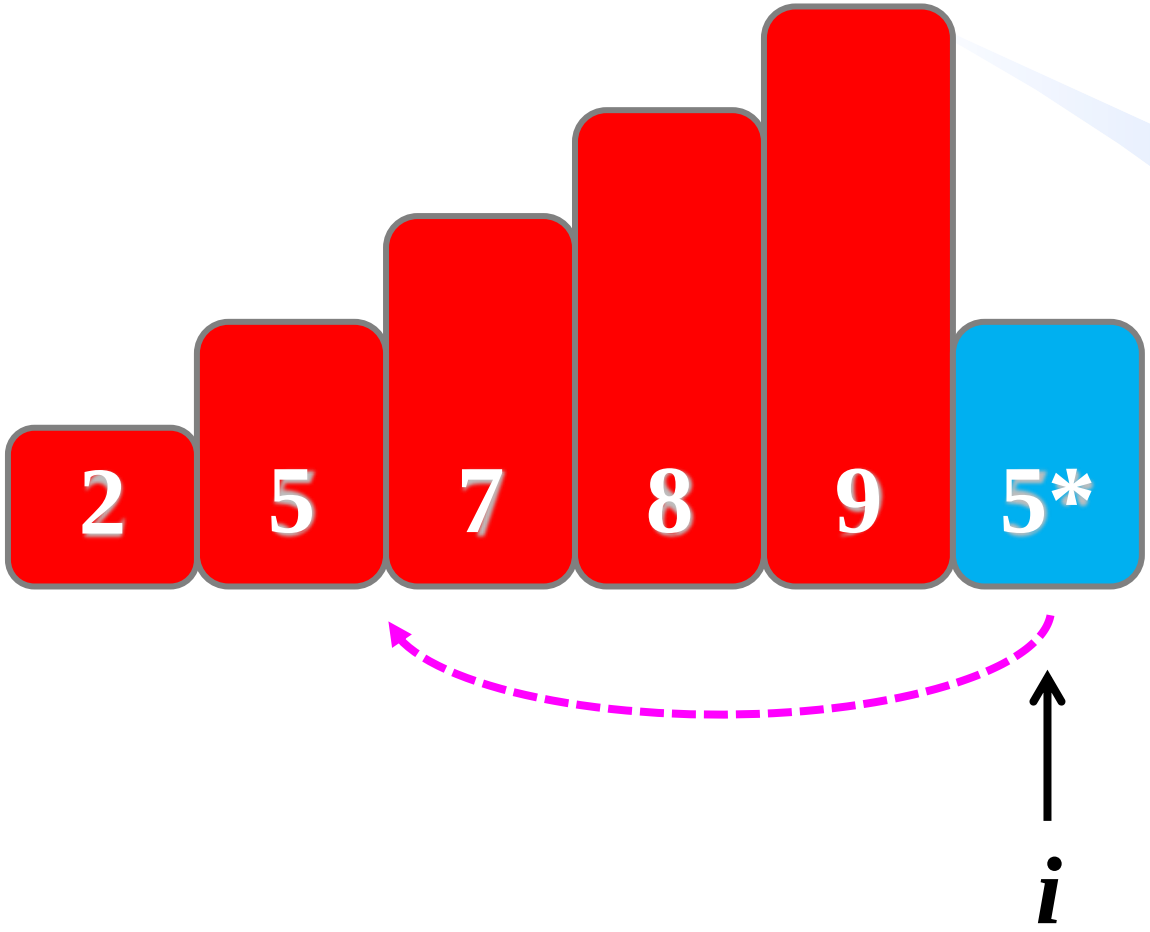


直接插入排序——示例

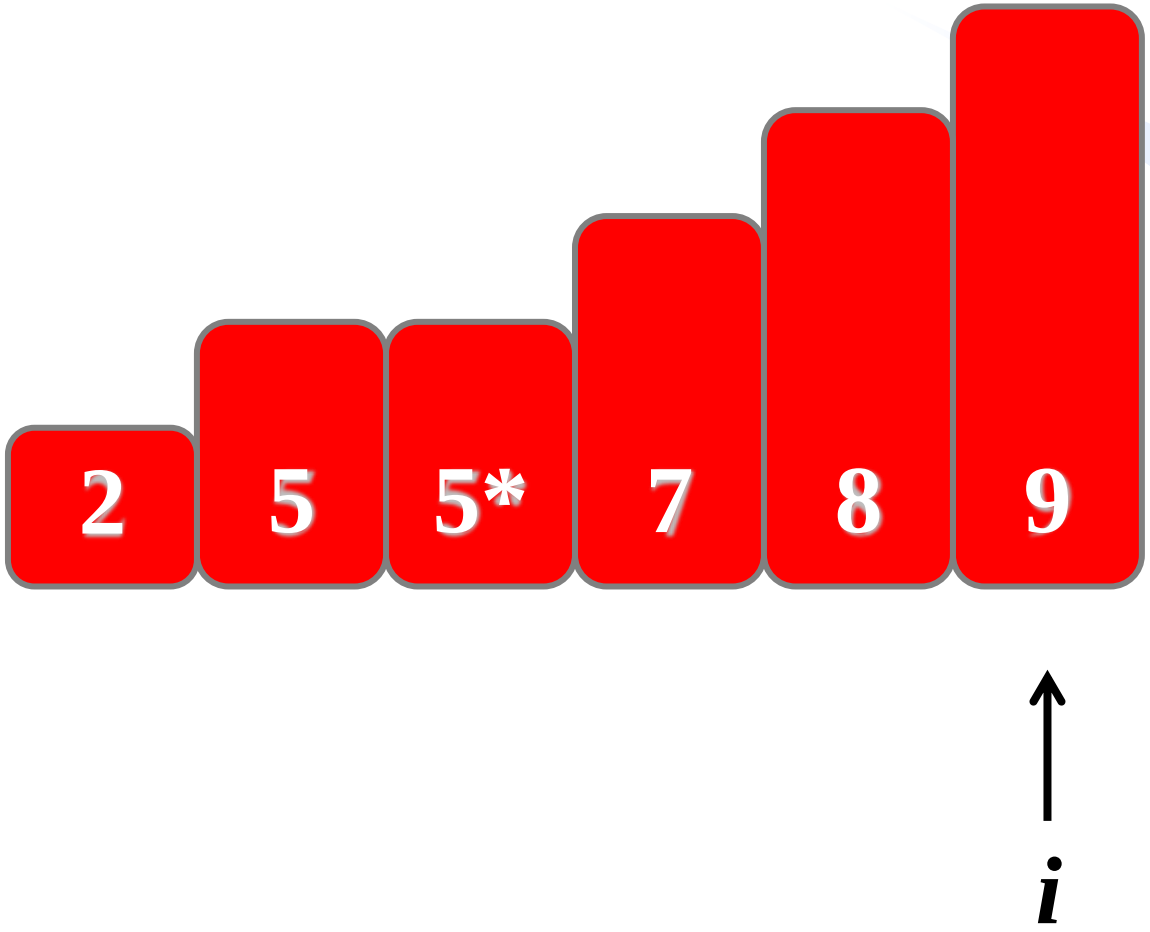


扫描 $R[i-1]...R[1]$,从右往左找第1个$\leq R[i]$的元素，在其后插入$R[i]$

直接插入排序——示例



直接插入排序——示例

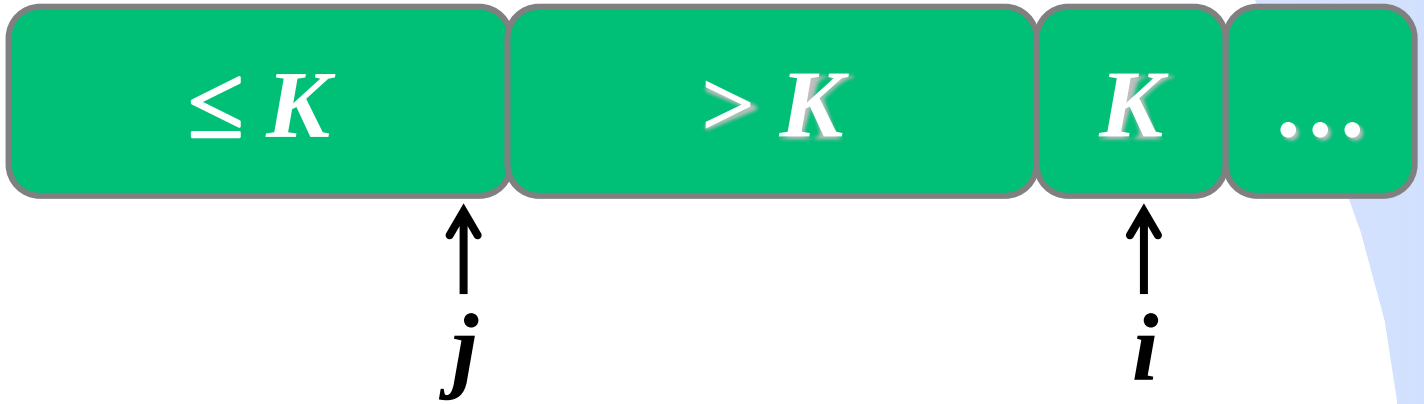


直接插入排序

A

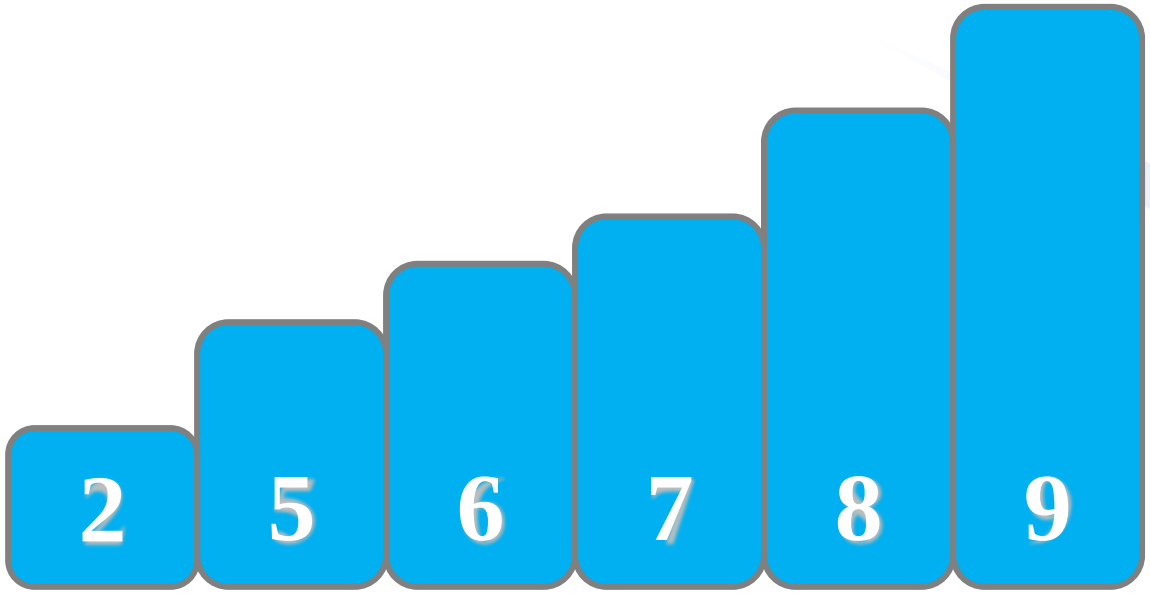
```
void InsertionSort(int R[],int n){ //对R[1]...R[n]排序
    for(int i=2; i<=n; i++){    //R[1]..... R[i-1] ← R[i]
        int K=R[i], j=i-1;
        while(j>=1 && R[j]>K){
            R[j+1]=R[j];
            j--;
        }
        R[j+1]=K;
    }
}
```

通过指针j扫描$R[i-1]$.....$R[1]$,从右往左找第1个$\leq R[i]$的元素



直接插入排序——最好时间复杂度

A

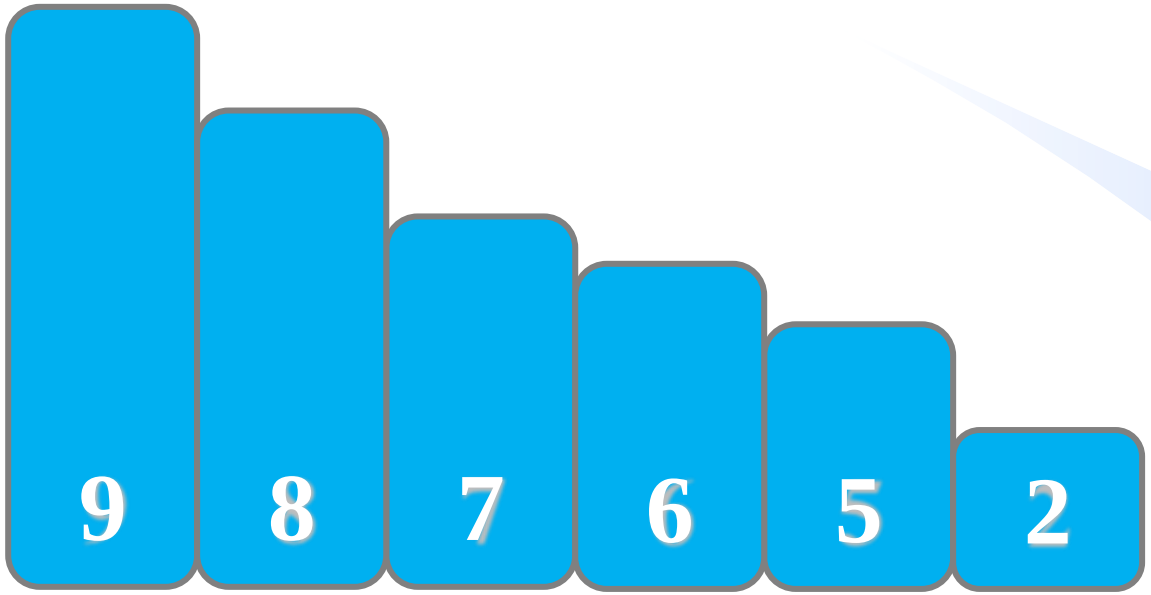


每次插入操作R_i只需与前面有序子数组的最后一个元素比较1次

最好情况：已经排好序
时间复杂度：O(n)

直接插入排序——最坏时间复杂度

A



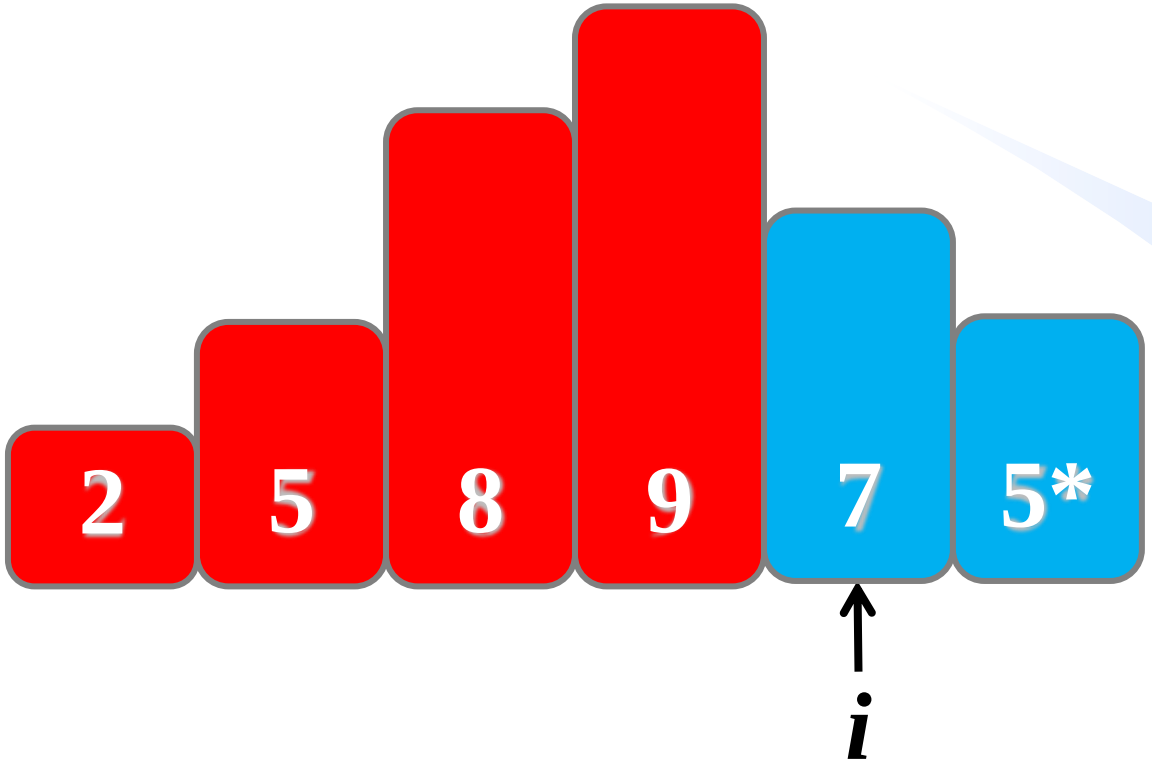
每次插入操作花费O(n)时间，R_i需要与其前面所有元素比较，前面所有元素都需移位

最坏情况：初始为逆序

时间复杂度：O(n^2)

直接插入排序——平均时间复杂度

A



若假定元素随机分布，R_i左面平均大约有一半的元素比R_i大，故每次插入操作花费O(n)时间

平均情况：假定元素随机分布

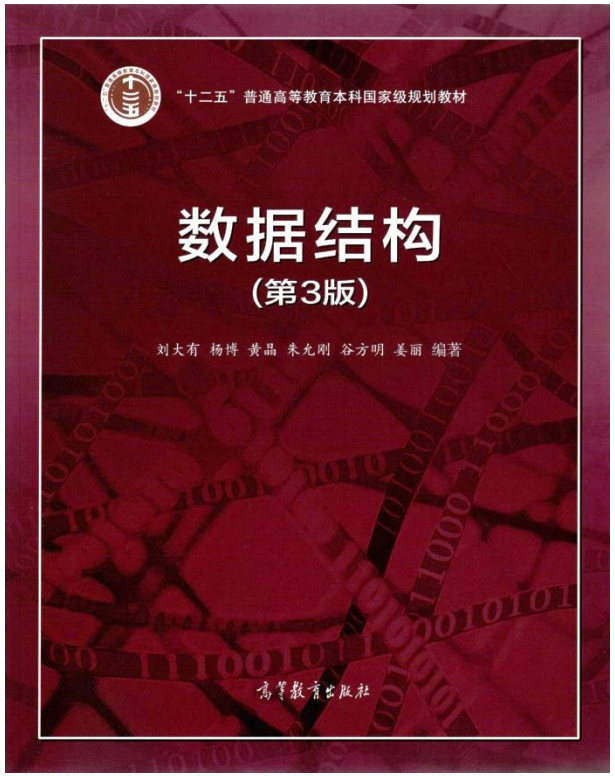
时间复杂度：O(n^2)

直接插入排序总结

A

排序算法	时间复杂度			空间复杂度	稳定性
	最好	平均	最坏		
直接插入排序	$O(n)$	$O(n^2)$	$O(n^2)$	$O(1)$	稳定

- 在“短文件”、“基本有序”时速度快。
- 适合增量数据环境。



平方阶排序算法

- 排序概念
- 回顾直接插入排序
- **回顾冒泡排序**
- 回顾直接选择排序

数据之法
结构之美
算法之道

zhuyungang@jlu.edu.cn

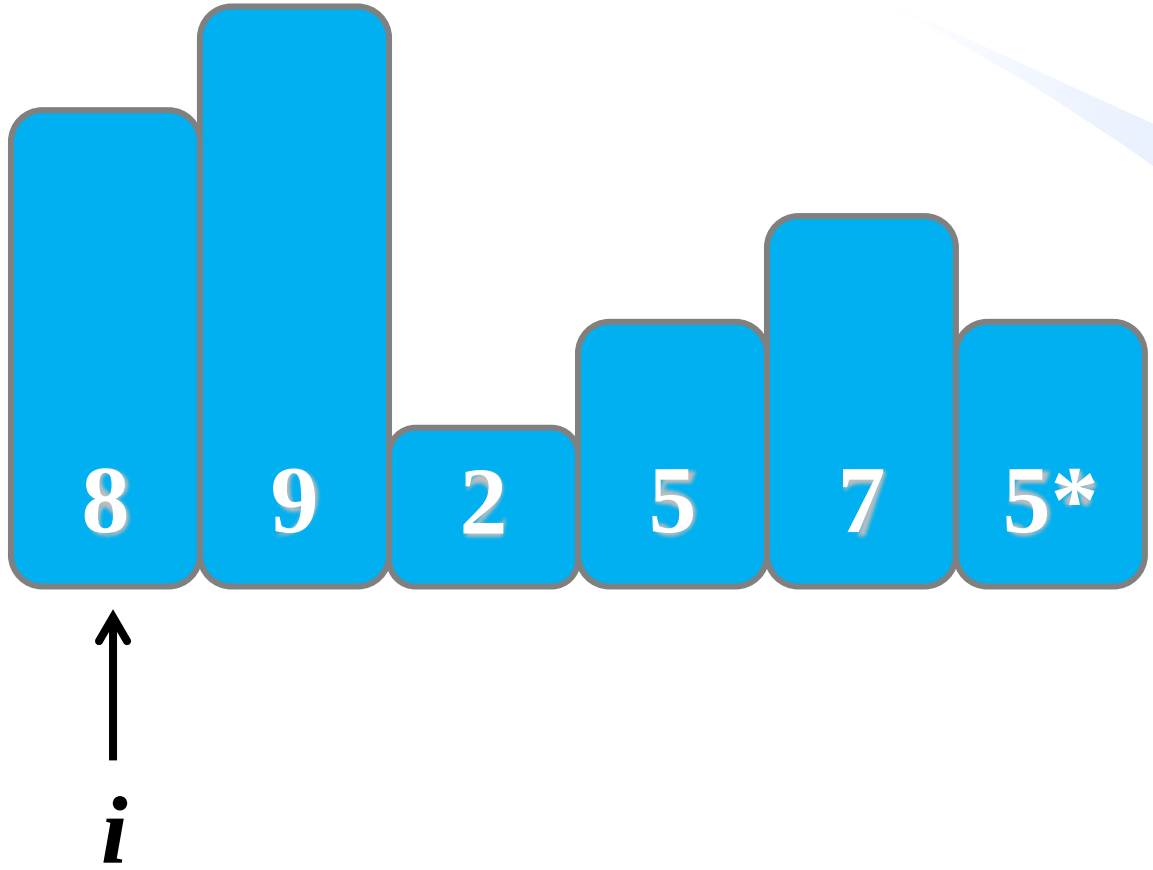
Last updated on 2025.5

冒泡排序

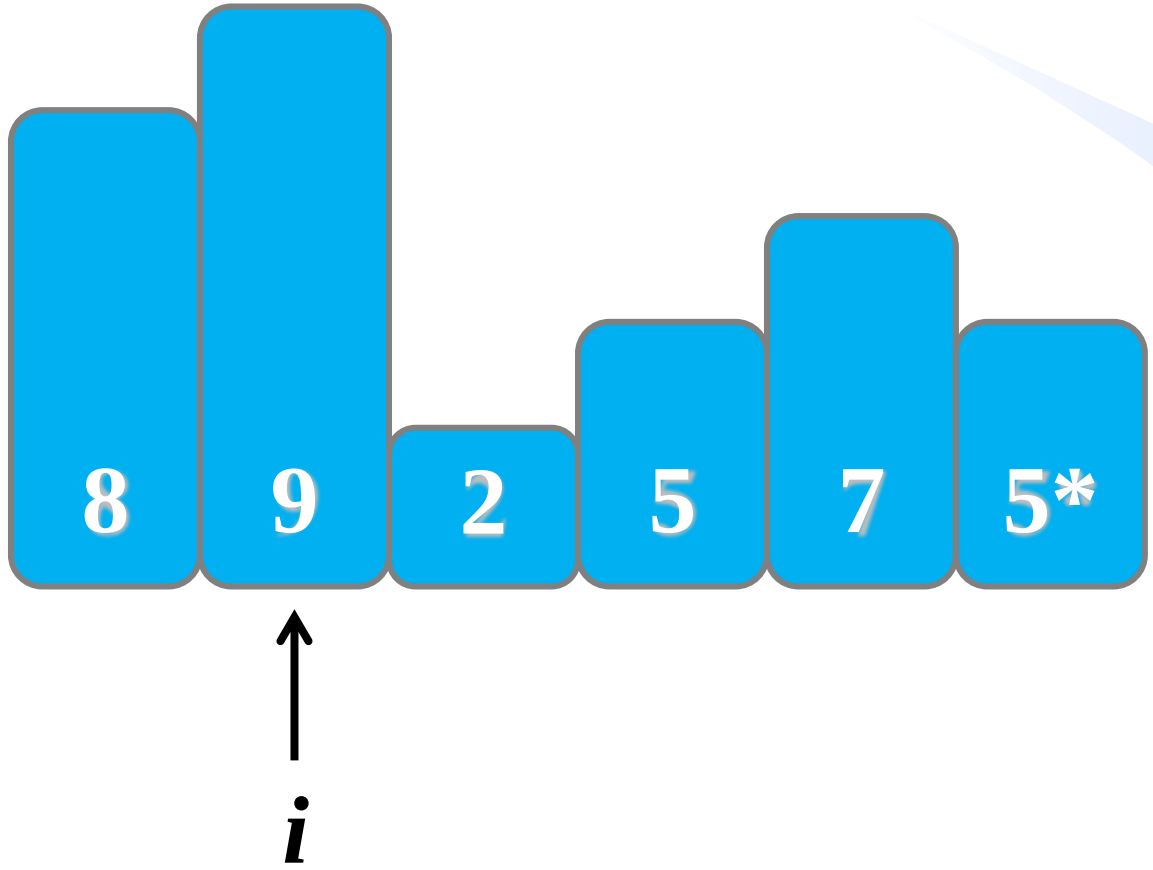
A

- 从左往右扫描数组，如果遇到反序对（两个相邻的元素，且前面的元素比后面大），则交换这两个元素。
- 扫描一趟之后，则最大元素被交换到最右边。如果把数组立起来，最大元素就像气泡一样，浮到上面。此过程称为一趟冒泡。

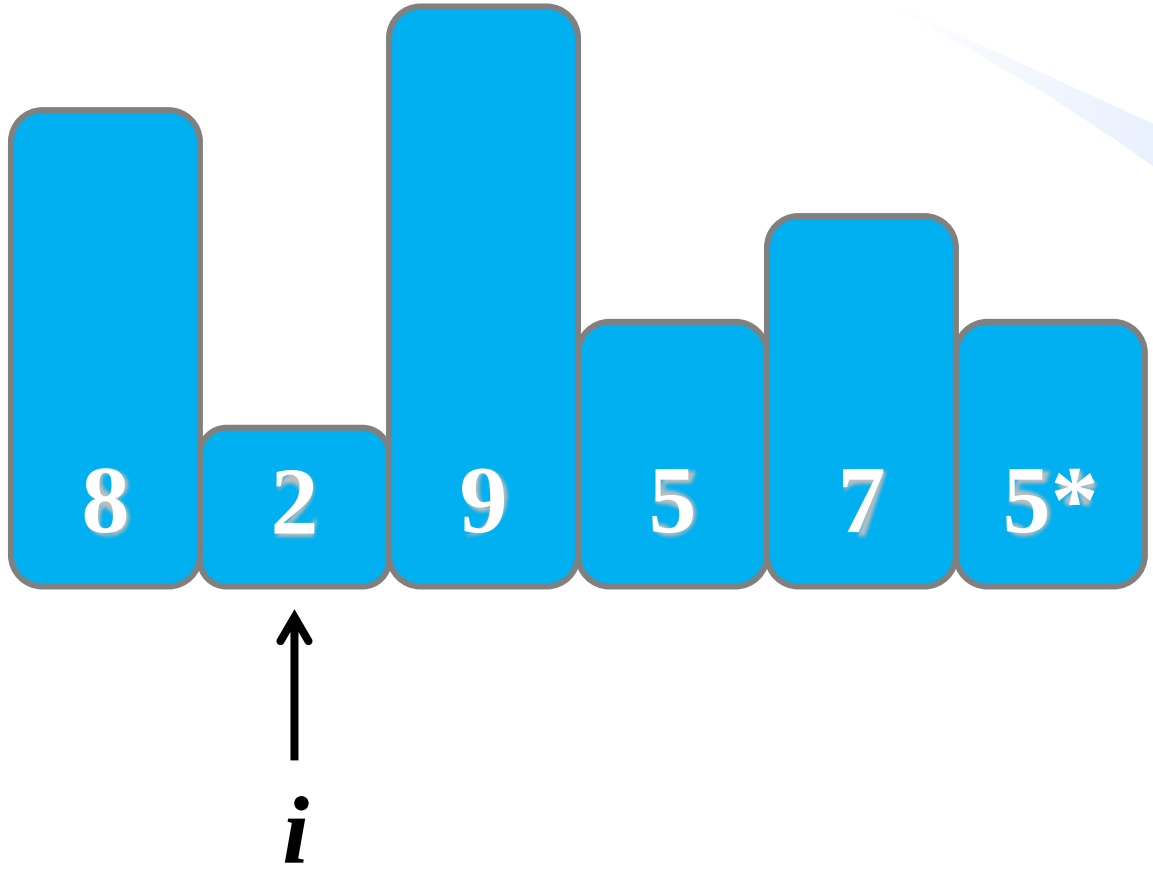
冒泡排序——示例



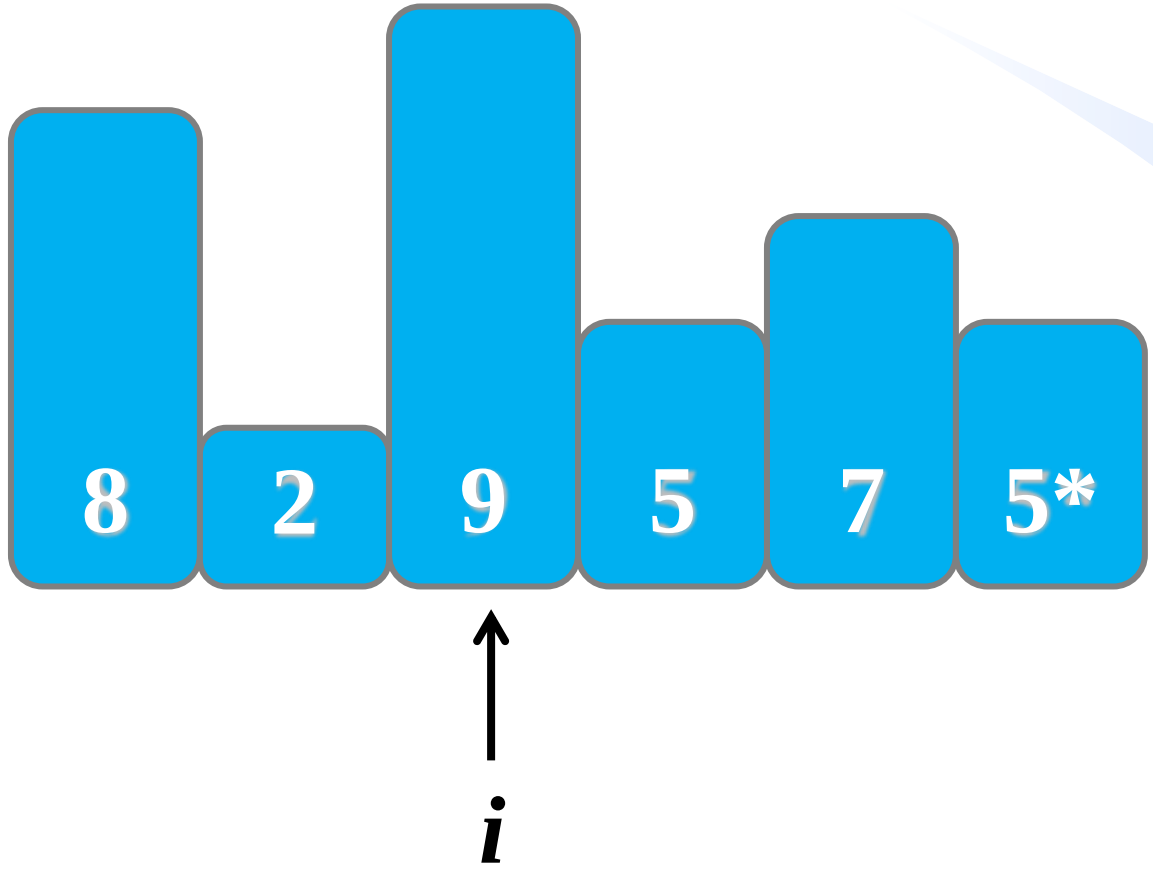
冒泡排序——示例



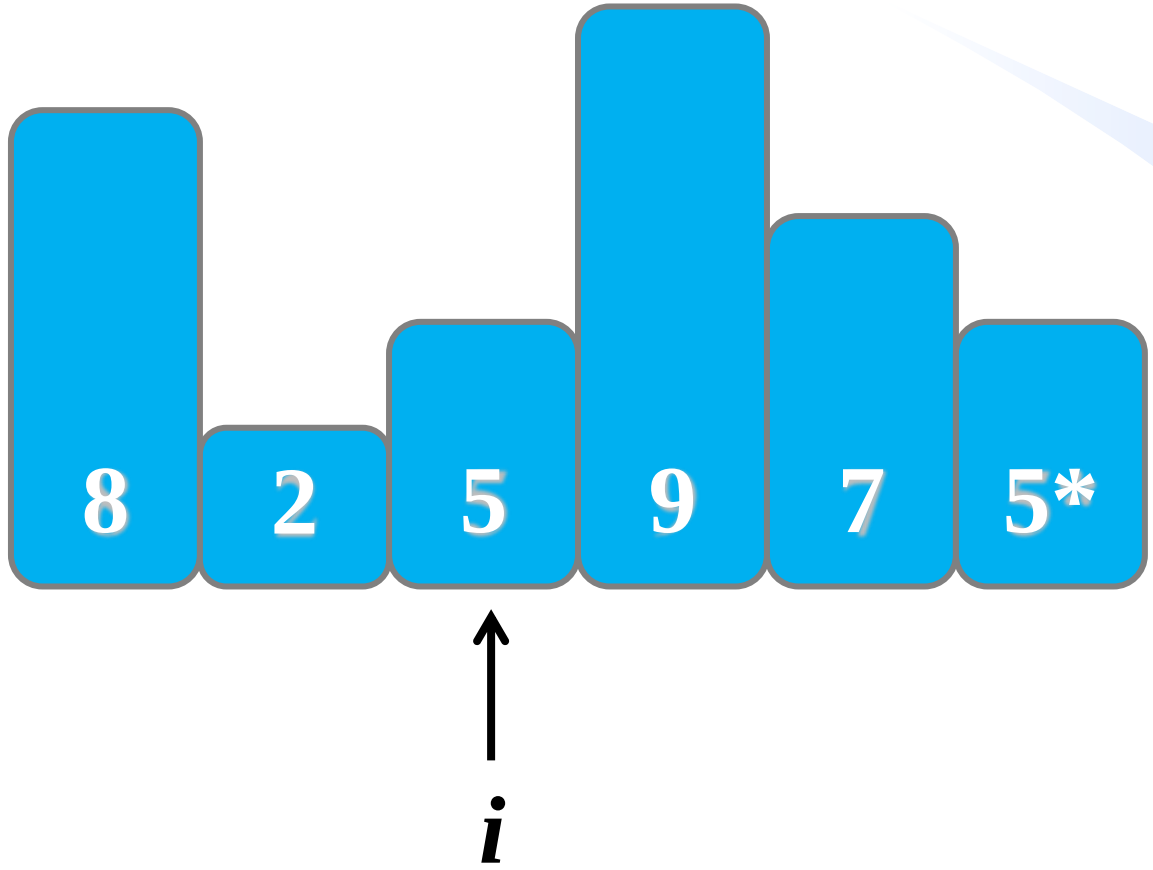
冒泡排序——示例



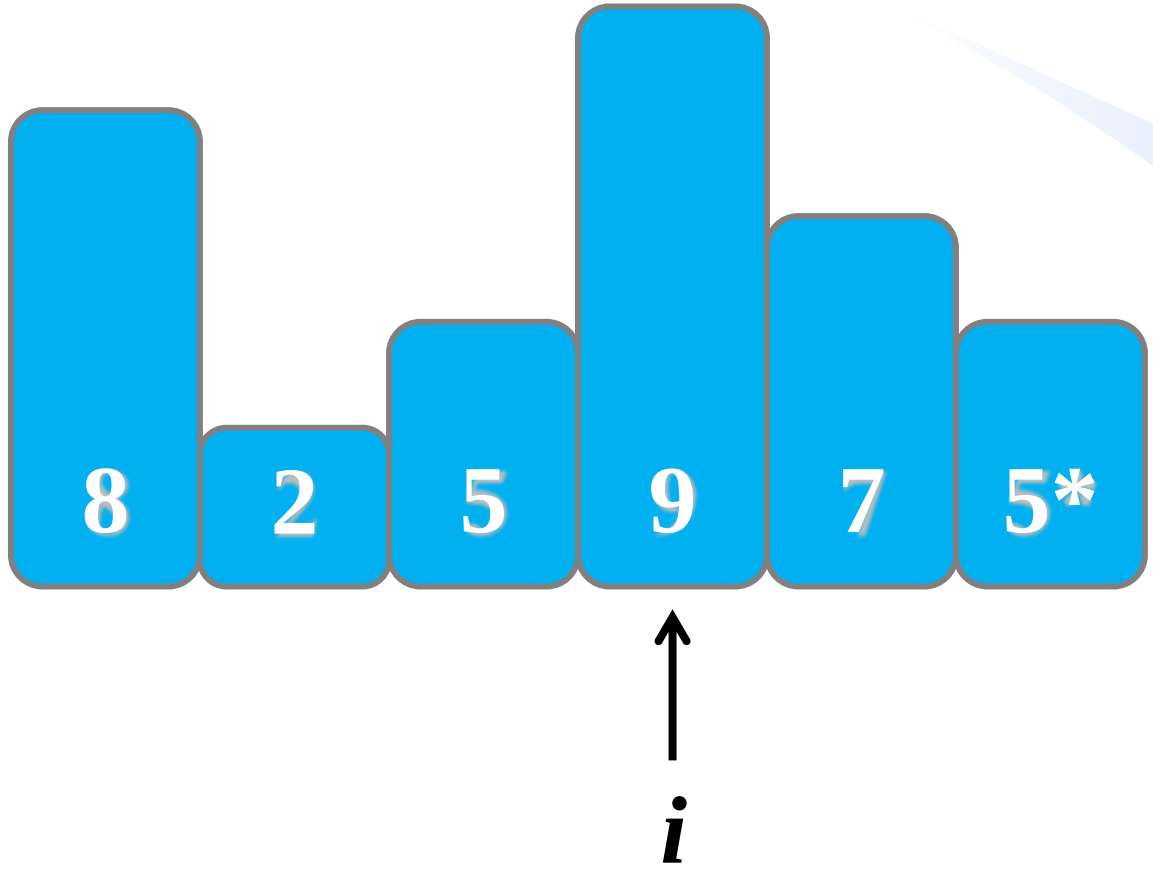
冒泡排序——示例



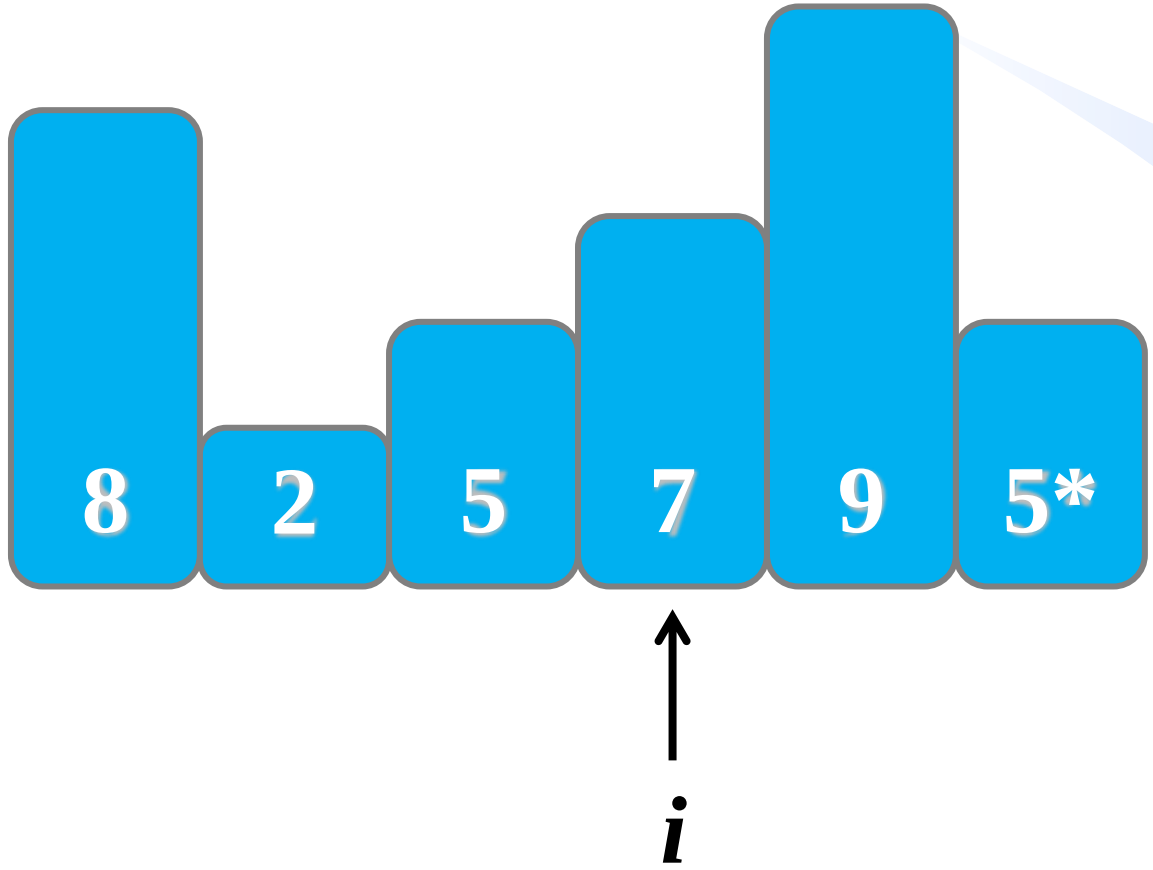
冒泡排序——示例



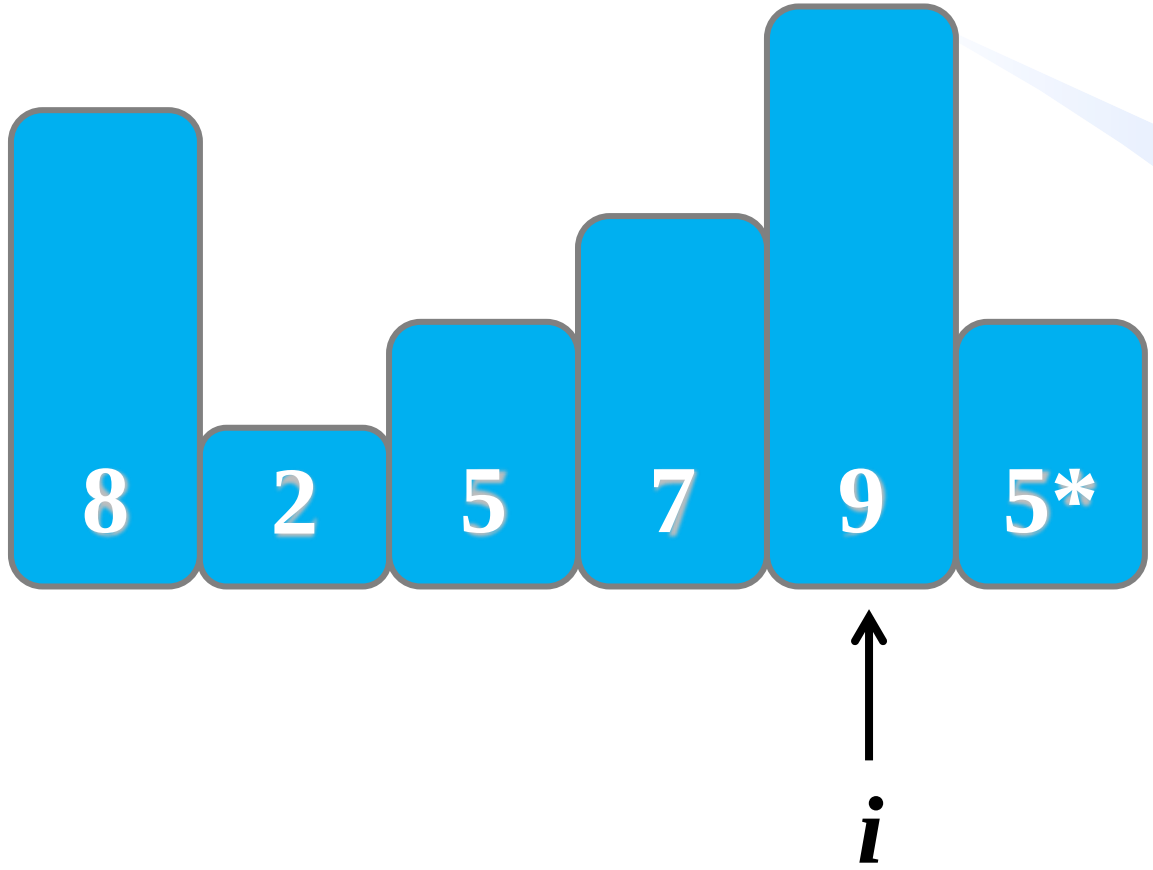
冒泡排序——示例



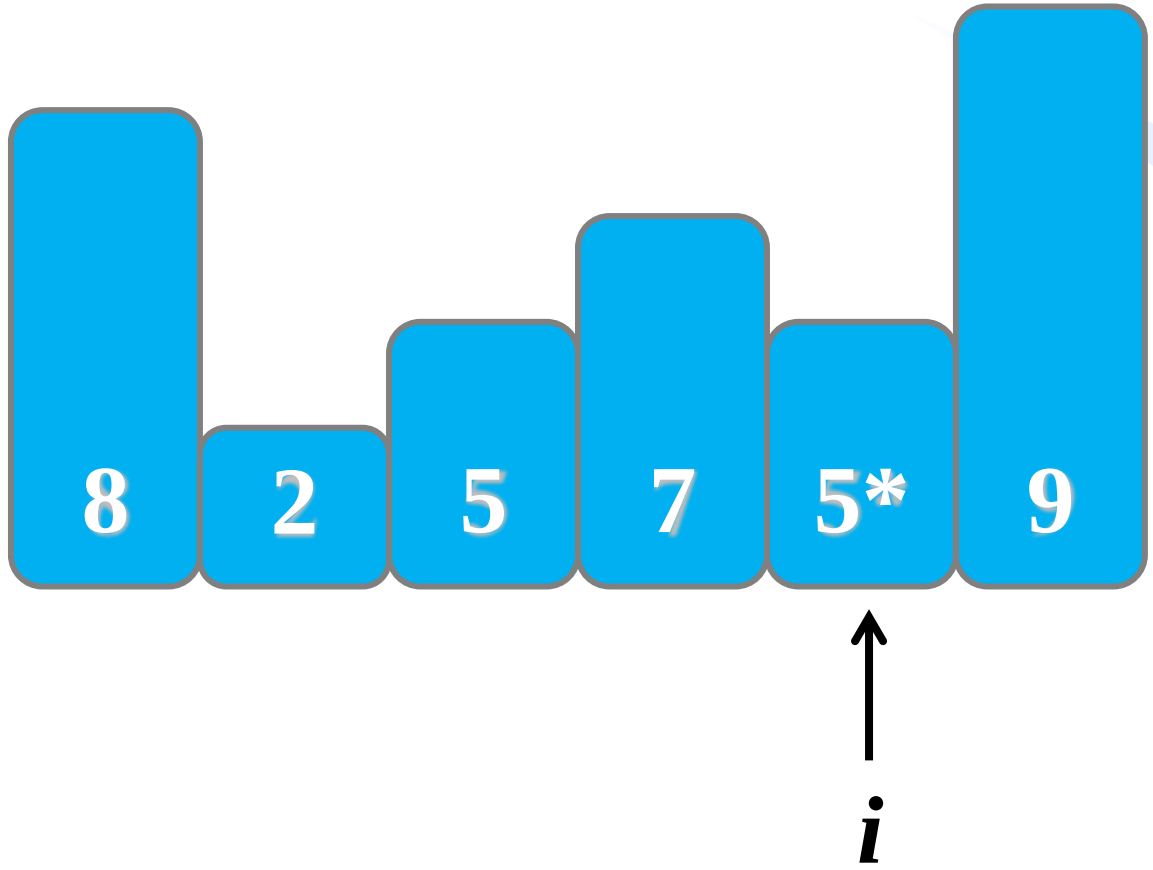
冒泡排序——示例



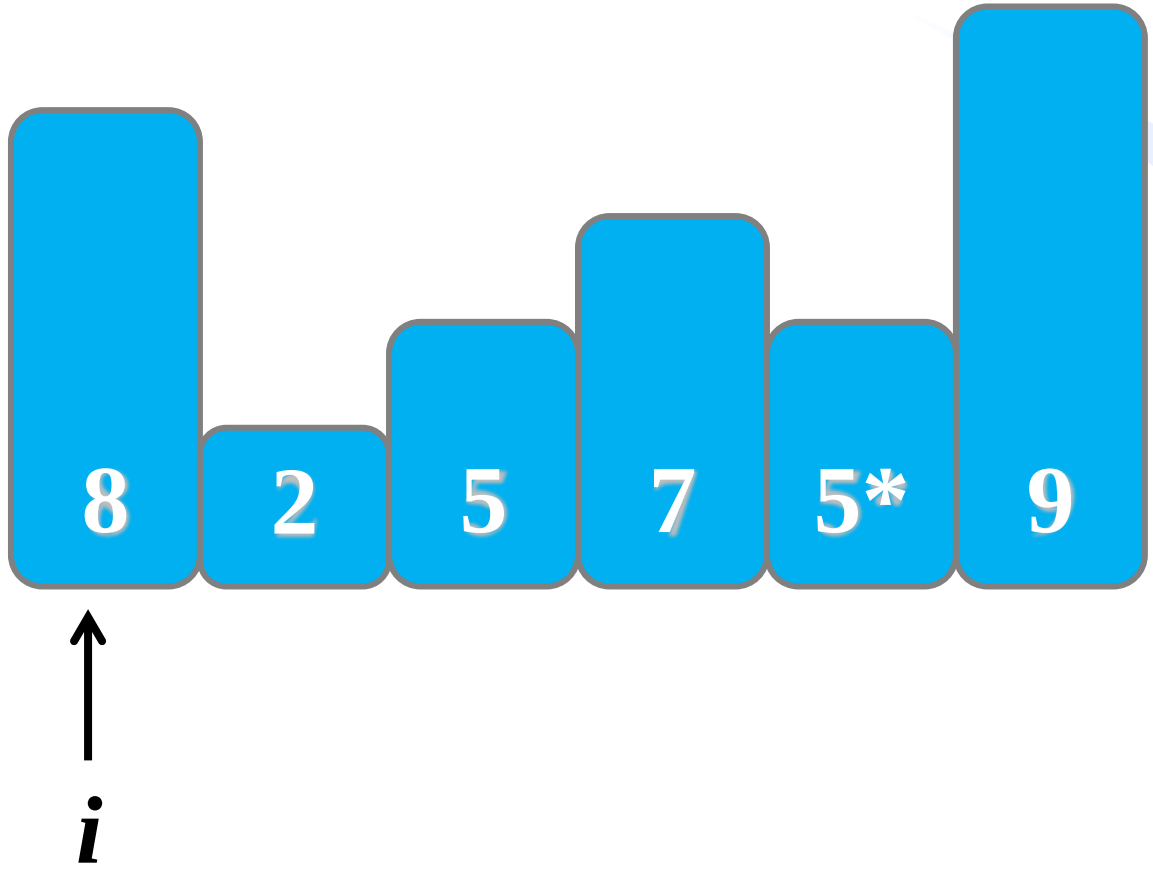
冒泡排序——示例



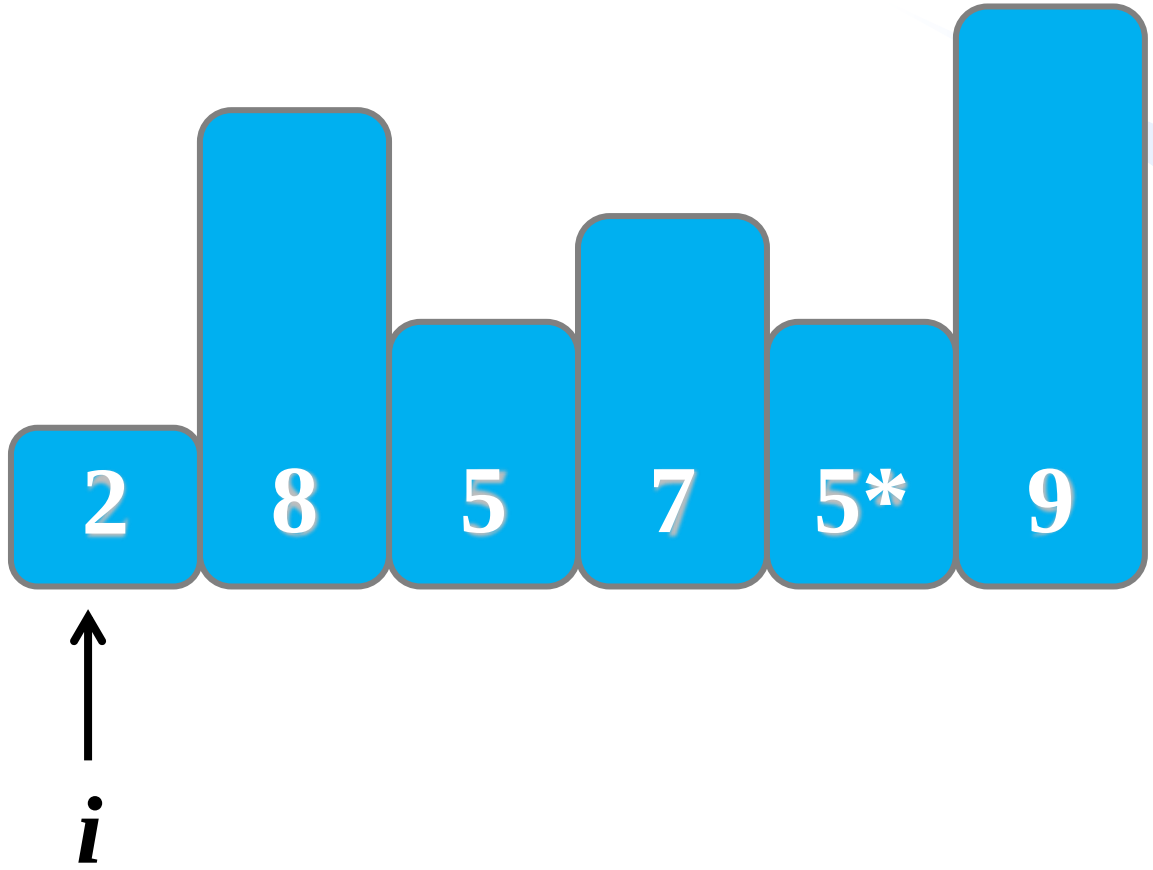
冒泡排序——示例



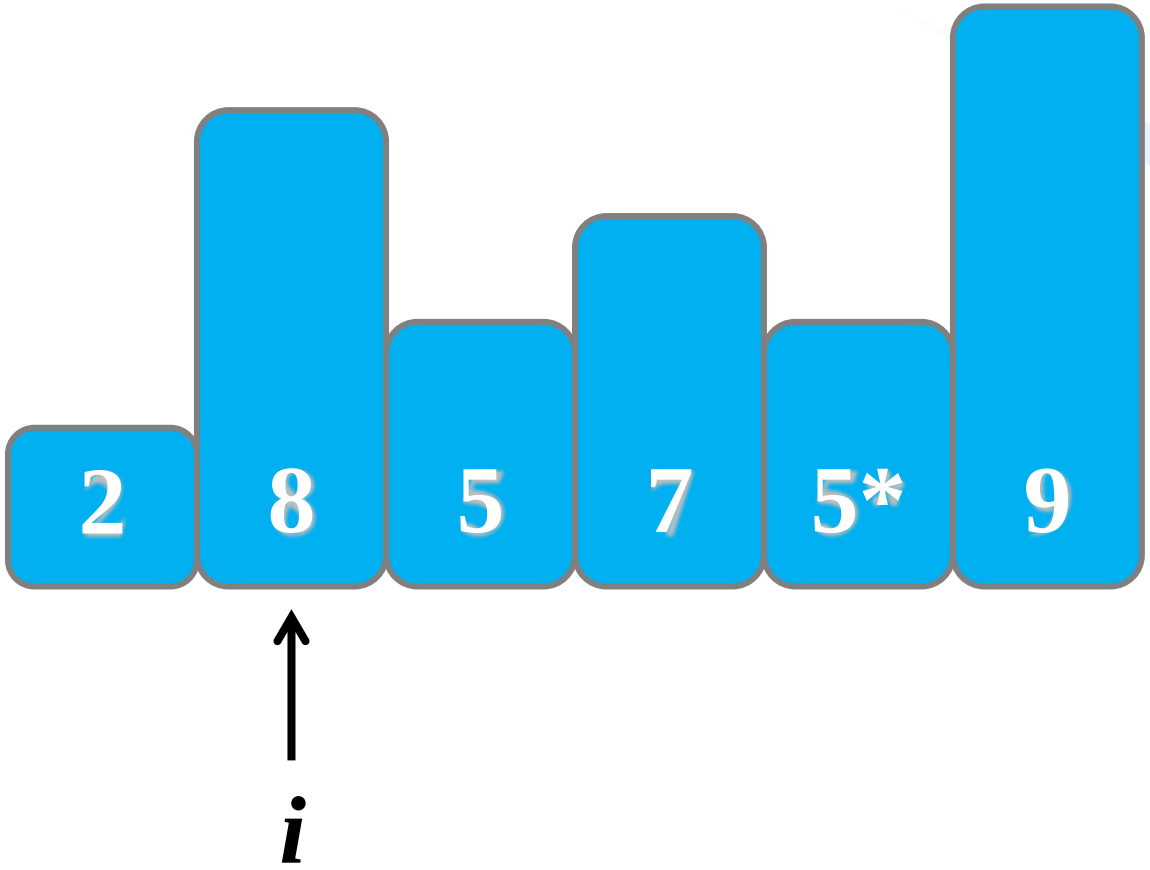
冒泡排序——示例



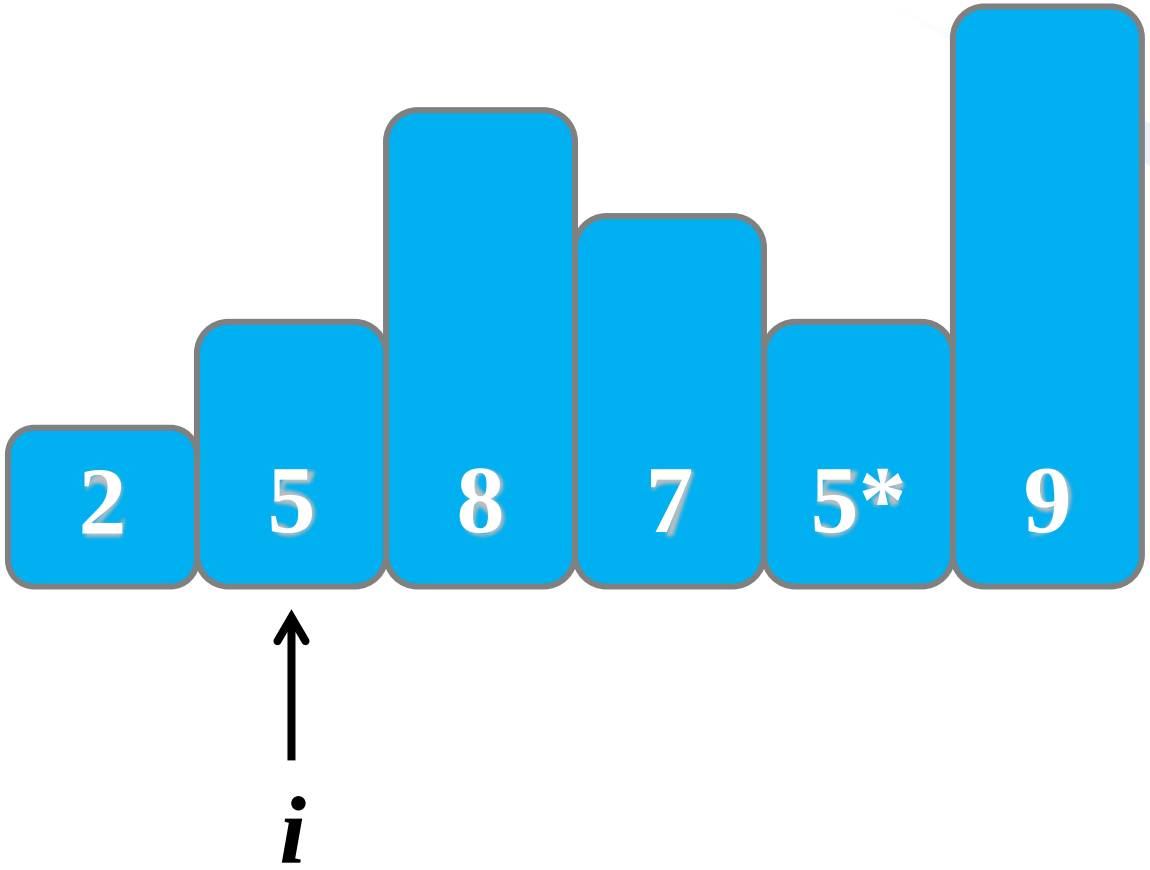
冒泡排序——示例



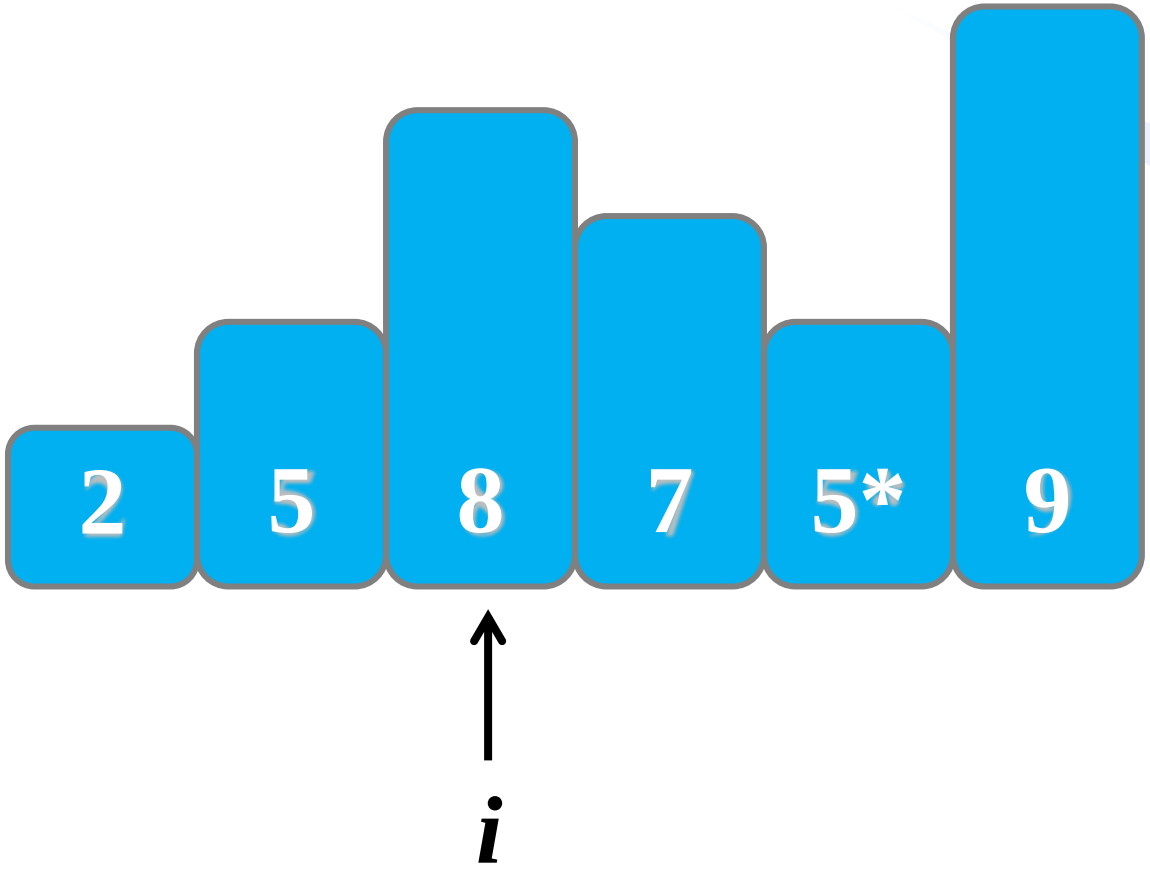
冒泡排序——示例



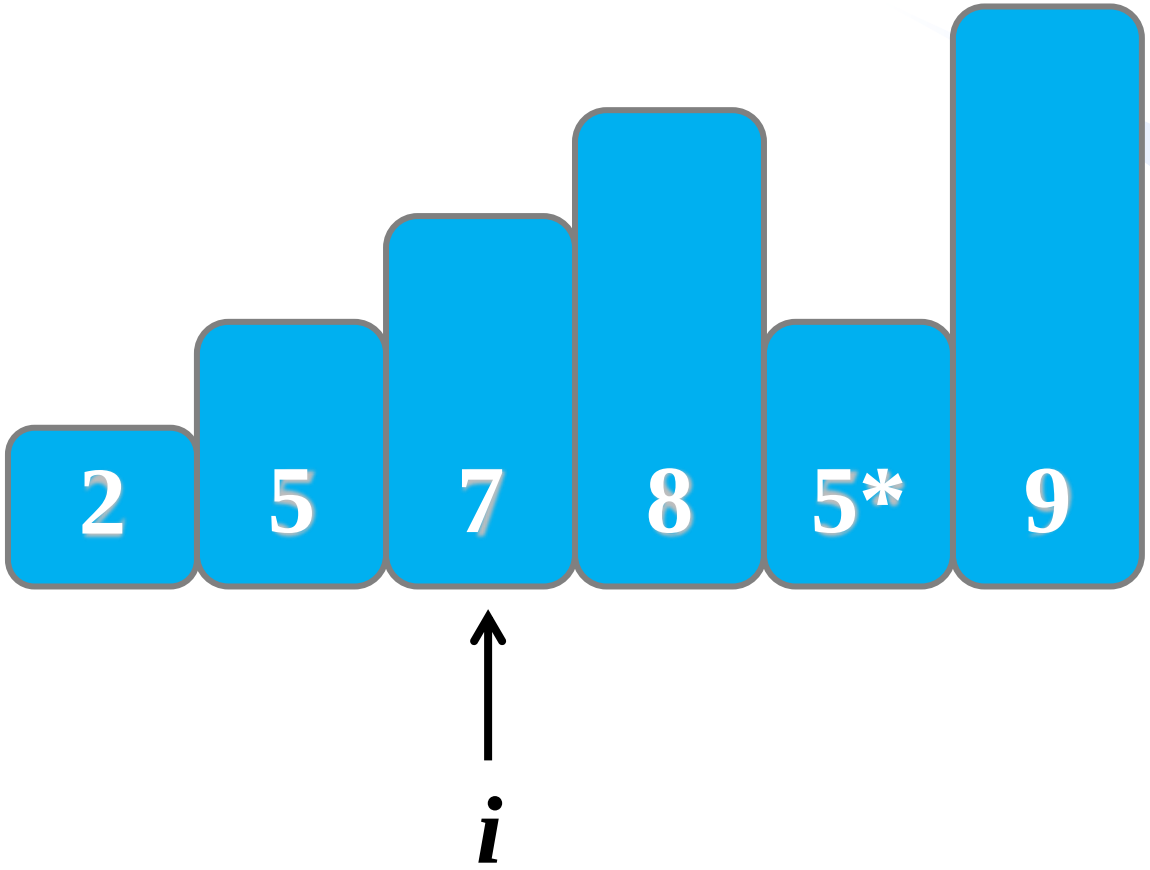
冒泡排序——示例



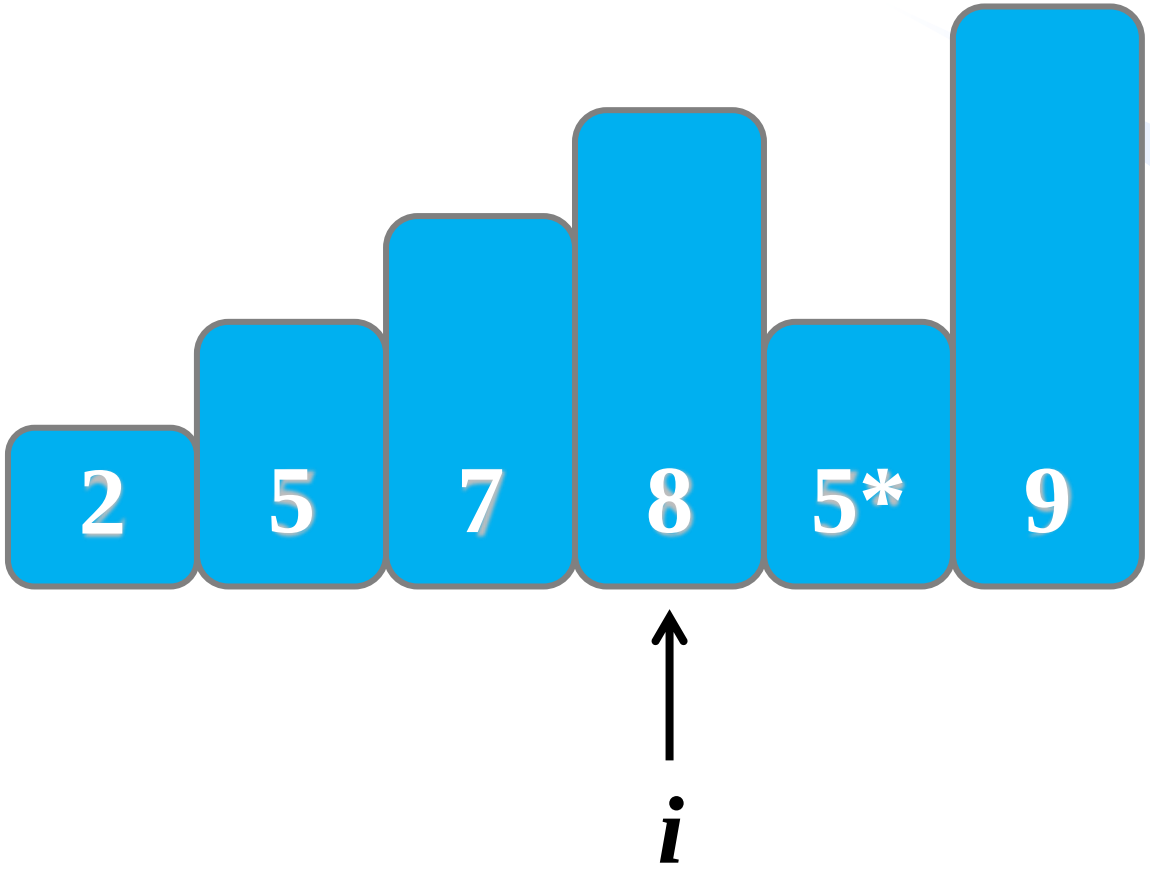
冒泡排序——示例



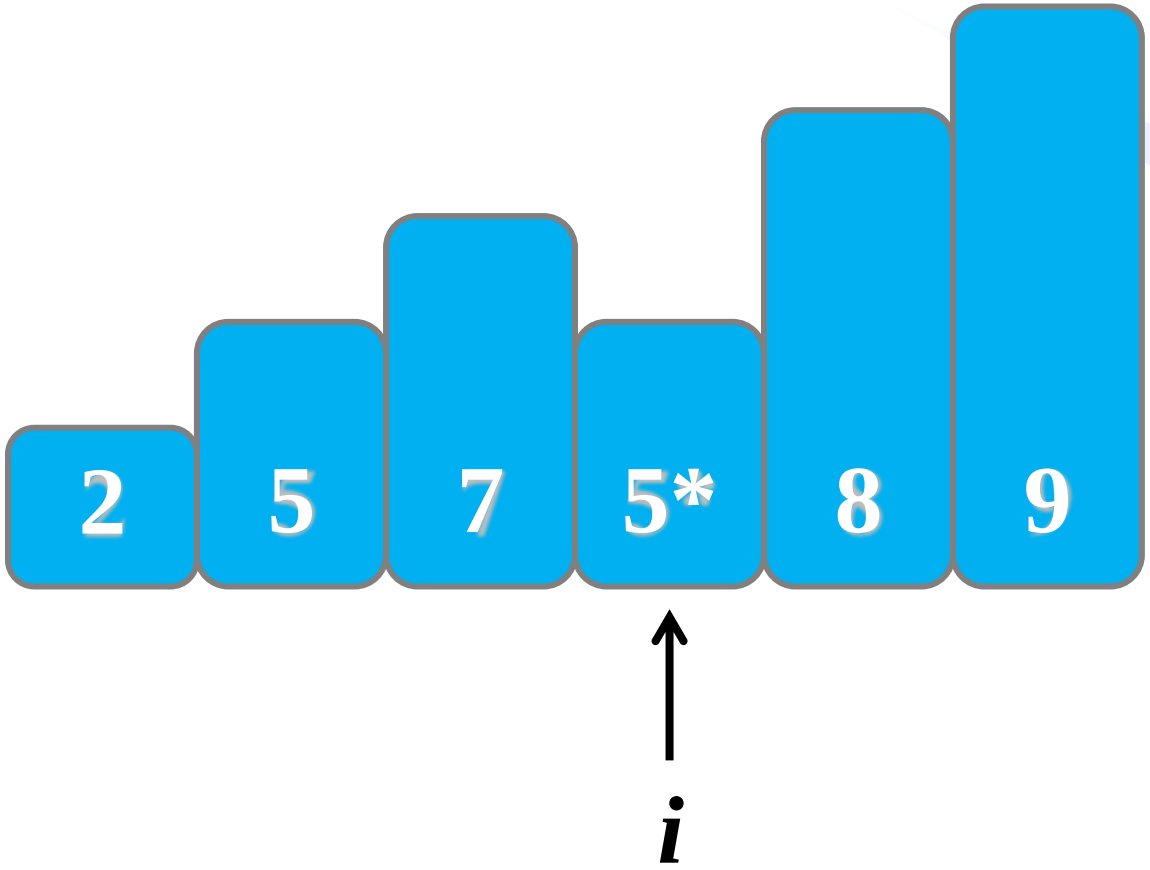
冒泡排序——示例



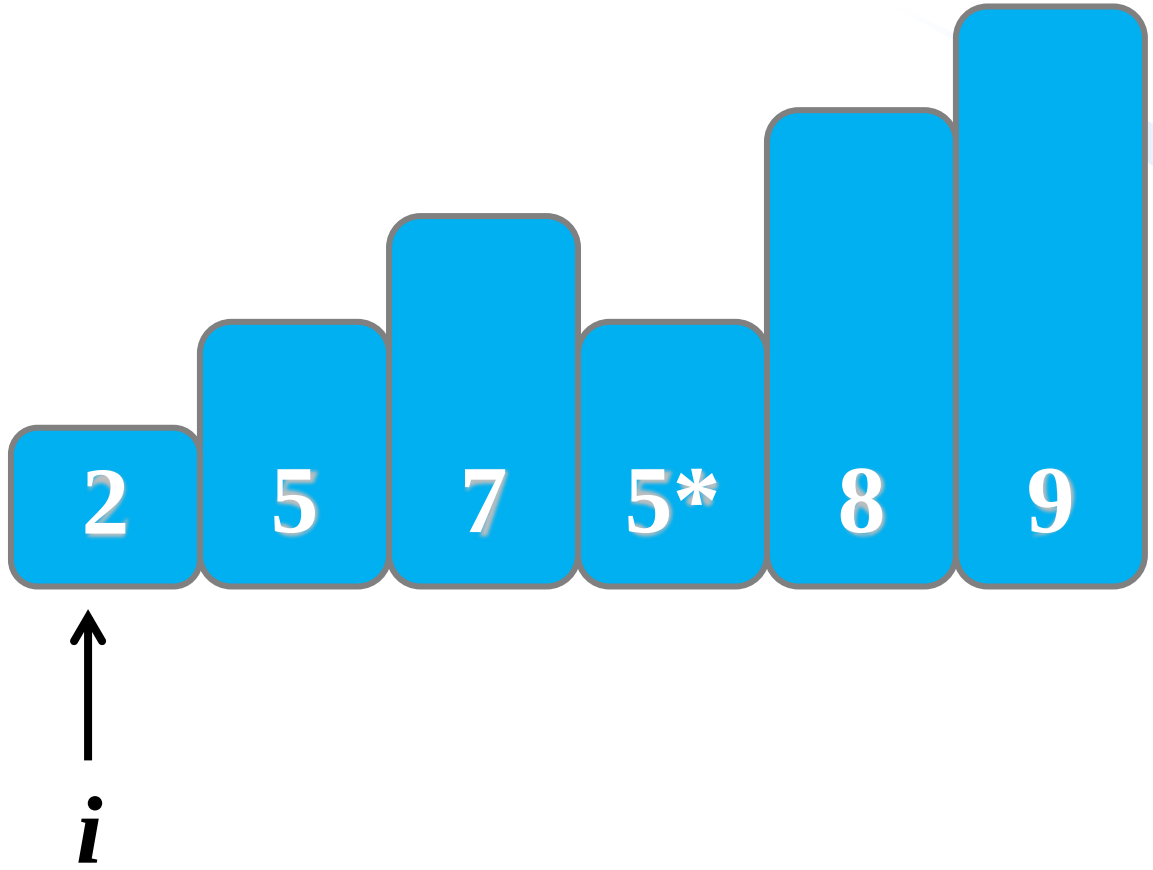
冒泡排序——示例



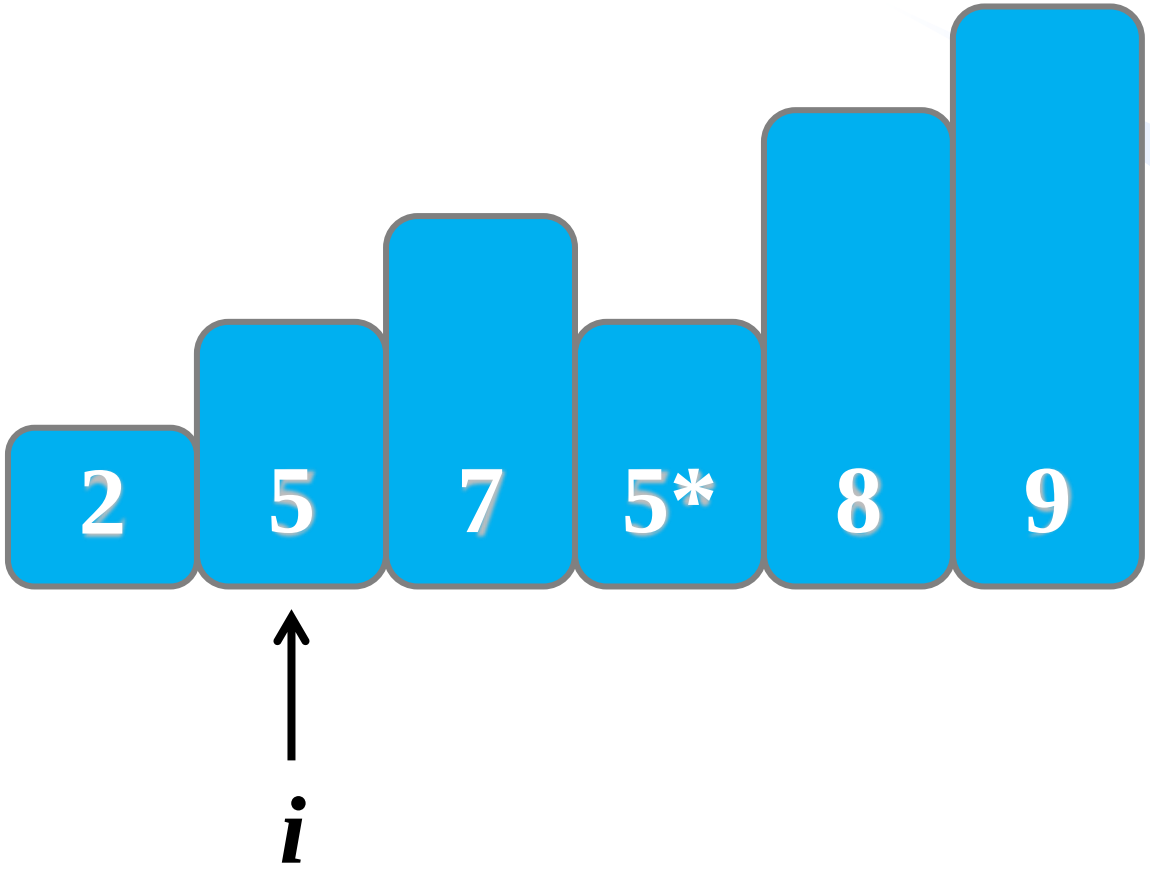
冒泡排序——示例



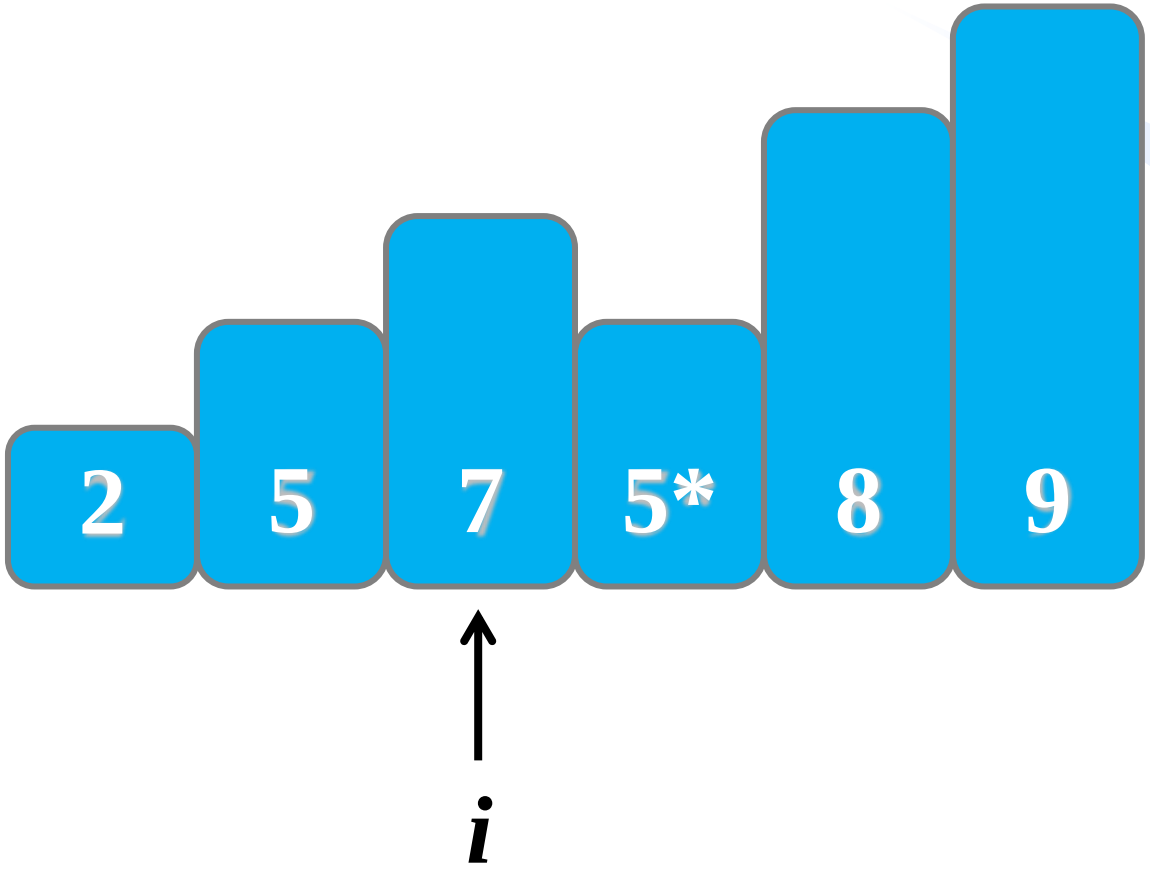
冒泡排序——示例



冒泡排序——示例

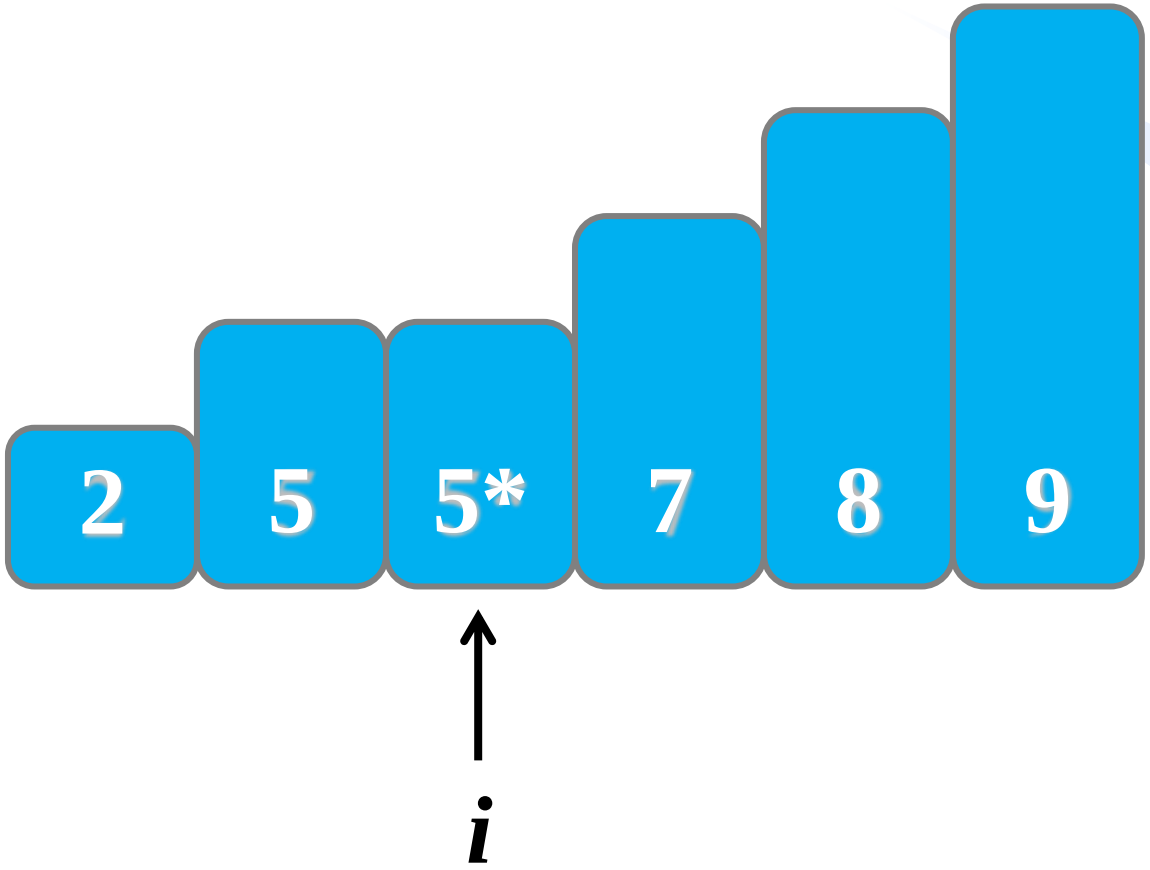


冒泡排序——示例



冒泡排序——示例

A



稳定：前面元素大于后面元素才交换，等于不交换

初级冒泡排序算法——课下阅读

A

```
void SimpleBubbleSort(int R[], int n){
    for(int bound=n; bound>=2; bound--)
        for(int i=1; i<bound; i++)
            if(R[i] > R[i+1])
                swap(R[i],R[i+1]);
}
```

为什么是引用参数?

```
void swap(int &a, int &b){
    int temp = a;
    a = b;
    b = temp;
}
```

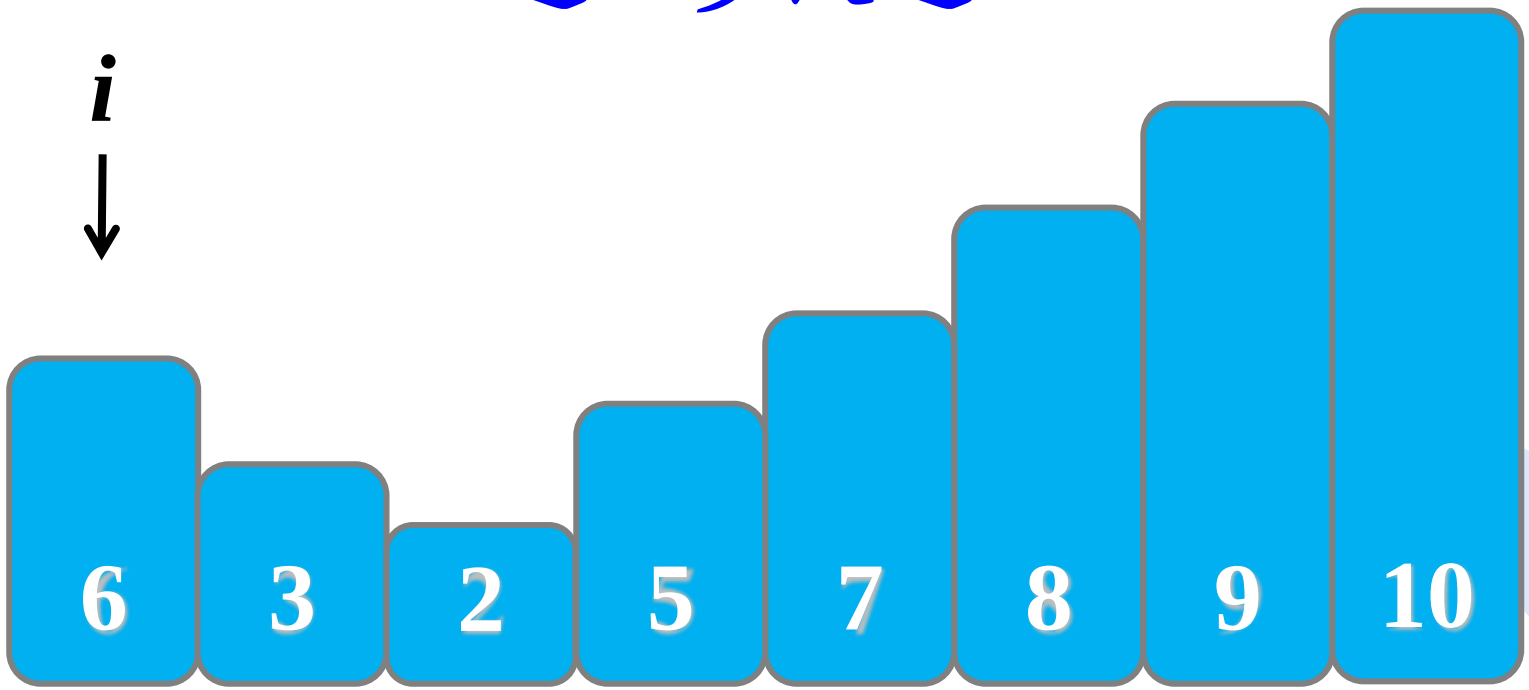
时间复杂度
$O(n^2)$

冒泡排序——特点

A

- 经过第1趟冒泡，最大元素移至最终位置（第 *n* 个位置）；
- 经过第2趟冒泡，第2大元素移至最终位置（第 *n*-1 个位置）；
- 经过第3趟冒泡，第3大元素移至最终位置（第 *n*-2 个位置）；
- ……

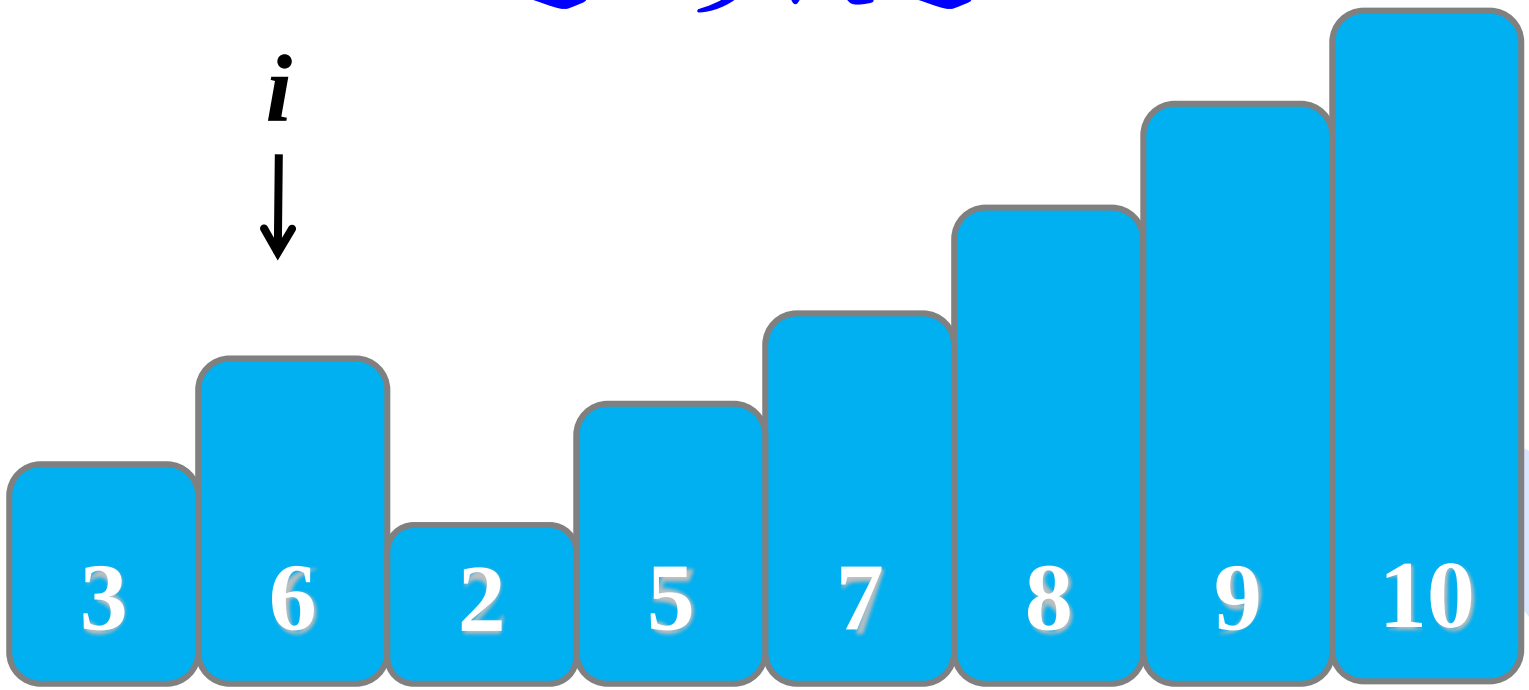
进一步改进



- 在每趟冒泡过程中，当比较结束后，如果发现位置t是最后一次元素交换的位置，即说明从R_{t+1}~R_n已经排序。
- 从而下一趟比较只要进行到位置t即可。这样可以减少算法的关键词比较次数。

进一步改进

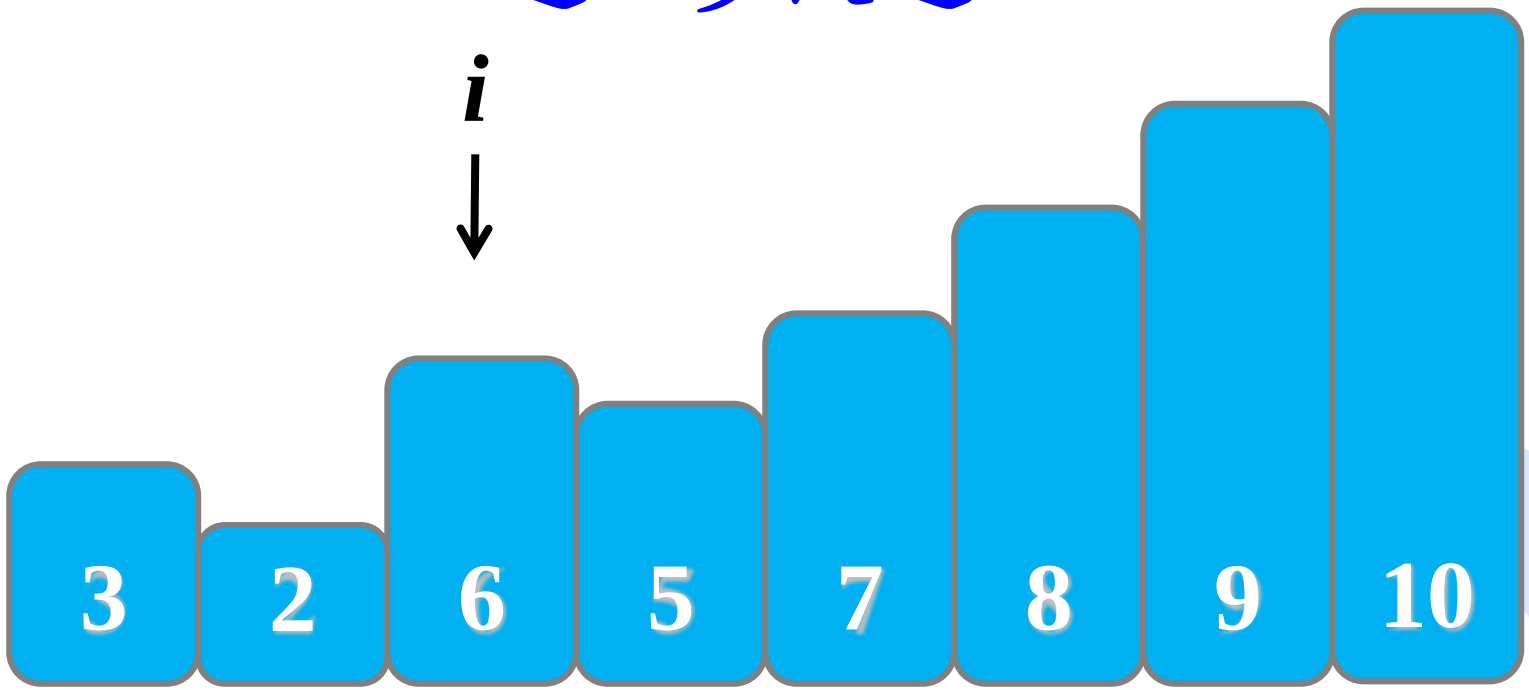
A



- 在每趟冒泡过程中，当比较结束后，如果发现位置t是最后一次元素交换的位置，即说明从R_{t+1}~R_n已经排序。
- 从而下一趟比较只要进行到位置t即可。这样可以减少算法的关键词比较次数。

进一步改进

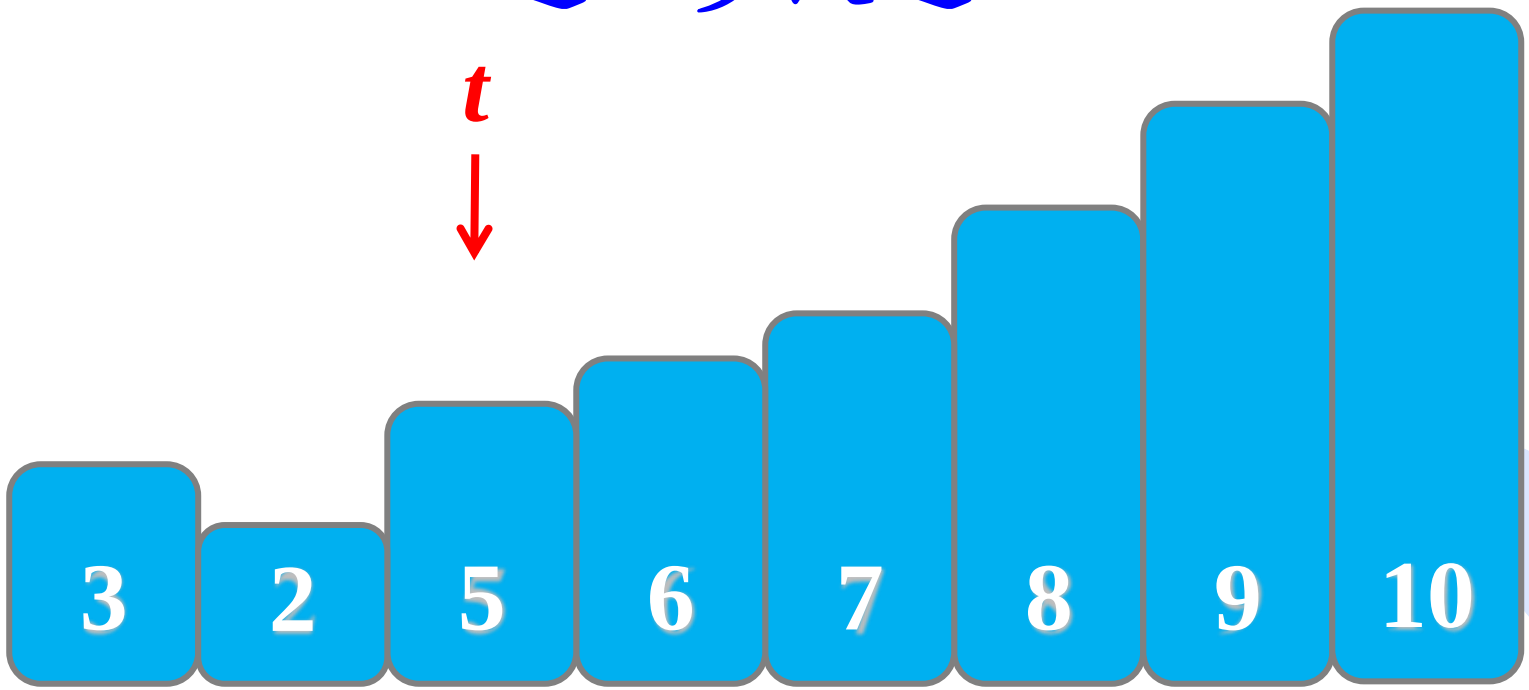
A



- 在每趟冒泡过程中，当比较结束后，如果发现位置t是最后一次元素交换的位置，即说明从R_{t+1}～R_n已经排序。
- 从而下一趟比较只要进行到位置t即可。这样可以减少算法的关键词比较次数。

进一步改进

A



- 在每趟冒泡后，如果发现位置t是最后一次元素交换的位置，即说明从R_{t+1}~R_n已经排序。
- 从而下一趟比较只要进行到位置t即可，这样可以减少算法的关键词比较次数。

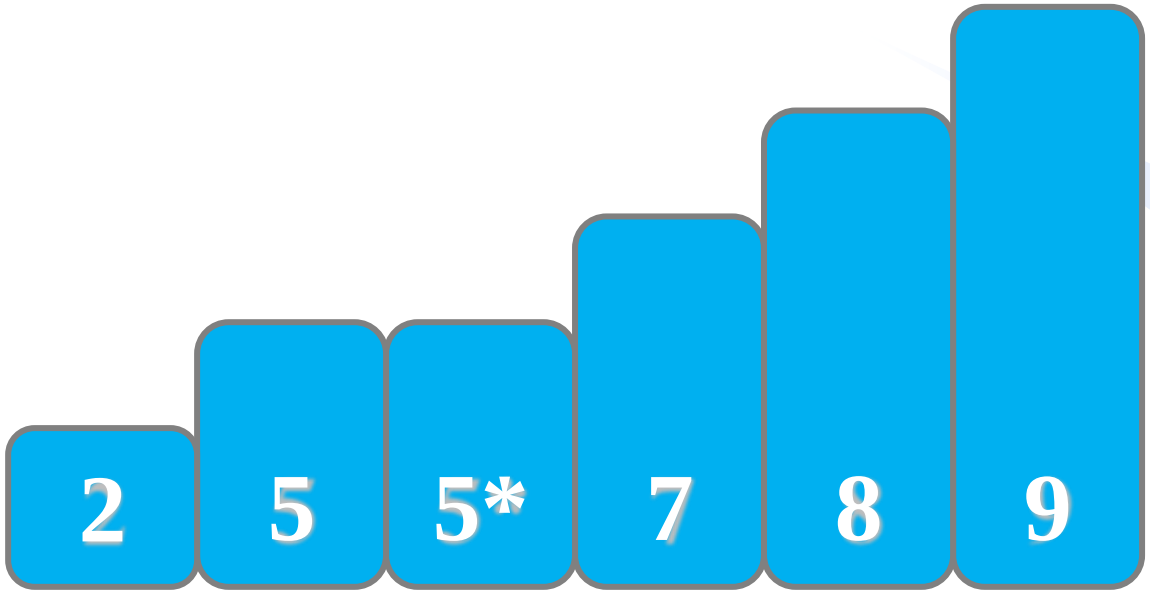
冒泡排序算法——课下阅读

A

```
void BubbleSort(int R[], int n){
    int bound=n; //每趟冒泡关键词比较的终止位置
    while(bound>0){
       int t=0;   //本趟冒泡元素交换的最后位置
       for(int i=1; i<bound; i++)
          if(R[i]>R[i+1]){ swap(R[i],R[i+1]); t=i; }
       bound=t;
    }
}
```

冒泡排序——时间复杂度

A

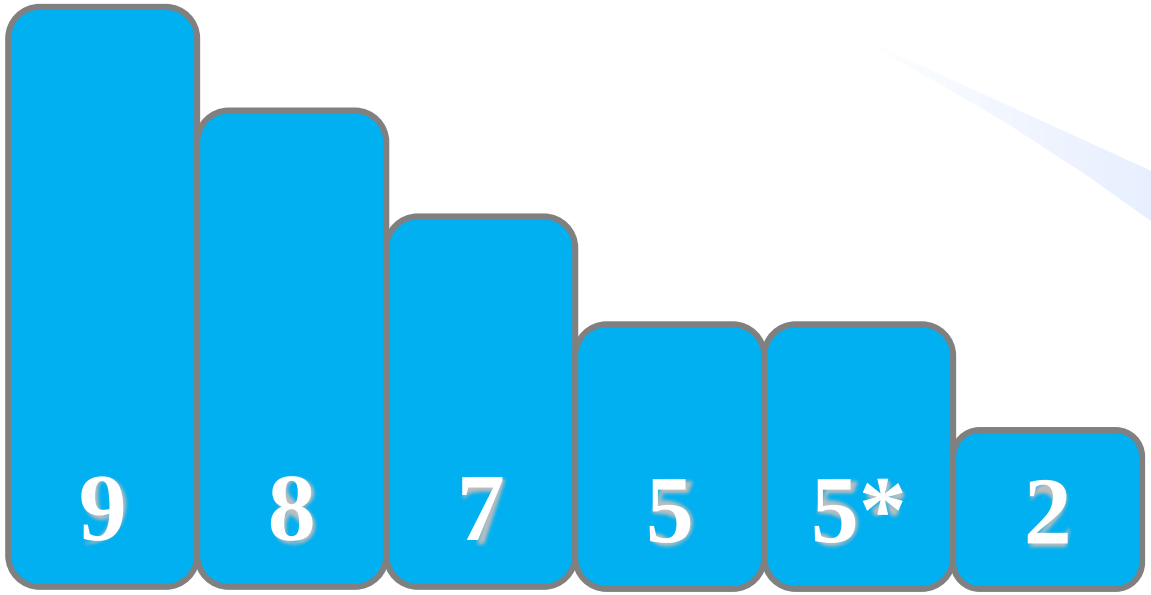


待排序文件已有序，只需一趟冒泡

最好情况：已经排好序
时间复杂度：O(*n*)

直接插入排序——时间复杂度

A



需要O(n)趟冒泡，且每趟冒泡都需要O(n)次关键词比较和元素移动

最坏情况：初始为逆序

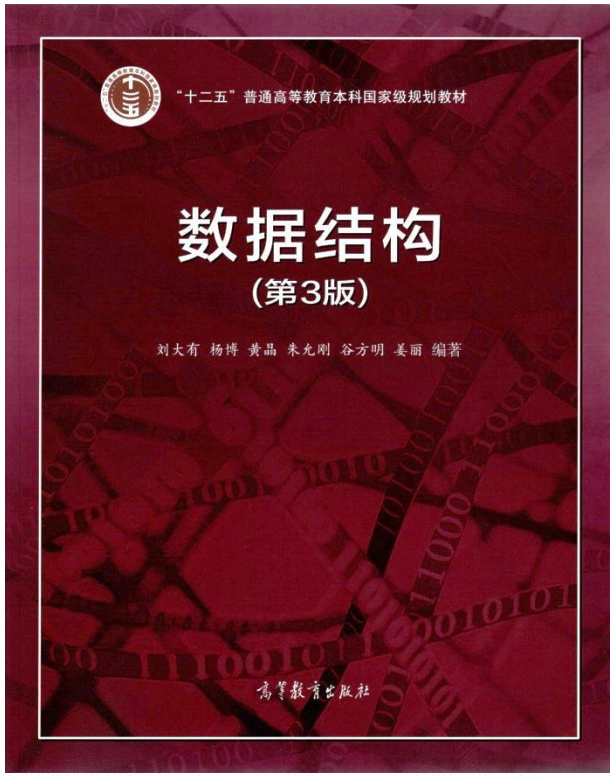
时间复杂度：O(n^2)

冒泡排序总结

A

排序算法	时间复杂度			空间复杂度	稳定性
	最好	平均	最坏		
冒泡排序	$O(n)$	$O(n^2)$	$O(n^2)$	$O(1)$	稳定

可以采用气泡上浮和下沉交替的方法加速（见慕课），但时间复杂度的阶不变。



平方阶排序算法

- 排序概念
- 回顾直接插入排序
- 回顾冒泡排序
- **回顾直接选择排序**

数据之法 结构之美 算法之道

zhuyungang@jlu.edu.cn

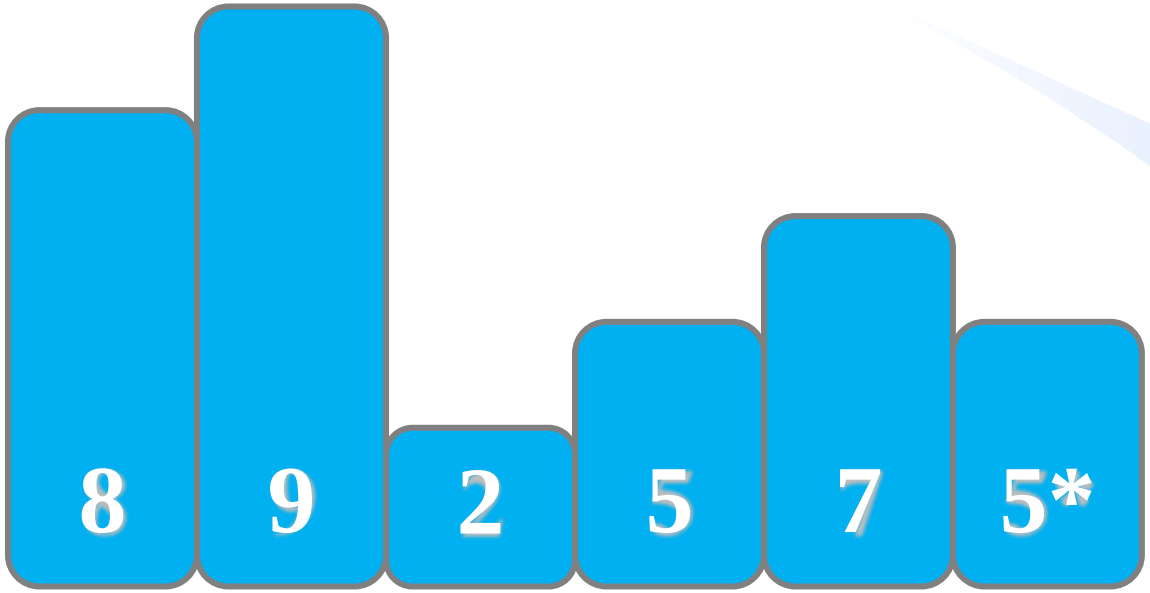
Last updated on 2025.5

直接选择排序

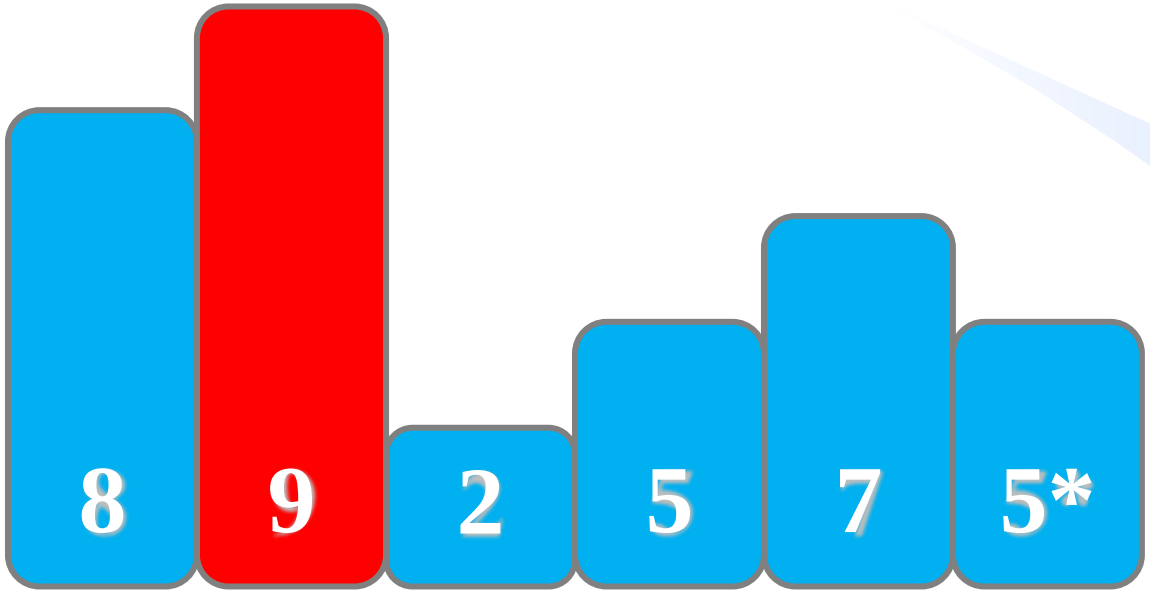
A

① 在前n个元素里找最大元素，与$R[n]$交换，使$R[n]$就位
② 在前n-1个元素里找最大元素，与$R[n-1]$交换，使$R[n-1]$就位
③ 在前n-2个元素里找最大元素，与$R[n-2]$交换，使$R[n-2]$就位
④ 在前n-3个元素里找最大元素，与$R[n-3]$交换，使$R[n-3]$就位
⑤ ……
⑥ 以此类推，最后形成递增的有序序列。

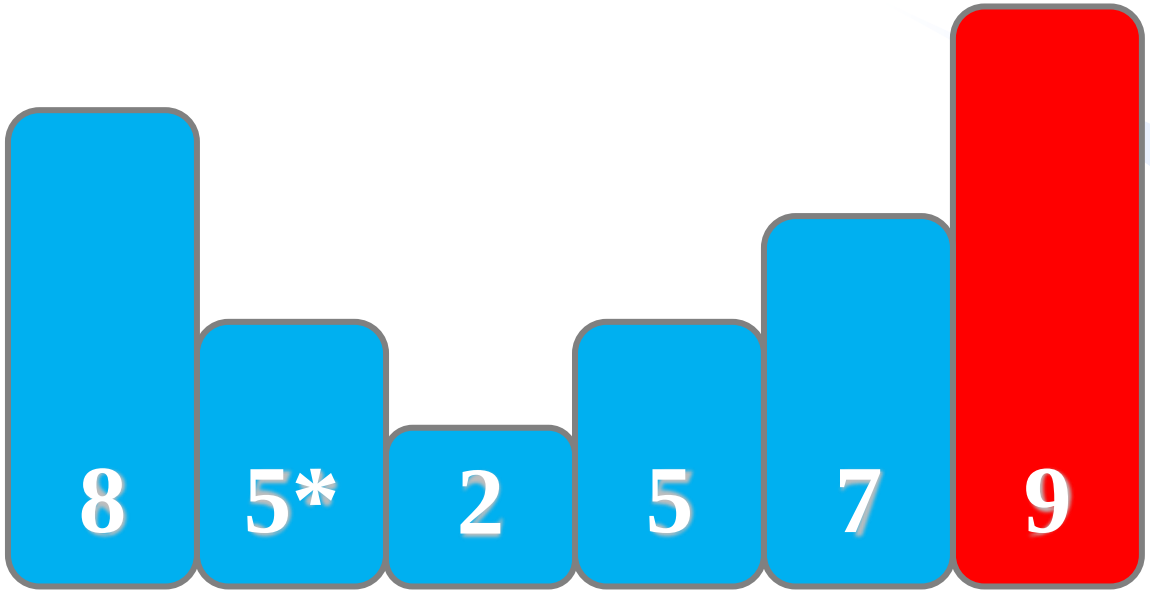
直接选择排序——示例



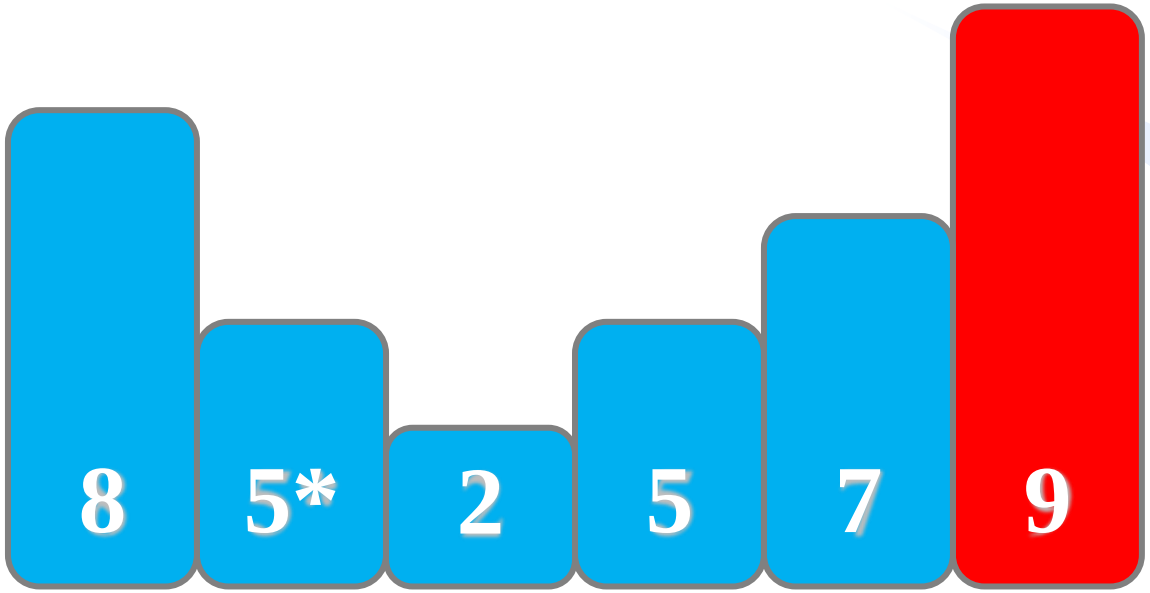
直接选择排序——示例



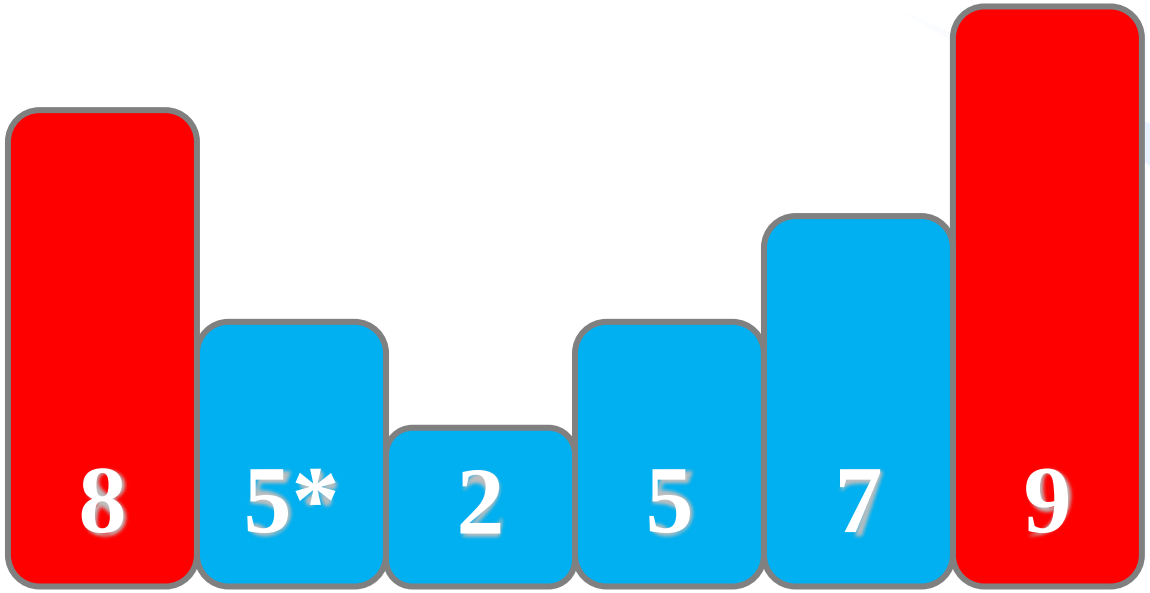
直接选择排序——示例



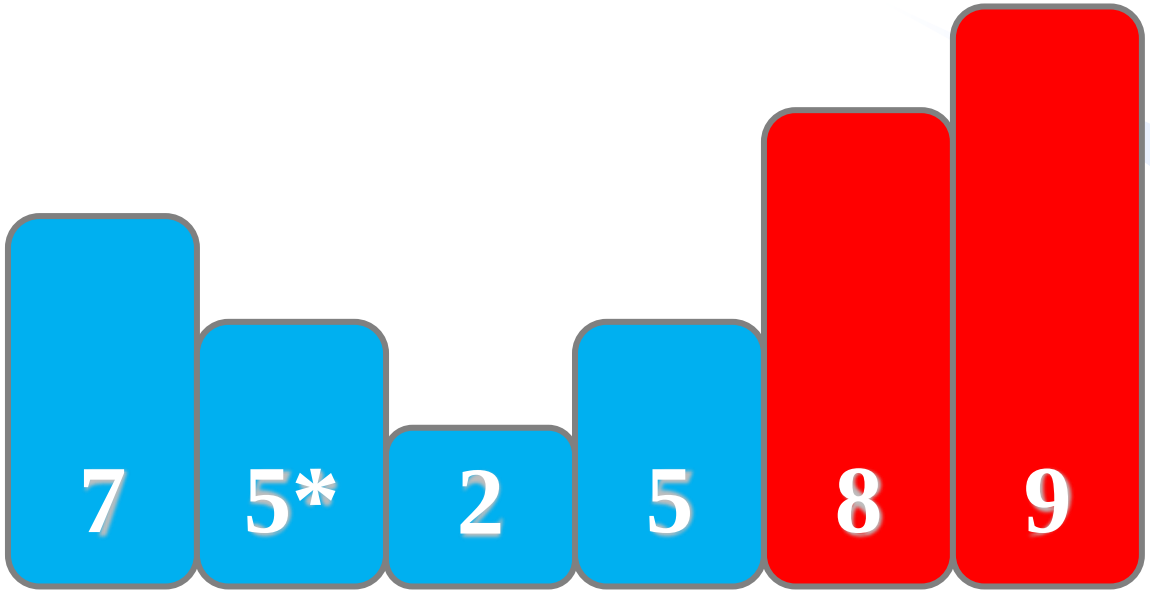
直接选择排序——示例



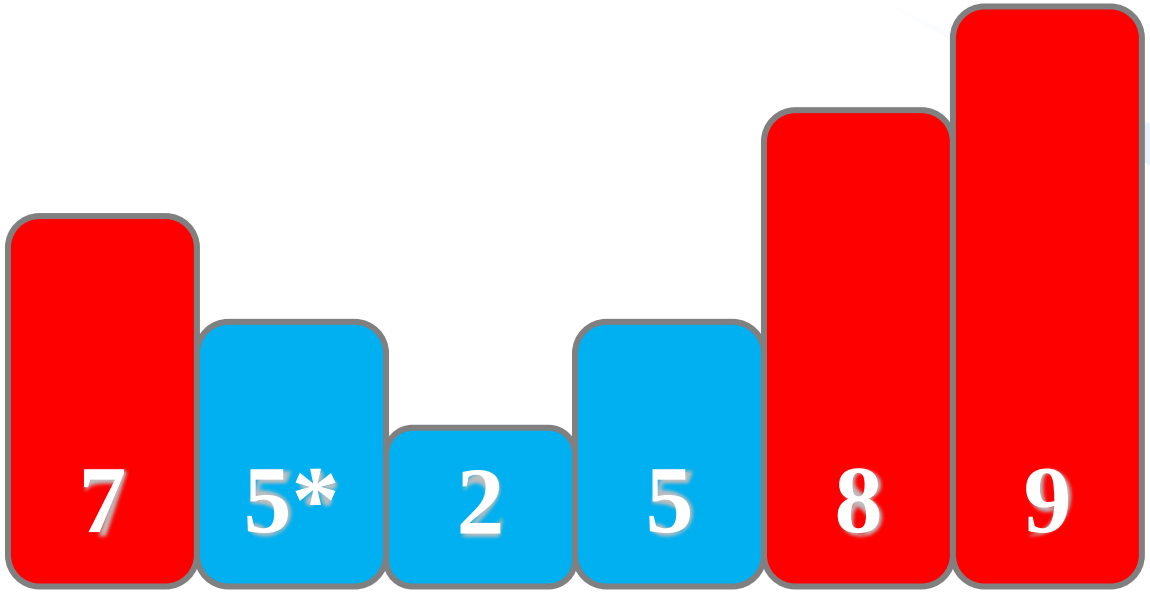
直接选择排序——示例



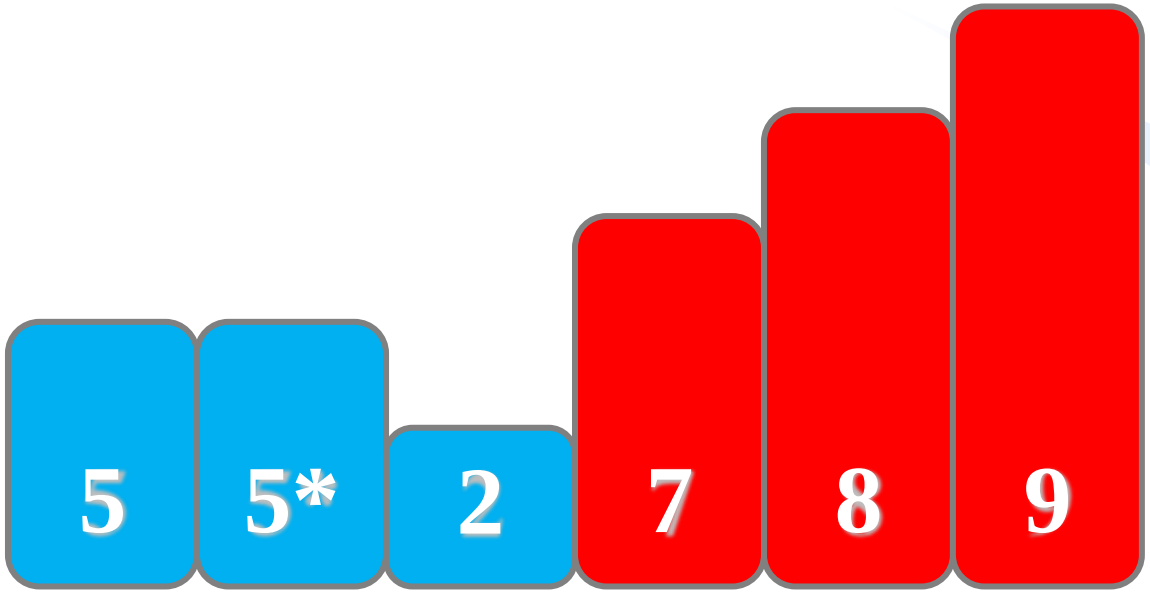
直接选择排序——示例



直接选择排序——示例

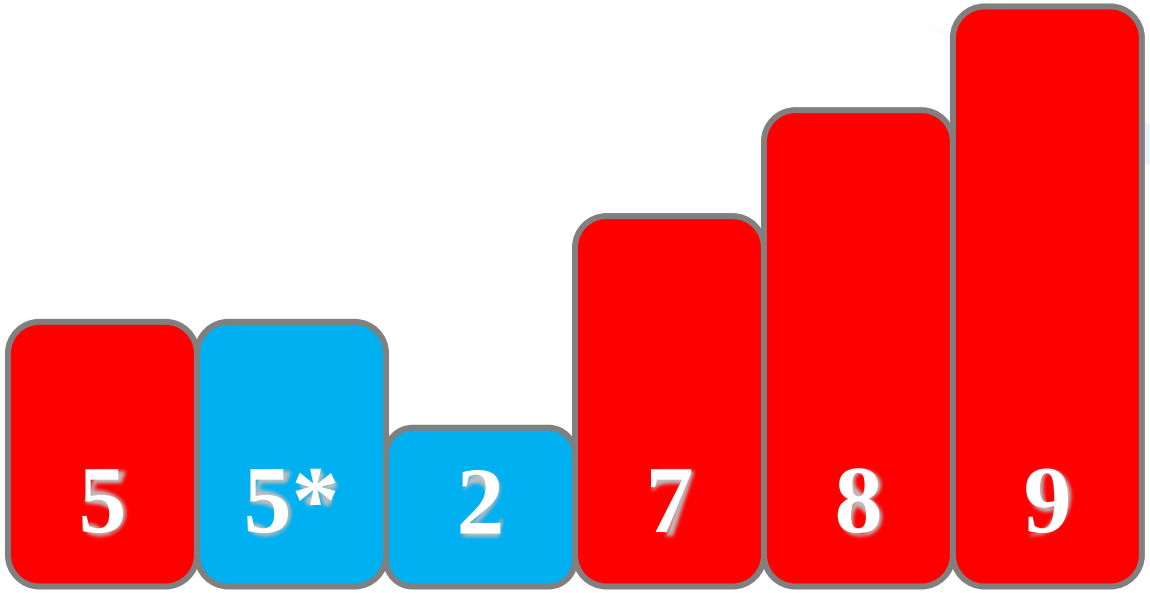


直接选择排序——示例



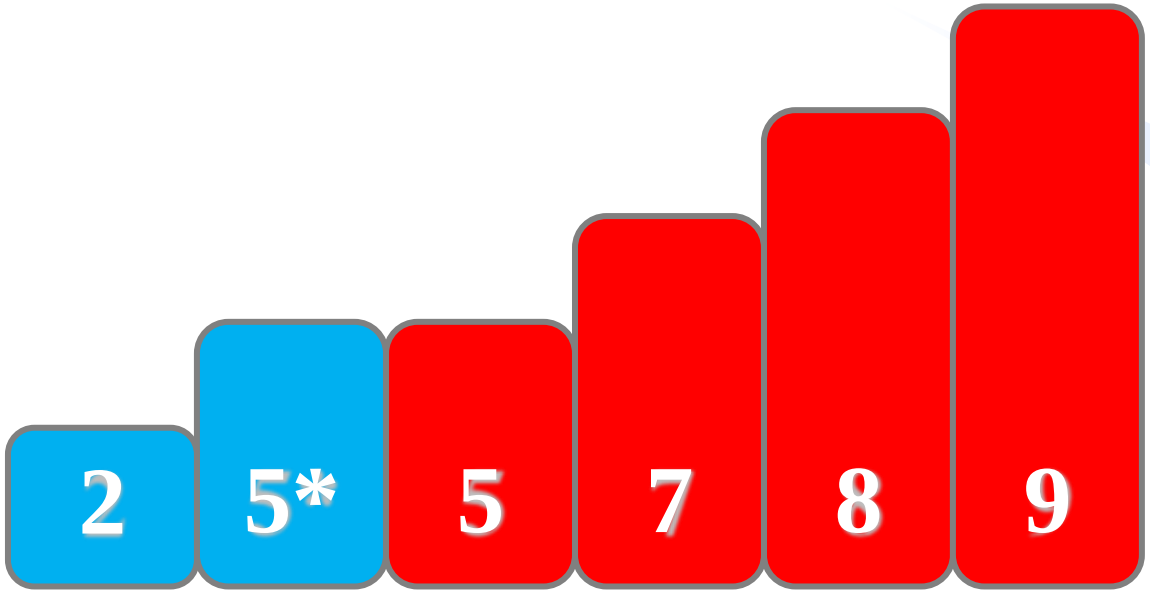
多个最大元素时，选择第1个最大的元素

直接选择排序——示例

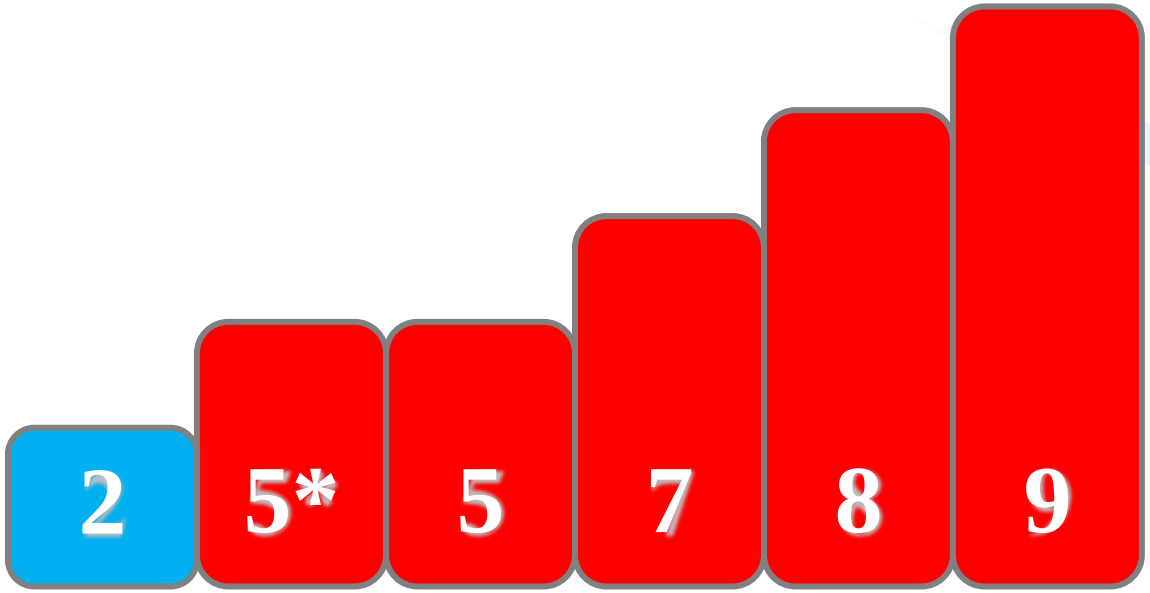


多个最大元素时，选择第1个最大的元素

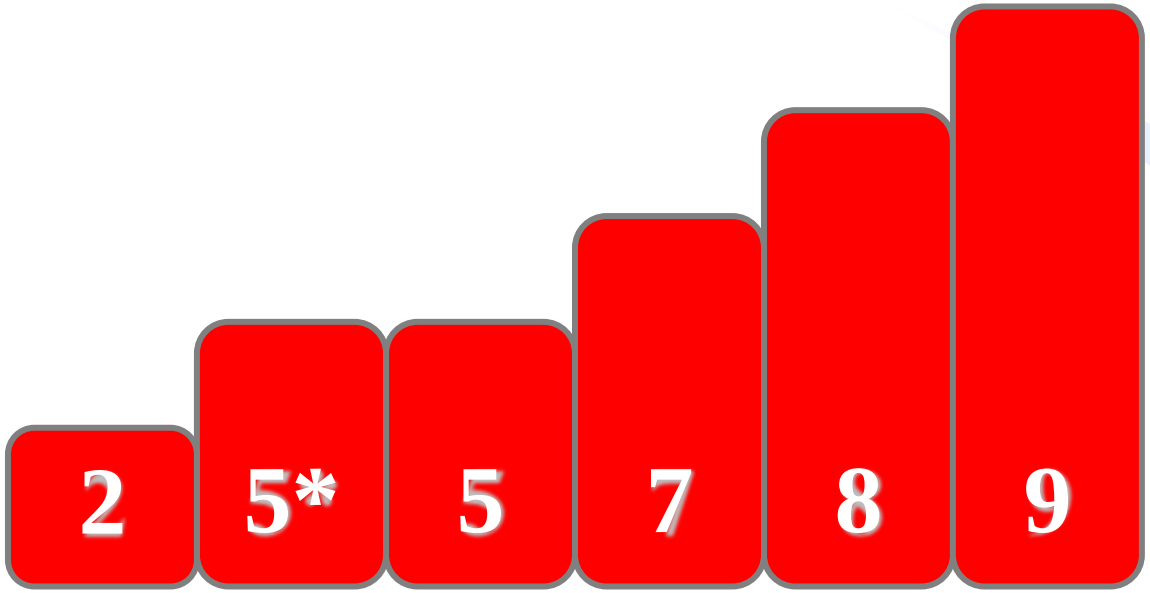
直接选择排序——示例



直接选择排序——示例



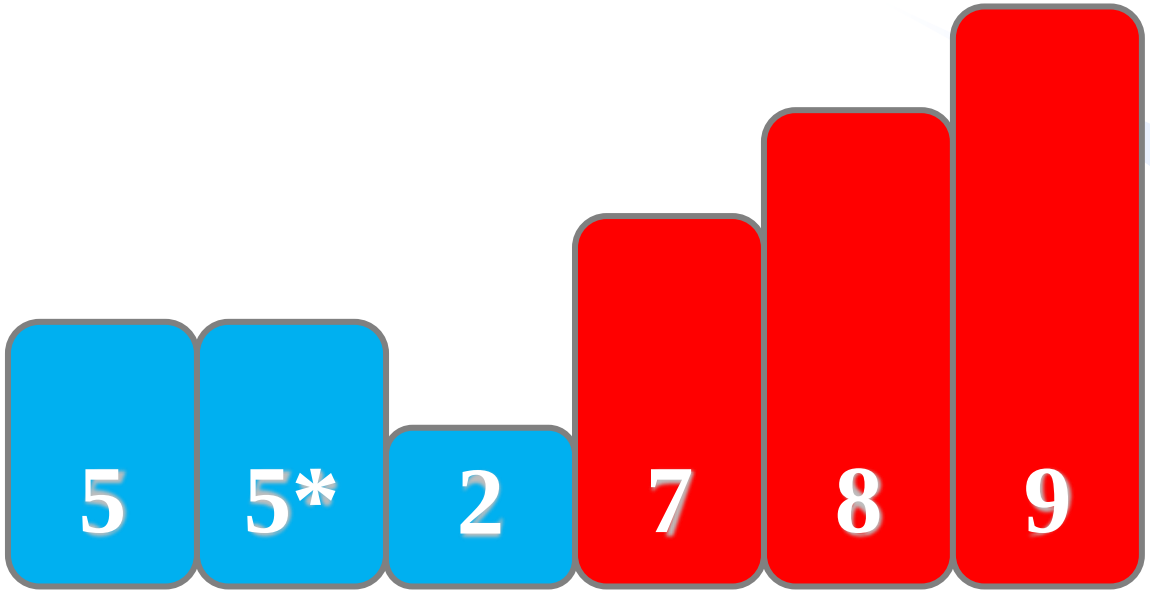
直接选择排序——示例



不稳定？

直接选择排序——稳定性

A



有多个最大元素时，若选最后1个最大元素，能否保证稳定？

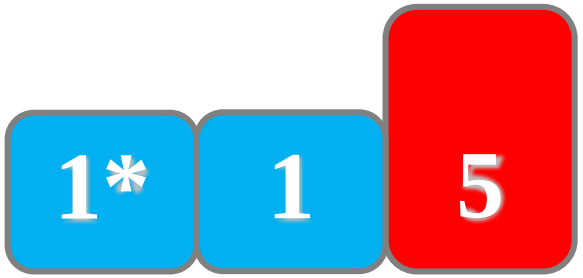
直接选择排序——稳定性

A

有多个最大元素时，若选最后1个最大元素，能否保证稳定？

直接选择排序——稳定性

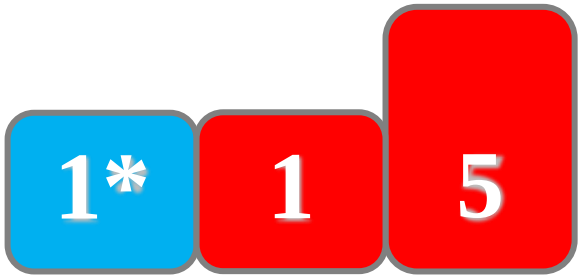
A



有多个最大元素时，若选最后1个最大元素，能否保证稳定？

直接选择排序——稳定性

A



有多个最大元素时，若选最后1个最大元素，能否保证稳定？

直接选择排序——稳定性

A

有多个最大元素时，若选最后1个最大元素，能否保证稳定？

直接选择排序算法——课下阅读

A

在前n个元素里找最大元素，与R[n]交换，使R[n]就位
在前n-1个元素里找最大元素，与R[n-1]交换，使R[n-1]就位
在前n-2个元素里找最大元素，与R[n-2]交换，使R[n-2]就位
在前n-3个元素里找最大元素，与R[n-3]交换，使R[n-3]就位
……

```
for(int i=n; i>=1; i--){ //i标识当前处理的子数组的右边界
    在R[1]…R[i]里找最大元素，与R[i]交换;
}
```

直接选择排序算法——课下阅读

A

```
void SelectionSort(int R[], int n){
    for(int i=n; i>=1; i--){
        //i标识当前处理的子数组的右边界
        //在子数组R[1]...R[i]里找最大元素，和R[i]交换
        int max=1;
        for(int j=2; j<=i; j++)
            if(R[j]>R[max]) max=j;
        swap(R[max],R[i]);
    }
}
```

关键词比较次数与元素的初始排列无关

时间复杂度 $O(n^2)$

1 2 3 … … … n-2 n-1 n

直接选择排序算法总结

A

排序算法	时间复杂度			空间复杂度	稳定性
	最好	平均	最坏		
直接选择排序	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(1)$	不稳定