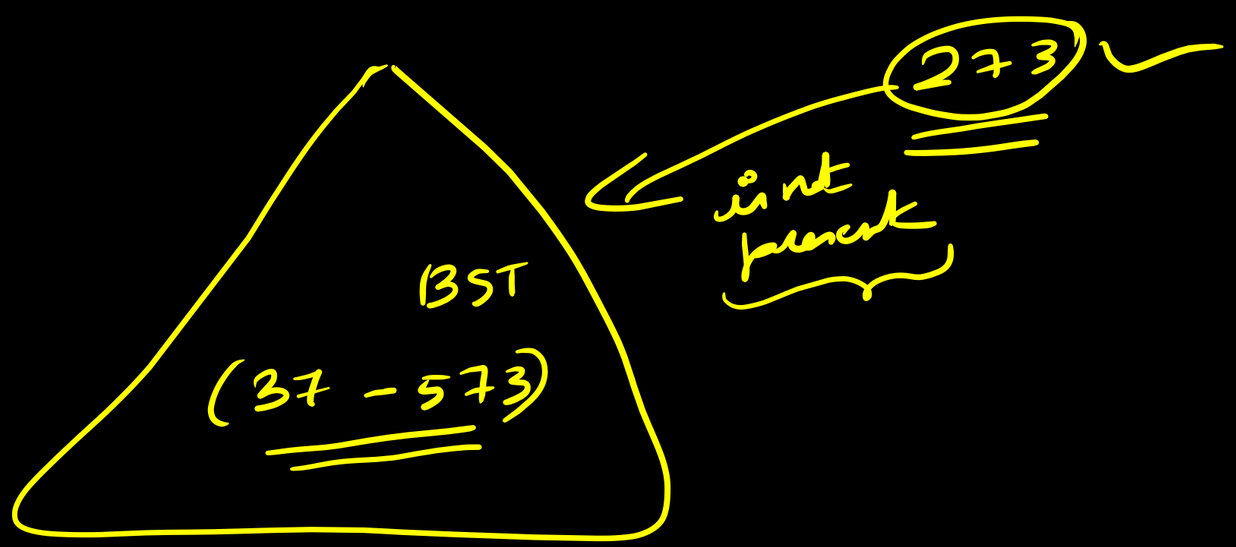
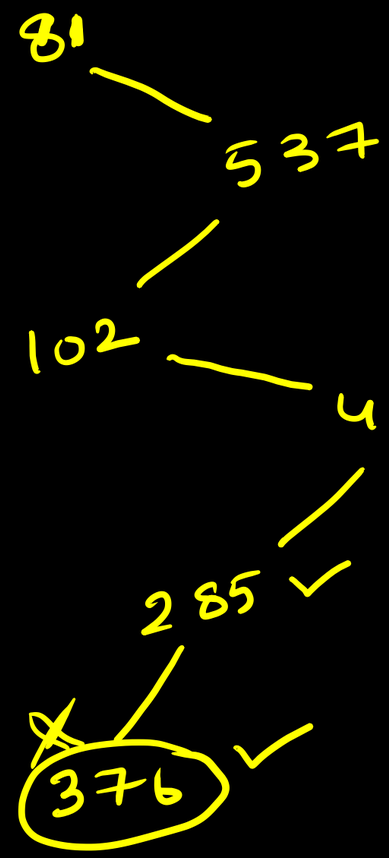
→ A BST stores values in the range 37 to 573. Consider the following sequences of keys for unsuccessful search of 273.



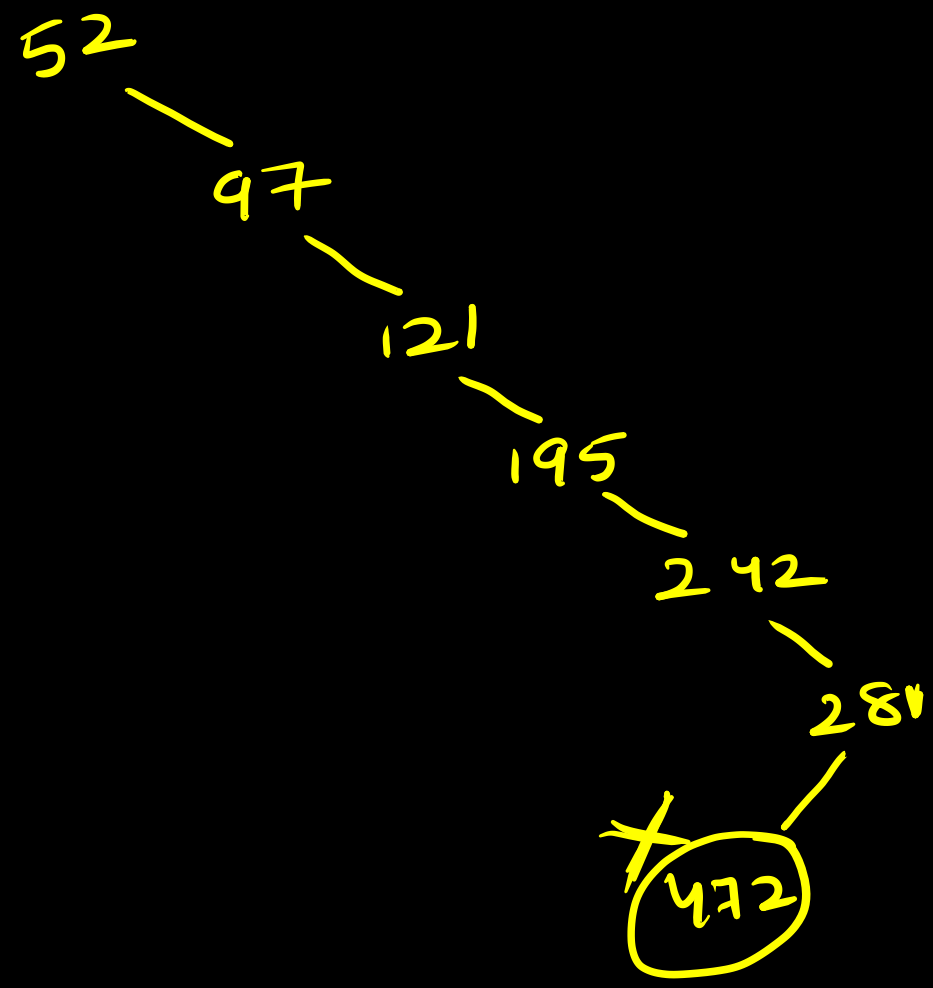
81, 537, 102, 439, 285, 376, 305

273

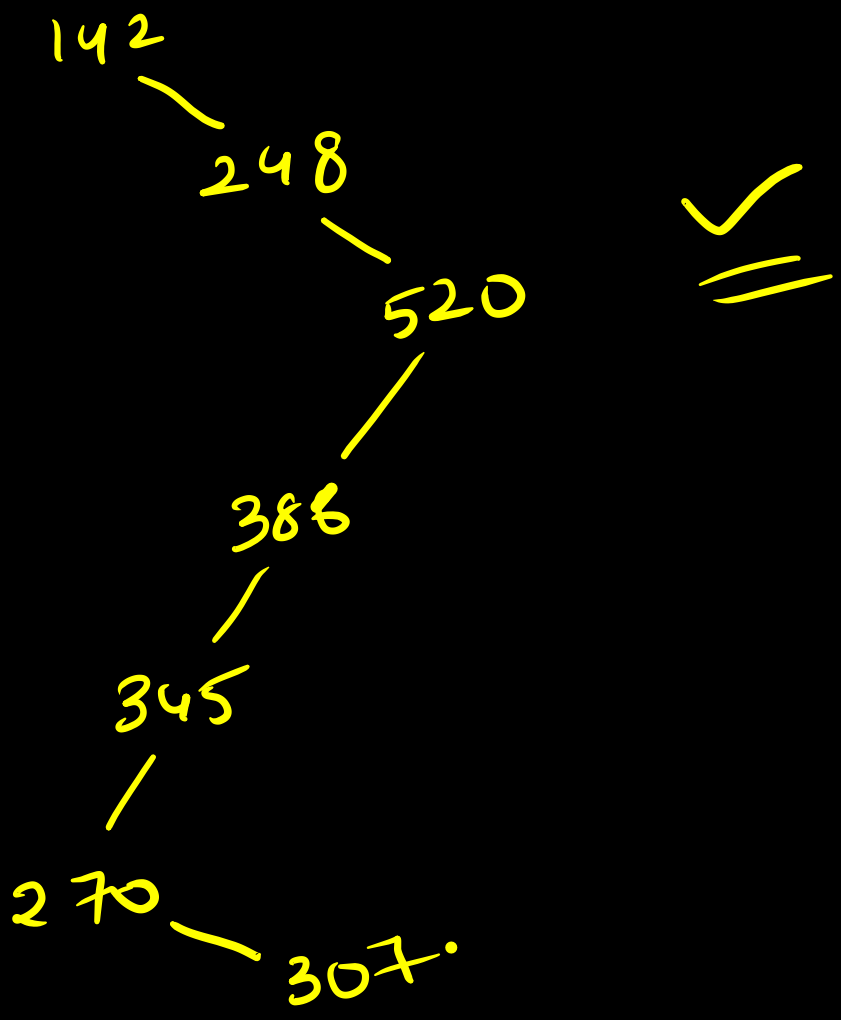


Operators	Associativity
`( ) [ ] -> .`	left to right
`! ~ ++ -- + - *` (*type*) `sizeof`	right to left
`* / %`	left to right
`+ -`	left to right
`<< >>`	left to right
`< <= > >=`	left to right
`== !=`	left to right
`&`	left to right
`^`	left to right
`|`	left to right
`&&`	left to right
`||`	left to right
`?:`	right to left
`= += -= *= /= %= &= ^= |= <<= >>=`	right to left
`,`	left to right

52 , 97 , 121 , 195 , 242 , 381 , 472

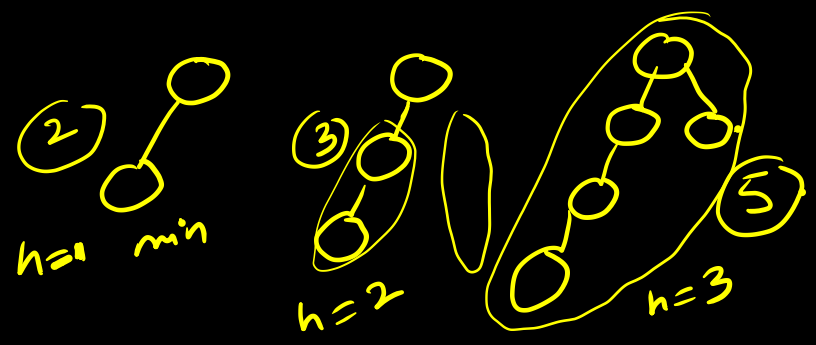


142, 248, 520, 386, 345, 270, 307.

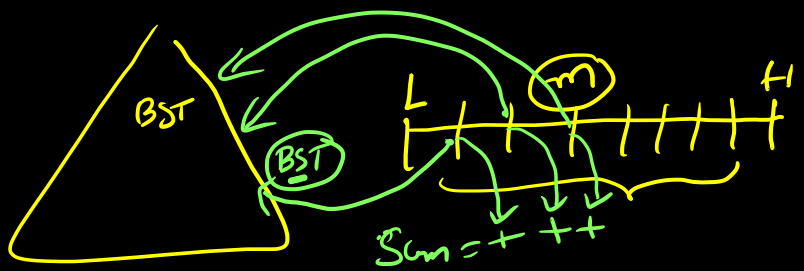


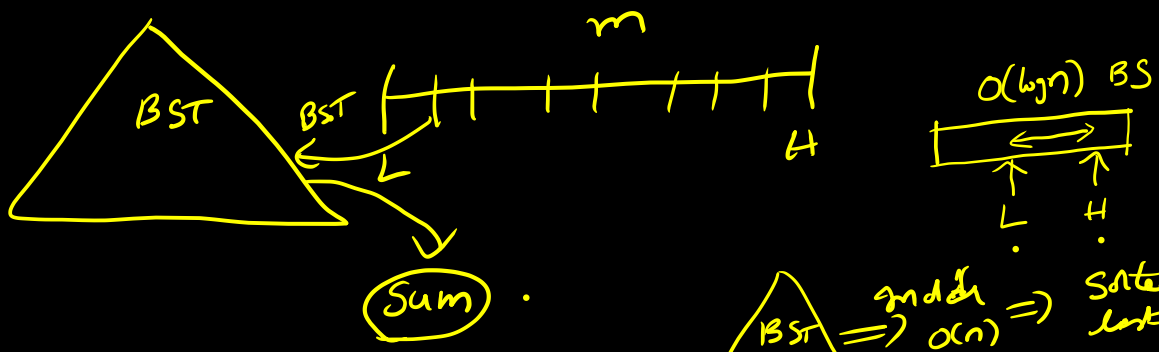
Note
→ In a binary tree, for every node the difference b/w the no of nodes in left and right subtree is atmost 2. If the height of the tree is $h>0$, then minimum no of nodes in the tree are

a) 2^{h-1} b) $2^{h-1}+1$ c) 2^h-1 d) 2^h.



→ Suppose we have a BST (balanced) T holding n numbers. We are given two numbers 'L' and 'H' and wish to sum up all numbers 'T' that lie b/w 'L' and 'H'. Suppose there are m such numbers in T. If the tightest upper bound to compute the sum is $n^a \log^b n + m^c \log^d n$

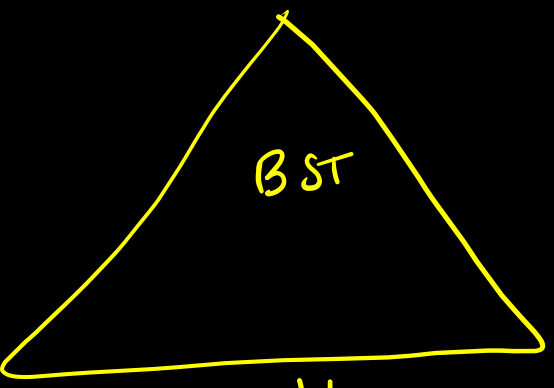


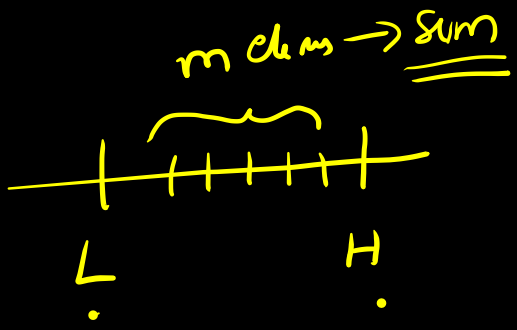


$m(\log n)$

In order traversal → sorted list.

$O(n)$ - $O(m)$ $O(n+m)$



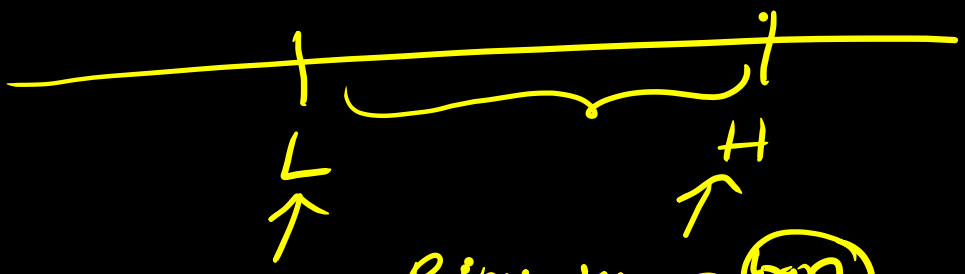


↓ In order traversal $O(n)$.

Sorted list

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

↓ traversal ↓ Binary search ↓ sum



Binary search = $\log n$

Expression tree :-

→ operator will be internal nodes and operands will be leaves

Inorder – a+b

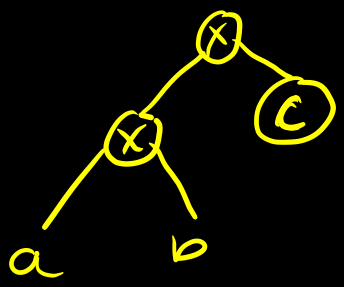
preorder = +ab

post = ab+

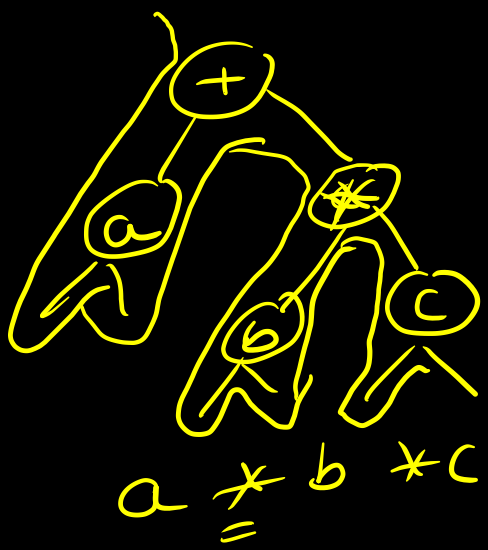
a * b * c

I , II .

left associa

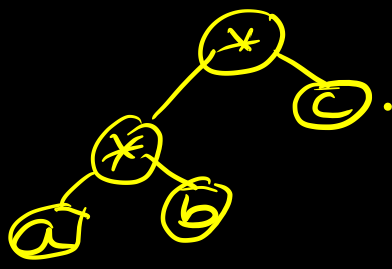


a + b * c



a * b * c

a + b * c .



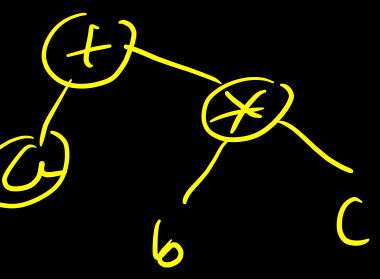
eg.

a+b

a*b

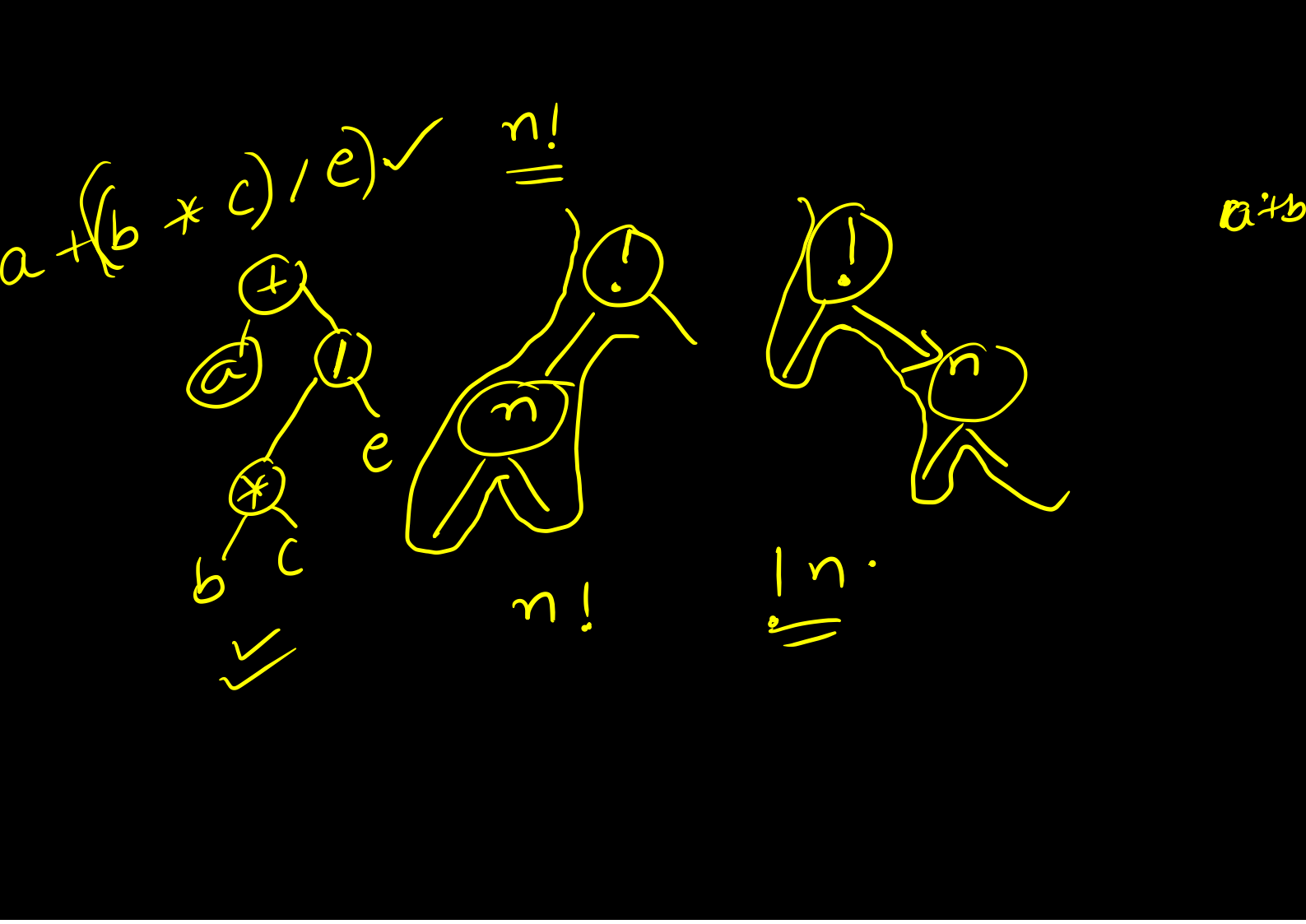
a+b*c

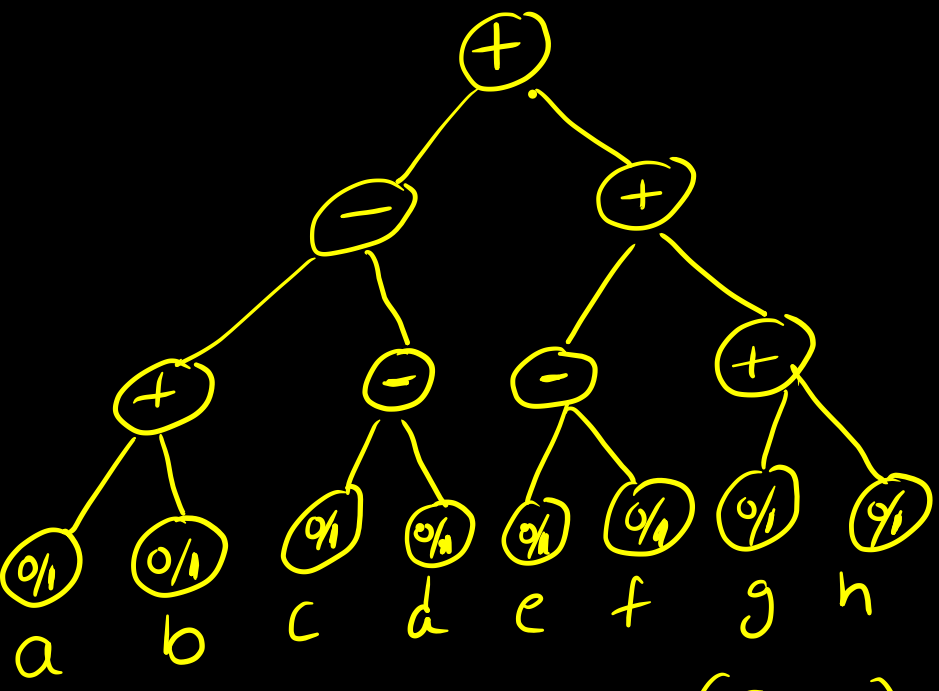
a*b*c



diff.

$\log n$


a+(b*c)/e)
n!
n!
!n.
a+b



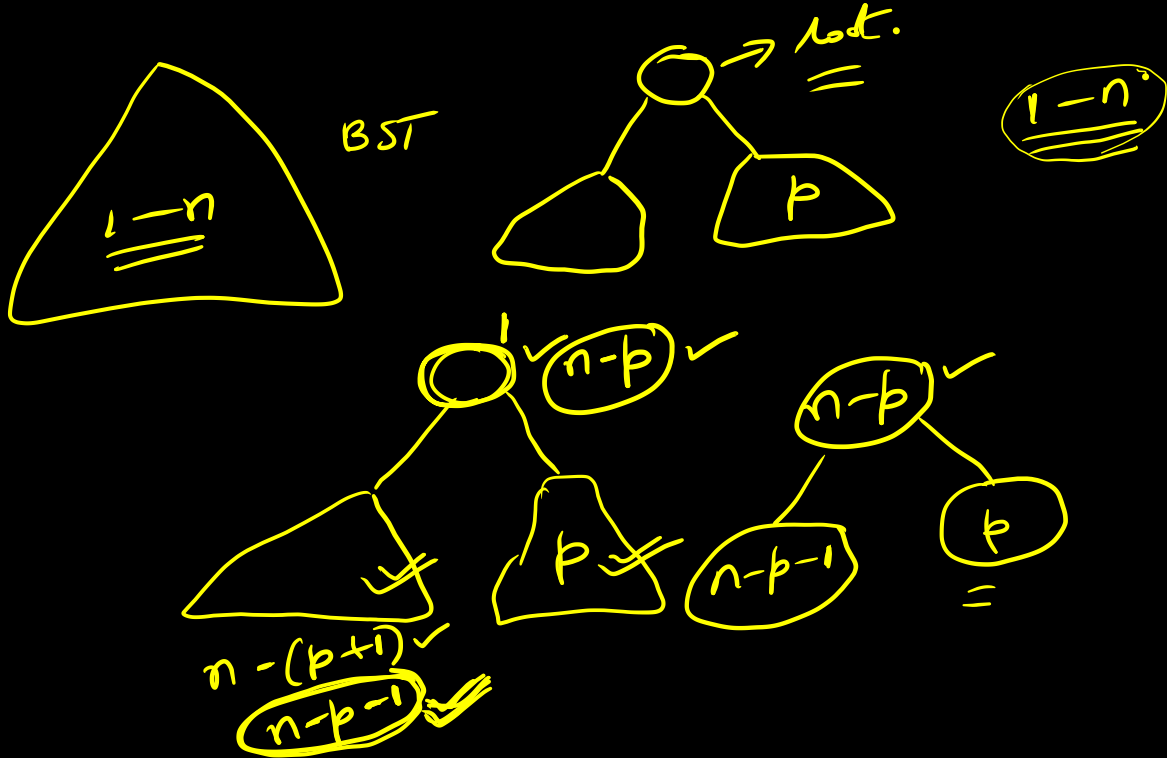
⇒ maximum value

$$\underset{1}{((\underset{1}{a}+\underset{1}{b})} - \underset{0}{(\underset{1}{c}-\underset{1}{d})}) + \underset{1}{((\underset{1}{e}-\underset{0}{f})} + \underset{1}{\underset{1}{(g+h)}}) \checkmark$$

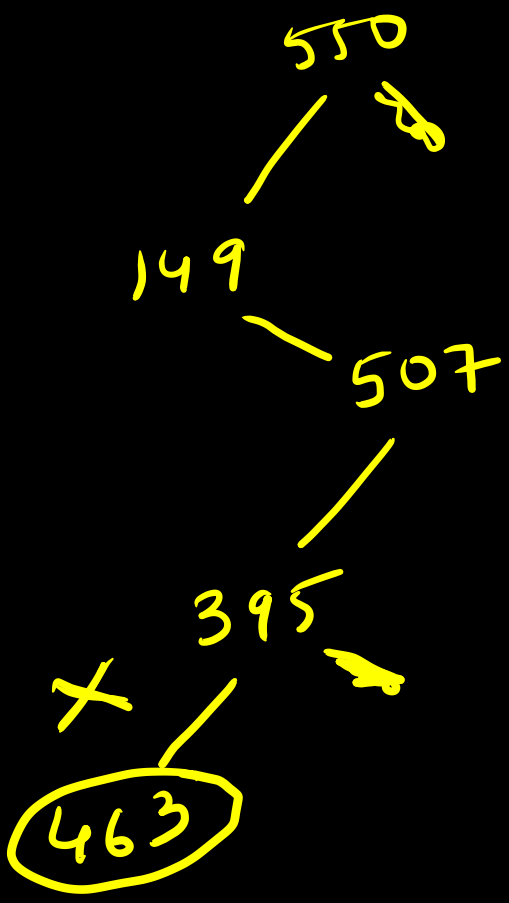
$$= 6$$

Ques: The numbers 1.... n are inserted into a BST. In the resulting tree, the right subtree of the root contains 'p' nodes The first number to be inserted in the tree must be:

a) p b) p+1 c) n-p d) n-p+1



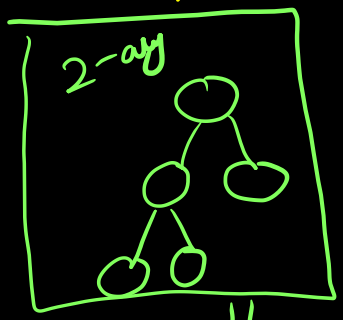
550, 149, 507, 395, 463, 402, 270

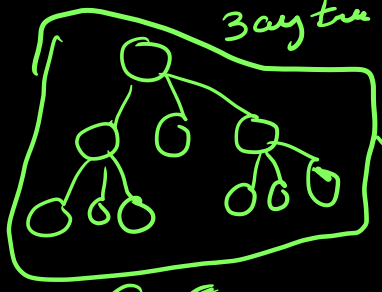


Gate → A Complete n-ary tree is one in which every node has 0 or n sons.

If 'X' is the number of internal nodes of a complete n-ary tree, the number of leaves in it is

a) X(n-1) b) ~~X+n~~ Xn-1 c) Xn+1 d) X(n+1)





Internal node — none leaves

Heap → Complete binary the definition was different.

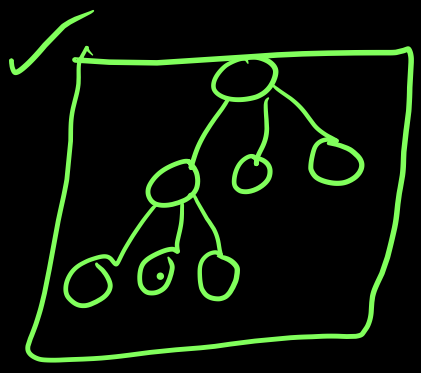
Leaves = (3)

n = 2
x = 2

Completeness is different in different books.

Gate: The number of leaves in a rooted tree of n nodes with each node having 0 or 3 children:

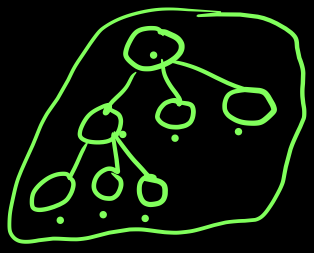
a) n/2 b) (n-1)/3 c) (n-1/2 d) (2n+1)/3.



L = 5.

n = 7

d) $\frac{2*7+1}{3} = 5$.



n = 7

L = 5.

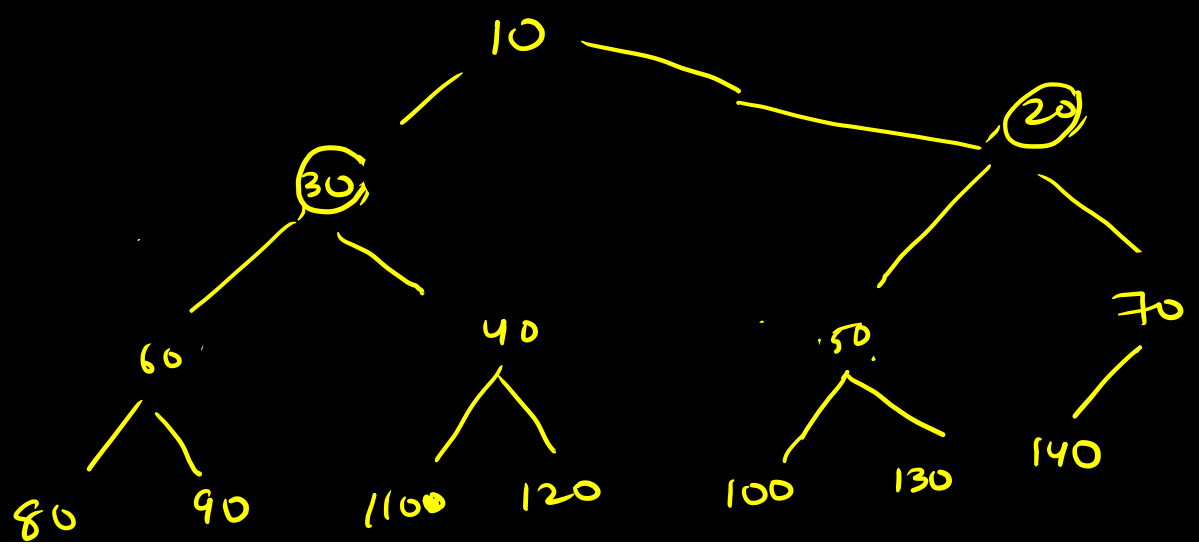
$\frac{2\times7+1}{3}$

In order of a min heap is given, what is the post order:

80 60 90 30 110 40 120 10 100 50 130 20 140 70

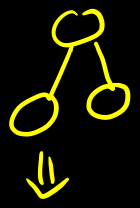
→ root is always min.

In order ⇒ left Root Right.



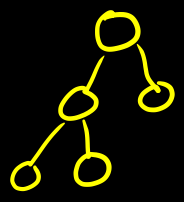
→ A binary tree T has n leaf nodes. the number of nodes of degree '2' in T is :—— a) $\log_2 n$ b) $n-1$ c) n d) 2^n

Take example



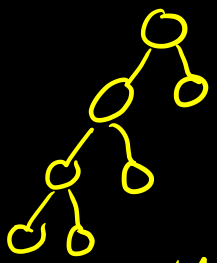
$n = 2$

deg 2 = 2-1 = 1



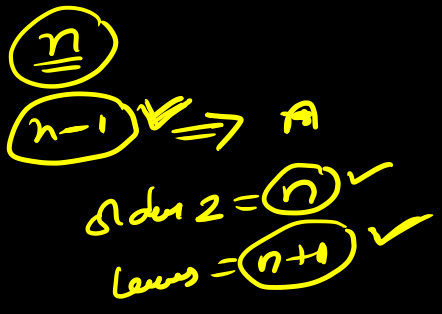
$n = 3$

deg 2 = n-1



leaves = 4

deg 2 = 3.

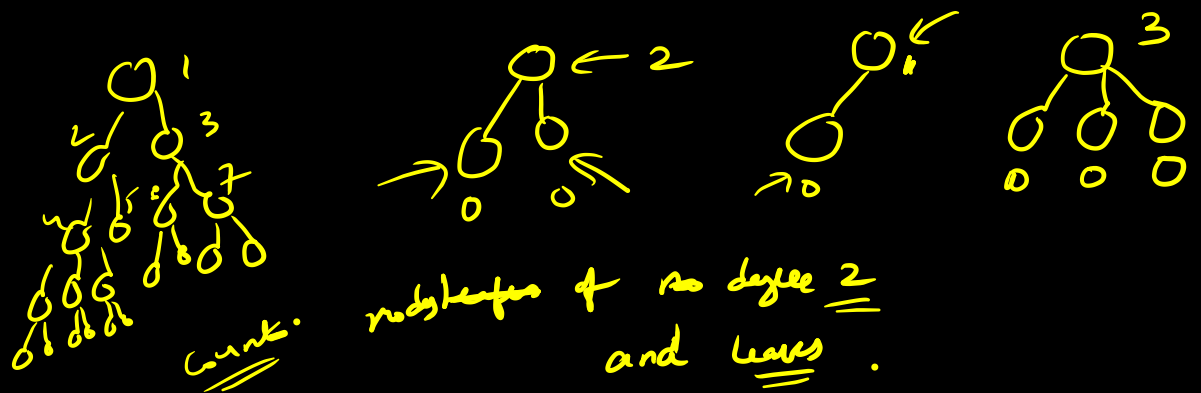


⇒ In a binary tree, the number of internal nodes of degree 1 is 5. and the number of internal nodes of degree 2 is 10. The number of leaf nodes in Binary tree is:

a) 10 b) 11 c) 12 d) 15.

one degree ⇒ not required

no of edges:

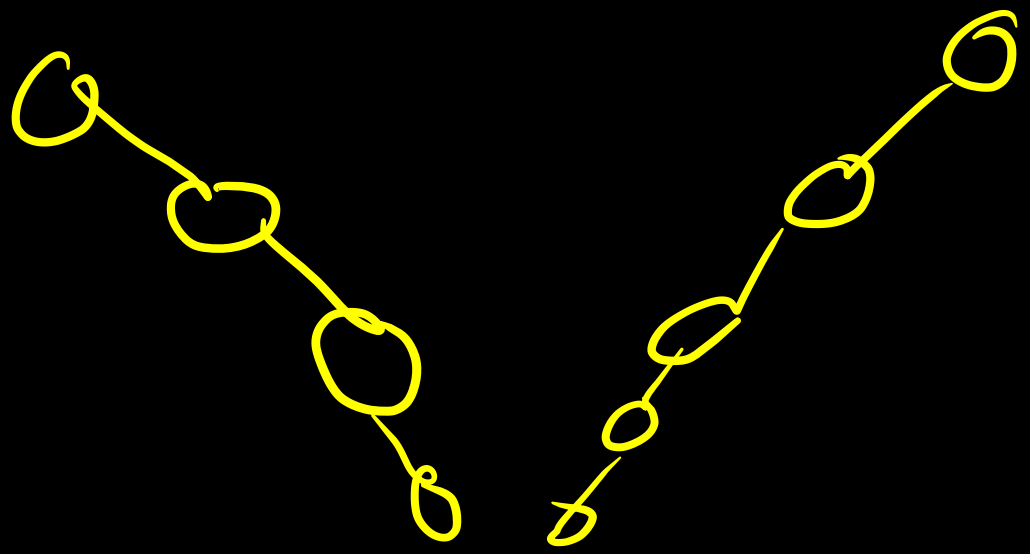


count. no of nodes of no degree 2 and leaves.

BST → Search

↓

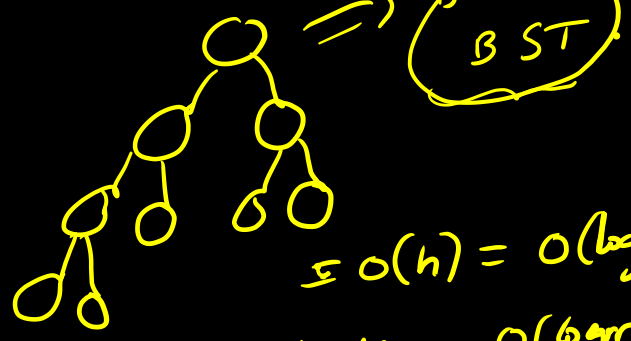
disadv → may be unbalanced:



Balanced BST:

AVL trees → Syllabus

- Red Black tree
- 2-3 tree
- hard movement ✓✓



$= O(h) = O(\log n)$

height = $O(\log n)$.

B AVL ⇒ Search time = $O(\log n)$

insertion = $O(\log n)$

delete = $O(\log n)$

B BST

AVL.

height = $O(\log n)$

Search = $O(\log n)$ → ~~$O(\log n)$~~

Q: Which of the following is true about search trees search time

	AVL	BST
✓a)	$\theta(\log n)$	$O(n)$
b)	$O(\log n)$	$O(n \log n)$
c)	$O(n)$	$O(\log n)$
d)	$O(n \log n)$	$O(n)$

Gate: which of the following statements are false!

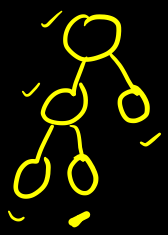
✓a) A tree with n nodes has n-1 edges

b) A labelled bock binary tree can be uniquely constructed given in post order and pre order traversals

✓c) A complete binary tree with n internal nodes has n+1 leaves

✓d) maxim number of nodes in a binary tree of height 'h' is $2^{h+1}-1$

Complete binary $\begin{cases} 0 \text{ child} \\ 2 \text{ child} \end{cases}$



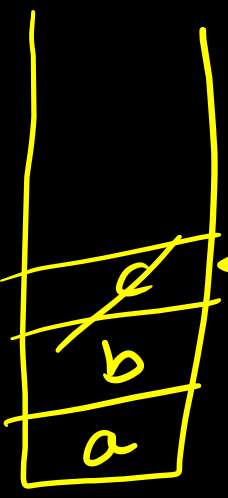
In-post

Infix → post fix.

Evaluation of post fix.

implementing Queue

using stacks.



stack

top == -1

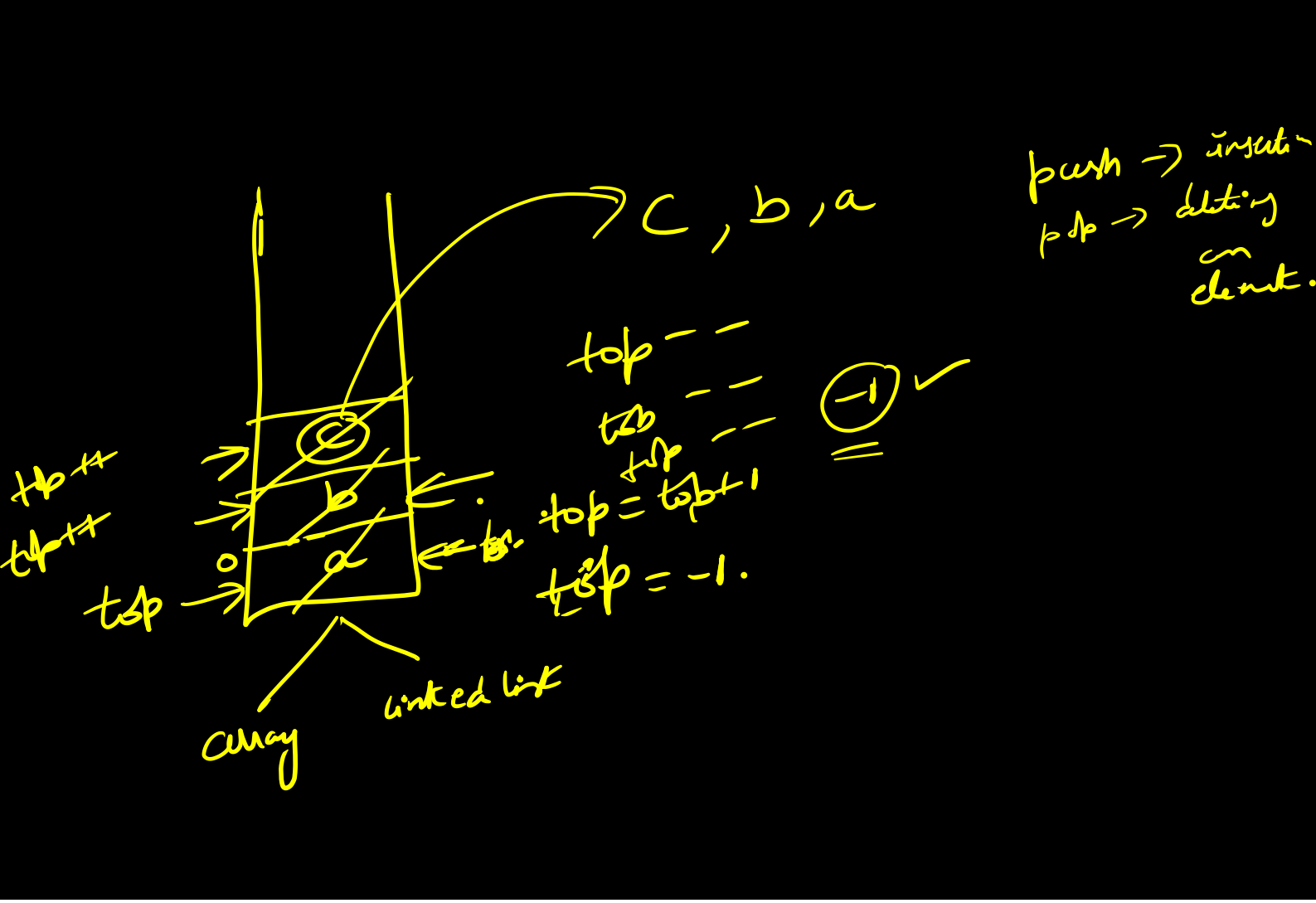
LIFO

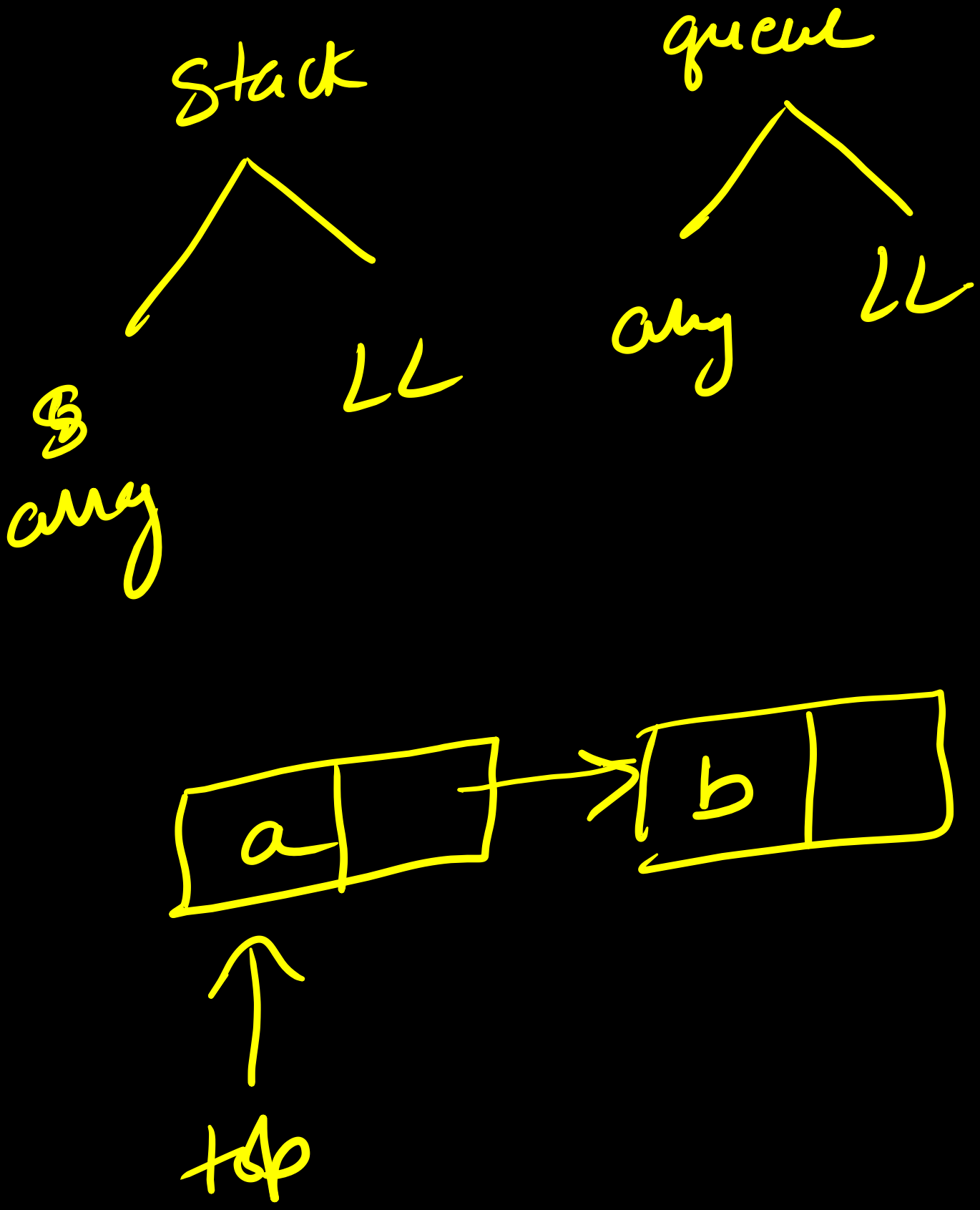
FILO

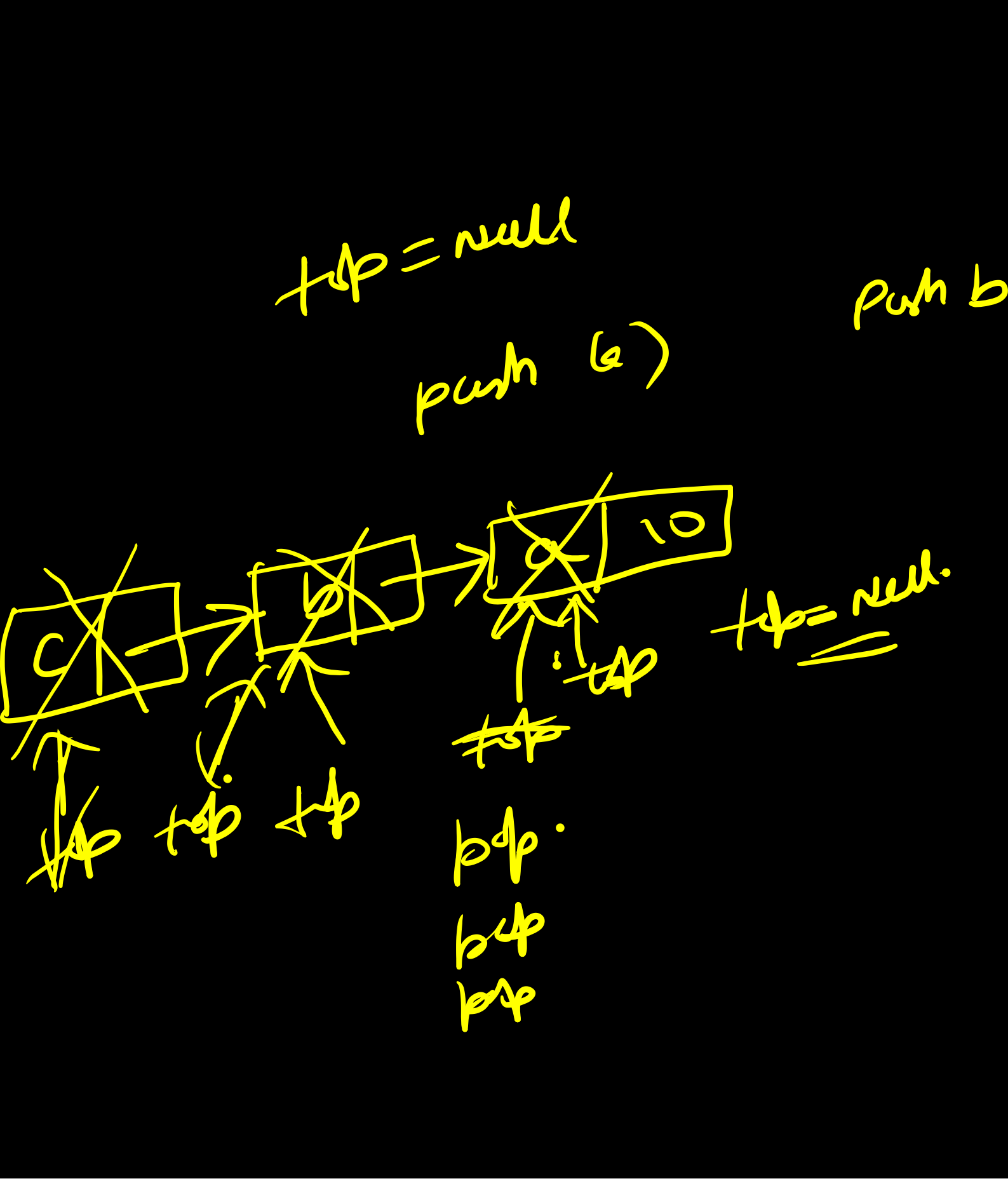
→ last in first out

push → insert

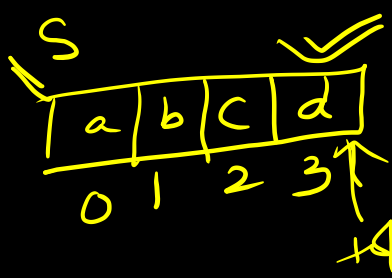
pop → delete


push → insertion
pop → deleting an element.
c, b, a
top = -1
top = top+1
top = -1.
top++
top++
top
array
linked list


Stack
queue
$
arr
LL
arr
LL
a
b
top


top = null
push (a)
push b
10
top
top
top
top
top
top
top = null.
bop.
bop
bop

Array implementation of stack:



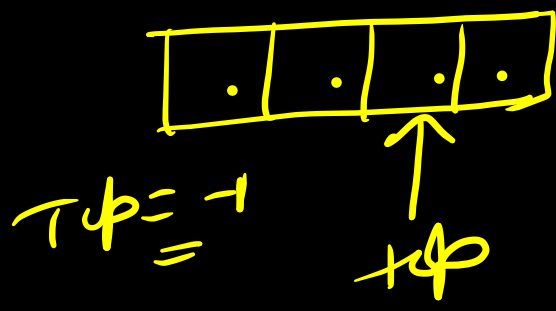
$n = 4$.

top == n-1

3 = 3

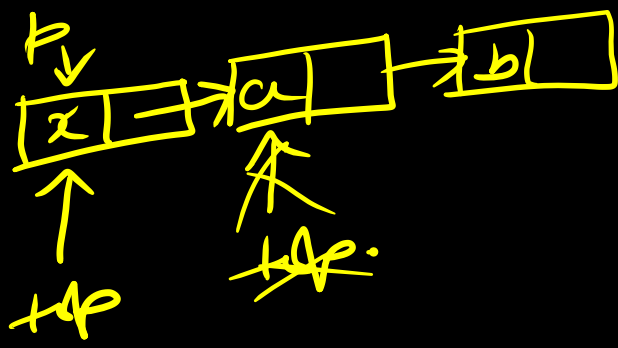
```
Push(x)
{
  if(Top==n-1)
  { pf(stack overflow)
    exit;
  }
  else
    { top++;
      S[top]=x;
    }
}
```

```
pop()
{
  if (top == -1)
  {
    pf (underflow);
    exit
  }
  else
  {
    y = S[top]
    top --
    return y
  }
```



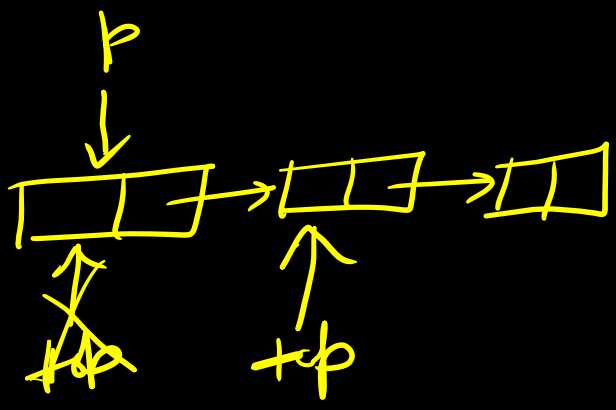
Implementing stack using LL:

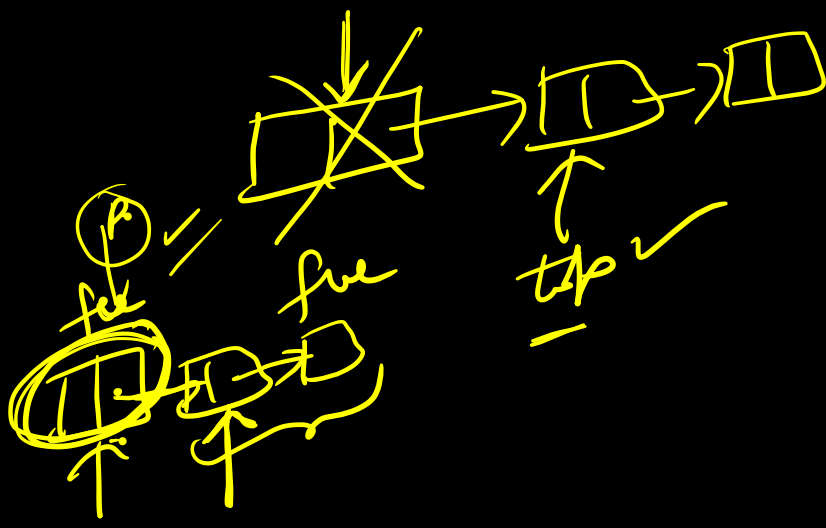
```
push (x)
{
  p = malloc (x);
  if (p == null)
    { pf (no mem)
      exit;
    }
    p -> next = top;
    top = p;
}
```

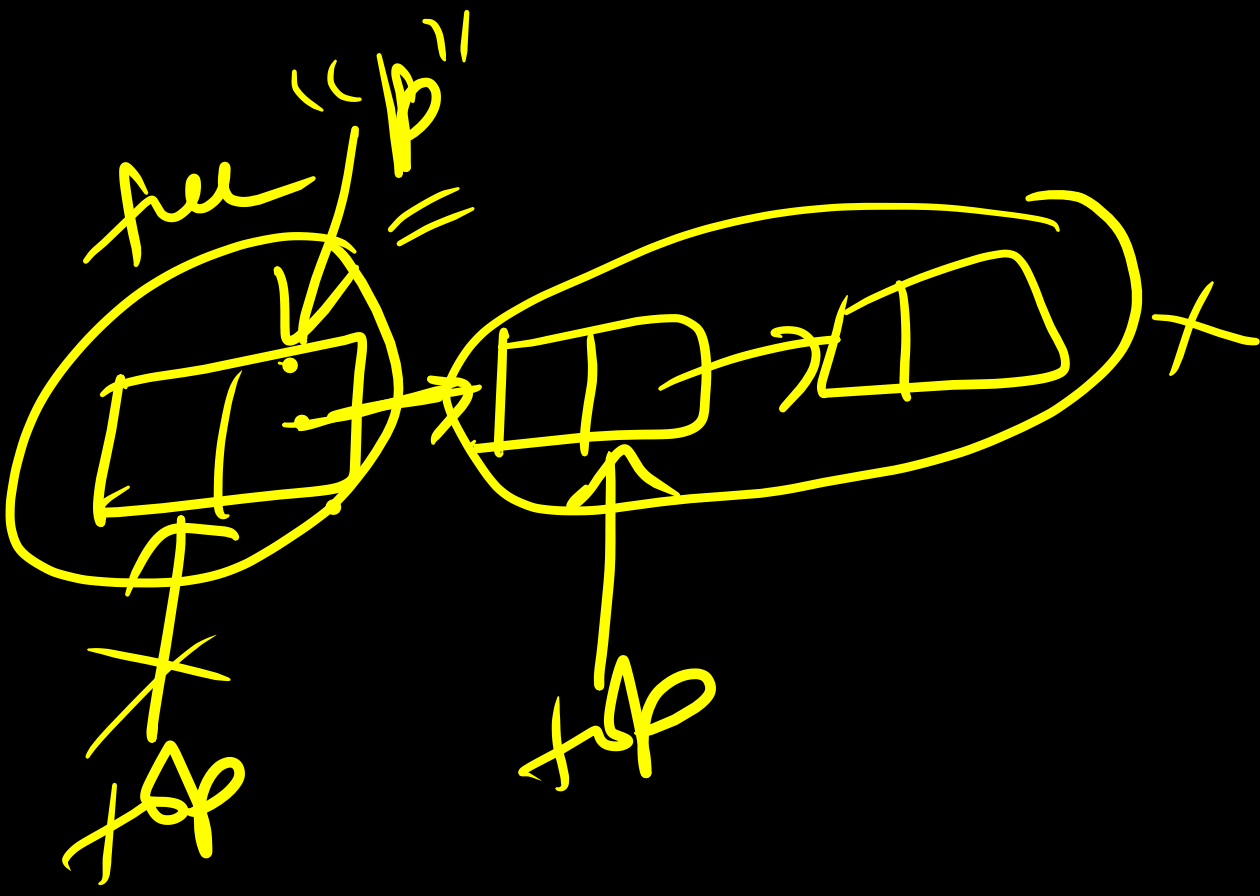


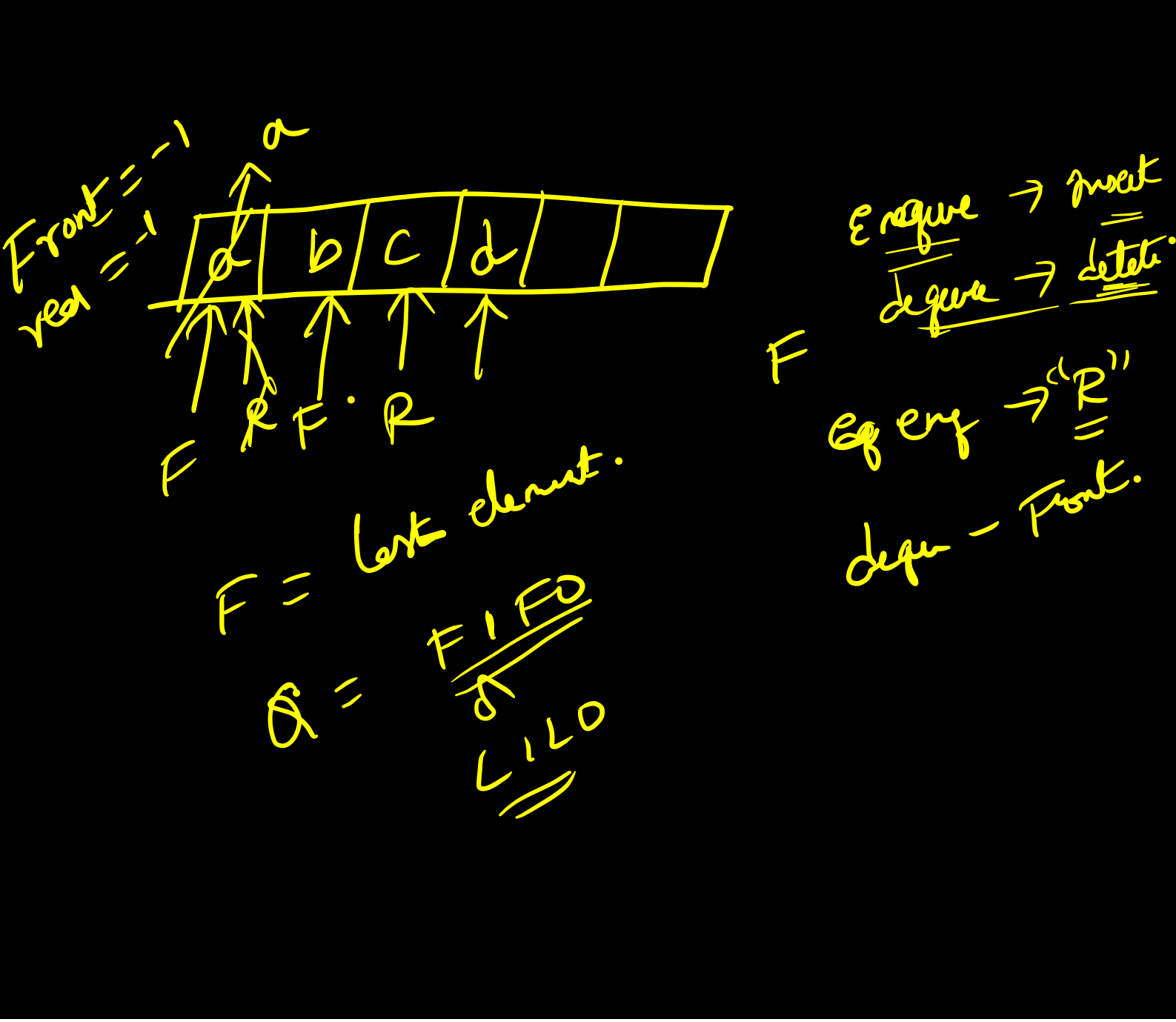
implementation of pop using LL:

```
int pop()
{ if (top==null)
  { pf( underflow);
    exit(1);
  }
  y = top->data;
  p = top
  t = top->next
  free(p);
  p = null;
  return y
}
```



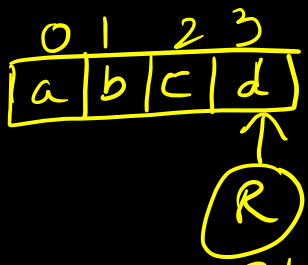





Front = -1
rear = -1
a
b
c
d
F
R
F . R
F = left element.
Q = FIFO
LILO
Enqueue → insert
dequeue → delete.
F
eq enq → "R"
deq - Front.

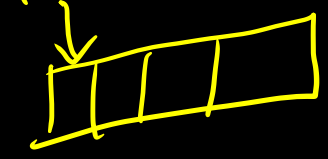
Enque: (x) → using array.

```
{ if (R+1)=N
  { pf (Q overflow);
    exit;
  }
else { if (R==-1) => Q is empty
       { F++; (R++)
       else (R++)
       Q[R] = x
       }
}
```

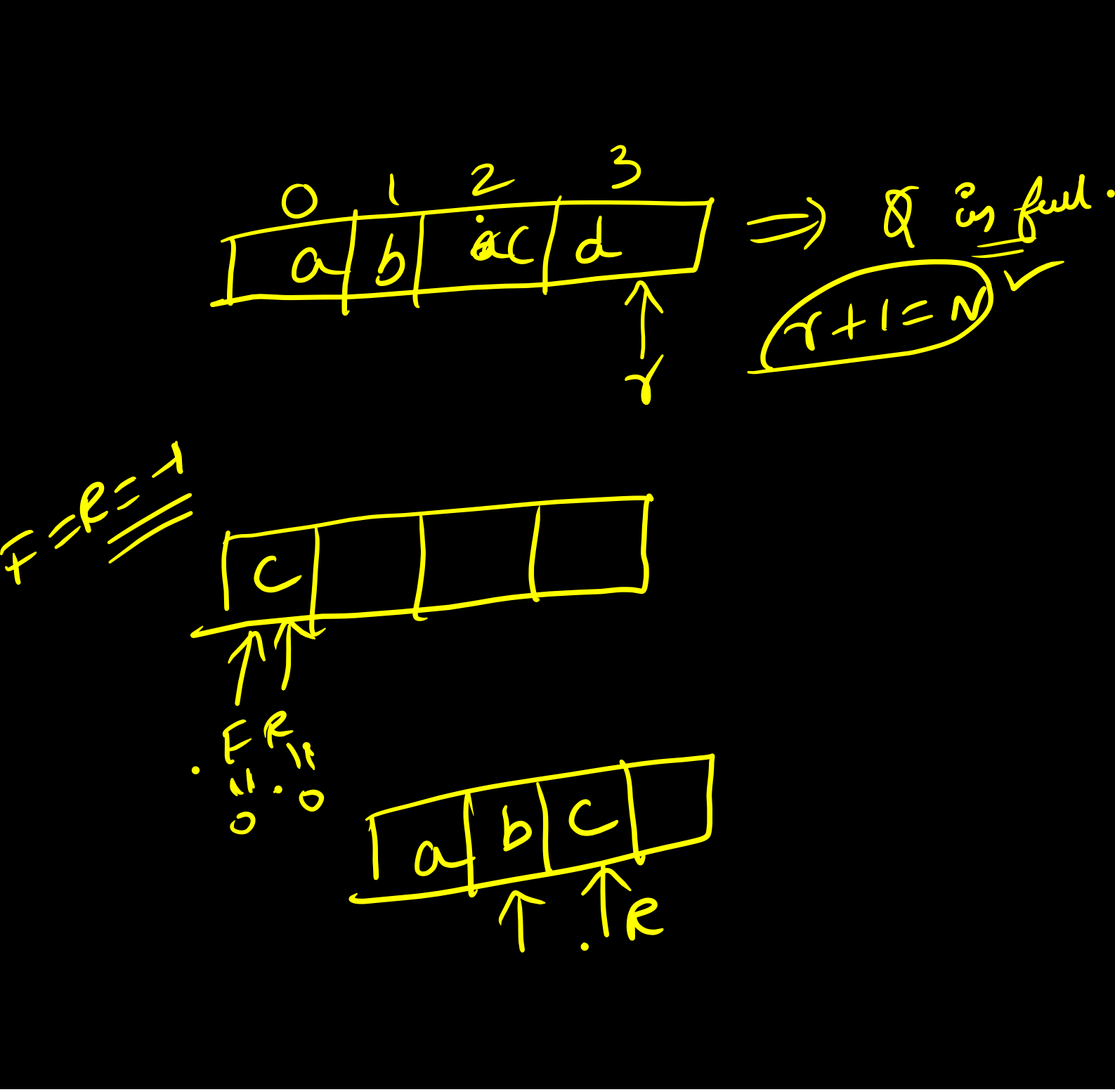


N=4

R+1=N ✓

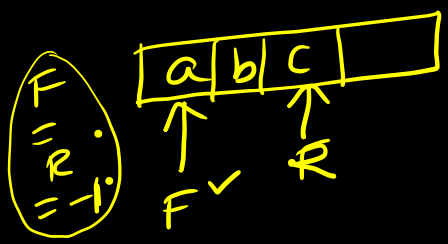


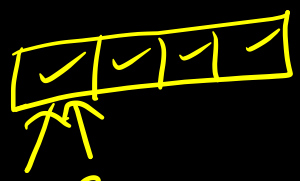
R=F=-1 .


0 1 2 3
a b c d
r
⇒ Q is full.
r+1=N
F=R=-1
c
F R
0 0
a b c
R

Dequeue:

```
{ if F == -1
  { pf (underflow)
    exit
  }
  else { y = Q[F]
         if (F == R) → only one elem
         { F = R = -1 }
         else F++
         return (y)
  }
}
```

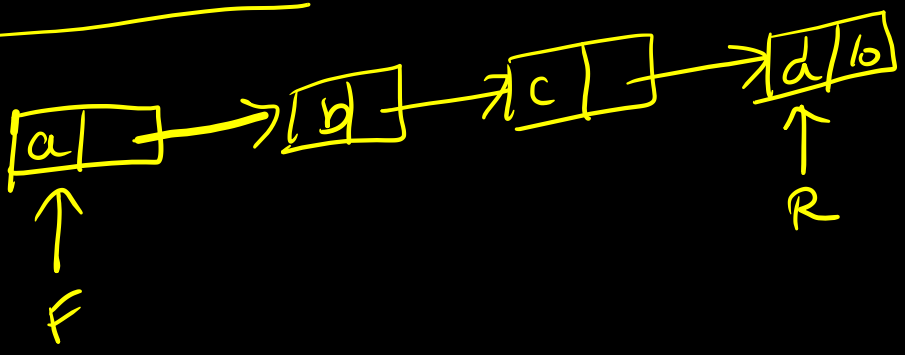




O(n) { push } stack
pop

enqueue } queue
dequeue

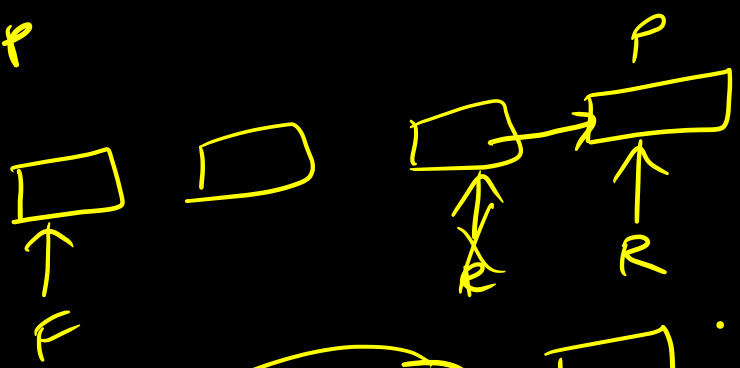
Queue using linked list:

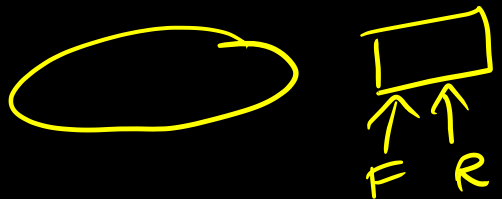



```
Enque(x)
{ p = malloc(x)
  if (p == null) { pf "no memory"; exit }
  else
  { p->data = x;
    p->next = null;
    R->next = p.
    R = p
  }
```

```
if (R == F == null)
{ F = R = P
```

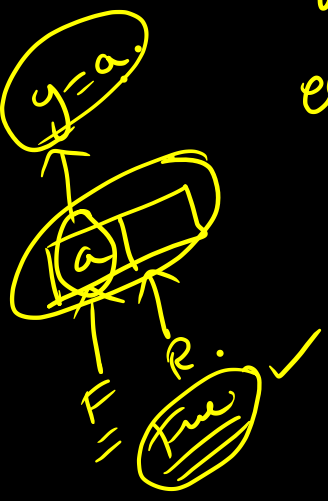
empty.

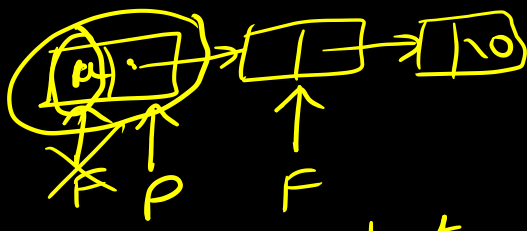




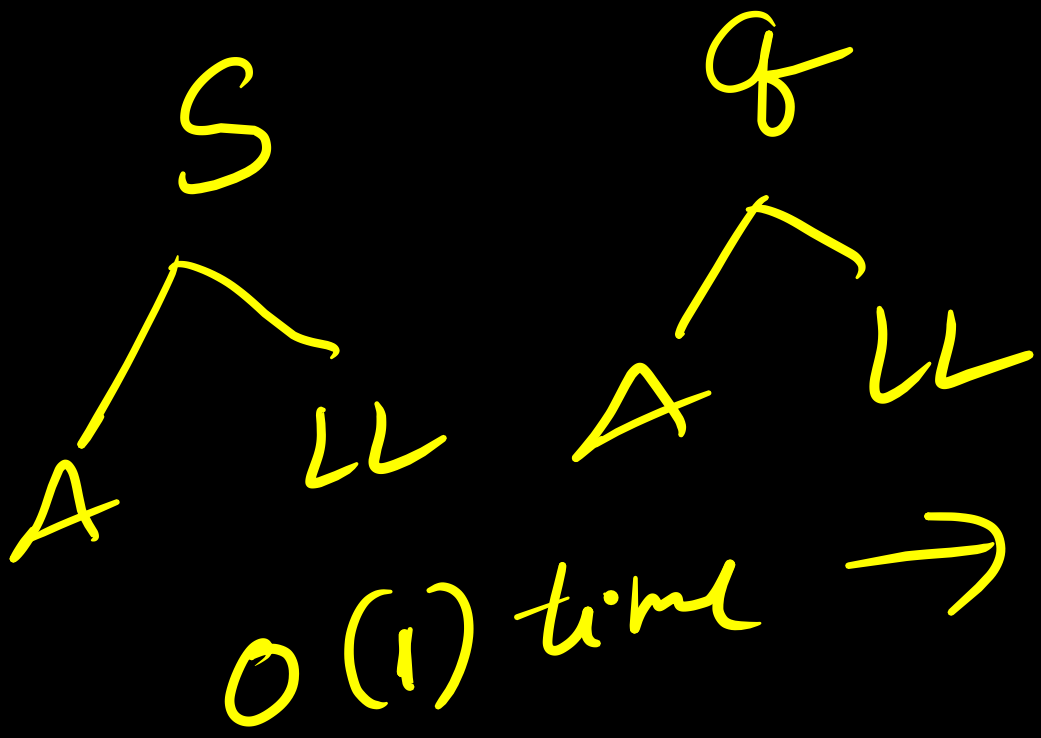
Dequeue :

```
if (F==Null) { Pf( Que underflow)}
else {  y = F->data
        p = F;
        F = F->next
        free (p)
        p = null
        return (y);
     }
```

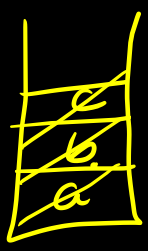


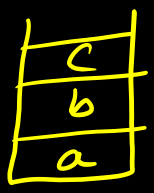



```
if (F==R) => last.
{  free(F);
   F=R=null ; return }
```


S
A
LL
q
A
LL
O(1) time

$n=3 \rightarrow (a,b,c) \rightarrow$ 3 elements are pushed followed by 3 pops. Each operation takes $x=5$ units of time. Time elapsed b/w two stack operations is $y=3$. life time of an element (a) in stack is time after push(a) and before pop 'a'. what is the average LT of all element.





push a | push b | push c | pop c | pop b | pop a.
x x x
y y y y y

$LT = a. = 35$

push b | push c | pop c | pop b.
x x
y y y

$= 19$

push c | (pop c

$= 3$

$\frac{35+19+3}{3} = 19$ ✓

$n(x+y)-x$ → not required.

$n=5$ (a, b, c, d, e), $x=4, y=7$

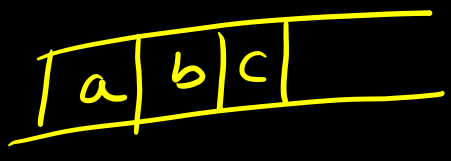
avg LT $= n(x+y)-x.$

$= 51$

Queue: ✓

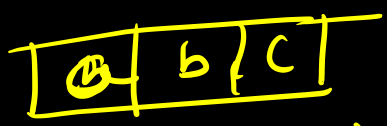
$n(x+y) - x$.

a b c



$e(a)$ | $e(b)$ | $e(c)$ | $d(a)$ =
y x y x y

In queue, ~~arrange~~ (T in same for all the ~~rest~~ element.



$e(b)$ | $e(c)$ | $d(a)$ | $d(b)$. ✓
x x x
y y y y

Infix to postfix Conversion

$a + b \overset{PF}{\Rightarrow} ab+$

$a+(b*c) \Rightarrow$
$a+(bc*)$

$abc*+$.

Evaluate an expression $\Rightarrow$

a + c & d ~~&~~ e = f / g $\Rightarrow$ postfix.

$a + (b * c)$

$a + x$

$a\ y$

* In one pass

Left → right

↓ post fix.

$a + b * c \Rightarrow$ postfix.

$a+b \Rightarrow a\ b\ +$.

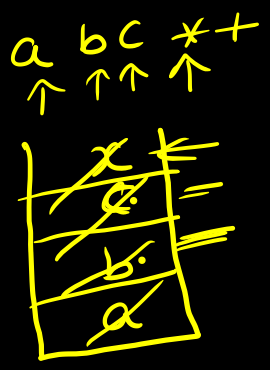
$a*b \Rightarrow a\ b\ *$

$a+(b*c)$

$a+(bc*)$

$a(bc*)+$.

$a\ bc*+$



$(b*c) = x$.

$x\ a+(x) =$