Kumar
11911064
CSE

Q1 Time complexity Analysis Insertion sort

Ans In this sorting first we assume we have two part in our array sorted part and unsorted portion we will take element from unsorted portion and put it into its correct position in sorted portion

algorithm

```
for(i=1 ; i<=n ; i++):
        J = i-1
     temp = a[J+1]
   while(J>=0 && a[J] > temp)
     {
          a[J+1] = a[J]
             J = J-1
     }
   a[J+1] = temp

return a
```

Let us take example

let list | a_0 | a_1 | a_2 | a_3 | a_4 |

| a_0 | a_1 | a_2 | a_3 | a_4 | - - - - a_n |

sorted portion — unsorted portion

Step 1 compare a_1 with sorted portion

| | | a_2 | a_3 | a_4 | - - - | a_n |

smaller of a_0 & a_1 — larger of a_0 & a_1 — unsorted list

Step 2 Now we will compare a_2 with sorted

| | | | a_3 | a_4 | - - - | a_n |

sorted list — unsorted portion

so go on

Date : / /20
Page:

In best case there will be condition that whenever we compare element from unsorted portion to the sorted portion then element in unsorted portion is always greater than all elements of sorted portion it means list is ascending than every time there will be only 1 call

for n times = n-1 call

Time complexity = O(n-1) = O(n)

02 Bubble sorting Time Complexity

Ans in this Method of sorting we start from 0th Index to last index if arr [j] > arr[j+1] then we will swap them 2nd time we start from 0th index to 2nd last Index and do the same thing. we will keep doing it n-1 time let us take an array

first time | a_1 | a_2 | | . . . . | a_n |

check & swap

in first time we check n-1 time

2nd time → filled

check & swap

in 2nd time we check n-2 time

⋮

n-1 time | | | | . . . . | |

check & swap

in n-1 th time we check only one time.

Scanned with CamScanner

Total calls = $(n-1) + (n-2) + \ldots \ldots 1$

By using AP sum formula.

Total calls = $\frac{(n-1) \times (n)}{2}$

Time complexity = $O(n^2)$

Merge Sort

in this sorting we divide list into two half the again apply merge sort on two half after it we will merge the two sorted half

In Normal condition Merge sort is faster than slection, insertio & Bubble sort.

let us take example

$| a_0 | a_1 | a_2 | \ldots | a_n |$ = [| | |] [| | |]

we break it into 2 half

Let total time = $T(n)$
then for each half Time = $T(\frac{n}{2})$

Page:

$T(\frac{n}{2})$ $T(\frac{n}{2})$

② apply Merge Sort — apply Merge sort

Now we have applied Merge sort on both halfes.

2x

4x

Merge Sort

③ i j

compare. (takes $K_1 N$ time)

Now we will compare the two half

④ + =

Now we will Merge them into 1

It will take $K_2 N$ time

Here Recurrence Relation is

Date : / /20
Page:

$T(n) = 2T\left(\frac{n}{2}\right) + K_1 N + K_2 N$

Let $K_1 + K_2 = K$.

$T(n) = 2\cancel{T\left(\frac{n}{2}\right)} + Kn$

$2 \times \left[\cancel{T\left(\frac{n}{2}\right)} = 2\cancel{T\left(\frac{n}{4}\right)} + K\frac{n}{2} \right]$

$4 \times \left[\cancel{T\left(\frac{n}{4}\right)} + 2T\left(\frac{n}{8}\right) + K\frac{n}{4} \right]$

$\vdots$

$\cancel{T(1)} = K.$

Now we will add them.

$T(n) = \underbrace{Kn + Kn + Kn + \dots}_{\log n \text{ times}}$

$T(n) = Kn \log n$

$T(n) = O(n \log n)$