→ celebrity Identify pgm (No. of Micro Writes)

→ Player Game (Winner)

→ nballs

	10n	$2n^2$
n=1	10	2
n=2	20	8
n=3	30	18
n=5	50	50
n=6	60	72

$5 \leq cn^2 n$

$5 \leq cn_0$

$10n = 2n^2$ $\quad c=1, n_0=5$

$f(n) \in O(g(n)) \Leftrightarrow \{ f(n) \mid \exists c, n_0 > 0, \; f(n) \leq c * g(n) \; \forall n \geq n_0 \}$

set

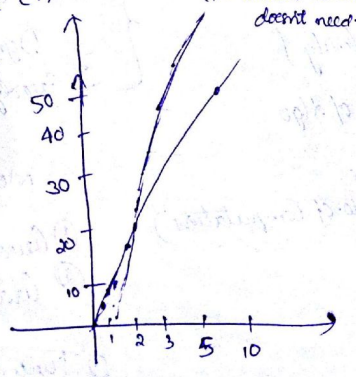
one particular function in set

Ex: $2n+30 \leq 3n$

$O(n^2) = \{ n^2, 2n^2, cn^2 + c_2 n + c_3, c_3 \}$

$c_3 \leq cn^2$

(For smaller values it doesn't need to satisfy)



→ $100n$ vs $1/5 n^2$

$100n \leq c\left(\frac{1}{5}\right) n^2 \cdot n_0$

$500 \leq cn_0$

$c=1, n_0 \geq 500$

$f(n) \in O(g(n))$, $\exists c, n_0 > 0$, $f(n) \leq cg(n)$ $\forall n \geq n_0$

	100n	$n^2/5$
n=1	100	1/5
n=5	500	5
n=700	10^4	$10^4/5$
n=500	5×10^4	5
		$\frac{25 \times 10^4}{5}$

⇒ $\log n$ vs n^2.

$\log n \leq c(n^2)$

$c=1, n_0 \geq 1$

⇒ $\log n$ vs n

$\log n \leq cn$

$c=1, n_0 = 1$.

$\log(\log n) \leq \log c + \log n$.

$\log(\log n) \leq \log n$

$\log n$ is small

⇒ $\log n$ vs $\sqrt{n}$.

$\log n \leq c(n^{1/2})$

$\log(\log n) \leq \log c + \frac{1}{2}\log n$

$\log n$ is small.

Informally:

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

⇒ $\log n$ vs $n^{0.00001}$.

$\log(\log n)$ $10^{-5} \log n$.

$\log n$ is small

$n = 10^{3}$

⇒ 2n vs 3n

Celebrity Algo:

n ppl.

Pose a Qstn to x., if he knows y, then throw x

else throw y.

∴ (n-1) Qstns

```
Input: n
Ask Qstn to x.
   if he knows y
     {
       throw x;
     }
   else
     {
       throw y
     }
Output: (n-1) Qstns
```

```
⇒ find Celebrity (S)
{
  //ip : S → a set of n people
  //op: one person
    while (|S| >= 2):
    {
      // let us choose two people say x & y.
        if (x knows y)
          S = S-x;
        else
          S = S-y;
    }
    Print (The person available in S);
}
```

(nlogn)

⇒ An Array is given (A). Consists of n Elements, consists of x. does there Exists any two values in n such that the sum = x.

(nlogn)

```
Summation(A)
{
  //ip:- A consists of n Element, x.
      Sort the Array
      Binary Search;
}
```

Sort the Array
Binary Search

Minimum Element.

```
min(A){
min(int A[], low, high){ if(low==high)
    mid = (low+high)/2;   return A[low];
    k1 = min(A[low], low, mid);
    k2 = min(A, mid+1, high);
    return minimum(k1, k2);
}
```

④ $\log 1 + \log 2 + \cdots + \log(n-1) + \log n \geq c_1 n \log n$

$\log 1 + \log 2 + \cdots + \log(n-1) \geq \left(\frac{1}{\log b}\right)(c_1 n - 1)$

$\log(n-1)! \geq (\log n)[c_1 n - 1]$

$\log 1 + \log 2 + \cdots + \log(n-1) + \log n \leq c_2 n \log n$

$\log(n-1)! \leq \log n (c_2 n - 1)$

$\log 1 + \log 2 + \cdots + \log(n-1) \geq c_1 \log n + c_1 \log n + \cdots + c_1 \log n$

$\log n! = \log(n)(n-1)\cdots 1)$

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

$(\log n + \cdots + \log n)$ n times $\geq \log n + \log(n-1) + \cdots + \log 1$

$n \log n \geq \log(n)!$

① $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$

② $\sum_{k=0}^{a} k \cdot x^k = \frac{x}{(1-x)^2}$

③ $\sum_{i=1}^{n} \frac{1}{i} \approx \log n$

④ $\log n! \geq c_1 n \log n$ & $\log n! \leq c_2 n \log n$

find values of c_1 & c_2

① $2^0 + 2^1 + \cdots + 2^n$

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

②

$0 \cdot x^0 + 1 \cdot x^1 + 2 \cdot x^2 + \cdots + = \left(\frac{1}{1-x}\right) + \left(\frac{x}{(1-x)^2}\right)$

$1 + x(1 + 2x + \cdots) = \left(\frac{a}{1-x}\right) + \frac{dx}{(1-x)^2} = 0 + \frac{1 \cdot x}{(1-x)^2}$

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

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

$\times$

$= 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$

$= 1 + 1 - \frac{1}{2} + \frac{1}{2} - \frac{1}{3}\ \}$

$= 1 + \frac{1}{2} + \frac{1}{3} = \frac{3}{2} + \frac{1}{3}$

$= \frac{1}{2}\left[1 + 1 - \frac{1}{2} + \frac{1}{2} - \frac{1}{3}\right.$

$> \frac{1}{2}\left[\frac{5}{3}\right]$

$1 - \frac{1}{3} \quad \frac{5}{6} + 1 = \frac{11}{6}$

$\int_1^n \frac{1}{x} = \log x$

③ $\int_1^n \frac{1}{x} = [\log x]_1^n = \log n - \log 1$

$= \log n$

④ $\log n! \geq c_1 n \log n$

④ $\log n + \log(n-1) + \dots + \log\left(\frac{n}{2}\right) \geq \frac{n}{2}\log\left(\frac{n}{2}\right)$

$\Rightarrow \left\{\log n + \dots + \log\left(\frac{n}{2}\right)\right\} + \left\{\log\left(\frac{n}{2}-1\right) + \dots + \log 1\right\}$

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

$\Rightarrow \log n! \geq \frac{n}{2}\log\left(\frac{n}{2}\right)$

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

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

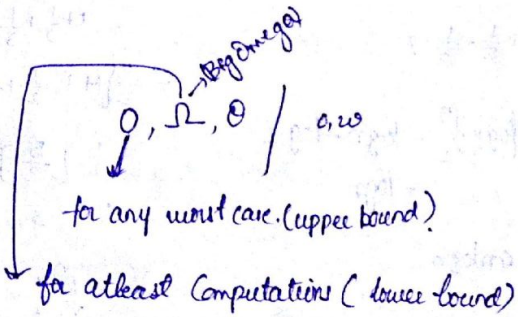
$\geq \frac{n}{4}\log n$

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

$\exists c, n_0 > 0$

$\forall n \geq n_0$

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



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

$\exists c, n_0 > 0$

$\forall n \geq n_0$

$f(n) \geq c\, g(n)$

Ex: $n^2 = \Omega(n)$

$\frac{1}{99}n^2 \geq 10^{20}(n) + 100$

$n^2 = \Omega(3n - 100)$

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

$\Omega \rightarrow$ Instead of finding Running time

$\Rightarrow \Theta(g(n)) = \left(O(g(n))\right) \cap \left(\Omega(g(n))\right)$

$\exists c_1, c_2, n_0$

$\forall n \geq n_0,\ c_2 g(n) \leq f(n) \leq c_1 \cdot g(n)$

Ex: $100n^2 = \Theta(n^2)$

$c_2(n^2) \leq 100n^2 \leq c_1(n^2)$

$c_1 \geq 100$

$c_2 \leq 100$

⑤ $100n^2 + 3n = \Theta(n^2)$

$100n^2 + 3n \geq c_1 n^2 \quad (c_1 = 1)$

$100n^2 + 3n \leq c_2 n^2 \rightarrow c_2 \geq 101$

$n^2(100 - c_2) + 3n \leq 0$

Small o:

$\Rightarrow o(g(n)) = O(g(n)) - \Omega(g(n))$

$f(n) = o(g(n))$

$\forall c > 0$

$\exists n_0 > 0$

$\forall n \geq n_0$

$f(n) < c(g(n))$

① $n = o(n^2)$

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

$\Rightarrow \lim_{n\to\infty} \frac{n}{n^2} = \lim_{n\to\infty} \frac{1}{n} = 0$

$\Rightarrow f(n) = \omega(g(n))$

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

$f(n) > g(n)$ $\forall c > 0$, $\exists n_0 > 0$, $\forall n \geq n_0$

Ex: Matrix Multiplication $= O(n^2)$

o (small o):

$\Rightarrow$ Now we want an Algo strictly better than Earlier

Ex: Matrix Multiplication $O(n^3)$

formulas:

① $a^{\log_a n} = n$

② $\log(n!) = \Theta(n \log n)$

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

$\log n! \geq \frac{n}{4} \log n$

$\log n! \leq n \log n$

$\therefore \log(n!) = \Theta(n \log n)$

③ $(2^2)^n \quad 2^{2n}$

$2^{2n} \leq 2^{2n}$ or $2^{2n} < 2^{2n} \Rightarrow$ Big O or small O

(for larger values of n)

④ $n \leq n^2 \rightarrow$ Big O or small o

⑤ $\frac{1}{2}(n^2) - 3n = O(n^2)$

$\frac{n^2}{2} - 3n \leq c_1(n^2)$

$c_1 \geq 1$

$\frac{n^2}{2} - 3n \geq c_2 n^2$ (or)

$n^2\left[\frac{1}{2} - c_2\right] \geq 3n$

$c_2 \leq \frac{1}{2}$

$c_1 n^2 \leq \frac{1}{2}(n^2) - 3n \leq c_2 n^2$

Divide by n^2

$c_1 \leq \frac{1}{2} - \frac{3}{n} \leq c_2$

$c_1 = \frac{1}{14}, c_2 = \frac{1}{2}$

$\frac{n^2}{2} - 3n \leq n^2$

$-3n \leq \frac{n^2}{2}$

$\frac{1}{2} - 1$

$\left(\frac{1}{2}-\frac{3}{n}\right) \geq 0$

$n-6 \geq 0 \Rightarrow n \geq 6$

$\Rightarrow n=7$

$\Rightarrow c_1 \leq \frac{1}{2}-\frac{3}{7}$

$\Rightarrow c_1 \leq \frac{1}{14} \Rightarrow c_1 = \frac{1}{14}$

$\Rightarrow \left(\frac{1}{2}-\frac{3}{n}\right) \leq c_2$.

$\Rightarrow c_2 = \frac{1}{2}$

$\Rightarrow$ $6n^3 = \Theta(n^2)$

$c_1 n^2 \leq 6n^3 \leq c_2 n^2$

$c_1 \leq 6n \leq c_2$.

$\times$

$(n_0 = 1)$ $(c_1 = 1, c_2 \geq 6)$ }

$\Rightarrow$ $6n^3 = \Theta(n^2)$ → False.

Case 1:

$6n^2 \leq 6n^3$

$c_1 \leq 6n$

$n_0 = 1$ ($c_1 = 1$ or 6)

Case 2:

$6n^3 \leq c_2 n^2$.

$6n \leq c_2$

$n \leq \frac{c_2}{6}$

(n is a variable which can't be bounded by a constant)

$\therefore$ $n \leq \frac{c_2}{6}$ → False → Contradiction

(7) 4 2 1 3 6 9 5 10 8

↓ Pivot

(less than or Equal to 7) → 7 (Keep on left side)

(greater than Elements) → Right side.

(7) 4 2 1 3 6 5 8 10 9

5 4 2 1 3 6 | 7 | 8 10 9

↓ pivot

(5 4 2 1 3 6

6 > 4 2 1 3 5)

5 4 2 1 3 | 6 7 8 10 9

3 4 2 1 | 5 6 7 8 10 9

3 2 1 | 4 5 6

1 2. | 3 4

1 | 2 3 4

⇒ Divide the Algo ⇒ O(n)

⇒ Combing part is not the

Quicksort

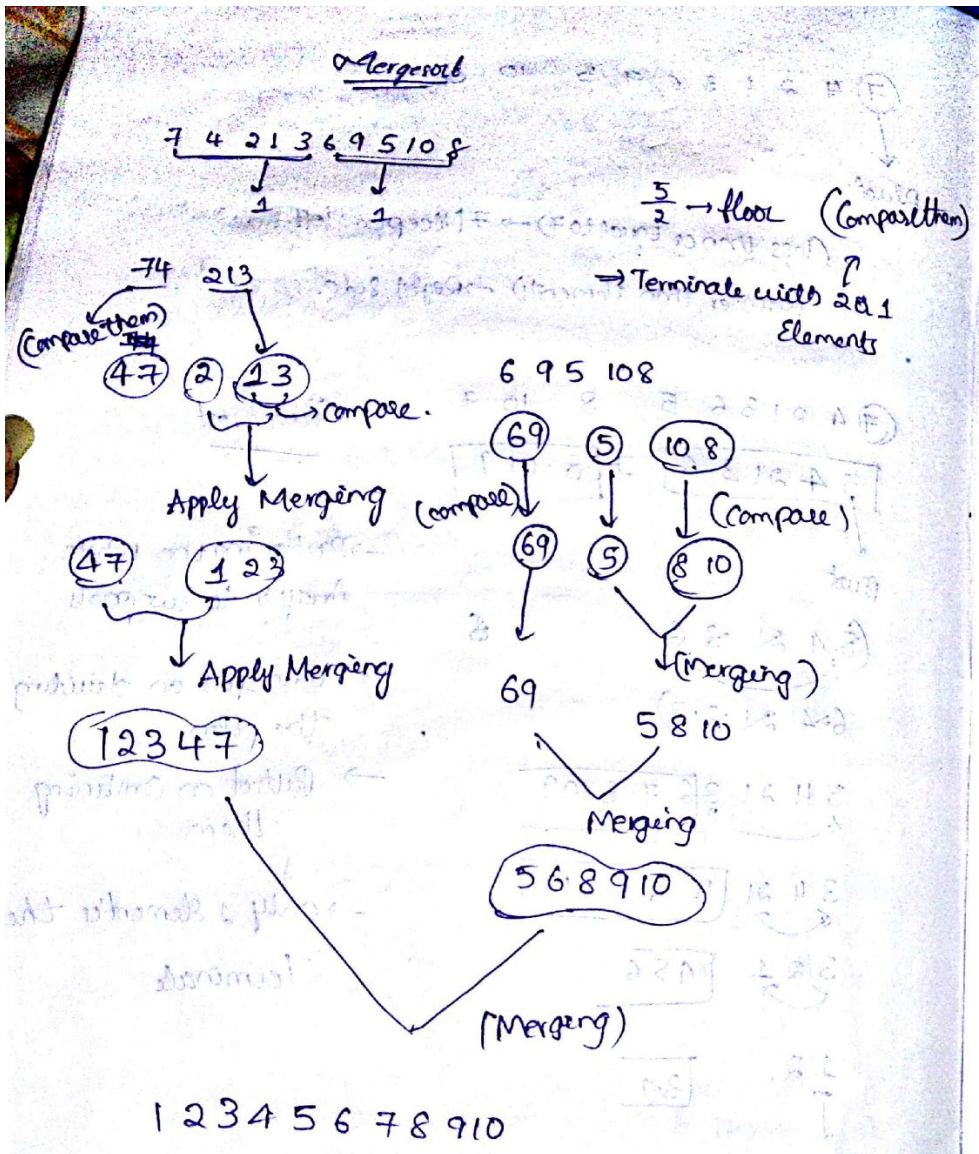
→ Partioning the whol Array in to two part

→ time Spent on divi the pblm

→ Pattrot on Combini them

→ only 1 Element is Terminate

Mergesort




```
mergesort (int A[], int low, int high)
{
  mid = ⌊(low+high)/2⌋;
  mergesort (A, low, mid);
  mergesort ( A, mid+1, high);
  merging (A, low, mid, high); → { Result is stored in Same Array (A) again }
}
```

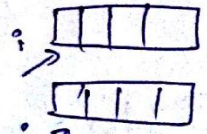
Inplace mergesort

```
merging (int A[], int low, mid, high)
{
 - copy elements of A from low to mid to Array B
 - Copy elements of (mid+1) to high to Array C

 n1 = (mid - low) + 1;
 n2 = high - (mid+1) + 1;
    i=1, j=1
 for(k=low; k=high; k++)
 {
  if (B[i] < C[j])
   { A[k] = B[i];
     i++;
   }
 else
  { A[k] = C[j];
     j++;
  }
}
```

Trick:

We initially at least 2 cases, Rest 2 cases are used throughout



(which is small keep it in Result Array & move the pointer)

① 1st Array is complete
② 2nd " "
③ Both are not complete

Tutorial

② Ternary Search

```
Ternary Search (int A[], int low, int high, int value)
{
  if(high >= low)
  { int mid1 = (2*low + high)/3;
    int mid2 = (low + 2*high)/3;
    if (A[mid1] == value)
        return mid1;
    if (A[mid2] == value)
        return mid2;
    if(value < A[mid1])
        return TernarySearch(A, low, mid1-1, value);
    if (value > A[mid2])
        return TernarySearch(A, mid2+1, high, value);
    else.
        return TernarySearch(A, mid1+1, mid2, value);
  }
}
```

① $n! = O(n^n)$

Apply log on both sides

$\log n!$ $\log n^n$

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

$\log 1 \leq \log n$

$\log 2 < \log n$

$\vdots$

$\log n \leq \log n$

$\log 1 + \log 2 + \cdots + \log n \leq n \log n$

$\log n! \leq \log n^n$

$c = 1 \;\forall\; n \geq 1, n_0 = 1.$

②

$\log n = O(n^{0.0001})$

$\log(\log n) \leq c\left(\frac{1}{10000}\right)\log n$

$\log 1 = 0, \log 2 = 0.3010$

$\log(\log 2) \leq c_1\left(\frac{1}{10^4}\right)(\log 2)$

② $\log 2 \leq c_1(2^{0.0001})$

$0.3010 \leq c_1(2^{0.0001})$

$\therefore c = 1, n \geq 1$

② $\log n \leq c(n^{0.0001})$

Apply log on both sides

$\log(\log n) \leq c * n^{10^{-4}}$

$\log(\log n) \leq \log c + \log(n^{10^{-4}})$

$\log(\log n) \leq 10^{-4} \log n$

So $n = 10^5 * 2$

$c = 1$

$5 \leq 10$ ✓

5) $5n^2 - 6n = \Theta(n^2)$

$c_1 n^2 \le 5n^2 - 6n \le c_2 n^2$

$c_1 \le 5 - \frac{6}{n} \le c_2$

$5 - \frac{6}{n} > 0$

$n = 2$

$c_1 \le 5 - \frac{6}{2}$

$c_1 \le 1$

$c_2 = 5$

$n^2 \le n^2$

(26)

(7) $\sum_{i=1}^{n} i^2 = \Theta(n^3)$

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

$c_1 n^3 \le \frac{n(n+1)(2n+1)}{6} \le c_2 n^3$

$c_1 n^3 \le \frac{2n^3 + 3n^2 + n}{6} \le c_2 n^3$

$c_1 \le \frac{1}{3} + \frac{1}{2n} + \frac{1}{6n^2} \le c_2$

$c_1 = \frac{1}{3}$

$n \ge 1$

$c_2 = \frac{1}{3}$

(3) $33n^3 + 4n^2 = \Omega(n^2)$

$33n^3 + 4n^2 \ge c_1 n^2$

$c_1 = 1, n_0 = 1$

(4) $33n^3 + 4n^2 = \Omega(n^3)$

$33n^3 + 4n^2 \ge c_1 n^3$

$c_1 = 1, n_0 = 1$

(3) $33n + 4 \ge c$

$33n + (4 - c) \ge 0$ $c_1 = 1, n_0 = 1$

X (Can't be predicted; dependent on c)

(4) $33 + \frac{4}{n} \ge c_1$

$c_1 = 33, n > 0$

$c_1 \le 33 + \frac{4}{n}$ $n_0 = 1, c_0 = 1$

(6) $2n^2 2^n + n\log n = \Theta(n^2 2^n)$

$c_1(n^2 2^n) \le 2n^2 2^n + n\log n$

$\log c_1 + n\log n + n\log 2 \le \log 2 + 2\log n + n\log 2$

$\log c_1 + \log n^n \le \log 2 + \log n^2$

$\log(c_1 n^n) \le \log(2 \cdot n^2)$

$c_1 n^n \le 2n^2$

$c_1 \le 2n^{2-n}$

$2 - n \ge 0$

$2 \ge n \Rightarrow n \le 2$

$2n^{2-n} \le c_2$

(6) $2n^2 2^n + n\log n = \Theta(n^2 2^n)$

$c_1(n^2 2^n) \le 2n^2 2^n + n\log n \le c_2(n^2 2^n)$

$c_1 \le 2 + \frac{n\log n}{n^2 2^n} \le c_2$

$c_1 \le 2 + \frac{\log n}{n(2^n)} \le c_2$

$c_1 = 2 + \frac{\log n}{n(2^n)} \le c_2$

Case 1: (Ω)

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

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

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

Case 2: (O)

$c_1 n^2 \leq c_1 n^2$

$2c_1 n^2 \leq 2c_1 n^2$

$2c_1 n^2 + n\log n \leq 2c_1 n^2 + c_1 n^2$

$n \leq n$
$n\log n \leq n^2$
$n\log n \leq c_1 n^2$

$2c_1 n^2 + n\log n \leq 3c_1 n^2$

[$c_1 = 1$, $c_2 = 3$, $n_0 = 1$]

Day

Recursion: Two elements are Such that, A[i] < A[j]

(a) (integer) — merge sort

Quicksort

```
Quicksort(int A[], int low, int high)
{
  if(low<high)
  {
    // copy elements from A to B
    j = partition(A, B, low, high);
    quicksort(B, low, j-1);
    quicksort(B, j+1, high);
  }
}
```

left values: less than or equal to pivot

Right: Strictly Greater than

```
int partition(int A[], int B[], int low, int high){
  pivot = A[low]; i = low; j = high;
  for(i = low; i <= high; i++)
  {
    if(A[i] < pivot)
    {
      B[i] = A[i];
      i++;
    }
  }
  B[...](A, low, j-1);
  return (j-1);
}
```

Recurrence Relations (Recurrence Relations)

Mergesort:

Running time of mergesort on n elements

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

($O(n)$: merging)

$$T(n) = O(n) + 2T\left(\frac{n}{2}\right) \rightarrow \text{when } n \geq 3$$

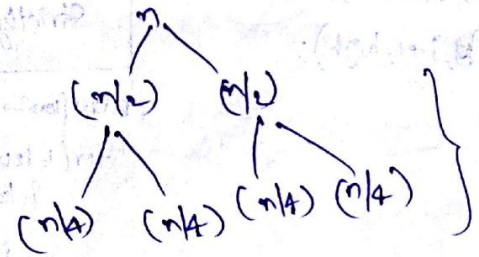
when $n = 1$ & $2 \Rightarrow$ constant time (c)

Quicksort:

Best case:

When the array is dividing equally in to two parts

$$n + \frac{n}{2} + \frac{n}{4} + \frac{n}{8} + \cdots = O(\log n) \rightarrow \text{height of the tree}$$



Terminate it when there is only 1 element

$$\Rightarrow \frac{n}{2^i} = 1$$

$$\Rightarrow n = 2^i$$

$$\log n = i \log 2$$

$$\Rightarrow i = \log_2 n$$

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

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

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

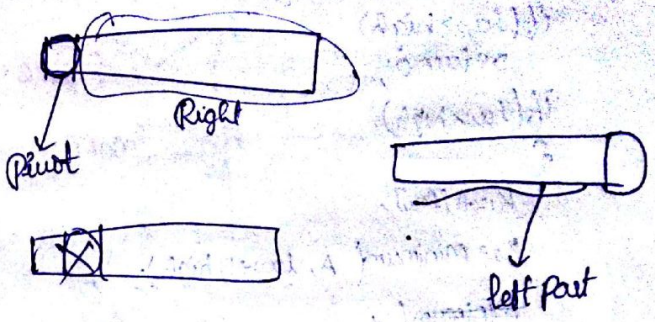
Best case:

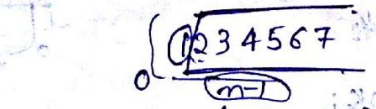
$O(n \log n)$

Best case

Worst:

when there is only 1 element in left side, rest of





$\Rightarrow$ pivot element should be kept on Right side so, on left side there are $(n-1)$ elements.

$$T(n) = O(n) + T(n-1) + T(0)$$

$T(0) \rightarrow c$

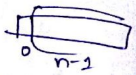
$$= O(n) + T(n-1) + c$$

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

$$\boxed{T(n) = O(n) + T(n-1)} \rightarrow \text{worst case}$$

⇒ Finding Minimum Element

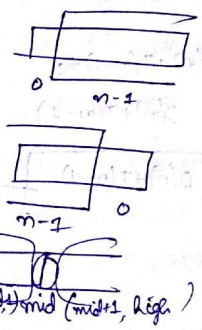
```
int Minimum (int A[], int low, int high)
{
    if (low > high)
        return 0;
    if (low < high)
    {
        K1 = A[low];
        K2 = Minimum( A, low+1, high);
        if (K1 > K2)
            return K2;
        else
            return K1;
    }
}
```



$T(n) = T(0) + T(n-1) + C_2$ → Comparing
$= C_1 + T(n-1) + C_2$
$= C + T(n-1)$

$T(n) = T(n-1) + T(0)$
$= C + T(n-1)$

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



Home Task:

Recursive Theorem
tree Method
) Master's theorem
Substitution
) Recursive Substitution Method
Characteristic Eqn Method

Recurrence Relation:-

$T(n) = 2T\left(\frac{n}{2}\right) + C$ → (Increasing a.p)
$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$ → constant ap
$T(n) = 2T\left(\frac{n}{2}\right) + O(n^2)$ → (Decreasing ap)

If no constraints are given, $n \geq 2$

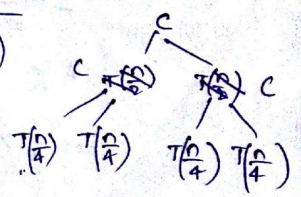
① Recursive Tree Method:

C → (Dividing & Combining) = C

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

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

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



⇒ When there is only 1 Element, we can't use terminating it

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

$\Rightarrow -C$
$-2C$
$-4C$
$-8C$
$= -2^{\log n} C$

$\Rightarrow C + 2C^1 + 2^2 C + \cdots + 2^{\log n} C$

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

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

$= O(n)$

⑥ nlogn (Any)

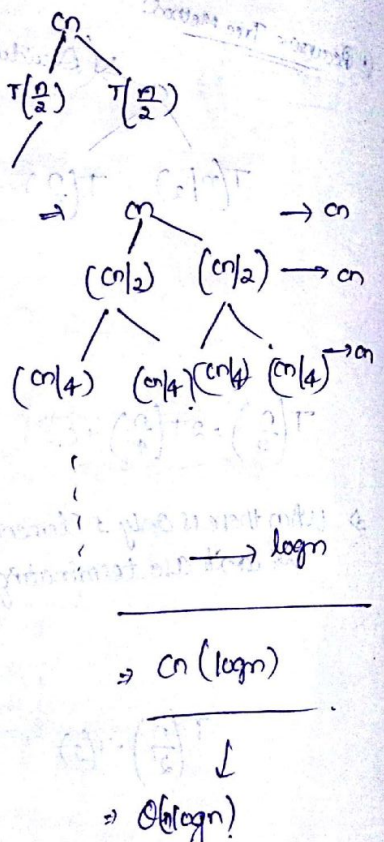
② $T(n) = 2T\left(\frac{n}{2}\right) + O(n)$

③ Master's theorem

→ Quickly we can solve.

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

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



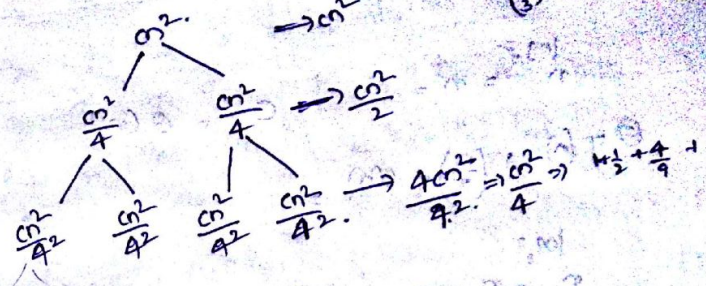
$\Rightarrow cn(\log n)$

$\Rightarrow O(n \log n)$

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

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

$\left(\frac{n}{2}\right)^2$



→ (log n) terms

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

$n = 2^k$

$k = \log_2 n$

$\log n$

$= cn^2\left(1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots\right)$

$= \frac{cn^2}{1 - 1/2} \leq 2cn^2$

$= O(n^2)$

$cn^2 \leq cn^2\left(1 + \frac{1}{2} + \cdots\right) \leq 2cn^2$

$= \Omega(n^2), O(n^2)$

*** $T(n) = aT(n/b) + cn^d$ (a, b, c, d → Anything

a → No. of subproblems

b → Each problem size is n/b

cn^d → for combing & Dividing

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

$n^{\log_b a} < n^d \rightarrow T(n) = O(n^d)$

$n^{\log_b a} > n^d \rightarrow T(n) = O(n^{\log_b a})$

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

$n^{\log_2 2} = n$ — c — $= O(n)$

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

$n^{\log_2 2} = n \quad n = O(n \log n)$

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

$n < n^2 \Rightarrow O(n^2)$

Polynomial Asymptotically Equal

Polynomially:

n, $n\log n$ → can't be compared
- Because it is logarithmic term
- Master's Theorem isn't Applicable

Variant of Master's theorem:

$$T(n) = aT(n/b) + cn^d \cdot (\log n)^k$$

① $n^{\log_b a} > cn^d$, $T(n) = O(n^{\log_b a} \cdot \log n)$

② $n^{\log_b a} = cn^d$, $T(n) = O(n^d \cdot \log n \cdot \log n)$

Assignment:

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

A_{11}: $a_{11}\ a_{12}$ / $a_{21}\ a_{22}$; A_{12}: $a_{13}\ a_{14}$ / $a_{23}\ a_{24}$; A_{21}: $a_{31}\ a_{32}$ / $a_{41}\ a_{42}$; A_{22}: $a_{33}\ a_{34}$ / $a_{43}\ a_{44}$

B_{11}: $b_{11}\ b_{12}$ / $b_{21}\ b_{22}$; B_{12}: $b_{13}\ b_{14}$ / $b_{23}\ b_{24}$; B_{21}: $b_{31}\ b_{32}$ / $b_{41}\ b_{42}$; B_{22}: $b_{33}\ b_{34}$ / $b_{43}\ b_{44}$

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

$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{matrix} ae+bg & af+bh \\ ce+dg & cf+dh \end{matrix}$

mul(A, B, n) ⇒ 8 subproblems

$C_{11} = (A_{11} * B_{11}) + (A_{12} * B_{21})$ — AC

$(A_{11} * B_{11})$ → Subpblm 1; $(A_{12} * B_{21})$ → Subpblm 2

⇒ (Dividing & Simply)

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

(Subproblems) (combining)

If Explicit

(Termination Condition, when $n = 1$)

(Some Constant time)

$n^{\log_2 8} = n^3$

$n^3 > n^2 = O(n^3)$

$P = (A_{11}+A_{22})(B_{11}+B_{22})$ → 1 subproblem

$Q =$

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

$S =$

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

$U =$

$V =$

$C_{12} = R+T = A_{11}B_{12} - A_{11}B_{22} + A_{11}B_{22} + A_{12}B_{22}$

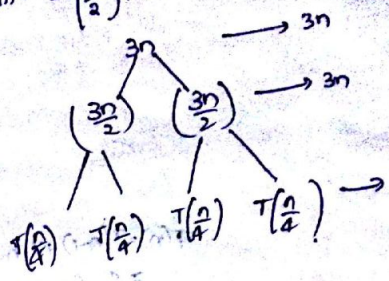
$= A_{11}B_{12} + A_{12}B_{22}$

$= O(n^{2.781})$

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

Tut-3

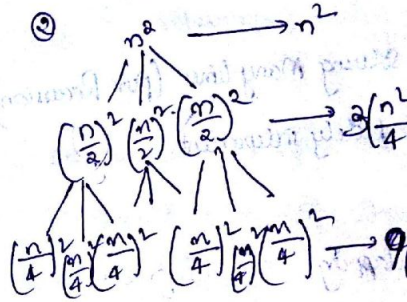
① $T(n) = 2T\left(\frac{n}{2}\right) + 3n$



$\rightarrow \frac{3n}{4} \cdot 4 = 3n$

$3n(\log n) \Rightarrow O(n \log n)$

② $O(n^2)$



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

$\rightarrow 9\left(\frac{n}{4}\right)^2 = \frac{n^2}{4^2}(9)$

$\rightarrow \left(\frac{3}{4}\right)^i n^2$

$\log n \left(1 + \frac{1}{2} + \frac{1}{4} + \cdots\right)$

At i^{th} level $\rightarrow \left(\frac{3}{4}\right)^i n^2$

$= \left(\frac{3}{4}\right)^{\log_4 n} n^2$

$= n^2 + \frac{3}{4}n^2 + \left(\frac{3}{4}\right)^2 n^2 + \cdots + O\left(n^{\log_{4/3} 1}\right)$ (Apply G.P series)

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

$\Rightarrow \frac{n^2}{4^i} = 1$

$\Rightarrow \frac{n}{4^i} = 1$

$n = 4^i$

$\log_4 n = i$

$\left(\frac{3}{4}\right)^i = 1$

→ Pure Recursion

→ Memoization

→ Dynamic programming

Pure Recursion

```
int fib(int n)
{
  if(n==0) return 0;
  if(n==1) return 1;
  return fib(n-1)+fib(n-2);
}
```

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

$= 1$ $(n=0 \text{ or } 1)$

(Masters Theorem can't be Applied)

$(1.61)^n$ (Exponential time)

→ Same pblm solving Many times (Pure Recursion)

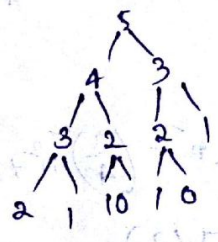
(Divide & Conquer) left & Right → Completely Independent

→ When subpblms are Overlapping

```
int fib(int n)
{
  f[i]=-1;
  if(f[i]==-1)
    return f[i]=f(i-1)+f[i-2];
```

Array f[P] → Initialize the table with -1

If f[i]=-1 Compute the value & Store it in f[i]




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

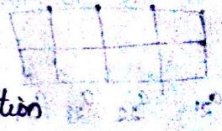
Memoization

O(n)

→ Both are based on top to bottom Recursion

→ One Requires table, other Requires table

→ Running time decreased



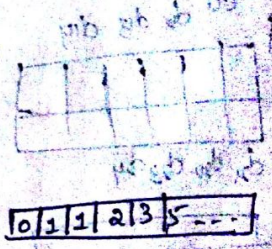
Memoization

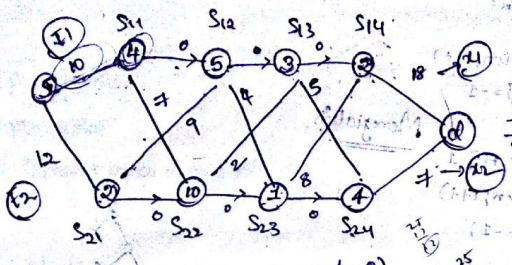
Dynamic Programming

→ No Recursion only loops,

→ 1 Table

```
for(i=2;i<n;i++)
{
  F[i]=F[i-1]+F[i-2];
}
```





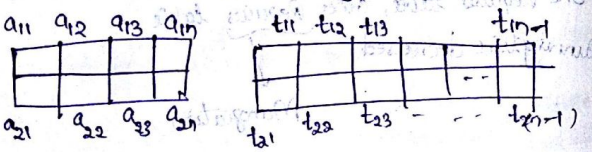
(on the same line, then cost = 0)

(2^n ways)

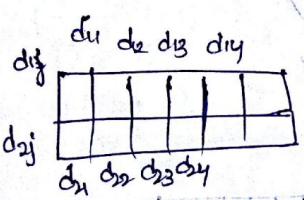
$f_n = f_{n-1} + f_{n-2}$

↳ Recursive formulation

S_{ij} → column
↳ Row



d_{13} = length of shortest path from source to S_{13}



d → length of the shortest path from source to Destination

$$d = \min \begin{cases} d_{1n} + x_1 \\ d_{2n} + x_2 \end{cases}$$

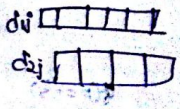
$d_{11} = e_1$, $d_{21} = e_2$ } Base conditions

$2 \le j \le n$

$$d_{ij} = \min \begin{cases} d_{1,j-1} + 0 + a_{ij} \\ d_{2,j-1} + t_{2,j-1} + a_{ij} \end{cases}$$

$$d_{2j} = \min \begin{cases} d_{2j-1} + a_{2j} \\ d_{1j-1} + t_{1,j-1} + a_{2j} \end{cases}$$

$d_{11} = 10 + 4$
$d_{21} = 12 + 2$
$d_{12} = \min \{ d_{11} + a_{12}, d_{21} + t_{21} + a_{12} \}$
$= \min \{ 10 + 5, 12 + 9 + 5 \}$
$d_{12} = \{ 15, 26 \} = 15$



Version 1: (Pure Recursion)

↳ d_{ij}, d_{2j}

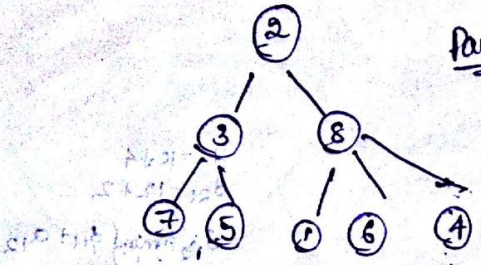
↳ Mathematical notion, not an array

Memorization:

d_{ij}, d_{2j} → Arrays

Dynamic:

```
for(j=2; j<=n; j++)
d[1][j] = min { d[1][j-1] + a[1][j]
               d[2][j-1] + t[2][j-1] + a[1][j] }
d[2][j] = min { d[2][j-1] + a[2][j]
               d[1][j-1] + t[1][j-1] + a[2][j] }
```



Parent Array

1	2	3	4	5	6	7	8
8	-1	2	8	3	8	3	2

DS:

(a, b)

↳ always Root

⇒ shortest path (a, b') + b

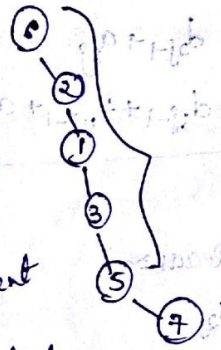
b' = b's parent

⇒ shortest path

```
if(r == s)
{
}
else
{
  shortest path(r, P[v]);
}
```

Parent

leaf



⇒ Recursively calculate the shortest path from Root & Parent & Explicitly add leaf value.

Assignment:

↳ Retrieve the whole path in assembly line.

Towers of Hanoi:

(Recursive substitution)

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

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

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

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

Terminate it when $k = n$

$= 2^n T(0)$

$= O(2^n)$

⇒ $T(n) = 2T(n-2)$

$= 2(2T(n-4))$

$= 2^2 T(n-4)$

$= 2^2(2(T(n-6)))$

$= 2^3 T(n-6)$

$= 2^4 T(n-8)$

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

$K = n/2$

$= 2^{n/2} T(0)$

$= (\sqrt{2})^n \Rightarrow O(1.414^n)$

$T(n) = T(n-1) + T(n-2)$ — Characteristic Eqn method

c^n

$x^n = x^{n-1} + x^{n-2}$

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

$x^2 - x - 1 = 0$

$x = \frac{1 \pm \sqrt{1+4}}{2} = \frac{1 \pm \sqrt{5}}{2}$ $\quad = (1.6)^n$

$= O(1.6)^n$

Given
⇒ Array - find minimum Element. (Dynamic programming)

find K^{th} Min Element, if K is not fixed.

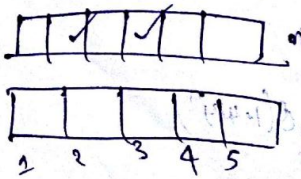
$O(Kn)$ = K in terms of n

$= O(n^2)$. → But we can obtain it

$O(n)$.

⇒ two strings are given,

LCS (longest Common Subsequence).



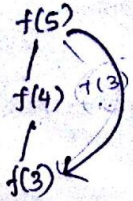
$1 \le i \le m$

$1 \le j \le n$

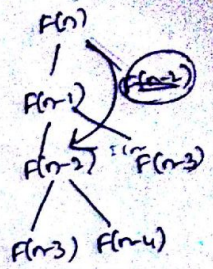
→ Both are moving forward simultaneously

$O(m+n)$

Dag :-



(Directed Acyclic Graph)



C_{ij} = length of longest Common Subsequence

C_{mn} =

$x^3 - x^2 - x + 1 > 0$

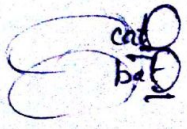
1 -1 -1 -1

-1 -1 +1 -1

$-i + 1 - i - 1$

from Backwards

AGCAT
GAC



If last two characters are matching then,

$= c_{i-1,j-1} + 1$ (If last character is Matching)

if (X[i] == Y[j])

Termination

① Empty string → if (i==0 || j==0)
return 0;

$= \max \begin{cases} c_{i-1,j} \\ c_{i,j-1} \end{cases}$ → If ma