CSA-0666

DAA - Design Analysis of Algorithm.

M. DHARANI
192324008
B.tech AI-DS.

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

$t_1(n) \leq c\, g_1(n)$ we write it as

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

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

$t_2(n) \leq c\, g_2(n)$ modify as

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

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

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

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

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

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

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

proved//

3) Find the time complexity of below recurrance equation

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

By master theorem.

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

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

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

Since, $\log_b^a > k$, $T(n) = O(n \log_b^a)$

$$= O(n^1)$$

$$T(n) = O(n)$$

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

Given,

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

Sub $n = n-1$

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

Sub (2) in (1)

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

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

Sub $n = n-1$

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

Sub (4) in (3)

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

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

$\vdots$

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

Stop when $n=k = n-k=0$

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

Given $T(0) = 1$

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

$= 2^n$

time complexity = $O(2^n)$

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

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

Since $O(n^2)$ needed to prove,

$f(n) \leq C \cdot n^2$

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

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

for $n > 1$,

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

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

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

now, $c = 9$, $n_0 = 2$

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

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

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

return it as.

$$f(n) = n3 + 2n^2 + 4n \quad g(n) = n3$$

$$\boxed{f(n) \leq cg(n)}$$

$$n3 + 2n^2 + 4n \geq cn3 \rightarrow (1)$$

divide both sides by n^3

$$\frac{n3}{n^3} + \frac{2n3}{n^3} + \frac{4n}{n3} \geq \frac{cn3}{n3}$$

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

when $n = 1$,

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

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

Substitute $7 \geq c$ in eqn (1)

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

when $n = 1$

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

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

$c \geq 7 \cdot n_0 = 1$

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

7) Big theta notation: determine whether $h(n) = 4n^2 + 3n$ is $\Theta(n^2)$ or not:-

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

condition is $c_1 g(n) \leq 4 + (n) \leq c_2 g(n)$

So, we need to show,

$$c_1 n^2 \leq 4n^2 + 3n \leq c_2 n^2$$

upper Bound $O(n^2)$

$$4n^2 + 3n \leq c_2 n^2$$

for $n \geq 1$, $4n^2 + 3n \leq 4n^2 + 3n^2 = 7n^2$

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

lower Bound (Ωn^2)

$$4n^2 + 3n \leq c_2 n^2$$

$n \geq 1$, $4n^2 + 3n \leq 4n^2 + 3n^2 = 7n^2$

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

$4n^2 + 3n \geq 4n^2$ $(c_1 = 4)$

$\therefore$ $h(n) = 4n^2 + 3n$ is $\Theta(n^2)$ with $C_1 = 4$, $C_2 = 7$, n_0

$\therefore h(n) = 4n^2 + 3n \in O(n^2)$

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.

Given.

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

$g(n) = n^2$

for $\Omega(g(n))$ $f(n) \geq C \cdot g(n)$

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

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

$n = 2$, $8 - 8 + 8 \geq 1 + 4$

$8 \geq 4$.

So, for $n \geq 2$ $f(n) = \Omega(g(n))$ is true since it satisfies the omega condition.

9) Determine whether $h(n) = n \log n + n$ is n $O(n \log n)$ prove a rigorous proof.

1) upper bound

$h(n) \leq C_1 \cdot n \log n$ and n_0 such that

$h(n) \leq C_1 n \log n$ for all $n \geq n_0$

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

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

$= 2n\log n$

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

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

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

$\geq \frac{1}{2} n\log n$ for $n > 2$

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

~~$\geq 1/2\ n\log n$ for $n \geq 2$~~

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

3) combining

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

is in $\Theta(n\log n)$

10) Solve the following recurrance relation

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

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

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

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

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

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

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

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

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

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

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

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

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

$n = 2^k$

$\log n = \log 2^k$

$\log n = k$

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

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

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

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

$= O(n \log n)$

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

1) array of [4, -2, 5, 3, 10, -5, 2, 8, -3, 6, 7, -4, 1, 9, -1, 0, -6, -8, 11, -9] 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 elements")
    arr.sort()
    maxproduct = max(arr[-1] * arr[-2], arr[0] * arr[1])
    minproduct = 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, 14, -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

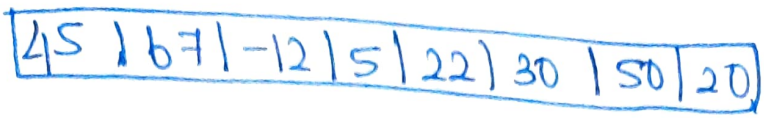
$$mid = \frac{low + high}{2} = \frac{0+9}{2} = 4$$

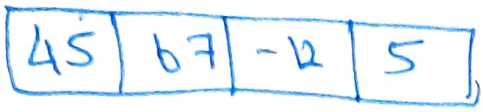
arr[4] = 16

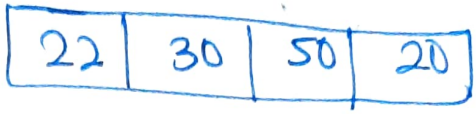
code.

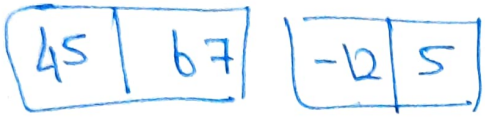
```
def binary_search (arr, key)
    low=0
    high = len(arr)-1
while (low ≤ high):
    mid =(low+high) //2
    if arr [mid] == key
        return mid
    elif arr [mid] +1
    el     low=mid +1
    else :
        high =mid -1
    return -1
arr = [2, 5, 8, 12, 16, 23, 38, 56, 72, 91]
key =23.
index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

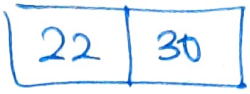
(3) mergesort (45, 67, -12, 5, 22, 30, 50, 20)

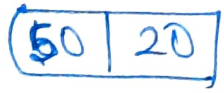


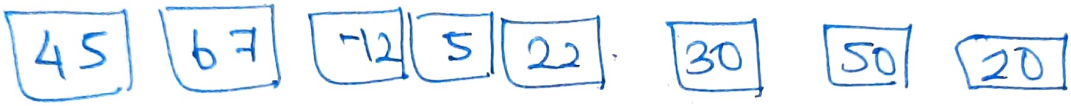


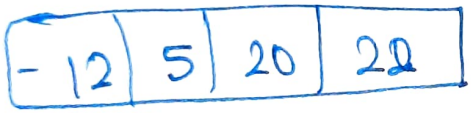


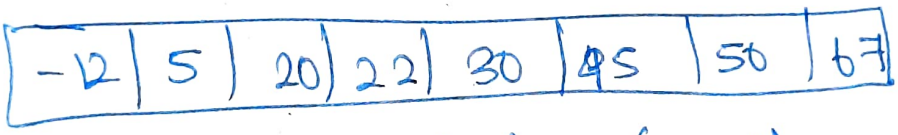












$= T(n) = 2T\left(\frac{n}{2}\right) + (n-1)$

recurrance relation T(n)

for n = 1 no comparison

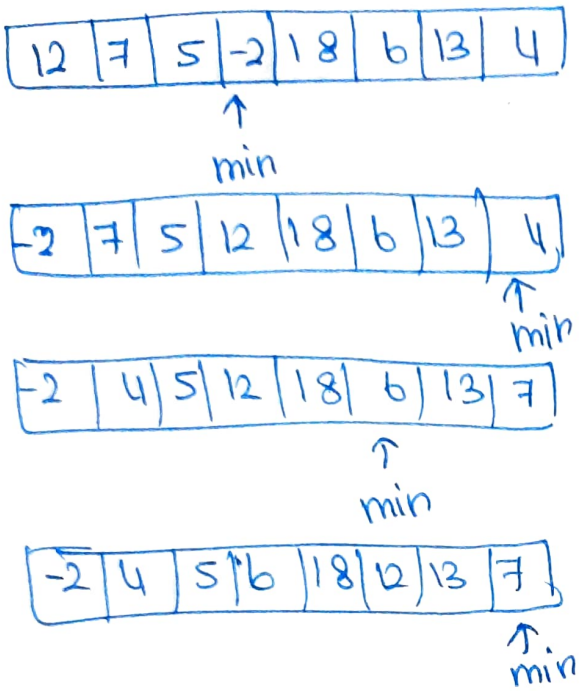
n > 1 mergeSort list to two values of n/2

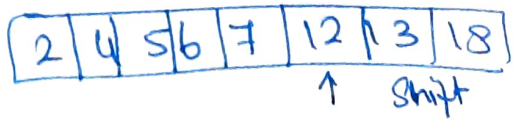
T(n/2) So, 2T (n/2)

$$T(n) = 2T\left(n/2\right) + n$$

14) find no of times to perform swapping for selection sort also estimate time complexity for order of notation

Set 8 (12, 7, 5, -2, 18, 6, 13, 4)





⇒ Sorted

So, min swap = 4.

time complexity = n-1

8-1 = 7.

$O(n^2)$.

15) Find the index of the target value 10 using binary search from the following list of elements

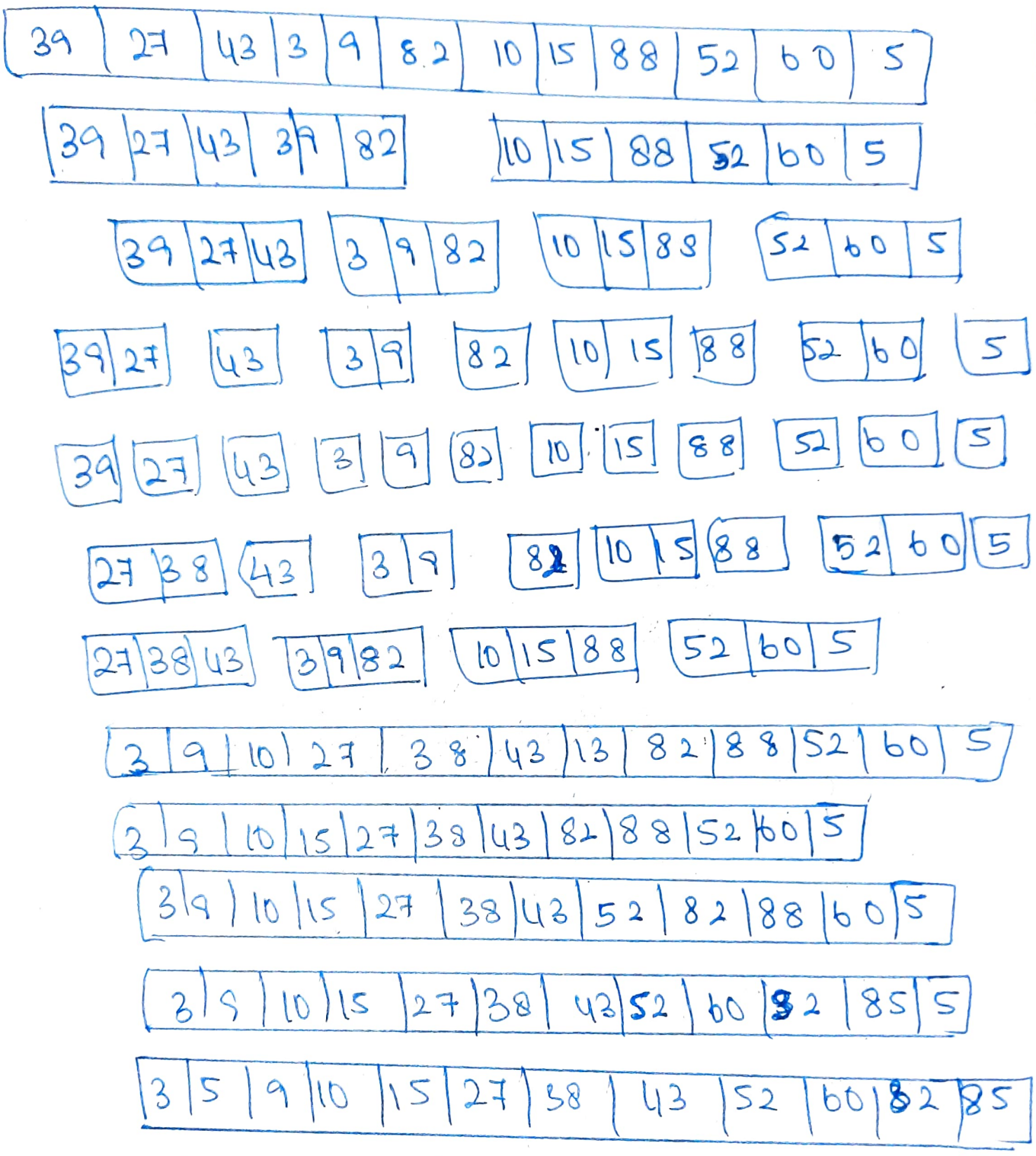
[2, 4, 6, 8, 10, 2, 14, 16, 18, 20]

```
def binarySearch(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 = [ ]
target = 10
index = BinarySearch
        (arr, target)
```

merge sort, divide and conquer method.

39, 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 bubble sort.

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, 14, 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 [12, 25, 22, 11, 34, 64, 90]

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

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

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

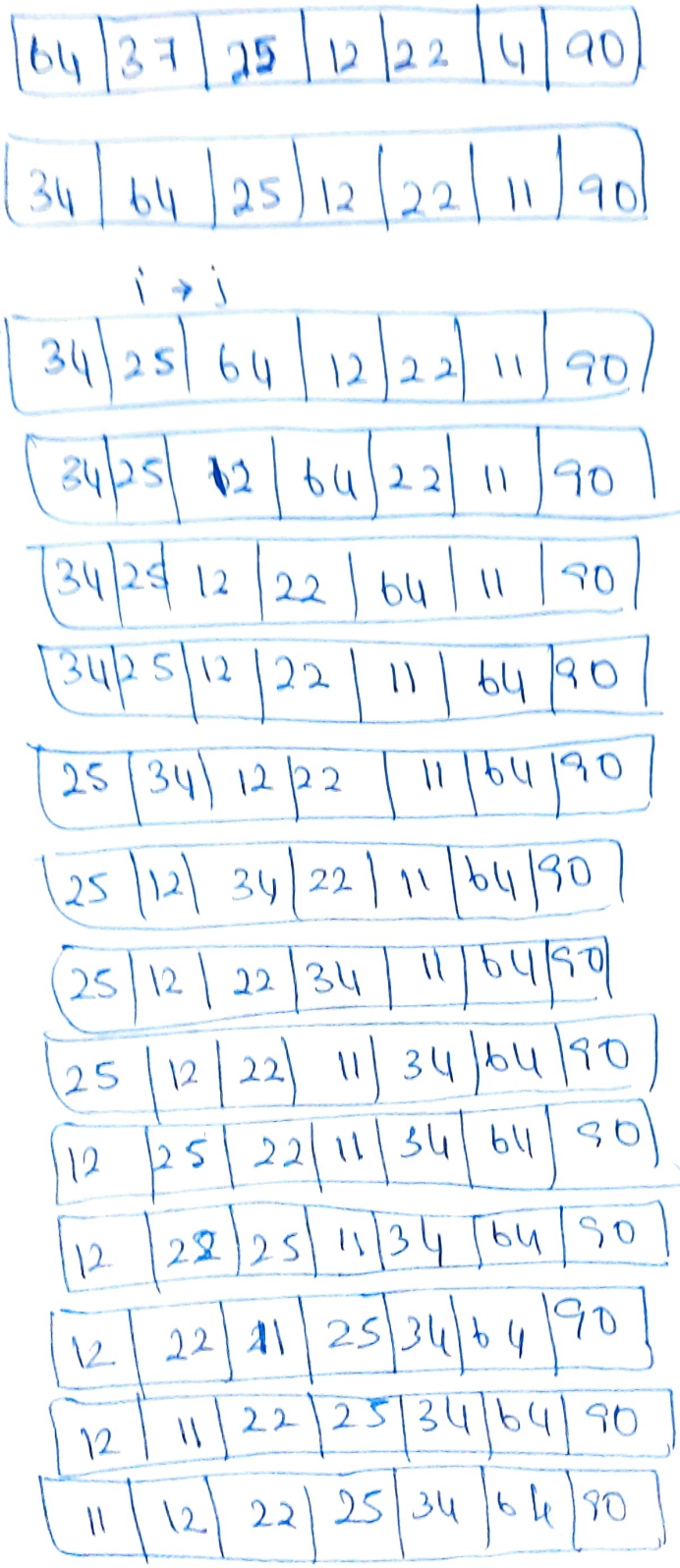
5th. [11, 12, 22, 25, 34, 64, 90]

Best = $O(n^2)$

worst = $O(n^2)$

Avg = $O(n^2)$

18) Sort the array 64, 37, 25, 12, 22, 4, 90.
using bubble sort.



Time complexity.

Worst $O(n^2)$ Best $O(n^2)$ Avg $O(n^2)$

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

Sort the array using insertion using Brute force approach

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

38	27	43	3	9	82	10	15	88	52	60	5	(35)

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

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

3	9	27	38	43	9	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)

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

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

3	9	10	15	27	38	43	52	60	52	85	5	60

3	9	10	13	27	38	42	82	60	32	85	5

3	5	9	10	15	27	38	42	52	60	82	88

Time : $O(n^2)$

Space : $O(1)$.