2 - Counting Sort

Wednesday, 20 April, 2022 2:45 PM

not 0-indexed

```
COUNTING-SORT(A, B, k)
let C[0..k] be a new array
for i = 0 to k
    C[i] = 0
for j = 1 to A.length
    C[A[j]] = C[A[j]] + 1
// C[i] now contains the number of elements equal to i.
for i = 1 to k
    C[i] = C[i] + C[i - 1]
// C[i] now contains the number of elements less than or equal to i.
for j = A.length downto 1
    B[C[A[j]]] = A[j]
    C[A[j]] = C[A[j]] - 1
```

- initialize count array C to 0's
- count the total of each number in the range of k
- make every element the cumulative of previous elements

key = A[j]
new_location = C[key]
B[new_location] = key
C[key] -= 1

A : original array
B : sorted array
k : max value of number in A

C : count array to store the count of each number of array A.
if k = 5 then C[0...5]
C [| | | | |] (indices 0 1 2 3 4 5)

0-indexed's version

```
COUNTING-SORT(A, B, k)
let C[0..k] be a new array
for i = 0 to k                        -> O(k)
    C[i] = 0
for j = 0 to A.length - 1             -> O(n)
    C[A[j]] = C[A[j]] + 1
// C[i] now contains the number of elements equal to i.
for i = 1 to k                        -> O(k)
    C[i] = C[i] + C[i - 1]
// C[i] now contains the number of elements less than or equal to i.
for j = A.length - 1 downto 0         -> O(n)
    B[C[A[j]]] = A[j]          (struck out)
    C[A[j]] = C[A[j]] - 1      (struck out)
```

B[C[A[j]] - 1] = A[j]
C[A[j]] = C[A[j]] - 1

key = A[j]
new_loc = C[key] - 1
B[new_loc] = key
C[key] -= 1

Time complexity = O(k) + O(n) + O(k) + O(n)
= O(2n + 2k)
O(n + k)

- useful when range of values (k) in array is small

For example, you can use if if you want to sort 10,000 people according to their age. We can safely assume (for now) that no human is older than 199 years old, so the range of values is very small in this case.

From <https://www.youtube.com/watch?v=OKd534EWcdk>

- The above implementation is a stable sorting algorithm

input = <4, 1, 3, 4, 3>
same 3 / same 3
sorted_array = <1, 3, 3, 4, 4>

The relative order of elements is preserved here

Unstable =>

input = <4, 1, 3, 4, 3>
diff 3 / diff 3
sorted_array = <1, 3, 3, 4, 4>

The relative order of elements is not preserved

input = <4, 1, 3, 4, 3> , k = max(array) = 4
output : <Null, Null, Null, Null, Null>
count = <0, 0, 0, 0, 0>
number 0 1 2 3 4

⇓ counting the occurance of each number

count = <0, 1, 0, 2, 2>
number 0 1 2 3 4

⇓ make every element in count array the cumulative of previous elements

count = <0, 1, 0, 2, 2>
(+ + + +)
count = <0, 1, 1, 3, 5>
number 0 1 2 3 4

=> now, count - 1 = <-1, 0, 0, 2, 4>
is showing the new location of each number (which, in fact, is True!)

number 3 at index 2 of new array
output = <Null, Null, 3, Null, Null>

---

i = 4 :
key = input[4] = 3
new_loc = count[key] - 1
= 3 - 1
= 2

input = <4, 1, 3, 4, 3> (i = 4)

count = <0, 1, 1, 3, 5>
number 0 1 2 3 4

output[new_loc] = key
output[2] = 3

output = <Null, Null, (3), Null, Null>
new_loc = 2

count[key] = count[key] - 1
count[3] = 3 - 1 = 2

count = <0, 1, 1, (2), 5>
number 0 1 2 3 4

---

i = 3 :
key = input[3] = 4
new_loc = count[key] - 1
= 5 - 1
= 4

input = <4, 1, 3, 4, 3> (i = 3)

count = <0, 1, 1, 2, 5>
number 0 1 2 3 4

output[new_loc] = key
output[4] = 4

output = <Null, Null, 3, Null, 4>
new_loc = 4

count[key] = count[key] - 1
count[4] = 5 - 1
= 4

count = <0, 1, 1, 2, (4)>
number 0 1 2 3 4

---

i = 2 :
key = input[2] = 3
new_loc = count[key] - 1
= 2 - 1
= 1

input = <4, 1, 3, 4, 3> (i = 2)

count = <0, 1, 1, 2, 4>
number 0 1 2 3 4

output[new_loc] = key
output[1] = 3

output = <Null, (3), 3, Null, 4>
new_loc = 1

count[key] = count[key] - 1
count[3] = 2 - 1
= 1

count = <0, 1, 1, (1), 4>
number 0 1 2 3 4

---

i = 1 :
key = input[1] = 1
new_loc = count[key] - 1
= 1 - 1
= 0

input = <4, 1, 3, 4, 3> (i = 1)

count = <0, 1, 1, 1, 4>
number 0 1 2 3 4

output[new_loc] = key
output[0] = 1

output = <(1), 3, 3, Null, 4>
new_loc = 0

count[key] = count[key] - 1
count[1] = 1 - 1
= 0

count = <0, (0), 1, 1, 4>
number 0 1 2 3 4

---

i = 0 :
key = input[0] = 4
new_loc = count[key] - 1
= 4 - 1
= 3

input = <4, 1, 3, 4, 3> (i = 0)

count = <0, 0, 1, 1, 4>
number 0 1 2 3 4

output[new_loc] = key
output[3] = 4

output = <1, 3, 3, (4), 4>
new_loc = 3

count[key] = count[key] - 1
count[4] = 4 - 1
= 3

count = <0, 1, 1, 2, (3)>
number 0 1 2 3 4

End output = <1, 3, 3, 4, 4>