Padala Deepthi pravallika
192311046

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.

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

Since $t_1(n) \in O(g(n))$, there exist some positive constant c_1 and some non-negative integer n, such that

$$t_1(n) \le c_1 g_1(n) \text{ for all } n \ge n_1$$

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

$$t_2(n) \le c_2 g_2(n) \text{ for all } n \ge n_2$$

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

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

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 $\max\{n_1, n_2\}$. The property implies that the algorithm's overall efficiency will be determined by the part with a higher order of growth, i.e., its least efficient part.

$\therefore t_1(n) \in O(g_1(n))$ 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) $T(n) = \begin{cases} 2T(n-1) & \text{if } n>0 \\ 1 & \text{otherwise} \end{cases}$

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

$T(n) = aT\left(\frac{n}{b}\right)+f(n)$ [∵ 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

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

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

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

$= 2^n T(0)$ [∵ $T(0)=1$]

$T(n) = 2^n$

∴ The time complexity is $\boxed{T(n) = O(2^n)}$ //

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 $n \geq n_0$:

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

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

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

Let $c = 2$, then

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

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

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

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

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

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

To solve this problem, we will use iteration method.

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

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

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

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

(2) in (1)

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

$= 2^2 T(n-2) \longrightarrow (3)$

$n = n-2 \Rightarrow (1)$

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

$= 2T(n-3) \longrightarrow (4)$

(4) in (3)

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

$= 2^3 T(n-3) \longrightarrow (5)$

Continuing this process:

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

$n-k = 0$

$k = n$

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

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

$\therefore n = 16$

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

Thus, for $c = 2$

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

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

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 $\boxed{c = 1}$

$n^3 + 8n^2 + 4n \geq 1 \cdot n^3$

$n^3 + 8n^2 + 4n \geq n^3$

$\therefore g(n) = n^3 + 8n^2 + 4n$ is $\Omega(n^3)$,,

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 $O(g(n))$ if 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 $O(n^2)$

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 $\boxed{c=1}$

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

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

Determine whether $h(n) = n\log n + n$ is in $O(n\log n)$ Pro
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 $\boxed{c=2}$, 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 -

$T(n) = aT\left(\frac{n}{b}\right) + f(n)$

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

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

$O(n^k \log_n^{1+1})$

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

ii) 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.

A) 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 = 10×11 = 110

② product of two most -ve numbers = -9×-8 = 72

Thus, the maximum product is 110

Minimum product calculation:

① product of largest +ve & Most -ve numbers

= 11×-9 = -99

② product of two smallest -ve numbers = -9×-8 = 72

Thus, minimum product is -99

∴ maximum product = 110

∴ minimum product = -99 //

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
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.

3) 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.

A) given

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

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

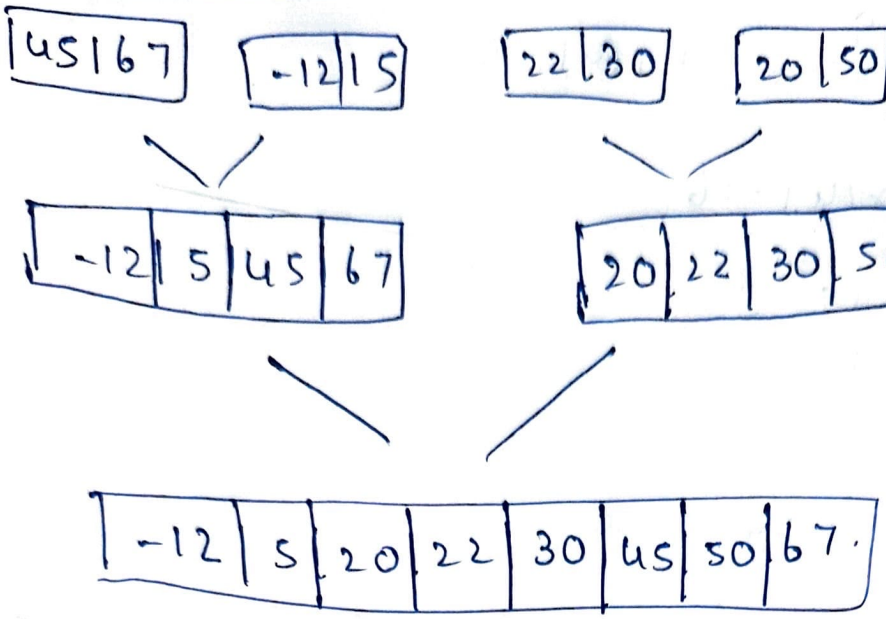
$mid = \frac{l+h}{2} = \frac{0+7}{2} = 3$

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

$mid = \frac{0+3}{2} = 1$ $\quad mid = \frac{4+7}{2} = 5.$

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

45	67	-12	5	22	30	50	20



Solving the recurrence relation -

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

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

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

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

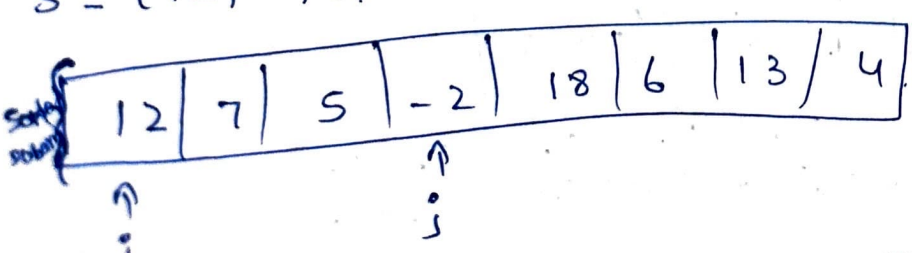
$$n \log_b a = n^1 = n$$

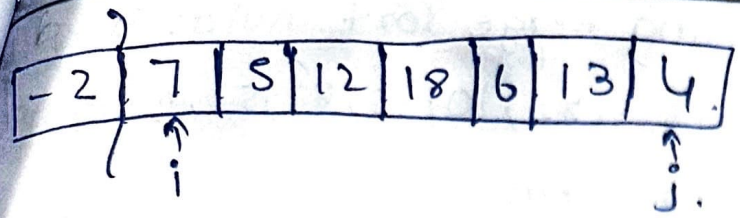
$$T(n) = O(n \log n)$$

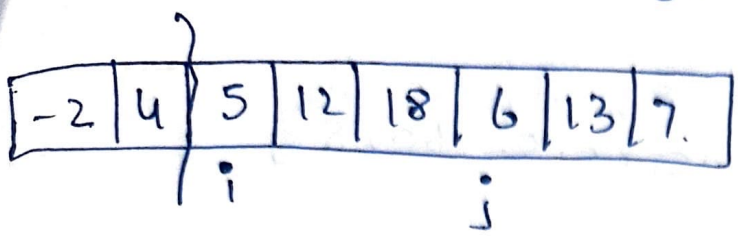
∴ Time complexity is $O(n \log n)$ //

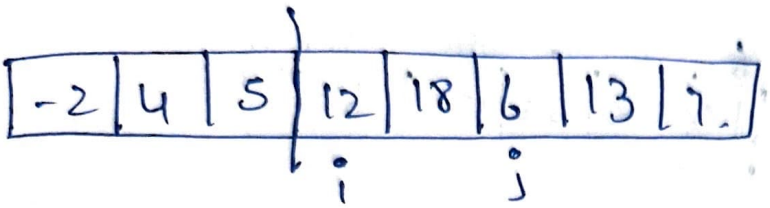
Find the order of notation set S(12, 7, 5, -2, 18, 6, 13, 4). no. of time to perform swapping for selection sort. given.

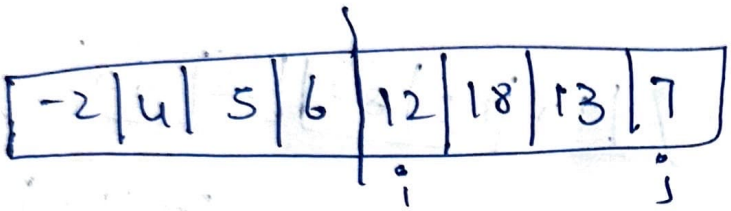
$S = (12, 7, 5, -2, 18, 6, 13, 4)$.

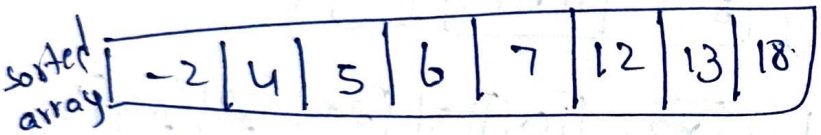












Total swaps = 4, bus as per theory, it should be n-1=7

Best case = $O(n^2)$

Average case = $O(n^2)$

Worst case = $O(n^2)$

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[] = h.

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

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

arr[4] = 10

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

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

A) 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

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
}
}
```

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

$a = 2 \mid k = 1$

$b = 2$

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

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

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

$= n \log n^2$

∴ Time complexity = $(n \log n)$ //

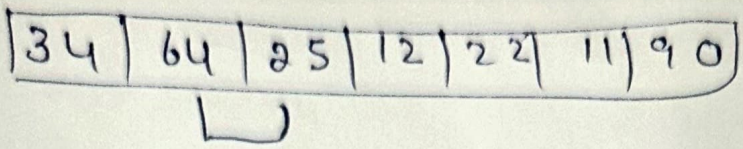
Space complexity = $O(n)$ //

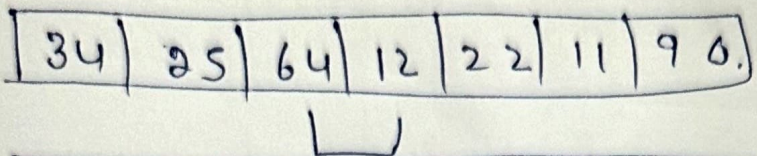
(13) sort the array 64, 34, 25, 12, 22, 11, 90 using Bubble sort. what is the time complexity of selection sort in the best, worst and average cases?

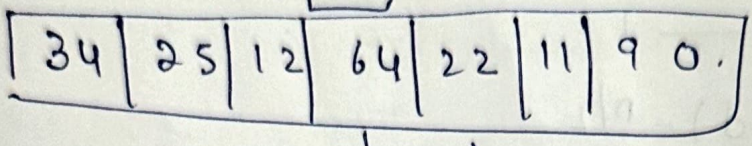
A) given array.

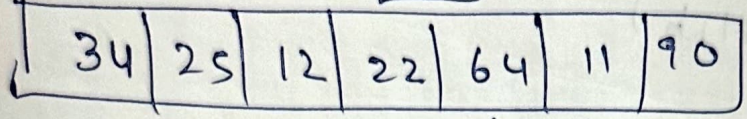
I-1

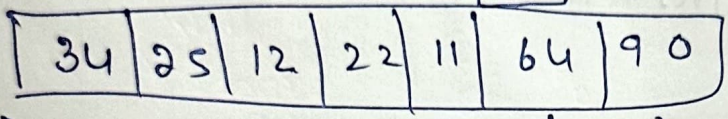
64	34	25	12	22	11	90


34 64 25 12 22 11 90

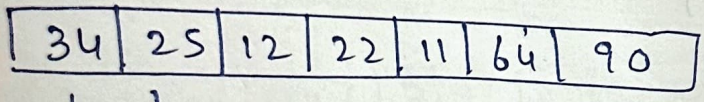

34 25 64 12 22 11 90

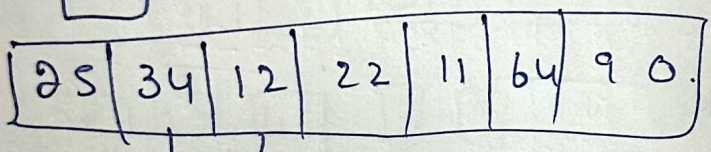

34 25 12 64 22 11 90

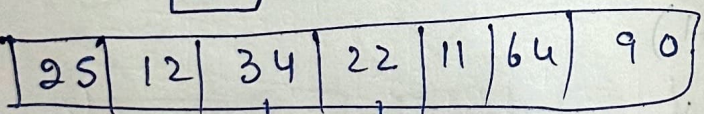

34 25 12 22 64 11 90

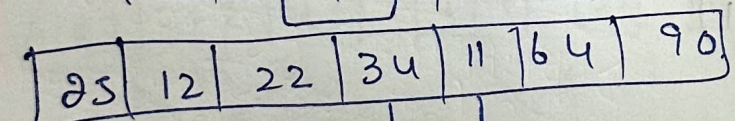

34 25 12 22 11 64 90

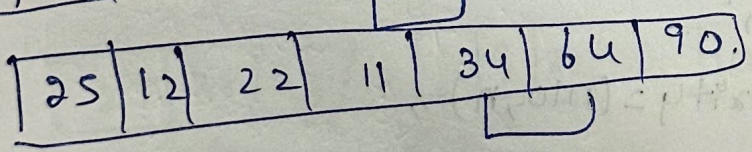
I-2


34 25 12 22 11 64 90

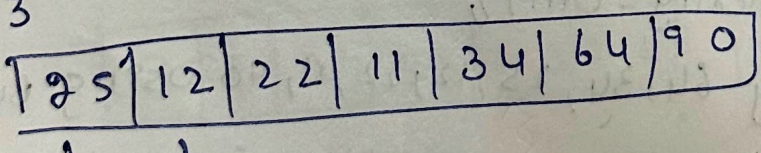

25 34 12 22 11 64 90

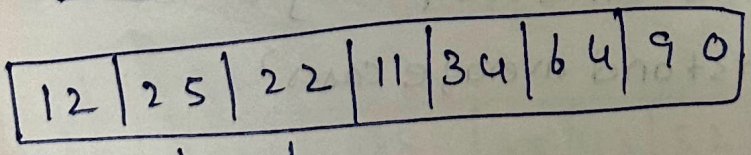

25 12 34 22 11 64 90

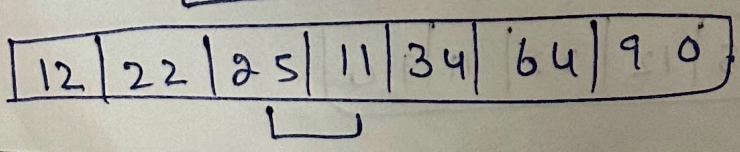

25 12 22 34 11 64 90

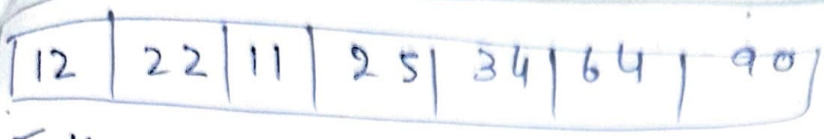

25 12 22 11 34 64 90

I-3

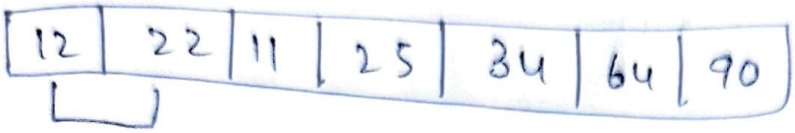

25 12 22 11 34 64 90

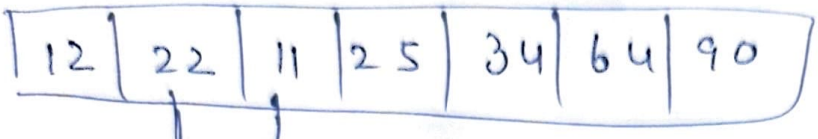

12 25 22 11 34 64 90

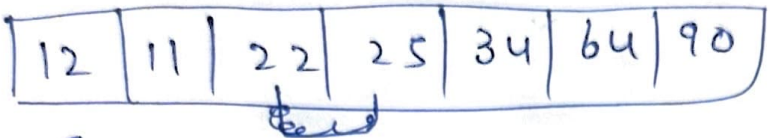

12 22 25 11 34 64 90



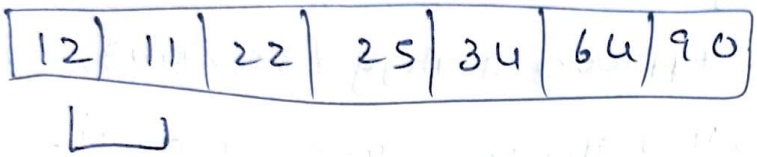
I-4.

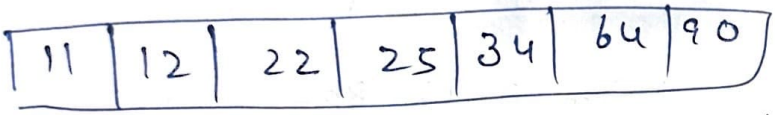






I-5





Time complexity of selection sort is -

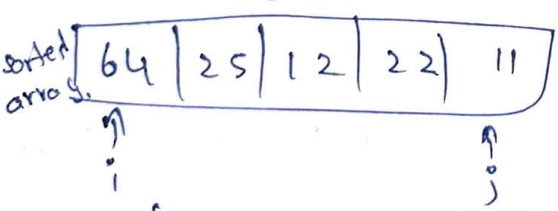
Best - $O(n)$

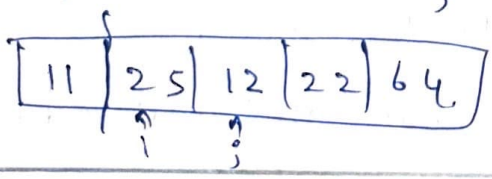
Average - $O(n^2)$

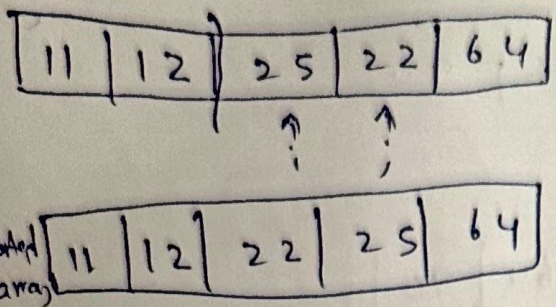
worst - $O(n^2)$ //

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?

A) 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 algorith

A) 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 15 88 52 60 5

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

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

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

3 9 10 15 27 38 43 52 60 82 88

3 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.

orst case - $O(n^2)$ - If the list is in reverse space.

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, 6, -6, -8, 11, -9] integers, sort the following element using insertion sort using Brute force approach strategy analyse complexity of algorithm.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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