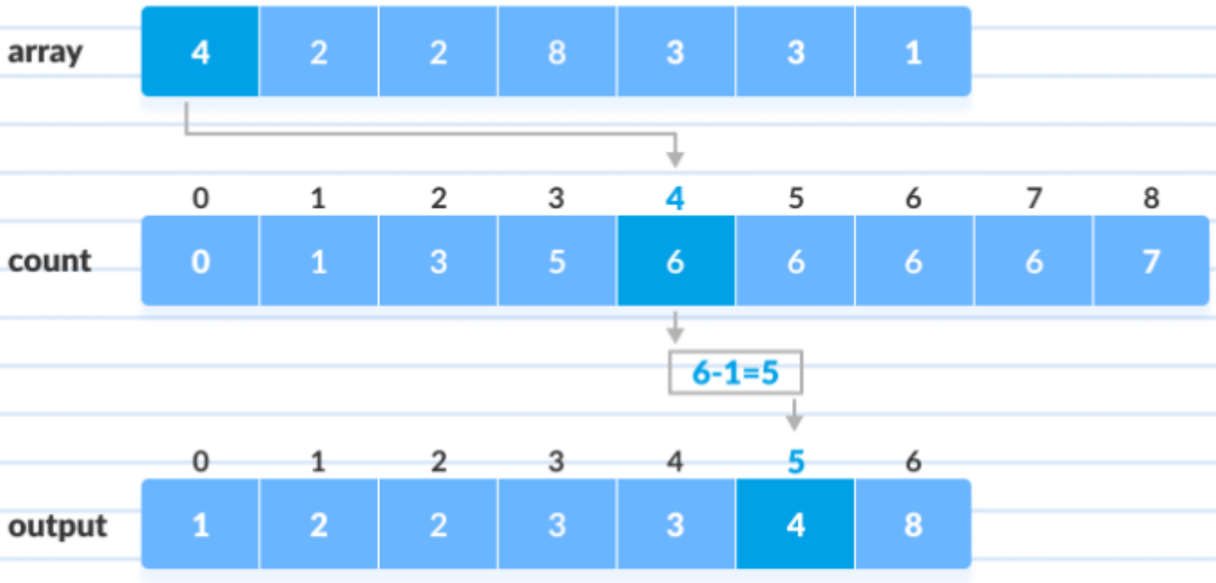
Distribution Counting Sort



Question: Are there any sorting algorithms that achieve time complexity superior to O(n log n) ?

Strategy: Space and Time Trade-Offs

Algorithm: Distribution Counting Sort

Define : Distribution Counting sort is an out-of-place, non-comparison sorting algorithm that sorts a list with duplicate values efficiently.

Example : an Array consists of these elements
[7,1,1,3,2,4,5,5,1,2,3]

Sort the Array

7,1,1,3,2,4,5,5,1,2,3

First : We make array to hold the Frequencies of the elements and it's size (biggest element -smallest elelment) we called it D

A 6,1,1,3,2,4,5,5,1,2,3

D 3,2,2,1,2,1

D 3,5,7,8,10,11

Second we calculate the distribution

Then : we use these frequencies to place the elements in order and put it in a new array S

S 1,1,1,2,2,3,3,4,5,5,6

The Code

```
def DistributionCountingSort(A, l, u):
  n=len(A)
  D=[0]*(u-l+1)
  S=[0]*n

  for j in range(u-l+1):
    D[j]=0

  for i in range(n):
    D[A[i] - l] += 1

  for j in range(1, u - l + 1):
    D[j] += D[j - 1]

  for i in range(n - 1, -1, -1):
    j = A[i] - l
    S[D[j] - 1] = A[i]
    D[j] -= 1
  return S
```

Analysis

$$\sum_{i=0}^{k-1} 1 + \sum_{i=0}^{n-1} 1 + \sum_{i=0}^{n-1} 1 + \sum_{i=0}^{k-1} 1$$

$$= [((k-1)-0)+1] + [((n-1)-0)+1] + [((k-1)-0)+1] + [((n-1)-0)+1]$$

$$= k + n + k + n$$

$$= 2n + 2k$$

$$= \in \theta(n + k)$$