Counting Sort

Counting sort assumes that each of the elements is an integer in the range 1 to k, for some integer k. When $k = O(n)$, the Counting-sort runs in $O(n)$ time. The basic idea of Counting sort is to determine, for each input elements x, the number of elements less than x. This information can be used to place directly into its correct position. For example, if there 17 elements less than x, than x belongs in output position 18.

In the code for Counting sort, we are given array $A[1 \, . \, . \, n]$ of length n. We required two more arrays, the array $B[1 \, . \, . \, n]$ holds the sorted output and the array $c[1 \, . \, . \, k]$ provides temporary working storage.

COUNTING_SORT (*A*, *B*, *k*)

```
for i ← 1 to k do
    c[i] ← 0
for j ← 1 to n do
    c[A[j]] ← c[A[j]] + 1
//c[i] now contains the number of elements
equal to i
for i ← 2 to k do
    c[i] ← c[i] + c[i-1]
// c[i] now contains the number of elements
≤ i
for j ← n downto 1 do
    B[c[A[i]]] ← A[j]
    c[A[i]] ← c[A[j]] - 1
```

Each line below shows the step by step operation of counting sort.

A	3	6	4	1	3	4	1	4	C	2	0	2	3	0	1

C	2	2	4	7	7	8

B							4		C	2	2	4	6	7	8
B		1					4		C	1	2	4	6	7	8
B		1				4	4		C	2	2	4	5	7	8

B	1	1	3	3	4	4	4	6

Analysis

1. The loop of lines 1-2 takes $O(k)$ time
2. The loop of lines 3-4 takes $O(n)$ time
3. The loop of lines 6-7 takes $O(k)$ time
4. The loop of lines 9-11 takes $O(n)$ time

Therefore, the overall time of the counting sort is $O(k) + O(n) + O(k) + O(n) = O(k + n)$

In practice, we usually use counting sort algorithm when have $k = O(n)$, in which case running time is $O(n)$.

The Counting sort is a stable sort i.e., multiple keys with the same value are placed in the sorted array in the same order that they appear in the input array.

Suppose that the for-loop in line 9 of the Counting sort is rewritten:

$$\text{for } j \leftarrow 1 \text{ to } n$$

then the stability no longer holds. Notice that the correctness of argument in the CLR does not depend on the order in which array $A[1 \,..\, n]$ is processed. The algorithm is correct no matter what order is used. In particular, the modified algorithm still places the elements with value k in position $c[k - 1] + 1$ through $c[k]$, but in reverse order of their appearance in $A[1 \,..\, n]$.

Note that Counting sort beats the lower bound of $\Omega(n \lg n)$, because it is not a comparison sort. There is no comparison between elements. Counting sort uses the actual values of the elements to index into an array.

Back