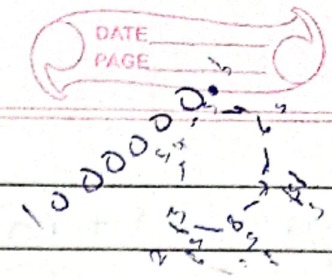
ALGOS



Priority analysis →

→ order of freq of execution of each statement
→ add the freq of each statement to get the total no of prog steps in the algorithm.

* Declarative steps don't contribute to programming steps but if you do assignment then it is counted as step. Eg int i=1;

* Simple assignment statements → 1 step
x = x + a[i]/4;

* Conditional statements →

if (condition) { statements1 }
else (condition) { statements2 }

1 program step for if

True
no. of prog steps req to execute statements 1
∴ 1 + count(stm1)

False
of statements 2
∴ 1 + count(stm2)

DATE
PAGE

* **Loops** 1 + no. of time i incremented (for loop)

for (int i=1 ; i<=n 1st step
then i=i+1, i<=n 2nd step

While loop →
i=1; 1 step
while (i<=n) { i++; }
1 step 1 step

Function call → 1 + no. of steps required to execute add (n)

return (1) – 1 prog step

if return has function call
return (1 + f(n)) → 1 + No. of steps req for that fun call

Count Method →

Global Variable count = 0;
When you encounter a prog step, increment count.
Value of count at the end of algo = Total prog steps

```
int suma (int a[], int n){
{ int sum = 0; count++ // Initialising
  for (int i = 1 ; i<=n ; i++){ sum+ = a[i] ; }
  return sum;
}
```

$ct_1 = 1$

$ct_2 = 2n$

$ct_3 = 2$

$\therefore ct = 2n+3$

```
count++    // Control part of for loop
count++    // Assignment statement
count++    // Last time of control part where i = n+1
count++    // for return
```

// Recursive →

```
Rsum (int a[], int n){
    count++;    // for if condition
    if (n<=0){
        count++;    //return
        return 0;
    }
    else {
        Count++    // return statement
        return Rsum(a, n-1);
```

$2n+2$

S/E Method

```
Algo Rsum (int a[], int n) {
1       if (n <= 0)
2           return 0;
3       else
4           return (Rsum (a, n-1) + a[n]);
}
```

S/e		freq of execut^n		Total no. of prog. steps contri by each step	
n<=0	n>0	n<=0	n>0	n<=0	n>0
1) 1	1	1	1	1	1
2) 1		1		1	
3)	0		1		0
4)	1+T(n-1)		1		1+T(n-1)
				2	2+T(n-1)

Asymptotic Notation
↳ (n → ∞)

Big O notation → $f(n) = O(g(n))$

or

$f(n) \in O(g(n))$

$O(g(n))$ $f(n) <= c \times g(n)$ $\forall$ $n >= n_0$

$O(g(n)) = \{ f(n) \mid f(n) <= c \cdot g(n) \ \forall \ n >= n_0 \ \& \ c > 0 \}$

A function $f(n)$ is said to be in $O(g(n))$ denoted as $f(n) \in O(g(n))$ if $f(n)$ is bounded above by some constant multiple of $g(n)$ $\forall$ large n i.e if $\exists$ some +ve const c & some non -ve integer n_0 s.t. $f(n) <= c \cdot g(n) \ \forall \ n >= n_0$

Eg → $f(n) = 100n + 5$

$f(n) = O(n)$

$f(n) <= 105 \cdot n \ \forall \ n >= 1$

$c = 105 \quad n_0 = 1$

Omega notation (Ω): $f(n) \in \Omega(g(n))$ iff there exist a +ve const c & a non negative integer n_0 s.t $f(n) >= c \cdot g(n) \ \forall \ n >= n_0$

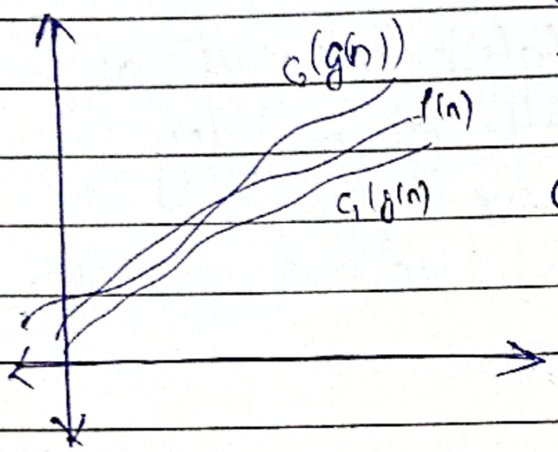
$\Omega(g(n))$ = Set of all functions $f(n)$ that have a higher order of growth than $g(n)$

$T(n) = \Omega(n) \qquad f(n) >= 10n \ \forall \ n >= 1$

$n = 1 \quad f(n) = 100 * 1 + 5 > 10 * 1$

$100 * 2 + 5 > 10 * 2$

Theta notation (Θ) → $f(x) \in \Theta(g(n))$ iff $\exists$ a +ve constt c_1, a non negative integer n_0 such that



$c_1 g(n) <= f(n) <= c_2 g(n) \quad \forall \; n >= n_0$

→ Eg: $f(n) = \frac{1}{2} n(n-1) \quad f(n) \in \Theta(n^2)$

$\therefore \; c_1 = \frac{1}{4} \quad c_2 = 2 \quad g(n) = n^2 \quad n_0 = 1$

$\frac{1}{4}(n^2) <= f(n) \leq 2 \cdot n^2$

→ $f(n) = a_r n^r + a_{r-1} n^{r-1} + \cdots\cdots + a_1 n + a_0$

$f(n) = \Theta(n^r)$

Small oh-o → Describe an upper bound on function $f(n)$ that is not asymptotically tight

$f(n) = 100n + 5 = O(n)$

$= o(n^2)$

$f(n) \notin o(n)$ but $f(n) \in o(n^2)$

$f(n) \in o(g(n))$ - iff $\exists$ +ve constt c & non negative integer n_0 such that $f(n) < c \cdot g(n)$

Complexity classes →

$$O(1) < O(\log n) < O(n) < O(n\log n) < O(n^2) < O(n^3) < O(2^n) < O(n!)$$

Using limits to compare & rank time complexities →

$f(n)$, $g(n)$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \begin{cases} 0 & f(n) \text{ has lower order of growth as compared to } g(n) \\ c & \text{---}''\text{---} \text{ same } \text{---}''\text{---} \\ \infty & \text{---}''\text{---} \text{ higher } \text{---}''\text{---} \end{cases}$$

L'Hôpital rule →

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f'(n)}{g''(n)}$$

Stirling → $n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$

Eg → $f(n) = \frac{1}{2}n(n-1)$ $g(n) = n^2$ $\therefore f_2 = \Theta(g(n))$

$$\lim_{n\to\infty}\left(\frac{f(n)}{g(n)}\right) = \frac{1}{2}\frac{(n^2-n)}{n^2} \overset{H}{=} \frac{1}{2}\left(1-\frac{1}{n}\right)$$

$$= \boxed{\frac{1}{2}}$$

(i) $f(n) = \log_2 n$ $g(n) = \sqrt{n}$

$$\lim_{n\to\infty}\left(\frac{\log_2 n}{\sqrt{n}}\right) = \frac{1}{n} \times \frac{\log_e 2}{\frac{1}{2}n^{-1/2}} = \frac{2\log_e 2}{n^{1/2}} = 0$$

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

Eg → $f(n) = 2^n$ $g(n) = n!$ $f(n) = \Omega(g(n))$

$$\lim_{n\to\infty} \frac{n!}{2^n} = \frac{\sqrt{2\pi n}\left(\frac{n}{e}\right)^n}{2^n} = \sqrt{2\pi n}\left(\frac{n}{2e}\right)^n = \infty$$

Q) $(n-2)!$, $5\log(n+100)^{10}$, 2^{2^n}, $0.001n^4 + 3n^3$,
$n^2\ln n$, $\sqrt[3]{n}$, 3^n

Sol) $O(n!)$, $O(\log n)$, $O(4^n)$, $O(n^4)$, $O(\ln^2 n)$, $O(n^{1/3})$.

* $\frac{\log(n)}{n^{1/3}} = \frac{1}{n} \times n^{2/3} = \frac{1}{n^{1/3}} = \Omega$

$\frac{1}{3}\frac{n^{2/3}}{n^{2/3}}$

* $2\log n \cdot \log n \times \frac{1}{3} n^{2/3} = O$

$$O(\log n) < O(n^{1/3}) < O(\log^2 n) < O(n^4) < O(3^n) < O(4^n)$$

Series

$f(n) = 1+2+3+\ldots\ldots+n-1+n \leq n+n+\ldots\ldots+n$

$\leq n^2$

② $1+2+3+\ldots\ldots\frac{n}{2}+\ldots\ldots n \geq \frac{n}{2}+\ldots\ldots n$

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

$\geq \Omega(n^2)$

Ans. Heap sort is actually $\log(n!)$ & not $n\log n$. It is actually approximated.

③ $f(n) = \log(n!)$

$= \log n + \log n-1 + \log n-2 + \cdots - \log 2 + \log 1$

$\leq \log n + \log n + \log n \cdots$ ——————

$\leq n\log n$

$\therefore f(n) \leq O(n\log n)$ — ①

Similarly, $f(n) = \log(n!)$

$\geq \log n + \log(n-1) + \cdots + \log \frac{n}{2}$

$\geq \frac{n}{2} \log \frac{n}{2}$

$= \Omega\left(\frac{n}{2}\log\frac{n}{2}\right)$ — ②

From ①, ②, $f(n) = \Theta\left(\frac{n}{2}\log\frac{n}{2}\right)$

Formulas: i) $\sum_{i=1}^{n} \frac{1}{i} = \Theta(\log n)$

ii) $\frac{1}{n} < \log(n+1) - \log(n) < \frac{2}{n}$

iii) GP: $\frac{a(r^n-1)}{r-1}$ if $r > 1$

Sol) $\sum_{i=1}^{n} = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \cdots\cdots \frac{1}{n}$

$\leq \sum_{i=1}^{n} \log(i+1) - \log(i)$

$\leq \log(n+1) - \log(n) + \log(n) - \log(n-1) + \cdots\cdots \log(1)$

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

$\leq \log(n+1)$

$O(\log n)$

Q) $f(i=1 \to n)$

$f(j=1 \to i)$

$f(k=1 \to j)$

$x += 1$

$i=1$ $j \to$ i times $k \to 1+2+\cdots\cdots i$ times

→ $i=1$ $j=1$ $k=1$ $\Rightarrow \Sigma j$

→ $i=2$ $j=1,2$ $k=1+2$ $\Rightarrow \Sigma j$

→ $i=3$ $j=1,2,3$ $k=1+2+3$ $\Rightarrow \Sigma j$

$i=n$ $j=1,2,3,\ldots n$ $k=1+2+3+\cdots\cdots+n$ $\Rightarrow \Sigma j$

$f(n) = \sum_{i=1}^{n} \sum_{j=1}^{i} (j)$

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

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

$$\sum_{i=1}^{n} \frac{i(i+1)}{2} \longleftarrow$$

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

$$= \frac{1}{2}\left(\frac{n(n+1)(2n+1)}{6} + \frac{n(n+1)}{2}\right)$$

$$= O(n^3)$$

Q) for(i=n to 1){
    j=i
    while(j>1) { x+=i; j-- }
    i=i/2;
}

Sol) Let $n = 2^k$

Sid

So, first iterat, $i = 2^{k-1}$, $j = 2^{k-1}$

$i = 2^{k-2}$, $j = 2^{k-2}$

$\vdots$

$i = 2^{k-k}$, $j = 0$

$\therefore \sum j = 1 + 2 + 2^2 + 2^3 + 2 \cdots 2^{k-1}$

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

$= 2^k - 1$

$= n$

Q) Prev ques

```
i=i/3;
```

$n = 3^k$

$\log_3 n = k$

$f(n) = n + n/3 + \frac{n}{9} + \dots\dots 1$

$= 3^k + 3^{k-1} + 3^{k-2} + \dots\dots 1$

$= 1 + 3 + 3^2 + \dots\dots 3^k$

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

$= O(3^k)$

$= \boxed{O(n)}$

Q) P.T $\sum_{i=0}^{k} \log\left(\frac{n}{2^i}\right) = \Theta(\log^2 n)$ assume $n = 2^k \Rightarrow k = \log_2 n$

Sol) $f(n) = \log\left(\frac{n}{2^0}\right) + \log\left(\frac{n}{2}\right) + \log\left(\frac{n}{4}\right) + \dots\dots + \log\left(\frac{n}{2^k}\right)$

$= \log n + \log n + \dots\dots k \text{ times} - (1\log 2 + 2\log 2 + 3\log 2 + \dots\dots k\log 2)$

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

$= k\left(k\log 2 - \frac{k\log 2}{2} - \frac{\log 2}{2}\right) = \frac{k}{2}(k\log 2 - \log 2)$

$= \log_2 (k)(k-1) = \Theta(k^2)$

$\log \frac{2^k}{2^0} + \log \frac{2^k}{2} + \log \frac{2^k}{2^2} + \cdots \log(1)$

$\log_2 [2^k * 2^{k-1} * 2^{k-2} * \cdots 2^0]$

$\log_2 2^{(k + (k-1) + (k-2) + \cdots 0)}$

$\frac{k(k+1)}{2}$ $\log_2 2$ $O(k^2)$

$= \Theta(\log_2 n)^2$

Q) Insert n elements in a heap.

$\log(1) + \log(2) + \log(3) + \cdots \log(n)$

$= \log(1 * 2 * 3 * 4 * \cdots n)$

$= \log(n!)$

$= O(n \log n)$

Same for deletion

Recursive Relations

$T(n) = a\,T(n/b) + f(n)$ → time to divide + time to conqu

a: no. of subproblem; n/b: size of subproblem

① Iterative substitution method →

Eg → $T(n) = T(n-1) + 2$

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

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

$= \vdots$

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

② Substitution method →

↳ Guess a solution for the rec. relat^n
↳ Prove the guess using Mathematical Induction

Eg: $T(n) = 2T(n/2) + O(n)$

Guess: $T(n) = O(n \log n)$ i.e $T(n) \leq c n \log n$

Let us assume that it holds true for $n/2$

$$T(n/2) \leq c\frac{n}{2}\log\left(\frac{n}{2}\right)$$

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

$$\leq 2 \times c \times \frac{n}{2} \log \frac{n}{2} + an$$

$$\leq cn\log n - cn\log 2 + an$$

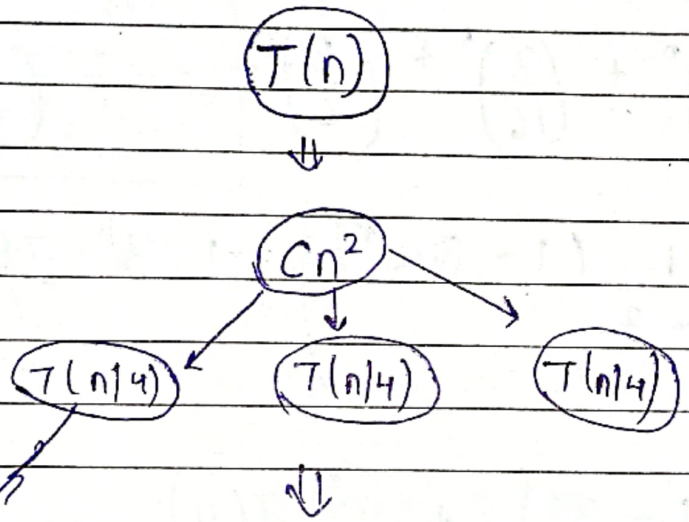
$$\leq cn\log n + (a-c)n$$

$$\leq cn\log n$$

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

3) Recursion tree method →

$$T(n) = 3T\left(\left[n/4\right]\right) + cn^2$$

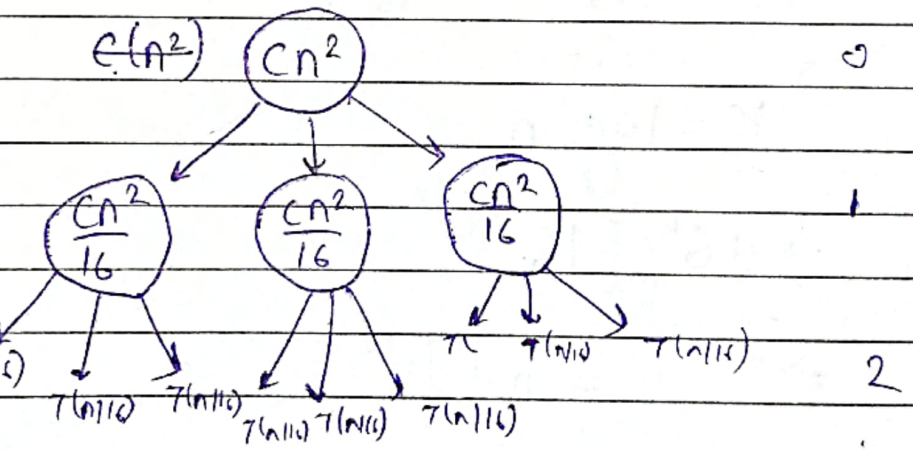


Cost

$3^0 * c\left(\frac{n}{4^0}\right)^2$ $\quad \frac{n}{4^0}$ $\quad \Theta(n^2)$

$3^1 * c\left(\frac{n}{4^1}\right)^2$ $\quad \frac{n}{4}$

$3^2 * c\left(\frac{n}{4^2}\right)^2$ $\quad \frac{n}{16}$



$k = \log_4 n$

$\frac{n}{4^k}$ $\quad$ k^{th} level

$(k-1)^{th}$ level

$3^{k-1} * c\left(\frac{n}{4^{k-1}}\right)^2$ ... level $k \rightarrow 3^k * \Theta(1)$

$$T(n) = 3^0 * c\left(\frac{n}{4^0}\right)^2 + 3^1 * c\left(\frac{n}{4^1}\right)^2 + 3^2 * c\left(\frac{n}{4^2}\right)^2 + \cdots\cdots$$

$$= cn^2 \left(\frac{3^0}{(4^2)^0} + \frac{3^1}{(4^2)^1} + \frac{3^2}{(4^2)^2} + \cdots\cdots \right)$$

$$= cn^2 \left(\left(\frac{3}{16}\right)^0 + \left(\frac{3}{16}\right)^1 + \left(\frac{3}{16}\right)^2 + \cdots + \left(\frac{3}{16}\right)^{k-1} \right) + 3^k T(1)$$

$$= cn^2 \left(\frac{1}{1-\frac{3}{16}} \left(1 - (3/16)^k\right) \right) + 3^k \cdot T(1)$$

$$\leq cn^2 \times \frac{16}{13}\left(1 - \frac{3^k}{16^k}\right) + 3^k T(1)$$

$$= \quad k = \log_4 n$$

$$\Rightarrow 16^k = 16^{\log_4 n}$$

$$\Rightarrow \quad = n^{\log_4 16}$$

$$= n^2$$

$$3^k = 3^{\log_4 n} = \boxed{n^{\log_4 3}}$$

$$\boxed{n^{\log_b a} = a^{\log_b n}}$$

Substituting

$$= c_1 n^2 \left(1 - \frac{n^{\log_4 3}}{n^2}\right) + \beta n^{\log_4 3} \cdot T(1)$$

$$= c_1 \left(n^2 - n^{\log_4 3}\right) + n^{\log_4 3} \cdot T(1)$$

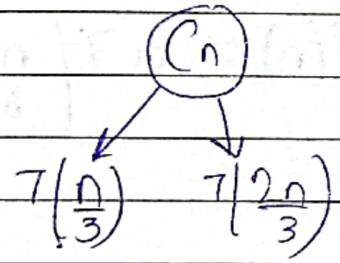
$$= c_1 n^2 - n^{\log_4 3} (-c_1 + T_1)$$

$$= \boxed{c_1 n^2 + c'' n^{\log_4 3}}$$

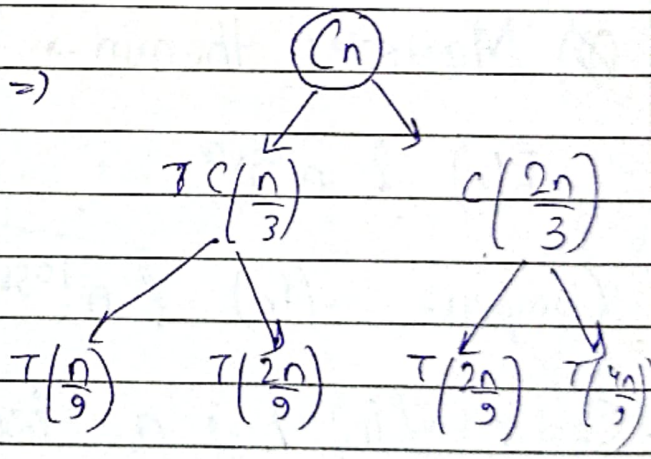
① $T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + cn$

Sol:

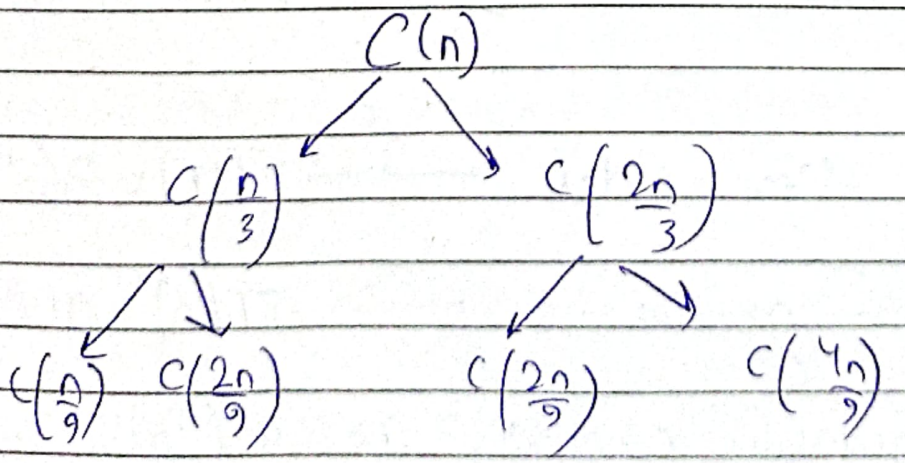
$T(n) \rightarrow$



$\Rightarrow$



$\Rightarrow$



2^0

2^1

2^2

2^k

n $\quad T(n) = cn\log_{3/2}(n) = O(n\log_{3/2} n)$

$\frac{n}{(3/2)^1}$

$\frac{n}{(3/2)^2}$

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

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

(*) Master's theorm → $T(n) = aT\left(\frac{n}{b}\right) + f(n)$

$f(n)$ & $n^{\log_b a}$

Compare $f(n)$ & $n^{\log_b a}$

Case 1: $f(n)$ has a lower order of growth than $n^{\log_b a}$ by a factor n^{ϵ} $\epsilon > 0$ $\Rightarrow T(n) = O(n^{\log_b a})$

Case 2: —"— same order —"— $T(n) = \Theta(n^{\log_b a} \log n)$

Case 3: —"— greater $\quad T(n) = O(f(n))$

→ regularity condition i.e $a \cdot f(n/b) \leq c \cdot f(n)$ for some constant $c > 0$

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

$\therefore a = 4 \quad b = 2, f(n) = n \quad n^{\log_b a} = n^{\log_2 4} = O(n^2)$

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

$a = 4 \quad b = 2 \quad f(n) = n^2 \quad n^{\log_b a} = n^2 \quad T(n) = O(n^2 \log n)$

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

$a = 4 \quad b = 2 \quad f(n) = n^3 \quad$ ~~$T(n) = O(n^3 \log n)$~~

$4\left(\frac{n}{2}\right)^3 \leq c \cdot n^3 \Rightarrow c = 1$

$\therefore f(n) > T(n/2) \Rightarrow \boxed{T(n) = O(n^3)}$

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

$n^{\log_b a} = n^2$

$\frac{n^2 \log n}{n^2} = \log n \quad \therefore$ logarithmically greater so masters theorem fail

Q) $T(n) = 3T\left(\frac{n}{4}\right) + n \log n$

$n^{\log_b a} = n^{\log_4 3}$

$a f\left(\frac{n}{b}\right) \leq c \cdot f(n) \Rightarrow 3 \frac{n}{4} \log \frac{n}{4} \leq c \cdot n \log n$

$c = \frac{3}{4}$

$\Rightarrow \frac{n}{4}\log n - \frac{n}{4}\log 4 \leq n \log n$

$\Rightarrow \quad 0 \leq$

① $T(n) = 2T(\sqrt{n}) + \log n$

Assume $n = 2^m \Rightarrow m = \log_2 n$

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

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

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

$a = 2 \quad b = 2 \quad f(m) = m \quad m^{\log_b a} = m^{\log_2 2} = m^1$

$S(m) = \Theta(m \log_2 m)$

$T(2^m) = \Theta(m \log_2 m)$

$T(n) = \Theta(\log_2 n \log_2 \log_2 n)$

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

Sol: Assume $n = 2^i$ $\quad O(n \log n)$

$C(m) = m S\left(\frac{2^i}{2}\right)$ A $\qquad a = 1 \quad b = 2 \quad = 1 \; n = n^0$

$f(n) = n\log_2 n$

$a f\left(\frac{n}{b}\right) \leq c \cdot f(n)$

$\frac{n}{2} \log \frac{n}{2} \leq cn \log n$

$\frac{1}{2} n \log \frac{n}{2} \leq \frac{1}{2} n \log n$

$T(n) = O(n \log_2 n)$

Divide And Conquer ⇒

Control Abstraction ⇒

```
DAC(P){
    if small(p)
        then solve p directly
    else{
        Divide problem P into k subproblems P1, P2 .... Pk
        Recursively solve P1, P2, .... Pk using DAC
        Combine the solution of P1, P2, ... Pk to get
        solution to original problem
    }
```

$T(n) = T(n_1) + T(n_2) + \cdots + T(n_k) + f(n)$ → time to divide + time to combine

$T(a) = c$

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

Binary search →

$T(n) = 1 \cdot T(n/2) + c$

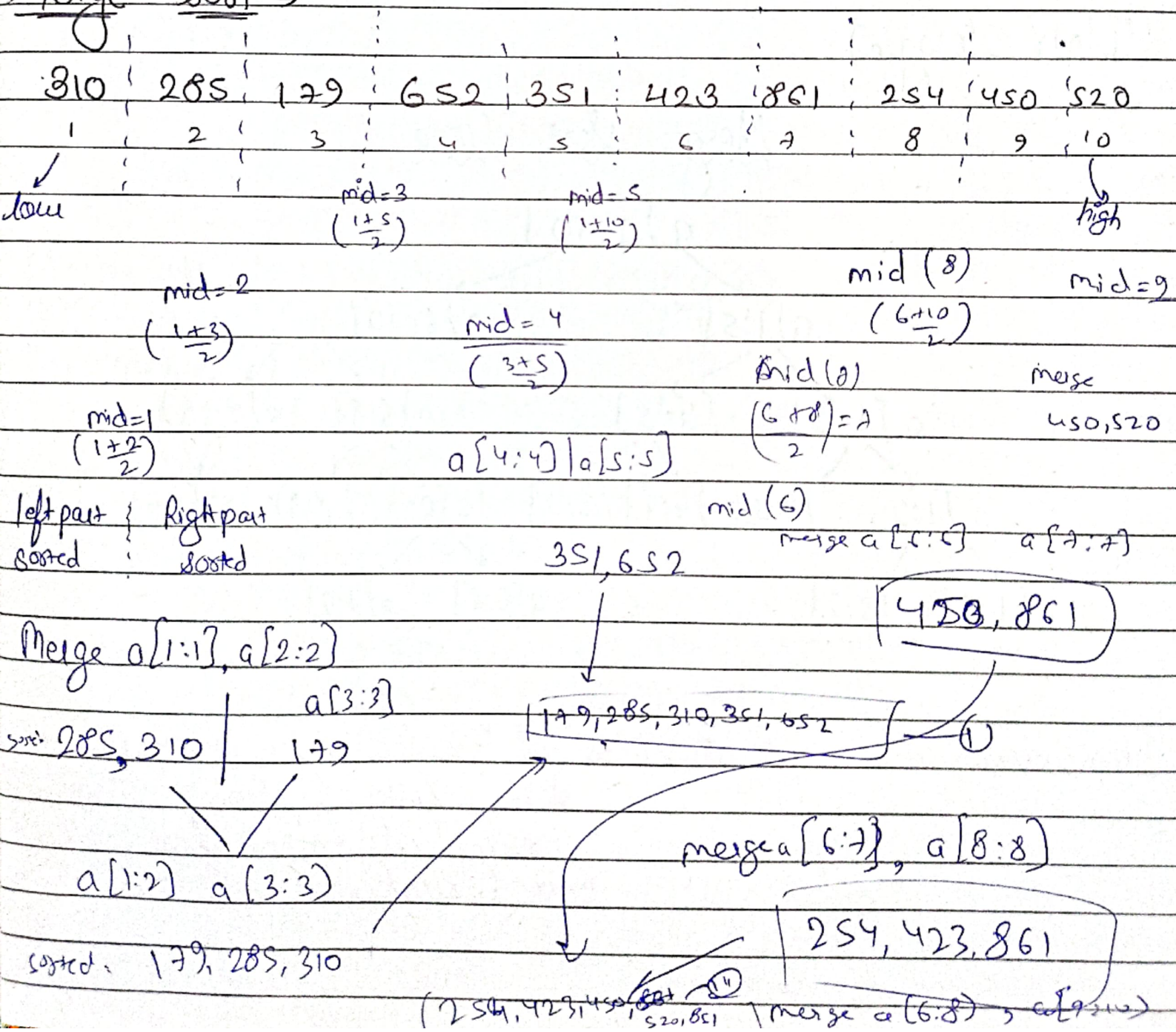
$a = 1, b = 2$ $\quad n^{\log_b a} = n^{\log_2 1} = c$

$$\boxed{f(n) = c}$$

$T(n) = O(\log_2 n)$

Best case = $O(1)$
Average case = $O(\log_2 n)$
Worst case = $O(\log_2 n)$

Merge sort ⇒



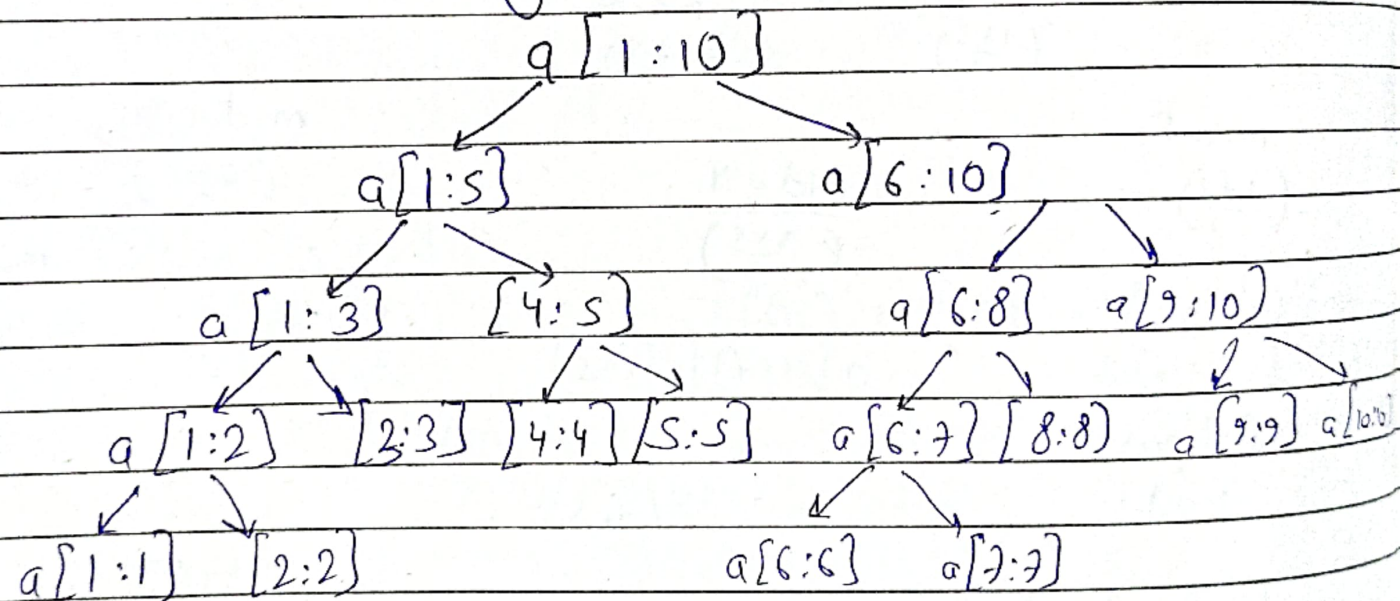
Merge ① , ② ,

173, 254, 285, ~~423~~, 310, 423, 450, 520, 652, 854

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

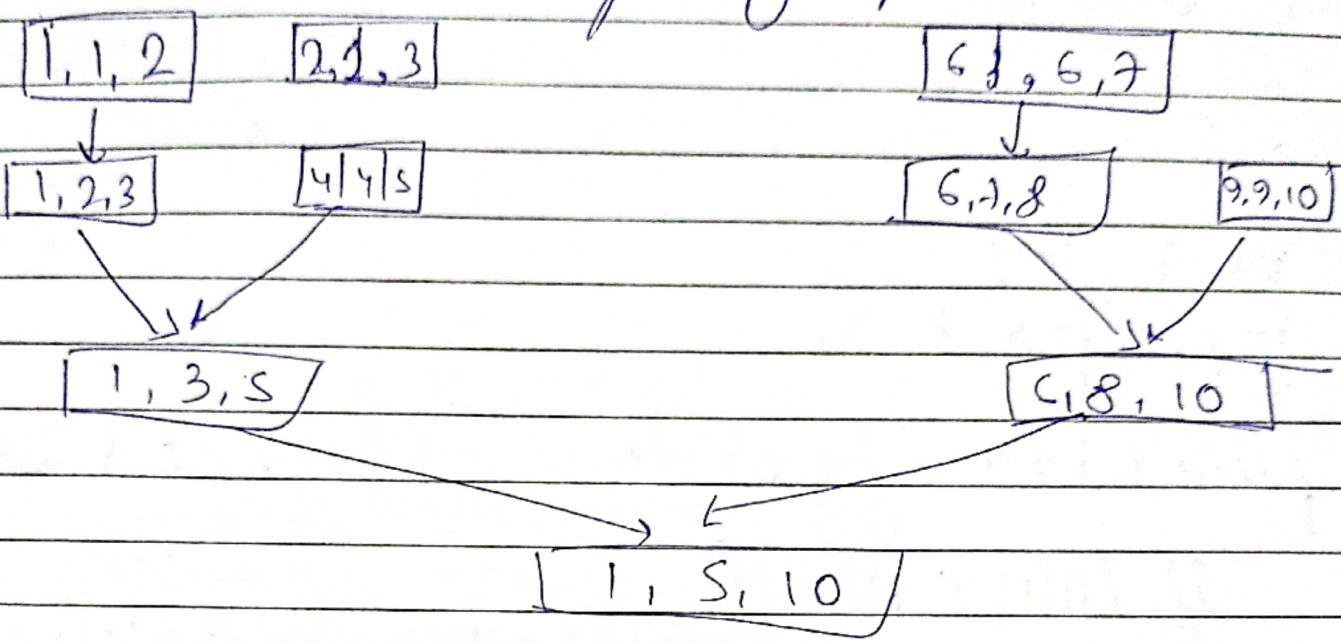
$T(n) = O(n \log_2 n)$

Merge Sort Tree



Improvement in merge sort ⇒ ~~X~~ Fixed length tak divide karo uske baad simple sorting algorithm jaise selection sort quadratic lagta do.

Calls of merge procedure



Merge k sort using link array ⇒

```
int mergesort (low, high){
    if (low == high) return low;
    else if (low < high) {
        mid = (low + high) / 2
        q = mergesort (low, mid)
        r = mergesort (mid+1, high)
        return merge (q, r)
    }
}
```

```
int merge(q,r){

    int i,j,k;
    i=q, j=r, k=0;

    while(i≠0 & j≠0) do          // while both list are non empty
    {
        if (A[i] ≤ A[j]){
                Link[k]=i ; k=i, i=Link(i);}
        else
            {Link[k]=j ; k=j ; j=Link(j);}
    } // end while

    if (i==0) // first list excluded
        Link(k) =j;
    else
        Link(k)=i;

    return Link(0);
}
```

50	10	25	30	15	70	35	55
1	2	3	4	5	6	7	0

$i=1 \quad j=2 \quad k=0$

Link	0	1	2	3	4	5	6	7	8
	2		1						

a[2] a[1]

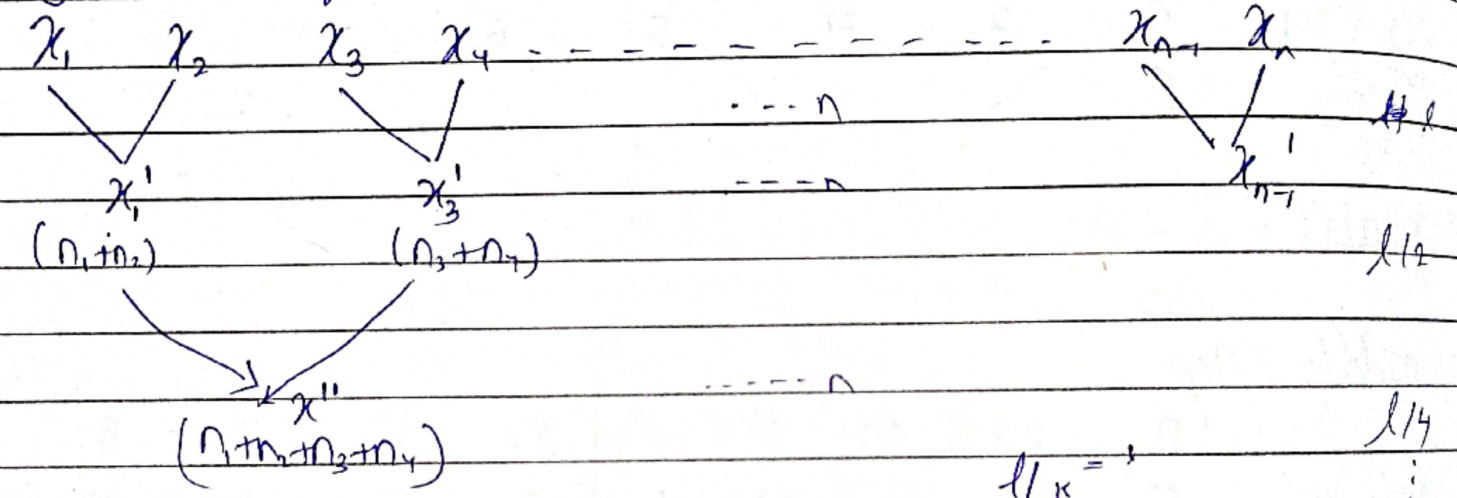
Some middle step

			[]	1	2	3	4	5	6	7	8
q	r	p	5	0	3	4	1	7	0	8	6
2	5		50	50	10	25	30	15	70	25	55
3	5										
3	7		a[2]	<	a[5]						
4			a[3]	>	a[5]						
2	5		2								
			2								

Q) You are given k sub array of size n. Merge them in $O(n \log k)$

variable

Merge in pairs of two



$\therefore$ Total $= \boxed{n \log_2 l}$

Prog. assign x $\Rightarrow$ Construct a $(x_{1min}, x_{2min}, x_{3min}, \ldots)$ heap of

→ Remove the root $(\log l)$
Say x_{3min} is at the root. Insert the next element from x_3 list & insert it in heap

Keep doing it until the heap becomes 0.

QUICK SORT

1	2	3	4	5	6	7	8	9	10	i	p
65	70	75	80	85	60	55	50	45	∞		
	i=2							p=9			
(65)	45	50	~~55~~	~~60~~	~~85~~	~~80~~	~~75~~	70			
Pivot		i=3	i=4	i=5		p=6	p=7	p=8			

50 75

55 80

60 85

p i=5 swap(p,i)

60	45	50	55	65	85	80	75	70	∞

n+1 comparison

Partioning = O(n)

Partioning element is placed at K=5

A[1...4] A[6...9]

Applying partioning to A[1....4)

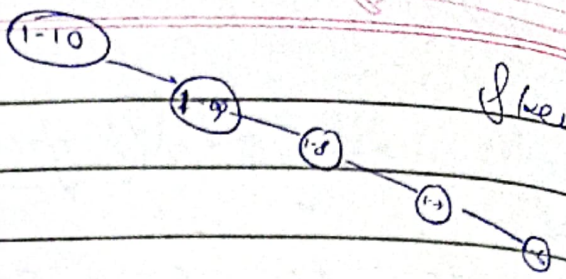
60	45	50	55
	i=2		p=4 i=5

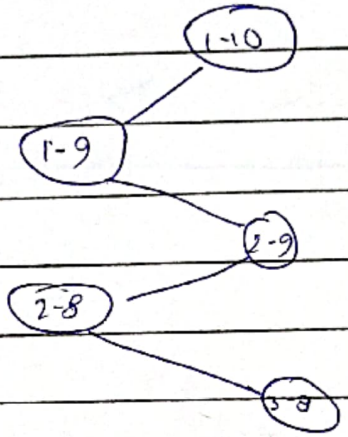
55	45	50	60

50	45	55	60
45	50	55	60

Sorted array → $O(n^2)$

Reverse sorted → $O(n^2)$





int Partition (Type a[], int m, int p)

{ Type v = a[m], int i = m, j = p;

do {

do i++;

while (a[i] < v);

do j--;

while (a[i] > v);

if (i < j) interchange (a[i] & a[j])

}

while (i < j)

// Interchange v & a[j]

a[m] = a[j], a[j] = v;

return j;

```
void Quicksort (int p, int q)
// a[1 ---- n]   a[n+1] = ∞
{ if (p<q) // if there are more than one element in array
  { int j = Partition (a, p, q+1);
    // j is the position of partition element
    // Solve the subproblem
    Quicksort (p, j-1);
    Quicksort (j+1, q);
  }
}
```

Average case analysis →

$$C_{avg}(n) = (n+1) + \sum_{K=1}^{n} P(K) \{ C_{avg}(K-1) + C_{avg}(n-K) \}$$

$$C_{avg}(n) = n+1 + \frac{1}{n} \sum_{k=1}^{n} \{ C_{avg}(K-1) + C_{avg}(n-K) \}$$

$$C_A(n) = n+1 + \frac{1}{n} \{ C_a(0) + C_a(n-1) + C_a(1) + C_a(n-2) + C_a(2) + C_a(n-3) + \cdots + C_a(n-1) + C_a(0) \}$$

$$= n+1 + \frac{1}{n} 2 \{ C_a(0) + C_a(1) + C_a(2) + \cdots + C_a(n-1) \}$$

Both side, multiply by n

$n C_a(n) = n(n+1) + 2\left(C_a(0) + C_a(1) + \cdots + C_a(n-1)\right)$ — ①

For $n = n-1$,

$(n-1) C_a(n-1) = (n-1)(n) + 2\left(C_a(0) + C_a(1) + \cdots + C_a(n-2)\right)$

Sub,

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

$\Rightarrow n C_a(n) - n C_a(n-1) + C_a(n-1) = 2n + 2 C_a(n-1)$

$\Rightarrow n(a_n) - (n+1) C_a(n-1) = 2n$

Div ③ by $n(n+1)$

$\Rightarrow \frac{n}{n(n+1)}(a - \frac{C_a(n-1)}{n} = \frac{2}{(n+1)}$

$\Rightarrow \frac{C_a(n)}{n+1} = \frac{2}{(n+1)} + \frac{C_a(n-1)}{n}$

Direct iterative substitution method

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

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

Base condition,

$$C_A(0) = C_A(1) = 0$$

$$= \frac{C_A(n-(n-1))}{2} + \frac{2}{3} + \cdots\cdots \frac{2}{n-1} + \frac{2}{n} + \frac{2}{n+1}$$

$$= \frac{C_A(1)}{2} + \frac{2}{3} + \frac{2}{4} + \cdots\cdots \frac{2}{n} + \frac{2}{n+1}$$

$$= \frac{C_A(1)}{2} + 2\left(\frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \cdots\cdots \frac{1}{n+1}\right)$$

- $\sum_{i=3}^{n+1} \frac{1}{i} \leq \sum_{i=2}^{n} \frac{1}{i}$

$$\sum_{i=2}^{n} \frac{1}{i} = \int_{x=2}^{x=n} \frac{1}{x}dx = [\log x]_2^n = \log_e n - \log_e 2$$

$$\frac{C_A(n)}{n+1} = \frac{0}{2} + 2\left[\log_e n - \log_e 2\right]$$

$$C_A(n) = 2(n+1)(\log_e n - \log_e 2)$$

$$= O(n \log_e n)$$

Partition types →

(i) Random no. as pivot

2) Median as pivot (O(n)) time is req to find median

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

3) Pick up pivot as median of three low, mid, high of array.

1) -ve to left, +ve to right in an array.

Algo: left → i, right → j. Then traverse, if a[i] < a[pivot] swap.

Break when i >= j

Q) Nuts & Bolts problem →

Nut ko select karo, bolt wale ko partition karo fir usi bolt ko use karo nut ko partition karne ke liye. Fir continue. ∴ Time = (n log n)

Strassen's Matrix Multiplication

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

A_{11}

$P = (A_{11} + A_{22})(B_{11} + B_{22})$

$Q = (A_{21} + A_{22}) B_{11}$

$R = (A_{11})(B_{12} - B_{22})$

$S = A_{22}(B_{21} - B_{11})$

$T = (A_{11} + A_{12}) B_{22}$

$U = (A_{21} - A_{11})(B_{11} + B_{12})$

$V = (A_{12} - A_{22})(B_{21} + B_{22})$

$C_{11} = P + S - T + V$

$C_{12} = R + T$

$C_{21} = Q + S$

$C_{22} = P + R - Q + U$

$$T(n) = 7T\left(\frac{n}{2}\right) + 18cn^2 \approx O(n^{2.81})$$

$$n^{\log_b 9} = n^{\log_2 7} = n^{2.81}$$

If n is odd, add 1 row of 0 & 1 col of 0

These division will continue till you get a single element matrix

TOMMOROW

Long integer multiplication optimization.

$$A = \begin{bmatrix} 1 & 2 & 1 & 2 \\ 2 & 0 & 1 & 2 \\ 1 & 0 & 2 & 1 \\ 2 & 1 & 0 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 2 & -1 & 0 \\ 2 & 1 & 0 & 1 \\ -1 & 2 & 0 & 2 \\ 0 & 1 & 1 & 2 \end{bmatrix}$$

Long integer multiplication →

x, y {n digits integer}

$x = a2^{n/2} + b$

$y = c2^{n/2} + d$

$x \cdot y = ac2^n + ad2^{n/2} + bc2^{n/2} + bd$ $\quad T(n) = 4T(n/2) + cn$

$\neq$

$\therefore (bc + ad) \equiv (a+b)(c+d) - ac - bd$

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

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

$= 9T\left(\frac{n}{4}\right) + 3\frac{cn}{2} + cn$

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

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

$= \left(\frac{3}{2}\right)^k T(1) + cn\left\{\frac{(3/2)^k - 1}{3k-1}\right.$

$\therefore n = 2^k \quad \Rightarrow \boxed{k = \log_2 n}$

$$= \left(\frac{3}{2}\right)^{\log_2 n} \cdot a + \frac{cn}{1/2}\left\{\left(\frac{3}{2}\right)^{\log_2 n} - 1\right\}$$

$$\left(a^{\log_b n} = n^{\log_b a}\right)$$

$$= \left(\frac{3}{2}\right)^{\log_2 n} = n^{\log 3 - \log 2}$$

$$\Rightarrow a n^{\log_2 3} + \frac{cn}{1/2}\left(n^{\log 3 - 1} - 1\right)$$

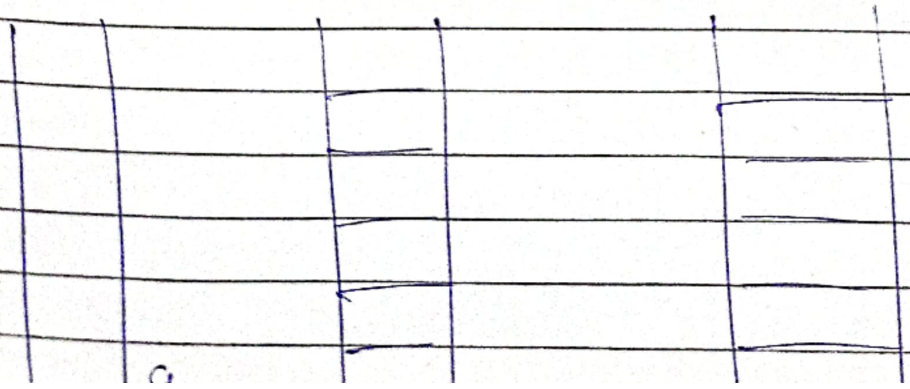
$$= a^{\log_2 3} + 2cn^{\log_2 3} - 2cn$$

$$T(n) = O\left(n^{\log_2 3}\right)$$

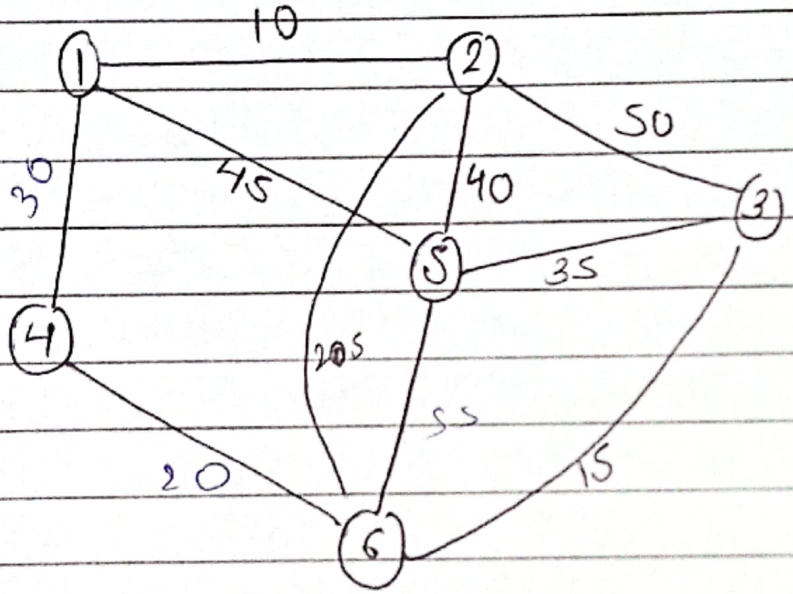
Space Complexity → int sum(int a, int b){ return a+b; }
O(1)

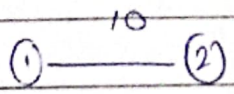
Recursive sum,

```
Rsum(int a[], int n){
    if (n<0) return 0
    else return Rsum(a[], n-1)}
```



Min Cost each → Create a heap O(e)
↳ selecting min edge - O(log e)





Near(1) = 0 Near(2) = 0

Kruskal's → Disjoint set
→ union, find
→ weighted union.

Colapsing find

IN EXAM, DRAW SET STRUCTURE

- (1) —10— (2) Union(S1, S2)

S1 S3 S4 S5 S6

(1) (3) (4) (5) (6)

(2)

- (3, 6) find(3), find(6)

S3 S6

(1) —10— (2)

(3) —15— (6)

S1 S2 S4 S5

(1) (3)

(2) (6)

- (4, 1) find(4) find(6)

S4 S3

(1) —10— (2)

(4) —20— (6) —15— (3)

S1 S4 S5

(1) (4) (5)

(2) (3) (6)

- (2, 6) find(2) find(6)

S1 S4

(1) — (2)

(6) (3)

(4)

S1 S5

(1) (5)

(2) (4)

(3)

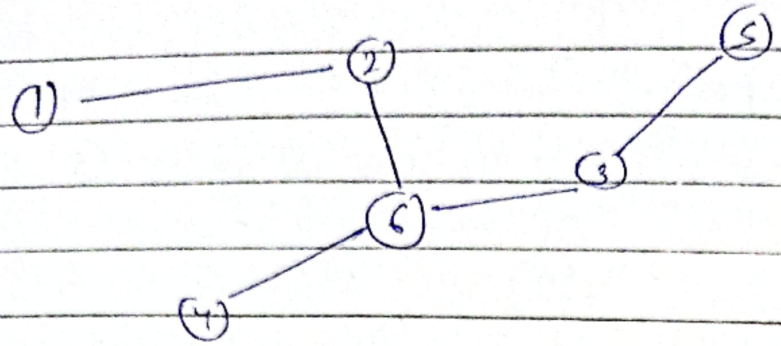
(6)

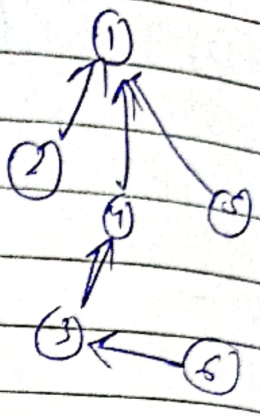
- (1, 4) find(1) find(4) same set so ignore

S1 S1

- (3, 5) find(3) find(5)

S1 S5





Kruskal's Algorithm

procedure Kruskal (E, Cost, n, T, mincost)

$E = \{(1,2), \ldots\}$ edges $n \rightarrow$ no. of vertices

// Cost = [ ] T : [St End]

real mincost, Cost (1:n, 1:n)

int Parent (1:n), T(1:n-1, 2), n, j, k;

// for storing disjoint set structure

Construct a heap out of edge costs O(E)

Parent ← -1 // each vertex is in diff set O(n)

i ← mincost ← 0

while i < n-1 & heap not empty do O(n)

  delete a mincost age (u, v) from the heap O(log e)

  reheapify O(log e)

  j ← find(u), k ← find(v)

  if (j ≠ k) then i++;

```
T[(i,1), T(i,2)] ← (u,v);
mincost ← mincost + cost(u,v);
call union(j,k)
end if
repeat
if { i ≠ (n-1) then print ("No spanning tree")
return
end Kruskal
```

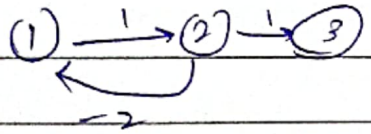
Time = $(n-1)\log e + (n-1)\log n$

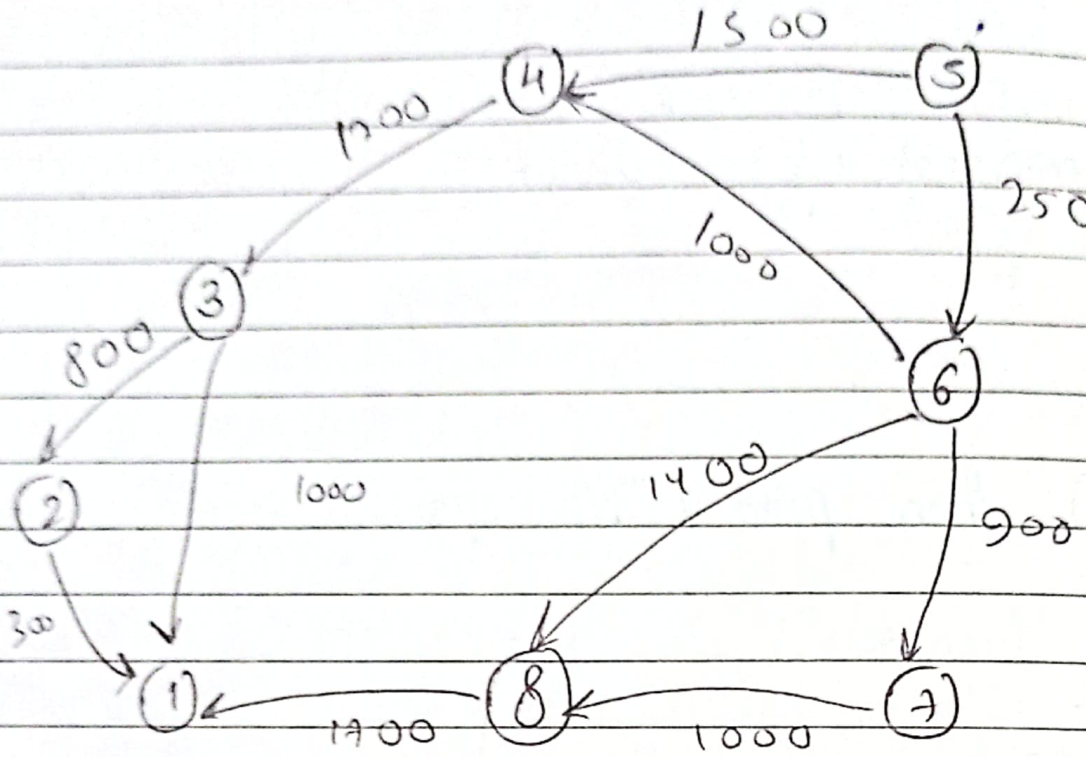
$= O(n \log e)$

Single Source Shortest Path Algo →

Won't work for negative cycles. (i.e a graph with -ve edge & a cycle)

1 → 3 ka shortest path nahi nikal sakta





S	Vertex selected	1	2	3	4	5	6	7	8
5	—	∞	∞	∞	1500	0	250	∞	∞
5	6	∞	∞	∞	1250	0	250	1150	1650
5	6, 7	∞	∞	∞	1250	0	250	1150	1650
5	6, 7, 4	∞	∞	2250	"	"	"	"	1650
5	6, 7, 4, 8	3350	∞	2250	"	"	"	"	"
5	6, 7, 4, 8, 3	3250	3050	"	"	"	"	"	"
5	6, 7, 4, 8, 3, 2,	3250	3050						

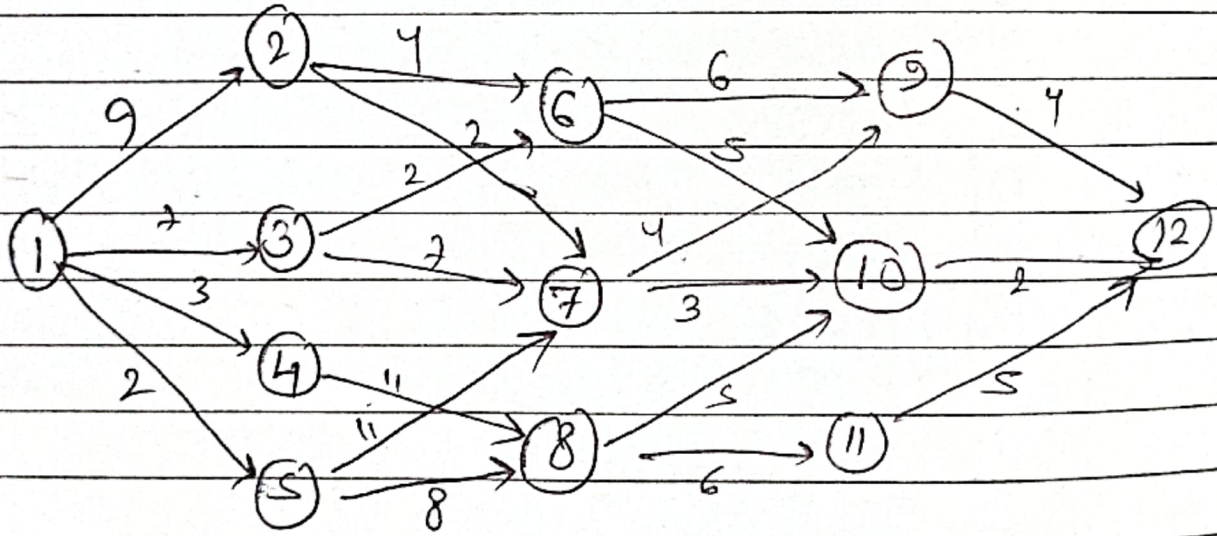
DYNAMIC PROG.

1. Overlapping subproblem are solved using it

Memoizat^n Tabulation

→ Optimal substructure → Basically each substructure of original problem must have optimal solution.
→ Overlapping substructure

Multistage graph problem →



Forward approach

→ cost (k-1, i) = c (i, t)

stage no. vertex no.

→ cost (k-2, i) = min { cost (k-1, j) + c ((6, 9), cost (k-1, 10)

$cost((k-3), 2) = \min\{cost(k-2, 6) + c(2,6),\ cost(k-2, 7) + c(2,7)\}$
, 3
, 4
, 5

$cost(k-4, 1) = \min\{cost(k-2, 2) + c(1,2)$
$cost(k-2, 3) + c(1,3)$
$cost(k-2, 4) + c(1,4)$
$cost(k-2, 5) + c(1,5)\}$

$cost(4,9) = c(9,12) = 4$
$cost(4,10) = c(10,12) = 2$
$cost(4,11) = c(11,12) = 5$

- $cost(3,6) = \min\{cost(4,9) + c(6,9),\ cost(4,10) + c(6,10)\}$
$= \min\{4+6, 2+5\}$
$= 7$ $\quad D(3,6) = 10$

$cost(3,7) = \min(cost(4,9) + c(7,9),\ cost(4,10) + c(7,10)$

$= \min(4+4, 2+3)$
$= 5$ $\quad D(3,7) = 10$

$cost(3,8) = \min(cost(4,10) + c(8,10),$
$cost(4,11) + c(8,11)\}$

$= \min\{10+5, 5+6\}$
$= 7$ $\quad D(3,8) = 10$

•

$cost(2,2) = \min\{cost(3,6) + c(2,6),\ cost(3,7) + c(2,7)\}$

$= (7+4,\ 5+2)$

$= 7$

$D(2,2) = 7$

$cost(2,3) = \min\{cost(3,6) + c(3,6),\ cost(3,7) + c(3,7)\}$

$= \min\{7+2,\ 5+7\}$

$= 9$

$D(2,3) = 6$

$cost(2,4) = \min\{cost(3,8) + c(4,8)\}$

$= 7 + 11$

$= 18$

$D(2,4) = 8$

$cost(2,5) = \min\{cost(3,7) + c(2,7),\ cost(3,8) + c(5,8)\}$

$= \min(5+11,\ 7+8)$

$= 15$

$D(2,5) = 8$

•

$cost(1,5) = \min\{cost(2,2) + c(1,2),\ cost(2,3) + c(1,3),\ cost(2,4) + c(1,4),\ cost(2,5) + c(1,5)\}$

$= 7+9,\ 9+7,\ 18+3,\ 15+2$

$= 16, 16, 21, 17$

Path : 1 2 7 10 12
Source Target

Backward approach -

cost (2, i)
↳ shortest path from vertex i to source vertex in stage 1

cost (2, 2) = c(1, 2) = 9 D(2, 2)
cost (2, 3) = c(1, 3) = 7
= c(1, 4) = 3
= c(1, 5) = 2

cost (3, 6) = c(2, 6) + cost (2, 2)

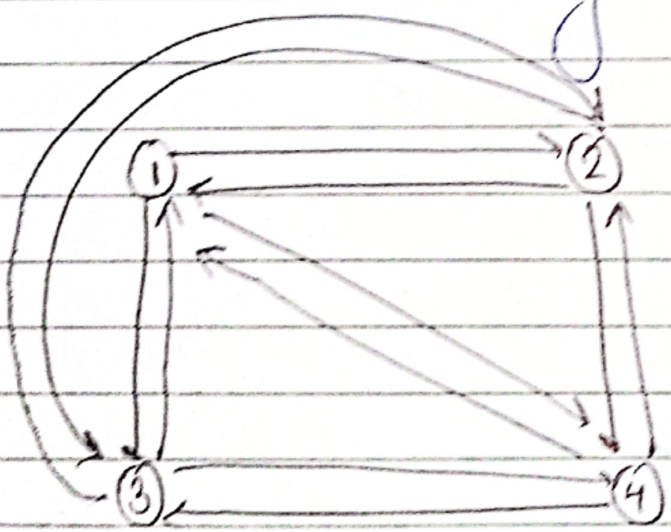
Travelling Salesperson Problem →

$$g(1, V-\{1\}) = \min\{c_{1k} + g(k, V-\{1,k\})\}$$

Shortest path starting at vertex k going thru remaining n-2 vertices & returning at vertex 1

Generalized eq,

$$g(i,s) = \min\{c_{ij} + g(j, s-\{j\})\}$$



	1	2	3	4
1	0	10	15	20
2	5	0	9	10
3	6	13	0	12
4	8	8	9	0

$g(2,\phi) = c_{21} = 5$ $\quad g(3,\phi) = c_{31} = 6$ $\quad g(4,\phi) = c_{41} = 8$

$g(2,\{3\}) = c_{23} + g(3,\phi) = 9+6 = 15$
$2 \to 3 \to 1$

$g(2,\{4\}) = c_{24} + g(4,\phi) = 10+8 = 18$

$2 \to 4 \to 1$

$g(3,\{2\}) = c_{32} + g(2,\phi) = 13+5 = 18$

$g(3,\{4\}) = c_{34} + g(4,\phi) = 12+8 = 20$

$g(4,\{2\}) = c_{42} + g(2,\phi) = 8+5 = 13$

$g(4,\{3\}) = c_{43} + g(3,\phi) = 9+6 = 15$

$g(2,\{3,4\}) = \min(c_{23} + g(3,\{4\}), c_{24} + g(4,\{3\}))$

$g = \min(20+9, 10+15) = 25$

Youtube

$$\sum_{k=0}^{n-2} (n-1)\binom{n-2}{k} = (n-1)\,2^{n-2}$$

$$\sum_{k=0}^{n-2} k\,(n-1)^{n-2}C_k = n^2 \cdot 2^n$$

8