



# 排序问题

---



# 排序（Sorting）

---

- 普遍、常用

- 目标

将输入序列转换为有序序列

- 定义

将杂乱无章的数据元素（输入序列），通过一定的方法按关键字顺序排列的过程（有序）

- 作用

- 非常基础的算法，很多问题的求解以排序为基础；
- 排序算法的一些思想可用来解决一些特殊问题。



## ■ Samlpe

- 在一个叫AC-Magic的游戏里，有个名为zhb的非常有爱的NPC。为了帮助对抗可怕的WA大魔王， zhb决定发给每个勇士一把附魔武器，这把武器的攻击力是它的本身属性值乘以使用者的等级决定的。现在zhb有n把不同的武器要发给n个等级不同的勇士，应该如何分配才能获得最大战斗力抵抗大魔王呢？

## ■ Answer

- 将武器属性值进行升序排序，再将勇士等级升序排序，这样每个勇士使用对应下标的武器就能达到总战斗力最大。



## ■ 排序算法

- 冒泡排序
- 选择排序
- 插入排序
- 归并排序
- 计数排序
- 快速排序
- 堆排序
- 希尔排序
- 猴子排序
- ...

区别：复杂度不一样  
时间复杂度  
空间复杂度

Q 网页 贴吧 知道 文库 图片 资讯 视频 地图 采购 更多

百度为您找到相关结果约100,000,000个

搜索工具

## 已到C 库!AI 帮助人类打破十年算法瓶颈,谷歌发现更快排序...



2023年6月13日 虽然人们在这一领域投入了大量时间研究出各种高效的排序算法,但是自从十年前以来,C库中的排序代码就没有变化过,这意味着现有的排序方法已经无法再有更大的进展了。然而,谷歌Dee...

老武黑科技

## 已到C库!谷歌发现更快排序算法,AI 帮助人类打破十年算法瓶...



2023年6月15日 近日,谷歌DeepMindAI小组开发出一种名为AlphaDev的强化学习工具,能够在不需要任何人类代码示例的情况下,开发出极限优化的算法。将AlphaDev应用于排序算法中,其发现的新算法已被...

吴老三科技小筑

## 十多年来,C++排序库首次更改,人工智能改进计算机编程语言

2023年6月8日 英国深度思维公司的人工智能体“阿尔法开发”(AlphaDev),已被证明能发现并改进C++(一种常用的计算机编程语言)库里广泛使用的计算机排序算法。《自然》7日...

极目新闻

## 谷歌借AI打破十年排序算法封印,每天被执行数万亿次,网友却...

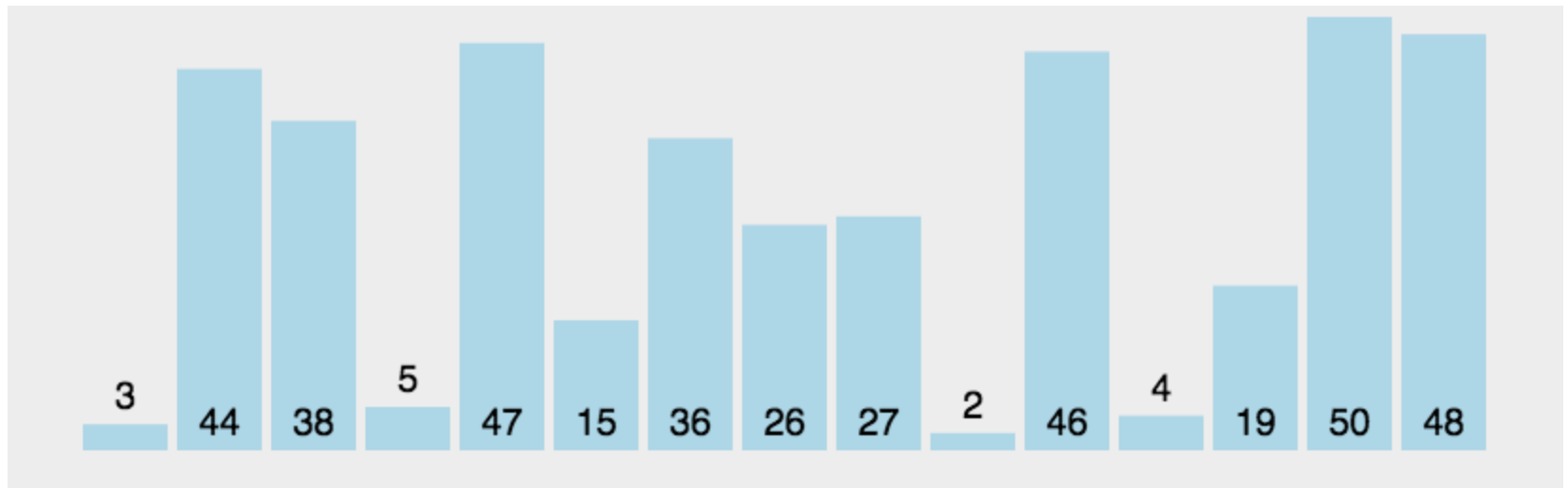
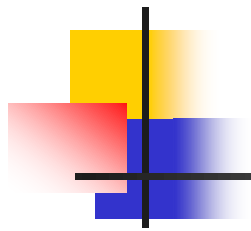


2023年6月18日 最终,AlphaDev 发现了新的排序算法,这些算法可以让 LLVM libc++ 排序库得到改进:对于较短的序列,排序库的速度提高了 70%;对于超过 250,000 个元素的序列,速度提高了约 1.7%。具体...



# 冒泡排序（Bubble Sort）

- 基本原理（以升序排序为例）
  - 比较相邻的元素，如果第一个比第二个大，就交换他们
  - 对每一对相邻元素做同样的处理，从开始第一对到结尾的最后一对。结果最后的元素就是最大的数。
  - 针对所有的元素重复以上的步骤，除了最后一个。
  - 持续每次对越来越少的元素重复上面的步骤，直到没有任何一对数字需要比较。
- 时间复杂度： **$O(n^2)$**
- 空间复杂度： **$O(1)$**
- 稳定的排序算法
  - 相同数的相对位置不会改变



<https://blog.csdn.net/hcz666/article/details/117810787>



# 选择排序（ Selection sort ）

- 基本原理

- 每一趟从待排序的数据元素中选择最小（或最大）的一个元素作为首元素，直到所有元素排完为止

- 简单直观

- 时间复杂度：  $O(n^2)$

- 空间复杂度：  $O(1)$

- 不稳定

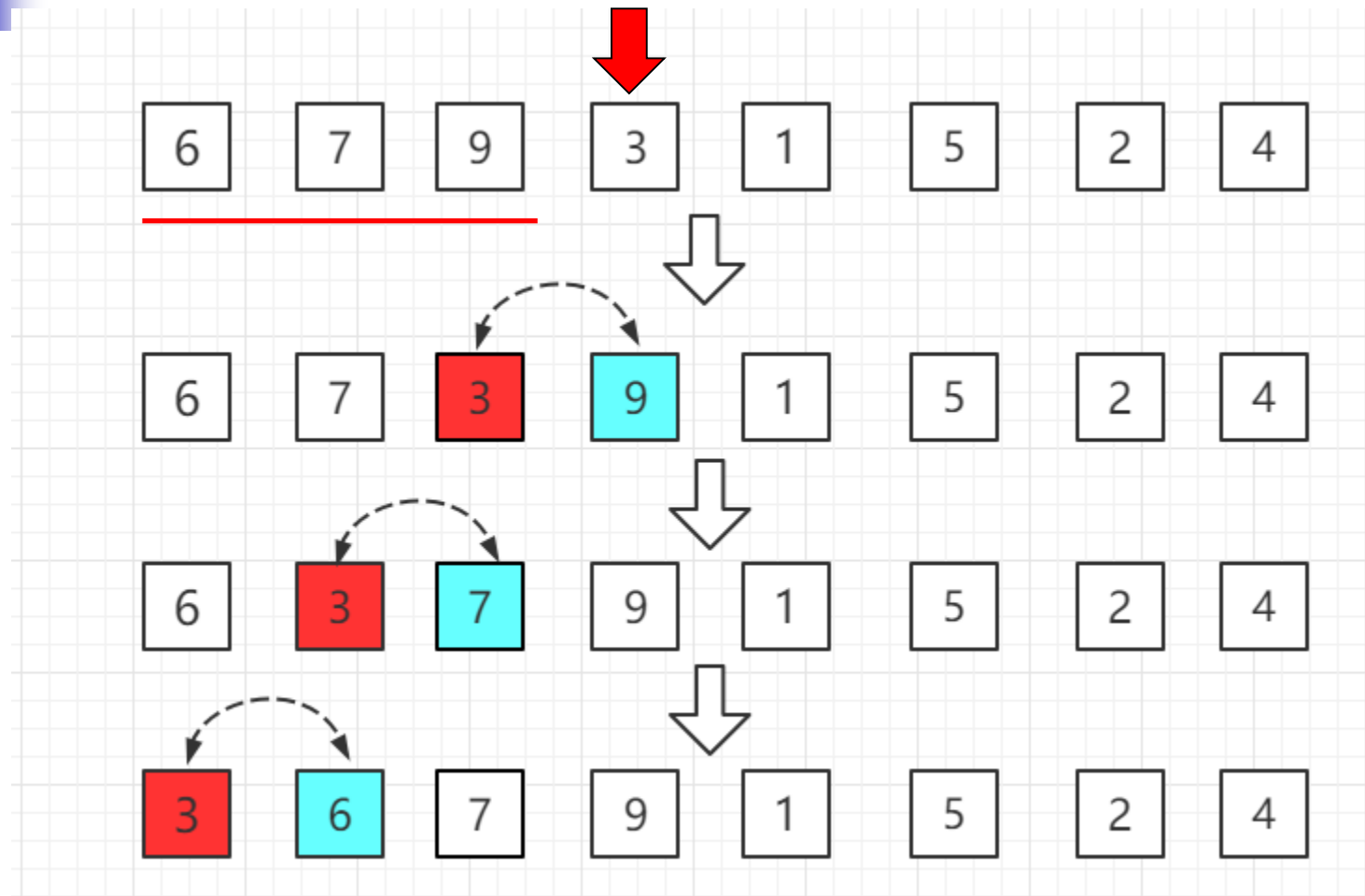
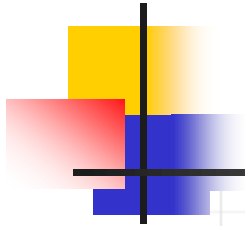
当出现相同元素的时候，有可能会改变相同元素的顺序。





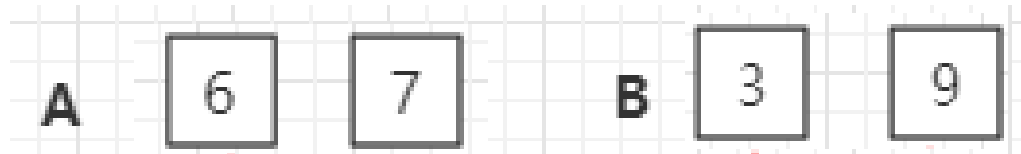
# 插入排序（Insertion Sort）

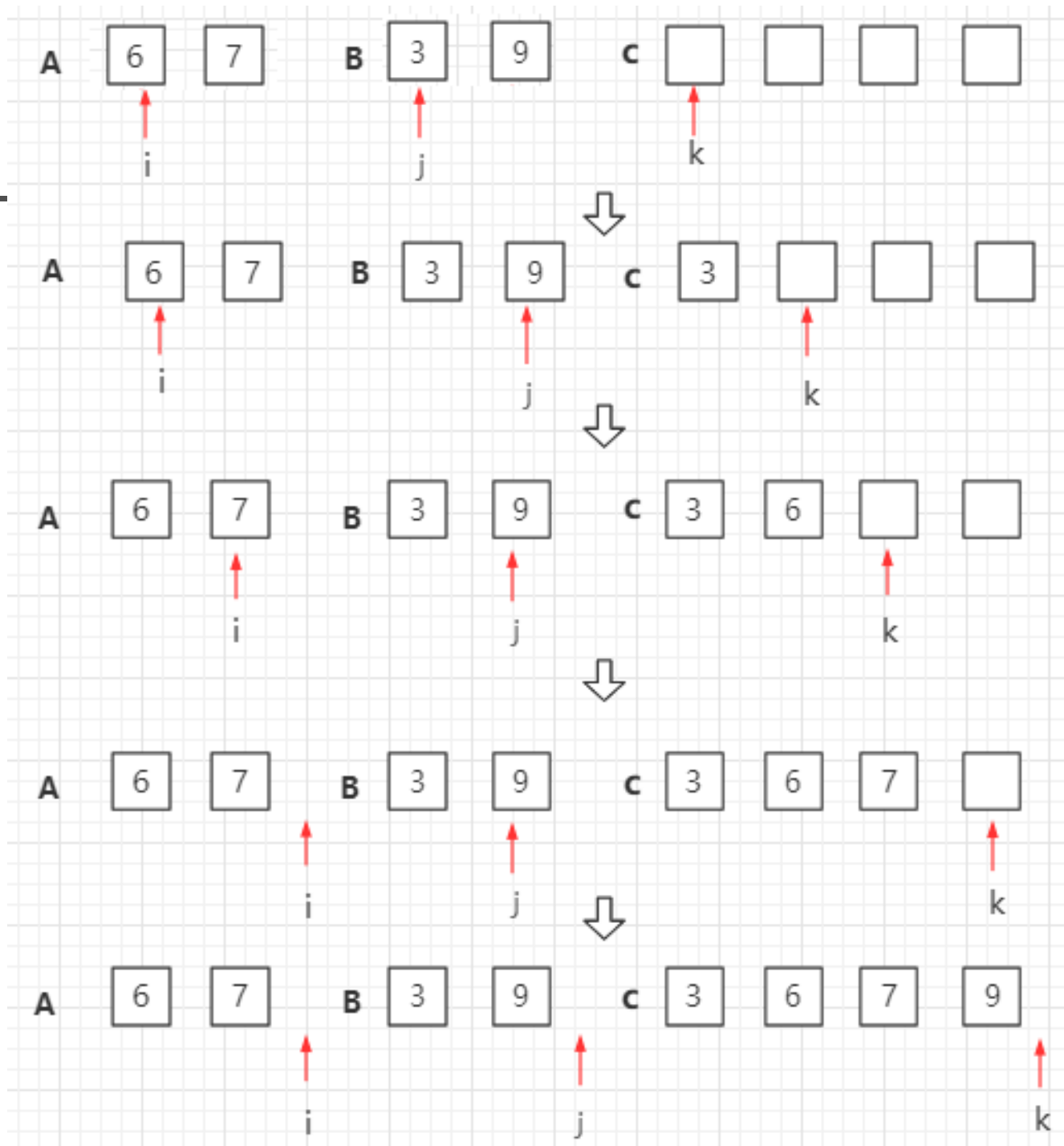
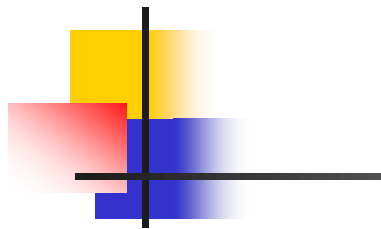
- 基本原理
  - 每次将一个元素插入到已经排好序的有序表中，得到一个新的、记录数增**1**的有序表。
- 实现过程
  - 双循环， .....
- 时间复杂度：  **$O(n^2)$**
- 空间复杂度：  **$O(1)$**
- 简单，稳定，适用于少量元素的排序



# 归并排序 (Merge sort)

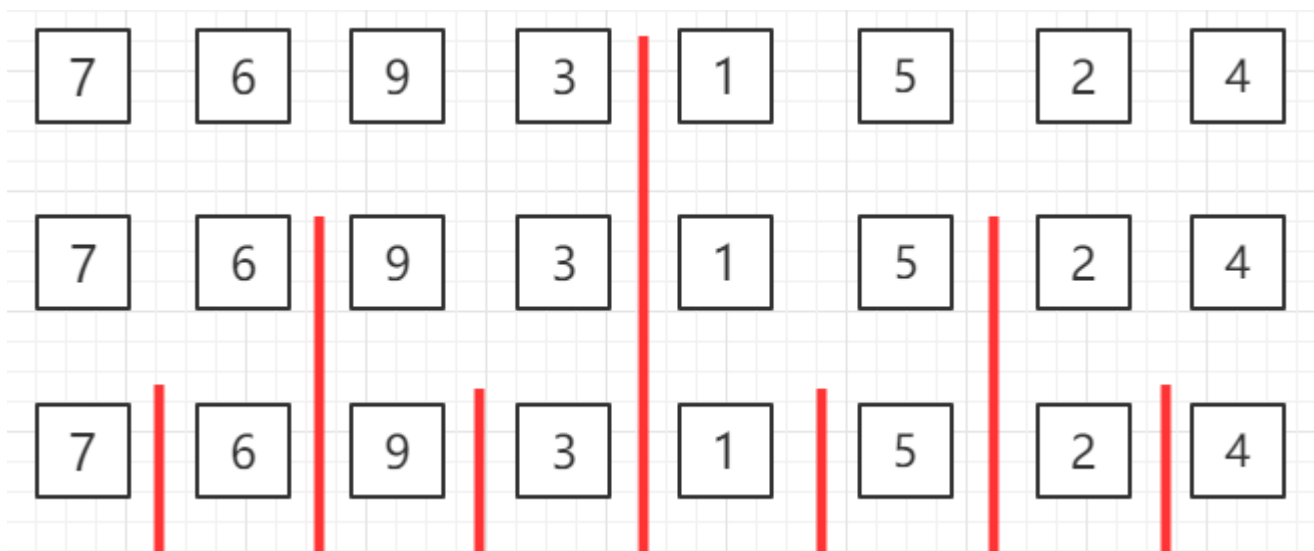
- 建立在归并操作上的一种有效、稳定的排序算法
- 采用分而治之的策略，将输入序列分解成子序列，先使每个子序列有序，再将已有序的子序列合并（归并操作），得到完全有序的序列。
- 归并操作





## ■ 实现过程

- 递归分组，递归调用



- 时间复杂度:  **$O(n \log n)$**
- 空间复杂度:  **$O(n)$**

# 计数排序 (Counting sort)

- 以空间换时间典型代表;
- 其核心是**将输入的数据值转化为键，存储在额外开辟的数组空间中**，以达到排序的效果;

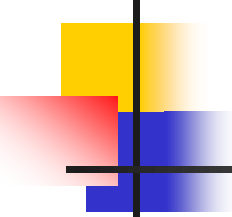


- 
- 给定一组取值范围为**0到9**的无序序列，对其进行排序：**1、7、4、9、0、5、2、4、7、3、4**。

0	0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9
1	1	1	1	3	1	0	2	0	1
0	1	2	3	4	5	6	7	8	9

遍历计数数组，输出计数数组下标值，元素的值是多少，就输出几次，输出的结果就是排序的结果：

**0、1、2、3、4、4、4、5、7、7、9**



## ■ 计数排序的特点

- 不需要比较数值的大小
- 时间复杂度:  **$O(n+m)$**
- 空间复杂度:  **$O(n+m)$**
- 不稳定 (可以做到稳定)

## ■ 计数排序的局限性

- 当待排序序列的值不是整数时, 不适合;
- 当待排序序列的最大值和最小值差值特别大时, 不适合

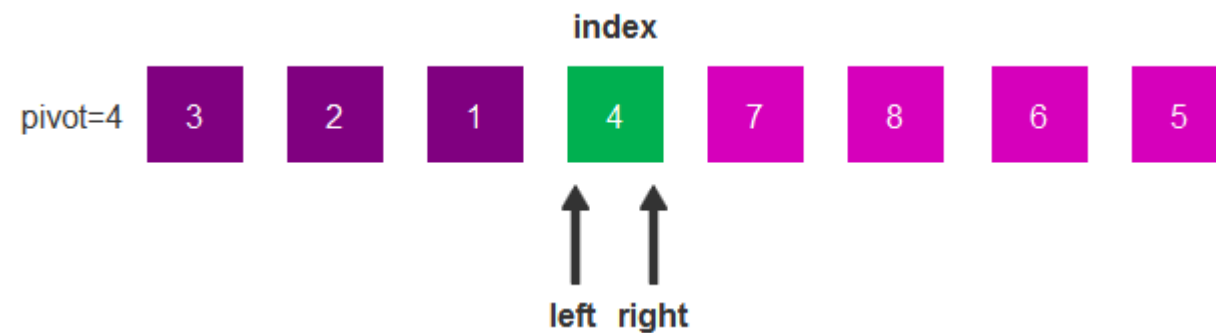
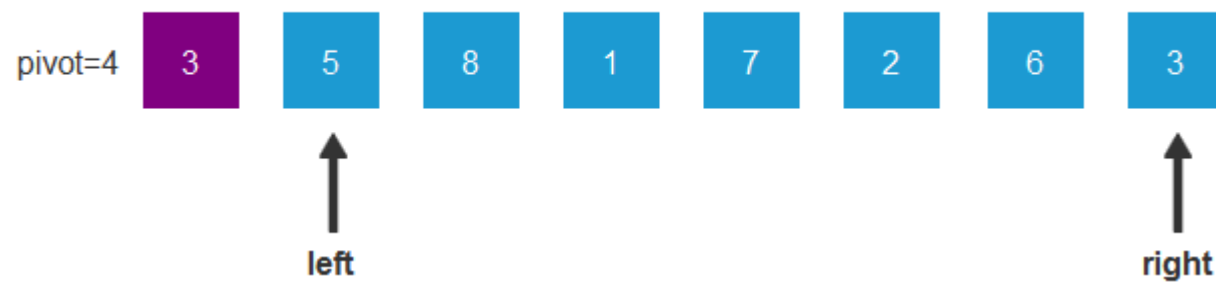
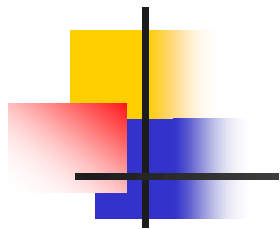


# 快速排序 (Quick Sort)

## ■ 基本思想

- 在每一轮挑选一个**基准元素**，并让其他比它大的元素移动到数列一边，比它小的元素移动到数列的另一边，从而把数列拆解成了两个部分；
  - 然后再对这两部分数据分别进行快速排序；
  - 整个排序过程**递归进行**，最后整个数据变成有序序列。
- 从冒泡排序算法演变而来的，实际上是在冒泡排序基础上的递归分治法。







## ■ 快速排序的性能

- 在各种数据中，综合性能是最好的；
- 最坏情况下时间复杂度： $O(n^2)$
- 平均时间复杂度： $O(n\log n)$   
但复杂度的常数系数比较小，所以敢称“快速”；
- 原地排序算法，只需要一个元素的辅助空间，空间复杂度 $O(1)$ ，更快的原因之一；  
但快速排序递归算法，需要一个栈空间来实现递归，空间复杂度实际为 $O(\log n)$ ；
- 不稳定的排序算法。



# C的qsort()函数

```
1 void qsort(  
2     void *base,  
3     size_t nmemb,  
4     size_t size,  
5     int (*compar)(const void *, const void *)  
6 );
```

头文件:<stdlib.h>

函数功能: qsort()函数的功能是对 **数组** 进行排序, 数组有nmemb个元素, 每个元素大小为size。

参数base : base指向数组的起始地址, 通常该位置传入的是一个数组名。

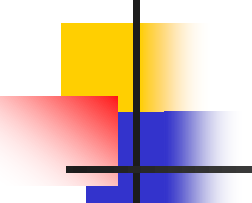
参数nmemb :nmemb表示该数组的元素个数

参数size :size表示该数组中每个元素的大小 (字节数)

参数 (\*compar)(const void \*, const void \*) :此为指向比较函数的函数指针, 决定了排序的顺序。

函数返回值: 无

注意: 如果两个元素的值是相同的, 那么它们的前后顺序是不确定的。也就是说qsort()是一个不稳定的排序算法。

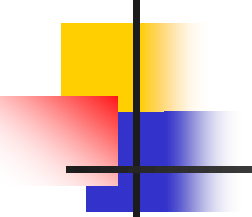


```
2
3  int compare(const void *p,const void *q)
4  {
5      if(*(int *)p<*(int *)q)
6          return -1;
7      else if(*(int *)p==*(int *)q)
8          return 0;
9      else return 1;
10 }
11
12 #define N 10 //数组长度
13 int data[N];
14 .....
15 qsort(data,N,sizeof(data[0]),compare);
16
```



# C++的sort()函数

- **#include<algorithm>**
- **sort(首地址,首地址+排序元素个数,比较函数)**
- 比较函数可省略，默认升序，元素必须可比大小
- 例  
n个整数升序排（字符，浮点数等类似）
  1. **a[1]~a[n]: sort(a+1,a+n+1);**
  2. **a[0]~a[n-1]: sort(a, a+n);**
  3. **a[l]~a[r]: sort(a+l,a+r+1);**
- 将n个整数**降序**排，  
**a[1]~a[n]: sort(a+1,a+n+1,cmp);**

- 
- **sort()**函数是类似于快速排序的方法，时间复杂度为 $n * \log_2(n)$ ；
  - 除了对普通的快速排序进行优化，还结合了插入排序和堆排序。根据不同的数量级别以及不同情况，能自动选用合适的排序方法。当数据量较大时采用快速排序，分段递归。一旦分段后的数据量小于某个阈值，为避免递归调用带来过大的开销，便会改用插入排序。而如果递归层次过深，有出现最坏情况的倾向，还会改用堆排序。



# 参考资料

---

- 十大经典排序算法详解  
[https://blog.csdn.net/qq\\_35344198/article/details/107206269](https://blog.csdn.net/qq_35344198/article/details/107206269)  
.....
- <https://blog.csdn.net/hcz666/article/details/117810787>
- <https://www.runoob.com/data-structures/merge-sort.html>
- .....