# 归并排序、统计排序

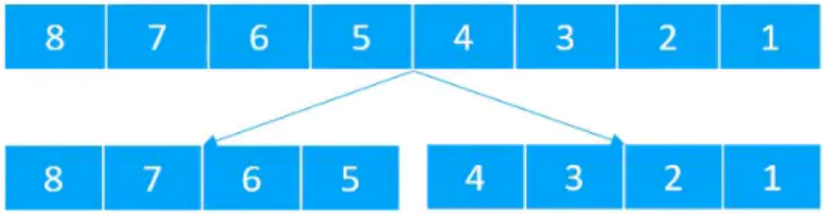
## 1. 归并排序概述

归并排序，其排序的实现思想是先将所有的记录完全分开，然后两两合并，在合并的过程中将其排好序，最终能够得到一个完整的有序表。

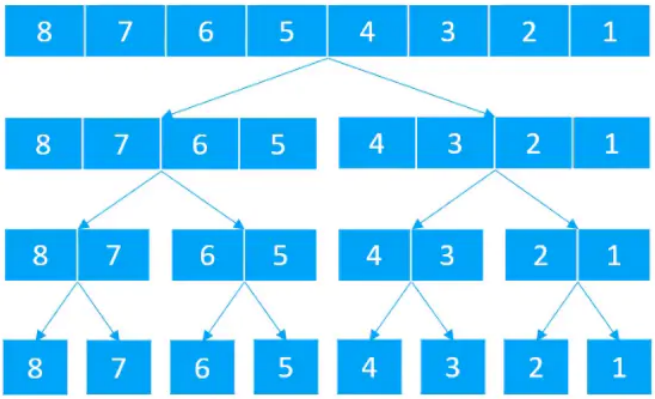
### 1.1 归并排序的执行流程：

- 归并排序执行流程

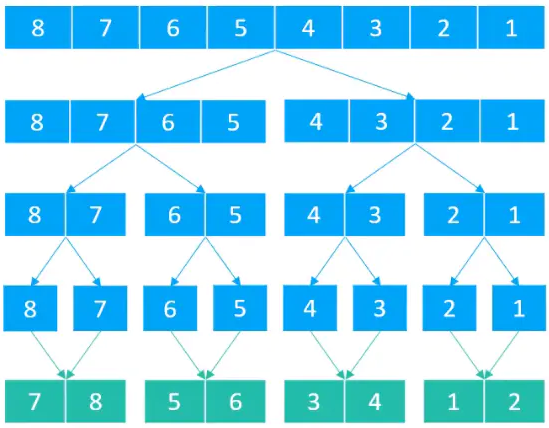
1. 不断地将当前序列平均分割成2个子序列；例如下面的序列，被分割成2个子序列



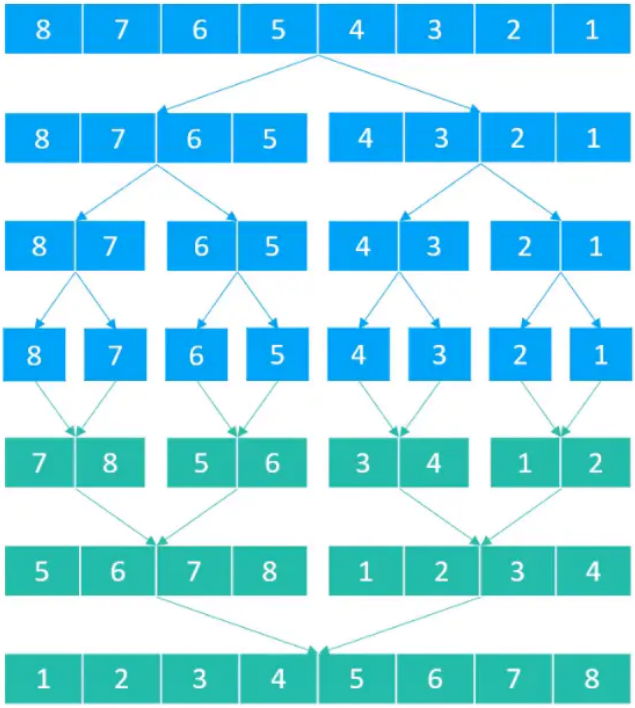

然后继续将这些子序列分割成子序列，直到不能再分割位置（序列中只剩一个元素）




2. 接下来，在不断的将两个子序列合并成一个有序序列；也就是说，刚刚是拆分，现在是合并




由于是不断的合并成一个有序序列，所以最终只剩下一个有序序列：




### 1.2 分割的过程

由于在分割时，都是将一个有序序列分割为2个两个子序列，并且该操作是重复执行的，所以肯定会使用到递归。由于是从中间进行分割，所以需要计算出中间的位置。所以实现流程是

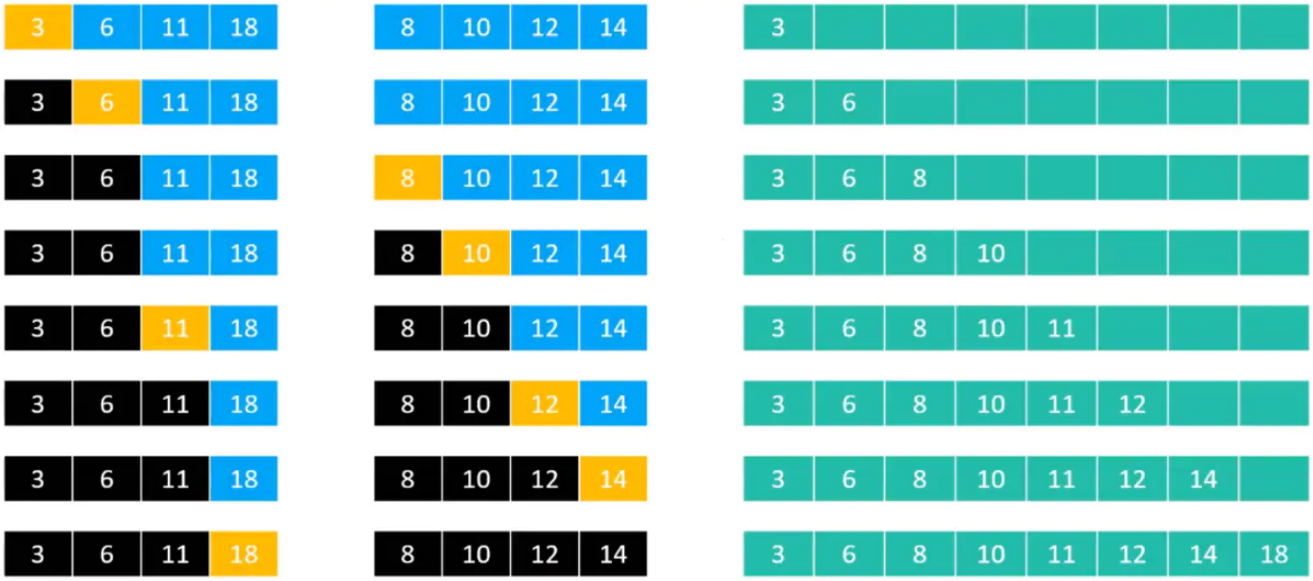
a. 计算拆分的中间位置

b. 分别继续对左边序列和右边序列进行拆分

### 1.3 合并的过程

在merge时，肯定会得到2个有序的数组。所以要做的事情就是将这两个有序的数组合并为一个有序的数组。现有两个有序的数组，然后根据这两个有序的数组合并为一个。

现在要将这两个有序序列合并为一个更大的有序序列，所以可以先比较两个序列的头元素，谁的值比较小，就先放入大序列中，利用两个索引，分别记录两个序列中待比较值的下标。然后将值小的序列下标右移一个单位，继续比较。最终将两个有序数组合并为一个的流程图如下：




合并后，会出现merge左边先结束，merge右边先结束的情况。

## 2. 计数排序

前面介绍的冒泡，选择，插入，归并，快速，希尔，堆排序，都是基于比较的排序，这些基于比较的排序，有以下几个特点：平均时间复杂度最低的是O(nlogn)

而计数排序，不是基于比较的排序。其中不基于比较的排序还有桶排序，基数排序等

它们是典型的用空间换时间，在某些时候，平均时间复杂度可以比O(nlogn)更低，也就是说，在某些时候，这种利用空间换时间的排序算法，性能比前面基于比较的排序算法更快。

### 2.1 计数排序

计数排序是在1954年由Harold H.Seward提出，适合对一定范围内的整数进行排序。

计数排序核心思想

统计每个整数在序列中出现的次数，进而推导出每个整数在有序序列中的索引。

我们以数组[1,4,1,2,5,2,4,1,8]为例进行说明。

第一步：建立一个初始化为 0 ，长度为 9 (原始数组中的最大值 8 加 1) 的数组count[]。

第二步：遍历数组 [1,4,1,2,5,2,4,1,8] ，访问第一个元素 1 ，然后将数组 标为 1 的元素加 1，表示当前 1 出现了一次，即 count[1] = 1 ;

依次遍历，对count进行统计。

### 2.2 计数排序的改进

上述算法的问题如下：

a. 无法对负整数进行排序

b. 极其浪费内存空间

c. 是一个不稳定排序

### 2.2.1 优化1

只要不再以输入数列的 [**最大值+1**] ，作为统计数组的长度，而是以数列 [**最大值−最小值+1**] 作为统计数组的长度即可。数列的最小值作为一个偏移量，用于计算整数在统计数组中的下标。

比如，假设下面的数组的数列：

95，94，91，98，99，90，99，93，91，92

统计出数组的长度为99−90+1=10，偏移量等于数列的最小值90。对于第1个整数95，对应的统计数组下标是95−90=5。

### 2.2.2 优化2

假设数据如下：

| 姓名 | 成绩 |
| --- | --- |
| A | 90 |
| B | 99 |
| C | 95 |
| D | 94 |
| E | 95 |

上述数据按照计数排序得到的结果如下：

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

如何判断5中的2个人谁是C谁是D那。那么需要进行变形。

这是如何变形的呢？其实就是从统计数组的第2个元素开始，每一个元素都加上前面所有元素之和。

这样相加的目的，是让统计数组存储的元素值，等于相应整数的最终排序位置的序号。例如，下标是9的元素值为5，代表原始数列的整数9，最终的排序在第5位。

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

首先，遍历成绩表最后一行E的成绩，E是95，那么5下标元素是4，表示E在最后的排名在第4位，同时将原来的值减1，表示下次再遇到95的成绩时，最终排名是第3。

## 3. 桶排序

其实桶排序重要的是它的思想，而不是具体实现，桶排序从字面的意思上看：

1、若干个桶，说明此类排序将数据放入若干个桶中。

2、每个桶有容量，桶是有一定容积的容器，所以每个桶中可能有多个元素。

3、从整体来看，整个排序更希望桶能够更匀称，即既不溢出(太多)又不太少。

假设有一个非整数数列，如下：

4.5, 0.84, 3.25, 2.18, 0.5

桶排序的第1步，就是创建这些桶，并确定每一个桶的区间范围。



具体需要建立多少个桶，如何确定桶的区间范围，有很多种不同的方式。我们这里创建的桶数量等于原始数列的元素数量，除最后一个桶只包含数列最大值外，前面各个桶的区间按照比例来确定。

区间跨度=（最大值–最小值）/（桶的数量–1）

第2步，遍历原始数列，把元素对号入座放入各个桶中。

第3步，对每个桶内部的元素分别进行排序。

第4步，遍历所有的桶，输出所有元素。

0.5，0.84，2.18，3.25，4.5

## 4. 基数排序

与基于比较的排序算法（归并排序、堆排序、快速排序、冒泡排序、插入排序等等）相比，基于比较的排序算法的时间复杂度最好也就是O（nlogn），而且不能比 O（nlogn）更小了。

计数排序（Counting Sort）的时间复杂度为O（n）量级，更准确的说，计数排序的时间复杂度 为O（n + k），其中k表示待排序元素的取值范围（最大与最小元素之差加 1 ）。那么问题来了，当这个元素的的范围在 1 到 n^2 怎么办呢?

此时就不能用计数排序了奥，因为这种情况下，计数排序的时间复杂度达到了O(n^2)量级。

比如对数组 [170, 45, 75, 90, 802, 24, 2, 66] 这个而言，数组总共包含 8 个元素，而数组中的最大值和最小值之差为 802 − 2 = 800 ，这种情况下，计数排序就 ”失灵了“ 。

那么有没有那种排序算法可以在线性时间对这个数组进行排序呢?

基数排序（Radix Sorting） 。基数排序的总体思想就是从待排序数组当 中，元素的最低有效位到最高有效位 逐位 进行比较排序；此外，基数排序使用计数排序作为一个排序的 子过程。

以数组[170, 45, 75, 90, 802, 24, 2, 66]为例：