Date

Day

▷ Commonly Used Summations:

1. $1+2+3+\dots+n = \frac{n(n+1)}{2} = \Theta(n^2)$

2. $1^2+2^2+3^2+\dots+n^2 = \frac{n(n+1)(2n+1)}{6} = \Theta(n^3)$

3. $1^3+2^3+\dots n^3 = \frac{n^2(n+1)^2}{4} = \Theta(n^4)$

4. $1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\dots+\frac{1}{n} = \log_e n = \Theta(\log n)$

5. $^nC_0 + {^nC_1} + {^nC_2} + \dots {^nC_n} = 2^n = \Theta(2^n)$

▷ Comparasion of Functions:

Q: $f(n) = n^2 \log n$ & $g(n) = n(\log n)^{10}$

→ Apply Log

$\log[n^2 \log n]$ $\qquad$ $\log[n(\log n)^{10}]$

$\log n^2 + \log(\log n)$ $\qquad$ $\log n + \log(\log n)^{10}$

$2\log n + \log(\log n)$ $\qquad$ $\log n + 10\log(\log n)$

↳ so this is greater → ignoring the other terms as they are smaller.

Page No.

Date Day

Q: $f(n) = 3n^{\sqrt{n}}$ $g(n) = 2^{\sqrt{n}\log_2 n}$

$\log(3n^{\sqrt{n}})$ $\log(2^{\sqrt{n}\log_2 n})$

$3n^{\sqrt{n}}$ $2^{\log_2 n^{\sqrt{n}}}$ $a^{\log_a b} = \log_a b^1$

$3n^{\sqrt{n}}$ $2(n^{\sqrt{n}})^{\log_2 2}$ $a^{\log_c b} = b^{\log_c a}$

$3n^{\sqrt{n}}$ > $n^{\sqrt{n}}$

Q: $f(n) = n^{\log n}$ $g(n) = 2^{\sqrt{n}}$

$\log(n^{\log n})$ $\log 2^{\sqrt{n}}$

$\log n \log n$ $\sqrt{n}\log_2 2$

$\log^2 n$ < $\sqrt{n}$

Q: $f(n) = 2^{\log n}$ $g(n) = n^{\sqrt{n}}$

$\log n \log_2 2$ $\sqrt{n}\log n$

$\log n$ < $\sqrt{n}\log n$

Q: $f(n) = 2^n$ $g(n) = 2^{2n}$

$\log_2 2^n$ $\log_2 2^{2n}$

$n\log_2 2$ $2n\log_2 2$

n < $2n$

Date Day

Q: $g_1(n) = \begin{cases} n^3 & n < 100 \\ n^2 & n \geq 100 \end{cases}$ $g_2(n) = \begin{cases} n^2 & n < 10{,}000 \\ n^3 & n \geq 10{,}000 \end{cases}$

$g_1 > g_2$ 100 $g_1 = g_2$ 10,000 $g_2 > g_1$

less than 100 $g_1 = n^3$ & $g_2 = n^2$ both are n^2 greater than 10,000 $g_2 = n^3$ & $g_1 = n^2$

10,000 k baad g_2 hi hamesha baara hai or usse phele kabhi g_1 or kabhi dono equal hai. So $g_1(n) = O(g_2(n))$

$g_2 > g_1$

" BEST, WORST, AND AVERAGE CASE

▸ Linear Search:

0	1	2	3	4	5	6
11	6	9	3	5	10	13

Order of (1)

Best case ($x = 11$) → $O(1)$ 0th index k liye 1 comparision 1st k liye 2, 2nd k liye 3rd and so on.

Worst case ($x = 13$) → $O(n)$

Avg case ($x = 3$) → $\frac{\text{Sum of all possible cases}}{n} = \frac{1+2+3+\dots+n}{n}$

$\rightarrow \frac{\cancel{n}(n+1)/2}{\cancel{n}} \rightarrow \frac{(n+1)}{2}$

Date
Day

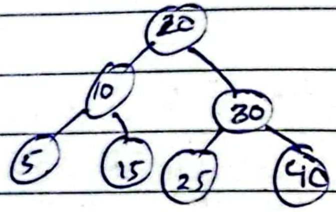
→ Ab h time Complexities ko hum kisi bhi notation se represent kar sake hai.

Best Case → $O(1)$, $\Omega(1)$, $\Theta(1)$

Worst Case → $O(n)$, $\Omega(n)$, $\Theta(n)$

Avg Case → $O(n)$, $\Omega(n)$, $\Theta(n)$

▷ Binary Search Tree

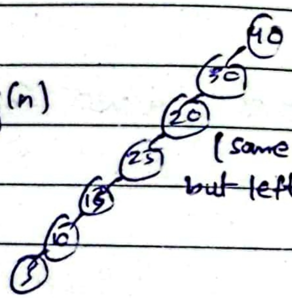


search for root element

Best Case: $O(1)$

Worst Case: $O(\log(n))$

↳ search for leaf element = heigh of tree = $\log(n)$



[same tree ~~but left skewed~~]

Best Case: $O(1)$ ←

Worst Case is search for leaf ele

but minimum Worst Case Time = $O(\log(n))$

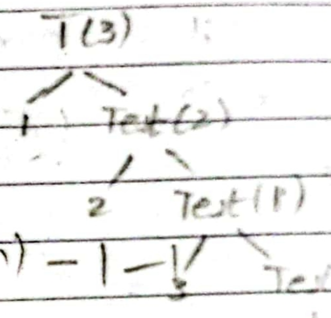
maximum " " = $O(n)$

Date
Day

" ~~Divide & Conqure~~ "

" Reccurence Relation "

```
void Test (int n)
{
    if (n>0) {
        print(n)
        Test(n-1)
    }
}
```



$(n-1) - 1 - 1$

$f(n) = n+1 \rightarrow O(n)$

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

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

$T(n) = T(n-2) + 2C \quad \Rightarrow T(n-2) = T(n-3)+C$

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

$\vdots$

$T(0) = 1$

$n - k = 0$

$n = k$

$\boxed{T(n) = T(n-k) + k}$

When $n=0$ the edge stops & at $n=0$ time complexity = 1

i.e $T(n) = \begin{cases} 1 & n=0 \\ T(n-1)+1 & n>0 \end{cases}$

solving this will give

$1 = T(0) + K$

$= T(0) + kC$

$= 1 + nC$ $\quad O(n)$

Date ______ Day ______

As we get the general exp: $T(n) = T(n-k) + k$

Now assume that we reach at $n=0$

then $n-k=0$

$n=k$

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

$= T(0) + n$

$\boxed{T(n) = 1+n}$ → we got the same ans by tree also.

$\boxed{\Theta(n)}$

```
void Test(int n){          → T(n)
    if(n>0){               → 1
        for(i=0; i<n; i++) → n+1
            printf("%d", n); → n
        Test(n-1)          → T(n-1)
    }
}
```

$T(n) = T(n-1) + 2n + 2$ → iski jaga 'n' consider krlenge bcz it will be easy to solve.

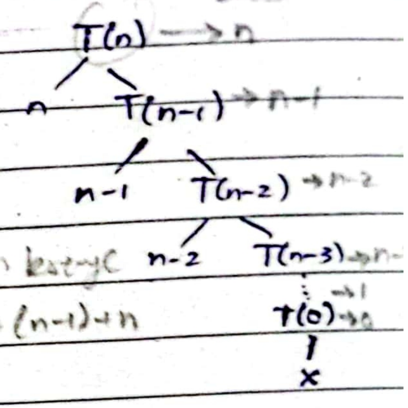
$T(n) = T(n-1) + n$

Page No. ______

Date ______ Day ______

Now we get the recurrence relation as: ye n times chala

$$T(n) = \begin{cases} 1 & n=0 \\ T(n-1)+n & n>0 \end{cases}$$



In sab ki complexities ko sum karenge

$0+1+2+3+\dots(n-3)+(n-2)+(n-1)+n$

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

$\Theta(n^2)$

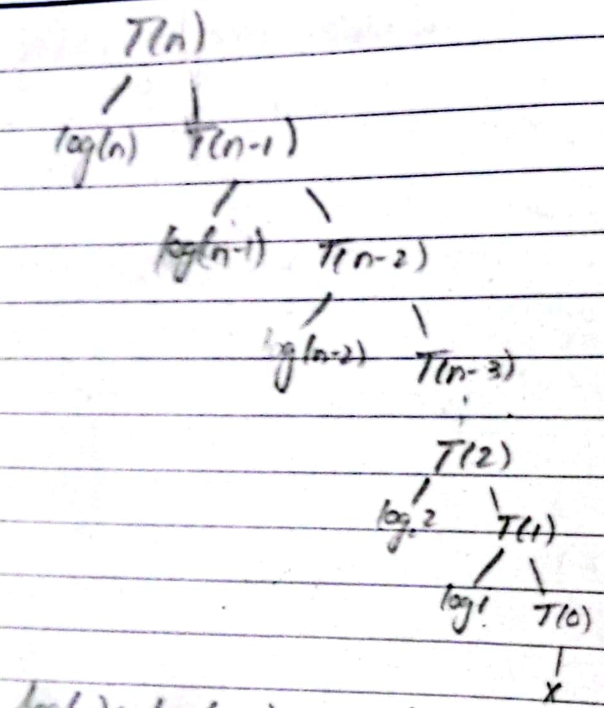
```
void Test(int n){            → T(n)
    if(n>0){                 → 1
        for(i=1; i<n; i=i*2)
            printf(n)        → log_2 n
        Test(n-1)            → T(n-1)
    }
}
```

$$T(n) = \begin{cases} 1 & n=0 \\ T(n-1)+\log n & n>0 \end{cases}$$

Page No. ______

Date

Day



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

$\log[n \times (n-1) \times \dots 2 \times 1]$

$\log[n!]$

$\log(n^n) \quad ; n! = n^n$

$O(n\log n)$

Day

Date

▷ Shortcut Technique: → this whole thing is gona repeat n times so ⊗ by n

$1 \times n$

$T(n) = T(n-1) + 1 \longrightarrow O(n)$

$T(n) = T(n-1) + n \longrightarrow O(n^2)$ $n \times n$

$T(n) = T(n-1) + \log n \longrightarrow O(n\log n)$ $\log n \times n$

$T(n) = T(n-1) + n^2 \longrightarrow O(n^3)$ $n^2 \times n$

$T(n) = T(n-2) + 1 \longrightarrow \frac{n}{2} \rightarrow O(n)$

$T(n) = T(n-100) + n \longrightarrow \frac{n}{100} \longrightarrow O(n^2)$

↳ isse kuch farq nhi parega. Aisa bas itna mtlab hai ki ye ~~har step then~~ 100 steps lega har call par.

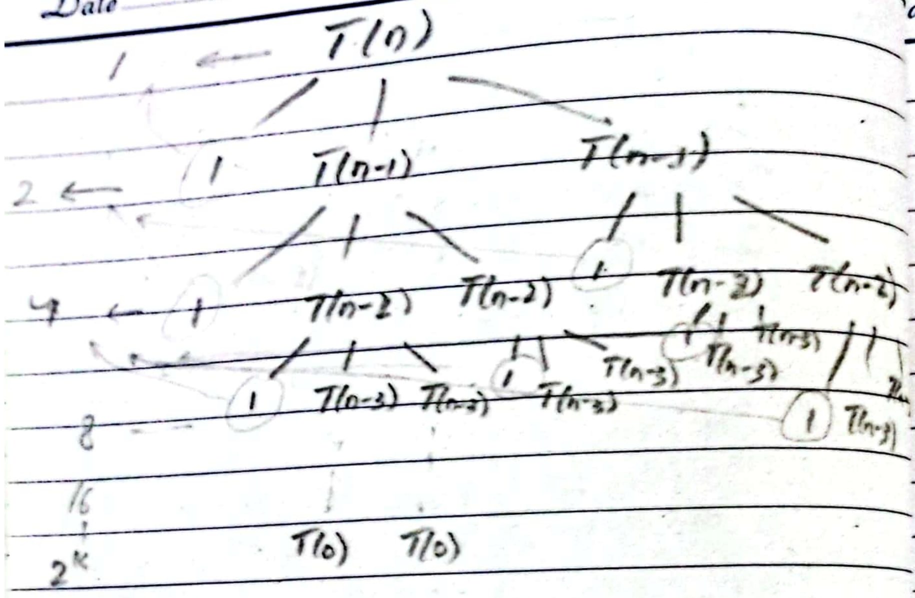
$T(n) = 2T(n-1) + 1 \rightarrow O(n)$ X

bcz of this

```
Algorithm Test (int n)          → T(n)
    if (n>0) {
        printf("%d ", n);       → 1
        Test(n-1);              → T(n-1)
        Test(n-1);              → T(n-1)
    }
}
```

$$T(n) = \begin{cases} 1 & n=0 \\ 2T(n-1)+1 & n>0 \end{cases}$$

Date



→ Has level ka individual time dekhenge or phir sub levels k time ko add karenge.

$1+2+4+8+16+ \dots 2^k$ = sum of geo series

$= \frac{a(r^n-1)}{r-1}$

$= \frac{1(2^k-1)}{2-1}$

$= 2^k - 1$

Assume that $n=0$

$n=k$

$2^n - 1 \rightarrow O(2^n)$

Date

- Substitution Method:

$T(n) = 2T(n-1)+1 \longrightarrow ①$

$T(n) = 2[2T(n-2)+1]+1 \rightarrow$

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

$T(n) = 2^2[2T(n-3)+1]+2+1 \rightarrow$

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

$T(n) = 2^3[2T(n-4)+1]+2^2+2+1 \rightarrow$

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

⋮

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

Assume $n=0$

$T(n) = 2^n T(0) + 2^{n-1} + 2^{n-2} + \dots 2^2+2+1$

G.P series $\nearrow 1+2+2^2+2^3+\dots 2^{n-2}+2^{n-1}+2^n$

$=$ ~~$2^{n}+1+2^2-4$~~

$= 2^n - 1$

$O(2^n)$

Date

Day

- Fact (n) { $\rightarrow T(n)$
  - if $(n \le 1)$ $\rightarrow 1$
    - return 1; $\rightarrow 1$ } c (constant)
  - else   ye multiplication wale time ko bhi 'c' mein include kr
    - return (n * Fact (n-1)) $\rightarrow T(n-1)$

}

$$T(n) = \begin{cases} 1 & n=1 \\ C + T(n-1) & n \ge 1 \end{cases}$$

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

$T(n) = T(n-2) + 2C$ $\qquad T(n-2) = T(n-3) + C$

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

$T(n) = T(n-k) + kC$

Assume we have reached termination condition.

i.e $T(1) = 1$ $\quad \therefore 1 = n-k \Rightarrow k = n-1$

$T(n) = T(1) + kc$

$T(n) = 1 + (n-1)c \rightarrow O(n)$

Date

Day

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

$T(n) = 2T(n/2) + n$ $\qquad , T(n/2) = 2T(n/2^2) + n/2$

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

$T(n) = 2^2 T(n/2^2) + 2n$ $\qquad , T(n/2^2) = 2T(n/2^3) + n/2^2$

$= 2^2[2T(n/2^3) + n/2^2] + n + n$

$T(n) = 2^3 T(n/2^3) + 3n$

$\vdots$

$T(n) = 2^k T(n/2^k) + kn$

Assume we reach termination condition.

ie $T(1) = 1$ $\quad \therefore \frac{n}{2^k} = 1 \Rightarrow n = 2^k \Rightarrow k = \log n$

$\rightarrow 2^k = n$

$T(n) = 2^k T(1) + kn$

$= n \times 1 + \log n \times n$

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

2^3 $\frac{2^6}{2^1}$

Date ______ Day ______

Q: $T(n) = \begin{cases} 1 & n=1 \\ 8T(n/2) + n^2 & n>1 \end{cases}$

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

$= 8[8T(n/2^2) + n^2/2^2] + n^2$

$T(n) = 8^2 T(n/2^2) + 2n^2 + n^2$, $T(n/2^2) = 8T(n/2^3) + \frac{n^2}{2^4}$

$= 8^2[8T(n/2^3) + \frac{n^2}{2^4}] + 2n^2 + n^2$

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

$T(n) = 8^3 T(n/2^3) + 2^2 n^2 + 2n^2 + n^2$

⋮

$T(n) = 8^k T(n/2^k) + 2^{k-1} n^2 + \cdots 2^2 n^2 + 2n^2 + n^2$

Assum,

$T(1) = 1$, $n/2^k = 1 \Rightarrow n = 2^k \Rightarrow k = \log n$

$8^k = (2^3)^k$

$\to \frac{a(r^n - 1)}{r-1} = n^2(2^k - 1)$

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

$n^3 [2^{k-1} + 2^{k-2} \ldots 2^0]$

$\because 2^k = n$ $\quad n^2[2^k \cdot 2^{-1} + 2^k \cdot 2^{-2} \ldots 2^0]$

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

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

$T(n) = 2n^3 - n^2 \longrightarrow O(n^3)$

Date ______ Day ______

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

$T(n) = 2T(n/2) + n\log n$, $T(n/2) = 2T(n/2^2) + \frac{n}{2}\log\frac{n}{2}$

$= 2[2T(n/2^2) + n/2 \log(n/2)] + n\log n$

$T(n) = 2^2 T(n/2^2) + n\log(n/2) + n\log n$; $T(n/2^2) = 2T(n/2^3) + \frac{n}{2^2}\log\frac{n}{2^2}$

$= 2^2[2T(n/2^3) + \frac{n}{2^2}\log\frac{n}{2^2}] + n\log(n/2) + n\log n$

$T(n) = 2^3 T(n/2^3) + n\log\frac{n}{2^2} + n\log\frac{n}{2} + n\log n$

⋮

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

$\to \frac{n}{2^{k-1}} = \frac{n}{2^k \cdot 2^{-1}} = \frac{n}{2^k} \cdot 2 = 1 \cdot 2$

Assume,

$T(1) = 1$, $n/2^k = 1 \Rightarrow n = 2^k \Rightarrow k = \log n$

$= n + n[\log_2(2) + \log_2(2^2) + \log_2(2^3) + \ldots + \log\frac{n}{2} + \log n]$

$= n + n[1\times1 + 2\times1 + 3\times1 + \cdots + (\log n - \log_2(2)) + \log n]$

$= n + n[\underbrace{\qquad\qquad (\log n - 1) + \log n}_{\text{Sum of } \log n \text{ natural no.s} = \frac{\log n(\log n+1)}{2}}]$

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

$O(n(\log n)^2)$

n	n^2+n	n^3
1	2	1
2	6	8
3	12	27

$n^2+n \le 1n^3$

$n^3-n^2-n \le 0$

Day

Date

$$T(n)=\begin{cases} 1 & n=1 \quad n\le 0 \\ 2T(n/2)+n/\log n & n>1 \end{cases}$$

$T(n)=2T(n/2)+n/\log n$, $T(n/2)=2T(n/2^2)+\frac{n/2}{\log n/2}$

$=2[2T(n/2^2)+\frac{n}{2\log n/2}]+\frac{n}{\log n}$

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

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

$T(n)=2^3T(n/2^3)+\frac{n}{\log n/2^2}+\frac{n}{\log n/2}+\frac{n}{\log n}$

$\vdots$

$T(n)=2^kT(n/2^k)+\cancel{\log}\, n\left[\frac{1}{\log n/2^{k-1}}+\frac{1}{\log n/2^{k-2}}+\dots\dots\frac{1}{\log n/2}+\frac{1}{\log n}\right] \rightarrow 1$

$\frac{n}{2^{k-1}}=\frac{n}{2^k\cdot 2^{-1}}=\frac{n}{2^k}\cdot 2 \rightarrow 2$

Assume that $T(1)=1$, then

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

$c=2$

$2>2.$

n	n^2+n	$2n^3$
1	2	2
2	6	16

order ↓ order detail

Day

Date

eq② ⇒ $\frac{n}{2^k}\cdot 2=2$ put in ①

eq① ⇒ $2^kT(n/2^k)+n\left[\frac{1}{\log_2 2}+\frac{1}{\log_2 2^2}+\dots+\frac{1}{\log 2}+\frac{1}{\log \frac{n}{2}}\cdot\frac{1}{\log n}\right]$

$2^kT(n/2^k)+n\left[\frac{1}{1}+\frac{1}{2}+\frac{1}{3}\dots+\frac{1}{\log n-1}+\frac{1}{\log n}\right]$

$\rightarrow 1+1/2+1/3+1/4+\dots 1/n=\log n$

$=2^kT(n/2^k)+n\log(\log n)$

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

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

$2^{-1}(2^{-n}-1)$

Date

Day

$$T(n) = \begin{cases} 2 & 0 \le n \le 2 \\ 2T(\sqrt{n}) + c & n > 2 \end{cases}$$

$2^{-1} - 1$

2^{-1}

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

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

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

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

$T(n) = 2^3 T(n^{1/8}) + 4c + 2c + c \rightarrow 7c$

$\vdots$

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

Assum,

$T(2) = 2 \Rightarrow n^{1/2^k} = 2$, $\frac{1}{2^k} \log n = \log_2 2$

$\log n = 2^k$

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

$k = \log(\log n)$

$T(n) = \log n \, T(2) + 2^k \cdot 2^{-1} c + 2^k \cdot 2^{-2} c + \cdots + c$

$= \log n \cdot 2 + 2^k c \,[2^{-1} + 2^{-2} + 2^{-3} \cdots + 2^{-n}]$

$2^0 + 2^1 + 2^2 \cdots\cdots 2^{k-1}$ → GP Series sum

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

$= 2\log n + (\log n - 1)c \rightarrow O(\log n)$

Date

Day

" MASTER'S THEOREM "

" Decreasing Function "

$T(n) = T(n-1) + 1 \rightarrow O(n)$

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

$T(n) = T(n-1) + \log n \rightarrow O(n \log n)$

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

$T(n) = 3T(n-1) + 1 \rightarrow O(3^n)$

$T(n) = 2T(n-1) + n \rightarrow O(n \times 2^n)$

$T(n) = aT(n-b) + f(n)$

$a > 0 \quad b > 0$ & $f(n) = O(n^k)$ where $k \ge 0$

if,

$a = 1 \rightarrow O(n \times f(n))$ or $O(n^{k+1})$ → a^0

$a > 1 \rightarrow O(a^n \times f(n))$ or $O(a^{n/b} \times n^k)$

$a < 1 \rightarrow O(f(n))$ or $O(n^k)$

Date ______ Day ______

"Dividing Function"

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

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

Case #01: if $a > b^k$ then, $T(n) = \Theta(n^{\log_b a})$

Case #02: if $a = b^k$ then,

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

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

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

Case #03: if $a < b^k$ then,

i- $p \geq 0$ then, $T(n) = \Theta(n^k \log^p n)$

ii- $p < 0$ then, $T(n) = \Theta(n^k)$

Date ______ Day ______

$T(n) = 4T(n/2) + n$

$a = 4,\ b = 2,\ k = 1,\ p = 0$

$\Theta(n^{\log_b a}) = n^{\log_2 4} = n^2$

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

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

$\Theta(n^{\log_b a} \log^{p+1} n) = \Theta(n^{\log_2 2} \log^{0+1} n)$

$\Theta(n \log n)$

$T(n) = 2T(n/2) + n/\log n$

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

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

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

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

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

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

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

$\Theta(n^k \log^p n) = \Theta(n^2)$

Date   Day

→ Relations on which Masters theo not applicable

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

↳ here must be a constant not a func

$T(n) = 64\,T(n/2) - n^2$

↳ should be +

$T(n) = T(n/2) + \sin n$

↳ sin & cos have +/- graph

$T(n) = 2T(n/2) + 1/n$ → n have -ve power

$T(n) = T(\sqrt{n}) + \log n$ → isee hum in masters theorem ke format mein convert krna hai.

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

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

$T(2^m) = T(2^{m/2}) + m\log_2 2$

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

Now let,

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

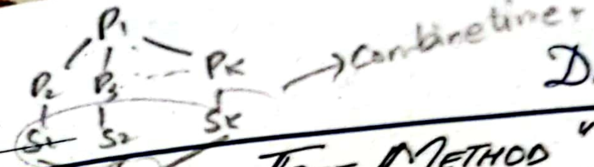
$S(m) = S(m/2) + m$

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

$\Theta(m)$ case 3

$\Theta(\log n)$ → $2^m = n$, $m = \log n$

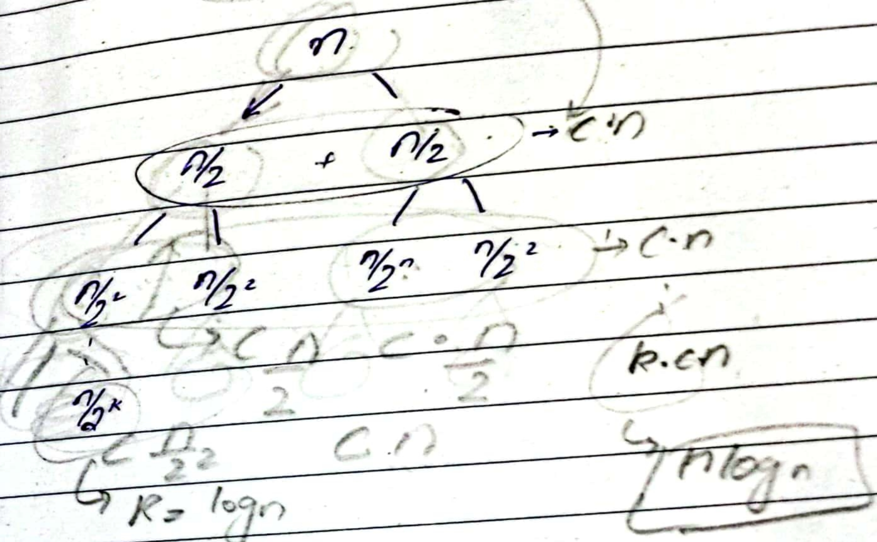
Date   Day



" RECURSIVE TREE METHOD "

Divide & Conquer

$$T(n) = \begin{cases} 1 & n=1 \\ 2T(n/2) + c\cdot n \end{cases}$$ → ye combine karne ka time hai



Date ______ → ek problem ko 5 sub problems mein divide karne   Day ______

$$T(n) = \begin{cases} 1 & n=1 \\ 5T(n/5) + cn \end{cases}$$

→ size of each problem

→ jitni ye value hogi utni branches bnengi

→ no. of sub problem

n

$n/5 \quad n/5 \quad n/5 \quad n/5 \quad n/5 \Rightarrow c \cdot n$

$\frac{n}{5^2} \; \frac{n}{5^2} \; \frac{n}{5^2} \; \frac{n}{5^2} \; \frac{n}{5^2} = \frac{c \cdot n}{5}$ $\Rightarrow \frac{5cn}{5} = cn$

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

$\frac{n}{5^k}$ → $k = \log_5 n$ → kcn

$\boxed{n \log n}$

$$T(n) = \begin{cases} 1 & n=1 \\ 3T(\lfloor n/4 \rfloor) + c \cdot n^2 \end{cases}$$

$T(n/4) = 3T(n/16) + c(n/4)$ $+ cn/4^2$

$T(n/4^2) = 3T(n/4^3) + n/4^4$

n

$n/4 \quad n/4 \quad n/4$ $\Rightarrow c \cdot n^2$

$n/4^2 \; n/4^2 \; n/4^2 \Rightarrow \frac{c \cdot n^2}{16}$ $\Rightarrow 3 \cdot \frac{cn^2}{16}$

$\Rightarrow 9 \cdot \frac{cn^2}{16^2}$

$\frac{n}{4^k}$ → $k = \log_4 n$

Date ______   Day ______

$$cn^2 + \frac{3n^2}{4^2} + \frac{3^2 n^2}{4^4} + \frac{3^3 n^2}{4^6} + \dots$$

$$r = \frac{3^2/4^4}{3/4^2} = \frac{3}{4^2}$$

$$cn^2 \underbrace{\left[1 + 3/4^2 + 3^2/4^4 + \dots\right]}_{\text{infinite G.P} = \frac{1}{1-r}}$$

$$= \frac{1}{1 - 3/4^2} = \frac{13}{16}$$

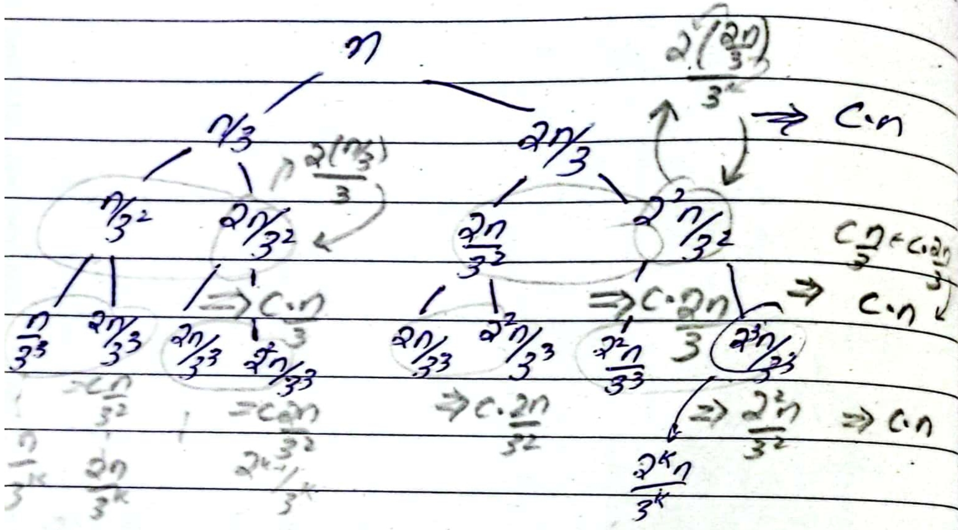
$cn^2 \left(\frac{13}{16}\right)$ constant so eliminate.

→ Vertical sum agr hamesha constant aarha hai har level par tab hi sirf height of tree se ⊗ hoga.

→ Agr har level par change horha hai to height se ⊗ nhi krenge balke jo bhi series ban rhi hai uski general form nikaalenge.

Date ______ Day ______

$T(n) = T(n/3) + T(2n/3) + cn$



So yaha par us leaf ki height ko consider karenge jo sabse last mein end hogi.

$\frac{n}{3}$ ⇒ yaha par hum n ko 3 se divide karhe

$\frac{n}{3/2}$ ⇒ yaha hum n ko 3/2 se divide karhe mtlb ye der se khatam hoga q k n ko choti value se divide karhe.

∴ we will consider,

$$\frac{2^k n}{3^k} = 1, \quad n = \frac{3^k}{2^k} \Rightarrow n = \left(\frac{3}{2}\right)^k$$

$$\log n = k \log_{3/2} \left(\tfrac{3}{2}\right)^{1}$$

$$k = \log_{3/2} n$$

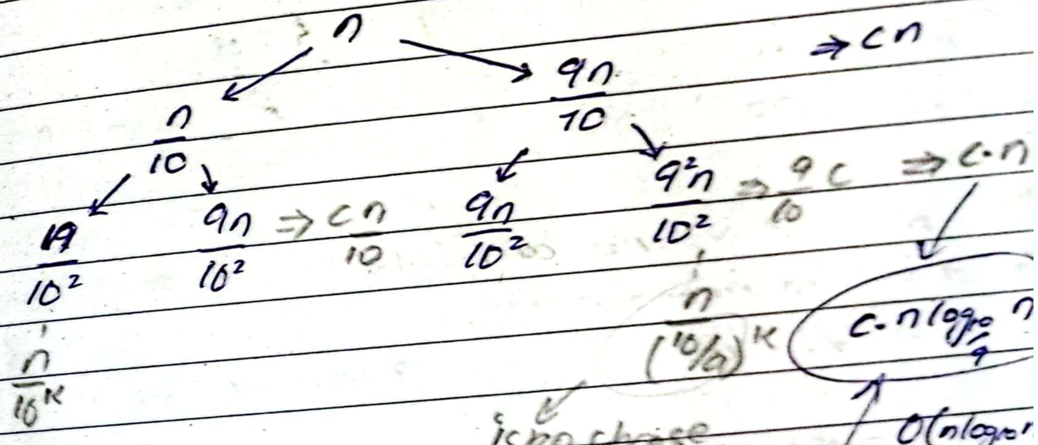
Day ______

Date ______

As we are getting same sum on each level

therefore $\log_{3/2} n \times cn = n \log n$

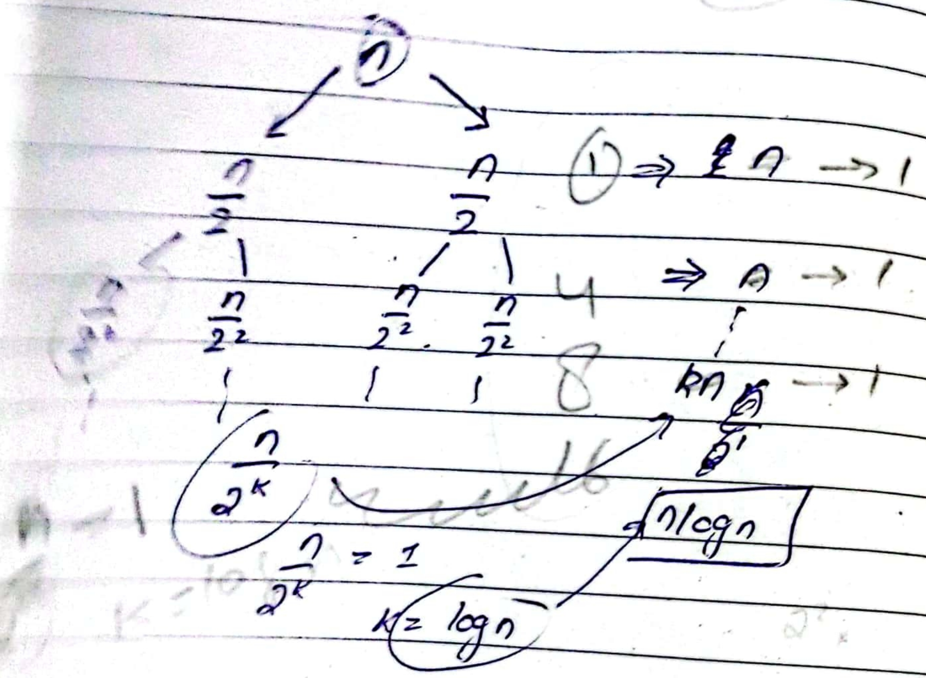
$T(n) = T(n/10) + T(9n/10) + cn$



isko choose karenge as denominator is small.

$$\frac{n}{(10/9)^k} = 1 \Rightarrow k = \log_{10/9} n$$

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



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

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

$T(n) = 2[T(n/2^2) + 1]$

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

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

$= 2^3 T(n/2^3) + 2^2 + 2$

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

$\frac{n}{2^k} = 1, \quad 2^k = n, \quad k = \log n \qquad 1(2^k - 1)$

$n + (n-1)$

$= 2n$

Date

"LOOP INVARIANCE"

- A tool used to prove the correctness of an algorithm, specifically in regards to a loop.
- It is a property or set of property of type boolean and must be aligned with the goal of algo.
- Must be true at initialization, maintainance, termination.

→ Initialization: The invariant holds prior to the first iteration of the loop.

→ Maintainance: Assume invariant holds before an iteration k, then it must hold before iter. $k+1$.

→ Termination: The invariant holds when loop terminates.

→ If invariant holds at initi., maint. and termi., it confirms that the algo. performs its function correctly.

Date

Day

→ Find max in an array:

```
def getmax(a):
    max = a[0];

    for i in range(1, len(a)):
        if (a[i] > max)
            max = a[i];

    return max;
```

Initialization:
↳ $i = 1$, $max = a[0]$

Maintenance:
↳ Assuming loop invariants holds at iteration k, the max is the maximum value in a[0... k-1]
↳ In next iter. k+1, the code compares and possibly updates max with a[k]. Max is the maximum in a[0...n]

Termination:
i = len(a), max is maximum value in a[0... len(a)-1]

Date

Day

```
void search(int arr[]){

    for(int i=0; i<length; i++)
        if(a[i] == val)
            return true;

    return false;
}
```

~~Stat~~ At the start of each iteration of the loop (indexed by "i"), the subarray A[1 ... i-1] doesnot contain the targeted element 't', and the remaining sub-array A[i... n] has not been checked

Initialization:
Before the 1st iteration (i = 1), the sub array A[0... 1] is empty, and the remaining sub-array A[1 ... n] contains all elements of the array. Thus invariant holds true.

Maintenance:

If invariant is true before an iteration, it remains true before the next iter. Before the i^{th} iter., the sub-array A[1 ... i-1] does not contains 't'. If A[i] is equal to 't' the fun() return true else invariant holds true for next iteration as the subarray [1 ... i] still doesnot contains 't'

Termination:

a) if it terminates with 'true' result it means 't' found in the array.

b) it terminates ~~fi~~ without finding 't', it means that 't' is not peresent in array.

"Guess & Test"

$$T(n) = 3T(n/2) + n^2$$

Guess #01: $T(n) = O(n)$

$$T(n) \le c \cdot n$$

$$T(n/2) \le c \cdot \frac{n}{2}$$

$$= 3\frac{c \cdot n}{2} + n^2$$

$$= \frac{3cn + n^2}{2}$$

$$1.5cn + n^2 \le n \quad \times$$

Guess #02: $T(n) = O(n^2)$

$$T(n) \le c \cdot n^2$$

$$T(n/2) \le c \cdot \frac{n^2}{4}$$

$$= 3\frac{c \cdot n^2}{4} + n^2$$

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

for $c \ge 4$

$3n^2 + 4n - 3 \le 5n^2$

Date $\frac{4}{17}\frac{4}{16}$ Day

Q1 MID-I (2020)

(i) $4^3n + 4^2n + 4n = \Theta(n^2)$

$4^3n + 4^2n + 4 \le c_1 n^2$ $\quad n^2 - 4^3n - 4^2n - 4 \ge 0$

at $c_1 = 10$, $n_0 \ge 9$

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

No c value exist

$2^n + n^2 \ge 2^n$

Master Theorem:

$T(n) = 2T(n/2) + n\log n$

$a = 2$, $b = 2$, $K = 1$, $p = 1$

Case#02: as $a = b^k$

$\because p > -1$

$\therefore n^{\log_b a} \log^{p+1} n = \Theta(n\log^2 n)$

$T(n) = 4T(n/2) + 4$

$a = 4$, $b = 2$, $K = 0$, $p = 0$

Case#01: as $a > b^k$, $4 > 2^0$ true

$\Theta(n^{\log_b a}) = \Theta(n^2)$

Page No.

Day

Date

$T(n) = 5T(n/1) + n^2$

$a = 5$, $b = 1$, $k = 2$, $p = 0$

As $a > b^k$ i.e. $5 > 1^2$

Case # 01,

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

Not applicable as $b \le 1$

$T(n) = 2T(n/2) + n \times 2^3$

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

as $a = b^k$, $2 = 2^1$

Case #02, as $p = 0$

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

Q#3: $T(n) = 3T(n/3) + n^3$ $\quad T(1) = 1$

Substitution:

$T(n) = 3T(n/3) + n^3 \longrightarrow ①$

$T(n/3) = 3T(n/3^2) + n^3/3^3$

$T(n) = 3^2 T(n/3^2) + n^3/3^2 + n^3$

$T(n/3^2) = 3T(n/3^3) + n^3/3^6$

$T(n) = 3^3 T(n/3^3) + n^3/3^4 + n^3/3^2 + n^3$

$(3^2)^2 \cdot 9^2 \cdot 9$

$T(n) = 3^3 T(n/3^3) + \frac{n^3}{9^2} + \frac{n^3}{9} + n^3$

Page No.

Date Day

$$T(n) = 3^k\, T(n/3^k) + n^3\left[\frac{1}{9^{k-1}} + \frac{1}{9^{k-2}} + \dots \frac{1}{9^2} + \frac{1}{9^1} \dots\right]$$

$$\frac{1}{9^0} + \frac{1}{9^1} + \dots \frac{1}{9^k} = \frac{a(r^n - 1)}{r - 1}$$

$$= \frac{1((1/9)^k - 1)}{1/9 - 1}$$

$$T(n) = 3^k\, T(n/3^k) + \frac{8}{9} n^3\left[(1/9)^k - 1\right]$$

Assum $T(1)$,

then $1 = n/3^k$, $3^k = n$, $k = \log_3 n$

$$T(n) = n \times 1 \mp \frac{8n^3}{9}\left[\frac{1}{9^k} - 1\right]$$

$(3^2)^k$ $(3^k)^2$ n^2

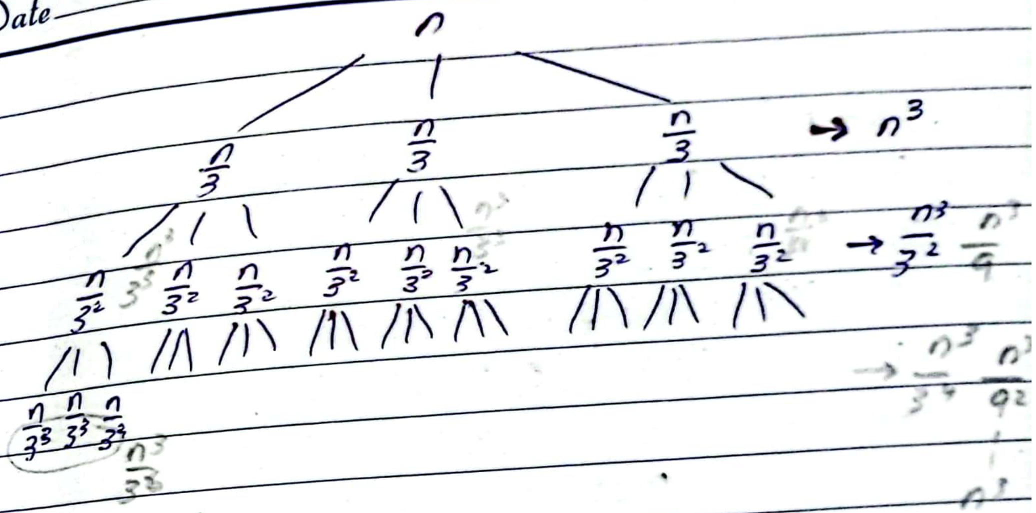
$$9^k = (3^2)^k = (3^k)^2 = n^2$$

$$T(n) = n \mp \frac{8n^3}{9}\left(\frac{1}{n^2} - 1\right)$$

$$T(n) = n \mp \frac{8}{9} n(1 - n^2)$$

$$T(n) = n \mp \frac{8}{9} n + \frac{8}{9} n^3 \qquad \Theta(n^3)$$

Date Day



→ Height of tree:

For $T(1) = \frac{n}{3^k} = 1$

$k = \log(n)$

→ Sum of levels:

$$n^3 + n^3/9 + n^3/9^2 + \dots n^3/9^k = \frac{a(r^n - 1)}{r - 1}$$

$$= \frac{n^3((1/9)^k - 1)}{\frac{1}{9} - 1}$$

$$= \frac{-8n^3}{9}\left(\frac{1}{9^k} - 1\right)$$

$$= \Theta(n^3)$$

$4 > 4^2$

Date ____ Day

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

$\frac{n^2}{4}$

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

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

$T(n) = 4[4T(n/4^2) + n^2/4^2] + n^2$

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

$T(n/4^2) = 4T(n/4^3) + n^2/4^4$

$T(n/4) = 4^2[4T(n/4^3) + n^2/4^4] + n^2/4 + n^2$

$T(n) = 4^3 T(n/4^3) + n^2/4^2 + n^2/4 + n^2$

$\vdots \quad \cancel{= 4^3 T(n/4^3) + n^2/16^2 +}$

$T(n) = 4^k T(n/4^k) + n^2/4^{k-1} + n^2/4^{k-2} + \dots n^2/4 + n^2/4^0$

$T(n) = 4^k T(n/4^k) + n^2[1/4^0 + 1/4^1 + \dots 1/4^{k-1}]$

$\rightarrow$ Sum of G.P $= \frac{1}{1-r}$

$T(n) = 4^k T(n/4^k) + n^2 [1/(1-1/4)]$

$T(n) = 4^k T(n/4^k) + \frac{4}{3} n^2$

- $T(1) = 1 \Rightarrow \frac{n}{4^k} = 1, \quad k = \log_4 n$

$T(n) = n + \frac{4}{3}n^2 \qquad \theta(n^2)$

Page No. ____

PAPE - 2017 (MID-1) Day

Date ____

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

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

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

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

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

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

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

$n^{1/2} \times n^{1/2^2} = ?$

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

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

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

$\frac{3}{2^2} \quad \frac{1}{2^1}$

$2+1$

$3+4$

$4+$

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

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

$\hookrightarrow T(n) = n^{1/2^3}[n^{1/2^3} T(n^{1/2^3}) + n^{1/2^2}] + n^{1/2^2} + n$

$\cancel{T(n) = n^{1/2^6} T(n^{1/2^3}) + n^{1/2^5} + n^{1/2^2} + n} \qquad n^{7/2^3} T(n^{1/2^3}) + n + n + n$

$\vdots \hookrightarrow$

$T(n) = n^{1-1/2^k} T(n^{1/2^k}) + kn \qquad n^{15/2^4} T(n^{1/2^4}) + 4n$

Page No. ____

Date

Day

$T(1)=1$ $\quad n^{\frac{1}{2^k}} = 2$, $\frac{1}{2^k}\log n = 1$, $\log n = 2^k$

$k = \log(\log n)$

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

Q# Master's Theorem

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

$a=4$, $b=4$, $k=2$, $p=0$

as $a < b^k$, $4<4^2$

Case#03: $\Theta(n^k \log^p n) = \Theta(n^2)$

$T(n) = T(n/2) + 8$

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

as, $a = b^k$, $1 = 2^0$

Using Case#02:

as $p=0$

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

Date

Day

$T(n) = 16T(n/2) + 16n$

$a=16$, $b=2$, $k=1$, $p=0$

as $16>2$, $a>b^k$

Using case#01

$\Theta(n^{\log_b a}) = \Theta(n^4)$

$T(n) = 2T(n/2) + 2n\log n$

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

as $a=b^k$, $2=2^1$

Using case#02

, $p > -1$

$\Theta(n^{\log_b a} \log^{p+1} n) = \Theta(n\log^2 n)$

Date MID-I (2018) Day

Q#01 Prove:

i) $n^2 + 4^5 = \Theta(n^2)$

$n^2+4^5 \leq c_1 \cdot n^2$

let $c_1 = 2$

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

$2n^2 - 4^5 \geq 0$

$n \geq 512$

$n^2+4^5 \geq c_2 \cdot n^2$

let $c_2 = 1/2$

$n^2+4^5 \geq \frac{1}{2}n^2$

$\frac{1}{2}n^2 + 4^5 \geq 0$

$n_0 \geq 0$

ii) $2^n + 2n = \Omega(n^2)$

$2^n + 2n \geq c_1 n^2$

let $c_1 = 1$

n	2^n+2n	n^2
0	1	0
1	4	1
2	8	4
3	14	9
4	42	25

at $n_0 \geq 0$

iii) $2n + 4^{\log_2 n} - 5 = \Theta(n^2)$

$2n + n^2 - 5 = \Theta(n^2)$

$2n+n^2-5 \leq c_1 n^2$

let $c_1 = 2$

$n^2 - 2n + 5 \geq 0$

n	n^2-2n+5
0	5
1	4
2	5
3	8

at $n \geq 1$

$2n+n^2-5 \geq c_2 n^2$

let $c_2 = 1/2$

$2n+n^2-5 \geq \frac{1}{2}n^2$

$\frac{1}{2}n^2 + 2n - 5 \geq 0$

$n_0 \geq 2$

Page No.

$a=4,\ b=2,\ k=3\quad p=0$

$a<b^k$

Date Day

$c = 1/2$

$4^{\log_2 n} + n = \Theta(2^n)$

$n^2 + n =$

n	2^n+n^2	$\frac{1}{2}2^n$
1	3	1
2	8	2
3	17	4
4	32	8

Q#5: Master Theorem

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

as $a=2$, $b=1$, $k=0$

as $a>1 \therefore \Theta(a^{n/b} \cdot n^k)$

$\Theta(2^n)$

hence proved

$n_0 \geq 1$

$T(n) = 32T(n/4) - n^2 \log n$

$T(n/4) = 32T(n/4^2) - \frac{n^2}{4^2}\log\frac{n}{4}$

$T(n) = 32^2 T(n/4^2) - n^2 \log n$

$T(n/4^2) = 32T(n/4^3) - \frac{n^2}{4^4}\log\frac{n}{4^2}$

$T(n) = 32^3 T(n/4^3) - \frac{n^2}{2^2}\log\frac{n}{4^2} - \frac{n^2}{2}\log\frac{n}{4}$

$= 32^k T(n/4^k) - \frac{n^2}{2^{k-1}}\log\frac{n}{4^{k-1}} - \frac{n^2}{2^{k-2}}\log\frac{n}{4^{k-2}} + \dots - \frac{n^2}{2^2}\log\frac{n}{4^2} - \frac{n^2}{2}\log\frac{n}{4}$

$= (2^{5k})$

$(2^k)^5 \quad (\sqrt{n})^5$

$T(1) = 1,\quad \frac{n}{4^k} = 1,\quad k = \log_4 n$

$n = 4^k \Rightarrow n = (2^k)^2 \quad 2^k = \sqrt{n} \quad 2^5 \quad 2^2$

Page No.

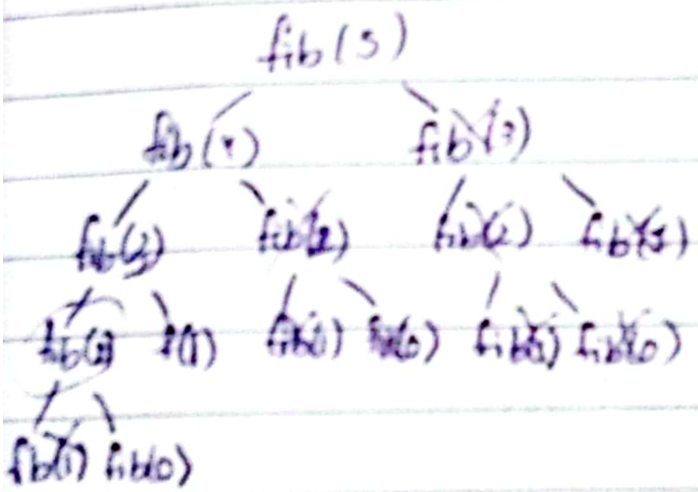
Memoization: Agar koi function call ki value pehle se calculated hai to use wapis calculate nhi karenge balki previous value hi use karenge.

Date ______ Day ______

"DYNAMIC PROGRAMMING"

Used to solve optimization problems.
↳ problems which req minimum or maximum results

In DP we find all possible sol and pick up the best one.



Same function calls are repetitive hence the time complexity is $O(2^n)$. We store the result of the call in order to use it in future.

are memoization

▶ This is top down approach
Starting for 5 going to 0.

```
int fib(int n){
    if (n<=1)
        return n;
    F[0]=0 ;F[1]=1;
    for(int i=2 ; i<=n ; i++)
        F[i] = F[i-2]+F[i-1];
    return F[n];
}
```

going from 2nd index to last index

This is are tabulation
bottom up approach.

```
↳ F(n){
    if(n≤1)
        ret n;
    if(A[n]!=-1)
        ret A[n];
    else
        ret A[n]=F(n-1)+F(n-2);
```

$P[1] = 1$, $P[stages] = n$;

for (i = 2 ; i < stages ; i++)
  $P[i] = d[P[i-1]]$;

" MATRIX CHAIN MULTIPLICATION "

- $A_{2\times5}$ $B_{5\times10}$ $C_{10\times3}$

  $2\times10\times3$

$A_{p\times q} \times B_{q\times r} = C_{p\times r}$
Total no. of multiplications will be $p\times q\times r$

Iske liye ya to (AB)C ya A(BC) dono se alag alag no. of ⊗ ayenge. To DP k zariye hum all possible solutions find out krenge or minimum dhoondenge.

$A_{2\times5}$ $B_{5\times10}$ $C_{10\times3}$
P_0 P_1 P_1 P_2 P_2 P_3

$(BC)^{2,3} = B_{5\times10}\ C_{10\times3} = 5\times10\times3 = 150$

$M[2,3] = \underset{0}{M[2,2]} + \underset{0}{M[3,3]} + 2\times5\times10 = 150$

M[2,2] → ka matlab no. of ⊗ req for multiplying B → there is only 1 so no. of ⊗'s 0

M[2,3] → " " " " " " " B & C

$A.(BC) = A_{2\times5} \times (BC)_{5\times3} = 2\times5\times3 = 30 + 150 = 180$ (BC ki value)

$M[1,3] = \underset{0}{M[1,1]} + \underset{150}{M[2,3]} + \underset{30}{P_0 P_1 P_3} = 2\times5\times3 = 180$

means no. of ⊗'s for multiplying A, B & C.

$(AB) = A_{2\times5}\ B_{5\times10} = 2\times5\times10 = 100$

$M[1,2] = M[1,1] + M[2,2] * 2\times5\times10 = 100$

$(AB)C = (AB)_{2\times10}\ C_{10\times3} = 2\times10\times3 = 60$

$M[1,3] = \underset{100}{M[1,2]} + \underset{0}{M[3,3]} * \underset{60}{P_0 P_2 P_3} = 160$

$$M[i,j] = \min \begin{cases} \{ M[i,k] + M[k+1,j] + P_{i-1} P_k P_j \} & i \le k < j \\ 0 & \text{if } i = j \end{cases}$$

Date

Q: A $_{10\times100}$ B $_{100\times20}$ C $_{20\times5}$ D $_{5\times80}$

$p_0\ p_1$ $p_1\ p_2$ $p_2\ p_3$ $p_3\ p_4$

initially empty

jis 'k' ~~ki value pak~~ min milega wo yaha likhenge

M	1	2	3	4
1	0	20K	15K	44K
2		0	10K	50K
3			0	8K
4				0

K	1	2	3	4
1	~~1~~	1	1	3
2		~~2~~	2	3
3			~~3~~	3
4				~~4~~

$m[1,1] = m[2,2] = m[3,3] = m[4,4] = 0$

$m[1,2] = \underset{0}{m[1,1]} + \underset{0}{m[2,2]} + 10\times100\times20 = 20000 = 20K$ (k=1)

diagonal wise calculation karenge take prev values use ho sakein

$m[2,3] = \underset{0}{m[2,2]} + \underset{0}{m[3,3]} + 100\times20\times5 = 10K$ (k=2)

$m[3,4] = \underset{0}{m[3,3]} + \underset{0}{m[4,4]} + 20\times5\times80 = 8K$ (k=3)

$m[1,3] = \underset{0}{m[1,1]} + \underset{10K}{m[2,3]} + 10\times100\times5 = 15K$ ✓ min (k=1)

$\underset{20K}{m[1,2]} + \underset{0}{m[3,3]} + 10\times20\times5 = 30K$ (k=2)

Day

Date

$m[2,4] = \underset{0}{m[2,2]} + \underset{8K}{m[3,4]} + 100\times20\times80 =$ ~~FG: 2..K~~ 168K (k=2)

$\underset{10K}{m[2,3]} + \underset{0}{m[4,4]} + 100\times5\times80 = 50K$ ✓ (k=3)

$m[1,4] = \underset{0}{m[1,1]} + \underset{50K}{m[2,4]} + 10\times100\times80 = 130K$ (k=1)

$\underset{20K}{m[1,2]} + \underset{8K}{m[3,4]} + 10\times20\times80 = 44K$ (k=2)

$\underset{15K}{m[1,3]} + \underset{0}{m[4,4]} + 10\times5\times80 = 19K$ ✓ (k=3)

$\boxed{((A)(B\ C))\ D}$ ⇒ A B C D → $m(1,4) = 3$ so

we split on 3rd pos → $\underbrace{(ABC)}_{m[1,3]}D$

→ $m(1,3) = 1$

we split on 1st $((A)(BC))D$ ✓

Date ______ Day ______

"0/1 Knapsnack"

i	1	2	3	4
P_i	10	40	30	50
w_i	5	4	6	3

max capacity

$W = 10$

jitni max capacity

total no. of objects

		W →	0	1	2	3	4	5	6	7	8	9	
		0	0	0	0	0	0	0	0	0	0	0	0
10	5	1	0	0	0	0	0	10	10	10	10	10	10
40	4	2	0	0	0	0	40	40	40	40	40	50	5
30	6	3	0	0	0	0	40	40	40	40	40	50	7
50	3	4	0	0	0	50	50	50	50	90	90	90	9

x_0	x_1	x_2	x_3
0	1	0	1

90 - 50

40 - 0

(1) Formula:

$$V[i,w] = \max\{V[i-1,w],\ V[i-1,\ w-w[i]] + P[i]\}$$

Page No. ______

Date ______ Day ______

(1) jab $i=0$ hai means we have no object so the whole row will have zero value.

(2) at $W=0$ means we have to put those profit by which we have max capacity $=0$, so we put 0 in 1st col.

(3) Now at $i=1$, $w=5$ & $P=10$, Now ~~we have to~~ the object having weight 5 can only come where max capacity is greater than or equal to 5 isiliye hume [1,5] par 10 profit rakhdiya. Or q k hamare paas currently ek hi object hai to aage bhi wohi likhdia from $(1,5) \to (1,10)$.

(4) Now we have two objects we will be considering both 1st & 2nd object. 2nd object ka weight 4 hai to usko ~~near~~ mein [2,4] par hi rakh sakta ho.

Page No. ______

```
int main(){

  int P[5] = {0, 1, 2, 5, 6}
  int wt[5] = {0, 2, 3, 4, 5}
  int m=8 , n= 4
  int K[5][9]

for(int i=0; i<=n ;i++){
    for(int w=0; w<=m ;w++){

        if(i==0 || w==0) //filling 1st row & col with 0's
            k[i][w]=0;

        else if( wt[i] <=w)

            k[i][w]= max(K[i-1][w], P[i]+ K[i-1, w-wt[i

        else
            k[i][w]= k[i-1][w]; //filling value of prev
    }
}
}
```

Page No.

```
// For checking which object to include.

i=n , j= m

while (i>0 && j>0) {

    if( K[i][j] == K[i-1][j] )

        cout<<i << " =0 "; //not included
        i--;

    }
    else
    {
        cout<<i << " =1 "; // included
        i--;
        j= j- wt[i];
    }

}
```

Page No.

Date

Day 4

" Longest Common Subsequence

	^	A	B	C	B	D	A	B
^	0	0	0	0	0	0	0	0
B	0	←0	↖1	←1	↖1	←1	←1	↖1
D	0	←0	1↑	←1	←1	↖2	←2	←2
C	0	←0	1↑	↖2	←2	←2	←2	←2
A	0	↖1	←1	2↑	←2	←2	↖3	←3
B	0	1↑	↖2	←2	↖3	←3	←3	↖4
A	0	↖1	2↑	←2	3↑	←3	↖4	←4

bcba

→ Agr dono letters same hai to hum us cell k upper diagonal ko choose krenge with a plus 1.

→ Agr dono letter different hai to current cell k upper wala ya left wala dono mein se jo max ho usko choose krenge or jo choose kraa hai uski taraf arrow.

→ When all cell filled we will then find the sequence letters.

→ Start from the bottom-right most 4 cuz it is the max no. Now follow the arrows. We include only those letters in which the arrow is diagonal. If arrow is left or upward we dont include those letters. We write letters right to left.

Page No.

Date

Day

if (A[i] = B[j])

LCS[i,j] = 1 + LCS[i-1, j-1];

↳ adding 1 to diagonal of current cell

else

LCS[i,j] = max (LCS[i-1,j], LCS[i,j-1]);

↳ maximum from upper or left cell.

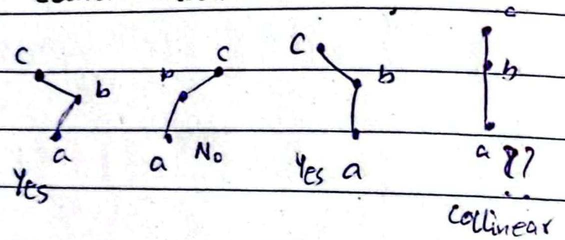
Page No.

Date Day

" Geometric Algorithms "

▷ Implementing CCW

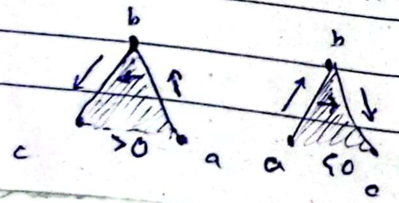
↳ Given three points a, b, c , is a-b-c a counterclockwise turn?



→ Determinant gives ~~2ic~~ twice area of Δ.

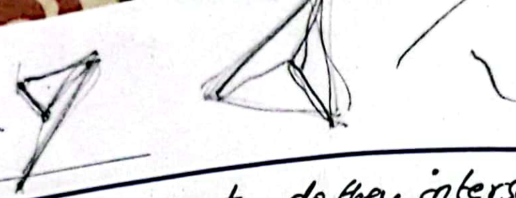
$$2 \times \text{Area}(a,b,c) = \begin{vmatrix} a_x & a_y & 1 \\ b_x & b_y & 1 \\ c_x & c_y & 1 \end{vmatrix} = (b_x - a_x)(c_y - a_y) - (b_y - a_y)(c_x - a_x)$$

if area > 0 then a-b-c counterclockwise
area < 0 then a-b-c clockwise
area = 0 then a-b-c collinear.



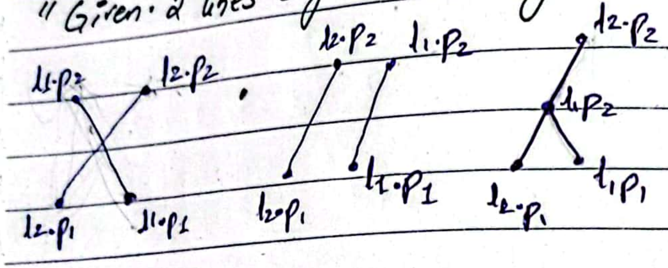
Date Day

```
public ~~final~~ class Point {

    int x;
    int y;

    Point (int x, int y){
        this.x = x ; this.y = y ;
    }

    int ccw ( Point a, Point b, Point c){

        double area = (b.x - a.x)*(c.y - a.y) -
                      (b.y - a.y)*(c.x - a.x);
        if ( area < 0 ) return -1;
        else if (area > 0) return +1;
        else    return 0;
    }
```



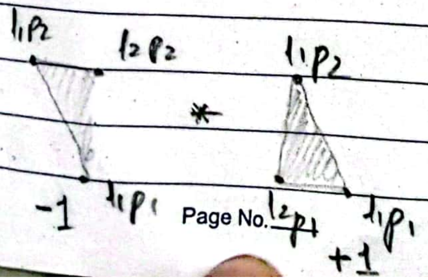
Date ____ Day ____

"Given 2 lines segment, do they intersect?"




```
boolean intersect (Line l1, line l2){

  int test1, test2;

  test1 = Point.ccw(l1.p1, l1.p2, l2.p1) *
          Point.ccw(l1.p1, l1.p2, l2.p2);

  test2 = Point.ccw(l2.p1, l2.p2, l1.p1) *
          Point.ccw(l2.p1, l2.p2, l1.p2)

  return (test1 <= 0) && (test2 <= 0);
}
```

test1 →



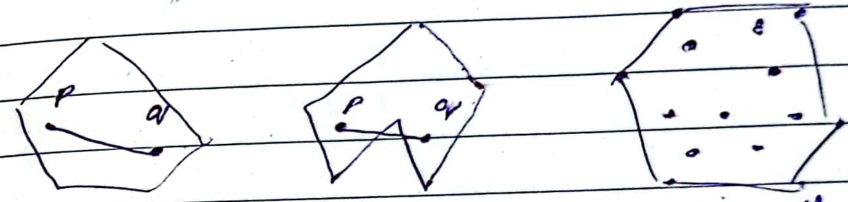
odd → within curve
even → outside curve

Date ____ Day ____

▷ Convex Hull:

A set of point is convex if for any two points p & q in the set, the line segment pq is completely in the set.

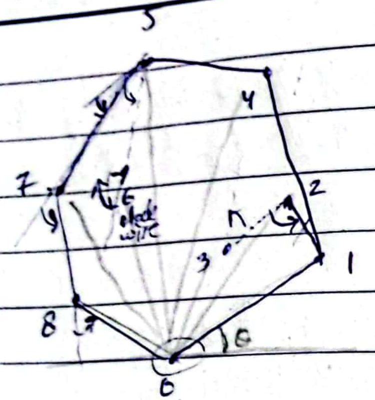
Small convex set containing all the points.



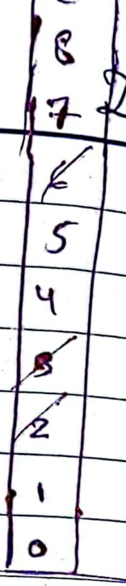
convex — not convex — convex hull

GRAM SCAN:

→ Select the point with the lowest Y coordinate.
→ Sort the points by the angle relative to the bottom most point
→ Iterate in sorted order placing each point on a stack, but only if it makes a counter clockwise turn relative to prev 2 points on stack
→ Pop prev point of the stack if making a clockwise turn.



bottom most



| 0 1 4 5 7 8 |

convex hull points.

```
for (int i = 2; i < points.size ; i++){
    Point next = points.get(i);
    Point p = stack.pop();
    while (stack.peek() != Null && ccw(stack.peek(), p,
                                        next ) <= 0 ) {
        p = stack.pop();
    }
    stack.push(p);
    stack.push(points.get(i));
}
```



```
Point p = stack.pop();
```

Date ______ Day

"STRING MATCHING"

0 1 2 3 4 5 6 7 8
a b a c d a b a d → haystack
first

a b a d → needle.
0 0 1 0
second

n = haystack.size(); m = needle.size();

```
for (i=0; i<=n-m; i++)
{
  first = i ; second = 0;
  while (second < m)
     if (haystack[first] != needle[second])
           break;
     else
           first++; second++;
  }
  if (second == m)
        return first-second; //returning index of 1st matched character
}
return -1;
```


Date ______ Day

▷ KMP Algorithm:

```
findLPS(int *lps, needle){

    int pre = 0 ; suf = 1;

    while (suf < needle.size()) {
       // match
       if (needle[pre] == needle[suf])
              lps[suf] = pre + 1;
              suf++ ; pre++;
       }
       //not matched
       else {
          if (pre == 0)
               lps[suf] = 0;
               suf++;
          else
               pre = lps[pre-1];
       }
    }
}  return lps;
```

Date   Day

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	B	C	A	B	D	A	B	C	A	B	C	A	B	D
0	0	0	1	2	0	1	2	3	4	5	3	4	5	6

(1) Prefix = 0 & suffix = 1

(2) Agr prefix suffix match krenge to lps k andar hum prefix +1.

Jese jab prefix = 0 & suffix = 3 hoga to hum lps [sufix] i.e lps[3] mein prefix +1 kdenge or suffix & prefix dono ++.

Agr match nhi krte to

hum phele to prefix ki value check krenge agr wo 0 hai.

Jese when prefix = 0 & suffix = 3 to needle [pre] != needle [suffix] is case mein sirf suffix age move hojayega. or lps [suf] = 0 because of unmatched.

Agr prefix 0 nhi to hum prefix mein lps[prefix-1] kdenge.

Page No.

Date   Day

```
        hystack,
KMP ( needle ) {

    int lps[needle.size()];
    findLPS(&lps, needle);
    int first = 0; second = 0;

    while(second < needle.size() && first < hyst.size()){
        //match
        if (needle[second] == hystack[first])
            second++; first++;
        else{ //not match
            if (second == 0)
                first++;
            else
                second = lps[second-1];
        }
    }
                    → mtlb poora loop chal gaya hoga.
    if (second == needle.size())  //match funded.
        return first - second;

    return -1;
}
```

Page No.

Date ______ Day ______

"BUCKET SORT"

→ Works on floating point numbers between range 0.0 to 1.
→ Input should be uniformly & independently distributed with [0, 1] to get running time O(n)

A ⇒
0.79, 0.13, 0.64, 0.39, 0.20, 0.89, 0.53, 0.42, 0.06, 0.76

$0.79 \times 10 = 7.9 \quad \lfloor 7.9 \rfloor = 7$

$B[7] = 0.79$

Buckets

0	0.06
1	0.13
2	0.20
3	0.39
4	0.42
5	0.53
6	0.64
7	0.79 – 0.76
8	0.89.
9	

```
Bucket Sort (A) {
    let B[N-1];
    n = A.length
    for i=0; i<n; i++
        B[i] = 0;
    for i = 1 → n
        B[⌊n×A[i]⌋] = A[i];

    for i=0 → n-1
        sort each list B[i] with
        insertion
```

agr ek hi index par zyada values hongi to har list ko individually sort krna hoga (n^2)

Date ______ Day ______

"CLASSES OF COMPLEXITIES"

→ **P- Problems:** (deterministic algo)

Problems that are solvable in polynomial time.

Example:
- Sorting algo (Bubble, Merge, Quick)
- Linear Search, Binary Search

→ **NP - Problems:** (non-determ Polynomial)

Problems that are verifiable in polynomial time.
NP algo mein bhi aksar statements deterministic hoti hai bs kuch statements ~~esi~~ non-deterministic hoti hai means k humein nhi ptaa hota k unka solution kese niklega.

```
Algorithm NSearch (A, n, key) {
    j = choice();  →O(1)  → non-determ  (as the function directly gives index value)
    if (key == A[j])
        print("Found");
        Sucess ();  → non-d
    }
    print("Not Found");
    Failure();  → non-d
}
```

ye choice(); function ek tareeke ki blank statement hai jab humein esa koi function maloom hojayega jo O(1) mein search krle we will put it here. But for now it is non-deterministic algorithm.

Date ______ Day ______

→ $P \subseteq NP$ (A P problem is also an NP-Problem)

→ Inorder to prove that $P \neq NP$ we would need to prove that there exists a set of problem X such that:

- X falls in NP. There exists an algo with which a non-deterministic Turing Machine could solve problems in X in poly-time.
- X doesnot fall in P. There exist no algo whatsoever with which a deterministic TM could solve X in poly-time.

- Exponential Time (2^n)

0/1 Knapsack
Travelling SP
Sum of subsets
Graph Coloring
Hamiltonian Cycle.

- If we are unable to get a polynomial time solution for these than at least show similarities b/w them so that one problem is solved then all other will be solved.
  We will not try to solve individually
  For this we need to show some relation b/w them.
- If not able to write a poly-time deterministic algo then why don't write a non-deterministic one?

A problem that require Yes/No answer is called Decision problem

Date ______ Day ______

→ We will take a base problem i.e CNF-satisfiability (2^n) problem and try to relate the other with this problem.

CNF-Satisfiablity:

$x_i = \{x_1, x_2, x_3\}$

$CNF = (x_1 \lor \bar{x}_2 \lor x_3) \land (\bar{x}_1 \lor x_2 \lor \bar{x}_3)$

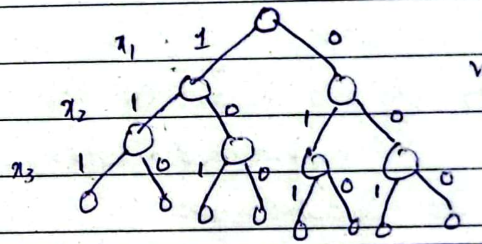
Problem: Find possible values of x_i for which CNF is true.

The possible values can be

x_1	x_2	x_3
0	0	0
0	1	0
0	0	1
⋮	⋮	⋮
1	1	1

→ (2^n)

So for n variables it will take 2^n.



showing all possible value in a state space tree.

0/1 Knapsack: $P = \{10, 8, 12\}$
$W = \{5, 9, 3\}$
$x_i = \{0/1, 0/1, 0/1\}$

0/1 Knapsack mein bhi all possible value to maximize weight hi dhoondni hai to iska bhi space tree CNF-ki tarha hi hoga.

$\boxed{\text{So if we find the solution of this state-space tree in P time then we can get sol of both problems.}}$

→ All the exponential algo are hard algorithms.

→ And the CNF-satisfiability is a NP-Hard

NP hard problems are atleast as difficult as NP.

→ Relating one algo to another is called **Reduction**

Satisfiablity ∝ 0/1 knapsack

means satis- reduces to 0/1 knapsack

→ we develop a polynom. time algo for conversion of satisfiablity → 0/1 knapsack

→ If an algo can solve 0/1 then it can also solve satisfiablity

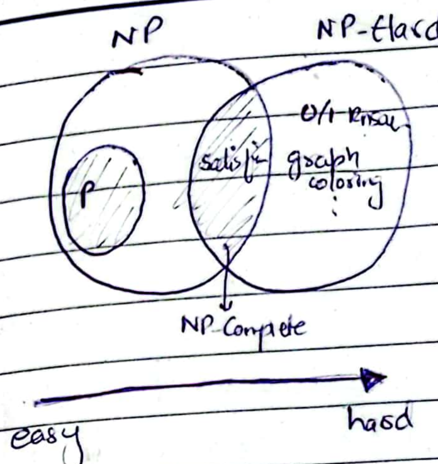
→ if satisfiablity is reducing to 0/1 knapsack problem then knapsack also bcms NP-Hard

→ And if we reduce knapsack into any other problem let say to Graph Coloring,

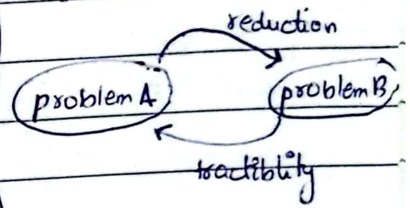
↳ then graph coloring also bcms NP-Hard

→ If an NP-Hard problem have a non-deterministic polynomial time algo then it is called NP-Complete.

↳ Satisfiablity is $\boxed{\text{NP-Complete}}$



If A reduces to B and B is poly-time solvable, then A is also poly-time solvable.



→ **Method for Proving an Algo as NP-Hard:**

If A reduces to B and A is NP-hard, then B is also NP-hard.

① Choose an NP-Hard problem A.

② Prove that A reduces to B.

→ **Reducing 3-SAT To Independent Set Problem:**

3-Sat Problem: (all clauses must have exactly 3 literals)

A boolean formula with clauses, where each clause contains exactly 3 literals. like $(\bar{x}_1 \vee x_2 \vee \bar{x}_3) \wedge (x_1 \vee x_2 \vee \bar{x}_3)$

clause 1 clause 2 literal

We need to find possible values of x_1, x_2 & x_3 to make this true.

Date ________ Day ____

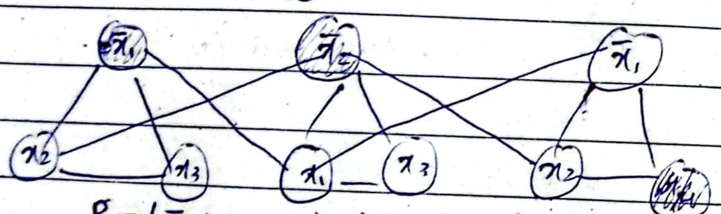
Independent Set Problems:

↳ A graph $G = (V, E)$ and an integer K.

↳ Find an independent set of size K (a set of vertices in which no two vertices are connected by an edge)

Reduction:

1. G will have one vertex for each literal in a clause
2. Connect the 3 literals in a clause to form a triangle; the indep set will pick at most one vertex from each clause, which will correspond to the literal to be set to true.

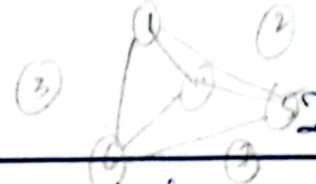


$$P = (\bar{x}_1 \vee x_2 \vee x_3) \wedge (\bar{x}_2 \vee \bar{x}_1 \vee x_3) \wedge (x_2 \vee \bar{x}_1 \vee x_4)$$

3. Connect 2 vertices if they label complementary literals

(Complimentary ko isiliye connect kara q k agr x_1 is ~~selected~~ included as independent ~~vertice~~ set it implies that x is assigned true in 3-SAT set. And if $\bar{x}$ is also included this will imply x is false, which creates contradiction)

By connecting x & $\bar{x}$ the independent set can only select any one of them.



Date ________ Day ____

4. Take 'k' to be the no. of clause.

(q k has clause (means har △ se ek hi vertex select hoga) to jitne clauses utna size of set)

"Now P' is only satisfiable iff G has an independent set of size k."

solution of 3-SAT ($\bar{x}_1 = 0$, $x_2 = 0$, $x_4 = 1$, $x_3 = 0/1$)

→ Reducing Independent Set Problem to Vertex Cover Problem:

Vertex cover set is complement of independent set.

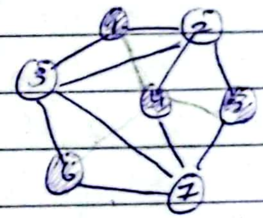
In a graph $G = (V, E)$

"S is an Independent Set if and only if $(V - S)$ is a vertex cover."

Consider an arbitrary edge $(U, V) \in E$

1. If S is an indep set, there is no edge $e = (U, V)$ in G such that both $U, V \in S$.

Hence for any edge $e = (U, V)$, at least one of U, V must lie in $(V - S)$



$S = \{1, 4, 5, 6\}$ indep set

$(V - S) = \{3, 2, 7\}$ vertex cover

Date_______ Day_______

2. If $(V-S)$ is a vertex cover, then b/w any pair of vertices $(U,V) \in S$ if there exist an edge e, none of endpoints of e would exist in $(V-S)$ voilating the def of vertex cover.

Hence no pair of vertices in S can be connected

S - independent $\Rightarrow$ either $U \notin S$, or $V \notin S$ or both $\notin S$

$(V-S)$ - Vertex Cover $\Rightarrow$ either $U \in (V-S)$ or $V \in (V-S)$ or both $\in (V-S)$

⇒ Reduce ~~Clique~~ ISP to Clique:

S is independent in G iff S is a clique in the Complement of G.

To reduce IS to clique

↳ we compute the complement of G

↳ The G' has a clique of size K iff the G has indep set of size K.

→ The construction of G' can be done in polynomial time.

Date_______ Day_______

▷ APPROXIMATION ALGORITHM:

An approx. algo is way of dealing with NP- Completeness for optimization problem. The goal of approximation Algo is come as close as possible to optimal solution in polynomial time.

$C \rightarrow$ cost of original sol

$C^* \rightarrow$ cost of optimal sol

$p(n) \rightarrow$ approximation ratio, $n \rightarrow$ input size

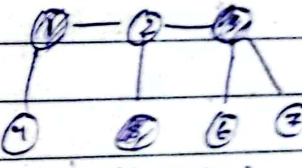
$$p(n) \geq \max\left(\frac{C}{C^*}, \frac{C^*}{C}\right)$$

The approx algo is then called a $P(n)$- approx algo

• VERTEX COVER APPROX. ALGO.:

$C \leftarrow \emptyset$

$E' \leftarrow E[G]$



min vertex cover = 3

while $(E' \neq \emptyset)$

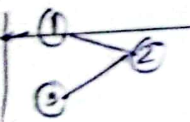
do let (U,V) an arbitrary edge in E'

Add that ~~edge~~ vertices in C

Remove all other edges associate with either U & V

Return C;

ishka min = 1 hai lekin approximation se hamesha 2 hi ayega



Date ______ Day ______

$\rho(n) = \max\left(\frac{1}{2}, \frac{2}{1}\right) = 2$

So ~~(struck word)~~ it is 2- approx algo.

The vertex cover produced by approx is at most 2S.

S: size of (minimum) vertex cover

o Approximation For TSP:

① First check the triangle inequality.

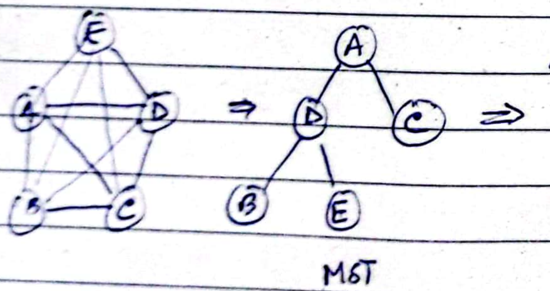
weight(a,c) <= weight(a,b) + weight(b,c)

② If satisfy triangle inequality then,

(i) Find MST 'M' of G.

(ii) Find preorder of 'M'

(iii) Remove duplicate vertices in Preorder and that will become hamiltonian path.



A, D, B, E, E, D, A, C, A

preorder walk.

⇓

A, D, B, E, C, A

hamiltonian path

The total weight will be twice as much

Date ______ Day ______

→ A 3/2 Approx Algo.

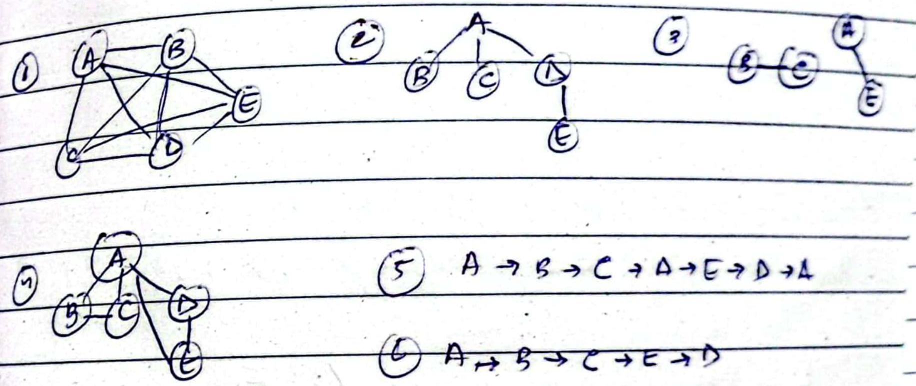
① Find a minimum Sp- Tree T

② Let O be the set of vertices with odd degree in T. Find a minimum-cost perfect matching M.

③ Add the set of edges of M to T.

④ Find an Eulerian Tour (visit every edge exactly once)

⑤ Shortcut the eulerian tour to make Hamiltonian cycle.



⑤ A → B → C → A → E → D → A

⑥ A → B → C → E → D

Date ____________ Day ____________

Set Cover:

Given set U, a collection of $S_1, S_2, S_3 \dots S_n$ of subsets of U, and an integer K, does there exist K or fewer subsets such that their union equal to U?

$U = \{1, 2, 3, 4, 5, 6, 7\}$, $K = 2$

$S_1 = \{3, 7\}$

$S_2 = \{3, 4, 5, 6\}$

$S_3 = \{1\}$

$S_4 = \{2, 4\}$

$S_5 = \{5\}$

$S_6 = \{1, 2, 6, 7\}$

If we union S_2 & S_6 we get U

$S_2 \cup S_6 = U$

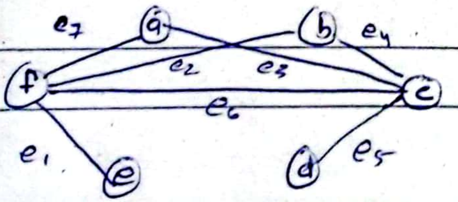
Reducing Vertex Cover to Set Cover:

Given $G = (V, E)$ and K as input for vertex cover, for set cover input use;

$U = \{$ all edges of $G \}$ i.e $U = E$

$S_i = \{ e \in E \mid e$ incident to $v_i \}$ for $\forall\, v \in V$

" There is a set cover of size at most K iff There is a vertex cover in G of size at most K."



$U = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$

$K = 2$

$S_a = \{e_3, e_7\}$, $S_b = \{e_2, e_4\}$

$S_c = \{e_3, e_4, e_5, e_6\}$

$S_d = \{e_5\}$, $S_e = \{e_4\}$

$S_f =$ [illegible] $e_7\}$

Date ____________ Day ____________

→ We need those sets of S that covers all edges.

$\boxed{S_c \cup S_f = U.}$ set cover of size 2.

Approximation algo:

Greedy Set Cover (X , S_n)

union of all subsets

family of subsets

$U = X$;

$C = \phi$;

while ($U \neq \phi$) { // wo S_i ka set choose karna hai jisme sabse zyada elements ho U k.

select S_i with max $|S_i \cap U|$

$U = U - S_i$ // phir select wale S_i k elements ko minus karde

$C = C \cup \{S_i\}$ // add that set in C

Return C;

Date ________ Day ________

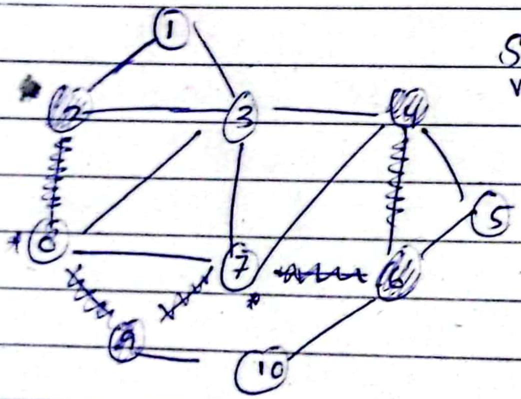
Steiner Tree Problem:

Given a Graph $G = (V, E)$ and ~~som~~ also a set of vertices $T \subseteq V$ which are called termin[als]. We need to find the minimum cost of joining the vertice in T.

Jab T vertices ko join karenge to kuch esi vertex bhi include karni parengi jo T mein initially include na ho. Those called Steiner vertices.

$T = \{2, 4, 9, 6\}$

Steiner vertices $= \{8, 7\}$



If $T = V$ then it becomes an MST Problem.

Date ________ Day ________

- Approximation Algo: $(2 - 2/n)$ - Approximate Algo.

① ~~Start~~ Start with a subtree T containing a single terminal vertex (any terminal vertex)

② While T doesnot include all terminals:
~~a. select a terminal vertex x not yet in T~~
   i- Find X that is closest to any vertex already in T
   ii- Use the shortest path algo to determine the "closest" vertex.
   b- Add the shortest path b/w X & T to the subtree T

~~③ Repeat until comp~~

③ Return T;

Date 13-Dec-2024 Day Friday

GRAPH ANALYSIS

- Breadth First Search : O(V+E)

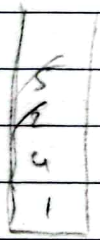
queue

1	2	4	5	

visited

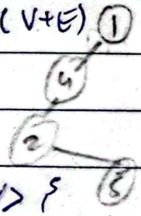
T	T		T	T

```
BFS ( s ):
    visited(s);
    queue.insert(s);
    while (!queue.empty()){          1245
        u = queue.extractHead();
        for each edge <u,d>{
            if(!visited(d)){
                visited(d);
                queue.insert(d);
            }
        }
    }
```



- DFS(curr, visited): O(V+E)

```
DFS(curr, visited):
    if(!visited[curr])
        visited[curr] = true;
        for each edge <curr,d>{
            DFS(d, visited)
        }
    }
}
```



Date Day

- Topological Sort: O(V+E)

Toposort mein poora DFS ka code hai bs DFS(curr, visited) k neeche sortOrder.push(curr) ye krdenge.

- DIJKSTRA'S ALGORITHM : O((V+E) log V)

```
for (i=1 to n) do
    d[i] = ∞
end for
d[s]=0;
PQ(n);
PQ.insert(s,0);
S ← Φ
while (PQ ≠ Φ) do
    u ← Extract-Min(PQ)
    for (each edge of the form e=(u,v)) do
        if (d[v] > d[u] + w(u,v)) then
            d[v] = d[u] + w(u,v)
            PQ.insert(v, d[v]);
        end if
    end for
    S ← S ∪ {u}     // Storing all visited vertices
end while
```

Date ________ Day ________

Proof Of Correctness:

Let $\delta(v)$ denote true shortest path distance of vertex v from the src s.

Lemma: "if $d[V] = \delta(V)$ for any vertex v, at any stage of the Algo, then $d[V] = \delta(V)$ for the rest of the algo"

Proof: Clearly, d[V] cannot become smaller than $\delta(v)$; likewise, the test condition in RELAX() procedure will always fail.

$O(E \log E)$ or $O(E \log V)$

Kuruskal's Algorithm: ~~E O log~~

Set each vertex in V as a vertex set

Set A as an empty set

while (there are more than 1 vertex set) {

- Add a smallest edge (U,V) to A where U,V are not in same set (i.e avoid cycle)
- Merge the set contains U with set contains V.

}

Date ________ Day ________

Prims Algorithm: $O((V+E)\log V)$

Shortest Path Problems:

- Single source: Find a shortest path from a given src to each of the vertices
  - ↳ Dijkstra's Algo → greedy algo; Faster than Bellman-Ford; works on non-neg weights
  - ↳ Bellman-Ford Algo → DP algo; Also works on neg weight

Bellman Ford Algorithm: $O(n^3)$

↳ Is mein sabse phele saare edges list kar lenge

↳ Phir har edge par relaxation condition apply karenge. For $e(U,V)$

$$\text{if}(d[U] + w(U,V) < d[V])$$
$$d[V] = d[U] + w[U,V]$$

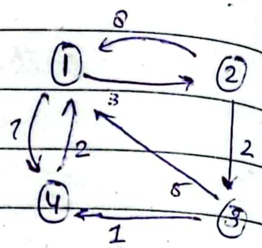
↳ Or ye cheez humein V-1 times apply karni hai.

Date ______ Day ______

- **All-pair shortest path :** Find shortest path for every pair of vertices.

Floyd Warshall Algo : $O(V^3)$

$$A^0 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & \infty & 7 \\ 2 & 8 & 0 & 2 & \infty \\ 3 & 5 & \infty & 0 & 1 \\ 4 & 2 & \infty & \infty & 0 \end{array}$$



→ Total 4 vertices hai to, total 4 matrices banenge.

→ Hamein har vertex ka doosre vertex se shortest path dekhna hai.

→ Jab A^1 matrix banayenge to iska matlab U ka V tak shortest path through vertex 1.

→ Jab A^2 matrix banayenge to iska matlab $d(U, V)$ through vertex 2.

→ Jab A^k hoga to 'k' will be the middle vertex.

$$A^k[i,j] = \min\left(A^{k-1}[i,j],\ A^{k-1}[i,k] + A^{k-1}[k,j]\right)$$

diagonal hamesha 0
8 agr A^1 nikalenge to 1st row 1st col poori matrix ke hi ayenge.

Date ______ Day ______

$$A^1 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & \infty & 7 \\ 2 & 8 & 0 & 2 & 15 \\ 3 & 5 & 8 & 0 & 1 \\ 4 & 2 & 5 & \infty & 0 \end{array}$$

$$A^2 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & 5 & 7 \\ 2 & 8 & 0 & 2 & 15 \\ 3 & 5 & 8 & 0 & 1 \\ 4 & 2 & 5 & 7 & 0 \end{array}$$

$$A^3 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & 5 & 6 \\ 2 & 7 & 0 & 2 & 3 \\ 3 & 5 & 8 & 0 & 1 \\ 4 & 2 & 5 & 7 & 0 \end{array}$$

$$A^4 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & 5 & 6 \\ 2 & 5 & 0 & 2 & 3 \\ 3 & 3 & 6 & 0 & 1 \\ 4 & 2 & 5 & 7 & 0 \end{array}$$

```
for(k=1 ; k<=n ; k++){
    for(i=1 ; i<=n ; i++){
        for(j=1 ; j<=n ; j++){
            A[i,j] = min(A[i,j] , A[i,k]+A[k,j])
```