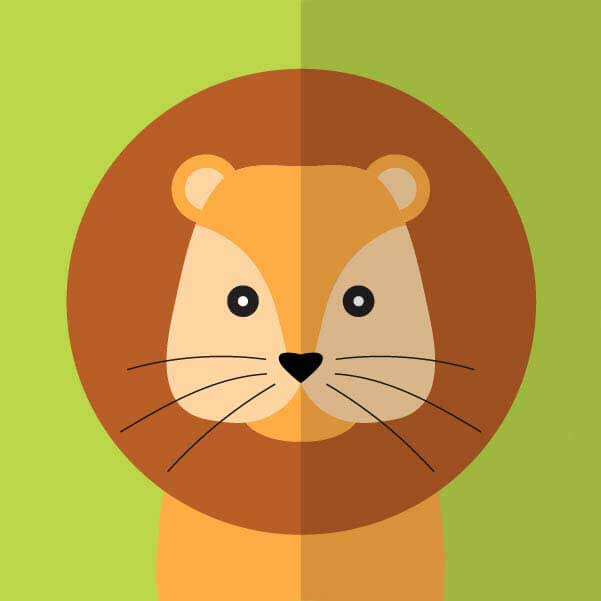
**计数排序**

[](https://www.jianshu.com/u/8bcd823f7b49)

[zhipingChen](https://www.jianshu.com/u/8bcd823f7b49) 关注

2018.08.28 02:45\* 字数 1225 阅读 1805评论 0喜欢 2

计数排序是一种非比较性质的排序算法，元素从未排序状态变为已排序状态的过程，是由额外空间的辅助和元素本身的值决定的。计数排序过程中不存在元素之间的比较和交换操作，根据元素本身的值，将每个元素出现的次数记录到辅助空间后，通过对辅助空间内数据的计算，即可确定每一个元素最终的位置。

比较性质排序算法的时间复杂度有一个理论边界，即 。 个元素的序列，能够形成的所有排列个数为 ，即该序列构成的决策树叶子节点个数为 ，由叶子节点个数可知，决策树的高度为 ，即由决策树根节点到叶子节点的比较次数为 ，由[斯特灵](https://zh.wikipedia.org/wiki/%E6%96%AF%E7%89%B9%E9%9D%88%E5%85%AC%E5%BC%8F" \t "_blank)公式， 转换可得，比较性质的算法复杂度理论边界为  。

**算法过程**

1. 根据待排序集合中最大元素和最小元素的差值范围，申请额外空间；
2. 遍历待排序集合，将每一个元素出现的次数记录到元素值对应的额外空间内；
3. 对额外空间内数据进行计算，得出每一个元素的正确位置；
4. 将待排序集合每一个元素移动到计算得出的正确位置上。

**演示示例**

待排序集合：[3, -1, 2, 3, 1]

***step 1:***

序列中最大值为：，最小值为：，根据序列中最大值和最小值的差值范围，可得申请额外空间大小为：

***step 2:***

因为申请的额外空间足以将  和  之间的所有元素记录，所以将待排序集合中每一个元素都记录到额外空间上，例如元素 3，对应的记录位置下标为：，即最后一个位置；元素 -1，对应的记录位置下标为：，即第一个位置。

所有元素的出现次数和元素值记录如下，其中  表示该元素出现的次数，表示元素值：

可以发现，计数排序的该过程，其实就是将待排序集合中的每个元素值本身大小作为下标，依次进行了存放。而记录的  次数，就是为了确定该元素值出现了几次。

***step 3:***

记录每个元素出现的次数，并对次数做计算，作用是当移动待排序集合元素到已排序集合中时，确保相同元素都被移动，且保持算法稳定性。因为额外空间中元素值是有序排列的，即额外空间的序列中每个元素，其元素的最终位置都是在前一个元素的后面，所以将其中每个元素的次数更新为加上前一个元素的次数和。例如元素 2 的最终位置在 元素 1 之后，而元素 1 只出现了 1 次，所以将元素 2 的  值更新为 3。

***step 4:***

根据额外空间已经确定的元素序列，移动待排序集合元素到已排序集合中。

**算法示例**

def countingSort(arr): # the elements in the array are all integers

maximum, minimum = max(arr), min(arr)

countArr = [0] \* (maximum - minimum + 1)

for i in arr: # record the number of times of every element in the array

countArr[i - minimum] += 1

for i in range(1, len(countArr)): # calculate the position of every element

countArr[i] += countArr[i-1]

targetArr = [None] \* len(arr)

for i in range(len(arr)-1, -1, -1): # reverse-order traversal is for the stability

countIndex = arr[i] - minimum

targetArr[countArr[countIndex] - 1] = arr[i]

countArr[countIndex] -= 1

return targetArr

第一个循环用于在额外空间中记录每一个元素出现的次数，复杂度为 ；第二个循环用于计算每一个元素的最终位置，复杂度为 ， 为申请的额外空间大小；第三个循环用于移动待排序集合中元素到已排序集合的正确位置上，复杂度为 。

**算法分析**

由算法示例可知，计数排序的时间复杂度为 。因为算法过程中需要申请一个额外空间和一个与待排序集合大小相同的已排序空间，所以空间复杂度为 。由此可知，计数排序只适用于元素值较为集中的情况，若集合中存在最大最小元素值相差甚远的情况，则计数排序开销较大、性能较差。通过额外空间的作用方式可知，额外空间存储元素信息是通过计算元素与最小元素值的差值作为下标来完成的，若待排序集合中存在元素值为浮点数形式或其他形式，则需要对元素值或元素差值做变换，以保证所有差值都为一个非负整数形式。