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

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

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

$f(n) \leq c(g_1(n))$, we write it as

$\boxed{t_1(n) \leq c g_1(n)}$ where $c_1 \rightarrow$ some constant

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

$f(n) \leq c \cdot g_2(n)$ modify as,

$\boxed{t_2(n) \leq c_2 g_2(n)}$ where $c_2 \rightarrow$ some constant

$c_3 = \max\{c_1, c_2\}$

$t_1(n) + t_2(n) \leq c_1 g_1(n) + c_2 g_2(n)$

$= c_3 g_1(n) + c_3 g_2(n)$

$= c_3 \{g_1(n) + g_2(n)\}$

$\leq 2 c_3 \max\{g_1(n), g_2(n)\}$

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

$\boxed{\text{Thus Proved.}}$

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

$$T(n) = 2T\left(n/2\right) + 2^0 \rightarrow (1)$$

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

$$T\left(n/2\right) = 2T\left(\frac{n}{2^2}\right)+1 \rightarrow (2)$$

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

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

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

Substitute (2) in (1)

$$T(n) = 2\left[2T\left(\frac{n}{2^2}\right)+1\right]+1$$

$$= 4T\left(\frac{n}{2^2}\right)+2+1$$

$$T(n) = 2^2T\left(\frac{n}{2^2}\right)+2^1+2^0 \rightarrow (4)$$

Substitute (4) in (3)

$$T(n) = 4\left[2T\left(\frac{n}{2^3}\right)+1\right]+3$$

$$T(n) = 2^3T\left(\frac{n}{2^3}\right)+2^2+2^1+2^0+2$$

$$T(n) = \left[2\left(\frac{n}{2}\right)+1\right] + \left[2^2\left(\frac{n}{2^2}\right)+3\right] + \left[2^3\left(\frac{n}{2^3}\right)+7\right] + \cdots$$

$$T(n) = 2^k + \left(\frac{n}{2^k}\right) + \left(2^{k-1}+2^{k-2}\right) + 2^3+2^2+2^1+2^0$$

$$\frac{n}{2^k} = 1$$

$$n = 2^k$$

$$\log n = \log 2^k$$

$$\boxed{\log n = k}$$

$$T(n) = 2^k T\left(\frac{n}{2^k}\right) + \left(2^k + 2^{k-1}\right)$$

$$= 2^k T(1) + \left(2^k + 2^{k-1}\right)$$

$$= n\,T(1) + n + \left(2^{k-1}\right)$$

$$\Rightarrow 2n + \left(2^{k-1}\right) = 2n + \frac{2^k}{2^1}$$

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

$$= \frac{1}{2}(3n)$$

$$\boxed{O(n)}$$

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

$$\boxed{T(n) = 2T(n-1)} \rightarrow (1)$$

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

$$\boxed{T(n-1) = 2T(n-2)} \rightarrow (2)$$

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

$$\boxed{T(n-2) = 2T(n-3)} \rightarrow (3)$$

Substitute ② in ①

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

$$\boxed{T(n) = 2^2 T(n-2)} \rightarrow ④$$

Substitute ③ in ④

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

$$\boxed{T(n) = 2^3 T(n-3)}$$

$$T(n) = 2(n-1) + 2^2(n-2) + 2^3(n-3) + \cdots$$

$$\boxed{T(n) = 2^K T(n-K)}$$

$n - K = 1$

$\boxed{n = K}$

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

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

$$= 2^n (1)$$

$$\boxed{O(2^n)}$$

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

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

$$\boxed{f(n) \leq c\,g(n)}$$

$$\boxed{n^2 + 3n + 5 \leq c_1 n^2} \rightarrow ①$$

divide both sides by n^2

$$\frac{n^2}{n^2} + \frac{3n}{n^2} + \frac{5}{n^2} \leq c_1 \frac{n^2}{n^2}$$

$$1 + \frac{3}{n} + \frac{5}{n^2} \leq c_1$$

when $n=1$,

$1 + 3 + 5 \leq c_1$

$c_1 \geq 9$

Substitute $c_1 \geq 9$ in equation (1)

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

$c_1 = 6$

$n_0 = 1$

when $n = 1$,

$1 + 3 + 5 \leq 9$

$9 \leq 9$

$\therefore n \geq 1$ we prove that n^2+3n+5 becomes $O(n^2)$

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

Thus Proved

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

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

rewrite it as,

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

$f(n) \leq cg(n)$

$n^3 + 2n^2 + 4n \geq cn^3$ → (1)

divide both sides by n^3

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

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

when $n=1$,

$$1 + \frac{2}{1} + \frac{4}{1} \geq C$$

$$\boxed{7 \geq C}$$

Substitute $7 \geq C$ in equation (1)

$$n^3 + 2n^2 + 4n \geq 7n^3$$

when $n=1$

$$1 + 2 + 4 \geq 7$$

$$\boxed{7 \geq 7}$$

$\therefore C \geq 7, n_0 = 1$

$\therefore n \geq 1$, we prove that $n^3 + 2n^2 + 4$ is $\Omega(n^3)$

7) Big Θ Notation: Determine whether $h(n) = 4n^2 + 3n$ is $\Theta(n^2)$ or not

rewrite it as,

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

$f(n) \leq c_1 g(n)$

$4n^2 + 3n \leq c_1 n^2$ → (1)

divide both sides by n^2

$$\frac{4n^2}{n^2} + \frac{3n}{n^2} \leq \frac{c_1 n^2}{n^2}$$

$$4 + \frac{3}{n} \leq c_1$$

when $n = 1$

$4 + 3 \leq c_1$

$7 \leq c_1$

Substitute in eqn (1)

$4n^2 + 3n \leq 7n^2$

when $n = 1$

$4 + 3 \leq 7(1)$

$7 \leq 7$

$c_1 \geq 7$, $n_0 \geq 1$

$c_1 \geq 7$, $n_0 \geq 1$

$f(n) \leq c_2 g(n)$

$4n^2 + 3n \geq c_2 n^2$ → (1)

divide both sides by n^2

$$\frac{4n^2}{n^2} + \frac{3n}{n^2} \geq \frac{c_2 n^2}{n^2}$$

$$4 + \frac{3}{n} \geq c_2$$

when $n = 1$

$4 + 3 \geq c_2$

$7 \geq c_2$

so, $c_2 = 1$

Substitute in eqn (2)

$4n^2 + 3n \geq (1)n^2$

when $n = 1$

$4 + 3 \geq 1$

$7 \geq 1$

$c_2 = 1$, $n_0 \geq 1$

$c_2(g(n)) \leq f(n) \leq c_1 g(n)$

∴ The condition is true.

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

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

$$\boxed{n^3 - 2n^2 + n \geq cn^2} \rightarrow (1)$$

divide both sides by n^2

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

$$\boxed{n - 2 + \frac{1}{n} \geq c}$$

when $n = 1$,

$$1 - 2 + 1 \geq c$$

$$0 \geq c$$

when $n = 2$,

$$2 - 2 + \frac{1}{2} \geq c$$

$$\boxed{0.5 \geq c}$$

Substitute in equation (1)

$$n^3 - 2n^2 + n \geq (0.5)\, n^2$$

when $n = 1$

$$1 - 2 + 1 \geq 0.5$$

$$\boxed{0 \geq 0.5}$$

when $n = 2$,

$$8 - 8 + 2 \geq (0.5)4$$

$$\boxed{2 \geq 2}$$

$c \leq 0.5$, $n_0 = 2$, $n \geq 2$

$\therefore$ It is true

a) Determine whether $h(n) = n \log n + n$ is in $\Theta(n \log n)$ prove a rigorous proof

1) Upper Bound:

We need to find c_1 and n_0 such that

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

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

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

$$= 2 n \log n.$$

when $c_1 = 2$

then $h(n) \leq 2n \log n$ for all $n \geq 1$

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

2) Lower Bound

We need to find c_2 and n_0 such that

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

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

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

let $c_2 = \frac{1}{2}$ $h(n) \geq \frac{1}{2} n \log n$

$h(n)$ is $\Omega(n \log n)$

3) Combining,

$h(n) = n \log n + n$ is in $\Theta(n \log n)$

(6) Solve the following recurrence relation.

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

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

$$T(n/2) = 4T\left(\frac{n/2}{2}\right) + \left(\frac{n}{2}\right)^2$$

$$T(n/2) = 2^2 T\left(\frac{n}{2^2}\right) + \left(\frac{n}{2}\right)^2 \rightarrow (2)$$

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

$$T\left(\frac{n}{2^2}\right) = 2^2 T\left(\frac{n}{2^3}\right) + \left(\frac{n}{2^2}\right)^2 \rightarrow (3)$$

$$T(n) = 2^2\left[2^2 T\left(\frac{n}{2^2}\right) + \left(\frac{n}{2}\right)^2\right] + n^2$$

$$= 2^4 T\left(\frac{n}{2^2}\right) + 2^2\left(\frac{n}{2}\right)^2 + n^2$$

$$= 2^4 T\left(\frac{n}{2^2}\right) + n^2$$

$$T(n) = 2^4 T\left(\frac{n}{2^2}\right) + 2n^2 \rightarrow (4)$$

$$T(n) = 2^6 T\left(\frac{n}{2^3}\right) + 3n^2$$

$$T(n) = 2^{k+2} T\left(\frac{n}{2^k}\right) + kn^2$$

$\frac{n}{2^k} = 1$

$n = 2^k$

$\log n = \log 2^k$

$\boxed{\log n = k}$

$T(n) = 2^{k+2} T\left(\frac{n}{2^k}\right) + kn^2$

$= 2^{k+2} T(1) + kn^2$

$= 2^k \cdot 2^2 (1) + (\log n) n^2$

$= 4n + n^2 \log n$

$= O(n \log n)$

$\boxed{O(n \log n)}$

ii) array of [4, -2, 5, 3, 10, -5, 2, 8, -3, 6, 7, -4, 1, 9, -1, 0, -6, -8, 11, -4] find the max and min product that can be obtained by multiplying 2 int from array

```
def find_m(arr):
    if len(arr) < 2:
        raise ValueError("Array atleast contain 2 element")
    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, -4]
max_product, min_product = find_m(arr)
print(find_m)
```

12) Demonstrate Binary Search method to search key=23 from the arr = [2,5,8,12,16,23,38,56,72,91]

→ Start with 2 pointers 'low' and 'high'

low = 0, high = 9, key = 23

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

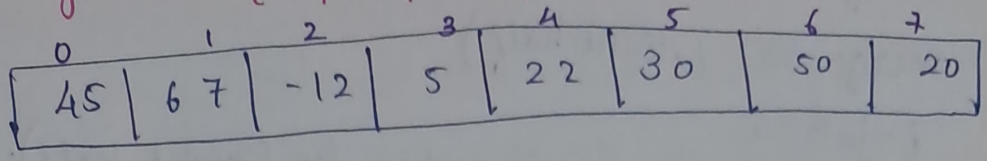
$$= \frac{0+9}{2} = 4$$

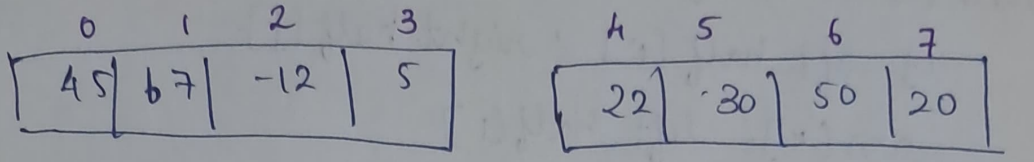
arr[4] = 16

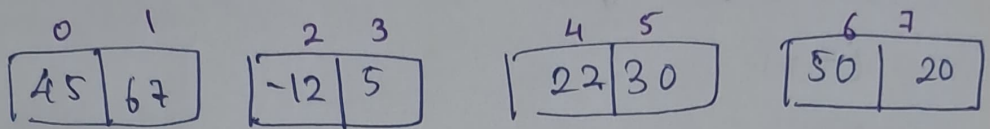
Code:

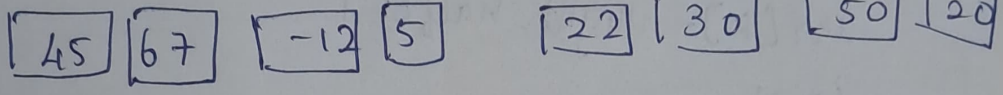
```
def binary-search (arr, key)
    low = 0
    high = len(arr) - 1
    while (low <= high):
        mid = (low+high)//2
        if arr[mid] == key:
            return mid
        elif arr[mid] < key:
            low = mid + 1
        else: high = mid - 1
    return -1
arr = [2,5,8,12,16,23,38,56,72,91]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

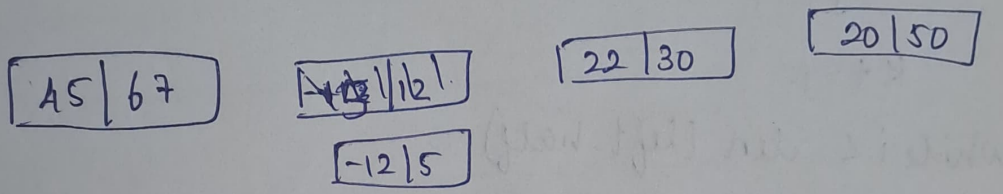
13) Merge Sort : (45, 67, -12, 5, 22, 30, 50, 20)

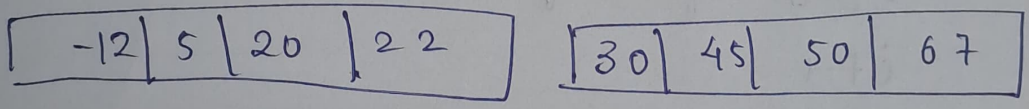


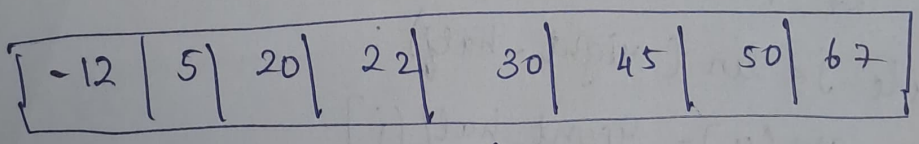












$\Rightarrow T(n) = 2T\left(\frac{n}{2}\right) + (n-1)$

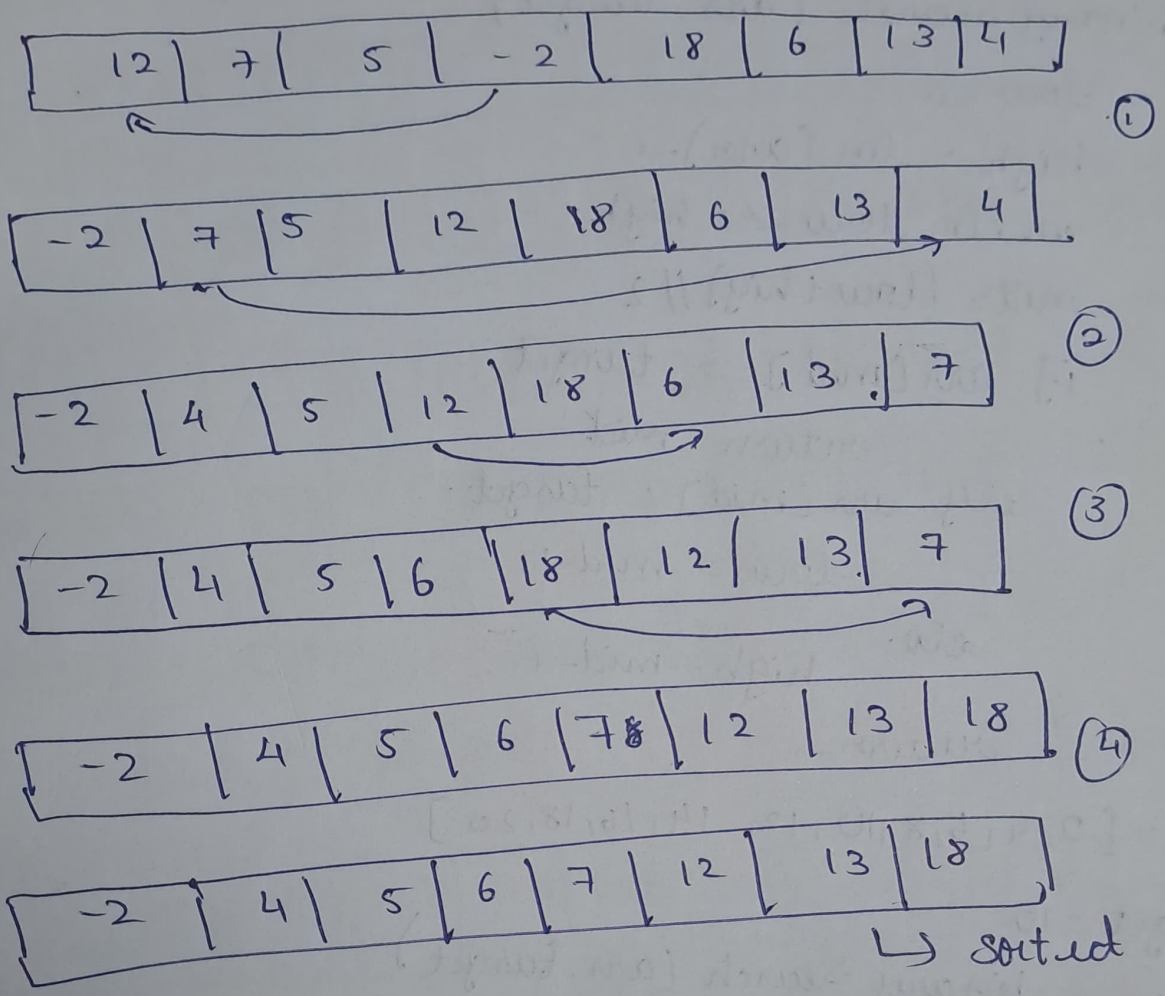
Code:

```
def merge_sort(arr):
    if len(arr) > 1:
        mid = len(arr)//2
        left_half = arr[:mid]
        right_half = arr[mid:]
```

```
    merge_sort (left half)
    merge_sort (right half)
     i=j=k=0
while i< len [ left_half) and j< len (right_half):
    if left_half [i] < right_half [j]:
    arr [k]= left_half [i]
     i+=1
    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) No. of times to perform selection sort and estimate time complexity

=) 12, 7, 5, -2, 18, 6, 13, 4



=) n-1 =) 8-1 = 7

7 swaps

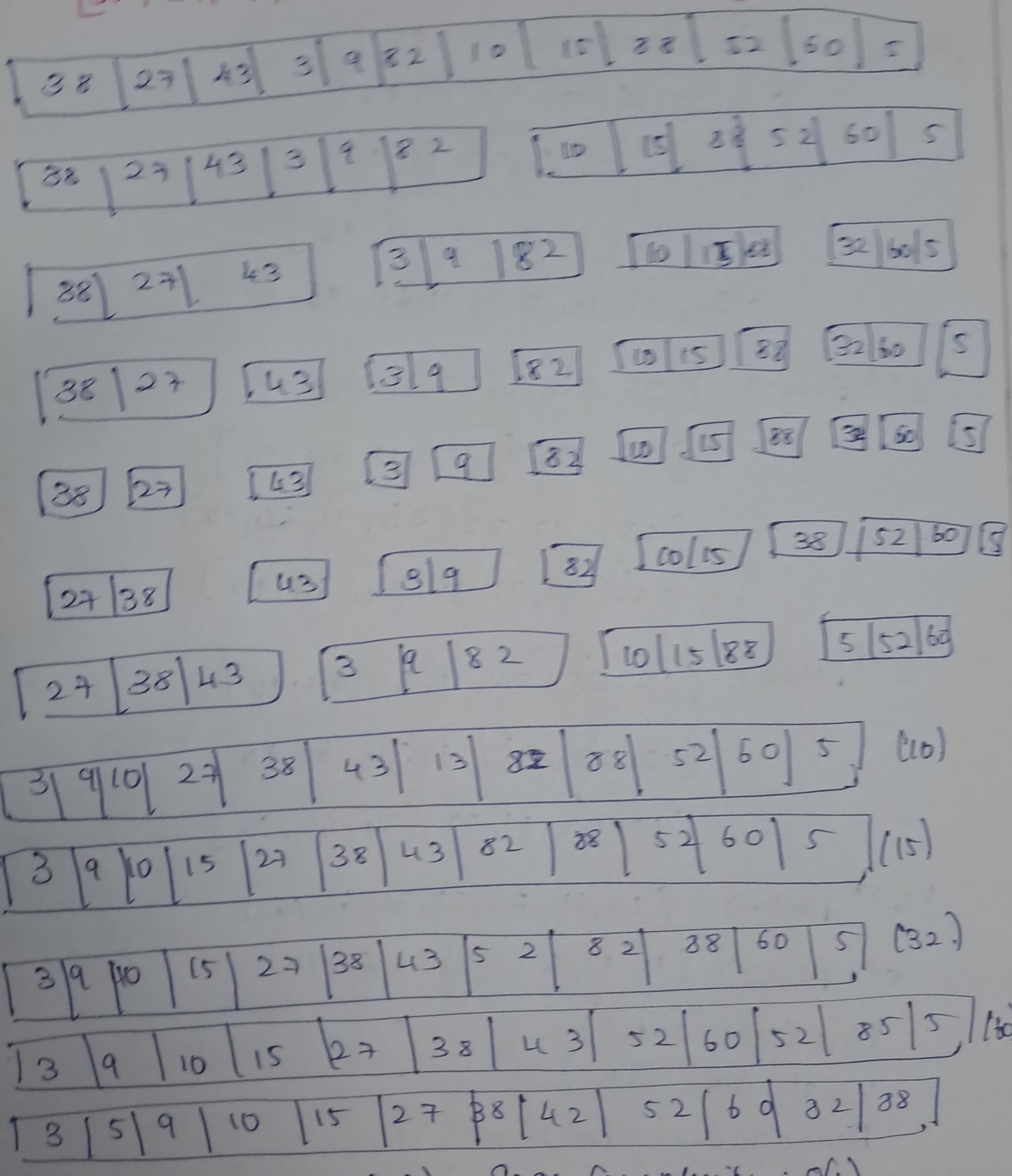
⇒ Time complexity = $O(n^2)$

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

```
def binary Search (arr, target)
    low = 0
    high = len (arr) - 1
    while low <= high
    mid = (low + hig) // 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)
```

6) Merge Sort, divide and conquer strategy,

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



Time complexity: $O(n^2)$, Space Complexity: $O(1)$

17) Sort the array 64, 34, 25, 12, 22, 11, 90 using babble Sort. What is the time complexity

1st pass => [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]

2nd pass => [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]

3rd pass => [12, 25, 22, 11, 34, 64, 90]

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

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

4th pass => [12, 11, 22, 25, 34, 64, 90]

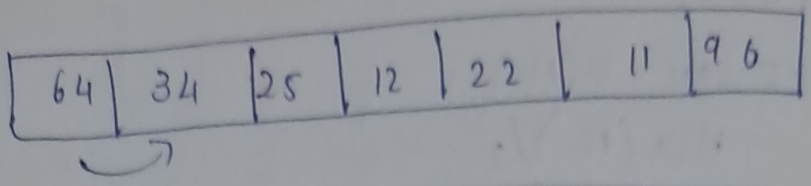
5th pass => [11, 12, 22, 25, 34, 64, 90]

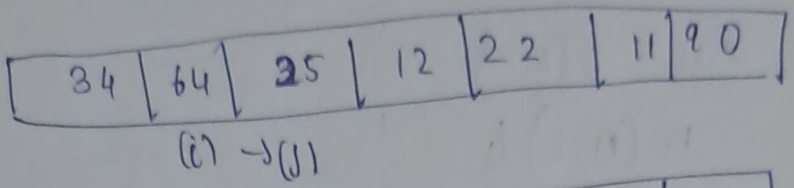
Best case: $O(n^2)$

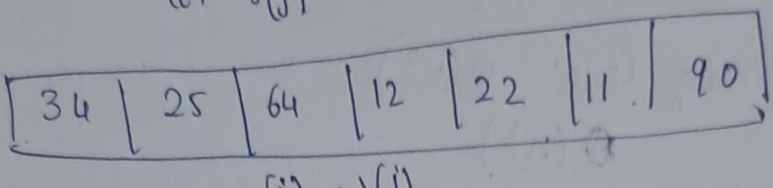
Worst case: $O(n^2)$

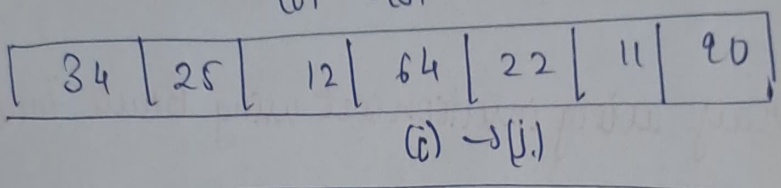
Average case: $O(n^2)$

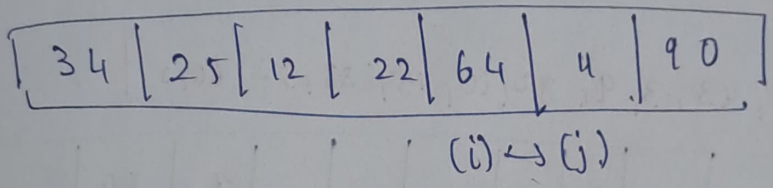
18). Sort the array 64, 37, 25, 12, 22, 11, 90 using Bubble Sort. What is the time complexity of solutions

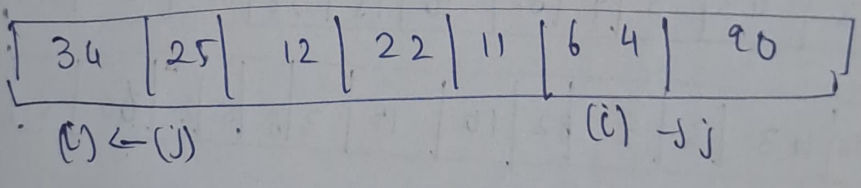


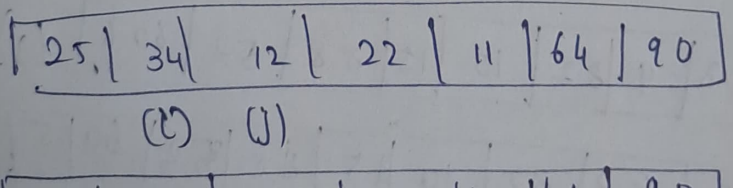


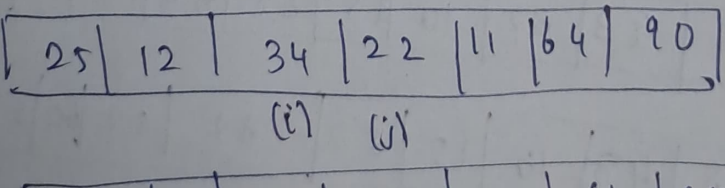


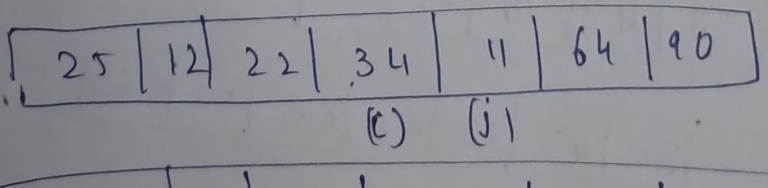


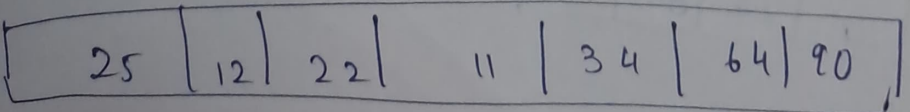












the array.

11	12	22	25	64

→ sorted

Time Complexity:

Worst Case: $O(n_2)$

$n \times (n-1)/2$

Best case: $O(n^2)$

$n(n-1)/2$

Average Case: $O(n^2)$

$O(n^2)$

19) sort the array using insertion sort using Brute force.

Approach

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

27	38	43	3	9	82	10	15	28	52	60	5	(38)

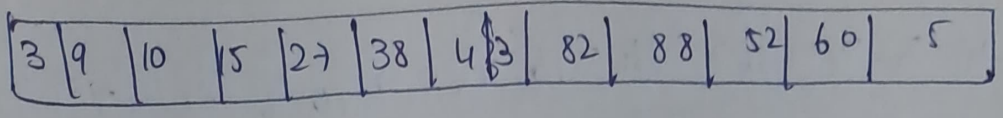
27	38	43	3	9	82	10	15	3	8	5	2	60	5	(27) (43)

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

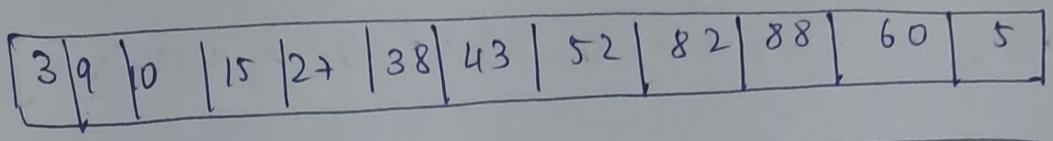
3	9	27	38	43	82	10	15	88	52	60	5	(9)

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

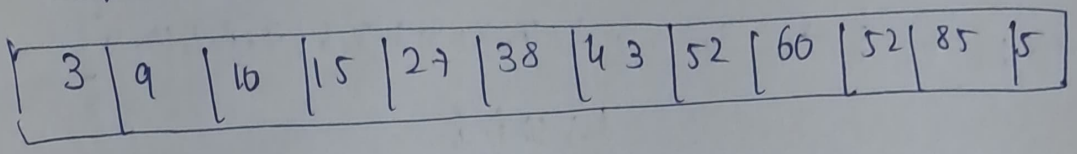
3	9	10	27	38	43	13	82	88	52	60	5	(10)



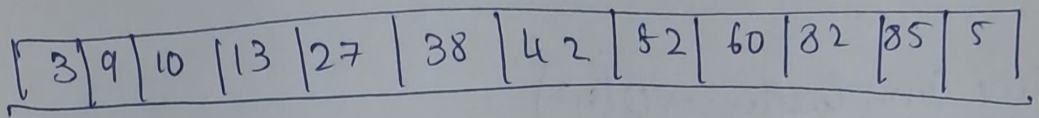
(15)

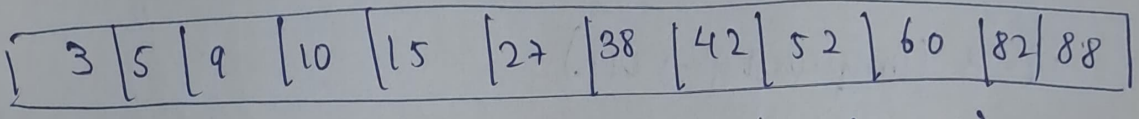


(32)



(60)





↳ (sorted)

Time complexity: $O(n^2)$

Space Complexity: $O(1)$

20) Given an arr sort using insertion sort

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

[-5, -2, 3, 4, 5, 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, -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, 7, 8, 10, -4, 1, 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, 0, 2, 3, 4, 5, 6, 7, 8, 9, 10, -6, -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 => $O(1)$