Analytical Assesment - 02

D. Hemalatha
192311107
CSA0669

1) If $t_1(n) \in O(g_1(n))$ and $t_2(n) \in O(g_2(n))$, then $t_1(n) + t_2(n) \in O(\max\{g_1(n), g_2(n)\})$. prove that assertions.

Proof:- The proof extends to orders of growth the following simple fact about four arbitary real numbers a_1, b_1, a_2, b_2: if $a_1 \leq b_1$ and $a_2 \leq b_2$, then $a_1 + a_2 \leq 2 \max\{b_1, b_2\}$.

Since $t_1(n) \in O(g_1(n))$, there exist some position positive constant c_1 and some nonnegative integer n, such that

$$t_1(n) \leq c_1 g_1(n) \text{ for all } n \geq n;$$

Similarly, since $t_2(n) \in O(g_2(n))$,

$$t_2(n) \leq c_2 g_2(n) \text{ for all } n \geq n_2.$$

Let us denote $c_3 = \max\{c_1, c_2\}$ and consider $n \geq \max\{n_1, n_2\}$ so that we can use both inequalities. Adding them yields the following.

$$\begin{aligned} t_1(n) + t_2(n) &\leq c_1 g_1(n) + c_2 g_2(n) \\ &\leq c_3 g_1(n) + c_3 g_2(n) \\ &\leq c_3 [g_1(n) + g_2(n)] \\ &\leq c_3 \, 2 \max\{g_1(n), g_2(n)\}. \end{aligned}$$

Hence $t_1(n) + t_2(n) \in O(\max\{g_1(n), g_2(n)\})$, with the constants c and n_0 required by the definition O being $2c_3 = 2 \max\{c_1, c_2\}$ and $\max\{n_1, n_2\}$.

$\therefore t_1(n) \in O(g_1(n))$ and $t_2(n) \in O(g_2(n))$, then

$$t_1(n) + t_2(n) \in O(\max\{g_1(n), g_2(n)\}).$$

② Find the time complexity of the below recurrence equation.

③ $T(n) = \begin{cases} 2T(n/2)+1 & \text{if } n>1 \\ 1 & \text{otherwise} \end{cases}$

④ $T(n) = \begin{cases} 2T(n-1) & \text{if } n>0 \\ 1 & \text{otherwise} \end{cases}$

Sol: $T(n) = \begin{cases} 2T(n/2)+1 & \text{if } n>1 \\ 1 & \text{otherwise} \end{cases}$

$T(n) = aT(n/b) + f(n)$ [from Master's theorem]

Here $a=2$, $b=2$ | $f(n)=1$, $k=0$

$\log_b a = \log_2 2 = 1 > k$

$\log_b a > k$ then

$T(n) = O(n^{\log_b a})$

$= O(n^{\log_2 2}) = O(n^1) = O(n)$;

$T(n) = 2^n T(n-n)$

$= 2^n T(0)$ [$\because T(0)=1$]

$T(n) = 2^n$

∴ The time complexity is $T(n) = O(2^n)$.

5) Big O Notation: Show that $f(n) = n^2+3n+5$ is $O(n^2)$.

Given $f(n) = n^2+3n+5$

A function $f(n)$ is $O(g(n))$ if there exist constants $c > 0$ and n_0 such that for all $n \geq n_0$: $f(n) \leq c.g(n)$

$n^2+3n+5 \leq c.n^2$ [∵ Divide both sides with "n^2"]

$\Rightarrow 1 + \frac{3}{n} + \frac{5}{n^2} \leq c$

let $c=2$, then

$\Rightarrow 1 + \frac{3}{n} + \frac{5}{n^2} \leq 2$

$\Rightarrow \frac{3}{n} + \frac{5}{n^2} \leq 2-1$

$\Rightarrow \frac{3}{n} + \frac{5}{n^2} \leq 1$

$\Rightarrow \frac{3}{n} + \frac{5}{n^2} \leq \frac{1}{2} + \frac{1}{2}$

$\frac{3}{n} \leq \frac{1}{2}$ & $\frac{5}{n^2} \leq \frac{1}{2}$

$$T(n) = \begin{cases} 2T(n-1) & \text{if } n>0 \\ 1 & \text{otherwise} \end{cases}$$

To solve this problem, we will use Iteration method.

$T(n) = 2T(n-1)$ — (1)

$n = n-1 \Rightarrow$ (1)

$T(n-1) = 2.T((n-1)-1)$

$= 2T(n-2)$ — (2)

(2) in (1)

$T(n) = 2(2T(n-2))$

$= 2^2 T(n-2)$ — (3)

$n = n-2 \Rightarrow ①$

$T(n-2) = 2T(n-2-1)$

$= 2T(n-3) \rightarrow ④$

④ in ③

$T(n) = 2^2 [2T(n-3)]$

$= 2^3 T(n-3) \rightarrow ⑤$

Continuing this process

$T(n) = 2^k T(n-k)$

$n-k = 0$

$\Rightarrow k = n$

$n \geq 6$ & $n^2 \geq 10$

$n^2 \geq \sqrt{10} \approx 3.16$

$\therefore \boxed{n=6}$

$1 + \frac{3}{n} + \frac{5}{n^2} \leq 1 + \frac{1}{2} + \frac{1}{2} = 2$

Thus, for $\boxed{c=2}$

$n^2 + 3n + 5 \leq 2n^2$

$\therefore f(n) = n^2 + 3n + 5 \in O(n^2)$ //

6) Big Omega Notation: prove that $g(n) = n^3 + 2n^2 + 4n$ is $\Omega(n^3)$.

Given $g(n) = n^3 + 2n^2 + 4n$

A function $g(n)$ is $\Omega(f(n))$ if there exist positive constants c and n_0 such that for all $n \geq n_0$.

$g(n) \geq c \cdot f(n)$

$n^3 + 2n^2 + 4n \geq c \cdot n^3$ [Divide with n^3]

$1 + \frac{2}{n} + \frac{4}{n^2} \geq c$

let $\boxed{c = 1}$

$1 + \frac{2}{n} + \frac{4}{n^2} \geq 1$

Thus for $\boxed{c = 1}$

$n^3 + 2n^2 + 4n \geq 1 \cdot n^3$

$n^3 + 2n^2 + 4n \geq n^3$

$\therefore g(n) = n^3 + 2n^2 + 4n$ is $\Omega(n^3)$ //

7) Big Theta Notation: Determine whether $h(n) = 4n^2 + 3n$ is $O(n^2)$ or not.

Given $h(n) = 4n^2 + 3n$

A function $f(n)$ is $O(g(n))$ is there exist constants $c > 0$ and n_0 such that for all $n \geq n_0$:

$f(n) \leq c \cdot g(n)$

$4n^2 + 3n \leq c \cdot n^2$ [Divide with n^2]

$4 + \frac{3}{n} \leq c$

let $c = 5$, then

$4 + \frac{3}{n} \leq 5$

This is true for all $n \geq 1$. Therefore, we can take $c = 5$ and $n_0 = 1$

Thus, $h(n) = 4n^2 + 3n$ is $O(n^2)$.

8) Let $f(n) = n^3 - 2n^2 + n$ and $g(n) = n^2$ show whether $f(n) = \Omega(g(n))$ is true or false and justify your answer.

Given $f(n) = n^3 - 2n^2 + n$

& $g(n) = n^2$

$f(n) \geq c \cdot g(n)$

$n^3 - 2n^2 + n \geq c \cdot n^2$

$n^3 - 2n^2 + n - c \cdot n^2 \geq 0$

$n^3 - (2+c)n^2 + n \geq 0$

Let $\boxed{c = 1}$, then

$n^3 - (2+1)n^2 + n \geq 0$

$n^3 - 3n^2 + n \geq 0$

$n^3 - 2n^2 + n \geq c \cdot n^2$

Thus $\boxed{c = 1}$

$n^3 - 2n^2 + n \geq n^2$

$\therefore f(n) = n^3 - 2n^2 + n \in \Omega(n^2)$ //.

9) Determine whether $h(n) = n \log n + n$ is in $O(n \log n)$. Prove a rigorous proof for your conclusion.

Given $h(n) = n \log n + n$

$f(n) \leq c \cdot g(n)$

$n \log n + n \leq c \cdot n \log n$

$\log n + 1 \leq c \log n$

Let $c=2$, then

$\log n+1 \leq 2 \log n$

$1 \leq \log n$

therefore, $h(n) = n \log n + n$ is $O(n \log n)$ //

Solve the following recurrence relations and find the order of growth for solutions.

$T(n) = 4T(n/2) + n^2, T(1) = 1$

Given

$T(n) = 4T(n/2) + n^2, T(1) = 1$

By Master's theorem

$T(n) = aT(n/b) + f(n)$

Here $a=4$, $b=2$ | $f(n) = n^2$, $k=2$

$\log_b a = \log_2 4 = 2 = k$

$\therefore \log_b a = k$

$p > -1$ then

$= O(n^k \log_n p+1)$

$= O(n^2 \log_n 1+1)$

$= O(n^2 \log n)$ //

11) Given an array of [4, -2, 5, 3, 10, -5, 2, 8, -3, 6, -7, -4, 1, 9, -1, 0, -6, -8, 11, -9] integers find the maximum and minimum product that can be obtained by multiplying two integers from the array.

Given an array is

array: [4, -2, 5, 3, 10, -5, 2, 8, -3, 6, 7, -4, 1, 9, -1, 0, -6, -8, 11, -9]

Maximum product :-

1) Product of two largest +ve numbers.

2) product of two most negative numbers.

Minimum product :-

1) Product of largest +ve and most +ve numbers.

2) Product of two smallest -ve numbers.

Sorting the Array :-

Sorted array: [-9, -8, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]

Maximum product calculation :-

1) Product of two largest +ve numbers: $10 \times 11 = 110$

2) product of two most -ve numbers $= -9 \times -8 = 72$

Thus the maximum Product is 110.

Minimum Product calculation :-

1) product of largest +ve & most -ve numbers $= 11 \times -9 = -99$

2) Product of two smallest -ve numbers $= -9 \times -8 = 72$.

Thus, minimum product is -99.

∴ Maximum product = 110

Minimum product = -99.

1 Demonstrate Binary Search method to Search key = 23, from the array arr[] = {2,5,8,12,16,23,38,56,72,91}.

Given

array[] = {2,5,8,12,16,23,38,56,72,91}

0	1	2	3	4	5	6	7	8	9 (h)
2	5	8	12	16	23	38	56	72	91

P_1 P_2

$mid = \frac{l+h}{2} = \frac{0+9}{2} = 4 \Rightarrow arr[4] = 16$

key = 23 > arr[4] = 16

$mid = \frac{l+h}{2} = \frac{5+9}{2} = 7$

arr[7] = 56

key = 23 < arr[7] = 56

$mid = \frac{5+6}{2} = 5 \Rightarrow arr[5] = 23$

key = 23 = arr[5] = 23

The key = 23 is found at index 5 in array using the Binary Search Method//.

13) Apply merge sort and order the list of 8 elements Data d: (45, 67, -12, 5, 22, 30, 50, 20). Set up a recurrence relation for the number of key comparisons made by mergesort.

Given

d = (45, 67, -12, 5, 22, 30, 50, 20)

l	0	1	2	3	4	5	6	7	h
	45	67	-12	5	22	30	50	20	

$mid = \frac{l+h}{2} = \frac{0+7}{2} = 3$

0	1	2	3
45	67	-12	5

4	5	6	7
22	30	50	20

$mid = \frac{0+3}{2} = 1$

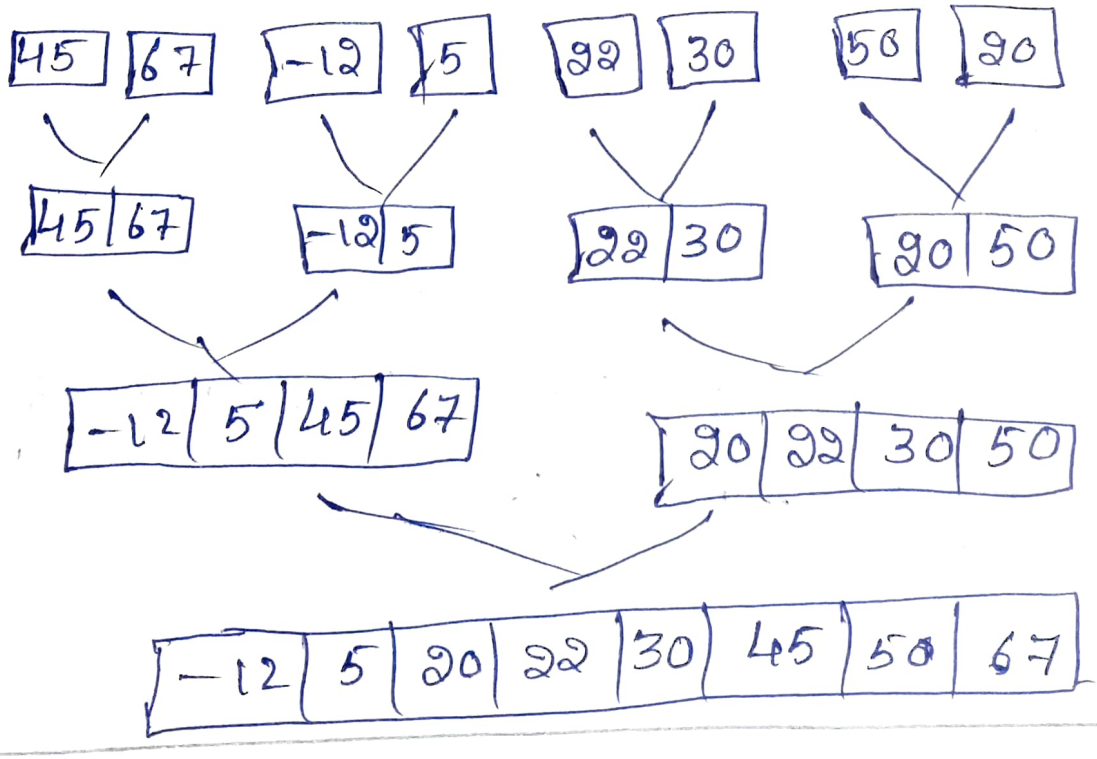
$mid = \frac{4+7}{2} = 5$

0	1
45	67

2	3
-12	5

4	5
22	30

6	7
50	20



Solving the recurrence relation:-

$T(n) = aT(n/b) + f(n)$

$T(n) = 2T(n/2) + O(n)$

Here $a = 2$, $b = 2$, $f(n) = O(n)$

$\log_b^a = \log_2^2 = 1$

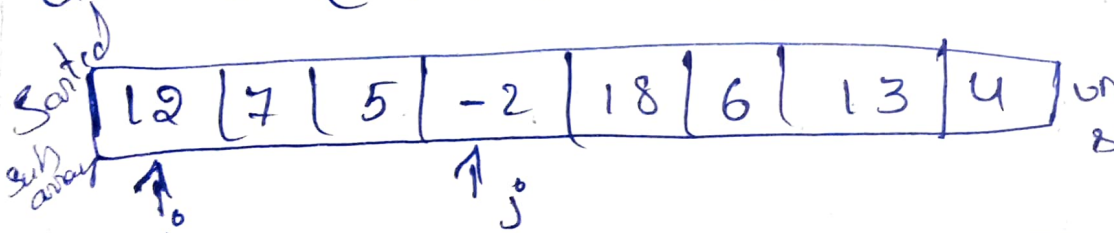
$n \log_b^a = n^1 = n$

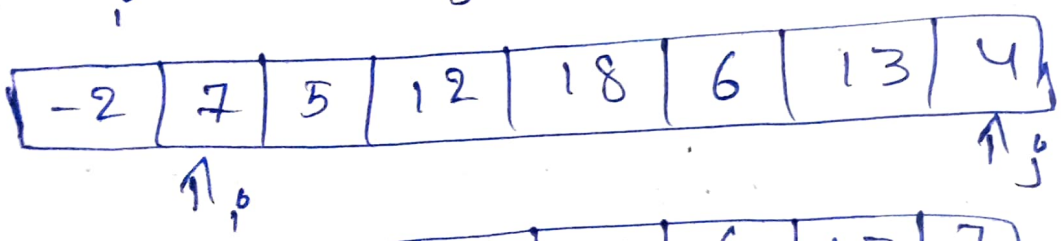
$T(n) = O(n \log n)$

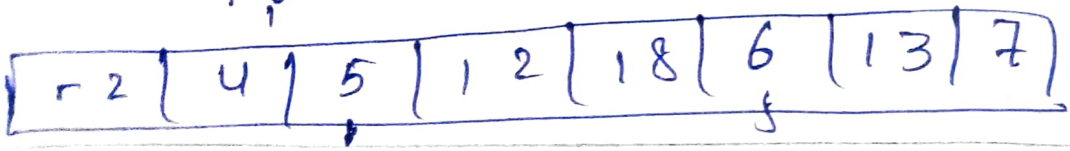
∴ Time complexity is $O(n \log n)$ //.

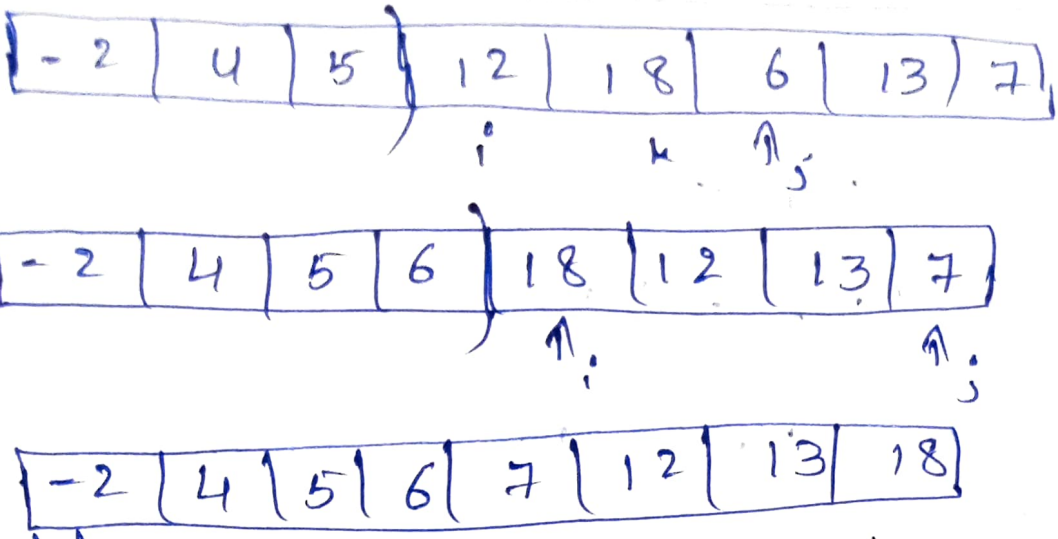
14) Find the no. of times to perform swapping for selection sort. Also estimate the time complexity for the order of notation set S(12, 7, 5, -2, 18, 6, 13, 4).

Given S = (12, 7, 5, -2, 18, 6, 13, 4)









Sorted array.

Total Swaps = 4, but as per theory, it should be n-1 = 7.

Best case = $O(n^2)$

Average case = $O(n^2)$

Worst case = $O(n^2)$ //

15) Find the index of the target value 10 using binary search from the following list of elements

[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]

Given list of elements arr[] =

[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
0 1 2 3 4 5 6 7 8 9

$mid = \frac{l+h}{2} = \frac{0+9}{2} = 4$

arr[4] = 10.

∴ the index of the target value 10 in the list is 4.

16) Sort the following elements using Mergesort divide-and-conquer strategy [38, 27, 43, 3, 9, 82, 10, 15, 88, 52, 60, 5] and analyze complexity of the algorithm.

Given elements

0	1	2	3	4	5	6	7	8	9	10	11
38	27	43	3	9	82	10	15	88	52	60	5

mid = $\frac{l+h}{2} = \frac{0+11}{2} = 5$

0	1	2	3	4	5
38	27	43	3	9	82

6	7	8	9	10	11
10	15	88	52	60	5

mid = $\frac{0+5}{2} = 2$

38	27	43

3	9	82

10	15	88

52	60	5

38	27

43

3	9

82

10	15

88

52	60

5

27	38

43

3	9

82

10	15

88

52	60

5

27	38	43

3	9	82

10	15	88

5	52	60

3	9	27	38	43	82

5	10	15	52	60	88

3	5	9	10	15	27	38	43	52	60	82	88

Algorithm :-

```
Merge sort (l, h)
{
 If (l < h)
 {
 Mid = (l+h)/2;          ← 1
 Merge sort (l, mid)     — n/2
 Merge sort (mid+1, h)
 n/2
 Merge (l, mid, h)       — n
 }
}
```

$T(n) = 2T(n/2) + n$

$a = 2$, $b = 2$ | $k = 1$

$\log_b^a = \log_2^2 = 1 = k$

$P > -1 \quad O(n^k \log_n^{P+1})$

$= (n^1 \log_n^{1+1})$.

$= n \log_n^2$

∴ Time complexity $= (n \log n)$.

Space complexity $= O(n)$ //.

17) Sort the array 64, 34, 25, 12, 22, 11, 90 using Bubble Sort. what is the time complexity of selection sort in the best, worst and average cases?

Given array

I-1

64 34 25 12 22 11 90

34 64 25 12 22 11 90

34 25 64 12 22 11 90

34 25 12 64 22 11 90

34 25 12 22 64 11 90

34 25 12 22 11 64 90

I-2

34 25 12 22 11 64 90

25 34 12 22 11 64 90

25 12 34 22 11 64 90

25 12 22 34 11 64 90

85 12 22 11 34 64 90

I-3

25 12 22 11 34 64 90

12 25 22 11 34 64 90

12 22 25 11 34 64 90

12 22 11 25 34 64 90

I-4

12 22 11 25 34 64 90

12 11 22 25 34 64 90

I-5

12 11 22 25 34 64 90

11 12 22 25 34 64 90

Time complexity of selection sort is -

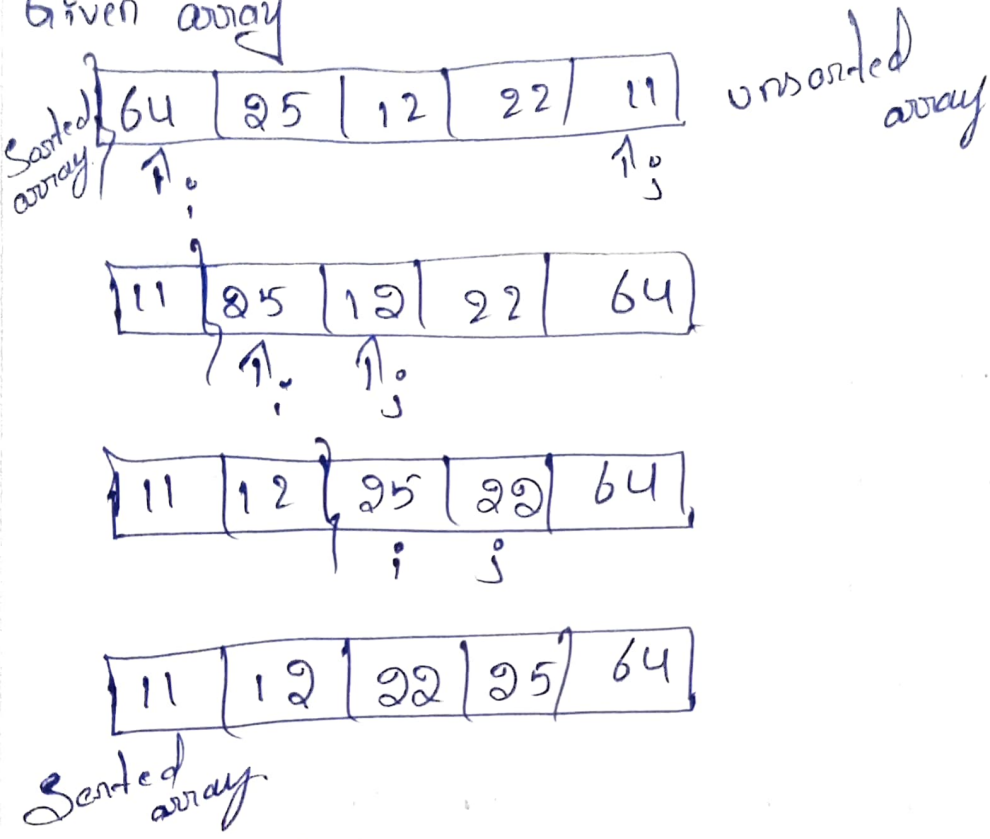
Best - $O(n)$

Average - $O(n^2)$

Worst - $O(n^2)$ //

18) Sort the array 64, 25, 12, 22, 11 using selection sort. what is the time complexity of selection sort in the best, average and worst cases?

Given array



Time complexity

Best — $O(n)$

Average — $O(n^2)$

worst — $O(n^2)$ //

19) Sort the following elements using insertion sort using Brute force Approach strategy [38, 27, 43, 3, 9, 82, 10, 15, 88, 52, 60, 5] and analyze complexity of the algorithm.

Given

38 27 43 3 9 82 10 15 88 52 60 5

27 38 43 3 9 82 10 15 88 52 60 5

3 27 38 43 9 82 10 15 88 52 60 5

3 9 27 38 43 82 10 15 88 52 60 5

3 9 10 27 38 43 82 15 88 52 60 5

3 9 10 15 27 38 43 82 88 52 60 5

3 9 10 15 27 38 43 52 82 88 60 5

3 9 10 15 27 38 43 52 60 82 88 5

3 5 9 10 15 27 38 43 52 60 82 88

Time Complexity :-

Best case - O(n) - this occurs when the array is already sorted. The inner loop will run only once.

Avg case - $O(n^2)$ - The list is randomly ordered,

Worst case - $O(n^2)$ - If the list is in reverse

Space complexity -

O(1) - Insertion sort.

20) Given an array of [4, -2, 5, 3, 10, -5, 2, 8, -3, 6, 7, -4, 1, 9, -1, 0, -6 - 8, 11, -9] integers sort the following elements using insertion sort using Basic

force approach strategy analyse complexity of algorithm.

Given array.

[4 -2 5 3 10 -5 2 8 -3 6 7 -4 1 9 -1 0 -6 -8 11 -9]

4 -2 5 3 10 -5 2 8 -3 6 7 -4 1 9 -1 0 -6 -8 11 -9

-2 4 5 3 10 -5 2 8 -3 6 7 -4 1 9 -1 0 -6 -8 11 -9

-2 4 5 3 10 -5 2 8 -3 6 7 -4 1 9 -1 0 -6 -8 11 -9

-2 4 3 5 10 -5 2 8 -3 6 7 -4 1 9 -1 0 -6 -8 11 -9

-2 3 4 5 10 -5 2 8 -3 6 7 -4 1 9 -1 0 -6 -8 11 -9

-2 3 4 5 -5 10 2 8 -3 6 7 -4 1 9 -1 0 -6 -8 11 -9

-2 3 4 -5 5 10 2 8 -3 6 7 -4 1 9 -1 0 -6 -8 11 -9

-2 3 -5 4 5 10 2 8 -3 6 7 -4 1 9 -1 0 -6 -8 11 -9

-2 -5 3 4 5 10 2 8 -3 6 7 -4 1 9 -1 0 -6 8 11 -9

-5 -2 3 4 5 10 2 8 -3 6 7 -4 1 9 -1 0 -6 -8 11 -9

-5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10 -6 -8 11 -9

-6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10 -6
-8 11 -9

-8 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9
10 11 -9

-9 -8 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10 11

Time Complexity -

Best case (O(n)) - This occurs when the array is already sorted. The inner loop will run only once for each element.

Average case (O(n²)) - The list is randomly ordered

Worst case (O(n²)) - If the list is in reverse order.//