各种排序

《前端精进 - 科班方向》

版权声明

联系方式

如果你想要购买本课程
请微信联系 xiedaimala02 或 xiedaimala03

如果你发现有人盗用本课程
请微信联系 xiedaimala02 或 xiedaimala03

1 选择排序

- 公式

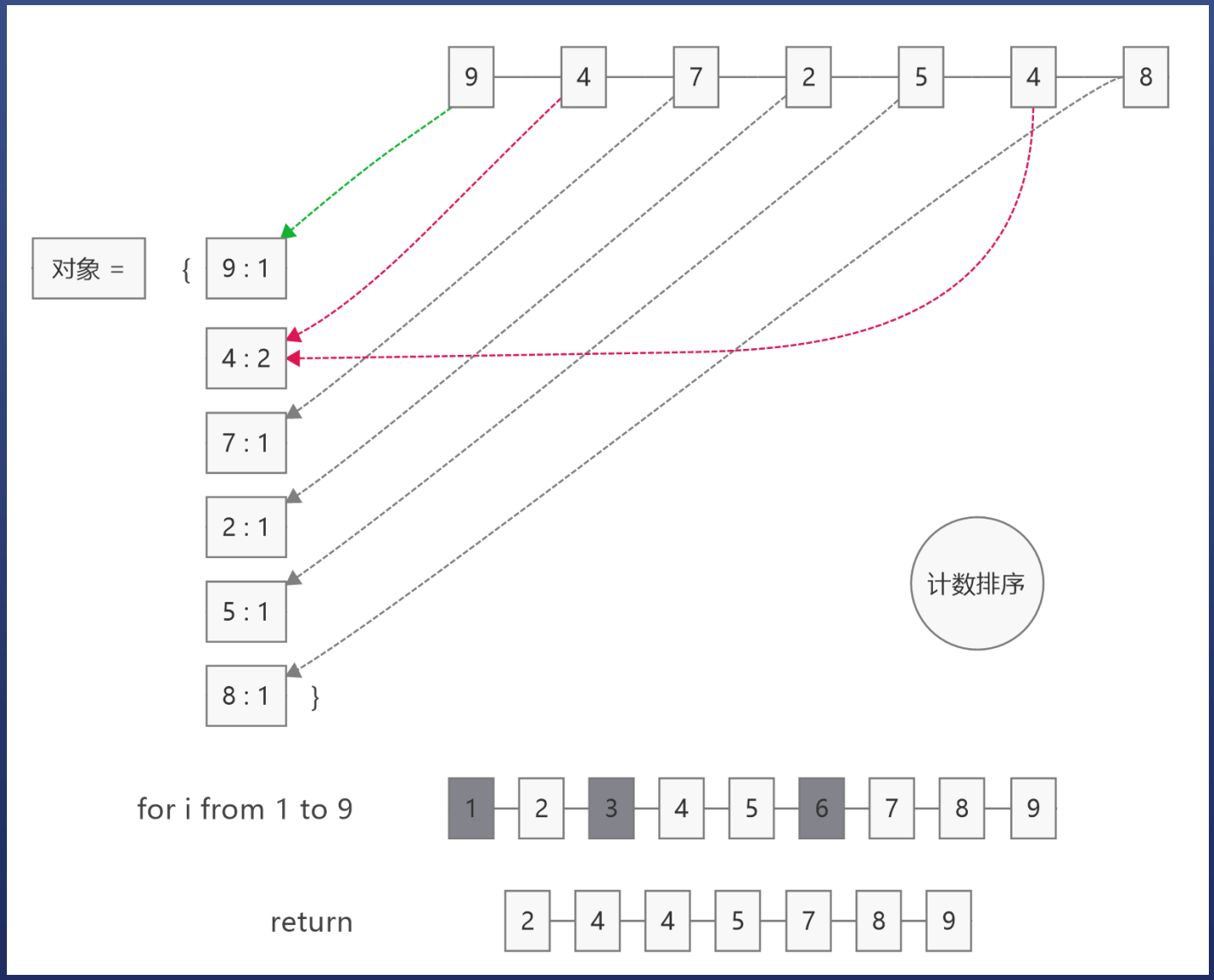
$$sort(a_n) = \begin{cases} a_n,\ n = 1 \\ [\min(a_n)] + sort(a_{n-1}) \end{cases}$$

- 时间复杂度
  - ✓ n + (n-1) + (n-2) + ... + 1
  - ✓ 约等于 n + n + n + ... + n
  - ✓ 即 O(n^2)

思维特点

- 分而治之
  - ✓ 把一个问题分解为两个相对简单的问题
- 降低规模
  - ✓ 每次规模减少1
- 对比
  - ✓ 目前学的归并排序、快速排序、选择排序都有这些特点
  - ✓ 只不过归并规模降低地最快，快排次之，选择排序最后

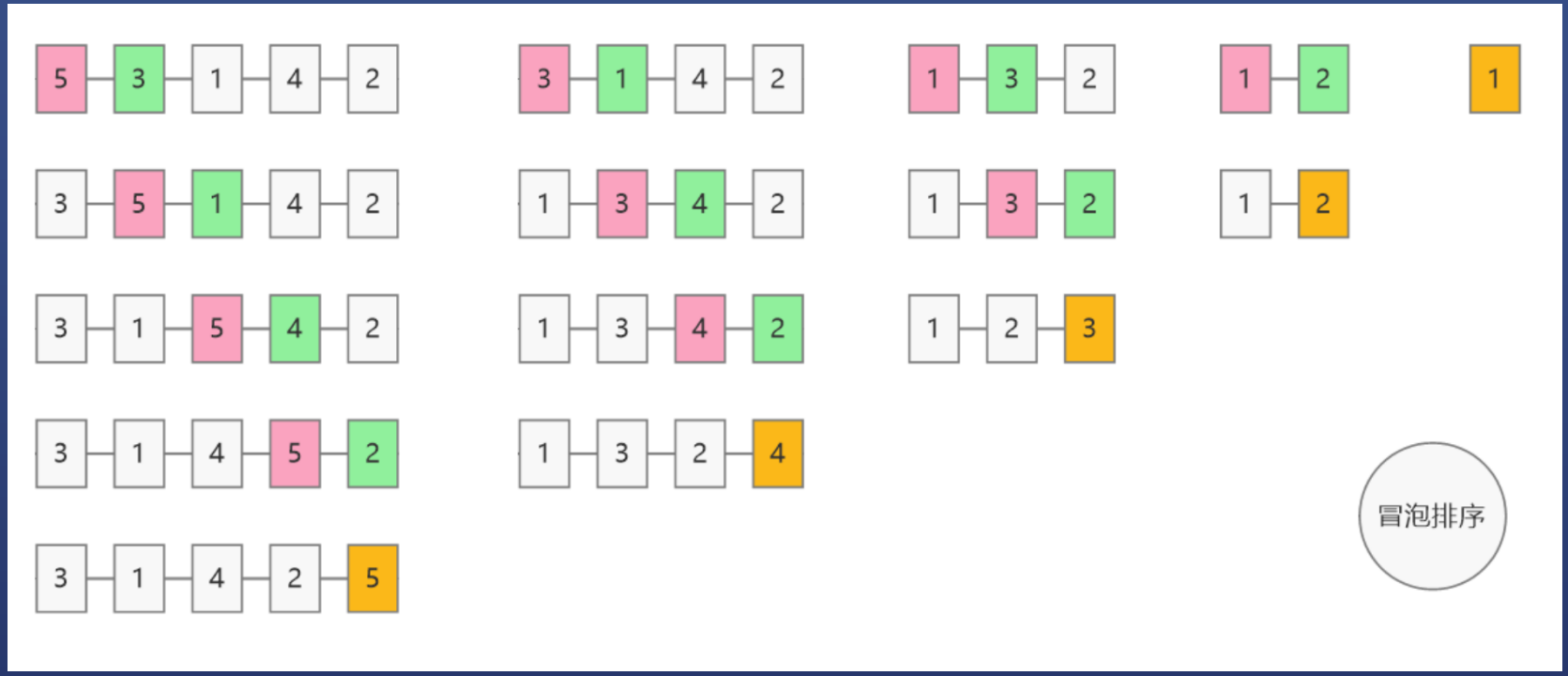
2 计数排序


9
4
7
2
5
4
8
对象 =
{
9 : 1
4 : 2
7 : 1
2 : 1
5 : 1
8 : 1
}
计数排序
for i from 1 to 9
1
2
3
4
5
6
7
8
9
return
2
4
4
5
7
8
9

思维特点

- 空间换时间
  - ✓ 用一个 hash table 计数，两次遍历解决问题
  - ✓ 第一次遍历将数组复制到 hash table
  - ✓ 第二次遍历 i 从 1 到 max（max 为最大值）
  - ✓ 时间复杂度降为 O(n+max)
- 在某些情况下效率很低
  - ✓ 元素个数少，值跨度大时
  - ✓ 比如 [1,9999999,5555] 排序时就很慢

3 冒泡排序



每次通过相邻元素两两对比来获取最大/小值

冒泡排序

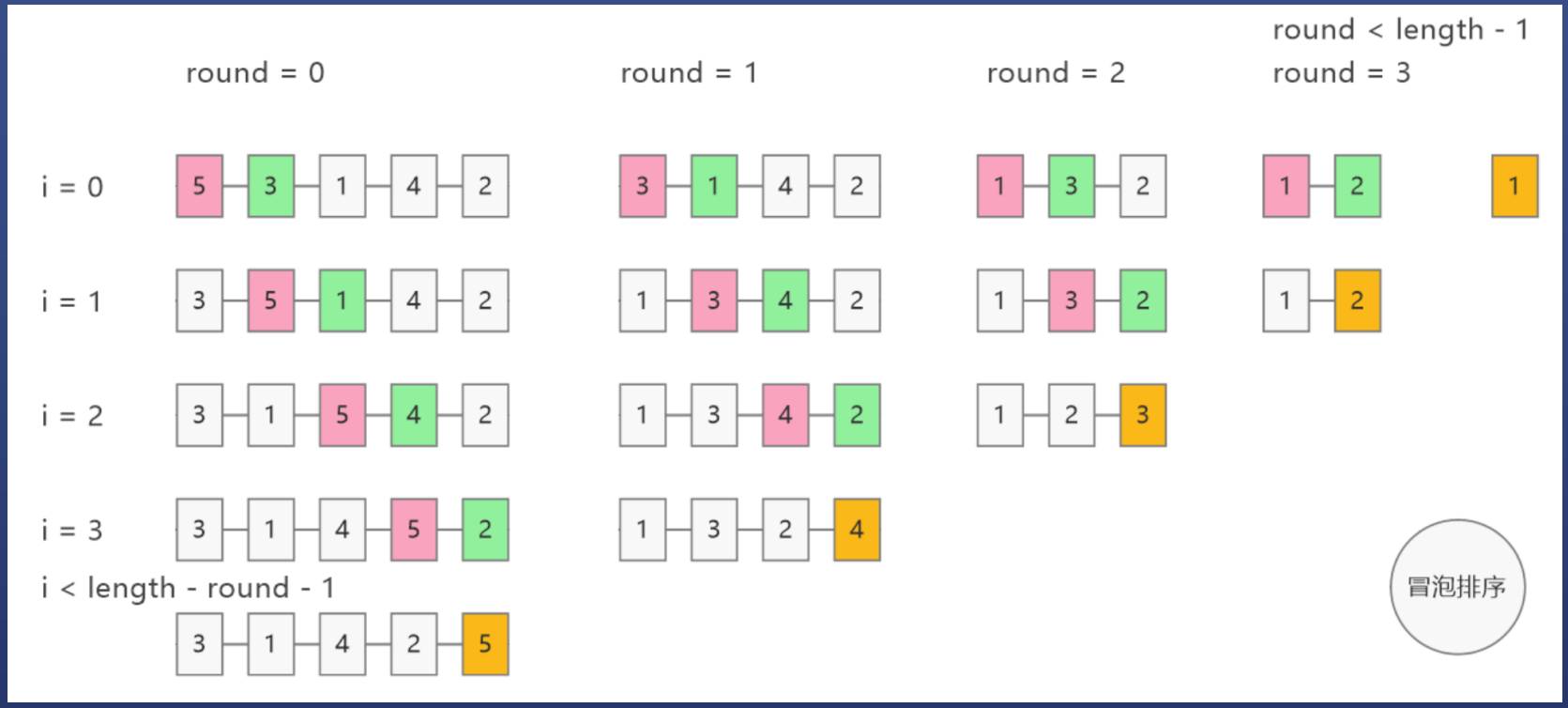
- 思路
  - 类似于选择排序
  - 但是是通过相邻两两对比来选出最大或最小的一个
  - 而选择排序是将每个元素与第一个元素对比
- 特点
  - 时间复杂度 n + (n-1) + (n-2) + ... + 1 约等于 n^2
  - 是一个低效的算法，不过面试可能会问到

冒泡排序代码

```
// 假设 arr 为 [5,3,1,4,2]
bubbleSort = arr => {
  const {length} = arr
  for(let r=0; r<length-1; r++){
    for(let i=0; i<length-r-1; i++){
      if(arr[i]>arr[i+1]){
        [arr[i], arr[i+1]] = [arr[i+1], arr[i]]
      }
    }
  }
  return arr
}
```

- 凭啥你知道这些细节?

辅助技巧

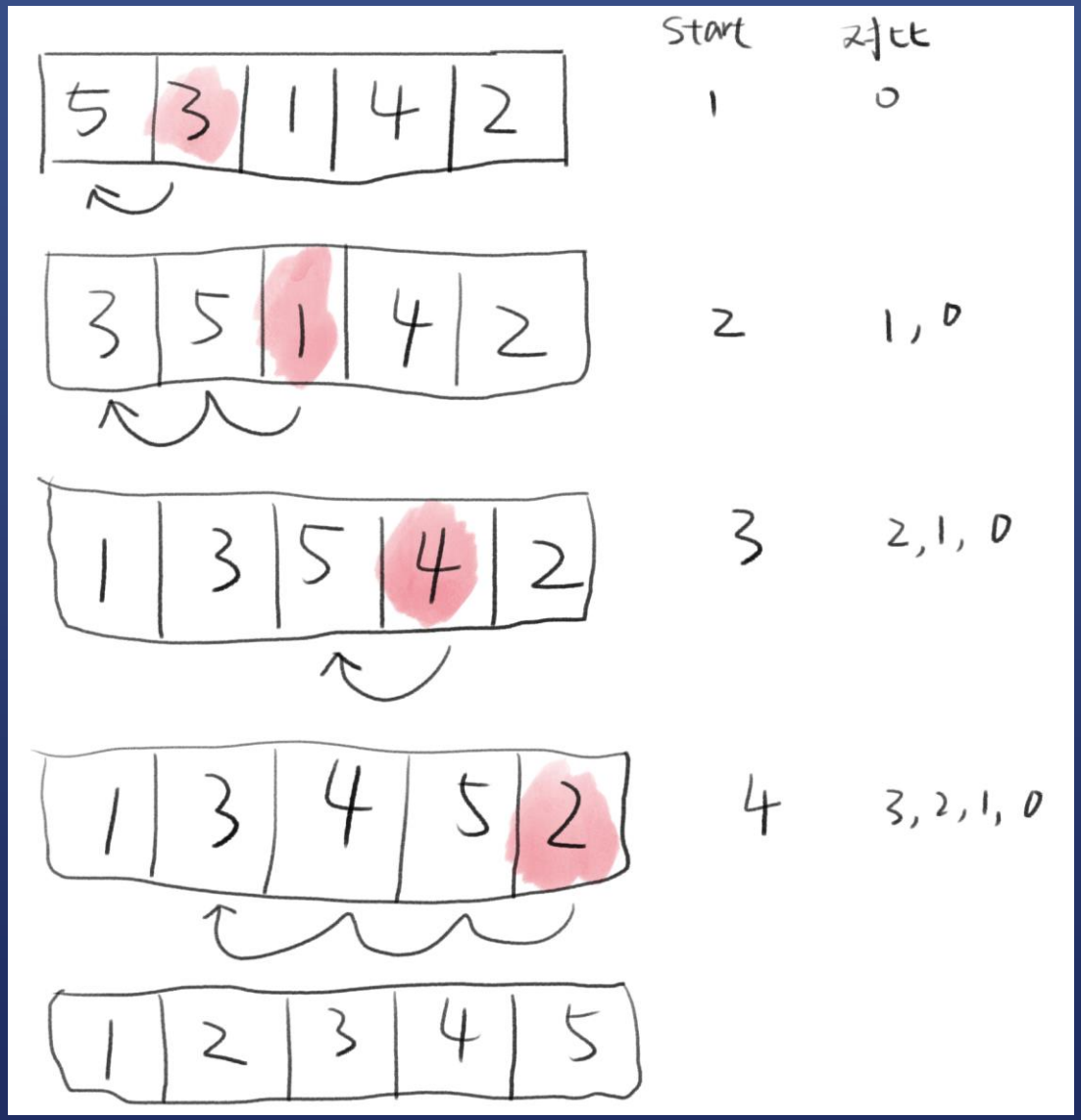


先画图找规律，再写代码

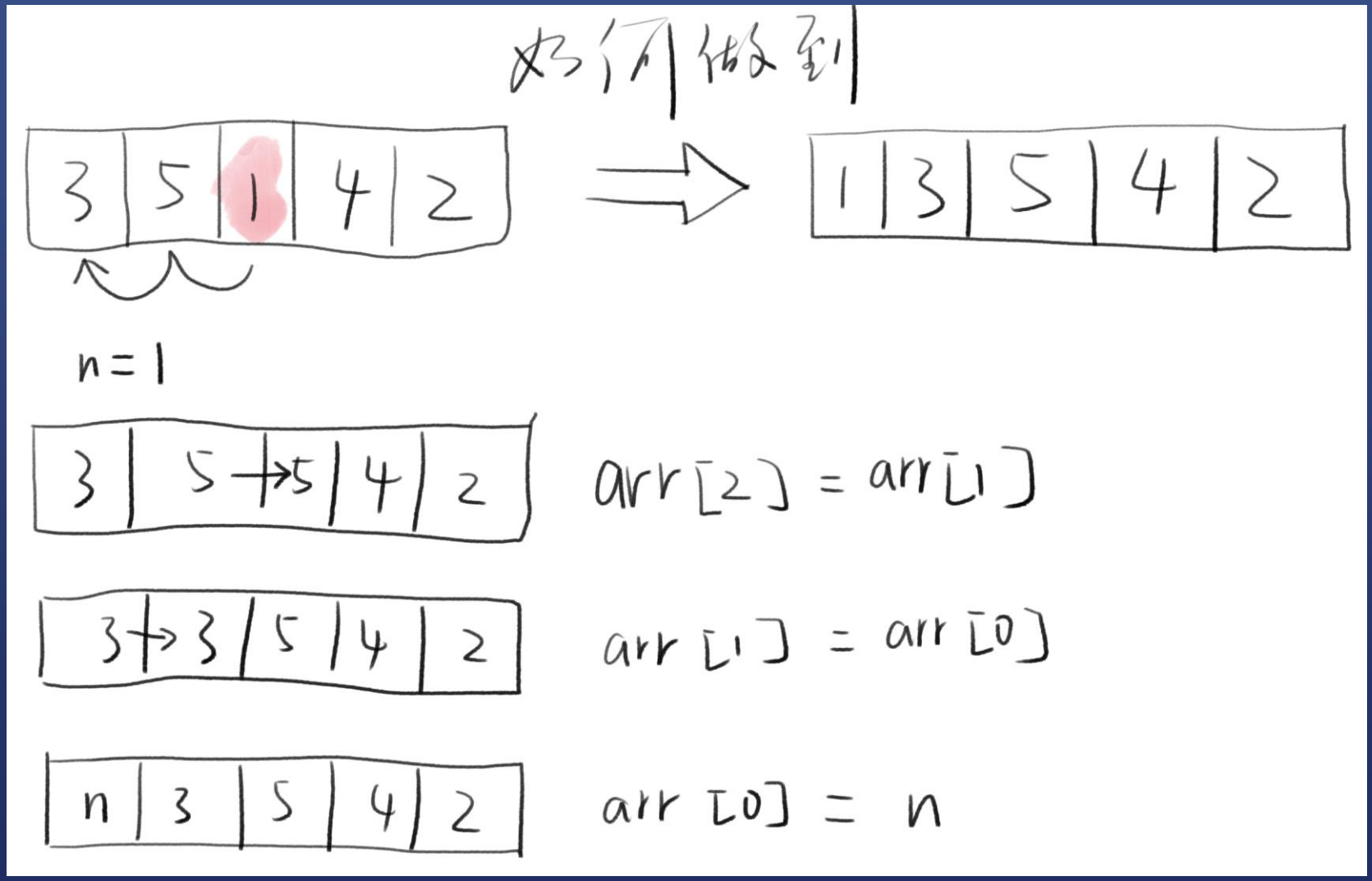
4 插入排序

每次起牌，将新牌插入到第一个比它小的牌的后面
虽然叫做插入排序，但是最好不要用 splice 方法，因为 splice 会导致整体挪动

起牌、插牌



细节


如何做到
3 5 1 4 2
1 3 5 4 2
n=1
3 5→5 4 2
arr[2] = arr[1]
3→3 5 4 2
arr[1] = arr[0]
n 3 5 4 2
arr[0] = n

插入排序代码

```
insertSort = arr => {
  if(arr.length<=1)return arr
  for(let s=1; s<arr.length; s++){
    const n = arr[s] // 记住n
    let i // i要在for循环之后使用
    for(i=s-1; i>=0; i--){
      //比 n 大的数字往后挪了一位
      if(arr[i]>n) arr[i+1] = arr[i]
      // 遇到比 n 小的就中断循环
      else if(arr[i]<=n) break
    }
    arr[i+1] = n
  }
  return arr
}
```

插入排序复杂度

- O(n^2)
  - 最差情况 1 + 2 + ... + n-1 + n
- 如何优化
  - 插入的时候不要从右往左一个一个对比
  - 而是使用二分查找法，这样可以快速找到正确的位置
  - 但插入的时候会引起整体挪动，所以复杂度并没有降低

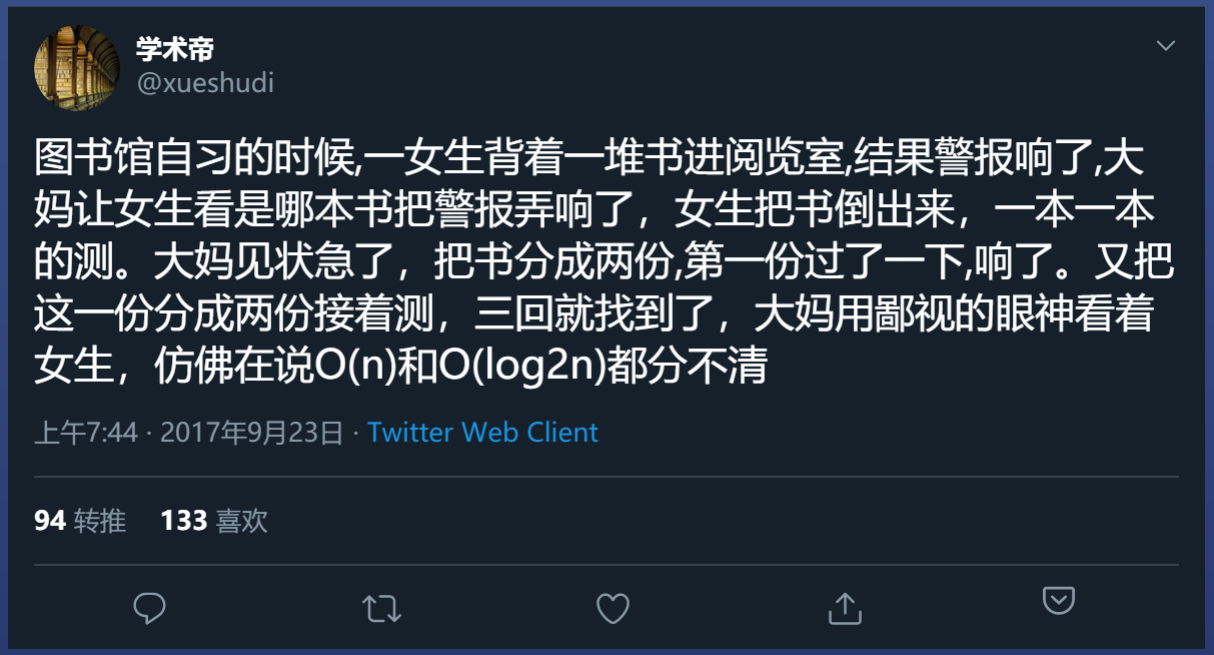
5 二分查找法

- 需求
  - 一个已经排好的数组 arr
  - 一个数字 n
  - 请告诉我 arr 里有没有跟 n 一样大的数字
- 思路
  - 将 arr 中间的元素与 n 对比，如果相等就找到了
  - 如果比 n 大，就用 arr[0..mid) 进行第一步
  - 如果比 n 小，就用 arr[mid..len) 进行第一步
  - 直到数组长度缩为0，还找不到就是找不到了

递归写法

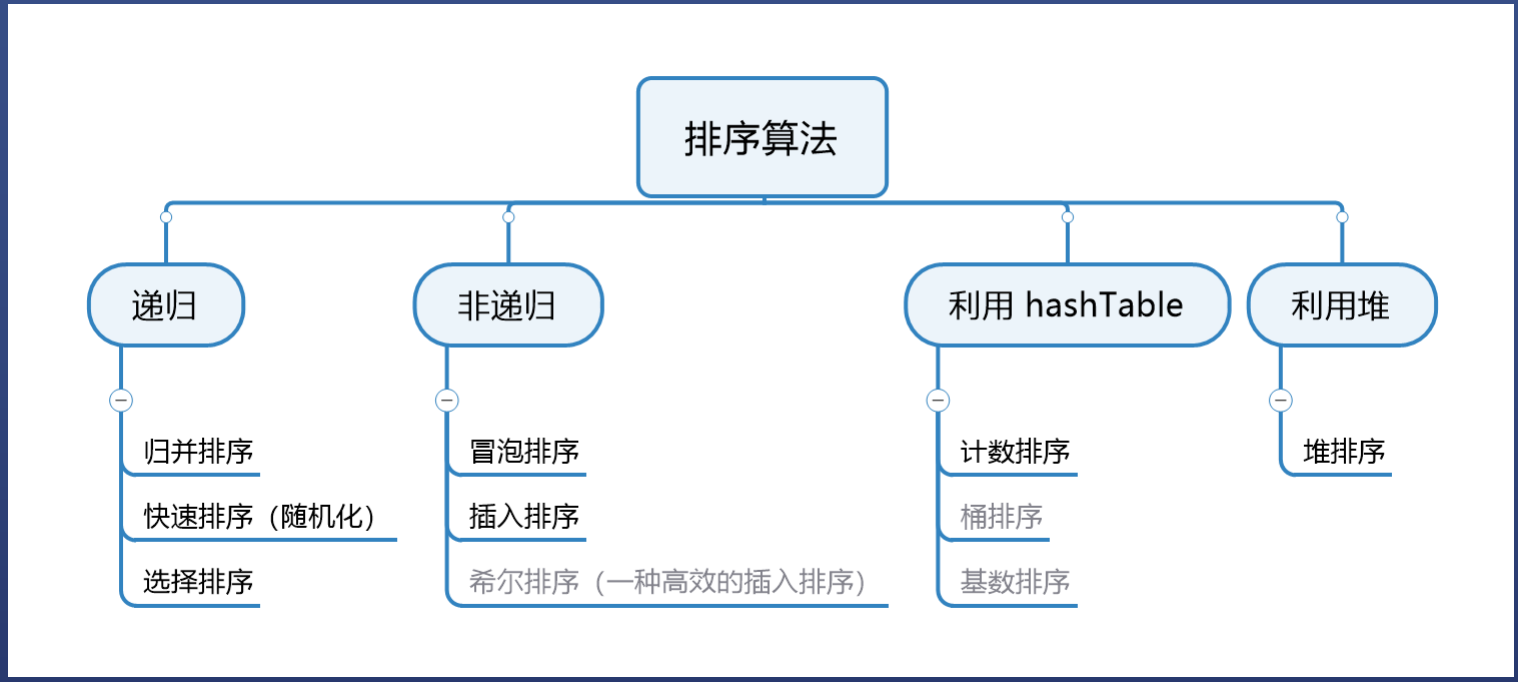
```
bSearch = (arr, n, start, end)=>{
 if(end === start) return -1
 let mid = parseInt((start + end) / 2) //某些语言会溢出
 console.log(mid)
 return n === arr[mid] ? mid :
        n > arr[mid] ? bSearch(arr,n,mid+1,end) :
        n < arr[mid] ? bSearch(arr,n,start,mid) :
        undefined
}

bSearch([2,2,2,4,5,6,7],1,0,7)
log: 3
log: 1
log: 0
结果:  -1
7 个数字只需要查找 3 次
```

二分查找的应用


学术帝
@xueshudi
图书馆自习的时候,一女生背着一堆书进阅览室,结果警报响了,大妈让女生看是哪本书把警报弄响了，女生把书倒出来，一本一本的测。大妈见状急了，把书分成两份,第一份过了一下,响了。又把这一份分成两份接着测，三回就找到了，大妈用鄙视的眼神看着女生，仿佛在说O(n)和O(log2n)都分不清
上午7:44 · 2017年9月23日 · Twitter Web Client
94 转推 133 喜欢


cattery @huangfujt · 2017年9月23日
回复 @xueshudi
万一好几本呢

总结


排序算法
递归
非递归
利用 hashTable
利用堆
归并排序
快速排序（随机化）
选择排序
冒泡排序
插入排序
希尔排序（一种高效的插入排序）
计数排序
桶排序
基数排序
堆排序

总结

算法	最坏时间复杂度	平均
归并排序	O(n log2 n)	同左
快速排序	O(n^2)	O(n log2 n)
选择排序	O(n^2)	同左
计数排序	O(n+max)	同左
冒泡排序	O(n^2)	同左
插入排序	O(n^2)	O(n^2)

参考文章：十大经典排序算法