1. If $t(1)(n) \in O(g1(n))$ and $t2(n) \in O(g2(n))$, then ~~t1(n)~~ $t1(n) + t2(n) \in O(\max\{g1(n), g2(n)\})$. Prove that assertions.

Solution:

Theorem:

if $t_1(n)$ belongs to $O(g_1(n))$ and $t_2(n)$ belongs to $O(g_2(n_2))$,

$$t_1(n) \in O(g_1(n))$$

$$t_2(n) \in O(g_2(n))$$

then,

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

Proof:

consider 4 orbitary real number a_1, a_2, b_1, b_2.

if $a_1 \leq b_1$ and $a_2 \leq b_2$.

then $a_1 + a_2 \leq 2 \max(b_1, b_2)$

$$t_1(n) < c_1 g_1(n) \qquad n \geq n_1$$

$$t_2(n) \leq c_2 g_2(n) \qquad n \geq n_2$$

Let $c_3 = \max\{c_1, c_2\}$ $\quad n \geq \max(n_1, n_2)$

$t_1(n) \leq c_3 g_1(n) \quad n \geq n \rightarrow (1)$

$t_2(n) \leq c_3 g_2(n) \quad n \geq n \rightarrow (2)$

$$t_1(n) + t_2(n) \leq c_3 g_1(n) + c_3 g_2(n)$$

$$\leq c_3 [g_1(n) + g_2(n)]$$

$$\leq c_3 \cdot 2 \max(g_1(n) \cdot g_2(n))$$

$$t_1(n) + t_2(n) \leq 2c_3 \max(g_1(n), g_2(n)).$$

$\therefore \quad t_1(n) + t_2(n) \in O(\max(g_1(n), g_2(n)))$

$c = 2c_3, \quad n \geq \max(n, n_2).$

$c = 2 \max(c_1, c_2).$

2. Find the time complexity of the below recurrence equation.

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

→ using master's theorem

$T(n) = A\,T(n/b) + f(n)$.

where, $A=2$, $b=2$, $K=1$, $P=1$. and $f(n) = n^K \log_a^b$

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

→ cases:

case 1: $\log_a^b > K. \rightarrow n^K \log_b^a$

case 2: $\log_a^b = K \Rightarrow P>-1 \rightarrow n^K \log n^{P+1}$

$P=-1 \rightarrow n^K \log\log n$

$P<-1 \rightarrow (n^K)$

case 3: $\log_b^a < K \Rightarrow P \geq 0, \rightarrow n^K \log^P n$

$P \leq 0 \rightarrow (n^K)$

Applying case 2, we get

$$P > -1 \qquad 1 > -1$$

$$n^k \log n^{1+1} \Rightarrow \log n^2$$

$$\Theta(n \log n^2) \Rightarrow \Theta(\log n)$$

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

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

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

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

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

$= 2T(n-3)$

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

$T(n) = 2^3(T(n-3).$

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

$\Rightarrow T(n) = 2^3.(2T(n-4)$

$T(n) = 2^4 T(n-4)$

General form: $2^k T(n-k)$ $n-k=1$

$k=n-1$

$T(n) = 2^{n-1} T(n-1-n)$

$\Rightarrow$ Time complexity is $O(2^n)$.

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

$\Rightarrow$ condition : $f(n) \leq c.n^2$.

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

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

divide by n^2,

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

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

Hence, $c=2$.

$\Rightarrow$ For all $n \geq 1$,

$$1+\frac{3}{1}+\frac{5}{1^2} = 9 \leq 2$$

so, $f(n) = n^2+3n+5$ is $O(n^2)$ with $c=2$ and $n_0=1$.

6. Big omega notation : prove that

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

$\Rightarrow$ condition : $g(n) \geq c.n^3$.

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

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

dividing by n^3,

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

take $c = 1$,

For all $n \geq 1$,

$$1 + \frac{2}{1} + \frac{4}{1^2} = 7 \geq 1.$$

so, $g(n) = n^3 + 2n^2 + 4n$ is $\Omega(n^3)$ with $c = 1, n_0 = 1$.

7. Big theta notation: Determine whether

$h(n) = 4n^2 + 3n$ is $\Theta(n^2)$ or not

$\Rightarrow$ condition :

$$g_1(n) \leq f(n) \leq g_2(n).$$

Take $c=4$,

for $n \geq 1$,

$4 + \frac{3}{1} = 7 > 4$.

$h(n) = 4n^2 + 3n$ is $\Omega(n^2)$ with $c=4$ and $n_0 = 1$.

⇒ Hence $h(n)$ is both $O(n^2)$ and $\Omega(n^2)$,

$h(n)$ is $\Theta(n^2)$.

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.

⇒ $f(n) \geq C \cdot g(n)$.

$n^3 - 2n^2 + n \geq C \cdot n^2$.

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

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

$n^3 \geq (C+2) \cdot n^2 - n$.

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

divide by n^2,

$$\frac{4n^2}{n^2} + \frac{3n}{n^2} \leq \frac{c.n^2}{n^2}$$

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

Take $c = 5$,

for $n \geq 1$,

$$4 + \frac{3}{1} = 7 \leq 5.$$

$h(n) = 4n^2 + 3n$ is $O(n^2)$ with $c = 5$ and $n_0 = 1$.

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

divide by n^2,

$$\frac{4n^2}{n^2} + \frac{3n}{n^2} \geq \frac{cn^2}{n^2}.$$

$$4 + \frac{3}{n} \geq c.$$

Take $c=1$

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

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

for sufficiently large n,

$$n^3 \geq 3n^2.$$

divide by n^2,

$$\frac{n^3}{n^2} \geq \frac{3n^2}{n^2}$$

$$n \geq 3$$

$\therefore f(n) = n^3 - 2n^2 + n$ is $\Omega(g(n))$. where $g(n) = n^2$ with

$c=1$ and $n_0 = 3$.

9. Determine whether $h(n) = n\log n + n$ is in $\Theta(n\log n)$.

prove a rigorous proof for your conclusion.

Solution: $h(n) \leq c\,g(n)$.

$h(n) = n\log n + n$. and $g(n) = n\log n$.

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

$n \geq n_0.$

simplifying: $n \log n$ (for $n\log n > 0$)

$$1 + \frac{1}{\log n} \leq c,$$

Take $c = 2$.

$$1 + \frac{1}{\log n} \leq 2.$$

$$\frac{1}{\log n} \leq 1.$$

$$\log n \geq 1.$$

$$n \geq e.$$

for all $n \geq e$, $\quad 1 + \frac{1}{\log n} \leq 2.$

$c = 2,\ n_0 = e.$

$n \log n + n \leq 2 \cdot n \log n$ for all $n \geq e$.

It implies:

$$n \log n + n \in O(n \log n).$$

we can conclude,

$$h(n) = n \log n + n \in O(n \log n).$$

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

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

→ using master theorem,

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

$a = 4, b = 2, f(n) = n^2, K =$

Case 1:

$\rightarrow (c = \log^a_b)$

if $f(n) < O(n^c)$ then $T(n) = \Theta(n^{\log^a_b})$.

case 2:

if ~~f(n)~~ $\log^b_a = k, \; p > -1 \Rightarrow n^k \log n^{p+1}.$

$p = -1 \Rightarrow n^k \log\log n$

$p < -1 \rightarrow (n^k)$

case 3: $\log^a_b < K \;\; \rightarrow p \geq 0 \rightarrow n^k \log^p n$

$p \leq 0 \rightarrow (n^k)$

$\log^b_a = \log^2_4 = 2.$

consider case 2,

$$T(n) = \Theta(n^{\log_b a}) \log n.$$

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

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.

solution:

→ sorted array:

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

⇒ To find the product, lets consider the maximum and minimum two values:

maximum = 10, 11.

mininum : −9, −8.

The maximum product = 10 × 11. = 110

The minimum product = −9 × 11 = 99

Algorithm :

Arr = [4, −2, 5, 3, 10, −5, 2, 8, −3, 6, 7, −4, 1, 9, −1, 0, −6, −8, +11, −9]

sorted [arr] .

n = len [arr]

max = arr [: −1] * arr [: −2]

min = arr [: 1] * arr [: 2].

product1 = max

product 2 = min.

print (" product 1 ; product 2).

12. Demonstrate Binary search method to search key = 2

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

$$mid = \frac{low + high}{2} = \frac{9}{2} = 4.5 \simeq 4.$$

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

if (A [mid] > (Key) || (A(mid] < Key) || (A [mid] = Key).

A [mid] < Key n+1.

4 + 1 = 5 → 23.

Hence value found at 5^{th} position.

Algorithm:

Sorted Array = A.

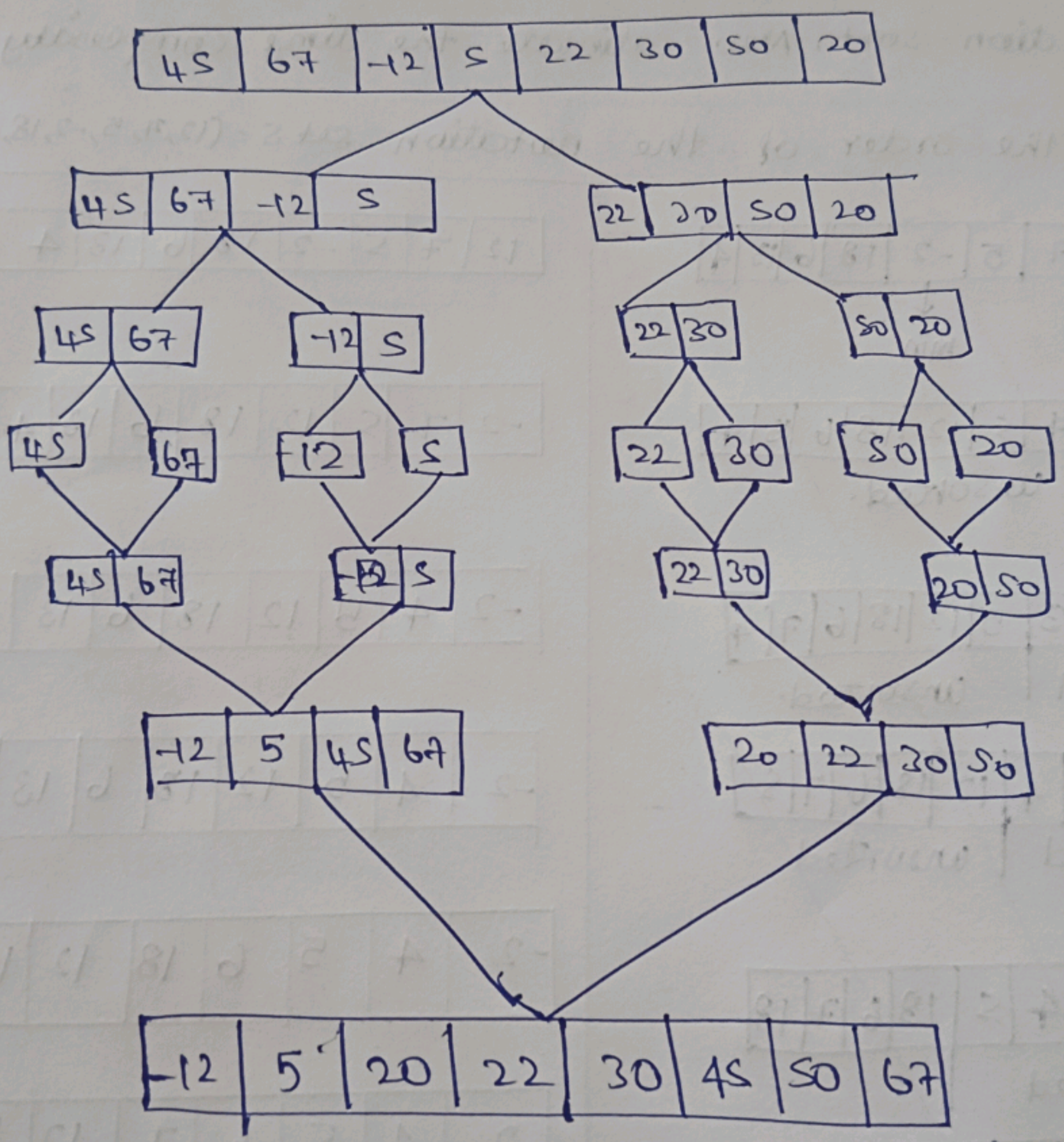
n, low = 0, high = n-1.

mid = (low + high) / 2

Key = 23.

```
while (low ≤ high)
{
 mid = (low + high) /2
 if (A[mid] > Key)
      high = mid-1
elseif (A[mid] < Key)
      low = mid + 1.
else return A [mid].
```

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.



Recurrence relation:

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

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

12	7	5	-2	18	6	13	4

min

-4	7	5	12	18	6	13	4

Sorted unsorted.

-4	3	5	12	18	6	7	4

Sorted unsorted.

4	3	4	12	18	6	7	5

sorted unsorted

-4	3	4	5	18	6.	7	18

sorted

-4	3	4	5	6	18	17	·

12	7	5	-2	18	6	13	4

-2	7	5	12	18	6	13	4

-2	4	5	12	18	6	13	7

-2	4	5	12	18	6	13	7

-2	4	5	6	18	12	13	7

-2	4	5	6	7	12	13	18

Total no of swaps

$= 1+1+3+2+2 = 9.$

Time complexity $= O(n^2)$.

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].

n = 2 4 6 8 10 12 14 16 18 20.

n/2 = 2 4 6 8 10 | 12 14 16 18 20

n/4 = 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

n/8 = 2 4 6 8 10 12 14 16 18 20

→ Binary search undergoes n/8.

→ Target value 10 is found at index value 4.

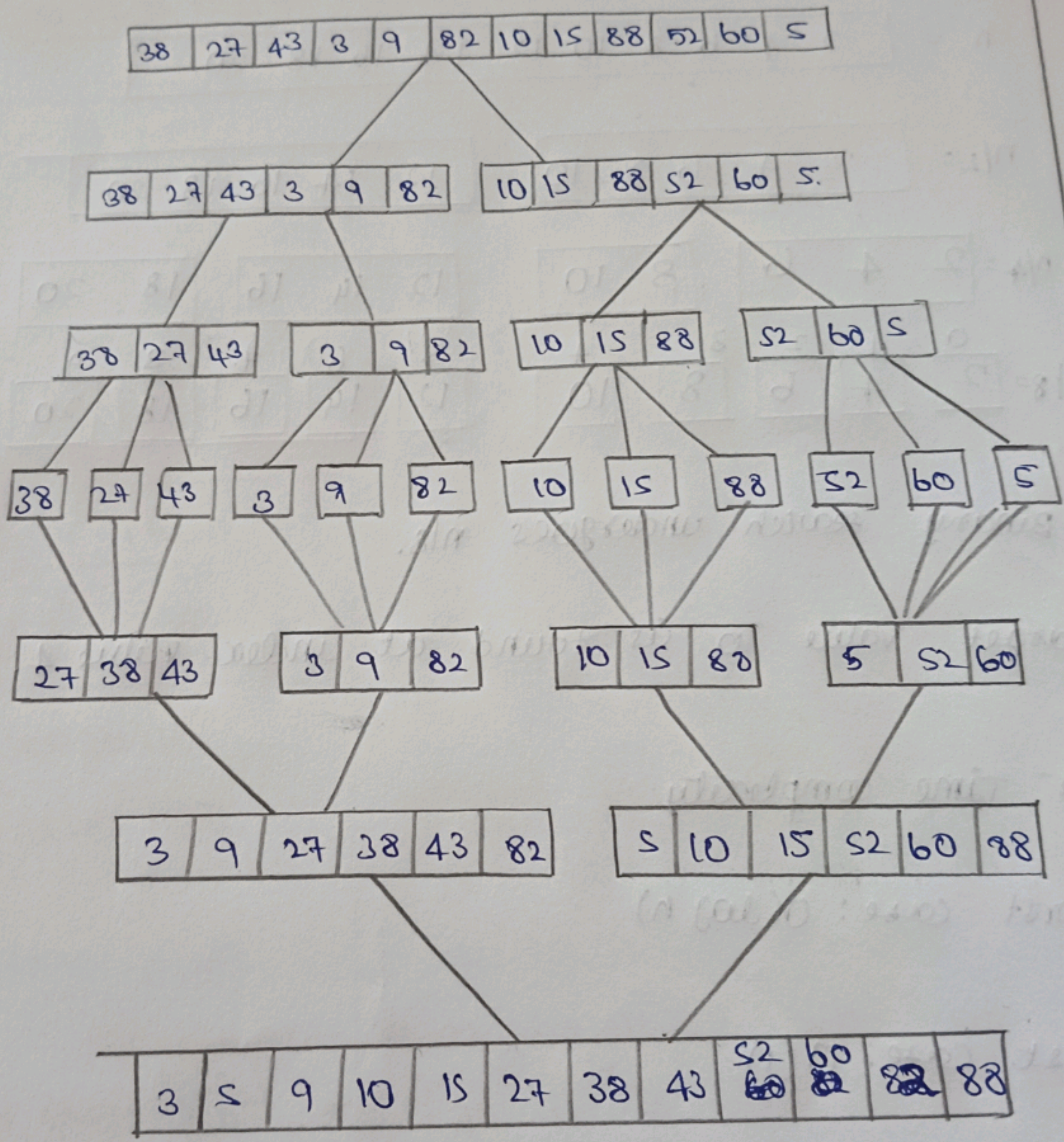
→ ~~wor~~ Time complexity:

Worst case: O(log n)

Best case: O(1)

Average case: O(log n).

16. Sort the following elements using merge sort - divide and conquer strategy and analyse complexity. → [38, 27, 43, 3, 9, 82, 10, 15, 88, 52, 60, 5].



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

17. Sort the array 64, 34, 25, 12, 22, 11, 90 using bubble sort. What is the time complexity in best, worst and average cases?

I:

64 34 25 12 22 11 90

34 64 25

25 34 64 12

12 25 34 64 22

12 22 25 34 64 11 90

11 12 22 25 34 64 90

Time complexity:

Best case: $O(n)$

Worst case: $O(n^2)$

Average case: $O(n^2)$

I

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.

II. 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

III 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

IV 12 22 11 25 34 64 90,

12 11 22 25 34 64 90

V 12, 11, 23, 25, 34, 64, 90

11, 12, 23, 25, 34, 64, 90.

18 sort the array using selection sort. [64, 25, 12, 22, 11].
what is the Time complexity in best, worst and average cases.

64	25	12	22	11.

↳ min

11	25	12	22	64

sorted | unsorted

11	12	25	22	64

sorted || unsorted

11	12	22	25	64

sorted | unsorted

11	12	22	25	64

sorted | unsorted

11	12	22	25	64

sorted

Time complexity :

Best case : $O(n)$

Average case : $O(n^2)$

worst case : $O(n^2)$.

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

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

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 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 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 5.

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 brute force approach strategy, analyse complexity of the Program.

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

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

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

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

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

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