计数排序 Counting Sort

计数排序不是比较排序，它的基本思想是对每一个输入元素 x，确定出小于 x 的元素个数，从而知道它应该处于的位置。例如，有 10 个元素小于 x，则 x 应该被放在第 11 位上。注意，如果有元素相同时，不能把它们放在同一个位置上。

除了输入 array A 以外，我们还需要一个 array B 存放排序结果，一个 array C 提供临时存储区。

```
COUNTING-SORT(A, B, k)
 1  for i ← 0 to k
 2      do C[i] ← 0
 3  for j ← 1 to length[A]
 4      do C[A[j]] ← C[A[j]]+1
 5  ▷C[i]包含等于 i 的元素个数
 6  for i ← 1 to k
 7      do C[i] ← C[i]+C[i−1]
 8  ▷ C[i]包含小于或等于 i 的元素个数
 9  for j ← length[A] downto 1
10      do B[C[A[j]]] ← A[j]
11         C[A[j]] ← C[A[j]]−1
```

代码第 1，2 行初始化数组 C。

代码第 3，4 行是在统计数组 A 中每个元素出现次数。也许用 C[A[j]]++ 更容易理解。统计后的结果被存放在数组 C 中。

代码第 6，7 行让数组 C 不再显示每个元素出现次数，而是显示小于等于该元素的元素总数。具体做法是将 index 位置的元素自身与前一个元素相加。这样一来，后一个元素同样可以通过与前一个元素的值相加来得到小于等于元素的个数。

最后，就是将得到的结果放到输出数组 B 中。代码第 10 行，B[C[A[j]]]表示 A[j]元素在 B 中应处于的位置，在该位置给予 A[j]的值。代码第 11 行做了 C[A[j]] - -操作，这是为了处理上面提到的元素相同的情况。让 A[j]位置元素出现次数减少 1，则避免了让相同元素被放置在同一位置，减少出现次数后的元素会被放在原相同元素的之前一位。

所以计数排序的总体流程就是计算每个元素的出现次数后，再根据出现次数为它们进行排序。

计数排序运行时间

我们知道，比较排序模型的运行时间下界是 omega(nlgn)，但计数排序不是基于比较排序模型的。通过上面的伪代码可以发现，计数排序的运行时间仅需要 O(n)。