

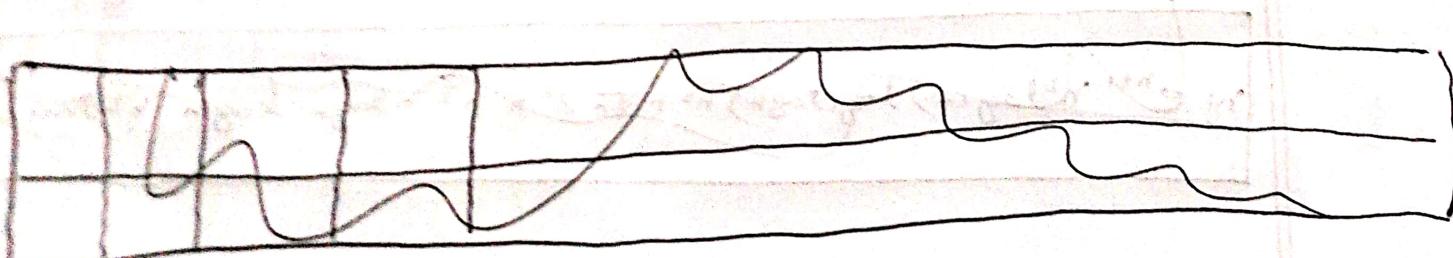
Q. (a) Given,

Need to sort $2^n, \log n, n^2 \log(n^2), \log(\log n)$,
 $n^2 \log n, \sqrt{n}, n!, n^3, n^{\frac{3}{2}}, n \log n, e^{n+1}, n^2 \log n$

For selection sort, let's put them in an array.

2^n	$\log n$	$n^2 \log(n^2)$	$\log(\log n)$	n^2	n	\sqrt{n}	$n!$	n^3	$n^{\frac{3}{2}}$	$n \log n$	e^{n+1}	$n^2 \log n$
0	1	2	3	4	5	6	7	8	9	10	11	12

As $n!$ is last the largest in array so there will be swapping.



$n!$	$\log n$	$n^2 \log(n^2)$	$\log(\log n)$	n^2	n	\sqrt{n}	2^n	n^3	$n^{\frac{3}{2}}$	$n \log n$	e^{n+1}	$n^4 \log n$
6	1	2	3	4	5	6	7	8	9	10	11	12

Now from the remaining array

e^{n+1} is the largest so they will swap.

$n!$	e^{n+1}	$n^2 \log(n^2)$	$\log(\log n)$	n^2	n	\sqrt{n}	2^n	n^3	$n^{\frac{3}{2}}$	$n \log n$	$\log n$	$n^2 \log n$
6	1	2	3	4	5	6	7	8	9	10	11	12

As now 2^n is the largest so it will swap.

$n!$	e^{n+1}	$n^2 \log(n^2)$	$\log(\log n)$	n^2	$n \sqrt{n}$	2^n	n^3	$n^{\frac{3}{2}}$	$n \log n$	$\log n$	$n^2 \log n$
6	1	2	3	4	5	7	8	9	10	11	12

$n!$	e^{n+1}	2^n	$\log(\log n)$	n^2	n	\sqrt{n}	$\frac{n^2 \log(n^2)}{2}$	n^3	$n^{\frac{3}{2}}$	$n \log n$	$\log n$	$\log \log n$	$n^2 \log n$
0	1	2	3	4	5	6	7	8	9	10	11	12	

As n^3 is the largest so it will swap.

$n!$	e^{n+1}	2^n	$\log(\log n)$	n^3	n^2	n	\sqrt{n}	$n \log n$	$\log(\log n)$	$n^2 \log n$	n^3	$n \log n$	$\log n$	$n^2 \log n$
0	1	2	3	4	5	6	7	8	9	10	11	12		

As $n^2 \log n$ is the largest so it will swap.

$n!$	e^{n+1}	2^n	n^3	$n^2 \log n$	n	\sqrt{n}	$n^2 \log(\log n)$	$n^{\frac{3}{2}}$	$n \log n$	$\log n$	$n^2 \log n$	
0	1	2	3	4	5	6	7	8	9	10	11	12

As the $n^2 \log n$ is the largest so it will swap.

DATA STRUCTURE

$n!$	e^{n+1}	2^n	n^3	$n^2 \log n$	$n^2 \log n$	\sqrt{n}	n^2	$\log(n \log n)$	$n^{\frac{3}{2}}$	$n \log n$	$\log n$	
0	1	2	3	4	5	6	7	8	9	10	11	12

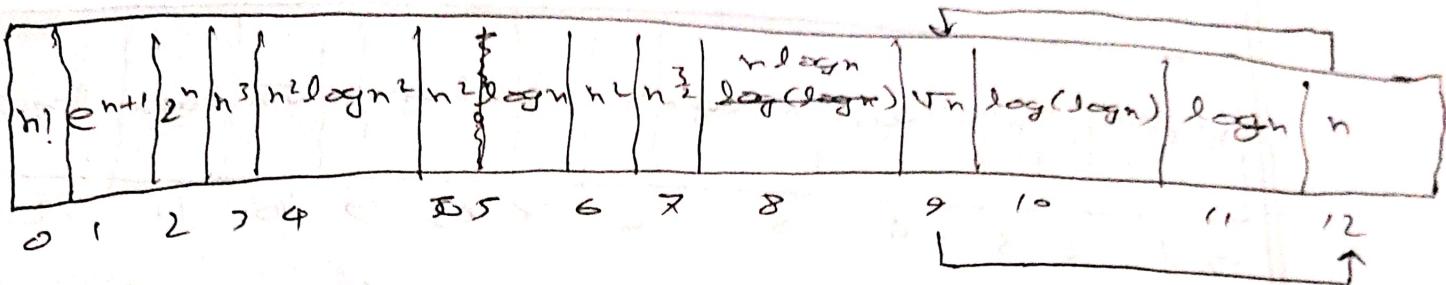
As n^2 is the largest so it will swap.

$n!$	e^{n+1}	2^n	n^3	$n^2 \log n$	$n^2 \log n$	\sqrt{n}	n^2	$\log(n \log n)$	$n^{\frac{3}{2}}$	$n \log n$	$\log n$	
0	1	2	3	4	5	6	7	8	9	10	11	12

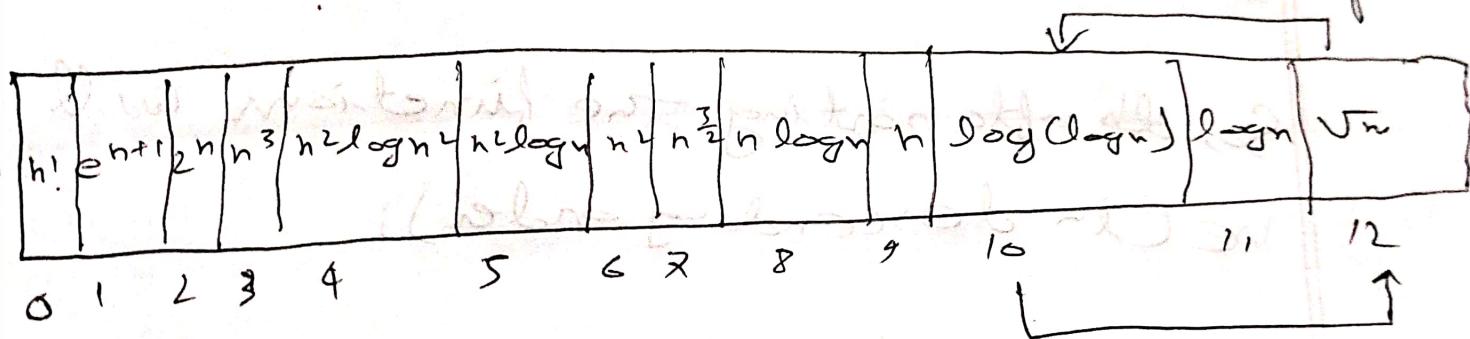
Now $n^{\frac{3}{2}}$ is the largest and it will swap.

$n!$	e^{n+1}	2^n	n^3	$n^2 \log n$	$n^2 \log n$	n^2	$n^{\frac{3}{2}}$	$\log(n \log n)$	\sqrt{n}	$n \log n$	$\log n$	
0	1	2	3	4	5	6	7	8	9	10	11	12

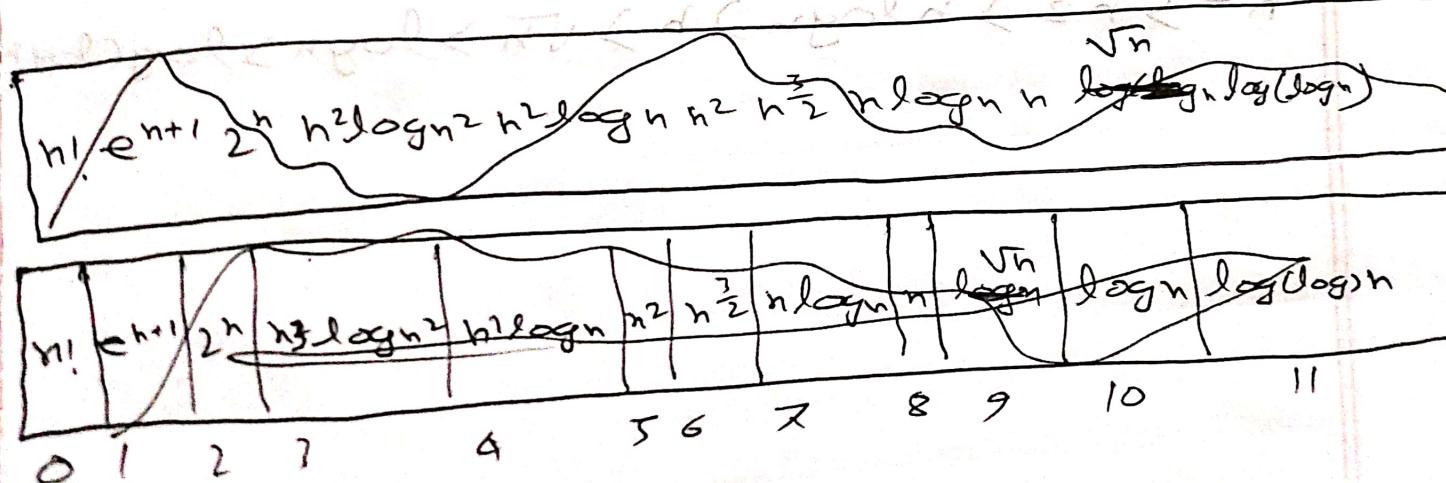
Now as $n \log n$ is the largest so it will swap.



As here n is the largest so it will swap.



As \sqrt{n} is the largest so it will swap.



$n!$	e^{n+1}	2^n	n^3	$n \log n$	$n^2 \log n$	n^2	$n^{\frac{3}{2}}$	$n \log \log n$	$\sqrt{n} \log n$	$\log \log n$
0	1	2	3	4	5	6	7	8	9	10
1	2	4	8	16	32	64	128	256	512	1024
2	3	6	12	24	48	96	192	384	768	1536
3	4	8	24	72	144	288	576	1152	2304	4608
4	5	16	64	192	384	768	1536	3072	6144	12288
5	6	32	128	480	960	1920	3840	7680	15360	30720
6	7	64	256	1152	2304	4608	9216	18432	36864	73728
7	8	128	512	2304	4608	9216	18432	36864	73728	147456
8	9	256	1024	4800	9600	19200	38400	76800	153600	307200
9	10	512	2048	10240	20480	40960	81920	163840	327680	655360
10	11	1024	4096	20480	40960	81920	163840	327680	655360	1310720

So, sorted array from selection sort.

So, after sorting the functions will be (in descending order)

$$n! > e^{n+1} > 2^n > n^3 > n^2 \log n > n^2 \log \log n >$$

$$n^2 > n^{\frac{3}{2}} > n \log n > n > \sqrt{n} > \log n > \log \log n$$

$$2(n-1) + n$$

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

$$= 2^n T(n-3) + 2^{n-2} (n-2) + 2(n-1) + n$$

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

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

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

$$\text{So, } T(n) = 2T(n-1) + n$$

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

$$\therefore T(n-2) = 2T(n-2-1) + n-2$$

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

$$\therefore T(n-1) = 2T(n-1-1) + n-1$$

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

Q6) Given,

$$\begin{aligned}
 &= 2^n T(0) + 2^{n-1} T(1) + 2^{n-2} T(2) + \\
 &\quad \dots + 2^0 T(n) + 2(n-1) + n \\
 &\leq 1 - \alpha + (1-\alpha) TS = (1-\alpha) T + S \\
 &\leq 2^n + 2^{n-1} + 2^n + 2^n + \dots [\text{which} \\
 &\quad \text{is for} \\
 &\leq \cancel{n} 2^n [\text{for } n \text{ times}] \\
 &\leq n 2^n
 \end{aligned}$$

$$\therefore T(n) = \Theta(n 2^n) + S$$

1(c) (i) The time complexity recurrence relation for fibonacci numbers is,

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

It can be written as,

$$T(n) \leq T(n-1) + T(n-1)$$

By telescoping method we will get,

$$\Rightarrow T(n) \leq 2T(n-1)$$

By telescoping method we will get,

$$\Rightarrow T(n) \leq 2\{2T(n-1-1)\}$$

$$\Rightarrow T(n) \leq 4T(n-2)$$

$$\Rightarrow T(n) \leq 2^2 T(n-2)$$

$$\Rightarrow T(n) \leq 2^2 \{2T(n-3)\}$$

$$\Rightarrow T(n) \leq 2^2 \cdot 2 \cdot T(n-3)$$

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

For 3ⁿ case of n,

$$TC(n) \leq 2^n T(0)$$

$$\therefore T(n) \leq 2^n$$

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

TOP SECRET
STAG SOOTAM

ii) Given,

$$f(n) = 2n^3 + 5n^2 + 6n + 18$$

From the function we can easily say that the highest degree will be dominating and all other can be easily ignored from the expansion of function of polynomial function.

So, This function can be written as $2n^3$

Also discarding the 2 we will get,

$$\Theta(n^3)$$

$$\therefore 2n^3 + 5n^2 + 6n + 18 = \Theta(n^3)$$

iii) Given,

$$5 + 2 \cos(n) = O(1)$$

$$\therefore f(n) = 5 + 2 \cos(n)$$

AS, ~~BE~~ $\cos(n)$ has range of from -1 to 1.

So we can write sample range

$$-1 \leq \cos(n) \leq 1$$

$$\Rightarrow -2 \leq 2 \cos(n) \leq 2$$

$$\Rightarrow 3 \leq 5 + 2 \cos(n) \leq 7$$

$$\Rightarrow 3 \leq 5 + 2 \cos(n) \leq 7$$

So, the function is between 3 to 7.

As the range is constant so we can say that $5 + 2 \cos(n) = O(1)$.

iv)

Given

$$\frac{3}{2}n^2 + 2n - 3 = \Omega(n^2)$$

So

Let S_0

$$f(n) \geq \frac{3}{2}n^2 + 2n - 3$$

An highest degree ~~term~~ is dominant
in the dominated one
can be discarded.

$$\therefore f = (\frac{3}{2}n^2)$$

Ignoring the $\frac{3}{2}$ we can say

$$f \Theta(n^2)$$

$\therefore F(n) = f(n^2)$ so we can say,

$f(n^2) = \Omega(n^2)$ as if tight bound
bound have to true then upper
and lower bound also have to
be become true.

$$\therefore \frac{3}{2}n^2 + 2n + 3 = \Omega(n^2) \text{ [Showed]}$$

V) Given,

$$n^3 + 5n \neq O(n^2)$$

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

As the highest degree in the term is dominated one and others can be ignored so $f(n) = \Theta(n^3)$.

Besides, $f(n) = O(n^3)$ or it can be also

said that $f(n) = n^3 + 5n = \Omega(n^3)$.

$$\therefore f(n) = \Theta(n^3) \text{ [Showed]}$$

$$\times f(n) = n^3 + 5n \neq O(n^2) \text{ [Showed with contradiction]}$$

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

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

$$(n) \in O(n^3)$$

From the notes
Comparing with the master's theorem
we can say,

$$a=2, b=2, k=3$$

As, $a < b^k$ if $a < b^k$

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

Here, $a < b^k > a$

$$= 2^3 > 2$$

∴ Time complexity will be $= O(n^k)$

$$= O(n^3)$$

$$\therefore T(n) = 2T\left(\frac{n}{2}\right) + n^3 = O(n^3) [\text{Shard}]$$

Vii) Given,

$$T(n) = T\left(\frac{n}{4}\right) + T\left(\frac{b \cdot 5n}{8}\right) + n$$

Comparing this term
(with master theorem)
we get,

$$a=1, b=\frac{8}{5}, k=1$$

As we know,

$$T(n) = O(n^k) \text{ if } b^k > a$$

So, so far,

$$\left(\frac{8}{5}\right)^1 > 1$$

$$\text{So: } T_1 = O(n^k) \\ = O(n)$$

Now,

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

Again from the master's theorem
we get,
 $a=1, b=4, k=1$

As, $4^1 > 1$ which is $b^k > a$

ROQATARA

∴ Time complexity will be $\Theta(n^2)$

$$= t\left(\frac{n^2}{8}\right) + t\left(\frac{n^2}{4}\right) + t\left(\frac{n^2}{2}\right) + n$$

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

$$\geq \Theta(n^2)$$

~~Q.1~~

$$P = 0$$

$$\longrightarrow x_1$$

 $\text{for } (i=1, i \leq n, i++) \rightarrow n+1$ $\text{while } (P < i) \rightarrow n+1$

$$P = P + i \longrightarrow n$$

$$\begin{aligned} \text{So total} &= 1 + n+1 + n+n \\ &= 3n+2 \end{aligned}$$

$$\therefore T(n) = 3n+2 = n$$

$$\therefore T(n) = \Theta(n)$$

2) $\text{for } (i=n, i \geq 1, i=i/3)$

$P = 2$
 $\text{while } (P < n)$

$\text{Print } ("Hello")$

$$P = P * P$$

for out the after for loop,

i.

loop will go for

3

2

2

2

2

1

9

8

7

6

5

4

So the outer for loop will go
for $(\log_2 n) + 1$

Now for while loop

$$\frac{P}{2^1} \\ 2^2 \\ 2^4 \\ 2^8 \\ \vdots \\ 2^{2^k}$$

A.S. $P < n$.

$$\therefore \text{So, } n = 2^{2^k}$$

A.S. $n = 2^2$ the loop will go for $\log_2 2$

$n = 2^4$ the loop will go for $\log_2 4$.

So, while loop executes $\log_2 2^k$ times.

Now,

$$\log_2 n = 2^k \log_2 2$$

$$\therefore \log_2 n = 2^k \cdot 2^k$$

$$\Rightarrow \log_2 (\log_2 n) = k \log_2 2 = k$$

So,

for the while total time will be,
 $\Theta(\log_2(\log_2 n))$

$$\therefore \text{total} = \{(\log_2 n) + 1\} \{\log_2 (\log_2 n)\}$$

$$\therefore \text{time complexity } \Theta(n) = (\log_2 n) (\log_2 \log_2 n)$$

3) $P = 0$

for $i=1, i \leq n, i=i+2$

$P++$

for $j=1, j \leq P, j=j+2$

Print ("Hello")

For the first for loop

1)

$$\text{Ans, } i = 2^k$$

1

$$\therefore 2^k \geq n$$

2^2

$$\therefore 2^k = n$$

2^3

$$\therefore k = \log_2 n$$

$\log_2 n$

So, first loop will go for $\log_2 n$

The second will go for $\log P$ - Can it go with the same structure with the first loop?

[

$\log P$ can be also written as $\log_2(\log_2 n)$

$$\therefore \text{Total} = \log_2 n + \log_2 (\log_2 n)$$
$$= \log_2 n + \log_2 \log_2 n$$

\therefore The complexity $T(n) = \Theta(\log_2 n)$

2)

Input size1

Input size2

Input array1

Input arr2

function searching(element, array, size):

start = 0

stop = size

point = size + 1

x = 0

while stop >= start:

middle = (start + stop) // 2

if array[middle] == element:

point = take min -t (middle, point)

stop = middle - 1

end -t if

else:

~~middle~~ = middle + 1
 start

end of while.

For i from point to size1
 If prime-check (arr2[i])
 $x = x + 1$
 end of if
 end of for
 return x

function prime-check (ele) : boolean

c = 0 // goto
 for i from 1 to ele+1:
 If ele % i == 0 :

statements & goto c = c + 1

(statements) -> label end do h
 end do for

If c == 2 :

return True

else:

return False

for a from 0 to size2:

print (arr2[a], arr1, size1)
 print (searching (arr2[a], arr1, size1))