Name => Dheeraj

Roll No. => 11912020

Branch => IT

Best Case time Complexity of Insertion sort

Given array 2, 3, 4, 5

i	0	1	2	3	4
Element i					
Key			3	4	5
no: of comparison			0	0	0
movements			1	1	1

Dheeraj

Time complexity = 1+1+1 = 3

$= O(n-1)$

$\cong O(n)$

Modification in Insertion sort (Shell sort)

We observed that if the list is already sorted then time complexity in case of Insertion sort is $O(n)$. So we first sort the elements with some gap Taken in decreasing order. Then after apply Insertion sort

50 40 30 20 10

Total elements = $n = 5$

$gap = \left[\frac{5}{2}\right] = 2$

Dheeraj

50 40 30 20 10 (50 > 30 exchange)

30 40 50 20 10 (40 > 20 exchange)

30 20 50 40 10 (50 > 10 exchange)

30 20 40 50 10 (30 > 10 exchange)

10 20 30 40 50

~~(Now apply insertion sort)~~

$gap = \frac{2}{2} = 1$

Now apply insertion sort

No: of element scaned in each pass (or gap) = $n-1$

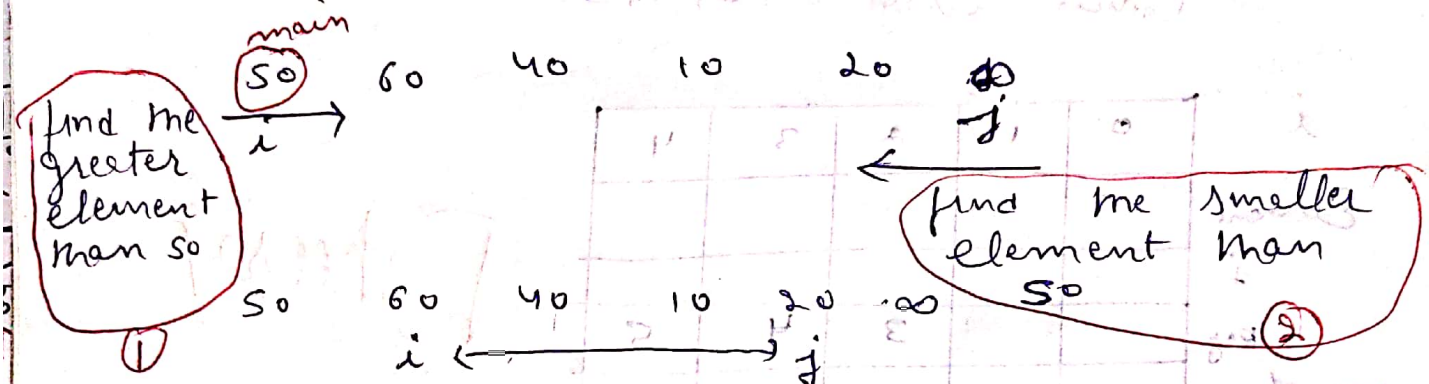
Dheeraj no: of passes performed = $2 = \log_2(n-1)$

time complexity = $(n-1)\log(n-1)$

$\approx O(n \log n)$

Quick Sort (Best or Avg Case)

50 60 40 10 20



50 60 40 10 20 ∞

i ← → j

swap elements

50 20 40 10 60 ∞

i j

repeat step 1 → ← repeat step 2

Cheers

50 20 40 10 60

j i

if $i > j$

swap j and main element

10 20 40 50 60

time complexity = $\underbrace{1+1}_{\text{compare}} + \underbrace{1}_{\text{swap}} + \underbrace{3+1}_{\text{compare}} + \underbrace{n}_{\text{swap}}$

$= 8 \approx n \log_2 n$

Time $\approx n \log n$

Quick Sort

Worst Case

10 20 30 40 50 ∞

(1) find element greater than 10

(2) find element smaller than 10

main 10 20 30 40 50 ∞

i j

(3) swap the main element with jth element

1 + n

swap comparisions

Dheeraj

20 30 40 50 ∞

do step 1 with 20

do step 2 with 20

20 30 40 50 ∞

j i

do step 3

1 + n − 1

swap comparisions

30 40 50

do step 1 with 30

do step 2 with 30

30 40 50

j i

do step 3 1 + n − 2

swap comparisions

⇒

40 50

do step 1 with 40 — do step 2 with 40

40 50 — 1 swap + n−3 comparisons

50

If a list contain n element then time complexity

$= 1 + n + 1 + n-1 + 1 + n-2 + \dots\dots\dots 1+1$

$= (1 + 1 + 1 \dots \text{n times}) + (n + n-1 + \dots\dots)$

$= n + \frac{n(n+1)}{2}$

$= \frac{2n + n^2 + n}{2}$

$= \frac{3n + n^2}{2}$

$\approx O(n^2)$ (Worst case)

Dheeraj

As it doesnot need an extra space so it is inplace algorithm.

Algo name	Cases	Time
Quick Sort	Best Avg	nlogn
	Worst	n^2
Merge Sort	Best case Avg Worst	n logn
insertion Sort	Best	n
	Avg	n^2
	Worst	n^2
Bubble Sort	Best	O(n)
	Avg	$O(n^2)$
	Worst	$O(n^2)$

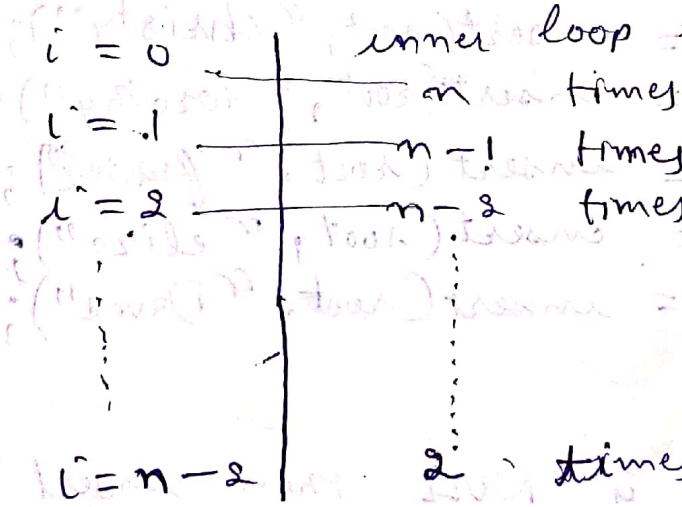
02 Bubble Sort

```
void BubbleSort(int arr[], int n)
{
    int i, j;
    for (i=0; i<n-1; i++)
    {
        for (j=0; j<n-i-1; j++)
        {
            if(arr[j] > arr[j+1])
                swap(&arr[j], &arr[j+1])
        }
    }
}
```

Dheeraj

Worst Case

when



$$\text{Total} = n + n-2 + n-3 + \dots + 2 + 1 - 1$$

$$= \frac{n(n+1)}{2} - 1$$

$$\approx O(n^2)$$

Best case $\approx O(n)$ (when element are already sorted)

Avg case $\approx O(n^2)$

Q4 BST for String.

Step 1: Insert the first string in tree

Step 2: Before insert second string

Step 3: compare key[i] & node → key[i]

Step 4: If key[i] is equal to node → key[i] then again do Step 3 with i++

Step 5: If it is key[i] < node → key[i] Insert that node to the left

Step 6: If it is key[i] > node → key[i] Insert that node to the right