Date

ADM-HW4

3. Algorithmic question.

We can sort an integer array in time $O(n+R)$ using Counting sort Algorithm, where n is the size of array and R is the range of array (min - max).

PROOF:-

$A =$

2	1	0	5	4	8	7	9	3
0	1	2	3	4	5	6	7	8

func CountSort(A)

$n = size(A)$, $min = +\infty$, $max = -\infty$, $r = 0$

∴ First find min and max from A

≡S1 for(i=0 i<n i++) — n times

if $A[i] < min \Rightarrow min = A[i]$

if $A[i] > max \Rightarrow max = A[i]$

∴ $min = 0$ $max = 9$ $\approx$ $r = max - min = 9$

∴ Initialize empty array Count of size $r+1$ with zeros.

Count = Array(r+1)

Count =

0	0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9

≡S2 for(i=0 i<n i++) — n times

++Count[A[i]]

Count =

1	1	1	1	1	1	0	1	1	1
0	1	2	3	4	5	6	7	8	9

∴ Update the Count Array with commulative sum at each index

≡S3 for(i=1 i<=K i++) — R times

Count[i] = Count[i] + Count[i-1]

Count =

1	2	3	4	5	6	6	7	8	9
0	1	2	3	4	5	6	7	8	9

∴ Now initialize an Output Array with the same size as A

Output = ~~A~~Array(size(n))

Output = | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

∴ Now traverse the A in reverse and assign values in output array using below mechanism.

≡S4 for (i = n-1 i>=0 i--) — n times

Output[--count[A[i]]] = A[i]

Output = | 0 | 1 | 2 | 3 | 4 | 5 | 7 | 8 | 9 |

∴ Output is the sorted Array.

COMPLEXITY ANALYSIS :-

In the above algorithm we have 4 looping statement

- S1, S2, S3, S4

S1 = n , S2 = n , S3 = R , S4 = n

Hence the time complexity can be defined as

$$O(n+n+R+n) = O(n+R)$$

∴ The above algorithm is not valid for negative integers, but with some modifications (like normalizing the values) we can also make it work for negative integers.