Thursday, 26 August 2021 12:08 PM

Counting Sort

→ sorts the elements by counting the [illegible] of distinct elements in the array.

Eg: 4 2 2 8 3 3 1

① find max of the array
max = 8

② make new array count (max + 1)

0	1	2	3	4	5	6	7	8
0	1	2	2	1	0	0	0	1

③ take info from original array to place elements in the

1	2	2	3	3	4	8	

Stability note

arr → [4 2 2^* 8 3 3 1]

output [arr[arr[i]] - 1] = arr[i]

count

0	1	2	3	4	5	6	7	8
0	1	2	2	1	0	0	0	1

output

1	2	2^*	3^*	3^*	4	8	

Time complexity → O(n + k)
k = max

Space complexity → O(max)

Randomized Quick Sort

Quick sort is different from other sorting algos because it can be randomized.

What is a randomized algo?

In a normal algo:

[input] → Algo → [output]

In a randomized algo:

[input] → Algo (random) → [output]

[input] → [output]

input remains same.

→ Quick sort is randomized.

arr

→ use the array ①.

random no. [0 . . . n-1]

We'll pick a random element
x = arr[random]
= arr[rand(0, . . . n-1)]

Split the array in 2 parts.

< x	≥ x

→ Linear time → we can partition the array

→ Recursively for the rest of the parts until the size of the array is 1 ①.

Same array

< x	≥ x

We need to sort all elements from l to r-1.

swap(l, r)
if (l < r) return ;
x = arr[random(l, r-1)]
m = l
for (i = l, . . . r-1)
if (arr[i] < x)
swap(arr[i], arr[m])
m++;

swap (l, m)
swap (m, r)

Time complexity

Worst case → pivot is min/max elt.

TC → O(n²)

Let's calculate avg. complexity of randomized algo, we'll need mean of variance of quantities.

good split

good split

good split → $\frac{1}{3}$
bad split → $\frac{2}{3}$

$\frac{n-1}{2}$ $\frac{n-1}{2}$ $\frac{n-1}{2}$ $\frac{n-1}{2}$

→ probability of each elt.

$T(n) = n + \left(\frac{n}{2}\right) + T(n-1) + T(n-1)$, $\frac{1}{n}$

$k = \frac{1}{3}$, $\left(k = \frac{2}{3}\right)$

$T(n) \leq \frac{1}{3}\left(\frac{n}{2} + T(n-1) + T(n-1)\right)$ + → good split

$+ \frac{2}{3}\left(\frac{n}{2} + T(n-1) + T(n-1)\right)$ → bad split

$T(n) \leq n + \frac{1}{3}\left(\frac{1}{2}T(n) + T(n-1)\right) + \frac{2}{3}\left(\frac{1}{2}T(n) + T(n-1)\right)$

$T(n) = \frac{2}{3}T(n) \leq n + T\left(\frac{2}{3}n\right) + T\left(\frac{2n}{3}\right)$

$T(n) \leq n + T\left(\frac{2}{3}n\right) + T\left(\frac{2n}{3}\right)$

$T(n) \leq 3n + T\left(\frac{2}{3}n\right) + T\left(\frac{2n}{3}\right)$

Proved by Induction:

$T(n) \leq c n \log_2 n$

$T(n) \leq 3n + c\frac{2}{3}n\log_2\frac{2}{3} + c\frac{2}{3}n\log_2\frac{2n}{3}$

$\log_2 . \log_3$ $\log_2 . \log_2$

$= n\left(3 + \frac{c}{3}\log_2 n + c\frac{2}{3}\log_2 n - \frac{2}{3}\log_2 3 - \frac{c2}{3}\log_2\frac{2}{3}\right)$

$= cn\log_2 n + n\left(3 - \frac{c}{3}\log_2 3 - c\frac{2}{3}\log_2\frac{2}{3}\right)$

≤ 0

$T(n) \leq c n \log_2 n$

$\frac{1}{3}$ good split $\frac{2}{3}$ bad split

→ every 3rd pick is a good pick.

① Pick 3 random elt.
② sort them
③ Pick middle one as pivot.

Java | Python

→ Tim sort
(Insertion sort + Merge sort)

C++

→ Intro sort
(Quick sort + Heap sort + Insertion sort)

Q. Given an array and a number k, k <= size of the array.

→ find the kth smallest elt. in the given array.

Eg: arr[7] → {7, 10, 4, 3, 20, 15, 8}
k = 3
ans → 8

Eg: arr[5] → {7, 3, 15, 20, 30, 5}
k = 4
ans → 30

Approach ①: Sort the array and pick the kth smallest elt.

TC → nlogn

Approach ②:

arr

pick a random elt. and split the array into 2 parts.

k = arr[random(0, . . . n-1)]

arr[k] → 8

< k	k	≥ k

pivot elt → ②

if (k < |<k|)
ans → [0, . . . m]

Yes

find (l, r, k):

if (l == r - 1) return a[l];
k = a[random(l, . . . r-1)]
m = l
for (i = l, . . . r-1)
if (a[i] < k)
swap(a[i], a[m])
m++;

if (k < m)
return find(l, m, k);
else return find (m, r, k);

$\left(n + \frac{2n}{3} + \frac{4n}{9} + \ldots\right) = 3n$

TC → O(n)