

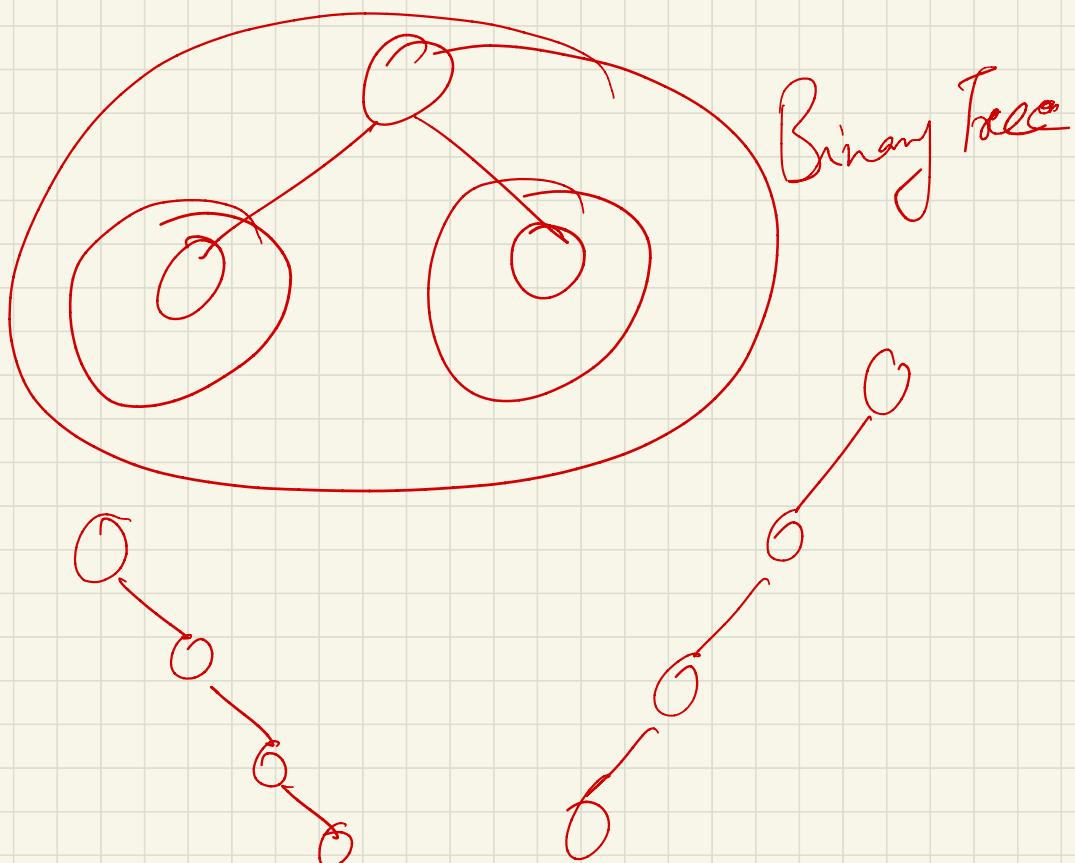


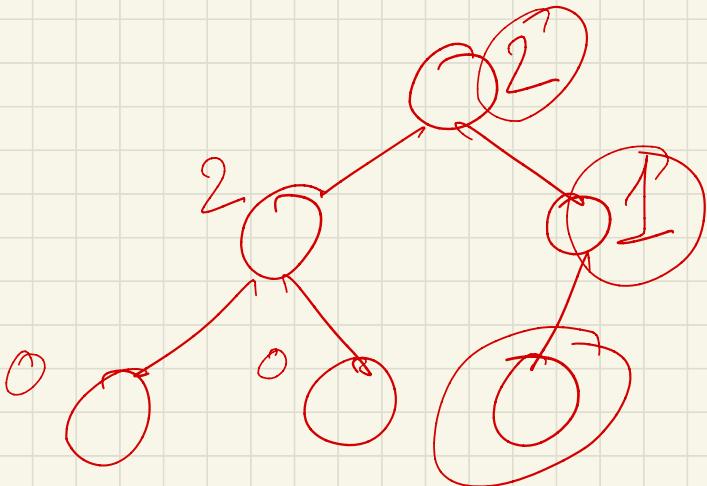
DS & ALGO - 12 Jul 20.

# Trees

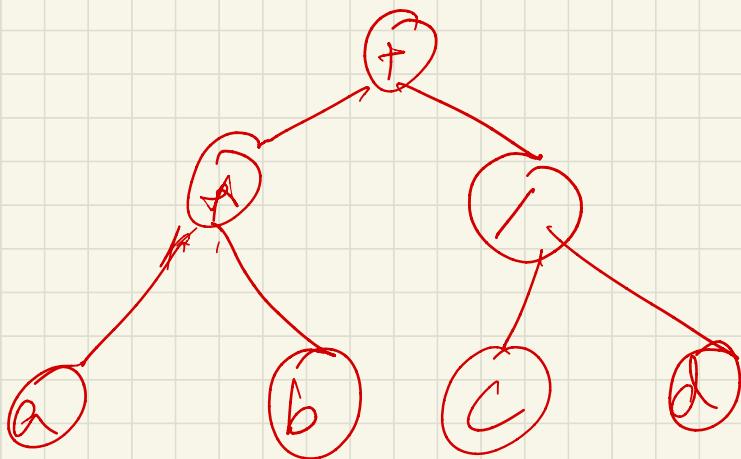
○ Root -

Is it a binary tree? -  
- Yes -





$0, 1, 2$   $\rightarrow$  children ..



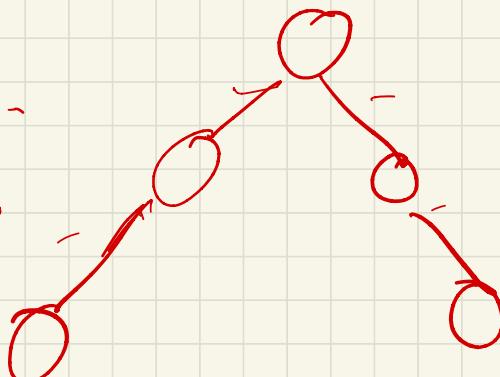
$(a * b) + (c / d)$

# Properties of Binary Tree -

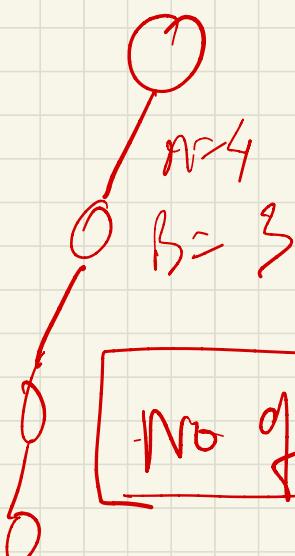
1.

$$n=5$$

$$B=4$$

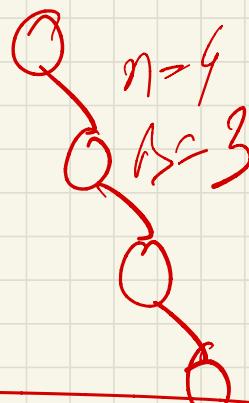


$$\begin{matrix} n=1 \\ B=0 \end{matrix}$$



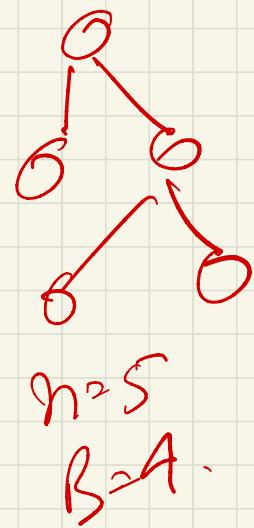
$$n=4$$

$$B=3$$



$$n=4$$

$$B=3$$



$$n=5$$

$$B=4$$

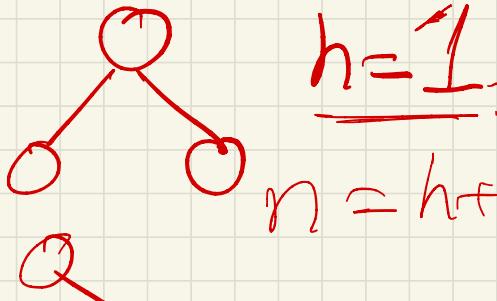
$$\boxed{\text{No of Branches} = n-1}$$

Q. height = 2.,  $h=0$   $\underline{K}$

a) Minimum no. of elements.

$$n = h+1.$$

$$\begin{cases} h=0 \\ n=1 \end{cases}$$



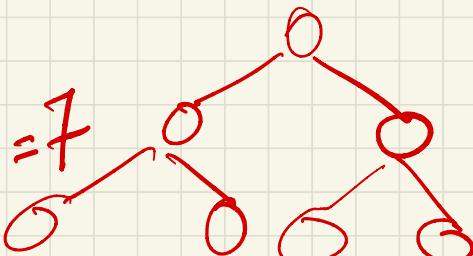
$$\underline{h=1}$$

$$n = h+1 = 2.$$

b) Maximum no. of elements.

For height  $h$ , max  $= \boxed{2^{h+1} - 1}$ .

$$\begin{aligned} h &= 2 \\ n &= 2^3 - 1 = 7 \end{aligned}$$



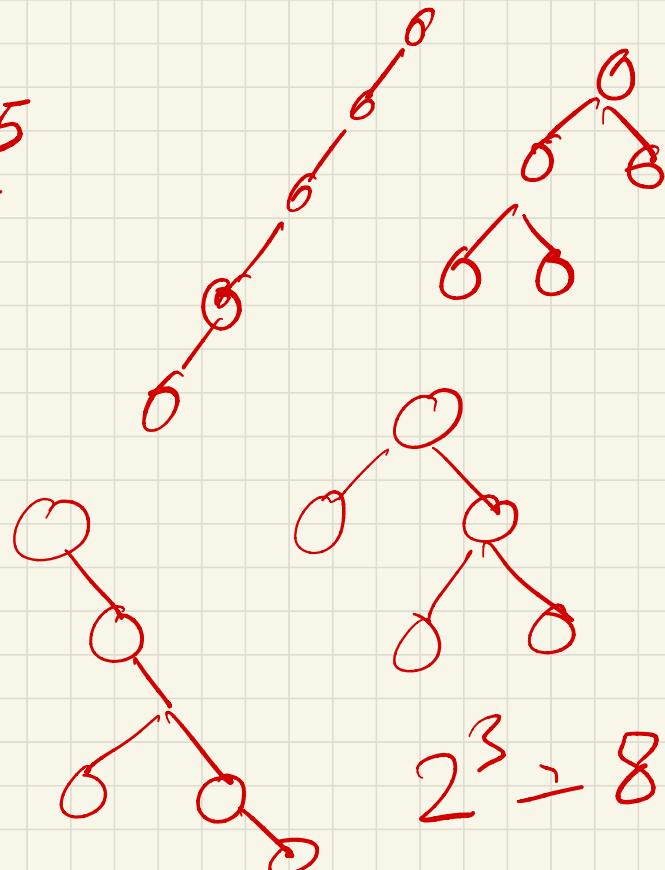
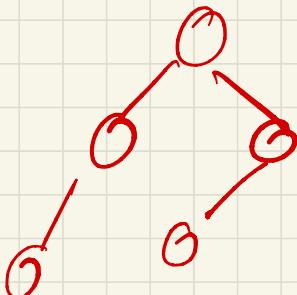
$$\begin{aligned} h &= 1 \\ m &= 2^{l+1} - 1 \\ &= 3 \end{aligned}$$

3) no. of elements,  $n$ .

$$h_{\min} = \lceil \log_2(n+1) \rceil - 1.$$

$$h_{\max} = n-1.$$

$$\therefore n=5$$

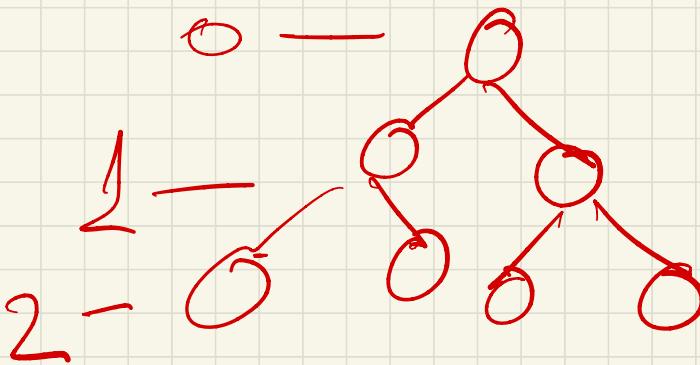


$$2^3 = 8.$$

$$\log_2 8 = \underline{\underline{3}}$$

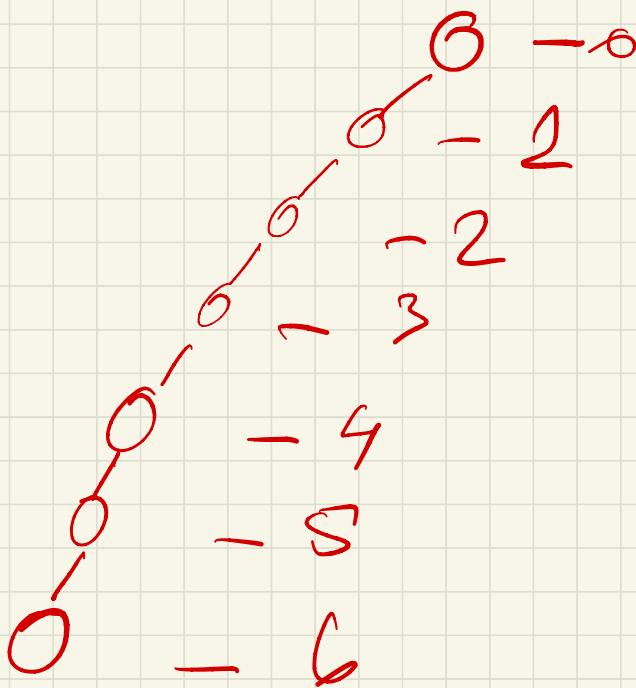
$$\underline{n=7}.$$

$$h_{\min} = \frac{\log 8 - 1}{2}.$$

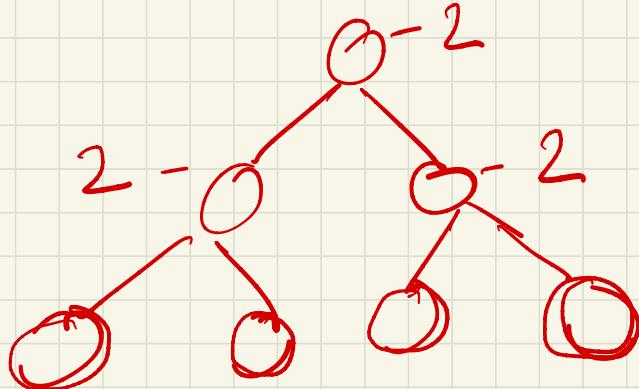


$$\underline{h=2}.$$

$$h_{\max} = 7 - 1 = 6.$$

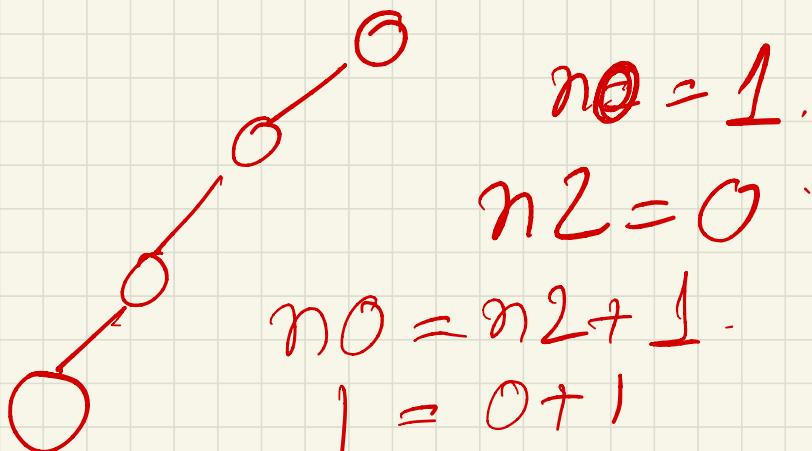


4)  
≡

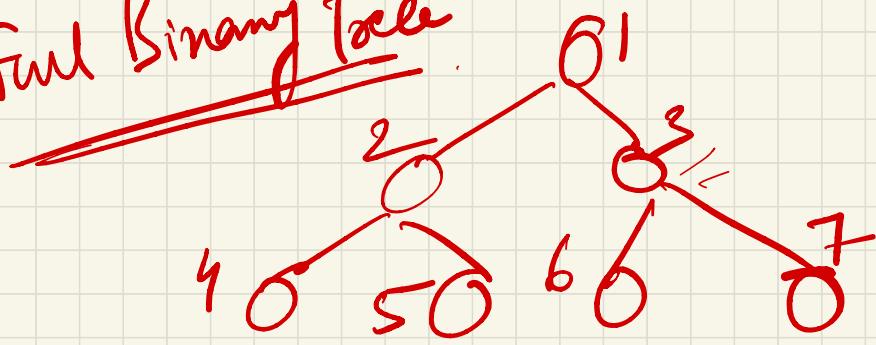


$$n_0 = 4. \text{ no node degree } (2)$$

$$\begin{aligned} n_0 &= n_2 + 1 \\ &= 3 + 1. \end{aligned}$$



full Binary Tree



node ( $i$ )  $\Rightarrow$  node 1  $\rightarrow$  Parent?

$$\lfloor i/2 \rfloor = 2.$$

$$\text{node}(5) = \lfloor 5/2 \rfloor = 2.$$

$$\text{node}(1) = \lfloor 1/2 \rfloor = \begin{array}{c} 0 \\ \hline \text{Root} \end{array}$$

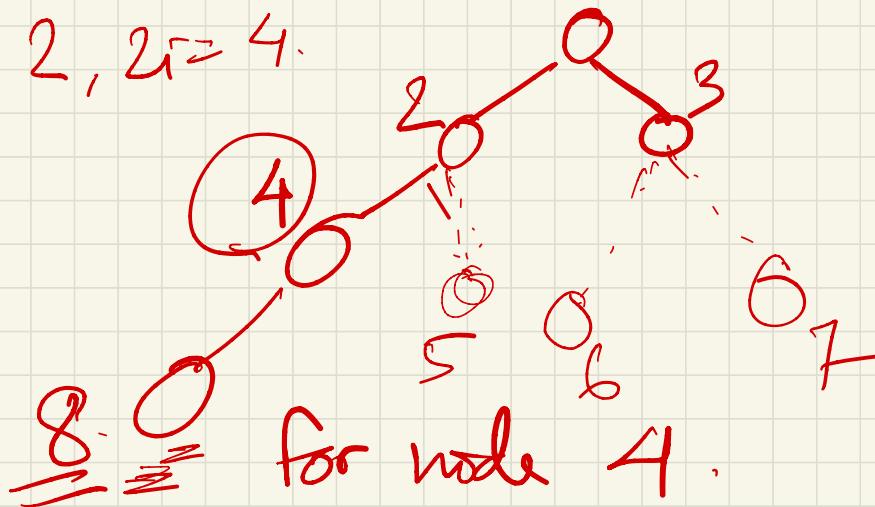
Left Child  
 $(2i)$

left child =  $2i$

o node 1,  $2i = 2^1$

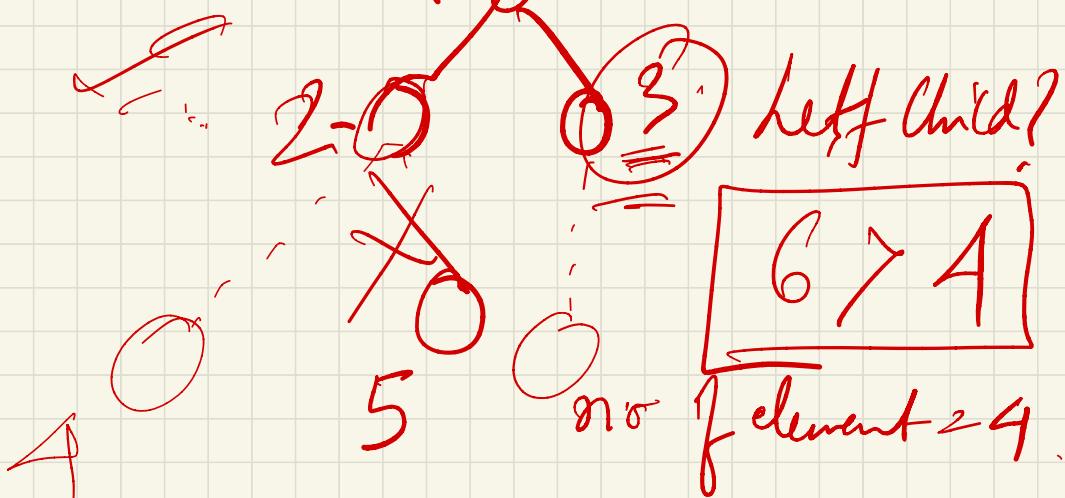
for node 2,  $2i = 4$ .

1



for node 4.

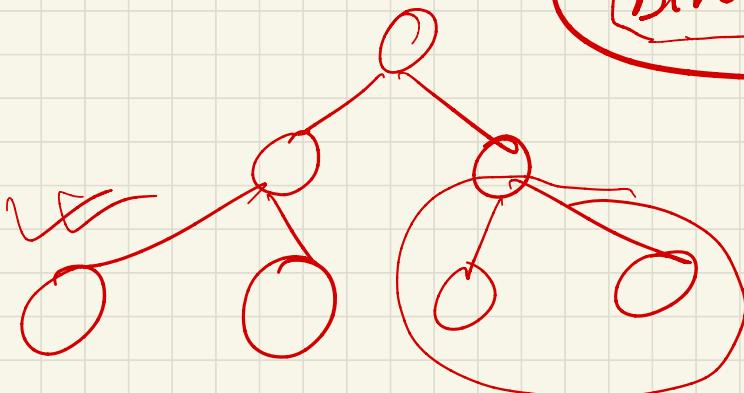
left child = 8



$2i > n$ , node  $i$  has no left child.

Full Binary Tree,

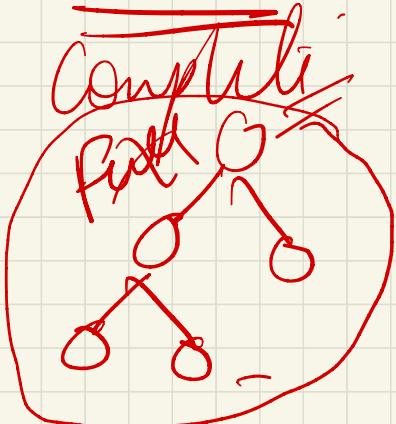
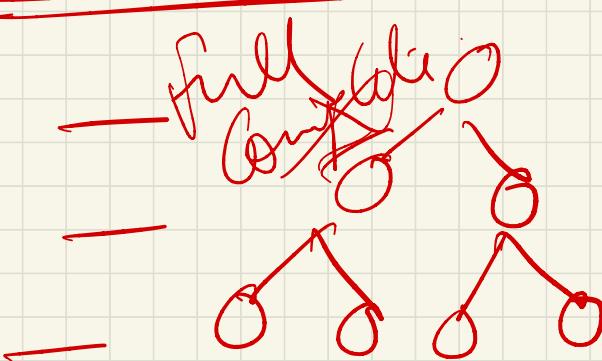
Complete  
Binary Tree



Right child

$\rightarrow 2i + 1$ .

full  
Complete



Data Structure with C. Lipschitz

Representation

- Array  $\alpha$
- LL. - LL.

---

---

## TREE TRAVERSAL

---

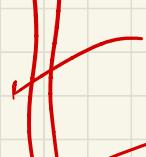


19 Jul 20 . Trees .

// Traversal -  
— Pre Order  
— Inorder  
— Post order.

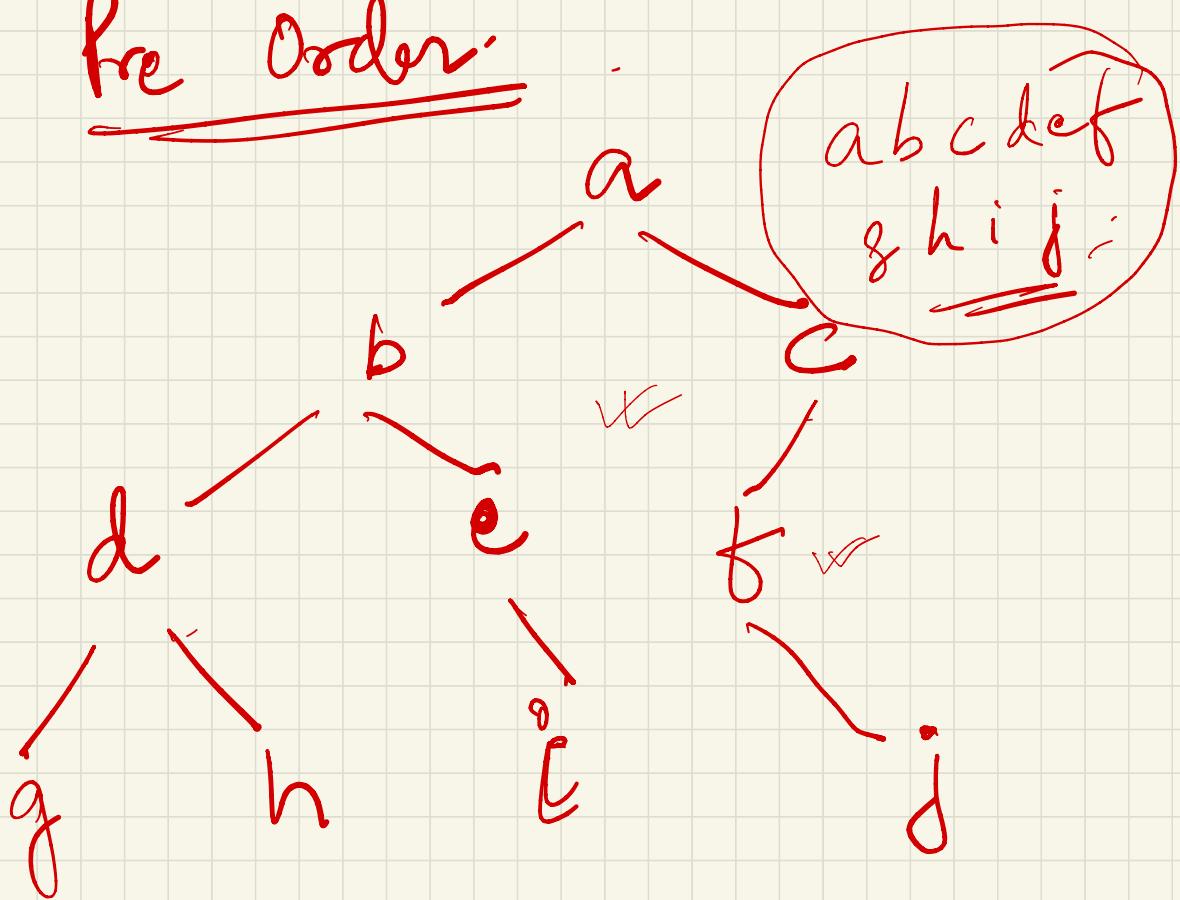
Order is always in respect of root .

Pre- root , left, right . 

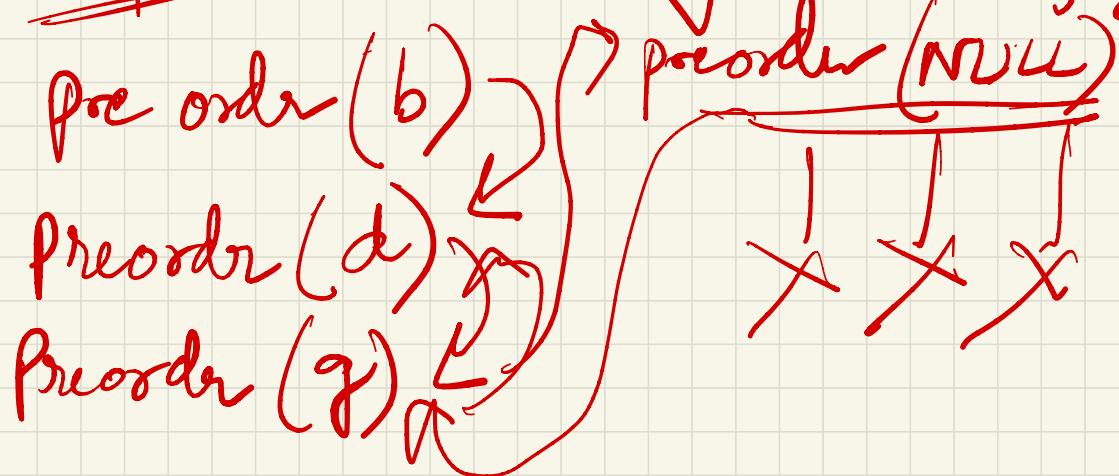
In- left, root, right . 

Post - left, right, root . 

## Pre Order



Output -  $\rightarrow a b d g h e i c f j$



$\hookrightarrow$  preorder(z)  $\rightarrow$  preorder(NRLL)  
↓

Preorder(d) ← XXX  
↓

Preorder(4n) → Preorder(NRLL)  
↓

Preorder(NRLL) ← XXX

XXX → Preorder(b)  
↓

Preorder(wr) ← Preorder(e)

XXX → Preorder(i)

(NULL) - x x x

(NULL) - x x x x.

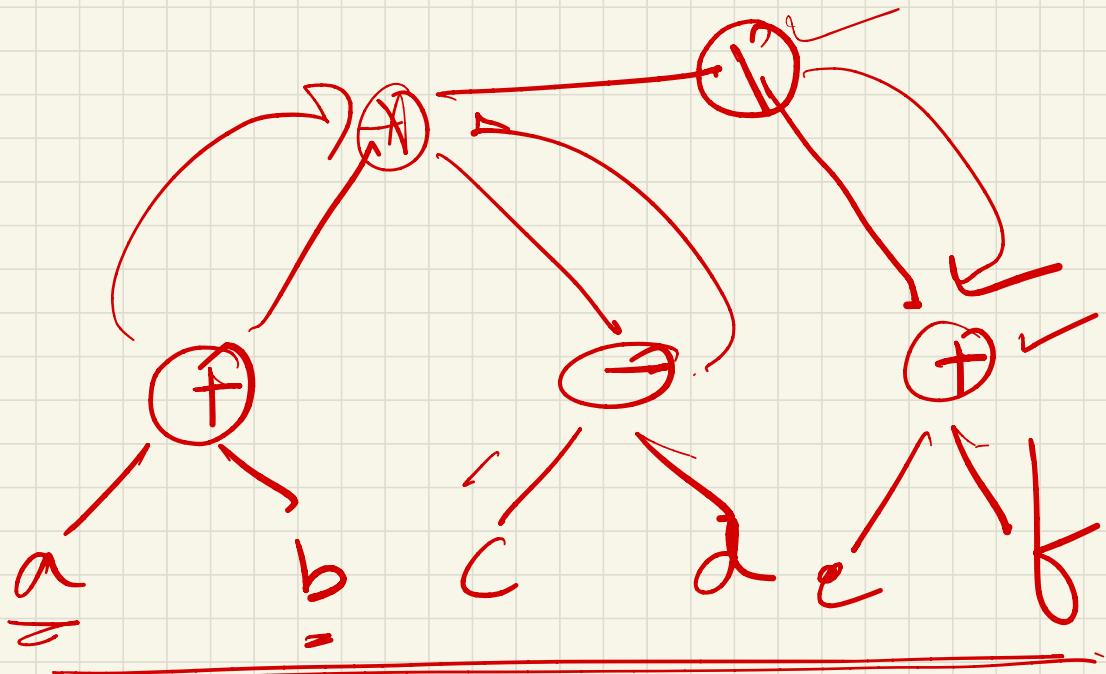
preorder(i)  $\rightarrow$  preorder(c)  $\rightarrow$  (b)

preorder(c)  $\leftarrow$  (a)  $\leftarrow$

↓  
preorder(f)  $\rightarrow$  NULL  $\rightarrow$  x x x

preorder(j)  $\leftarrow$  (d)  $\leftarrow$

Expression  $\frac{(a+b)*(c-d)}{(e+f)}$

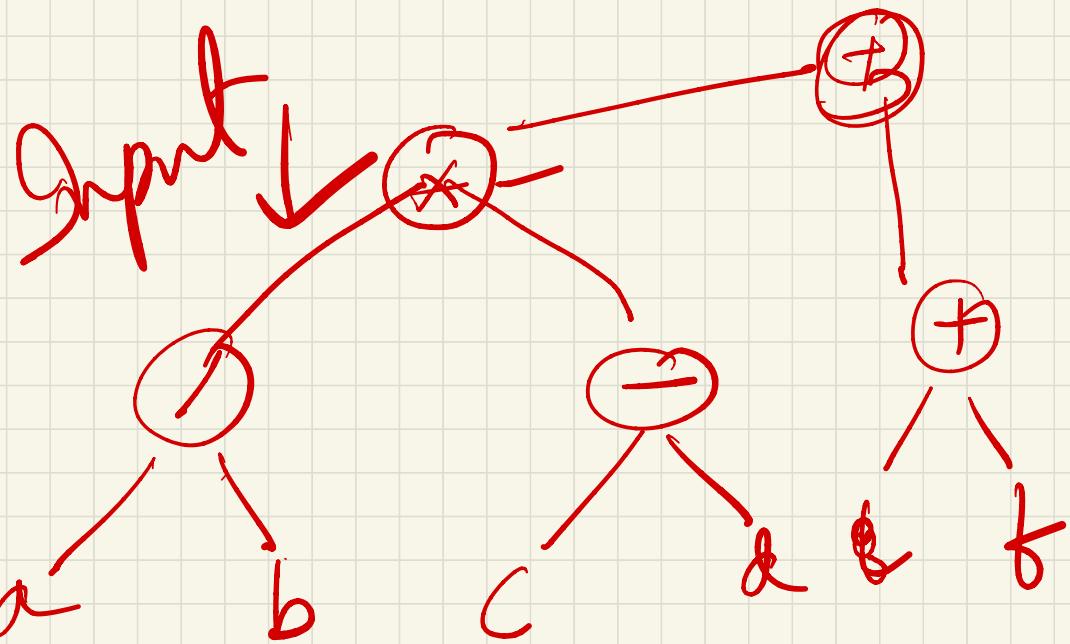


$/ * + a b - c d + e f$

Prefix form: ~~X~~

~~$/ * + + - a b c d e f X$~~

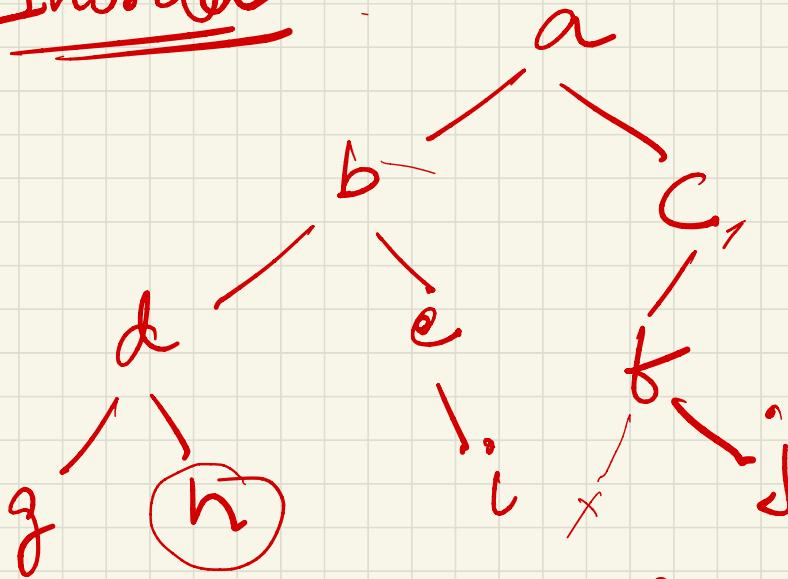
$$(((\underline{\underline{a/b}}) * (\underline{\underline{c-d}})) + (\underline{\underline{e+f}}))$$



$$\begin{array}{r}
 + * / a b - c d + e f \\
 \hline
 \end{array}
 \quad \xrightarrow{\quad} \text{prefix} \cdot$$

Expression Tree  $\rightarrow$  prefix (with help of preorder)

## Inorder:



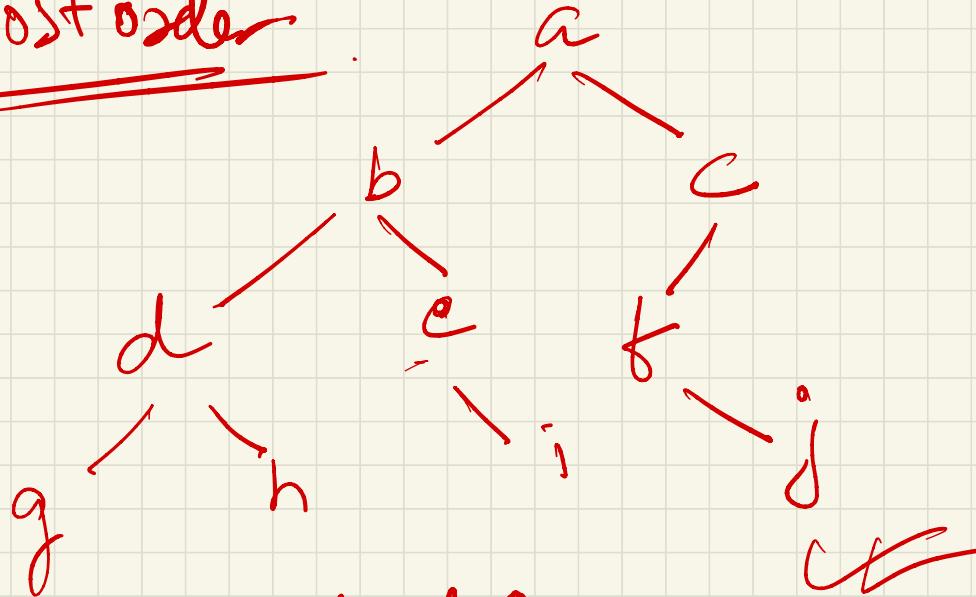
Result: - g d h b e i a f j c

In(a) → In(b) → In(d) → In(g)

as LLL → print(g) ← XXX ← NLL

↓  
XXX → In(d) → In(h)

## Postorder



Output :- ghdiecebjtca

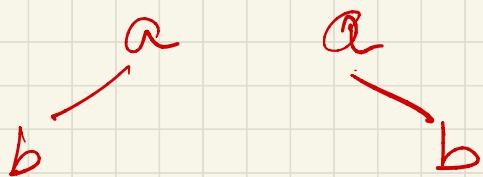
Result:-  $a \rightarrow b \rightarrow d \rightarrow g$

$b \leftarrow d \leftarrow h \leftarrow de \searrow$   
 $\nwarrow e \rightarrow i \rightarrow e \rightarrow b \rightarrow a$

$a$   
 $\uparrow c \leftarrow f \leftarrow j \leftarrow f \leftarrow c$

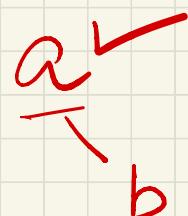
# Binary Tree Construction

Preorder - ab



Inorder - ? ab ?

↳ ab



? @ ab



ab ?

b

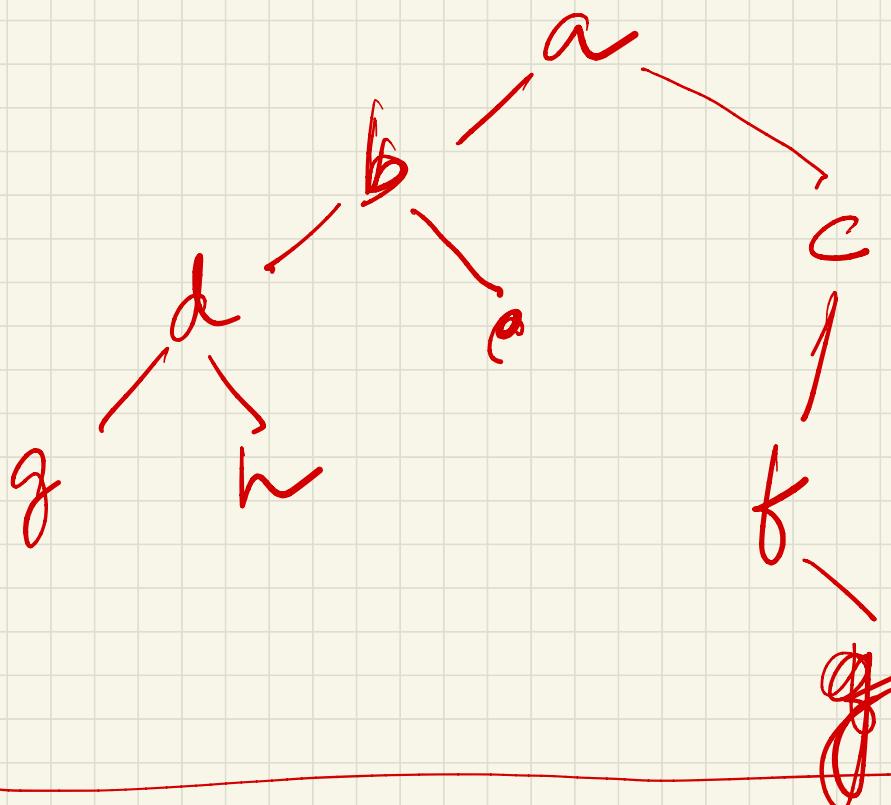
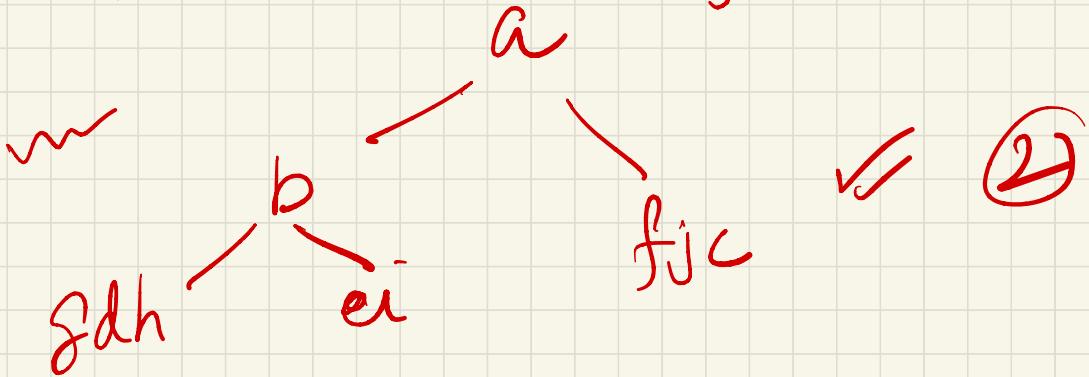
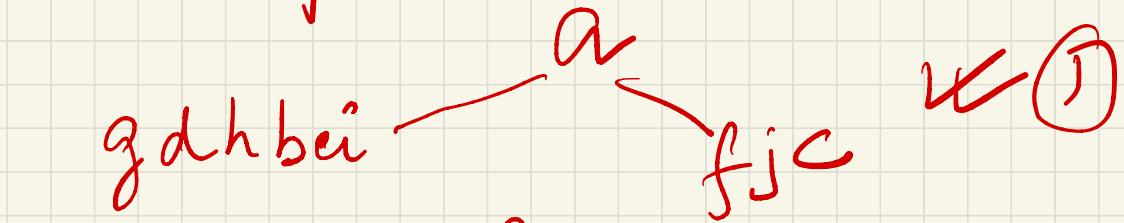
Inorder = g d h b e i @ f j c

Preorder = a b & g h & i f j

→ As per Preorder → Root = a.

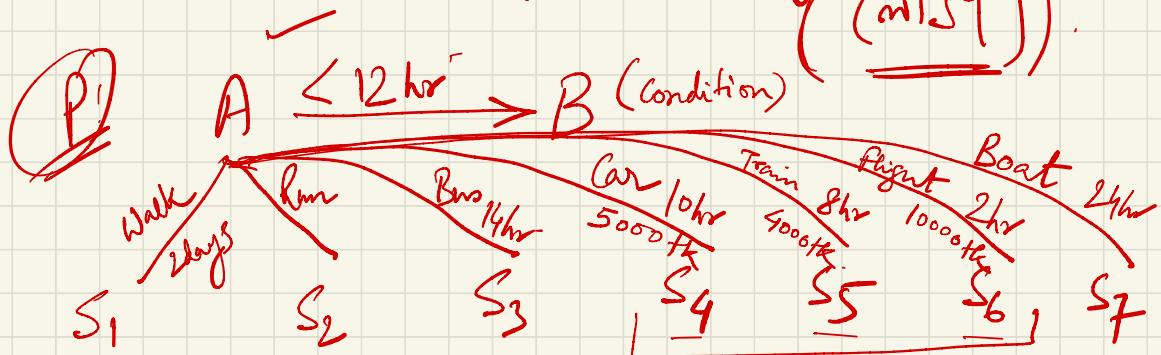
→ Scan the Inorder; and divide the tree as per root found above.

Inorder-left, Root, right



13 Aug 20

## Greedy Method · (Algorithm technique) (Contd.)



Out of 7 Solutions,  $S_4, S_5$  and  $S_6$  meet the conditions.

$S_4, S_5, S_6 \rightarrow$  Feasible Solution ✓

Conditions. 1) 12 hr. 2) Minimum cost.

Out of  $S_4, S_5, S_6, S_7$ ,  $S_5$  has minimum cost.

$S_5 \rightarrow$  Optimal Solution ✓

We can have multiple feasible solutions, but Only One optimal solution ✓

So, these type of problems are called optimization problems.

Optimization problem (Either Minim or Maxim).

- 1) Greedy Method. ✓ ↗ Kruskal, Prim, Dijkstra, Bellman
- 2) Dynamic Programming. ✓ ↗ MCM, LCS, 0/1 knap.
- 3) Branch & Bound. ↗

Algorithm Greedy ( $a, n$ )

{ for  $i=1$  to  $n$  do

$a = \text{Select}(a)$

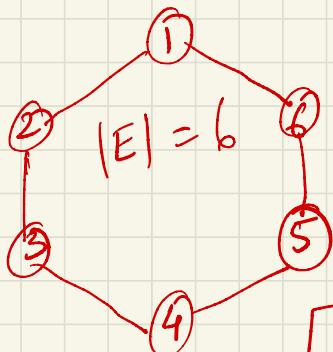
    if Feasible ( $a$ ) then  
        Solution = Solution +  $a$

}

Solution Set =  $\{\underline{a_3, a_4, a_5}\}$ .

Optimal Solution =  $a_3$  (Train). Optimal/Best.

### Minimum Spanning Tree



$$G = (V, E)$$

$$V = \{1, 2, 3, 4, 5, 6\}$$

$$E = \{(1,2), (2,3), (3,4) \dots\}$$

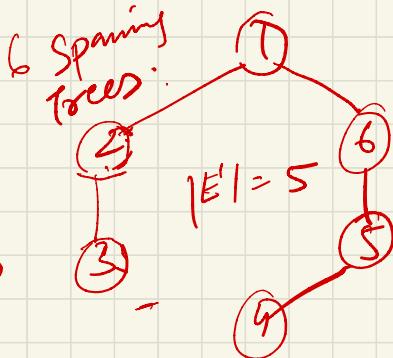
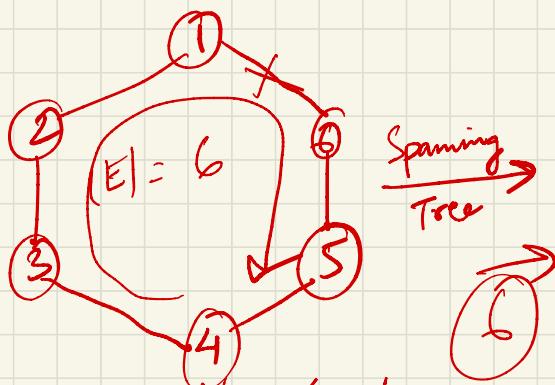
$S \subseteq G$  ( $S \rightarrow \text{Spanning Tree}$ ).

$S$  is subset of  $G$ .

$$S = \{V', E'\}$$

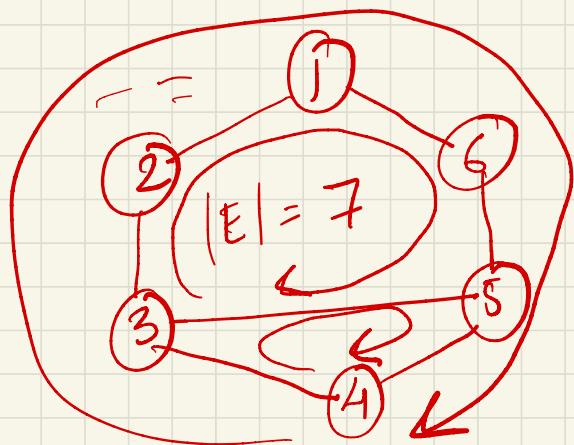
$$\underline{V' = V}, \quad |E'| = |V| - 1$$

Spans across  
all vertices -



From a graph  $G$ , how many possible spanning trees can be formed ??  $\geq 6$

$$\text{Combination} \rightarrow {}^6 C_5 = \frac{n!}{r!(n-r)!} = \frac{6!}{5! \times 1!} = \frac{6 \times 5!}{5!} = 6$$



$$|E'| = {}^7 C_5 = \frac{7!}{5! \times 2!} = 21$$

Q) How many cycles with less than total no. of vertices ?

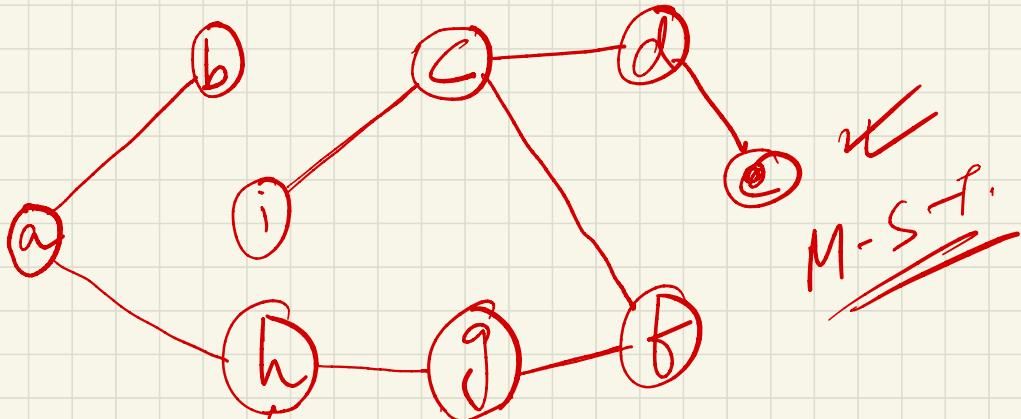


$$\text{No. of Spanning Trees} = |E|^{V-1} - \text{no. of cycles with } \leq V \text{ vertices}$$

✓  $= |E|^{V-1} - \text{no. of cycles with } \leq V \text{ vertices}$ 
Imp.

19 Aug 20 Sessional - Prims.  
- Kruskal.

Kruskal from Cormen book -



$A = \text{list of final edges. initially } A = \emptyset$ .  
 if,  $\text{findset}(u) \neq \text{findset}(v)$ .  
 $A = A \cup \{u, v\}$ .  $A = \{(h, g), \{i, c\}\}$

Next edge  $(i, c)$  -  $\text{find}(i) = ?$  ✓  
 $\text{find}(c) = c$

Next edge  $= (g, f)$  -  $\text{Find}(g) = \{h, \underline{g}\}$   
 $\text{Find}(f) = \underline{f}$

Next  $(c, f) \rightarrow \text{Find}(c) = \{c, \underline{i}\}$ .  
 $\text{Find}(f) = h, g, \underline{f}$ .

Next  $(i, g) = \text{Find}(i) = \underline{i, c, f, g, h}$   
 $\text{Find}(g) = \underline{i, c, f, g, h}$ .

is  $\text{find}(i) = \text{find}(g)$ ?.

if Yes, then don't include that edge.

---

→ ~~Dijkstra~~  
→ ~~Bellman Ford~~ } Lee ·  
Swarnil ·

---

X X — X X M

31/08/20 Dynamic Programming - MCM & LCS

→ Greedy Methods vs DP.

→ Kruskal - Select the shortest edge & proceed.

// Before starting, we decided that we will select the shortest / minimum value edge & then proceed -

// Dynamic Programming → Principle of optimality.

At every stage, we will decide the optimal solution.

// Dynamic Programming generally deals with recursive formulae.

// Example. → To find the  $n^{\text{th}}$  fibonacci term.

$$\text{fib}(n) = \begin{cases} 0 & \text{if } n=0 \\ 1 & \text{if } n=1 \\ \underline{\text{fib}(n-2) + \text{fib}(n-1)} & \text{if } n>1 \end{cases}$$

Fib → 0, 1, 1, 2, 3, 5, 8, 13. . . .

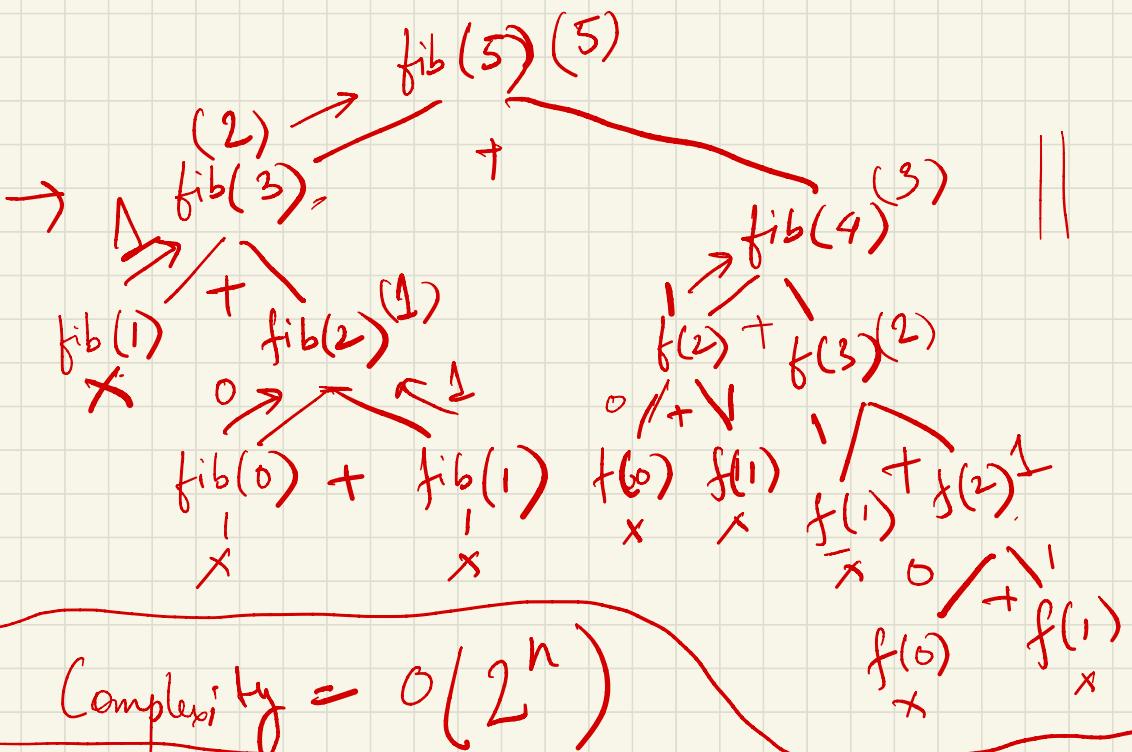
$$0 \overset{f}{\uparrow} 1 \overset{f}{\uparrow} 1 \overset{f}{\uparrow} 2 \overset{f}{\uparrow} 3 \overset{f}{\uparrow} 5 \overset{f}{\uparrow} \text{fib}(5) = 5$$

```

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

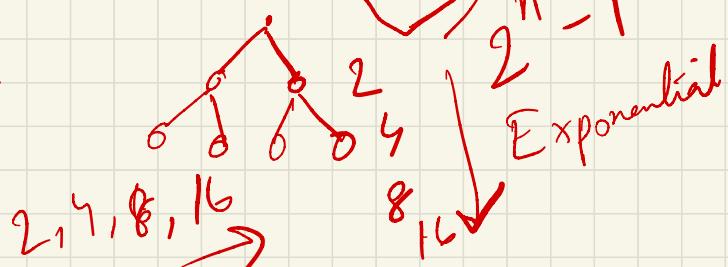
```

Find, the 5<sup>th</sup> term in Fibonacci Series

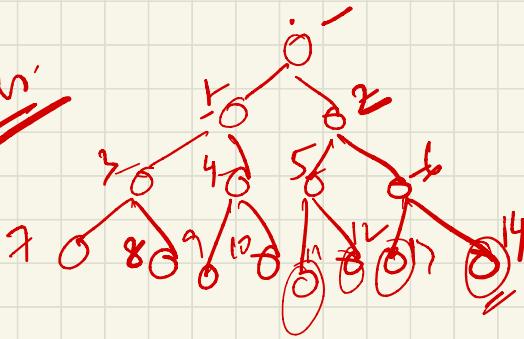


level fib(5) → 2

1 level 2 → 4  
lrl 3 → 6



For call:



$\lceil \log n \rceil$  // 14

$$\begin{aligned}
 & f(5) = f(3) + f(4) \\
 & f(3) = f(1) + f(2) \\
 & f(1) = f(0) \\
 & f(4) = f(2) + f(3) \\
 & f(2) = f(1) + f(0) \\
 & f(3) = f(1) + f(2) \\
 & f(4) = f(2) + f(3) \\
 & f(5) = f(3) + f(4)
 \end{aligned}$$

Exponential ↓

$$f(0) = 3, f(1) = 5, f(2) = 3.$$

Can the complexity be reduced ? .

// What if we take a global array .

$a[5]$	-1	-1	-1	1	-1	-1
	0	1	2	3	4	5

$$f(5) - f(3) - f(1) \rightarrow 1$$

$\diagdown f(4) - \diagup f(2)$

0	1	1	2	3	5
0	1	2	3	4	5

global array

$(n+1)$  - calls.

So, from  $2^n$  calls, we reduced to  $(n+1)$  calls

$2^n \rightarrow n$ , Exponential  $\rightarrow$  linear.

This above Concept is known as Memoization

↗ Memoization (Memoize)

Dynamic Programming

Memoization  
(Recursive)

Tabulation  
(Iterative)



# Tabulation Method / Iterative Method

```
int fib (int n)
{
```

```
    if (n <= 1)
```

```
        return n;
```

```
    F[0] = 0'; F[1] = 1';
```

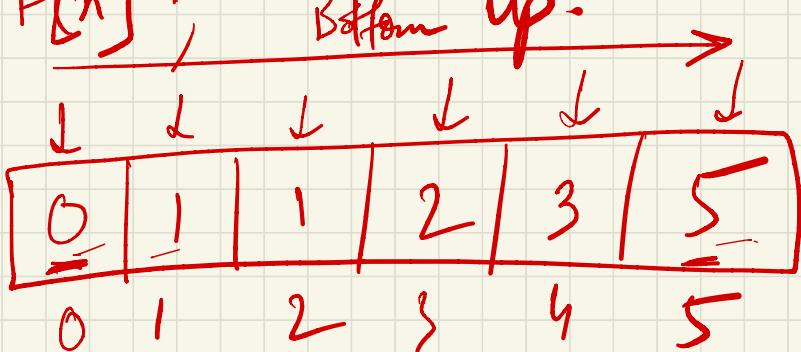
```
    for (int i = 2'; i <= n; i++)
    {
```

```
        F[i] = F[i-2] + F[i-1]
```

```
}
```

```
    return F[n];
```

```
}
```



## Memoization

1) Recursive

2) Top down Approach

We Started fib(5)

vs

## Tabulation

i) Iterative

2) Bottom up Approach

We Started with F[0]

# MCM: Matrix Chain Multiplication

$A_1 \times A_2 = \text{Conditions} \quad ??$   
 $r_1 \times c_1 \quad r_2 \times c_2 \quad (c_1 = r_2) \quad \text{!}$   
 $5 \times 4 \quad 4 \times 3 \quad \text{product} = (r_1 \times c_2)$   
 $\text{final} \quad \text{---}$

$$\begin{array}{ccc}
 \overbrace{A \times B}^{\substack{(5 \times 4) \quad (4 \times 3)}} & = & C \\
 \left[ \begin{array}{cccc|ccc} \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \end{array} \right] & \times & \left[ \begin{array}{ccc|cc} \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \end{array} \right] & = & \left[ \begin{array}{ccc|cc} \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \end{array} \right] \\
 & & & & \substack{4 \text{ multiplications} \\ 1 \quad 1 \quad 1 \quad 1} \\
 & & & & \dots \quad \dots \quad \dots \quad \dots
 \end{array}$$

$$\begin{aligned}
 C &= 5 \times 3, \text{ Total no. of multiplications} \\
 &= 5 \times 3 \times 4
 \end{aligned}$$

$$\begin{array}{ccc}
 \overbrace{A \times B}^{\substack{10 \times 10 \quad 10 \times 5}} & = & C \\
 & & \left\{ \begin{array}{l} \text{Total Multiplication} \\ = 10 \times 10 \times 5 \end{array} \right.
 \end{array}$$

What if, we need to multiply more than two matrices.

MCM, its goal is to find the minimum cost of multiplication and not the result.

$$(A_1 \cdot A_2 \cdot A_3) \cdot A_4 \cdot \\ 5 \times 4 \quad 4 \times 6 \quad 6 \times 2 \quad 2 \times 7$$

$$\boxed{(A_1 \cdot A_2) \cdot (A_3 \cdot A_4)} \dots \textcircled{1} - \\ (5 \times 6) \cdot (6 \times 7) \rightarrow (5 \times 7) -$$

$$\frac{A_1 \cdot (A_2 \cdot A_3) \cdot A_4 \dots \textcircled{2} -}{((A_1 \cdot A_2) \cdot A_3) \cdot A_4 \dots \textcircled{3} -}$$

$$\overbrace{(10 \times 10) \times 5 \times 20} \\ ((10 \times 10) \cdot (5 \times 20) = R_1) \\ 10 \times (10 \times 5) \times 20 \rightarrow R_2 \\ ((10 \times 10) \cdot 5) \times 20 \rightarrow R_3$$

} Equal??  
Yes they  
will be  
equal.

$$\begin{array}{c} A_1 \quad A_2 \quad A_3 \\ 2 \times 2 \quad 2 \times 3 \quad 3 \times 1 \end{array}$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} \text{---} \\ \text{---} \\ \text{---} \end{bmatrix}$$

$$\begin{array}{ll} \textcircled{1} ((\overbrace{A_1 \cdot A_2}^{\text{---}}) \cdot A_3) & \textcircled{2} (\overbrace{A_1 \cdot (A_2 \cdot A_3)}^{\text{---}}) \\ \downarrow & \downarrow \\ \underbrace{(2 \times 3)}_{\downarrow} \cdot (3 \times 1) & (2 \times 2) \cdot \underbrace{(2 \times 1)}_{\downarrow} \\ \downarrow & \downarrow \\ (2 \times 1) & (2 \times 1) \end{array}$$

$$\begin{bmatrix} 78 \\ 170 \end{bmatrix} \xleftarrow{\text{Equal}} \xrightarrow{\text{---}} \begin{bmatrix} 78 \\ 170 \end{bmatrix}$$

Whether  $\textcircled{1}$  is better than  $\textcircled{2}$  ??  
 It will depend on the minimum no. of multiplications.

$$(A_1 : A_2 = A_3) = \frac{4C_2}{3} = 2$$

①  $((\underline{A_1 \cdot A_2}) \cdot \underline{A_3}) = [(\underline{2 \times 2 \times 3}) + (\underline{2 \times 3 \times 1})]$

$\underline{B} \underline{2 \times 3} \times \underline{3 \times 1} = 12 + 6 = 18 \times$

②  $(A_1 \cdot (A_2 \cdot A_3)) = [(2 \times 2 \times 1) + (2 \times 3 \times 1)]$

$(2 \times 2) \cdot (2 \times 1) = 10 \times$

So, ② is minimum,

MCM using DP, will guide us how to parenthesize the chain of matrices.

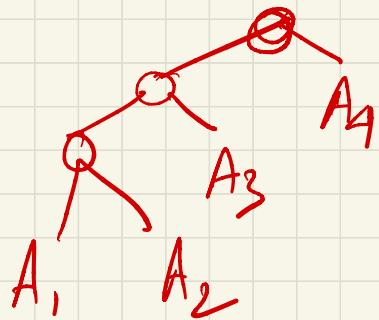
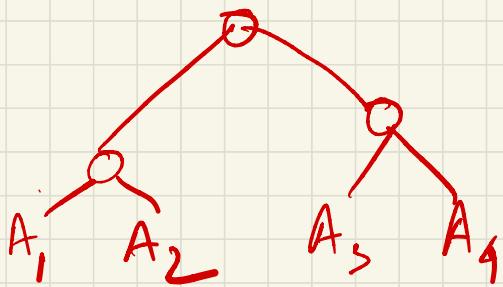
Now,  $[A_1 \cdot A_2 \cdot A_3 \cdots A_7]$ .

How many ways we can parenthesize ???

No. of parenthesis =  $\frac{2n C_n}{n+1}$

$n = \frac{\text{no. of nodes}}{2}$

Tree:



No. of nodes = 3.

$$\text{No. of parenthesis (P)} = \frac{6C_3}{3+1} = \frac{20}{4} = 5$$

MCM Process

$$(A_1 \cdot A_2) : (A_3 \cdot A_4) \\ 5 \times 4 \quad \quad \quad 4 \times 6 \quad \quad \quad 6 \times 2 \quad \quad \quad 2 \times 7$$

Total 5 distinct ways to multiply.

$$A_1 - \underbrace{5 \times 4}_{P_0}, A_2 - \underbrace{4 \times 6}_{P_1}, A_3 = \underbrace{6 \times 2}_{P_2}, A_4 = \underbrace{2 \times 7}_{P_3}$$

$$P_0 = 5, P_1 = 4, P_2 = 6, P_3 = 2, P_4 = 7$$

5x4, 4x6, 6x2, 2x7

General Formula  $A_1 = P_0 \times P_1$ ,  $A_3 = P_2 \times P_3$   
 $A_2 = P_1 \times P_2$ ,  $A_4 = P_3 \times P_4$

$$A_i = P_i - P_{-i}$$

$$(A_1 \cdot (A_2 \cdot A_3) \cdot A_4))$$

$$\frac{5 \times 4}{= 4 \times 6} \quad 6 \times 2 \quad 2 \times 7$$

$$m[1,1] = 0$$

$$A_1 \times A_1 = m(2,2) = 0$$

$$\underline{m[1,2]} = 5 \times 4 \times 6 \\ = 120$$

$$m[2,3] = 4 \times 6 \times 2 = 48$$

$$\underline{m[3,4]} = 6 \times 2 \times 7 = 84$$

$$\underline{\underline{m[1,3]}} = (A_1 \cdot A_2 \cdot A_3)$$

$$\underline{\underline{m[2,4]}} = (A_2 \cdot A_3 \cdot A_4)$$

	1	2	3	4
1	0	120	88	158
2	120	0	48	184
3	88	48	0	84
4	158	184	84	0

	1	2	3	4
1	0	120	88	158
2	120	0	48	184
3	88	48	0	84
4	158	184	84	0

$$A_1 \cdot (A_2 \cdot A_3) \\ (A_1 \cdot A_2) \cdot A_3$$

$$\begin{cases} A_2 \cdot A_3 \cdot A_4 \\ A_2 \cdot A_3 \cdot A_1 \end{cases} \text{ Start min}$$

$$\begin{cases} A_2 \cdot A_3 \cdot A_4 \\ A_2 \cdot A_3 \cdot A_1 \end{cases} \text{ Start min}$$

General formula

$$m[i, j] = \begin{cases} 0 & \text{if } (i=j) \\ \min_{\substack{i \leq k \leq j}} \{ m[i, k] + m[k+1, j] + p_{i-1} \cdot p_k \cdot p_j \} & \text{otherwise} \end{cases}$$

Now,

$m[1, 3] \Rightarrow$  Here k ranges from (1, 2).

For,  $k=1$ .

$$m[1, 1] + m[2, 3] + p_0 \times p_1 \times p_3$$

$$= 0 + 48 + 40 = \underline{\underline{88}} \quad \dots \quad (1)$$

for  $k=2$ .

$$m[1, 2] + m[3, 3] + p_0 \cdot p_2 \cdot p_3$$

$$= 120 + 0 + 60 = 180 \quad \dots \quad (2)$$

$$m[2,4] \Rightarrow k \rightarrow 2, 3.$$

For,  $k=2$  ;

$$m[2,2] + m[3,4] + p_1 \cdot p_2 \cdot p_4.$$

$$= 0 + 84 + 168 = 252 \dots \textcircled{1}$$

For  $k=3$ .

$$m[2,3] + m[4,4] + p_1 \cdot p_3 \cdot p_4.$$

$$= 48 + 0 + 56 = \textcircled{104} \dots \textcircled{2}$$

---

$$\underline{m[1,4]} = k \rightarrow 1, 2, 3.$$

For  $k=1, k=2$

$\textcircled{k=3}$   $\times$

$$m[1,1] + m[2,4] + p_0 \cdot p_1 \cdot p_4.$$

$$= 0 + 104 + 140$$

---

---

w

How to form the optimal parenthesis · ??

1	0	1	1	3	
2	0	2	3		S
3	0	3			
4	0				

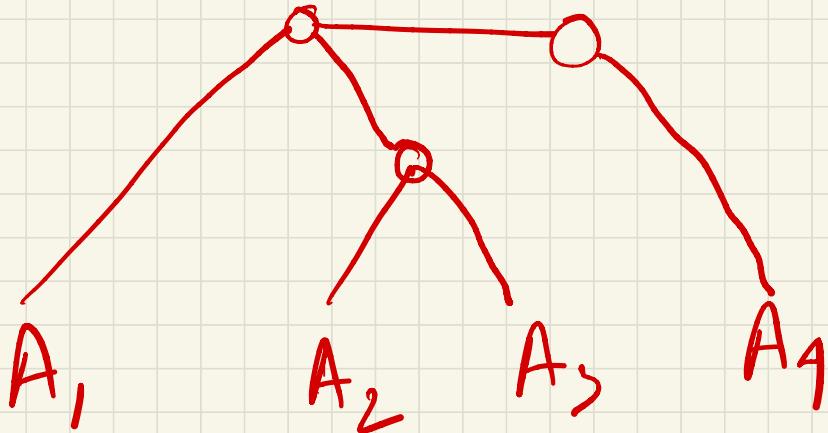
$$(A_1 \cdot A_2 \cdot A_3) \cdot (A_4) \quad (5)$$

$(1, 4) \rightarrow 3$  (Here K was 3).

$(1, 3) \rightarrow 1$  (Here K was 1).

$$(A_1) \cdot (A_2 \cdot A_3) \cdot (A_4)$$

$$((A_1 \cdot (A_2 \cdot A_3)) \cdot A_4) \cdot \boxed{U}$$



time complexity -  
 MCM complexity  $\Rightarrow n^2 \times n$   
 $= O(n^3)$ .

Space complexity  $\Rightarrow (n^2 \times 2)$  -

How can you reduce the space complexity?

~~Ans:~~ Vacant spaces of  $m$  can be filled  
 for  $(S)$  matrix



# longest common subsequence (LCS)

- 09 Sep 20

LCS is like string matching algorithm.

✓ Str 1 :- a b c d e f g h i j -  
✓ Str 2 :- ~~c d g i~~ -

Are these strings equal → No.

What is a sequence ?.

↳ A particular fixed pattern.

DNA sequencing.

String matching -

abc d.  
bcd.

1) - c d g i - / -  
2) - d g i - -  
3) - g i - -  
4) - i -

LCS = cdgi

$S_1 \rightarrow a \ b \ c \ d \ \textcircled{e} \ f \ g \ h \ i \ j$   
 $S_2 \rightarrow \underline{\phantom{c}} \ c \ c \ \underline{d} \ \underline{g} \ i$

- 1) e g i    2) c d g i  
 3) d g i    4) g i    5) i

LCS = c d g i

h

$S_1 = a \ b \ \underline{d} \ a \ c \ e$   
 $S_2 = \underline{b} \ \underline{a} \ b \ c \ e$

1) b a c e    2) a b c e ✓

LCS = bace, abce ✓

## LCS using Recursion

A	b	d	\o
	0	1	2

i = 0  
j = 0

B	\a	b	c	d	\o
	0	1	2	3	1

int LCS(i, j) -

if(A[i] == '\o' || B[j] == '\o')

return 0;

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

ret (1 + LCS(i+1, j+1))

else return

max(LCS(i+1, j), LCS(i, j+1))

A[0] B[0]  
b, a

= 2

✓ A[i, j], B[i, j] = 1  
d, a

A[0] B[1]  
b, b

= 2  
Next page

✓ A[2], B[0]  
\o, a

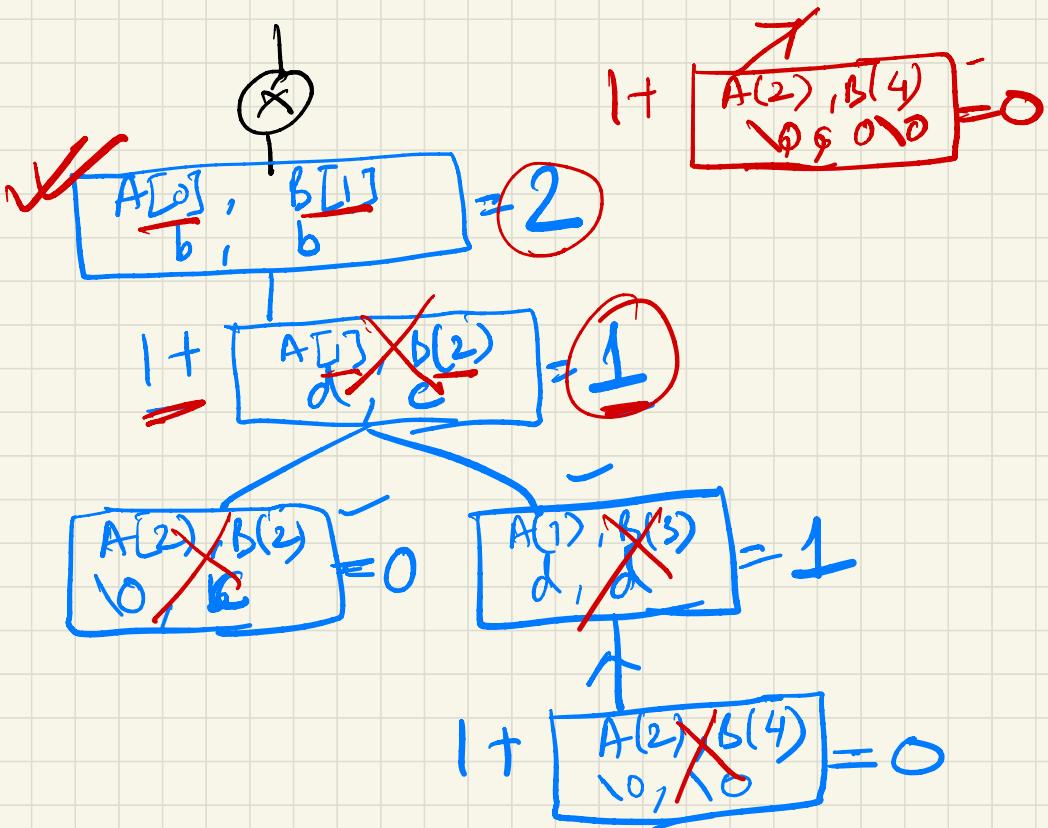
A[1], B[1]  
d, b

A(2), B(1)  
\o, b

A(1), B(2)  
d, c

A(2), B(2)  
\o, c

A(1), B(3)  
d, d



LCS length = 2

LCS using memoization

		b	a	g	d	v	
		b	0	2	2		
		A	d	1	1	1	1
b	0	0	0	0	0	0	
d	1	1	1	1	1	1	
v	0	0	0	0	0	0	

$A(2), B(0) = 0$

→ Time complexity using recursion =  $O(2^n)$   
 → " " using memoisation =  $O(m \times n)$

So, using memoisation time complexity reduced from  $O(2^n) \rightarrow O(m \times n)$ .

LCS using Recursion  $\checkmark$       } Top down approach.  
LCS using Memoisation  $\checkmark$       }

LCS using Iterative approach  $\rightarrow$  Bottom up.

At  $i=0, j=0$   $\rightarrow$  if ( $A[i] == B[j]$ )  
initial values  
 $(i=1, j=1)$        $LCS[i, j] = [1 + LCS[i-1, j-1]]$   
else       $LCS[i, j] = \max\{LCS(i-1, j),$   
                        array  $\rightarrow LCS(i, j-1)\}$ .

$m \rightarrow$	$A$	$b$	$d$
-----------------	-----	-----	-----

$n \rightarrow$	$B$	$a$	$b$	$c$	$d$
		1	2	3	4

[Changes]  
 $\hookrightarrow$  NO NULL  
 $\hookrightarrow$  Index starts with 1 and not 0.

LCS array

	0	1	2	3	4	(m x n)
B →	a	b	c	d		
A ↓	0	0	0	0	0	
1	b					
2	d					

By prefilling with '0', it will help the algo to start.

$A(1), B(1)$  → If they are not equal, find the max value from North & West (cells)

North value = 0, West Value = 0.

$A(1), B(2)$ , →  $(b, b)$  → If they are equal add 1 to the North West diagonal element.

So, LCS length = 2

How, to find the sequence ?

	a	b	c	d	
a	0	0	0	0	0
b	0	0	1	0	1
d	0	0	1	1	2

LCS =  $\rightarrow b \ d$  Start :- Stone .  
 Str 1 :- longest

	l	o	n	g	e	s	t
o	0	0	0	0	0	0	0
i	0	0	0	0	0	0	1
s	0	0	0	0	0	1	1
t	0	0	0	0	0	1	2
z	0	0	0	0	0	1	2
o	0	0	0	1	1	1	2
n	0	0	1	2	2	2	2
e	0	0	1	2	2	3	3

~~O n e~~  
LCS = 3, Pattern = One.

~~Stone~~  
~~longest~~ = One

## 24 Sep 20 O/I knapsack Problem

Optimal Solution

↳ Greedy Method KKT ✓  
↳ Dynamic Progr., MCM, LCS, O/I ✓

↳ Branch & Bound O(1) ✗

↳ Backtracking - N Queries  
DS & Algo -

↳ KMP String Selection / Master theorem  
↳ Recurrence Relations -  
↳ NP Completeness -

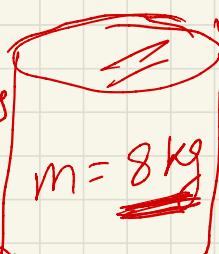
## O/I Knapsack Problem : (Dynamic Programming)

$n = \text{no. of weights}$

$= 4$

$m = \text{max weight}$

$= 8$



$$W = \{2, 3, 4, 5\} \rightarrow$$

$$P = \{1, 2, 5, 6\} \rightarrow$$

$$\hookrightarrow \text{Profit} \cdot x = \{1, 0, 0, 1\}$$

Problem Definition: Move Sack from one place to another so that the profit is maximum.

$$5 \rightarrow 6 \text{ Tk}, \quad 5, 2 \rightarrow 7 \text{ Tk}.$$

$$5, 3 \rightarrow 8 \text{ Tk}, \quad 5, 4 \rightarrow \text{X} > m.$$

Solution/Objective  $\rightarrow$  Maximum profit -

$$\boxed{x \rightarrow 0, 1} \quad x = \{1, 0, 0, 1\}, \quad \underline{x} = \{0, 1, 0, 1\}.$$

Maximum profit  $\Rightarrow \sum p_i x_i$  Should be max.

$$\hookrightarrow (1 \times 1) + (0 \times 2) + (0 \times 5) + (1 \times 6) = 7$$

$$\hookrightarrow (0 \times 1) + (1 \times 2) + (0 \times 5) + (1 \times 6) = 8.$$

Total no. of possible combinations = ??.

[0,1],

0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1

$$4 \xrightarrow{n} = 16$$

(Brute force  
combinations)

General form =  $\Theta(2^n)$

Exponential

Costly time

So, solution is optimisation problem —  
Greedy, DP, Branch & Bound.

So, 0/1 knapsack using DP → Tabulation.

P.	w.	0	1	2	3	4	5	6	7	8	→ Rows
i	↓	0	0	0	0	0	0	0	0	0	→ Columns
1	2	1	0	1	1	1	1	1	1	1	(weight)
2	3	2	0	1	2	3	3	3	3	3	
5	4	3	0	0	1	2	5	6	7	7	
6	5	4	0	0	1	2	5	6	7	8	(6,8)

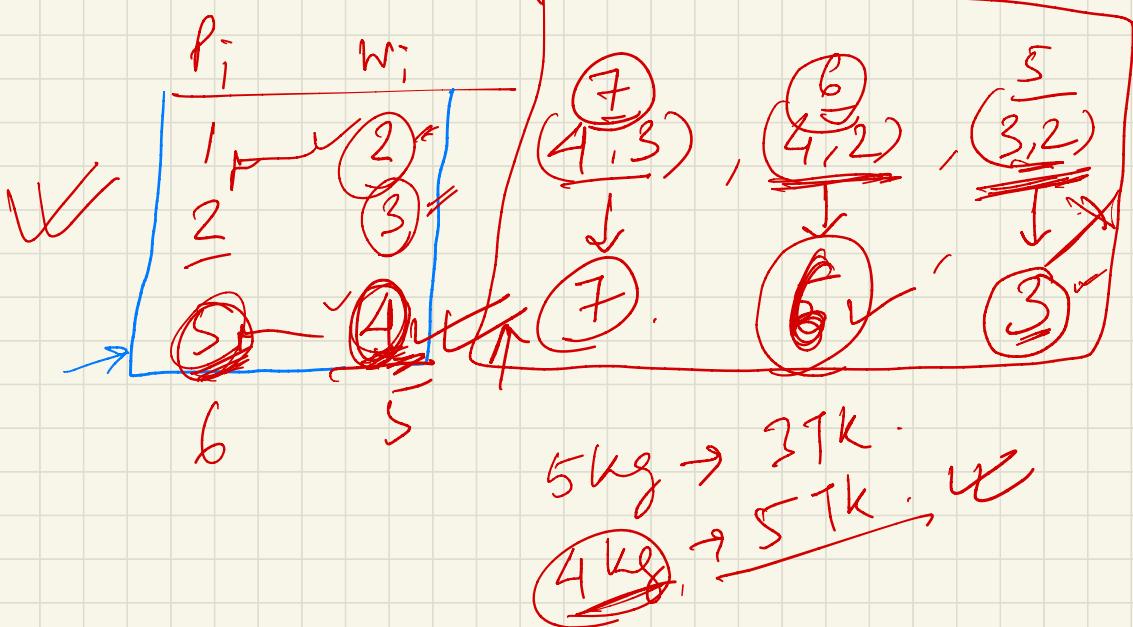
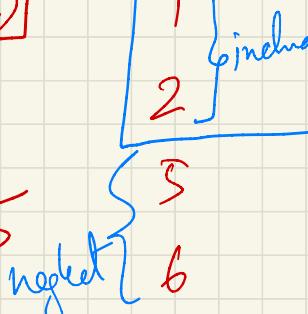
↓ Rows  
↓ Columns (items)

$m = 8$  Inside Table ↗ Profit

Rule:

$P_i$	$w_i$		$P_i$	$w_i$	$2+3=5$
1	2		1	2	
2	3		2	3	
3	4		3	1	
4	5		4	5	
5	6		5	6	
6			6		

neglect



Formula →

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

rows      Column



$$\underline{V[4,1]} = \max \left\{ \underline{V[4-1, 1]}, \underline{V[3, 1-5]} + 6 \right\}$$

$$= \max \left[ \underline{\circ}, \underline{V[3, -4]} + \underline{6} \right] \cdot \xrightarrow{\text{undefined}} \underline{-w[1]} := \underline{5}.$$

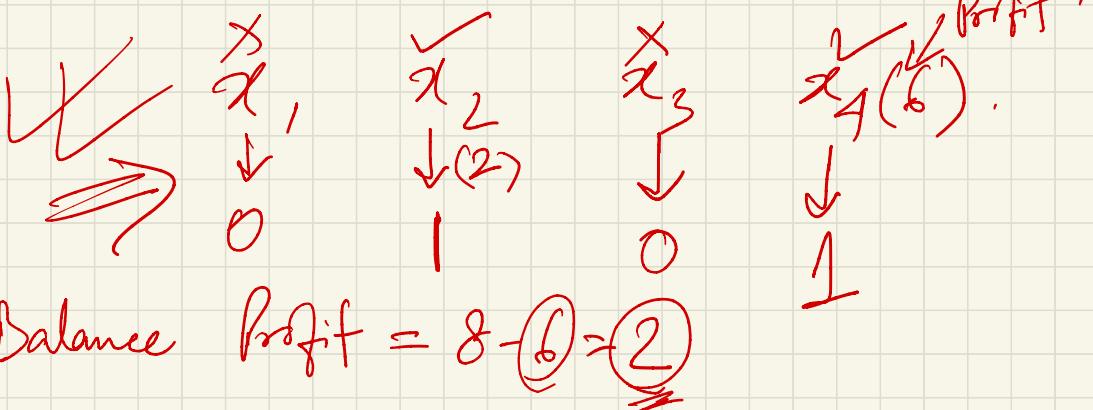
$$\underline{V[4,5]} = \max \left[ \underline{v(3,5)}, \underline{V[3,0]} + 6 \right]$$

$$= \max [5, 8+6] = \underline{13}$$

~~$$\underline{V[4,6]} = \max \left\{ \underline{V[3,6]}, \underline{V[3,6-5]} + 6 \right\}$$~~

$$= \max \left\{ 6, v(3,1) + 6 \right\} = \max \{ 6, 6 \} = 6.$$

$$\checkmark V[6,8] = \underline{\underline{8}} \rightarrow \text{Total profit earned.}$$



Profit Remaining = 2 - 2 = 0.

$$d = \{0, 1, 0, 1\} . \quad P = \{1, 2, 5, 6\}$$

$\sum p_i x_i$ , should be maximum (optimal solution).

$$\sum p_i x_i = 2 + 6 = 8 \text{ (Max profit)} .$$

---

D/I knapsack Problem.

---

P →	0	1	2	3	4
	0	1	2	5	6

$$m = 1 .$$

$$m = 8 .$$

wt	0	2	3	4	5
	0	2	3	4	5

Program 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)

      K[i][w] = 0;

    else if (w+[i] <= w)

      K[i][w] = max(P[i] + K[i-1]  
                  [w-w+[i]], K[i-1][w]);

    else

      K[i][w] = K[i-1][w];

    }

  cont << K[m][w];

}

