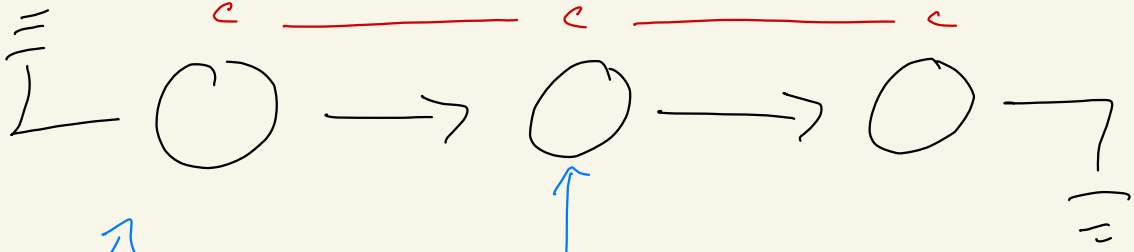
AGENDA

1. REVIEW LL
2. REVIEW S/Q
3. TREES
    1. BINARY TREE
    2. BINARY SEARCH TREE

RECURSION

BIG O

LINKED LIST



HEAD

NODE<T>

- VALUE : ANYTHING
- NEXT

TAIL

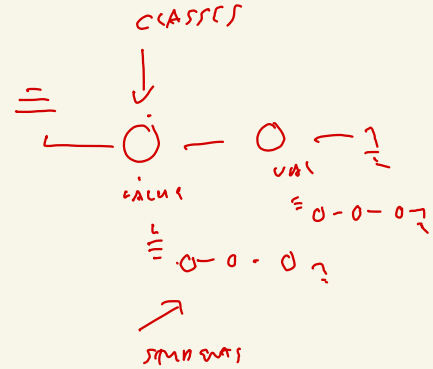
"TRAVERSED ITERATED LIST"

TRAVERSALS

- FIND HEAD (CURRENT)
- WHILE (HAVE A NODE)
  - MOVE CURRENT TO CURRENT.NEXT

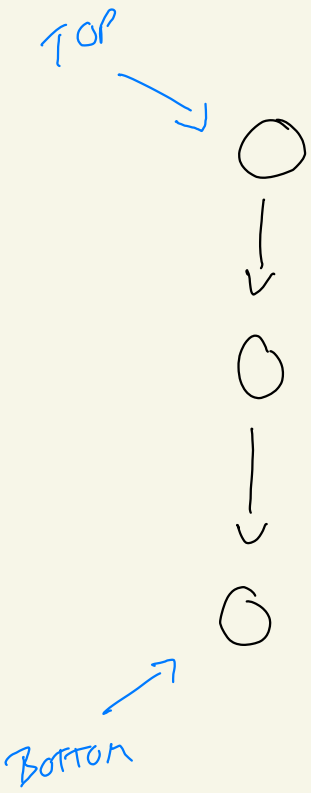
SECRET

2ND RUNNERS



STACK

"PROCESSED"



ORDER MATTERS

LAST ON FIRST OUT

AS YOU TRAVERSE... THE ENTRY IN VIEW IS REMOVED

TRAVERSAL

WHILE I CAN PEEK, POP THE NEXT ENTRY

METHODS

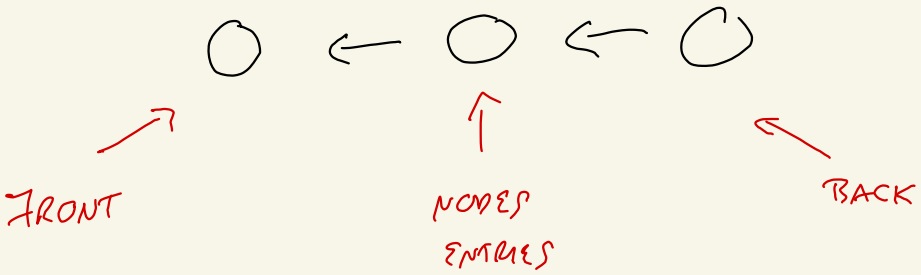
- Peek() ← BOOL / VALUE ← "TOP"
- isEmpty() ← BOOL (IF TOP)
- Push() ← PUTS ENTRY ON TOP OF STACK
- Pop() ← REMOVE & RETURN TOP

Queue

ORDER MATTERS

FIRST IN FIRST OUT

"PROCESSED"



TRAVERSE

WHILE (CAN Peek())

I Dequeue()

METHODS

Peek ← Bool / Val

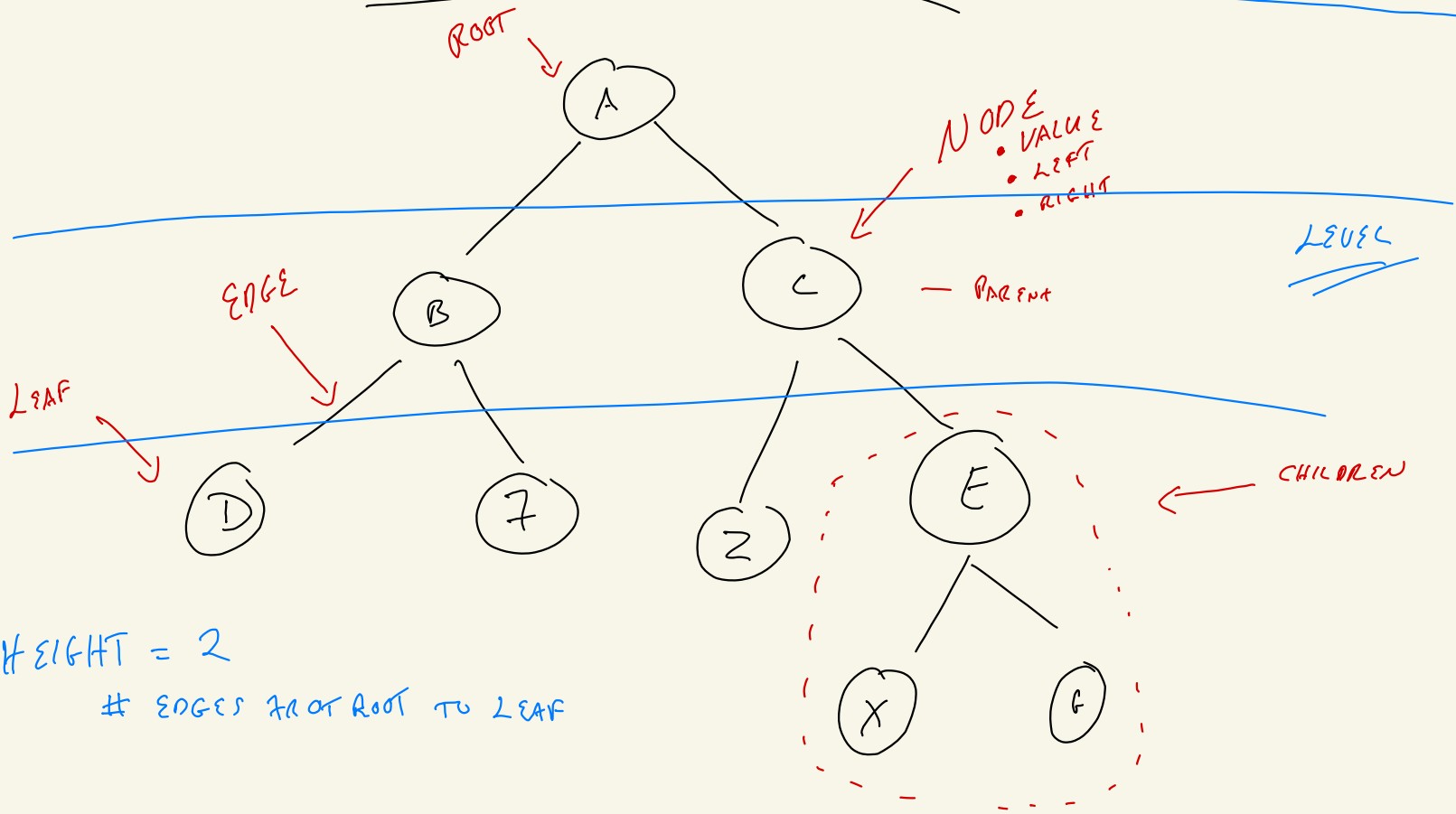
IsEmpty ← Bool

Dequeue ← Removes & Returns front

Enqueue ← puts NEW ENTRY IN BACK

TREES [BINARY]

[K-ARY] K = # OF CHILDREN



HEIGHT = 2

EDGES FROM ROOT TO LEAF

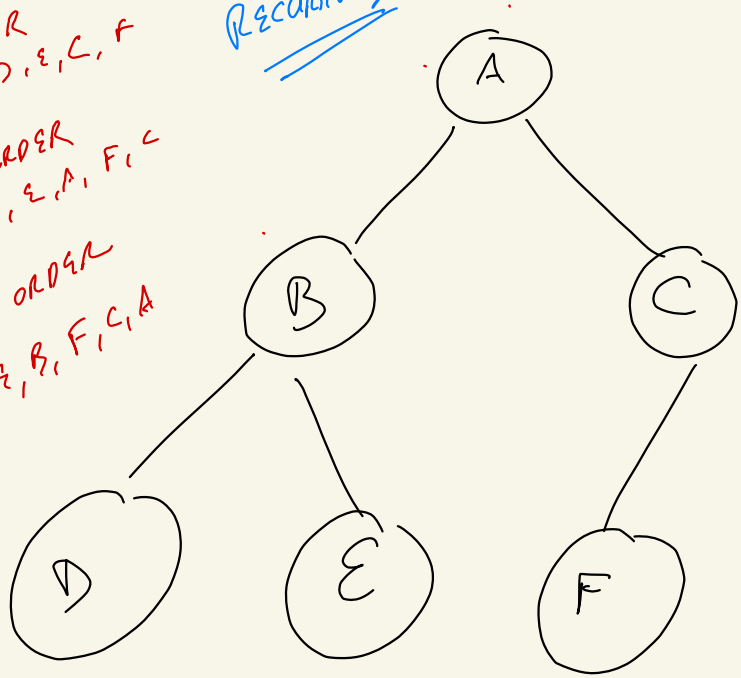
DFT

① PRE-ORDER
A, B, D, E, C, F

② IN-ORDER
D, B, E, A, F, C

③ POST-ORDER
D, E, B, F, C, A

RECURSIVELY



BIG O(H)
HEIGHT

BFT

WHILE LOOP / QUEUE

BIG O(W)
(WIDTH)

ENQUE ROOT

WHILE PEEK

ENQUE LEFT
RIGHT

DEQUE

TRAVERSE

① DEPTH FIRST

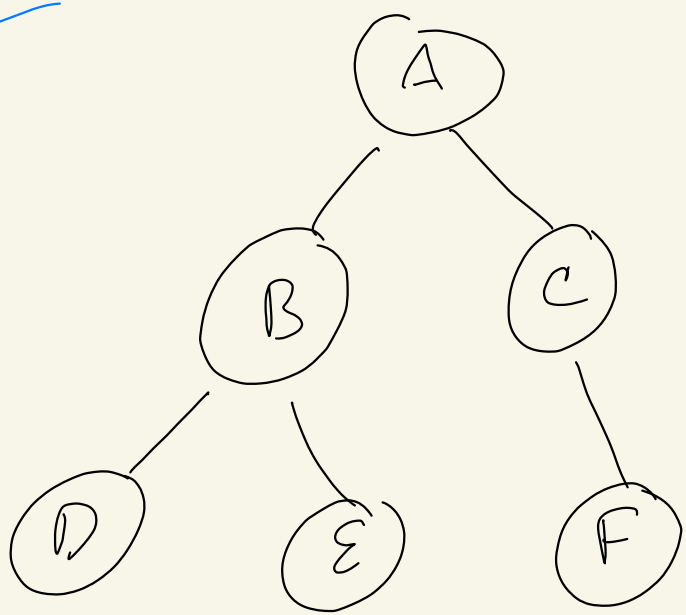
② BREADTH FIRST

SEE ALL NODES
* POTENTIALLY

A, B, D, E, C, F

A
B C
D E F

PRE ORDER



PREORDER (ROOT)

OUTPUT ROOT.VALUE ✓

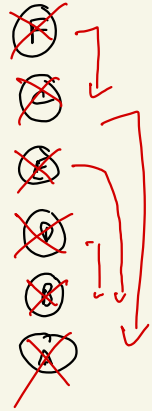
IF (ROOT.LEFT) ✓
PREORDER (ROOT.LEFT)

IF (ROOT.RIGHT)
PREORDER (ROOT.RIGHT)

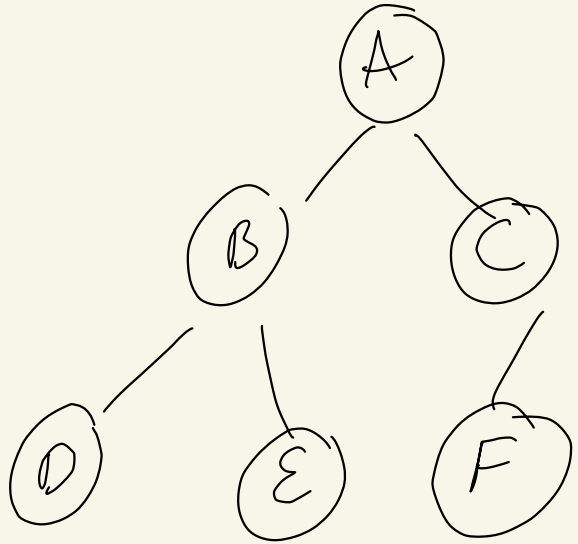
OUTPUT

A, B, D, E, C, F

CALL STACK



IN ORDER



IN ORDER (ROOT)
IF (ROOT.LEFT)
INORDER (ROOT.LEFT)

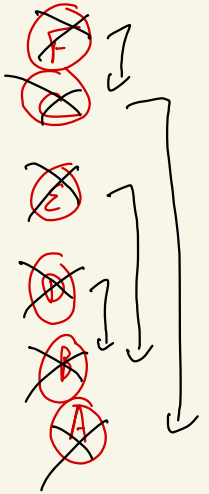
OUTPUT

IF (ROOT.RIGHT)
IN ORDER (ROOT.RIGHT)

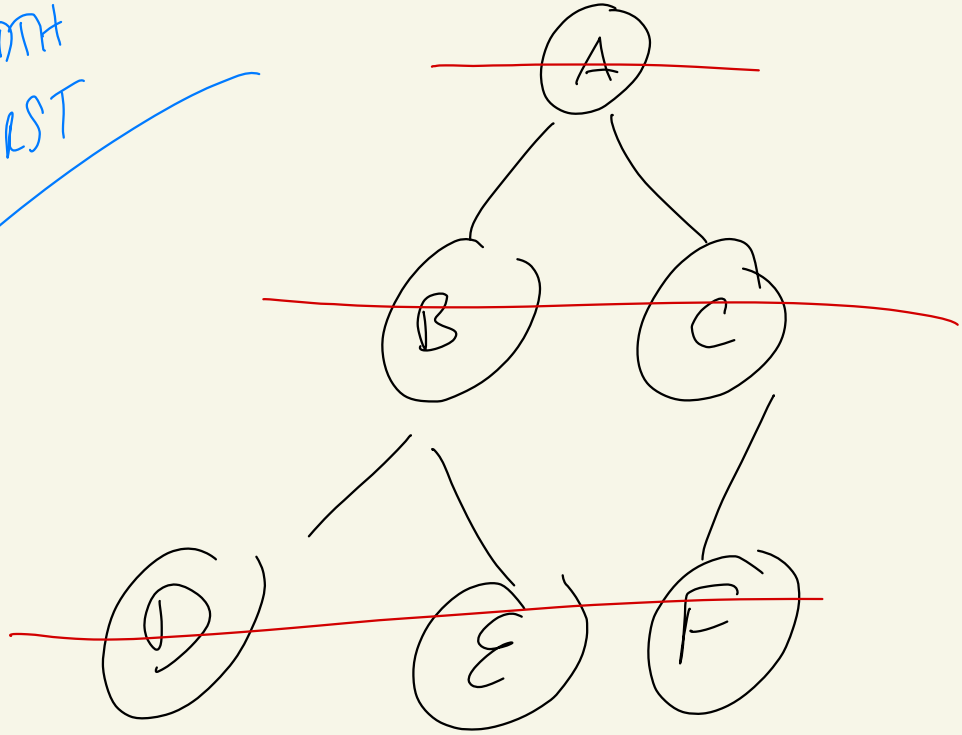
OUTPUT

D, B, E, A, F, C

CALL STACK



BREADTH FIRST



QUEUE

ENQUE (ROOT)

WHILE (Q.PEEK) {

DEQUE()

OUTPUT -

ENQUE (LEFT)

ENQUE (RIGHT)

}

OUTPUT

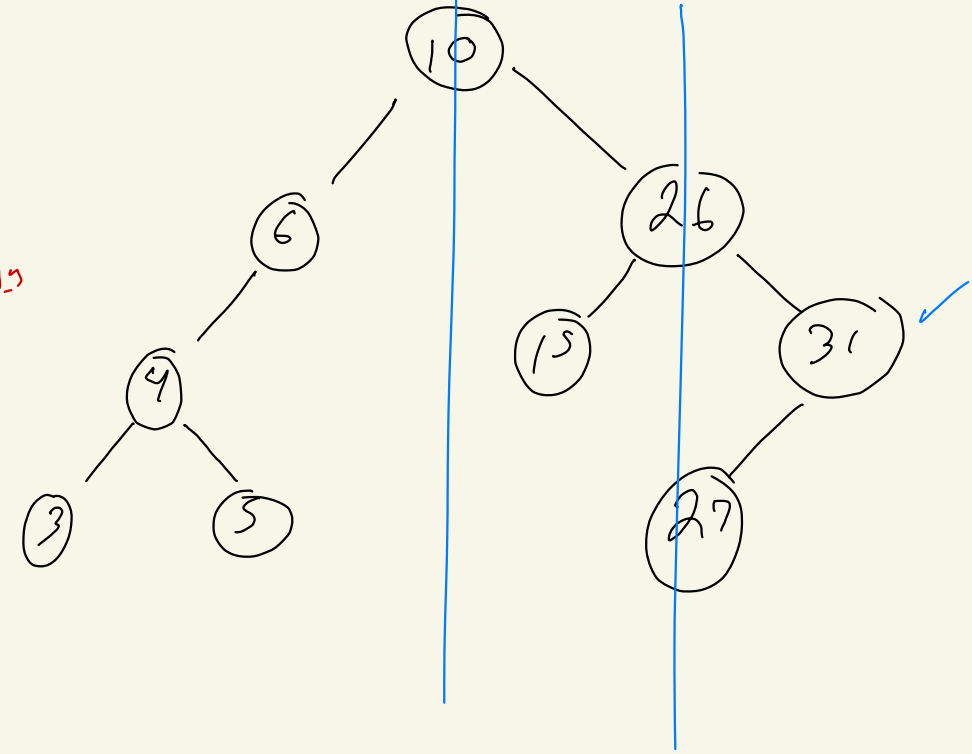
A B C D E F

Q : ~~A~~ ~~B~~ ~~C~~ ~~D~~ ~~E~~ ~~F~~

BINARY SEARCH TREE (BST)

< = LEFT

> = RIGHT



37 ?

IN-ORDER TRAVERSAL

3, 4, 5, 6, 10

BIG O (LogN)