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 the assertions.

For any four arbitrary real numbers, a_1, b_1 and a_2, b_2 such that $a_1 \le b_1$ and $a_2 \le b_2$.

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

$t(n) \in O(g_1(n))$, then there exists some constant c_1 and non negative integer n_1 such that $t_2(n) \le c_1 g_1(n)$ for $n \ge n_1$;

$t_2(n) \in O(g_2(n))$, then there exists some constants c_2 and non-negative integer n_2 such that $t_2(n) \le c_2 g_2(n)$ for all $n \ge n_2$.

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

$$t_1(n) + t_2(n) \le c_1 g_1(n) + c_2 g_2(n)$$
$$\le c_3 g_1(n) + c_3 g_2(n)$$
$$= c_3\{g_1(n) + g_2(n)\}$$
$$\le 2 c_3 \max\{g_1(n), g_2(n)\}.$$

2) Find the time complexity of the below recurrance equation:

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

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

$n = 2$

$$= 2T(2/2) + 1$$
$$= 2T + 1$$
$$= 3$$

$n=3$

$= 2T\left(\frac{3}{2}\right)+1$

$= 3+1 = 4$

$n=4$

$\Rightarrow 2T\left(\frac{4}{2}\right)+1$

$4+1$

$= 5$

$T(n) = 3+4+5+\ldots T(n)$

$\Rightarrow (2+n) \rightarrow O(n)$

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

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

$n=1$

$T(1) = 2T(1-1) = 0$

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

$n=3$ $T(3) = 2T(3-1) = 4$

$\Rightarrow 0+2+4+\ldots+T(n)$

$\frac{2n+2}{2} = 2\left(\frac{n+1}{2}\right)$

$O(n)$

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

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

$n > n_0$, $f(n) \leq c \cdot g(n)$

$g(n) = n^2$

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

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

divide by n^2

$\frac{n^2}{n^2} + \frac{3n}{n^2} + \frac{5}{n^2}$

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

$n \geq 1$

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

$c = 9$, $n_0 = 1$

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

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

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

$n \geq n_0$, $g(n) \geq c \cdot n(n)$

$n(n) = n^3$

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

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

divide with n^3

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

$1+\frac{2}{n}+\frac{4}{n^2}$ is close to 1

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

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

Determine whoreveer $n(n)=4n^2+3n$ is $\theta(n^2)$ or not.

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

$n \geq n_0$

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

$c=5 \quad n_0=1$

$4n^2+3n \leq 4n^2+3n \leq 4n^2+n^2=5n^2$

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

since, $n(n)=4n^2+3n$ is $O(n^2)$ and $\Omega(n^2)$,

so $n(n)=4n^2+3n$ is $\theta(n^2)$.

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

where $f(n)=\Omega(g(n))$ is ~~$4n^2+3$~~ true & false and justify your answer

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

$g(n)=n^2$

$n \geq n_0$

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

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

$n^3-2n^2+n \geq *c.n^2$

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

$n^3 > -2n^2 + n$

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

$n^3 + (-1-2)n^2 + n = n^3 - 3n^2 + n$

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

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

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

for $n \geq 3$, $n^2(n-3) \geq 0$

$\Rightarrow n^3 - 2n^2 + n \geq -1 \cdot n^2$

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

$\Rightarrow g(n) = -n^2$

True

9) Determine whether $n(n) = n\log n + n$ is $\theta(n\log n)$. prove a rigorous proof for your conclusion

$n(n) = n\log n + n$ is $\theta(n\log n)$

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

divide by $n\log n$ on both sides

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

$n \geq 2$, $\log(n) \geq 1$

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

$n\log n + n \leq an\log n$

$n(n) = n\log n + n$ is $O(n\log n)$

$n\log n + n \geq 1 \cdot n\log n$

$n(n) = n\log n + n$ is $\Omega(n\log n)$

since $n(n) = n\log n + n$ is both $O(n\log n)$ and $\Omega(n\log n)$

$n(n) = n\log n + n$ is $\Theta(n\log n)$.

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

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

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

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

case-1 : $f(n) = \Theta(n^c)$

$\Rightarrow T(n) = \Theta(n^{\log_a^b})$

case-2 : $f(n) = \Theta(n^{\log_b^a})$

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

case-3 : $f(n) = \Omega(n^c)$, $c > \log_b^a$

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

$f(n) = n^2$

$n^{\log_b^a} = n^2$

$f(n) = \Theta(n^{\log_b^a})$

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

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.

```
def find_max_min_product(arr):
    if len(arr) < 2:
        raise ValueError("Array must name atleast
                          two elements).
    arr.sort()
    max_product = max(arr[-1] * arr[-2], arr[0] * arr[1])
    min_product = min(arr[-1] * arr[0], arr[0] * arr[1])
    return max_product, min_product
arr = [4,-2,5,3,10,-5,2,8,-3,6,7,-4,1,9,-1,0,-6,-8,11,-9]
max_product, min_product = find_max_min_product(arr)
print(f"Maximum product: {max_product}")
print(f"Minimum product: {min_product}")
```

12) Determine Binary Search Method to search key = 23, form use arr[] = {2,5,8,12,16,23,38,56,72,91}.

→ start with two pointers 'low' and 'high'

low = 0, high = 9

$$mid = \frac{(low+high)}{2} = \frac{0+9}{2} = 4$$

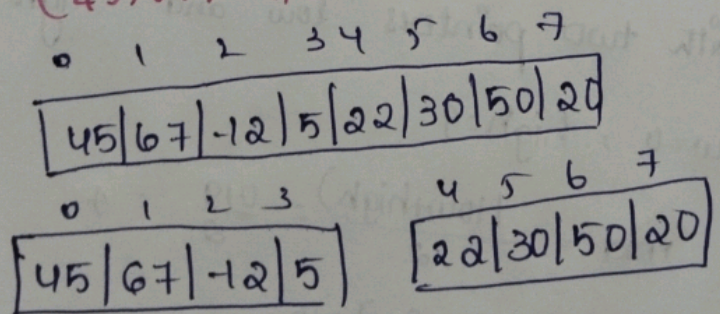
arr[4] = 16

code:

```
def binary_search(arr, key):
    low = 0
    high = len(arr) - 1
    while low <= high
        mid = (low + high) // 2
    if arr[mid] == key:
        return mid
    elif arr[mid] < key:
        low = mid + 1
    else:
        high = mid - 1
    return -1
arr = [2, 5, 8, 12, 16, 23, 38, 56, 72, 91]
key = 23
index = binary_search(arr, key)
print(f"key {key} found at index {index}").
```

13) Merge sort:

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



```
else:
    arr[k] = right_half[j]
    j+ = 1
    k+ = 1
while i<len(left_half):
    arr[k] = left_half
    i+ = 1
    k+ = 1
while j<len(right_half):
    arr[k] = right_half[j]
    j+ = 1
    k+ = 1
```

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

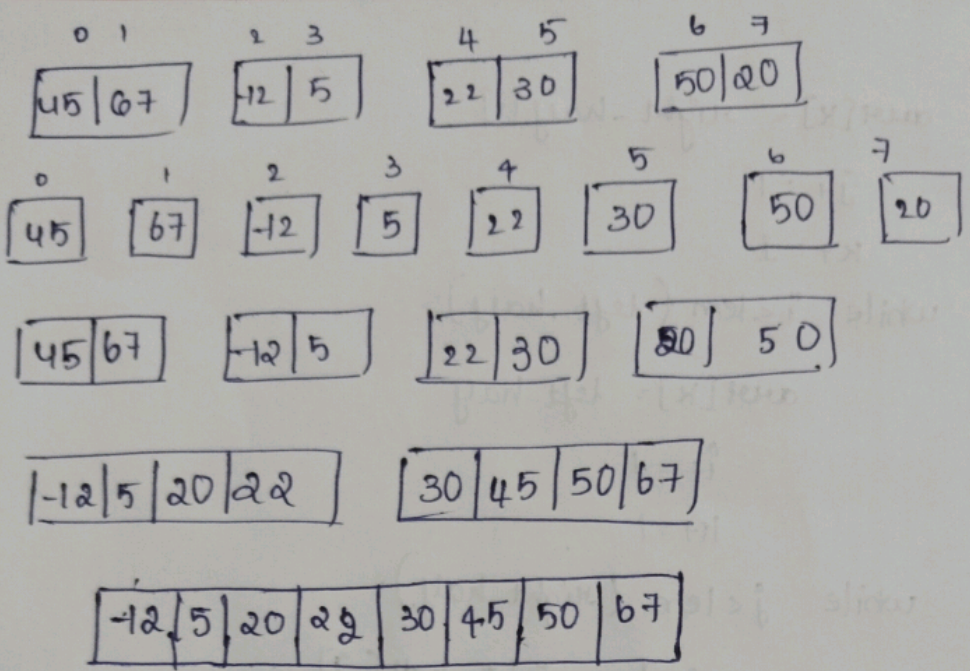
12	7	5	-2	18	6	13	4

-2	7	5	12	18	6	13	4

-2	4	5	12	18	6	13	7

-2	4	5	6	18	12	13	7

-2	4	5	6	7	12	13	18



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

code:

```
def merge_sort(arr):
    if len(arr) > 1:
        mid = len(arr)//2
        left_hay = arr[:mid]
        right_hay = arr[mid:]
        merge_sort(left_hay)
        merge_sort(right_hay)
        i = j = k = 0
    while i < len(left_hay) and j < len(right_hay):
        if left_hay[i] < right_hay[j]:
            arr[k] = left_hay[i]
            i += 1
```

| -2 | 4 | 5 | 6 | 7 | 12 | 13 | 18 | → sorted

⇒ n-1 ⇒ 8-1 = 7

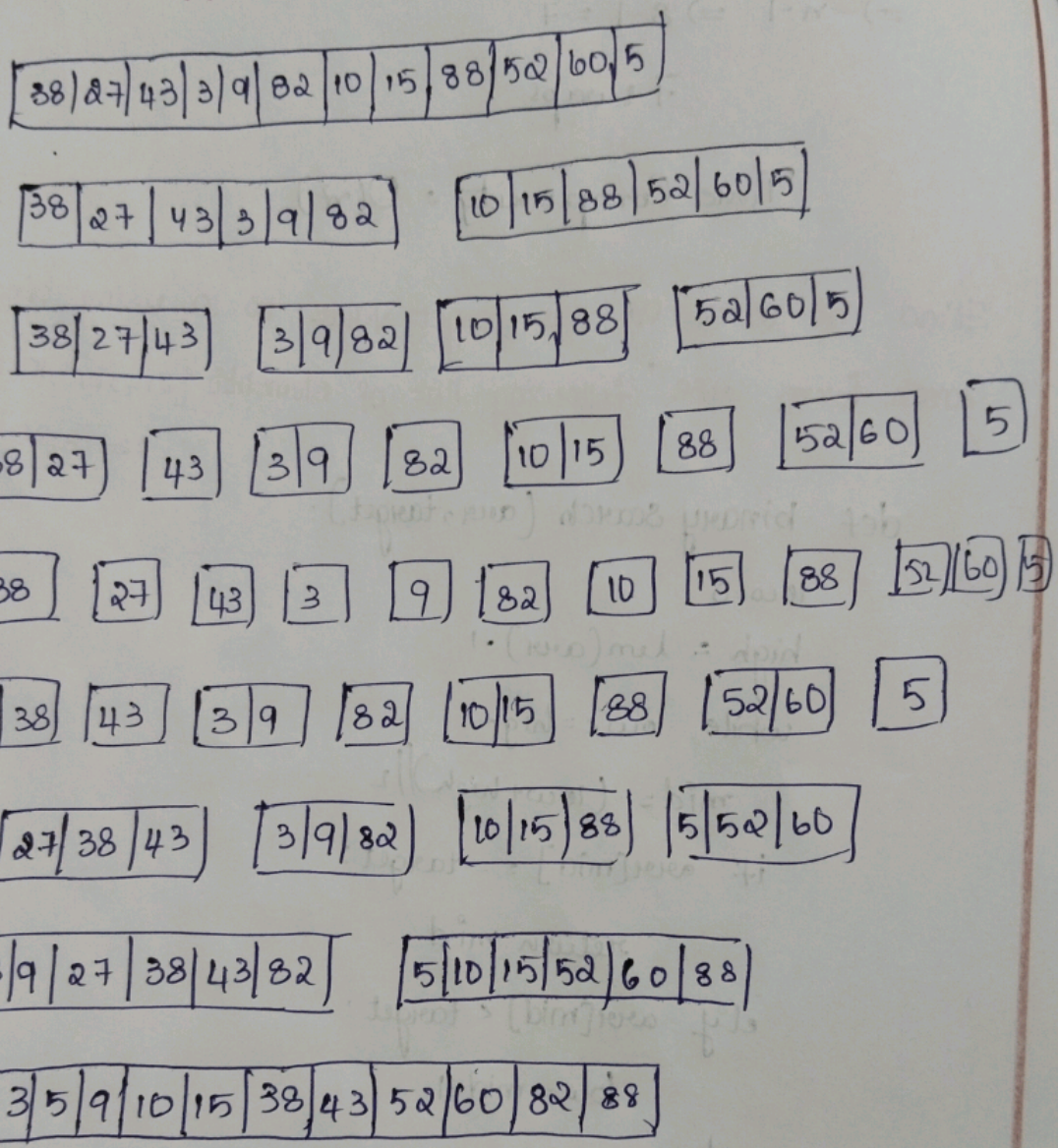
7 swaps

Time Complexity = O(n²)

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

```
def binary search (arr, target):
    low=0
    high = len(arr)-1
    while low<=high
        mid= (low+high)//2
    if arr[mid] == target:
        return mid
    elif arr[mid] < target:
        low = mid+1
    else:
        high = mid-1
    return -1
arr = [2,4,6,8,10,12,14,16,18,20]
target=10
index = binary search (arr, target).
```

16) Sort the following elements using Merge sort divide and conquer strategy [38, 27, 43, 3, 9, 82, 10, 15, 88, 52, 60, 5]



Algorithm:

```
Merge (arr, low, mid, high):
n1 = mid - low + 1
n2 = high - mid
left = new Array [n1]
Right[] = new Array [n2]
for i = 0 to n1 - 1:
    left [i] = arr [low + i]
for i = 0 to n2 - 1:
    Right [j] = arr [mid + 1 + j]
i = 0
j = 0
k = low
while i < n1 and j < n2:
    if left [i] <= Right [j]:
        arr [k] = left [i]
        i = i + 1
    else:
        arr [k] = right [j]
        j = j + 1
    k = k + 1
while i < n2:
    arr [k] = Right [j]
    j = j + 1
    k = k + 1
```

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

| 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 |

| 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 |

| 12 | 25 | 22 | 11 | 34 | 64 | 90 |

| 12 | 22 | 25 | 11 | 34 | 64 | 90 |

| 12 | 22 | 11 | 25 | 34 | 64 | 90 |

11	12	22	23	34	64	90

worst case: $n \times (n-1)/2$

best case: $n-1$

Average case: $O(n-1)$

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

64	25	12	22	11

11	25	10	22	64

11	12	25	22	64

11	12	22	25	64

→ sorted

Time Complexity:

worst case: $O(n^2)$

$n(n-1)/2$

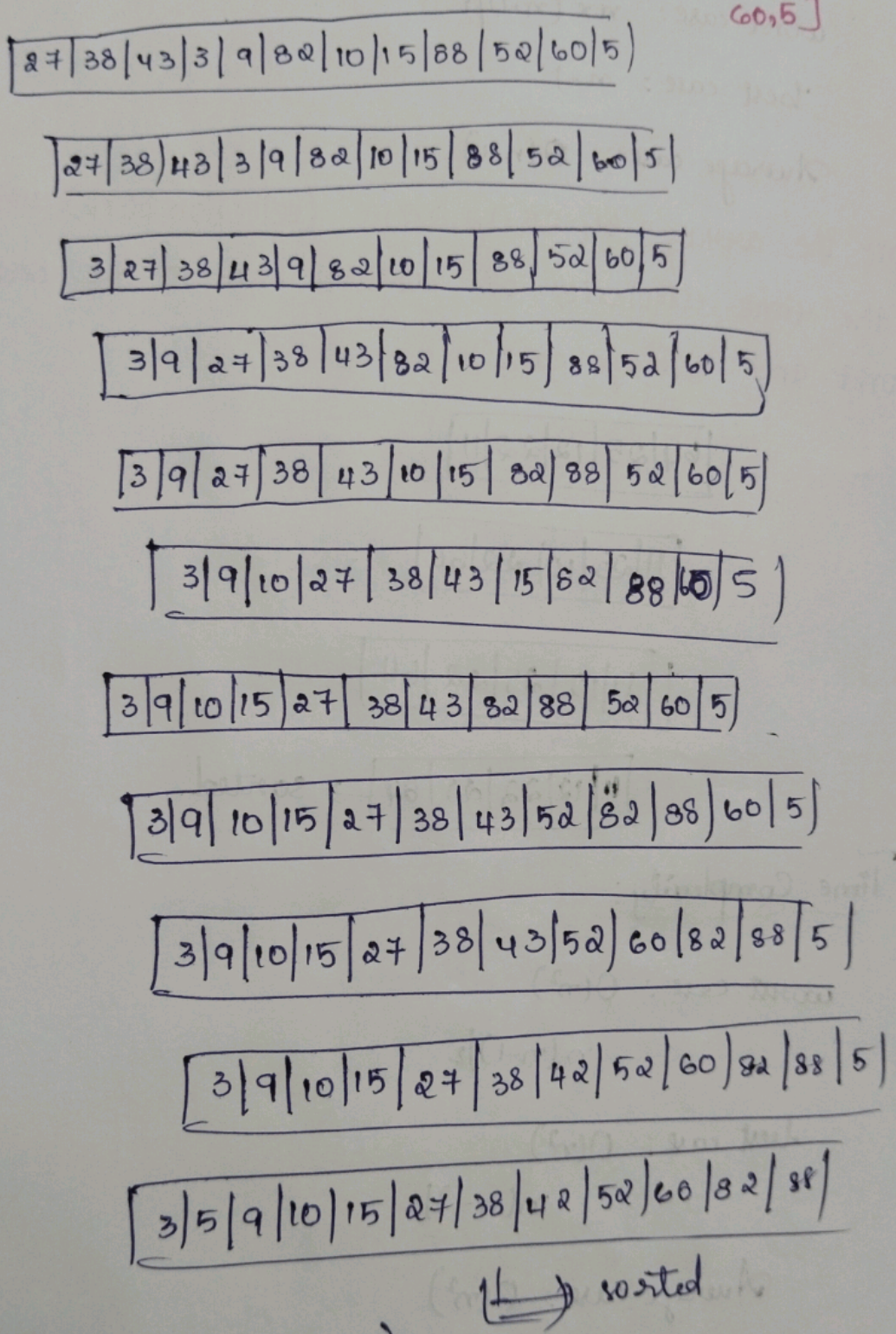
best case: $O(n^2)$

$n(n-1)/2$

Average case: $O(n^2)$

$O(n^2)$

19) Sort the following elements using insertion sort using BruteForce Approach strategy. [38, 27, 43, 3, 9, 82, 10, 15, 88, 52, 60, 5]



Time Complexity = $O(n^2)$

Space Complexity: $O(1)$

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 using insertion sort (Brute force Approach).

[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, 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]

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

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

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

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

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

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

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

↳ sorted

Time Complexity => $O(n^2)$

Space Complexity => $O(1)$