1) If $T_1(n) \in O(g_1(n))$ & $t_2(n) \in O(g_2(n))$, then $t_1(n) + t_2(n) \in O(\max\{g_1(n), g_2(n)\}$.

Given,

$T_1(n) \in O(g_2(n)$, for all negative integers n_1 & c_1.

$T_1(n) \leq c_1 g_1(n)$ for all $n \geq n_1$.

since, $T_2(n) \in O(g_2(n))$. then, there is constant c_2 & non negative integer n_2.

$T_2(n) \leq g_2(n)$ for all $n \geq n_2$

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

$$
\begin{aligned}
T_1(n) + T_2(n) &\leq c_1 g(n) + c_2 g_2(n) \\
&\leq c_3 g_1(n) + c_3 g_2(n) \\
&\leq c_3 \{g_1(n) + g_2(n)\} \\
&\leq \max\{g_1(n) + g_2(n)\}
\end{aligned}
$$

$\therefore T_1(n) + T_2(n) \in O(\max\{g_1(n) + g_2(n)\}$

Hence proved.

2) Find time complexity

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

$$
T(n) = \{ 2T(n/2) + 1 \quad \text{if } n > 1.
$$

By master Theorem,

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

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

since,

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

$$\log_b^a > k, \quad T(n) = \theta(n^{\log_b^a})$$
$$= \theta(n^1)$$
$$= \theta(n)$$

$$\therefore T(n) = \theta(n).$$

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

$$T(n) = 2T(n-1) \quad \text{— (1)}$$

$n = n-1$

$$T(n-1) = 2T(n-1-1) = 2T(n-2) \quad \text{— (2)}$$

sub (2) in (1)

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

$$T(n) = 2^2 T(n-2) \quad \text{— (3)}$$

$n = n-2$

$$T(n-2) = 2T(n-3) \quad \text{— (4)}$$

sub (4) in (3)

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

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

$n = k, \quad n-k = 0.$

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

$T(0) = 1 \quad \Rightarrow$ Given.

so, $T(n) = 2^n \cdot (1)$

$= 2^n$

$\therefore \; T(n) = O(2^n).$

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

Given

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

$f(n) \leq c \cdot n^2.$

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

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

$n \geq 1,$

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

$n^2 + 3n + 5 \leq (1+3+5)n^2$

$n^2 + 3n + 5 \leq 9n^2.$

$c = 9, \; n_0 = 1$

$\therefore$ $f(n)$ is $O(n^2)$ with $c = 9$.

6) Big Omega notation:-

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

Given

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

$f(n) = n$

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

n=1

$1 \geq 7$

n=2

$2 \geq 24$ - NO.

$n4 = n^3 + 2n^2 + 4n$ $(n=3)$.

$3^4 = 81$

$81 \geq = 57$.

$f(n) = n4$

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

7) Big Theta notation:- $h(n) = 4n^2 + 3n \Rightarrow \Theta(n^2)$.

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

$c_1 g(n) \leq 4n^2 + 3n$

since $3n$ is less than n^2.

$h(n) \leq 7n^2$.

for $n \geq 1$

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

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

for $(n) \geq 1$

$h(n) \geq 4n^2$.

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

$\therefore$ since $h(n)$ is $O(n^2)$ & $\Omega(n^2)$ it is $\Theta(n^2)$.

8) Let $f(n) = n^3 - 2n^2 + n$ & $g(n) = n^2$

show whether $f(n) = \Omega(g(n))$ is true or false.

Given,

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

$g(n) = n^2$.

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

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

$n=1$, $1 - 2 + 1 \geq c \cdot 1$

$n=2$, $8 - 8 + 8 \geq 4$

$8 \geq 4$.

so, for $n \geq 2$, $f(n) = \Omega\, g((n))$ is true.

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

$n \geq n_0 1$

$c_1 \cdot n\log n \leq n(n) \leq c_2 \cdot n\log n$

upper bound

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

for $n \geq 1$, $\log n$ is positive

$\therefore n\log n + n \leq n\log n + n\log n$

$n\log n + n \leq 2n\log n$.

Lower bound :-

for $n \geq 1$, $\log n$ is positive,

$n + n\log n \geq n\log n$

$\therefore c_1 = 1;\ n(n) = n\log n + n \geq n\log n$

Thus, $n\log n \leq n\log n + n \leq 2n\log n$, for all $n \geq 1$.

Hence, $h(n) = n\log n + n$ is $\theta(n\log n)$.

10) solve the following recurrence relation

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

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

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

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

$k = 2,\ p = 1$

now,

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

$\log_b^a = k$.

$P \geq -1$, so, $\Theta(n^k \log^{P+1} n)$

$\Rightarrow \Theta(n^2 \log^1 n)$

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

$\therefore$ Order of growth of $T(n) = 4T(n/2) + n^2$ is

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

11) Given array of [4, -2, 5, 3, 10, -5, 2, 8, -3, 6, 7, -4, 1, 9, -1, 0, -6, -8, 11, -9] integers find minimum & maximum.

Given array

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

Sorted array

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

Max Product :-

-> 2 largest (+ve) number
-> product of 2 smallest (-) numbers

min product

-> 2 largest (+) number (if one is 0)/(-)
-> 2 smallest (-) negative numbers

Max Product

$10 \times 11 = 110$

↳ $(-9)\times(-8) = 72$

⇒ Max product = 110.

min product

$10 \times 11 = 110$

$-8 \times -9 = 72$

$-9 \times 11 = -99$

↳ max product = 110

min product = -99.

code:-

```
function maxmin_product (arr)
arr.sort()
max prod = max (arr[0]*arr[1] ,arr[-1] * arr[-2])
min pro = min (arr[0]* arr[1], arr[0]*arr[-1],
               arr[1]*arr[-1],)

return maxmin_product.

max - product, min_product = max_min product(arr).
```

12) Binary search method

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

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

low = 0 mid = 0+9/2

high = 9 = 4

arr[mid] = arr[4] = 16.

16 < 23, search in right half } 1st iteration
now, low = mid + 1 = 5.

=> mid = 5+9/2 = 7 } 2nd iteration
now,
56 > 23

high = mid - 1 = 6

mid = 5 + 6/2 = 5 } 3rd iteration
23 = 23; key found in index 5

	0	1	2	3	4	5	6	7	8	9
=>	2	5	8	12	16	23	38	56	72	91
					mid					high

=> 2 5 8 12 16 | 23 38 56 | 72 91
low ... mid

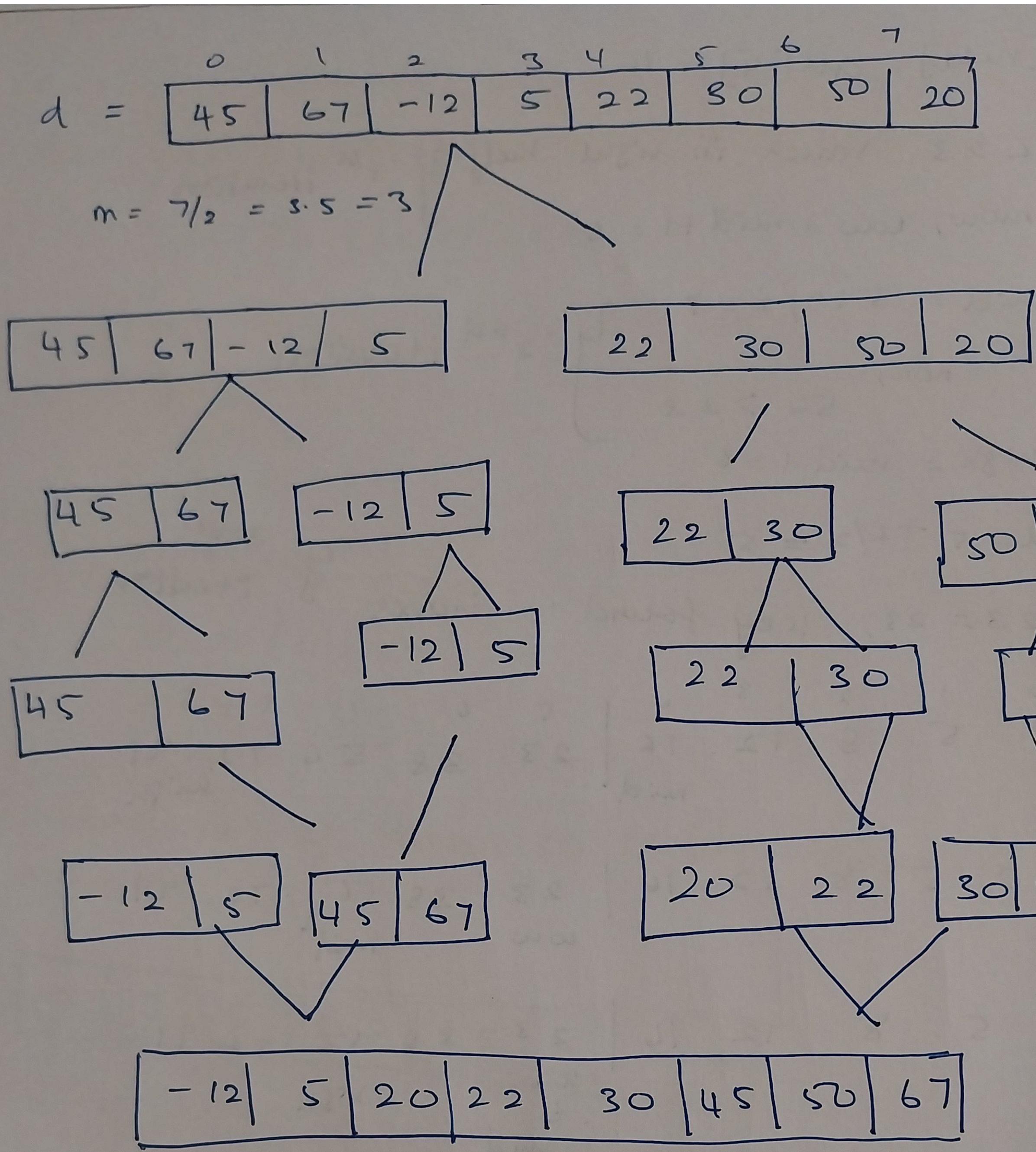
=> 2 5 8 12 16 | 23 38 56 72 91
low ↓ mid ... high

23 = 23.

13) Apply merge sort in order the list

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

d =	45	67	-12	5	22	30	50	20



Recurrence relation

$n = 1$ no comparisions

$T(1) = 0$.

for $n > 1$, merge sort.

$T(n/2) \Rightarrow$ so, $2T(n/2)$.

so, recurrence relation $\Rightarrow T(n) = 2T(n/2) + n$

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

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

$a=2,\ b=2,\ f(n)=n,\ k=1$

$\log_b^a = 1$

$\log_b a = k$

$p=0,\ 0 > k-1$

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

$= \theta(n^1 \log^1 n)$

$= \theta(n \log n)$

14) Find no. of times to perform swapping for selection sort.

12	7	5	-2	18	6	13	4

min

swap -2 4 12

-2	7	5	12	18	6	13	4

swap 4 4 7

min

-2	4	5	12	18	6	13	7

min

no swap

shift i

-2	4	5	12	18	6	13	7

-2	4	5	6	18	12	13	7

swap 7 & 18 min↑

-2	4	5	6	7	12	13	18

min↑ swf i

-2	4	5	6	7	12	13	18

↑min

-2	4	5	6	7	12	13	18

No, swap = 4.

time complexity :-

```
arr : array of items
 n = size
 for ( i = 0 ; i<n-1; i++)  - n
 min = i
   for ( j = i++ ; j < n ; j++)  - n x n
      if arr [j] < arr [min)  - 1
         min = j   - 1
          endif   - 1
      if (min! = i )
         swap arr [min] and arr [i]
         endif
         end for.
```

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

Swaps = 4.

15) Find index of target value 10 using binary search from the following list [2, 4, 6, 8, 10, 12, 14, 16, 18, 20].

Target = 10.

arr = [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
				mid					high

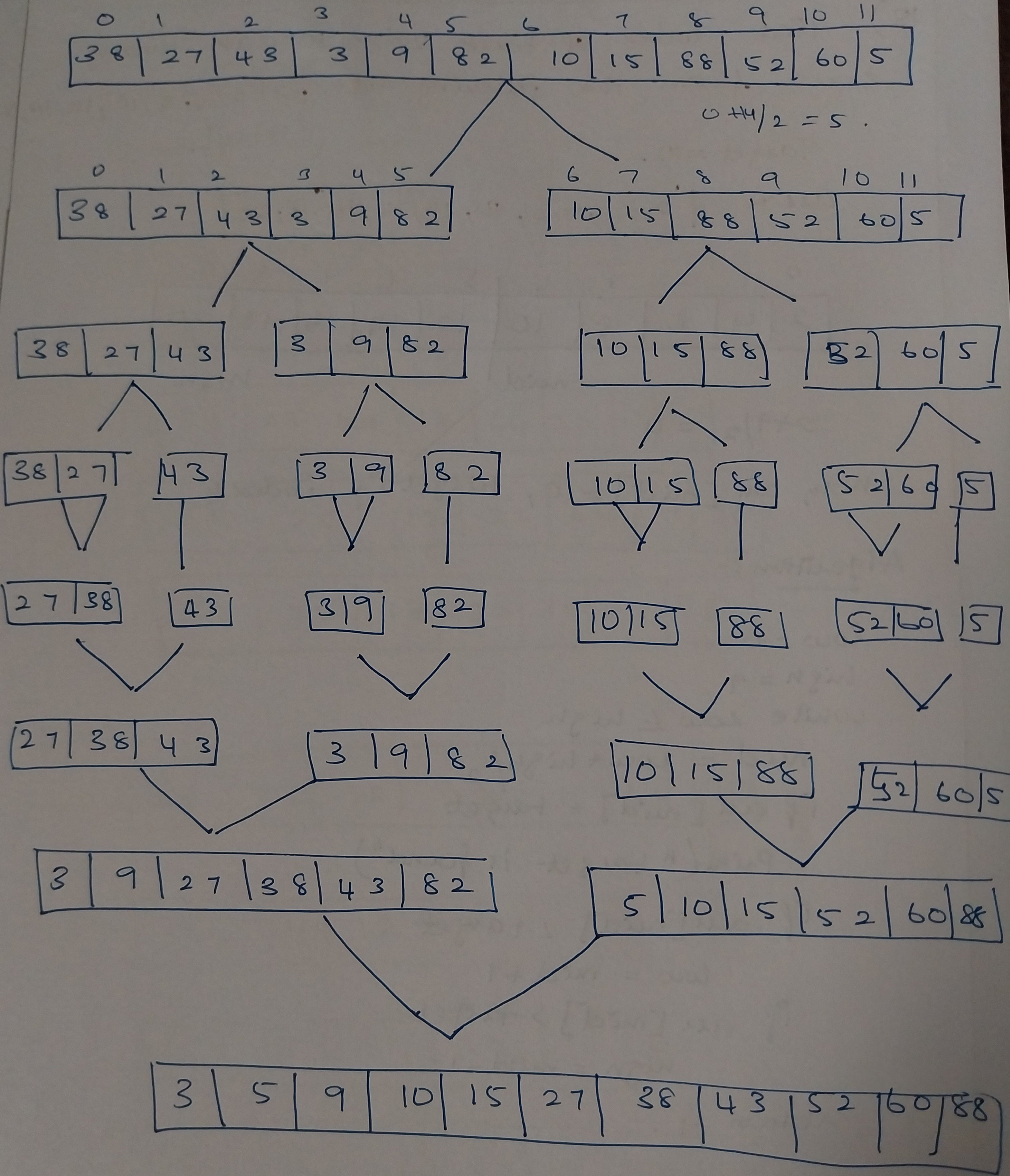
$0+9/2 = 4$

now, arr [mid] = 10, target is index 4.

Algorithm :-

```
low = 0
high = 9
while low ≤ high
  mid = low + high / 2
 if arr [mid] = target
   Print(" target is found")
 if arr [mid] < target
     low = mid + 1
 if arr [mid] > target
     high = mid - 1
return -1.
```

16) Sort the following array using merge sort divide & conquer the strategy [38, 27, 43, 3, 9, 82, 10, 15, 88, 52, 60, 5].



time complexity

$\therefore O(n \log n)$

17) Sort array using Bubble sort

64, 34, 25, 12, 22, 11, 90.

P1 :-

64	34	25	12	22	11	90

34	64	25	12	22	11	90

34	25	64	12	22	11	90

34	25 ~~64~~	12	64	22	11	90

34	25	12	22	64	11	90

34	25	12	22	11	64	90

– no swap

P2 :-

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

P-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

12	22	11	25	34	64	90

P-4

12	22	11	25	34	64	90

12	11	22	25	34	64	90

12	11	22	25	34	64	90

P-5

12	11	22	25	34	64	90

11	12	22	25	34	64	90

Sorted array :-

time complexity :-

Best cases :-	worst case :-	Average case :-
$O(n)$	$O(n^2)$	$O(n^2)$.

18) Sort the array 64, 25, 12, 22, 11 using selection sort, what is time complexity.

64	25	12	22	11

swap min

11 & 64

11	25	12	22	64

11	12	25	22	64

min swap 25 & 22

11	12	22	25	64

min

11	12	22	25	64

Best case : $O(n^2)$

worst case: $O(n^2)$

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

19) Sort the following elements using insertion sort using Brute force approach [38, 27, 43, 3, 9, 82, 10, 15, 88, 52, 60, 5] & Analyze $T(n)$.

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

swap

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	38	3	9	43	82	10	15	88	52	60	5

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

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

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

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

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

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

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

Sorted array :- 3, 5, 9, 10, 15, 27, 38, 43, 52, 60, 82, 88.

time complexity :-

insert sort [A]
for j = 2 to A-length-n
key = a[j] - 1
i = j-1 - 1
while i>0 & arr[i] > key - n×n
a[i+1] = a[i] - 1
i = i-1 - 1
a[i+1] = key - 1

$\Rightarrow O(n^2)$

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 elements using the insertion sort. Brute force.

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

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

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

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

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

↳ 5

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

11 -9] - 6

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

11 -9]

Sorted array:-

[-9, -8, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7,

8, 9, 10, 11]

Complexity:-

Best case -> $O(n)$

worst case -> $O(n^2)$ $n(n-1)/2$

Average case -> $O(n^2)$