ex: linear & binary search — running time ↓ depend on
1) speed of comp.
2) compiler & translator
3) programming lang.

* Asymptotic analysis :-
→ Idea is to have measure of efficiency of algo that doesn't depends on.
1) m/c const^n.
2) Algo to be implemented.
3) Time taken by pgm to prepare.
→ Expressing complexity in terms of its relationship, to know f^n → Asymptotic analysis.

* Asymptotic notation :-
mathematical way of representing time complexity
→ Asymptotic growth :- Rate at which f^n grows.
→ Growth rate ⇒ time + memory.

* classification of growth :-
1) growing with same rate.
2) Growing with slow rate.
3) Growing with fast rate.

* Big oh(O) notation :-
represent upper bound of algo runtime.
→ we have to calculate worst case.

Formula :- $f(n) \leq c \cdot g(n)$, $n > n_0$, $c > 0$, $n_0 \geq 1$,
$f(n) = O(g(n))$.

★ Ω omega notation :-

Represent lower bound of algorithm running time we can calculate min^m amt. of time. (best case).

formula $f(n) \geq c \cdot g(n)$ $\quad n > n_0, c > 0, n_0 \geq 1$

$f(n) = \Omega(g(n))$.

★ Θ theta notation :-

represent avg. bond of algo running time.
we can calculate avg. amt of time;

formula :- $c_1 g(n) \leq f(n) \leq c_2 \cdot g(n)$

★ little oh notation :-

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0 \qquad f(n) = o(g(n))$$

★ little omega (ω) notation :-

: loose lower bound of $f(n)$.

★Q) Difference blw big oh & little oh is that in formula of little oh there is no equal to sign.

viva:

Q) Find value of c & n [rare] [6M].

Q) Explain asymptotic notation with an ex. [6M]

DAA

14/2/25.

(1) find the upper bound of :-

$f(n) = 3n+8.$

→ cond^n of Big O notation.

$f(n) \leq c \cdot g(n)$ where $n \geq n_0$ & $c > 0$ & $n > 1$.

Assume $c = 4$.

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

$3n+8 \leq 4n$?

$\therefore 3n+8 \leq 4n$ true

	$f(n)$ $(3n+8)$	$g(n)$ $(4n)$
$n=1$	11	4
$n=5$	23	20
$n=6$	26	24
$n=7$	29	28
$n=8$	32	32
$n=9$	35	36.

$\therefore 3n+8 = O(n)$ for

$c = 4$ & $n_0 = 8$.

n=1 1+10

② $f(n) = n^2 + 10$, find the upper bound.

⇒ cond^n for Big(O) notation.

$f(n) \leq c \cdot g(n)$ where $n \geq n_0$ & $n > 1$, & $c > 0$.

Asume $c = 2$.

$n^2 + 10 \leq 2n^2$

	n^2+10	$2n^2$
$n=1$	11	2
~~$n=3$~~	~~25~~ 19	~~50~~ 18.
✓ $n=4$	26	32
~~$n=5$~~	~~35~~ 35	~~75~~ 50
~~$n=2$~~	14	8.

$n^2 + 10 \leq 2n^2$ is true.

for $c = 2$ & $n_0 = 4$.

$n^2 + 10 \Rightarrow O(n^2)$

③ $f(n) = 5n + 50$ find the upper bond.

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

Assume = 6.

$5n + 50 \leq 6n$

	$5n+50$	$6n$
$n=1$	55	6
$n=5$	75	30
$n=10$	100	60
$n=15$	125	90
$n=50$	300	300

4. Assume that the $f(n) = 5n+50$ and $g(n) = 100\log_{10}(n)$ is $g(n)$ upper bound of $f(n)$.

	$5n+50$	$100\log_{10}(n)$
~~$n=100$~~	~~550~~	200
~~$n=1000$~~		
$n=1000$	5050	300
~~$n=8$~~		

Since it is not $\leq \log_{10}(n)$
$\therefore$ not a upper bound.

DAA

7/2/25

LS :- I Introduction to Algorithm.

- Analysis :- Predict cost of algo in terms of resources & performance.
- Design :- design algorithm which min the cost.
- Algo :- set of steps of operations to solve problem performing calculation, data processing & automatic reasoning tasks.
- Algo is efficient method that can be expressed with finite amt of time & space.

Properties of algorithm :-

1) I/P :- algo should be given 0 or more i/p.
2) O/P :- at least 1 quantity is produced.
3) Definiteness :- each instruction is clear & unambiguous.
4) Finiteness :- If we trace out instructions of algo then for all cases, algo terminates after finite no. of steps.
5) Effectiveness :- Each instruction is very easy, which can be written with the help of pen and papers.

→ Pseudo code :- notation resembling simplified programming lang used in pgm. design.

* Difference Between algo & progm :-

Algorithm	Program
① At design phase.	① At implementation phase.
② Natural language.	② written in programming language.
③ Analysis	③ Testing.
④ Domain knowledge	④ Programmer.

→ Once algo is created it is necessary, to show that it computes correct o/p for all possible legal i/p → algorithm validation.

→ Analyses of algorithm by time & storage algorithm.

→ Test of pgm. Debugging & profiling / performance measurement.

★ Analysis of Algorithms:-

Priori	Posteriori
1) Run algo on specific system & language.	1) Analysis after running it on system.
2) H/w independent.	2) Depends on h/w & langu
3) Approx analysis	3) Actual statistics of algo
4) Dependent on no. of time statements is executed.	4) They do not depend on posteriori algo.
5) Algorithm	5) Program.
6) Analysis	6) Testing

* Need of algorithm / advantage

1. Use of asymptotic notations.
→ understand basic idea of problem.
→ Find approach to solve problem.
→ Improve efficiency of existing techniques.
→ good design can produce good soln.
→ Understand flow of problem.

* Performance Analysis :-
Algo is said to be efficient & fast if it takes less time to execute and consumes less memory space at run-time.

→ 2 types of space complexity.
- Const^n space complexity :- fixed amt of space for all variables, i/p values.
- Variable space complexity :- fixed amt of space for all i/p values.

ex :-
```
int square (int a)
{
    return a*a;
}
```

→ Linear space complexity :- space needed for algo is based on size.

Ex:- int sum (int A[], int n)

```
{                                   Space
  int sum = 0;                        1
  for (i=0; i<n; i++){                n
    sum = sum + A[i];                 1
    return sum;                       1
  }}
```

space = n+3

Time = 2n+3

* Space complexity:-

Amt. of memory it needs to run to complete

$S[P] = C + S_P$

fixed part + variable part.

Ex:- algo sum (a, b, c)

{ a = 10;
  b = 20;
  c = a+b; }

$S(P) = C + S_P = 3 + 0 = 3$

O(1) ⇒ complexity.

* Time complexity:-

amt of comp time it needs to run to complete

T(P) = compile time + execution time.

T(P) = tp (execution time).

Step count :-

- algo heading → 0
- Braces → 0.
- expressions → 1
- → for any looping statements → no of times loop is repeating.

ex:- for (i=0; i<n; i*2){ ⇒ $\log_2 n$.
...
}

ex:- for (i=0; i<n; i/2){ ⇒ 2^n
...
}

→ const^n time complexity & linear time complexity.

Q) what is the complexity of the given algo.

→ Freq. count can be vary but complexity does not.

Const^n time complexity :- If pgm requires fixed amt of time for all i/p values.

Ex:- int sum (int a, int b)
{ return a+b;
}

- Te, linear time complexity :- I/p values increase then time complexity will change.

comments = 0 steps.
Assignment statement = 1 step.
cond^n statement = 1 step.
loop cond^n for n times = n+1.
Body of loop = n steps.

Ex:- int sum (int A[], int n)
{ int sum=0;
for (i=0; i<n; i++){ ⇒ 4n+4
sum = sum + A[i];
return sum;
}}

13/2/25

* space complexity (components)

Space req^d in comp inorder to store ...

1) Instruction space :-
2) Data space : Required to store all const^ variable.
3) Environmental stack :- Amout of space req^ to store partially executed f^n.

* Kinds of analysis :-

1) Worst Case :- (usually)
$T(n) = max^m$ time of algo on any i/p of size n.

2) Avg. Case :- (sometimes)
$T(n)$ = expected time of algo on any i/p of size n.

3) Best Case :-
$T(n) = min^m$ time of algo on any i/p of size n.

→ Complexity refers to ratate at which storage time grows as f^n of problem size.

Imp * Analysis of algo :-

→ Goal is to compare algo in running time depend on & also memory management

ex: Linear & binary search — running time ↓ depend on
1) speed of comp
2) compiler & translator
3) programming lang.

* Asymptotic analysis :-
→ Idea is to have measure of efficiency of algo that doesn't depends on
1) m/c const^n.
2) Algo to be implemented.
3) Time taken by pgm to prepare.
→ Expressing complexity in terms of its relationship, to know t^n → Asymptotic analysis.

* Asymptotic notation :-
mathematical way of representing time complexity
→ Asymptotic growth :- Rate at which t^n grows.
→ growth rate ⇒ Time + memory.

* classification of growth :-
1) growing with same rate
2) Growing with slow rate.
3) Growing with fast rate.

* Big oh(O) notation :-
represent upper bound of algo runtime.
→ we have to calculate worst case.

Formula :- $f(n) \leq c \cdot g(n)$, $n > n_0$, $c > 0$, $n_0 \geq 1$,
$f(n) = O(g(n))$.

* Ω omega notation :-

Represent lower bound of algorithm running time. we can calculate min^m amt. of time. (best case)

formula $f(n) \geq c \cdot g(n)$ $n > n_0$, $c > 0$, $n_0 \geq 1$
$f(n) = \Omega(g(n))$.

* θ theta notation :-

represent avg bond of algo running time.
- we can calculate avg. amt of time,
formula :- $c_1 g(n) \leq f(n) \leq c_2 \cdot g(n)$

* little oh notation :-

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0 \qquad f(n) = o(g(n))$$

* little omega (ω) notation :-

: loose lower bound of $f(n)$.

*Q) Difference b/w big oh & little oh is that in viva: formula of little oh there is no equal to sign.

Q) Find value of c & n [rare] [6M].

Q) Explain asymptotic notation with an ex. [6M]

DAA

19/2/25.

(1) find the upper bound of:

$f(n) = 3n+8$.

→ cond^n of Big O notation:

$f(n) \leq c \cdot g(n)$ where $n \geq n_0$ & $c > 0$ & $n \geq 1$.

Assume $c = 4$.

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

$3n+8 \leq 4n$?

$\therefore 3n+8 \leq 4n$ true

	$(3n+8)$ $f(n)$	$(4n)$ $g(n)$
$n=1$	11	4
$n=5$	23	20
$n=6$	26	24
$n=7$	29	28
$n=8$	32	32
$n=9$	35	36.

$\therefore 3n+8 = O(n)$ for

$c=4$ & $n_0=8$.

n=1 1+10

② $f(n) = n^2 + 10$, find the upper bound.

⇒ cond^n for Big(O) notation.

$f(n) \leq c \cdot g(n)$ where $n \geq n_0$ & $n > 1$, & $c > 0$.

Asume $c = 2$.

$n^2 + 10 \leq 2n^2$

	n^2+10	$2n^2$
$n=1$	11	2
$n=3$	~~35~~ 19	~~50~~ 18.
$n=4$	26	32
$n=5$	~~35~~ 35	~~50~~ 50
~~$n=2$~~	14	8.

$n^2 + 10 \leq 2n^2$ is true.

for $c = 2$ & $n_0 = 4$.

$n^2 + 10 \Rightarrow O(n^2)$

③ $f(n) = 5n + 50$ find the upper bond.

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

Assume = 6.

100

15 15
5 x 6
75 90

n = 30
6
180

150
150
200

4 49
5
5

5 49
6
294

500
50
550

2

$5n + 50 \leq 6n$

	$5n+50$	$6n$
$n=1$	55	6
$n=5$	75	30
$n=10$	100	60
$n=15$	125	90
$n=50$	300	300

$n=50$
250
+ 50
300

300
250
50
300

$n=51$
× 5
255
50
305

51
6
306

(4) Assume that the $f(n) = 5n+50$ and $g(n) = 100\log_{10}(n)$ is $g(n)$ upper bound of $f(n)$.

	$5n+50$	$100\log_{10}(n)$
$n=100$	550	200
~~$n=1000$~~		
$n=1000$	5050	300
~~$n=8$~~		

$n=1$
40
50
90

Since it is not $\leq \log_{10}(n)$
∴ not a upper bound.

⑤ $f(n) = 2^n + 3n^3$.

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

$g(n) = 2^{n+1}$

	$2^n + 3n^3$	2^{n+1}
$n=5$	66	64.
$n=6$	208	128.
$n=7$	3~~4~~ 3.	256
$n=10$	~~1024~~ 4024	2048.
$n=14$		
$n=13$	14783	16384.

⑥ find the lower bound of $f(n) = 10n^2 + 5$.

⇒ consider Big-Ω notation.

$f(n) \geq c \cdot g(n)$ where $n \geq n_0$, $c > 0$, $n_0 \geq 1$.

तेव्हा आपन lower bound find karto तेव्हा आपन dominating term use karto.

⑦ show that the $f(n) = n^3 + 3n^2 = \Theta(n^3)$

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

	$f(n)$ n^3+3n^2	upper bound $2n^3$	lower bound n^3
$n=1$	4	2	1
$n=2$	20	16	8
$n=3$	54	54	27
$n=4$	112	128	64

for $n=1$

$1 \leq 4 \nleq 2$ ×

for $n=2$

$8 \leq 20 \nleq 16$ ×

for $n=3$

$27 \leq 54 \leq 54$ ✓

for $n=4$

$64 \leq 112 \leq 128$ ✓

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

is true for $c_1=1$, $c_2=2$ & $n_0=3$.

$\therefore f(n)=n^3+3n^2 \Rightarrow \underline{\underline{\Theta(n^3)}}$

OR

for upper bound

15/2/25

Q) S.T $f(n) = n^3 + 3n^2 = \Theta(n^3)$

⇒ consider for Avg bound (Θ notation).

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

where $n \geq n_0$ and c & n_0 are constant.

① $f(n) \leq c_2 g(n)$

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

Assume $g(n) = 2n^3$

$c = 2$

$n^3 + 3n^2 = O(n^2)$ for $c = 2$ & $n_0 = 3$.

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

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

$n^3 + 3n^2 = O(n^2)$ for $c = 2$ & $n_0 = 3$.

for Θ notation

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

for

$c_1 = 1$, $c_2 = 2$,

$n_0 = 3$.

is true.

$f(n) \geq c_1 \cdot g(n)$

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

Assume $g(n) = n^3$

$c = 1$

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

$n_0 = 1$, $c = 1$

* Decrement function :-

A f^n which has a denominator value is greater than the numerator.

* Constant f^n :-

Parallel to x-axis.

eg: 100, 2^{10}, 5, 1000.

Constant f^n asymptotically bigger than the decrement f^n.

a) $f(n) = \frac{100}{n}$, $g(n) = 1000$

is $f(n) = O(g(n))$.

$f(n) \leq c \cdot g(n)$ where $n \geq n_0$, and c, n_0 are const.

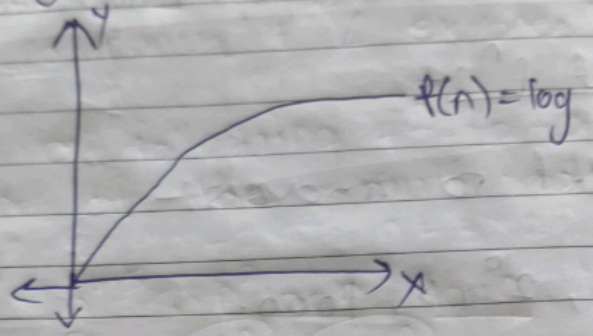
	$\frac{100}{n}$	$1 \quad c = \frac{1}{1000}$
$n = 10$	10	1
$n = 100$	1	1
$n = 1000$	0.1	1
$n = 10000$	0.01	1
	$\therefore \frac{1}{n}$	$\therefore \frac{100}{n} \leq 1$

$\therefore f(n) = O(1)$

is true for $n \geq 100$, $c = 1/1000$.

★ Logarithmic f^n :-

eg:- $\log n (\log n)^5$, $\log \log n$, $\log n \cdot \log \log n$.



$f(n) = 100$, $g(n) = \log_{10} n$

logarithmic f^n is Asymtotically bigger than the constant f^n.

③ ★ $f(n) = 100$, $g(n) = \log_{10} n$.

$f(n) \leq c \cdot g(n)$ $\quad g(n) = \log_{10} n$.

	100	$\log_{10} n$
$n = 10$	100	1
$n = 10^{100}$	100	100
$n = 10^{1000}$	100	1000
$n = 10^{10000}$	100	10000

∴ $f(n) \leq c \cdot g(n)$ for, $c = 1$, $n_0 = 10$

* Polynomial f^n :- The polynomial f^n is asymtotically bigger than logarithmic.

eg :- $n^2, \sqrt{n}, n^{10}, n^k$.

(2) $f(n) = \sqrt{n}$ $g(n) = \log_{10} n$.

$f(n) \leq c \cdot g(n)$.

	$\log_{10} n$	$\sqrt{n}$
$n = 10$	1	$\sqrt{10} = 3.16$.
$n = 10^{100}$	100	10^{50}
$n = 10^{1000}$	1000	10^{500}

$\therefore$ $f(n) \leq c \cdot g(n)$ is true.

* Exponential f^n :-

eg :- $2^n, n!, n^n, 5^n$ .....

The Exponential f^n is asymtotically bigger than the polynomial f^n.

(2) $g(n) = 2^n$, $f(n) = \sqrt{n}$.

	$\sqrt{n}$	2^n
$n = 2$	1.41	4
$n = 4$	2	16

2	10
5	5
	1

★ Conclusion:-

Decrement f^n < constant f^n < logarithmic f^n < polynomial f^n < exponential f^n.

Q) Write a following f^n in a asymptotical increasing order of the growth rate.

① ⇒ n^4, $n^{4.5}$, $n^2 \log n$, $n \log n$, $\log(n) \cdot (\log(n))^{10}$, 100, 2^{5^2}, 0.5^n, 5.5^n → exp.

decrement

⇒ $(0.5)^n$, 2^{5^2}, 100, $\log(n) \cdot (\log(n))^{10}$, $n^2 \log n$, $n \log n$, $n^{4.5}$, n^4, 5.5^n.

⇒ $(0.5)^n$, 2^{5^2}, 100, $\log(n) \cdot (\log(n))^{10}$, $\log n$, $n^2 \log n$, n^4, $n^{4.5}$, 5.5^n.

✓

12/25

sum:-

```
Algo sum(a,n)
{
  sum=0 ——— (1)
  for(i=1; i<=n; i++) ——— n+1
  {
    sum = sum+a[i]; ——— n
    return sum; ——— (1)
  }
}
```

Time complexity = $n+1+n+1+1$

$= 2n+3$

Time complexity $\Theta(n)$

ex2:- for(

ex3:-

Ex4:-

2^n $2^n + n^2 + \log$

Ex2:-
```
for (i=1 ; i<=n ; i+2)
{
  Print ("computer");
}
```

n=6

i<=6	i	i=i+2
1<=6 ✓	1	3
3<=6 ✓	3	5
5<=6 ✓	5	7
7<=6 ✗	7	

i=i+2

i=1, 1+2x2, 1+3x2, 1+4x2, 1+5x2 ... 1+kx2 ....

1+kx2 = n

Assume $k = \frac{n-1}{2}$.

~~Assume~~ ~~k~~

Ex3:-
```
for (i=1 ; i<=n ; i+5)
{
  Pf ("computer")
}
```

i = 1, 1+2x5, 1+3x5, ... 1+kx5.

Ex4:-
```
for (i=n ; i>=1 ; i--)
{
  Pf ("computer");
}
```

(n-1). Time complexity.

∴ n-k = 1

$k = n-1$.

EX) for (i=n; i>=1; i-2)
{
pf ("computer")
}

i = n, n-2, n-2-2, n-2-2-2, ...

i = n, n-2, n-2×2, n-3×2, n-4×2 ...
n-k×2

n-k×2 = 1
n-2k = 1
n-1 = 2k
$\frac{n-1}{2} = k$. $\Theta(n)$.

EX) for (i=1; i<=n; i*2)
{
pf ("computer")
}

i = 1, 1×2, 1×2², 1×2³, ... 2^k

$2^k = n$
$k \log_2 2 = \log_2 n$
$k = \log_2 n$
$\Theta(\log n)$.

Ex)
```
for(i=n, i>=1 ; i/2)
{
  pf("n");
}
```

$i = n, n/2, \frac{n}{2}, \frac{n/2}{2}, \frac{n/4}{2}, \frac{n}{8} \cdots \frac{n}{2^k}$

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

$n = 2^k$

$\log_2 n = k$

$\therefore \boxed{\Theta(\log n)}$

Ex)
```
for(i=3, i<=n, i=i^2)
{
  a=a+5
}
```

$i = 3, 3^2, (3^2)^2, ((3^2)^2)^2 \cdots$

$3^1, 3^2, 3^4, 3^8, 3^{16}, \cdots 3^{2^k}$

$i = 3, 3^2, (3^2)^2, (3^4)^2, (3^8)^2,$

$3, 3^2, 3^4, 3^8, 3^{16} \cdots$

$3, 3^2, 3^{2^2}, 3^{2^3}, 2^{2^4} \cdots$

Assume $3^{2^k} = n$

$2^k \log_3 3 = \log_3 n$

$2^k = \log_3 n$

$k \log_2 2 = \log_2 (\log_3 n)$

$\boxed{k = \log_2 (\log_3 n)}$.

$\boxed{\Theta(\log(\log n))}$.

Sol) for ($i = n$; $i \geq 5$; $i = \sqrt{i}$)

{

Pf - - -

}

$i = n$, $\sqrt{n}$, $\sqrt{(\sqrt{n})}$

n, $n^{1/2}$ $n^{1/4}$, $n^{1/8}$, - - -

$n^{1/2^k} = 5$.

$\frac{1}{2^k} \log_5 n = 1$.

$\frac{1}{2^k} \log_5 n = 1$

$2^{-k} \log_5 n = 1$

$2^{-k} = \frac{1}{\log_5 n}$.

$n^{2^{-k}} = 5$

$-k \log_2 2 = \log_2 \left(\frac{1}{\log_5 n} \right)$

$-k = \log_2(1) - \log_2(\log_5 n)$

$+k = +\log_2(\log_5 n)$

$k = \log_2(\log_5 n)$

$\theta(\log(\log n))$.

OR.

$\frac{1}{2^k} \log_5 n = 1$

$\log_5 n = 2^k$

$\log_2 \log_5 n = k$

for (i = 1, i <= 2^n, i++)
{
  printf("Hi");
}

i = 1, i = 2, i = 3, i = 4, ... - - k.

$k = 2^n$

~~$\log k = n$~~ $\theta(2^n)$.

Ex) for (i=1 ; i<=√n ; i++)
{ pf ("Ty")
}

i=1, $k = n^{1/2}$

$\Theta(n^{1/2})$.

Ex) for (i= n^2, i>=1 ; i=i/2)
{
pf ("Ty");
}

$i = n^2, n^2/2, n^2/4, n^2/8 \dots\dots n^2/2^k$.

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

$n^2 = 2^k$

$2\log_2 n = k$.

$2\log_2 n^{\frac{1}{2}} = k$.

$\Theta(\log n)$

$\Theta(n)$ <

$\Theta(n^2)$

$\log(n^2)$

Ex) for (

i= 5

25
5

Ex) fo

i=

2n+

$(2^m)^n$

$2^{m \times n}$.

Ex) for ($i = 5^n$; $i >= 5$; $i = \sqrt[5]{i}$)

$i = 5^n, 5^{n/5}, 5^{n/25}, 5^{n/125}$ ——

$5^{n/5^k} = 5$

$\log_5$ on both

$\frac{n}{5^k} = 1$

$n = 5^k$

$\log_5 n = k$

$\Theta(\log n)$

Ex) for ($i = 2$; $i < 2^n$; $i = i^2$)

{

Pf ("TY")

}

$i = 2, 2^2, 4^2, 16^2, (32)^2, \ldots$

2^{2^k}

$2n+1$

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

$2^{2k} = 2^n$

$2k = n$

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

$(2^2)^k = 2^n$.

$\Theta(n)$.

$k \log_2 2^2 = n$

$2k = n$.

$2 \log 2^k = n$

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

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

$- \log_2 2$

$k = \log_2 n - 1$

$(2^m)^n$

$2^{m \times n}$.

① for (i=3, i<=n ; i=i^3)

② ~~for~~ {

for(j=n; j<=1 ; j=j/2)

{

for(k=1 ; k<=n ; k=k*4)

{

Pf(a);

}}}.

$i = 3, 3^3, (3^3)^3, ((3^3)^3)^3 \ldots$

$= 3, 27, (27)^3$

$3^{3^k} = n$

$3^k \log_3 3 = \log_3 n$

$3^k = \log_3 n$

$k = \log_3 \log_3 n$ ——— ①

$j = n \quad n/2 \quad \frac{n}{4} \quad \frac{n}{8} \quad \frac{n}{2^k} = 1$

$n = 2^k$

$\log_2 n = k$ ——— ②

$k = 1, 1*4, 1*4*4, \ldots 4^k$

$4^k = n$

$k = \log_4 n$ ——— ③

$= \log_3(\log_3 n) \times (\log_3 n) \times (\log_4 n)$

$= \log(\log_3 \log_3 n + \log_3 n + \log_4 n)$.

$\log(a+b) = \log a \cdot \log b$.

$\log(a \cdot b) = \log a + \log b$.

$\Rightarrow \boxed{(\log \log n) \cdot \log^2 n}$ Time complexity.

21/2/25

Ex:- for(i=1; i<=n; i++) —— n+1
{
for(j=1; j<=n; j++) —— n.
{
a=a+5; }} —— n^2

Ex:- for(i=1; i<=n; i++) —— n+1
for(j=1; j<=n; j=j+1) —— n^2
{
a=a+5 }}.

i+i = n.	i=1
n-1+n	i<=n
2n-1	j=2
	2<=n.

$n \cdot \frac{n}{2} \cdot \frac{n}{3} \cdot \frac{n}{4} \cdot \frac{n}{5}$

$K = \frac{n}{K}$

a = 1

b = 1/2

$$n\left(\frac{1}{1}+\frac{1}{2}+\frac{1}{3}+\cdots\cdots+\frac{1}{n}\right)$$

$$(n \cdot \log n).$$

* Recurrance relation: is a mathematical Recurrance relation that describes the cost of the problem in terms of cost of solving the smaller sub problem.

Algo § fact (n) —— T(n)

if (n==1) —— 1

return 1

else

return n * fact(n-1). —— n * T(n-1)

$$T(n) = \begin{cases} n * T(n-1) & n > 1 \\ 1 & n = 1. \end{cases}$$

Sol

$$T(n) = \begin{cases} T(n-1) + c & \text{if } n > 1 \\ c & n = 1 \end{cases}$$

$T(n) = T(n-1) + c$ — (1)

put $n = n-1$ in eq^n (1).

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

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

put the value of $T(n-1)$ in eq^n (1).

$T(n) = T(n-2) + 2c$ — (2).

put n as $n-2$ in eq^n (1).

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

$T(n) = T(n-3) + 3c$ — (3).

$T(n) = T(n-k) + kc$ — (n).

Put $n-k=1$

$k = n-1$

$T(n) = T(1) + nc - c$

$T(n) = c + nc - c$

$T(n) = nc$

$\therefore \boxed{\Theta(n)}$

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

$T(n) = T(n-1)+n$ ——— (1)

Put $n = n-1$

$T(n-1) = T(n-2)+n-1$ ——— (2)

Put eqn 2 in 1 :

$T(n) = T(n-2)+(n-1)+(n)$ ——— (3)

Put $n = n-2$

$T(n-2) = T(n-3)+n-2$ ——— (4)

put eqn (4) in eqn (3).

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

$\vdots$

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

Put $n-k = 1$.

$k = n-1$.

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

$\cdots\cdots (n)$.

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

$= 1+2+3+\cdots\cdots n$.

$T(n) = \frac{n^2+n}{2}$ $\quad \Theta(n^2)$.

3) $T(n)$

Sol:- $T(n$

3) $$T(n) = \begin{cases} T(n-1) + \log n & \text{if } n > 1 \\ 1 & n = 1 \end{cases}$$

Sol:- $T(n) = T(n-1) + \log n$ ——— (1)

put $n = n-1$

$T(n-1) = T(n-2) + \log(n-1)$ --- put in eq^n (1).

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

put $n = n-2$

$T(n-2) = T(n-3) + \log(n-2)$ -- put in eq^n (1).

$T(n) = T(n-3) + \log(n-2) + \log(n-1) + \log n.$

$T(n) = T(n-4) + \log(n-3)(n-2)(n-1)(n).$

$T(n) = T(n-k) + \log.(n-(k-1)).(n-(k-2)).(n).$

put $n-k=1.$

~~$n=1+k$~~.

$k = n-1.$

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

$= 1 + \log(2)\cdot(3)\cdot\cdots\cdot(n).$

$= 1 + \log(n!)$

$n! = n \times (n-1) \times (n-2) \cdots\cdots - (n-k)1$

$= n\left[n\left(1-\frac{1}{n}\right)n\left(1-\frac{2}{n}\right)\cdots n\left(1-\frac{k}{n}\right)!\right]$

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

$T(n) = T(n-1)+n$ ——— (1)

Put $n = n-1$

$T(n-1) = T(n-2)+n-1$ ——— (2)

Put eqn 2 in 1 :

$T(n) = T(n-2)+(n-1)+(n)$ ——— (3)

Put $n = n-2$

$T(n-2) = T(n-3)+n-2$ ——— (4)

put eqn (4) in eqn (3) :

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

$\vdots$

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

Put $n-k = 1$.

$k = n-1$.

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

$\cdots\cdots (n)$.

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

$= 1+2+3+\cdots\cdots n.$

$T(n) = \frac{n^2+n}{2}$ $\quad \Theta(n^2)$.

3) $T(n)$

Sol:- $T($

3) $T(n) = \begin{cases} T(n-1) + \log n & \text{if } n > 1 \\ 1 & n = 1 \end{cases}$

Sol:- $T(n) = T(n-1) + \log n$ ——— (1)

put $n = n-1$

$T(n-1) = T(n-2) + \log(n-1)$ --- put in eq^n (1).

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

put $n = n-2$

$T(n-2) = T(n-3) + \log(n-2)$ -- put in eq^n (1).

$T(n) = T(n-3) + \log(n-2) + \log(n-1) + \log n.$

$T(n) = T(n-4) + \log(n-3)(n-2)(n-1)(n).$

$T(n) = T(n-k) + \log.(n-(k-1)).(n-(k-2)).(n).$

put $n-k = 1.$

$k = n-1.$

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

$= 1 + \log(2)\cdot(3)\cdot\cdots\cdot(n).$

$= 1 + \log(n!)$

$n! = n \times (n-1) \times (n-2) \cdots\cdots - (n-k)!$

$= n\left[n\left(1-\frac{1}{n}\right)n\left(1-\frac{2}{n}\right)\cdots n\left(1-\frac{k}{n}\right)!\right]$

8) $T(n) = \begin{cases} T(n-1) + \log n & \text{if } n > 1 \\ 1 & n = 1 \end{cases}$

Sol:- $T(n) = T(n-1) + \log n$ ——— ①

put $n = n-1$

$T(n-1) = T(n-2) + \log(n-1)$ --- put in eq^n ①.

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

put $n = n-2$

$T(n-2) = T(n-3) + \log(n-2)$ -- put in eq^n ①.

$T(n) = T(n-3) + \log(n-2) + \log(n-1) + \log n.$

$T(n) = T(n-4) + \log(n-3)(n-2)(n-1)(n).$

$T(n) = T(n-k) + \log.(n-(k-1)).(n-(k-2)).(n).$

put $n-k = 1.$

$k = n-1.$

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

$= 1 + \log(2) \cdot (3) \cdot \cdots (n).$

$= 1 + \log(n!)$

$n! = n \times (n-1) \times (n-2) \cdots (n-k)!$

$= n\left[n\left(1-\frac{1}{n}\right)n\left(1-\frac{2}{n}\right) \cdots n\left(1-\frac{k}{n}\right)!\right]$

Q ⇒ $T(n) = \begin{cases} T(n-1) + n & \text{if } n > 1 \\ 1 & \text{if } n = 1 \end{cases}$

$n! = n^n \sum_{k=0}^{n-1} \left(1 - \frac{k}{n}\right)$

$\log n^n$

$T(n) = \boxed{\Theta(n \log n)}$

27/2/25

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

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

$n = n-1$

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

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

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

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

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

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

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

put $n-k = 1$

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

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

① $\frac{a(1-r^n)}{1-r}$, $r \neq 1$ and $r < 1$.

② $\frac{a(r^n-1)}{r-1}$ $r \neq 1$ and $r > 1$.

$= 2^{n-1} + 2^{n-2} \times 2 + 2^{n-3} \times 2^2 + \cdots + 2^{n-(n-2)} \times 2^{(n-2)} + 2^{(n-1)} + n.$

$= 2^0(n) + 2^1(n-1) + 2^2(n-2) + \cdots + 2^{n-2} \times 2^{n-2} + 2^{n-1}$ — (4).

Multiply by 2 both side.

$2T(n) = 2(n) + 2^2(n-1) + 2^3(n-2) + \cdots + 2^n \cdot 2^{n-1} + 2^n \cdot 1$ — (5).

$2n-2n+2$

(5) − (4).

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

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

$= 2^n \cdot 2 - (n+2)$

$[O(2^n)]$.

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

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

$n = \frac{n}{2}$

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

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

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

$$n = \frac{n}{4}$$

$$T\left(\frac{n}{4}\right) = 2T\left(\frac{n}{8}\right) + \frac{n}{4}.$$

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

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

$$= 2^3 T\left(\frac{n}{8}\right) + \frac{2^2 n}{4} + \frac{2n}{2} + n.$$

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

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

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

$n = 2^k$

$\log_2 n = k \log_2 2$

$\log_2 n = k$.

$= 2^{\log_2 n} T(1) + \log_2 n \cdot n$.

$= n + n \log_2 n = n \log n$ dominating term

~~$= n + n \log_2 n^n$~~ $= \boxed{\Theta(n \log n)}$.

~~$= n$~~

Q) $T(n) = \begin{cases} \sqrt{n}\, T(\sqrt{n}) + n & \text{if } n \geq 2 \\ 2 & \text{if } n = 2 \end{cases}$

$T(n) = \sqrt{n}\, T(\sqrt{n}) + n$

put $n = \sqrt{n}$

$T(\sqrt{n}) = n^{1/4} T(n^{1/4}) + n^{1/2}$

put $n = n^{1/4}$

$T(n^{1/4}) = n^{1/16} T(n^{1/16}) + n^{1/8}$.

$T(n) =$

Nahi tha.

28/2/25

(Q). $T(n) = n^{1/2}\, T(n^{1/2}) + n$.

put $n = n^{1/2}$

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

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

$T(n) = n^{3/4}\, T(n^{1/4}) + n + n$.

put $n = n^{1/4}$

$T(n^{1/4}) = n^{1/8}\, T(n^{1/8}) + n^{1/4}$

$T(n) = n^{3/4}\left(n^{1/8}\, T(n^{1/8}) + n^{1/4}\right) + n + n$.

$= n^{\frac{1}{2}+\frac{1}{4}+\frac{1}{8}}\, T(n^{1/8}) + n^{\frac{1}{2}+\frac{1}{4}}$.

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

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

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

$T(n) = n^{7/2^3}\, T(n^{1/2^3}) + n + n + n$.

$T(n) = n^{\frac{1}{2^k}-1}\, T(n^{1/2^k}) + kn$.

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

Put $n = 2^k$

$\frac{1}{2^k} \log_2 n = 1 \Rightarrow 2^k = \log_2 n$

$k = \log_2 (\log_2 n)$

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

$= n^{1 - 1/\log n} \cdot 2 + n \log(\log n)$

$= \frac{n}{n^{1/\log n}} \cdot 2 + n \log(\log n)$

$= \frac{n}{2} \cdot 2 + n \log(\log n)$

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

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

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

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

put $n = n^{1/2}$

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

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

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

put $n = n^{1/4}$

$$T(n^{1/4}) = 2T(n^{1/8}) + 1$$

$$T(n) = 2^2\left[2T(n^{1/8}) + 1\right] + 2 + 1$$

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

$$T(n) = 2^k T(n^{1/2^k}) + 2^{k-1} + 2^{k-2} + \cdots + 2^0$$

put $n^{1/2^k} = 2$

$$\frac{1}{2^k}\log n = \log_2 2$$

$$\log n = 2^k$$

$$\boxed{k = \log_2(\log n)}$$

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

$$= \log(n) \cdot 2 + \frac{2^k - 1}{1}$$

$$= 2\log(n) + \log n - 1$$

$$= 3\log(n) - 1$$

$$\Rightarrow \boxed{\Theta(\log(n))}$$

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

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

put $n = \frac{n}{2}$

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

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

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

$$T(n) =$$

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

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

put $n = \frac{n}{2^2}$

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

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

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

$n = 2^k$

$k = \log n.$

$T(n) = (\sqrt{2})^{\log n} T\left(\frac{n}{2^{\log_2 n}}\right) + \log_2 n \sqrt{n}.$

$= \sqrt{2}$ +

$= n$

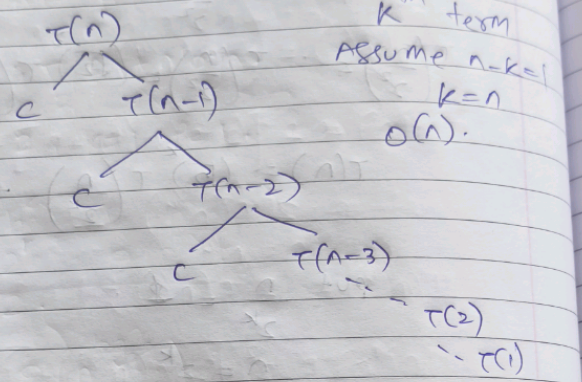
$T(n) = (\sqrt{2})^{\log_2 n} T(1) + \log n \sqrt{n}.$

$= \sqrt{2}\, n^{\log_2 \sqrt{2}} + \log \sqrt{n} \log n.$

$\boxed{\Theta = \sqrt{n} \log n}.$

* Recurrance Tree method (2nd Method) (not asked in exam).

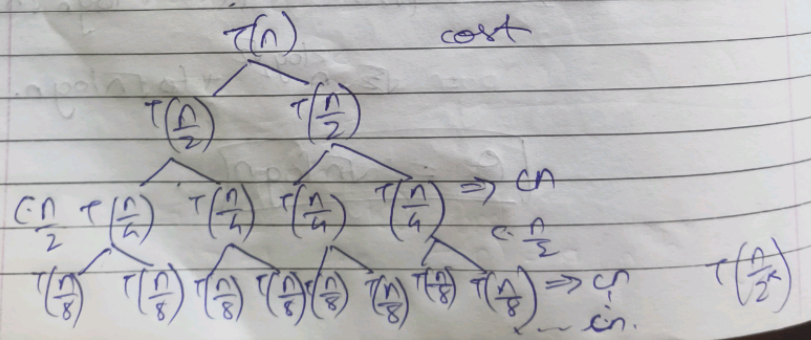
ex:- $T(n) = \begin{cases} T(n-1)+c & n>1 \\ 1 & n=1 \end{cases}$



k^{th} term
Assume $n-k=1$
$k=n$
$O(n)$.

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

→ $T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}\right) + cn$



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

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

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

$2^k = n$

$k = \log n$

$T(n) = 1 + \log n \cdot cn$

$T(n) = 1 + n \log n$

$\Rightarrow \boxed{\Theta(n \log n)}$

6/3/25

Master th^m :- Method 3.

$T(n) = aT(n/b) + \Theta(n^k \log^p n)$

$a \geq 1$, $b > 1$, $k \geq 0$ and p is real no.

case 1 :- if $a > b^k$ then $T(n) = \Theta(n^{\log_b a})$

case 2 :- if $a = b^k$ then

a) $p > -1$ then $T(n) = \Theta(n^{\log_b a} \log_n^{p+1})$

b) $p = -1$ then $T(n) = \Theta(n^{\log_b a} \log(\log n))$

c) $p < -1$ then $T(n) = \Theta(n^{\log_b a})$

Case 3:- if $a < b^k$ then

a) if $p \geq 0$ then
$T(n) = \theta(n^k \log^p n)$

b) if $p < 0$ then
$T(n) = \theta(n^k)$

Q1) $T(n) = 4T\left(\frac{n}{2}\right) + n\log n$.

Q2) $a=4,\ b=2,\ k=1,\ p=1$.

$a \quad b^k$

$4 > 2^1$

$4 > 2$.

$T(n) = \theta\left(n^{\log_2 4}\right)$

$= \theta(n^2)$

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

Q2) $T(n) = 4T\left(\frac{n}{2}\right) + n^2\log n$.

$a=4,\ b=2,\ k=2,\ p=1$

$a \quad b^k$

$4 = 2^2$

$4 = 4$.

$\therefore$ case 2 :- b) condⁿ.
$p > -1$ the $T(n) = \theta(n^{\log_b a} \log^{p+1} n)$.

Q3) T(n

Q4) T(

Q5)

$\therefore T(n) = \Theta(n^{\log_2 4} \log^2 n)$

$\boxed{T(n) = \Theta(n^2 \log^2 n)}$

Q3) $T(n) = 4T\left(\frac{n}{2}\right) + n \log n$.

$a = 4 \quad b = 2 \quad k = 1 \quad p = 1$

$a \quad b^k$

$4 \rightarrow 2^1$

$\boxed{T(n) = \Theta(n^2)}$.

Q4) $T(n) = 2T\left(\frac{n}{2}\right) + \frac{n}{\log n}$.

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

$a = 2 \quad b = 2 \quad k = 1 \quad p = -1$

$a \quad b^k$

$2 = 2^1 \quad p = -1$

$T(n) = \Theta(n^{\log_2 2} \log \log n)$

$\boxed{T(n) = \Theta(n \log(\log n))}$

Q5) $T(n) = 2T\left(\frac{n}{2}\right) + \frac{n}{\log^2 n} \Rightarrow \boxed{\Theta(n)}$.

$T(n) =$

Q6) $T(n) = 2T\left(\frac{n}{2}\right) + n^2$

$a = 2$, $b = 2$, $k = 2$, $P = 0$

a	b^k
2	2^2
2	< 4

$\therefore$ case 3 :- a) $P \geq 0$

then $T(n) = \theta(n^k \log^P n)$

$T(n) = \theta(n^2 \log^0 n)$

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

Q7) ~~$T(n) = 2T\left(\frac{n}{2}\right) + n$~~

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

Q8) $T(n) = 3T\left(\frac{n}{2}\right) + \frac{n^2}{\log n}$

$a = 3$, $b = 2$, $k = 2$, $P = -1$

a	b^k
3	2^2
3	< 4

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

(9) $T(n) = \sqrt{2}\,T\left(\frac{n}{2}\right) + \log n$

$a = \sqrt{2} \quad b = 2 \quad k = 0 \quad p = 1$

a	b^k
$\sqrt{2}$	2^0
$\sqrt{2} > 1$	

$T(n) = \Theta(n^{\log_2 2^{1/2}})$

$\boxed{T(n) = \Theta(\sqrt{n})}$

(10) ~~$T(n) = \sqrt{n} + \log n$~~

$T(n) = T(\sqrt{n}) + \log n$

~~$a = 1 \quad b = \quad k = 0 \quad p = 1$~~

assume $n = 2^m$

$\sqrt{n} = (2^m)^{1/2}$

$= 2^{m/2}$

$T(2^m) = T(2^{m/2}) + \log 2^m$

$T(2^m) = T(2^{\frac{1}{2}m}) + m$

$T(2^m) = T(2^{m/2}) + m.$

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

$T\left(\frac{\sqrt{n} \times \sqrt{n}}{\sqrt{n}}\right)$

$T\left(\frac{n}{\sqrt{n}}\right)$

T

$f(a) = g(b)$ $\quad T(a)$

$f\left(\frac{a}{2}\right) = g\left(\frac{b}{2}\right)$

assume.

$T(2^m) = S(m)$ $\quad T(2^{m/2}) = S\left(\frac{m}{2}\right)$

T

$S(m) = S\left(\frac{m}{2}\right) + m$

$a=1,\ b=2,\ k=1,\ p=0.$

a	b^k
1	2
1	< 2

Case 3:- a)

$T(n) = \Theta(n^k \log^p n)$

$T(n) = \Theta(m \log^0 m)$

$T(n) = \Theta(m)$ --- a $\quad 2^m = n$, $\log n = m$

$\therefore \boxed{T(n) = \Theta(\log n)}$

$n = 2^m$

$\log_2$

$\log_2 n = \log_2 2^m$

$\boxed{\log_2 n = m}$

Q11) $T(n) = 2$

assum

$T(2^m$

$T(2$

T

$n \log$

$\log n$

7/8/25

Q11) $T(n) = 2T(\sqrt{n}) + \log n$

assume $\Rightarrow 2^m = n$

$\sqrt{n} = 2^{m/2}$ ($n^{1/2} = 2^{m/2}$)

$T(2^m) = 2T(2^{m/2}) + \log 2^m$

$T(2^m) = 2T(2^{m/2}) + m$

$T(2^m) = S(m)$

$S(m) = 2T(2$

$S(m) = 2S\left(\frac{m}{2}\right) + m$

$a = 2 \quad b = 2 \quad k = 1 \quad p = 0.$

$a \quad b^k$

$2 = 2^1$

case 2 :- a)

$T(n) = \Theta(n^{\log_b a} \log^{p+1} n)$

$= \Theta(m \log^0 n) = \Theta(m \log m)$

$m \log m$

$\log n \log \log n.$ $= \Theta(m) = \Theta(\log$

$= \Theta(\log m^m)$

~~$T(n) = \Theta(\log n)$~~

$T(n) = \log n \log(\log n)$

$2^m = n$

$\log n = m$

Q12) $T(n) = 2T(\sqrt{n}) + \log\log n$.

$n = 2^m$ and $\log n = m$

$T(2^m) = 2T(2^{m/2}) + \log m$.

put $T(2^m) = S(m)$

$S(m) = 2S\left(\frac{m}{2}\right) + \log m$

$a = 2,\ b = 2,\ k = 0,\ p = 1$

$a \quad b^k$

$2 \quad 2^0$

$2 > 1$

$\theta(n^{\log_b a})$

$\theta(m^{\log_2 2})$

$\theta(m)$

$\theta(\log n)$.

Q13) $T(n) = \sqrt{n}\, T(\sqrt{n}) + n$

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

assume $n = 2^m$

$T(2^m) = 2^{m/2} T(2^{m/2}) + 2^m$

$\frac{T(2^m)}{2^m} = \frac{T(2^{m/2}) \times 2^{m/2}}{2^m} + \frac{2^m}{2^m}$

$\frac{T(2^m)}{2^m} = T(2^{m/2}) \times 2^{\frac{m}{2} - m} + 1$

$\frac{T(2^m)}{2^m} = T(2^{m/2}) \times 2^{-m/2} + 1$

$\frac{m}{2} - m$

$\frac{m - 2m}{2}$

$-\frac{m}{2}$

$\frac{T(2^m)}{2^m} = \frac{T(2^{m/2})}{2^{m/2}} + 1$

$S(m) = \frac{T(2^m)}{2^m}$

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

$S(m_1) = \frac{T(2^{m_1})}{2^{m_1}}$

put $m_1 = \frac{m_2}{2}$

$S\left(\frac{m_2}{2}\right) = \frac{T(2^{m_2/2})}{2^{m_2/2}}$

$a = 1, \; b = 2, \; k = 0, \; p = 0.$

a	b^k
1	2^0

$1 = 1$

Q14)

case 2a)

$$T(n) = \theta(n^{\log_b a} \log^{p+1} n)$$

$$S(m) = \theta(m^{\log_b a} \log^{p+1} n)$$

$$= \theta(\log m)$$

$$\frac{T(2^m)}{2^m} = \theta(\log \log n)$$

$$\therefore \boxed{T(n) = \theta(n \log \log n)}$$

Q14) $T(n) = T(\sqrt{n}) + 1$

put $n = 2^m$

$T(2^m) = T(2^{m/2}) + 1$ $\quad \therefore T(2^m) = S(m)$

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

$a = 1, \; b = 2, \; k = 0, \; p = 0.$

$a = b^k$

case 2a) :- $T(n) = \theta(n b^{\log_b a} \log^{p+1} n)$

$S(m) = \theta(m^{\log_2 1} \log^1 m)$

$S(m) = \theta(\log m)$

$T(2^m) = \theta(\log \log n)$

$T(n) = \theta(\log(\log n))$

Q(Q15) find the optimal solⁿ for the Napsac in N=7, M=15, and the profits are

$p_1 \ldots p_7$ (i_1 … i_7)
10, 5, 15, 7, 6, 18, 3.

and their weights are

2, 3, 5, 7, 1, 4, 1.
(i_1 i_2 i_3 i_4 i_5 i_6 i_7)

$\frac{p_i}{w_i}$ =

items	$w_i x_i$	~~$p_i x_i$~~ $P + p_i + x_i$

table natar zahoda yeehya

	i_1	i_2	i_3	i_4	i_5	i_6	i_7
$\frac{p_i}{w_i}$ =	$\frac{10}{2}$	$\frac{5}{3}$	$\frac{15}{5}$	$\frac{7}{7}$	$\frac{6}{1}$	$\frac{18}{4}$	$\frac{3}{1}$
	= 5	1.66	3	1	6	4.5	3

arrange in decreasing order p_i/w_i

$\frac{p_i}{w_i} \Rightarrow$ 6, 5, 4.5, 3, 3, 1.66, 1.

$i_5, i_4, i_6, i_3, i_7, i_2, i_1$.

items	$w_i x_i$	$P = P + p_i x_i$
$\{i_1, i_2, i_3, i_4, i_5, i_6, i_7\}$ $\{0,0,0,0,1,0,0\}$	$(15-1)\times 1 = 14$	$6 \times 1 = 6$.
$\{1,0,0,0,1,0,0\}$	$= 14-2 = 12$	$10+6 = 16$.
$\{1,0,0,0,1,1,0\}$	$= 12-4 = 8$	$= 18+16 = 34$
$\{1,0,1,0,1,1,0\}$	$= 8-5 = 3$	$= 34+15 = 49$
$\{1,0,1,0,1,1,1\}$	$= 3-1 = 2$	$= 49+3 = 52$.
$\{1,1,1,0,1,1,1\}$	$= 2 - \frac{2\times 3}{3} = 0$	$= 52 + \frac{2}{3}\times 5$

$= 52 + \frac{10}{3}$

$= \frac{156+10}{3} = \frac{166}{3}$

$= 55.33$

55.3
3|166
15
16
15
10
9
10

psac instan
are

Q). let $S = A, B, C, D, E, F, G$ be a coll^n of object with profit weight values as follows $a = 12, 4$, $b = 10, 6$, $c = 8, 5$
p & w $d = 11, 7$, $e = 14, 3$,
$f = 7, 1$, $g = 9, 6$.

What is the optimal sol^n to the fractional napsak problem to S ; assume napsak capacity as a weight is 18 what is the complexity of this method.

⇒ Fractional napsack problem:-

	a	b	c	d	e	f	g
p =	12	10	8	11	14	7	9
w =	4	6	5	7	3	1	6
	i1	i2	i3	i4	i5	i6	i7

$M = 18$.

$\frac{p_i}{w_i} = \frac{12}{4}^3, \frac{10}{6}^5, \frac{8}{5}, \frac{11}{7}, \frac{14}{3}, \frac{7}{1}, \frac{9}{6}$.

$= 3, 2.5, 1.6, 1.57, 4.67, 7, 1.5$.
i1 i2 i3 i4 i5 i6 i7

decreasing order.

$\frac{p_i}{w_i}$ ⇒ 7, 4.67, 3, 2.5, 1.6, 1.57, 1.5
i6 i5 i1 i2 i3 i4 i7

$M = 18$.

1.6
5)8
-5
30
-25
7)11
-7
40
-35
50

sequence and profit also.

items	$w_i x_i$	$P_i = P + P_i x_i$
$\{a_1, a_2, a_3, a_4, a_5\}$	$18 - 1 = 17$	$7 \times 1 = 7$
② $\{0,0,0,0,0,1,0\}$	$17 - 3 = 14$	$7 + 14 = 21$
③ $\{0,0,0,0,1,0,0\}$	$14 - 4 = 10$	$21 + 12 = 33$
④ $\{1,0,0,0,1,1,0\}$	$10 - 6 = 4$	$33 + 10 = 43$
⑤ $\{1,1,0,0,1,1,0\}$	$4 - \frac{4 \times 5}{5} = 0$	$43 + \frac{4}{5} \times 8 = 49.4$

$43 + \frac{32}{5} = \frac{215 + 32}{5}$

Time complexity :-

① $\frac{P_i}{w_i} = O(n)$ +

② sorting $\Rightarrow O(n \log n)$

$\Rightarrow \boxed{O(n \log n)}$ → Time complexity.

☆ Job sequencing problem's (aim:- max profit)

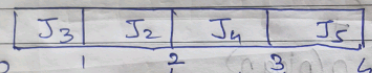
(Q1)

jobs	J_1	J_2	J_3	J_4	J_5
Deadlines	2	2	1	3	4
profits	20	60	40	100	80

Maximum deadline = 4.

Arrange jobs in decreasing order of their profit.

Jobs	J_4	J_5	J_2	J_3	J_1
Deadlines	3	4	2	1	2
profit	100	80	60	40	20



gannt chart.

profit = ~~100~~ + 40 + 60 + 100 + 80

= 280

Job sequence = {J_3, J_2, J_4, J_5}.

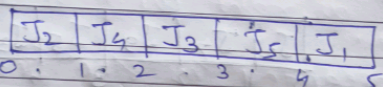
(Q2) given the jobs and their deadlines as shown table write the optimal shedule that gives the max profit ② are all the completed in a optimal shedule ③ what is the maximum earn profit.

Jobs	J_1	J_2	J_3	J_4	J_5	J_6
Deadline	5	3	3	2	4	2
profit	200	180	190	300	120	100

Max deadline = 5

Arrange jobs in decreasing order of their profit.

Jobs	J_4	J_1	J_3	J_2	J_5	J_6
deadline	2	5	3	3	4	2
profit	300	200	190	180	120	100



gantt chart

∴ J_6 is discarded.

profit = 180 + 300 + 190 + 120 + 200

= 990 → maximum profit.

Job sequence = $\{J_2, J_4, J_3, J_5, J_1\}$.

→ optimal shedule $(J_2, J_4, J_3, J_5, J_1)$.

Q3) Jobs J_1 J_2 J_3 J_4 J_5 J_6 J_7 J_8

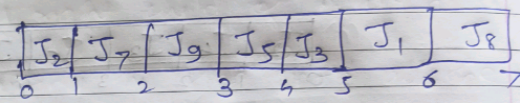
D:-	7	2	5	3	4	5	2	7
P:-	15	20	30	18	18	10	23	16

Max profit = ?

Max deadline = 7.

arrange in decreasing order.

Jobs	J_3	J_9	J_7	J_2	J_4	J_5	J_8	J_1
D:-	5	3	2	2	3	4	7	7
P:-	30	25	23	20	18	18	16	15



profit = 20 + 23 + 25 + 18 + 30 + 15 + 16

= 147 ⟶ Max profit.

Job sheduling = { J_2, J_7, J_9, J_5, J_3, J_1, J_8 }

Prefix code :- a value ≠ b values characters having diff.

* Huffman Coding :- (uses variable length code)

→ to compress the data.
→ encryption & decryption.

Huffman coding is a lossless data compression the idea is to assign the variable length code to input characters length of the assign code are based on the frequency of corresponding character the variable length code assign two input characters are prefix code means that the codes are assign in such a way that the code assign to one character is not the prefix of code assign to any other character the code length of the character depends on how frequently it occurs in the given text the character which occur most frequently get the smallest code and the character which occur least frequently get the largest code.

* There are two major steps of Huffan coding.

① → building a huffman tree from the input character.

② → assigning code to the character by traversing a huffman tree.

* Steps involved to construct the huffman tree.

① Create a leaf node for each character of the text, leaf node of character contains the occuring frequency of character.
② arrange all the nodes in increasing order of their frequency value.
③ Consider the first two node having minimum frequency create a new internal node the frequency of this new node is the sum of frequency of this two node, make the first node as left child and other node as right child of newly created node.
④ keep reapting this step 2&3 until all the nodes form a single tree.
⑤ Tree finally obtain is the desire huffman tree.

① Avg code length / character = $\frac{\sum(\text{freq} * \text{code length})}{\sum \text{freq}}$

② Total no. of bits in huffman encoded msg
= Total no. of character in the msg
× Avg code length per character