Chapter 34: Counting Sort

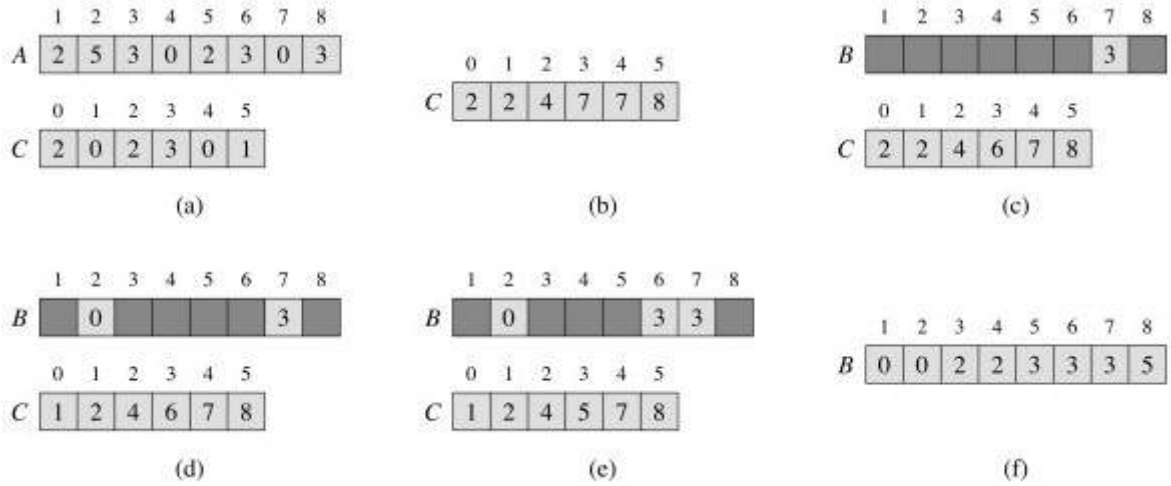
Section 34.1: Counting Sort Basic Information

Counting sort is an integer sorting algorithm for a collection of objects that sorts according to the keys of the objects.

Steps

1. Construct a working array *C* that has size equal to the range of the input array *A*.
2. Iterate through *A*, assigning *C*[x] based on the number of times x appeared in *A*.
3. Transform *C* into an array where *C*[x] refers to the number of values ≤ x by iterating through the array, assigning to each *C*[x] the sum of its prior value and all values in *C* that come before it.
4. Iterate backwards through *A*, placing each value in to a new sorted array *B* at the index recorded in *C*. This is done for a given *A*[x] by assigning *B*[*C*[*A*[x]]] to *A*[x], and decrementing *C*[*A*[x]] in case there were duplicate values in the original unsorted array.

Example of Counting Sort



Auxiliary Space: O(n+k)
Time Complexity: Worst-case: O(n+k), Best-case: O(n), Average-case O(n+k)

Section 34.2: Psuedocode Implementation

Constraints:

1. Input (an array to be sorted)
2. Number of element in input (n)
3. Keys in the range of *0..k-1* (k)
4. Count (an array of number)

Pseudocode:

```
for x in input:
    count[key(x)] += 1
total = 0
for i in range(k):
    oldCount = count[i]
    count[i] = total
    total += oldCount
```

```
for x in input:
    output[count[key(x)]] = x
    count[key(x)] += 1
return output
```