1) if $t_1(n) \in O(g_1(n))$ and $t_2(n) \in O(g_2(n))$, Prove $t(n)+t_2(n) \in O(\max\{g_1(n), g_2(n)\})$

Sol:-

for any four real number a_1, b_1, a_2, b_2,

such that $a_1 \le b_1$ and $a_2 \le b_2$

we have $a_1 + a_2 \le 2 \max\{b_1, b_2\}$

Since $t_1(n) \in O(g_1(n))$, then there exists some Constant c_1 and non-negative integer n_1 such that

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

Since $t_2(n) \in O(g_2(n))$, then there exist some Constant c_2 and non negative integer n_2 such that

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

Let $c_3 = \max\{c_1, c_2\}$ and $n_0 = \max\{n_1, n_2\}$

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

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

2) find the time complexity of the reccurence relation

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

Sol:- $a=2$
$b=2$

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

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

$K=0$

$1 > 0$

$\log_b^a > K$

Case (i):-

$\Theta(n.\log_b^a)$

$\Theta(n.1)$

$\Theta(n)$

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

Backward Substitution:-

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

$n = n-1$

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

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

Sub (2) in (1)

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

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

$n = n-2$

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

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

Sub (4) in (3)

$T(n) = 2^2[(n-2)-1]$

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

$n = n-3$

$T(n-3) = 2T((n-3)-1)$

$T(n-3) = 2T(n-4) \rightarrow (6)$

Sub (6) in (5)

$T(n) = 2^3[2T(n-4)]$

$= 2^4 T(n-4) \rightarrow (7)$

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

$n-k=0$

$T(0)=1$

$T(n) = 2^k . T(0)$

$T(n) = 2^k \cdot 1$ $\therefore n = k$

$T(n) = 2^k$ $\boxed{T(n) = O(2^n)}$

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

To prove that $f(n) = n^2 + 3n + 5$ is $O(n^2)$

we need to find const 3 and n_0 such that

$f(n) \leq c \cdot n^2$ for all $n \geq n_0$

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

for $n \geq 1$, $n^2 \geq n$, ...

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

$f(n) = n^2 + 3n + 5 \leq 9n^2$ for $n \geq 1$

So, for $c = 9$ and $n_0 = 1$

$f(n) \leq c \cdot n^2$ for all $n \geq n_0$

That proves $f(n)$ is $O(n^2)$

6) Big omega notation: Prove that $g(n) = n^3 + 2n^2 + 4n$ is $\Omega(n^3)$

To prove that $g(n) = n^3 + 2n^2 + 4n$ is $\Omega(n^3)$

we need to prove, find const c and n_0 such that

$g(n) \geq c \cdot n^3$ for all $n \geq n_0$

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

for $n \geq 1$

$g(n) = n^3 + 2n^2 + 4n \geq n^3$

Since $2n^2$ and $4n$ are both less then n^3 when n

So, for $c = 1$ and $n_0 = 1$

$g(n) \geq c \cdot n^3$ for all $n \geq n_0$

That proves $g(n)$ is $\Omega(n^3)$.

7. Big Theta notation. Determine weather $h(n) = 4n^2 + 3n$ is $\Theta(n^2)$ or not.

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

for $n \geq 1$, $h(n) \leq 4 \cdot n^2 + 3n^2$

(since $3n$ is less than n^2 when $n \geq 1$)

for this simplifies to $h(n) \leq 7n^2$ for $n \geq 1$

therefore, $h(n)$ is $O(n^2)$

2. $h(n) = 4n^2 + 3n$ is $\Omega(n^2)$

for $n \geq 1$, $h(n) \geq 4n^2$

(since $3n$ is positive)

$\therefore$ $h(n)$ is $\Omega(n^2)$

Since $h(n)$ is both $O(n^2)$ and $\Omega(n^2)$ it is $\Theta(n^2)$

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

$n = 1$

$f(1) = 1^3 - 2(1)^2 + 1$ $\quad$ $g(1) = (-n)^2$

$= 1 - 2 + 1$ $\quad$ $= (-1)^2$

$= 0$ $\quad$ $= 1$

$n = 2$

$f(2) = 2^3 - 2(2)^2 + 2$ $\quad$ $g(2) = (-2)^2$

$= 8 - 8 + 2$ $\quad$ $= 4.$

$= 2$

$n = 3$

$f(3) = 3^3 - 2(3)^2 + 3$ $\quad$ $g(3) = (-3)^2$

$= 27 - 9 + 3$ $\quad$ $= 9.$

$= 21$

$n = 4$

$f(4) = 4^3 - 2(4)^2 + 4$ $\qquad$ $g(n) = (-4)^2$

$= 64 - 32 + 4$ $\qquad$ $= 16$

$= 32 + 4$

$= 36$

$n = 5$

$f(5) = 5^3 - 2(5)^2 + 5$ $\qquad$ $g(n) = (-5)^2$

$= 15 - 50 + 5$ $\qquad$ $= 25$

$= 35 + 5$

$= 40$

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

so, it is best case according to asymptotic notation.

$f(n) = \Omega(g(n))$

9) Determine weather $h(n) = n\log n + n$ is in $O(n\log n)$ prov a rigtous proof for your conclusion.

1. <u>upper bound</u>: (O notation)

we need to find c_1 and n_0 such that

$h(n) \leq c_1 \cdot n\log n$ for all $n \geq n_0$

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

$\leq n\log n + n\log n$ (since $\log n$ is increasing)

$= 2n\log n$.

now, Let $c_1 = 2$, then $h(n) \leq 2n\log n$ for all $n \geq 1$

So, $h(n)$ is $O(n\log n)$.

2. <u>Lower bound</u>: (Ω notation)

we need to find c_2 and n_0 such that

$h(n) \geq c_2 \cdot n\log n$ for all $n \geq n_0$

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

$\geq \frac{1}{2} \cdot n\log n$ (for $n \geq 2$)

now, let $c_2 = \frac{1}{2}$, then $h(n) \geq \frac{1}{2} \cdot n\log n$.

for all $n \geq 2$ so, $h(n)$ is $\Omega(n\log n)$.

3. <u>Combining Bounds</u>:

Since $h(n)$ is both $O(n\log n)$ and $\Omega(n\log n)$, it is so also $\Theta(n\log n)$

Thus, $h(n) = n\log n + n$ is $\Theta(n\log n)$.

10) Solve the following recurrence relation and find the order of growth for solution:

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

$a = 4$

$b = 2$

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

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

$K = 2 \quad \log_b a = K$

$2 = 2$

<u>Case (ii)</u>:-

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

$\Theta(n^2 \cdot \log_n^{1+1})$

$\Theta(n^2 \log_n^2)$

$T(n) = \Theta(n^2 \cdot \log(n))$

$\therefore$ The order of growth for the solution is $n^2 \cdot \log(n)$.

2. Demonstrate the binary search method to search key = 23 from the array arr[] = {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

low = 0

high = 9

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

mid = 4

a[mid] == key

a[4] == 23

16 != 23

16 < 23

low = mid+1

5	6	7	8	9
23	38	56	72	91

low = 5 high = 9

$mid = \frac{5+9}{2} = \frac{14}{2} = 7$

a[mid] == key

a[7] == 23

56 != 23 56 > 23

high = mid-1

5	6	7
23	38	56

low = 5 high = 7

$mid = \frac{5+7}{2} = 6$

a[6] == 23

38 != 23

38 > 23 high = mid-1

5
23

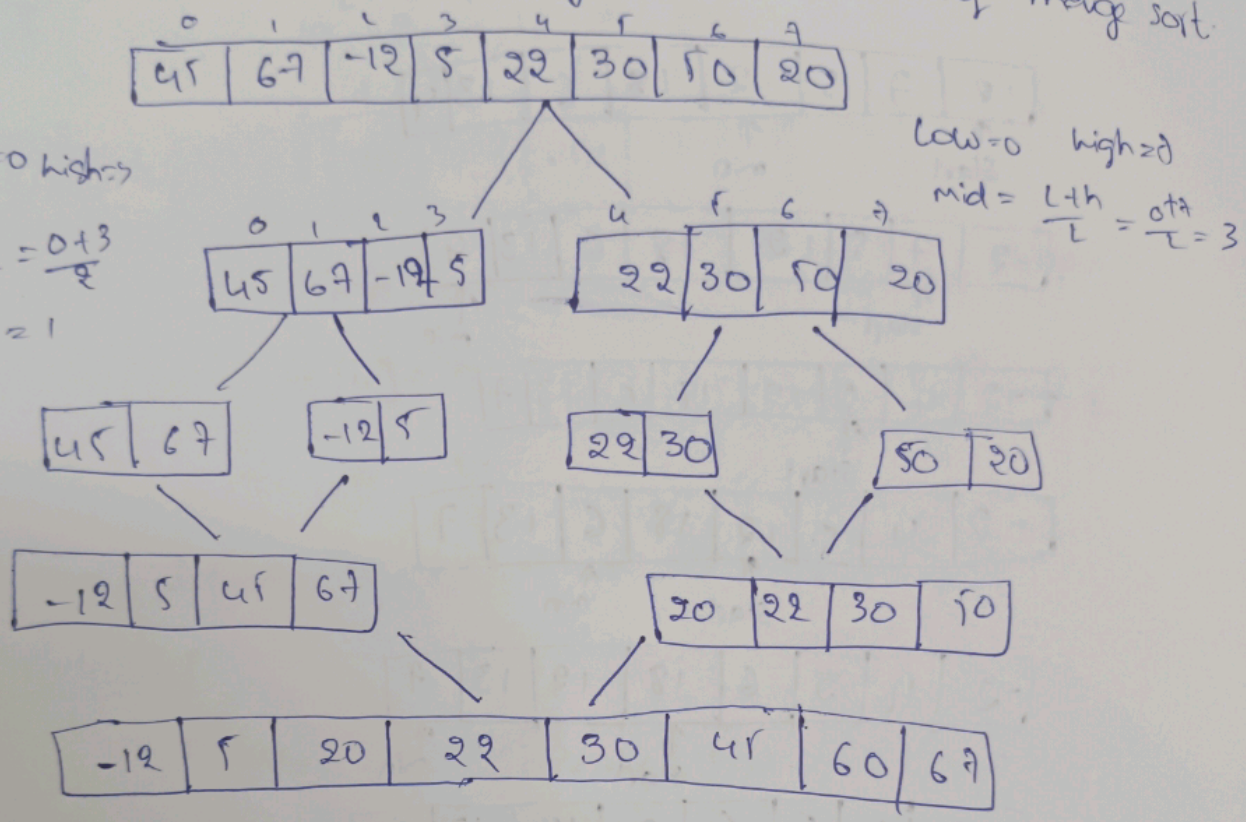
a[5] = key low = 5 high = 5

23 = 23

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

Return the position of the key i.e 5

3) Apply merge sort and order the list of 8 elements $d = \{45, 67, -12, 5, 22, 30, 50, 20\}$. Set up a recurrance relation for the number of key comparies made by merge sort.



∴ The sorted list is;

-12, 5, 20, 22, 30, 45, 60, 67

Time complexity $O(n\log n)$:

Recurrance relation: $T(n) = T(n/2) + (n-1)$

14. find the no of times to perform swaping for Selection sort. Also estimate the time complexity.

$S = \{12, 7, 5, -2, 18, 6, 13, 4\}$

12	7	5	-2	18	6	13	4

start min

-2	7	5	12	18	6	13	4

start min

-2	4	5	12	18	6	13	7

start

-2	4	5	12	18	6	13	7

start min

-2	4	5	6	18	12	13	7

start min

-2	4	5	6	7	12	13	18

start

-2	4	5	6	7	12	13	18

min

Sorted list: -2, -4, 5, 6, 7, 12, 13, 18

Usually, the number of swaps required will n-1 But for this question there are only 4 swaps.

Time Complexity: $O(n)^2$. it is n^2 in all the three co

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}

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

a [mid] = = key

low = 0 high = 9

a [4] = 10

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

10 = = 10 ✓

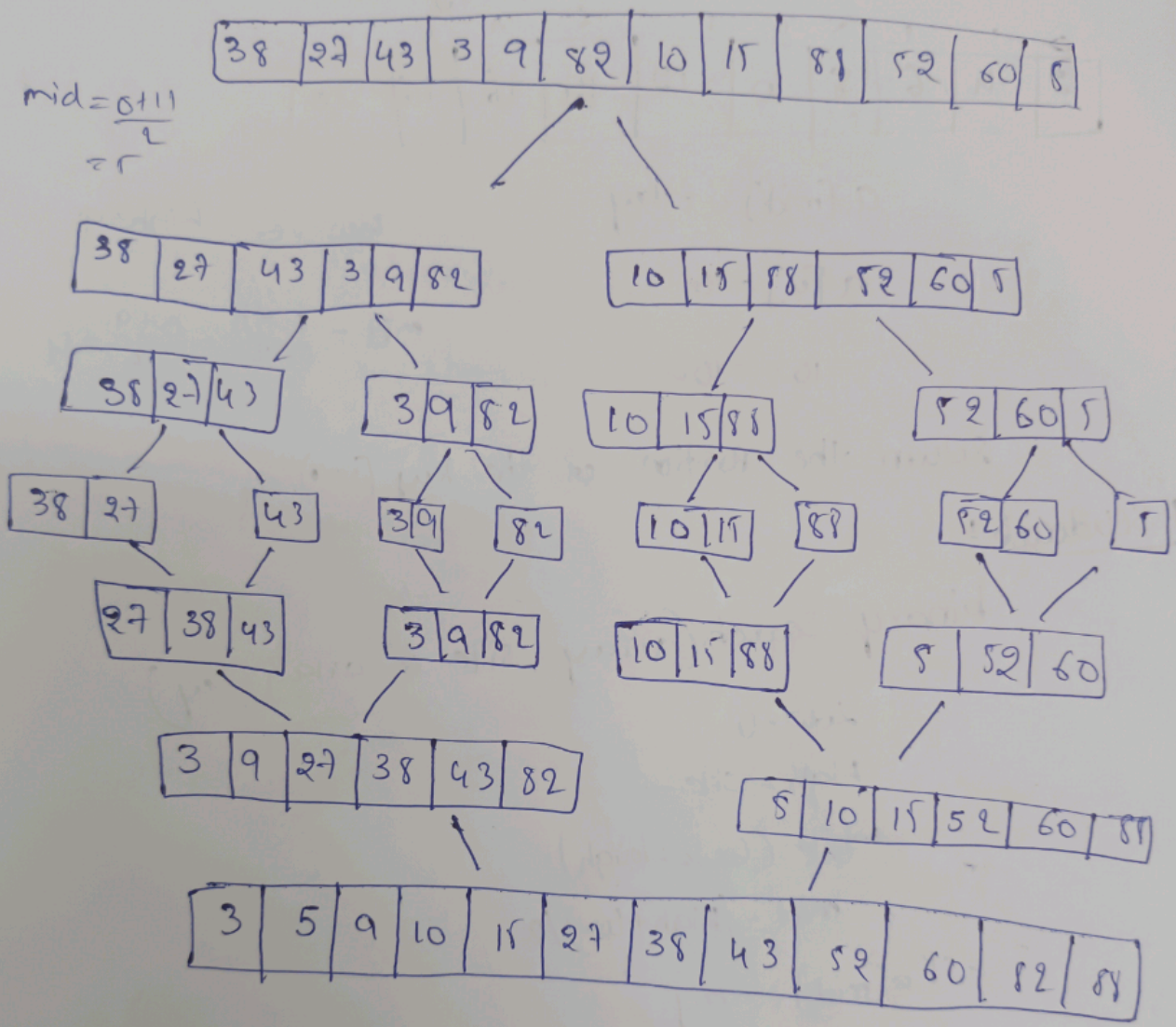
Return the position of the key (i = 4) 4

Pseudocode:

```
binary_search(array, size of array, key)
    low = 0
    high = size - 1
    while (low <= high)
        mid = (high + low)/2
    if a[mid] = = key
            return mid
    a[mid] > key
        high = mid - 1
    a[mid] < key
        low = mid + 1
return -1 [if not found]
```

16. Solve the elements using merge sort divide & conquer strategy [38, 27, 43, 3, 9, 82, 10, 15, 88, 52, 60, 5] & analyze the time complexity.

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



∴ The sorted list is, 3, 5, 9, 10, 15, 27, 38, 43, 52, 60, 82, 88.

P. Sedocode :

Partition (low, high)

if $l < h$

~~mid = l+h/2~~

mid = h+l/2

Partition (l, mid)

Partition (mid+1, h)

merge (l, mid, h)

end if

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

By using master's theorem.

$a = 2 \quad b = 2 \quad k = 1$

$\log_a^b = \log_2^2 = 1$

Comparing $\log_a^b$, k

$\log_a^b = k$

$\therefore$ Case(ii)

$p < -1$

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

$\therefore O(n \log n)$

$\therefore$ Time Complexity is $O(n \log n)$.

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

[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]

34 25 12 22 11 64 [90]

1st phase.

[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

25 12 22 11 34 64 [90]

2nd step

[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

12 22 11 25 [34] [64] [90]

[12 22] 11 25 34 64 90

12 [22 11] 25 34 64 90

12 [11 22] 25 34 64 90

12 11 22 [25] [34] [64] [90]

step-4

12 11 22 25 34 64 90

11 12 22 25 34 64 90 Step 5

11 12 22 25 34 64 90

---

11 12 22 25 34 64 90

11 12 22 25 34 64 90 Step 6

∴ The sorted array is {11, 12, 22, 25, 34, 64, 90}

Time Complexity: $O(n^2)$

<u>Selection sort:</u>

<u>Best case</u>: $O(n^2)$

<u>Avg case</u>: $\Theta(n^2)$

<u>worst case</u>: $\Omega(n^2)$

16. Sort the array 64, 25, 12, 22, 11 using Selection sort, What is the time Complexity.

64	25	12	22	11	90

start min

11	25	12	22	64	90

start mid

11	12	25	22	64	90

start min

11	12	22	25	64	90

start

11	12	22	25	64	90

i-bat (min)

11	12	22	25	64	90

The sorted list is 11, 12, 22, 25, 64, 90

<u>Time Complexity:-</u>

The time complexity of selection sort algorithm is always $O(n^2)$.

Best Case : $O(n^2)$

Avg case : $O(n^2)$

Worst Case : $O(n^2)$

19. insertion Sort of following elements and follow up with Time complexity & Analyze.

[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 52 60 5

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3 9 10 15 27 38 43 52 82 88 60 5 2)

3 9 10 15 27 38 43 52 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 5

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

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

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

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

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

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

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

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

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

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

∴ sorted list = 3, 5, 9, 10, 15, 27, 38, 43, 52, 60, 82, 88

Pseudocode :-

```
insertion_sort (a, size of a):
    for i in range (2, size): → (n-2)
        key = a[i] → 1
        j = i-1 → 1
    while j>0 and a[j]>key n*(n-2)
        a[j+1] = a[j] → 1
        j-= 1 → 1
        a[i+1] = key → 1
    return a - 1
```

Time Complexity :-

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

$$= n^2 - 2n + n - 2 + 6$$

$$= n^2 - n + 4$$

$$\therefore O(n) = O(n^2)$$

Time Complexity is $O(n^2)$.