

DATA

STRUCTURES

F	1	2	3	4	5	6	7	8	9	10	11	12	13	14
E	15	16	17	18	19	20	21	22	23	24	25	26	27	28
B	M	T	W	T	F	S	S	M	T	W	F	S	S	

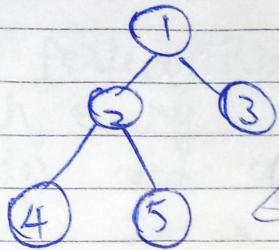
Binary Tree

FRIDAY

JANUARY

22

21 Binary Tree is a data structure in which each parent node can have atmost two children.



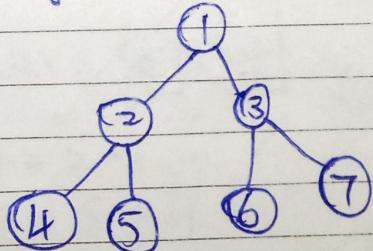
Types of Binary Tree:

(i) Full Binary Tree

In a full binary tree, every parent node or internal node has either two or no children.

(ii) Perfect Binary Tree

In a Perfect Binary Tree, every internal node has exactly two children,



and all the leaf nodes are at the same level

(iii) Complete Binary Tree

Just like a full Binary Tree, but

- Every level must be completely filled.
- All leaf elements must lean towards the left.
- The last leaf element might not have a right sibling.

23

SATURDAY

JANUARY

023-342 • WK 04

WST world

1	2	3	4	5	6	7	8	9	10	11	12	13	
14	15	16	17	18	19	20	21	22	23	24	25	26	27
28	29	30	31										

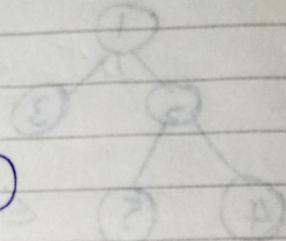
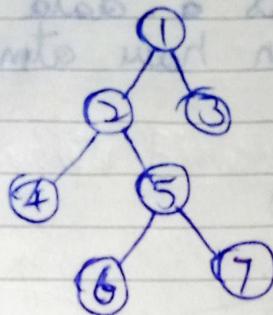
DEC
20

M	T	W	T	F	S	S	M	T	W	T	F	S
---	---	---	---	---	---	---	---	---	---	---	---	---

Full Binary Tree:

8

Either two or
no children.

Theorems

11

 $i = \text{No. of internal nodes}$
 $n = \text{total number of nodes}$
 $l = \text{No. of leaves}$
 $\lambda = \text{No. of levels}$

then,

$$2 \quad l = i + 1$$

$$n = 2i + 1$$

$$3 \quad i = \frac{n-1}{2}$$

$$l = \frac{n+i}{2}$$

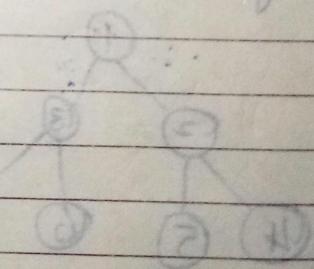
$$n = 2l - 1$$

$$i = l - 1$$

$$\max(\text{leaves}) = 2^{\lambda-1}$$

Class Node:
def __init__

24 SUNDAY

WST world. uttarakhand

Two, WST world has a lot of work.
 . but if you take a look at them level wise:
 . then it's a lot of work to do in streams for
 . like a lot less than the streams that I do. WST.

2021

F 1 2 3 4 5 6 7 8 9 10 11 12 13 14
 E 15 16 17 18 19 20 21 22 23 24 25 26 27 28

B M T W T F S S M T W T F S S

21

MONDAY

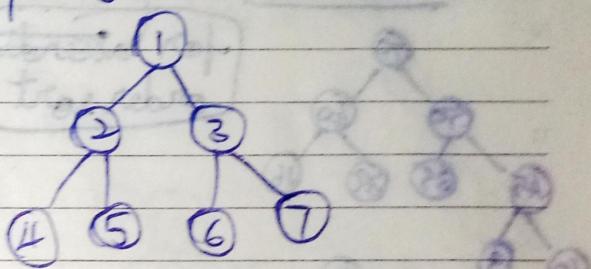
JANUARY

WK 05 • 025-340

25

Perfect Binary Tree:

A Perfect Binary tree is a type of a tree in which every internal node has exactly two child nodes and all the leaf nodes are at the same level.



Theorems

$$\text{No of nodes} = 2^{h+1} - 1$$

$$\text{no of leaf nodes} = 2^h$$

$$\text{height} = \lceil \log_2 n \rceil$$

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

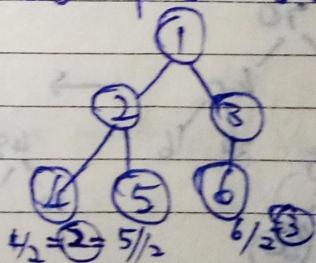
Complete Binary Tree:

A complete Binary tree is a tree in which all the levels are completely filled except possibly the deepest one which is filled from the left.

$$\text{parent index} = \left\lfloor \frac{j}{2} \right\rfloor$$

$$\text{left child} = 2 * i$$

$$\text{right child} = (2 * i) + 1$$



(REPUBLIC DAY) TUESDAY 26

M New year and new life tree (the 5th M for tree)

$\leftarrow [0, P, 0, J, PE, 2E, 2H, 04, 08, 09]$

$\leftarrow [26, P, 01, J, PE, 02, 24, 04, 07, 08]$

[Ec, P, 01, J, PE, 02, 24, 04, 07, 08]

2021

27

WEDNESDAY

JANUARY

027-338 • WK 05

1	2	3	4	5	6	7	8	9	10	11	12	13	
14	15	16	17	18	19	20	21	22	23	24	25	26	27
28	29	30	31										

DEC

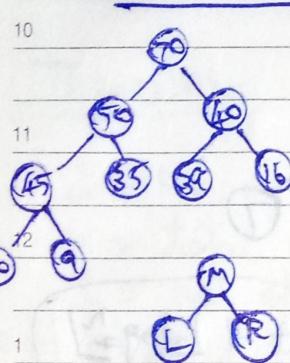
'20

Heap

It is almost a complete binary tree

Max-heap

descending
for ascending or
order sort

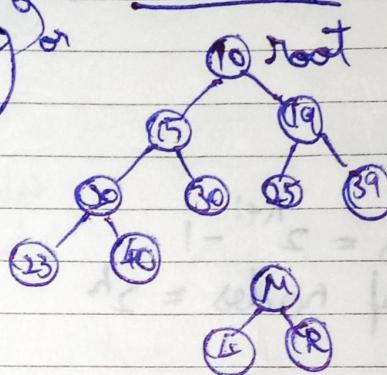


$$M \geq L \text{ or } R$$

mostly, $L > R$

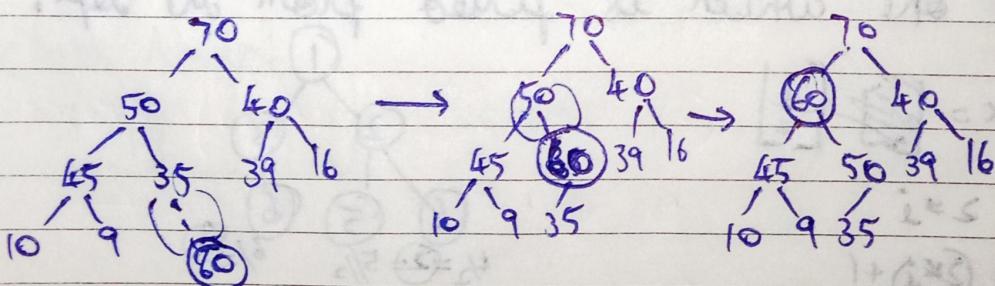
Min-heap

(a) ascending
for descending
order sort



$$M \leq L \text{ or } R$$

mostly $L < R$

Insertion:

Always start from leftmost leaf node, and
check if $M \geq L \text{ or } R$ and if ~~=~~ true swap with M.

[70, 50, 40, 45, 35, 39, 16, 10, 9, 60] \rightarrow

2021 [70, 50, 40, 45, 60, 35, 39, 16, 10, 9, 35] \rightarrow

[70, 60, 40, 45, 50, 39, 16, 10, 9, 35]

F 1 2 3 4 5 6 7 8 9 10 11 12 13 14
 E 15 16 17 18 19 20 21 22 23 24 25 26 27 28
 B M T W T F S S M T W T F S S

'21

THURSDAY

JANUARY

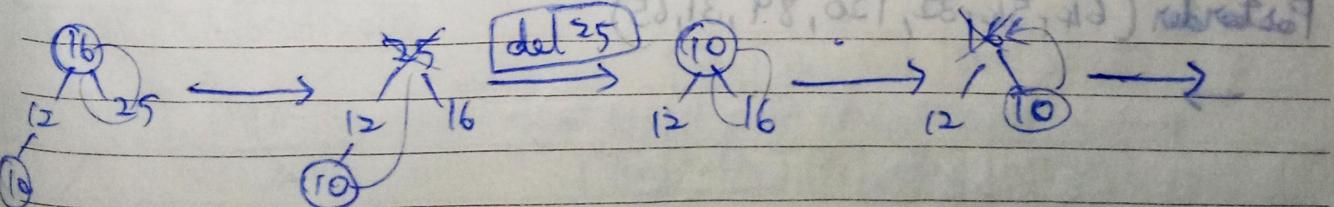
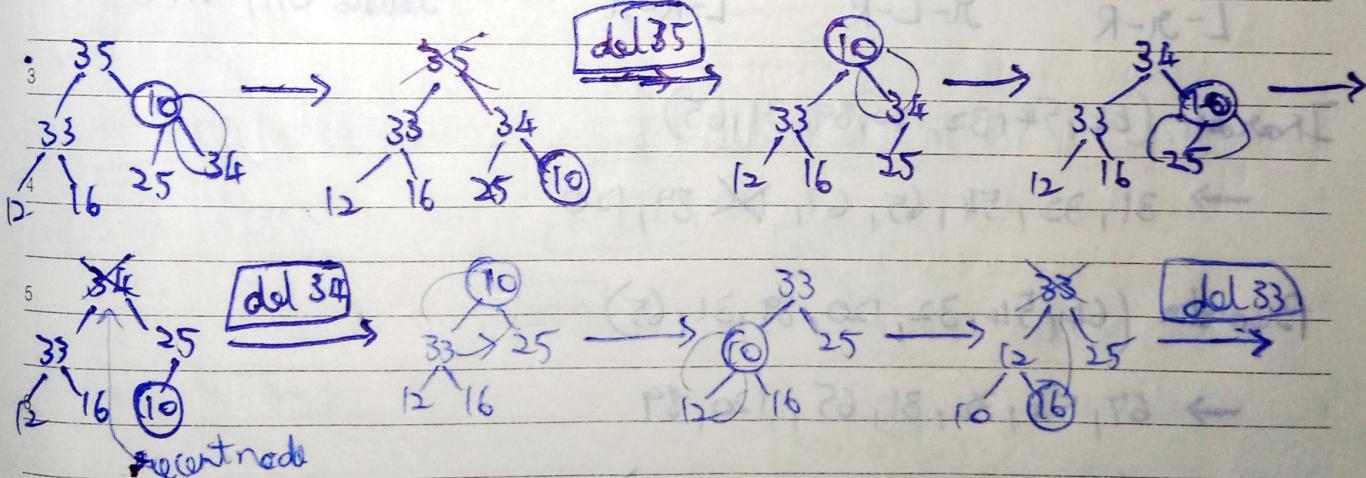
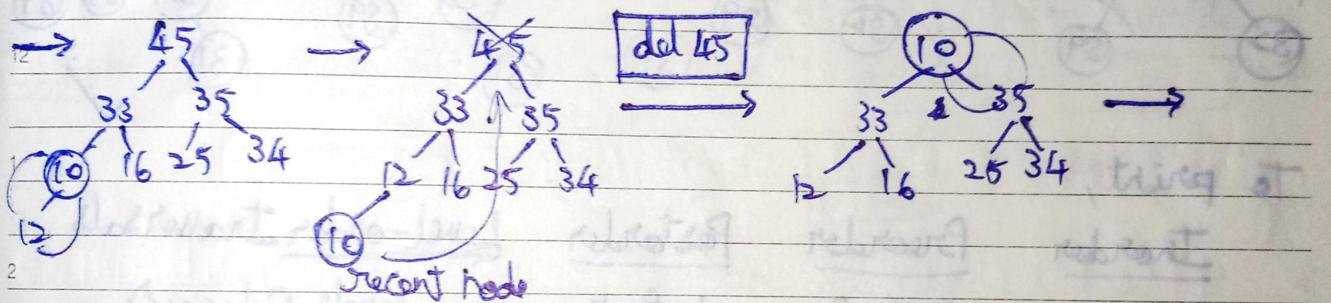
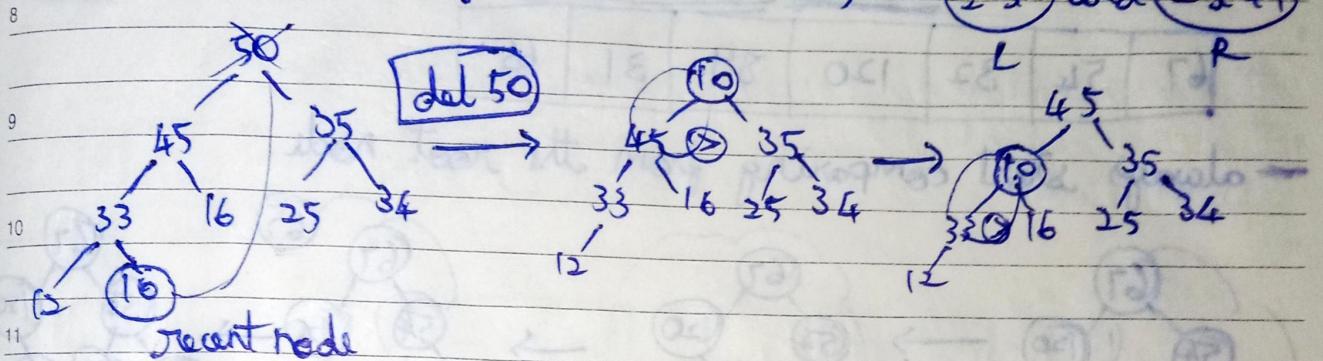
WK 05 • 028-337

28

Deletion:

(only delete root node)

$2^{*}j$ and $2^{*}j+1$



29

FRIDAY

JANUARY

029-336 • WK 05

1	2	3	4	5	6	7	8	9	10	11	12	13
14	15	16	17	18	19	20	21	22	23	24	25	26
28	29	30	31									

D

E

C

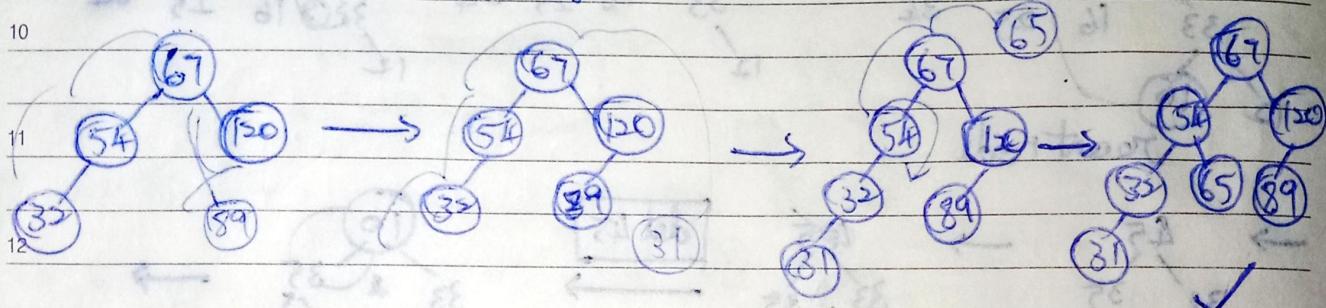
'20

Binary Search Tree

: visited

8	67	54	32	120	89	31	65
---	----	----	----	-----	----	----	----

— always start comparing from the root node



To print,

Inorder

L-R-R

Preorder

R-L-R

Postorder

L-R-R

Level-order Traversals

levels 0, 1, ..., n

Inorder (64, 54, 32, 120, 89, 31, 65)

→ 31, 32, 54, 65, 67, ~~89~~, 120

Preorder (64, 54, 32, 120, 89, 31, 65)

→ 67, 54, 32, 31, 65, 120, 89

Postorder (64, 54, 32, 120, 89, 31, 65)



F 1 2 3 4 5 6 7 8 9 10 11 12 13 14
 E 15 16 17 18 19 20 21 22 23 24 25 26 27 28
 B M T W T F S S M T W T F S S

SATURDAY

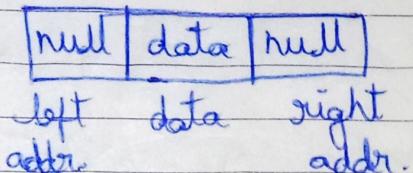
JANUARY

30

WK 05 • 030-335

'21

Node -



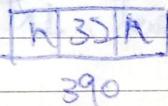
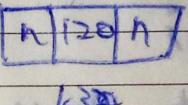
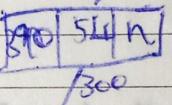
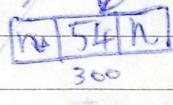
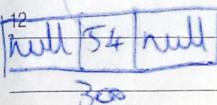
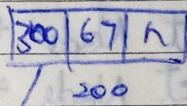
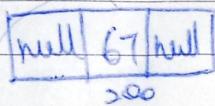
67	54	32	120	89	31	65
----	----	----	-----	----	----	----

T29 to morrow

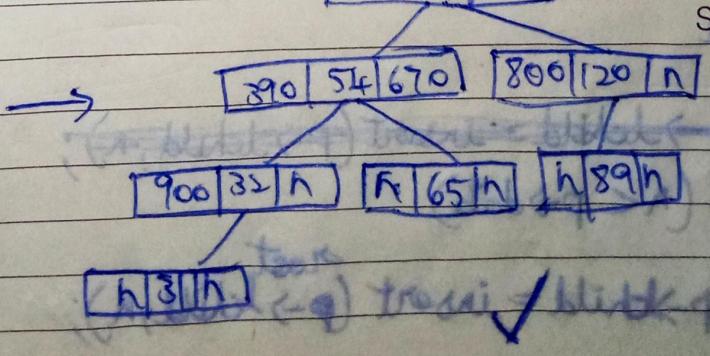
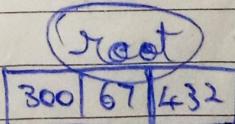
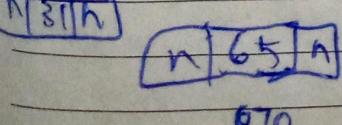
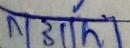
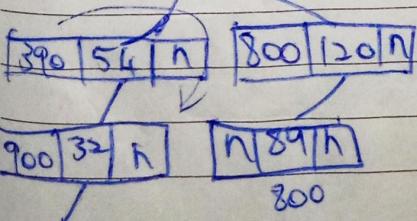
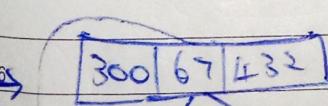
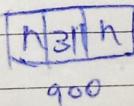
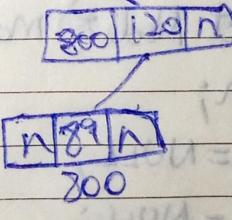
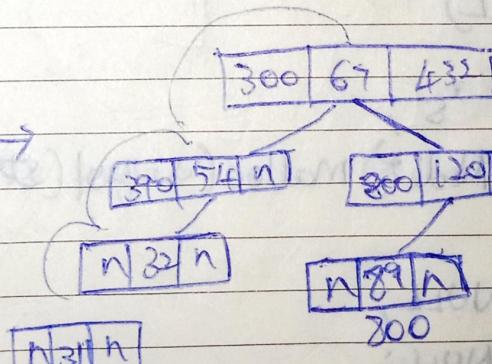
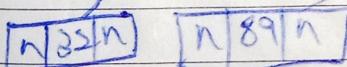
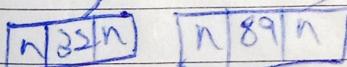
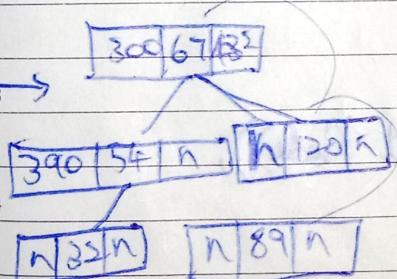
<A. diltz> abubari #

<A. diltz> abubari #

shall taught



if 67.left is null
and
54 < 67



SUNDAY 31

2021

03

WEDNESDAY
FEBRUARY

034-331 • WK 06

AVL TREE

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	J	A				
25	26	27	28	29	30	31			N

'21

AVL Tree

root changes with balancing

AVL tree is a balanced Binary Search Tree.

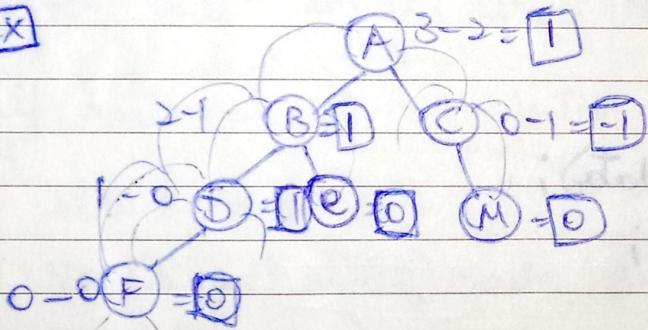
Must check if every node is balanced or not.

Find balanced factor

height of left - height of right
subtree subtree

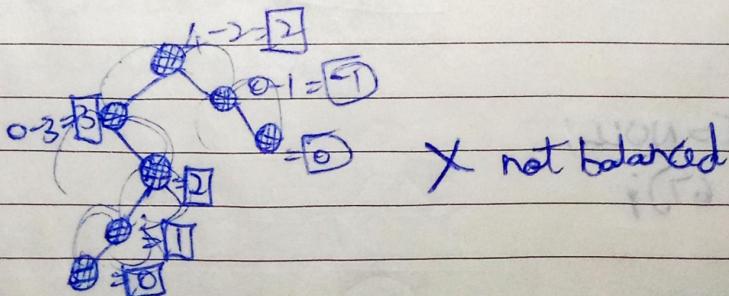
$$\boxed{BF = h_L - h_R}$$

$$\text{where } BF = [-1, 0, 1]$$

ex

balanced

hence AVL

ex

In order to balance,

LL LR RR RL rotations

M 1 2 3 4 5 6 7 8 9 10 11 12 13 14
 A 15 16 17 18 19 20 21 22 23 24 25 26 27 28
 R 29 30 31

'21

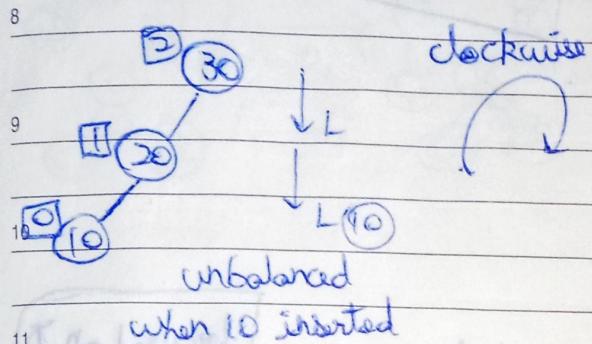
THURSDAY

FEBRUARY

WK 06 • 035-330

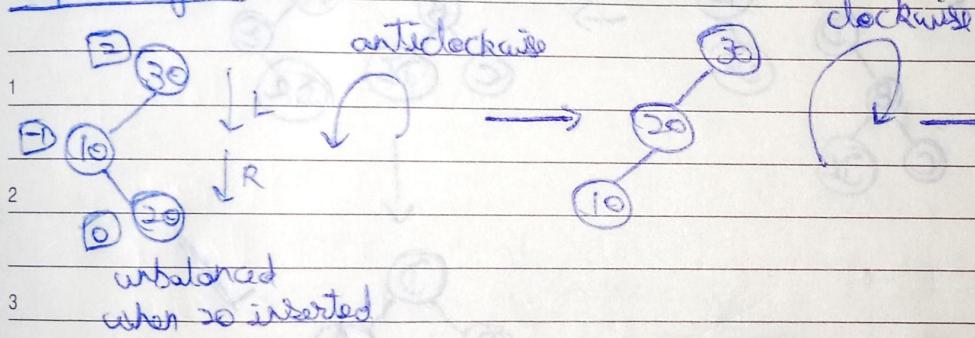
04

Left-left :

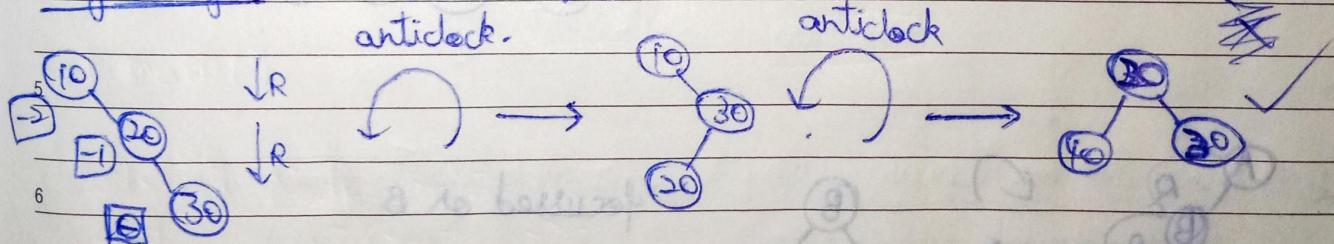


: initiatör ist stabilisiert

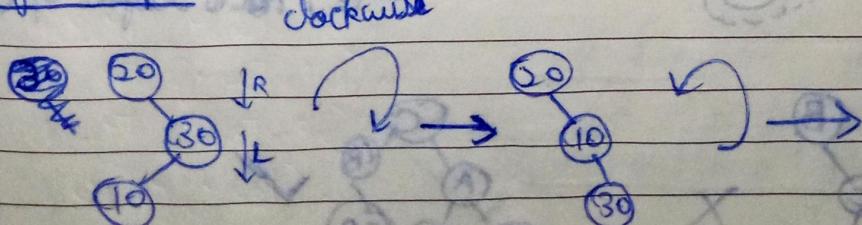
Left-Right :



Right-Right : (once right)



Right-Left :



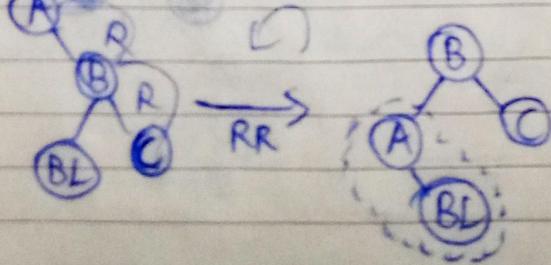
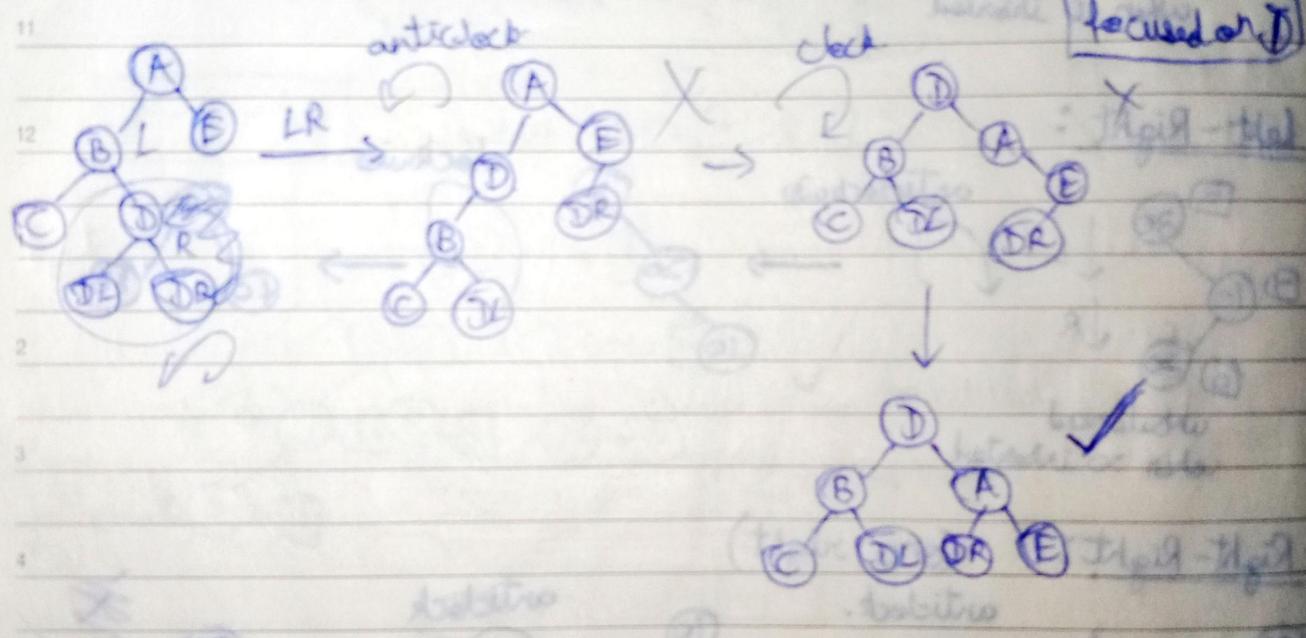
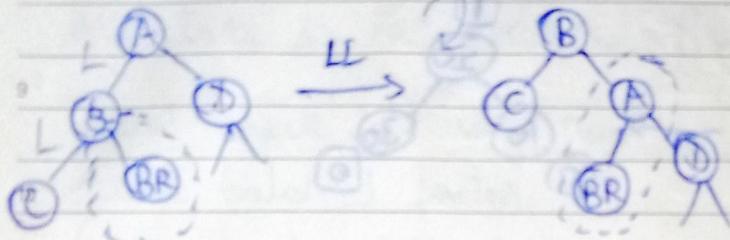
2021

05

FRIDAY
FEBRUARY
036-329 • WK 06

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
	1	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31			
	F	S	S	M	T	T	F	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S

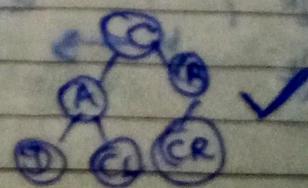
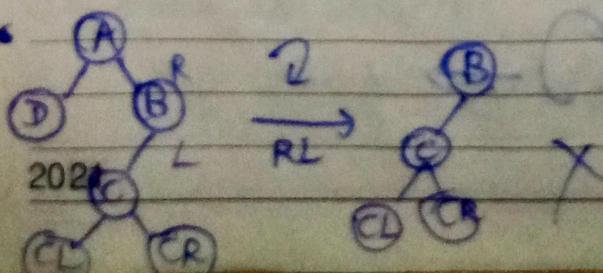
Formulate the rotation:



focused on B

always $BL < B$

here, N to right of C



M	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	15	16	17	18	19	20	21	22	23	24	25	26	27	28
R	29	30	31											
	M	T	W	T	F	S	S	M	T	W	T	F	S	S

2MHTCS2003A

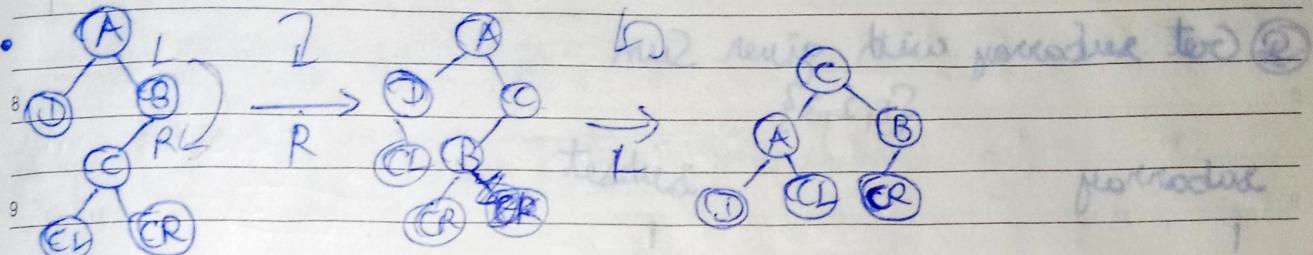
SATURDAY

FEBRUARY

06

WK 06 • 037-328

21



10

11

12

1

2

3

4

5

6

81 3 2 8 2

o-new + o-split, o-split

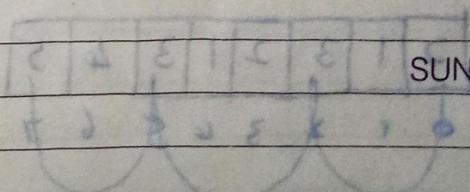
o-new = (new - old) / 2

o-split = (old - new) / 2

o-new = (new < old) fi old

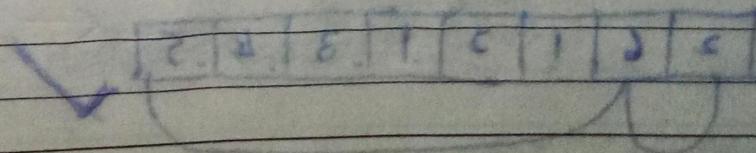
o-split = (new > old) fi old

o-new = - old



SUNDAY 07

counted, own Exam Exam



2021

A 1 2 3 4 5 6 7 8 9 10 11
 P 12 13 14 15 16 17 18 19 20 21 22 23 24 25
 R 26 27 28 29 30

M	T	W	T	F	S	S	M	T	W	T	F	S
---	---	---	---	---	---	---	---	---	---	---	---	---

HEIRHOLZER'S

-rasterized of

THURSDAY

MARCH

04

WK 10 • 063-302

'21

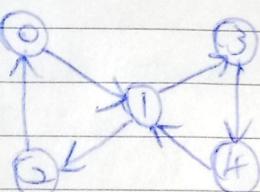
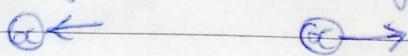
Heirholzer's Algorithm:

8

When a graph is traversed from a particular vertex and ends in the same vertex, it's called

- Must be closed

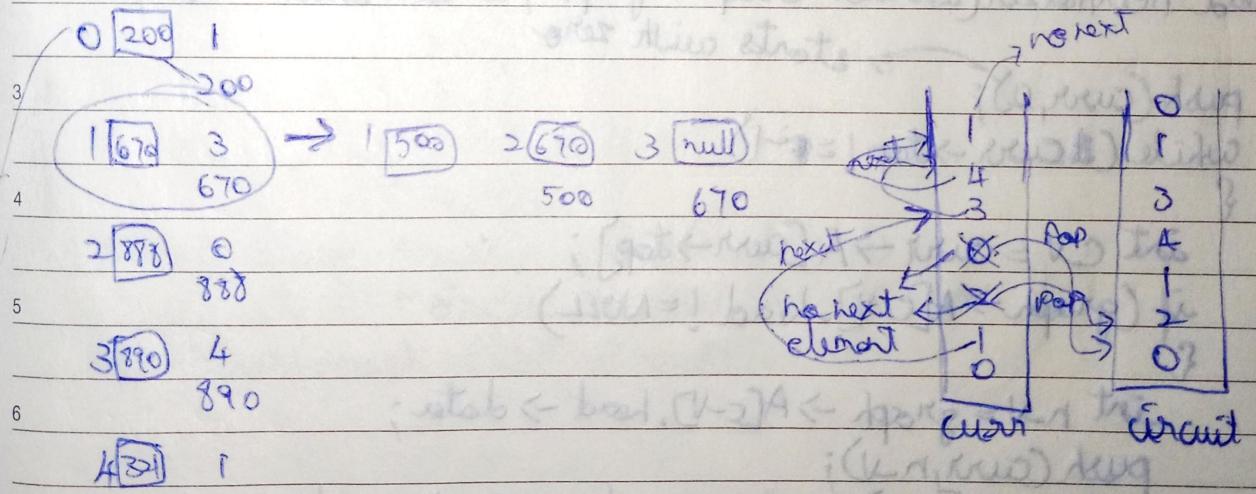
- Indegree and Outdegree must be equal.



→ 1-2-0-1-3-4-1

0-1-3-4-1-2-0

- Create graph with size $V \rightarrow 2V$ for connection



- delete the top element of curr stack in the adjacency list.

circuit = [0, 1, 3, 4, 1, 2, 0]

when starting from zero

2021

15

MONDAY

MARCH

074-291 • WK 12

SPANNING
TREES

1	2	3	4	5	6	7	8	9	10	11	12	13	14
15	16	17	18	19	20	21	22	23	24	25	26	27	28

M	T	W	T	F	S	S	M	T	W	T	F	S
---	---	---	---	---	---	---	---	---	---	---	---	---

Spanning Trees:

8

Tree representation of a graph.

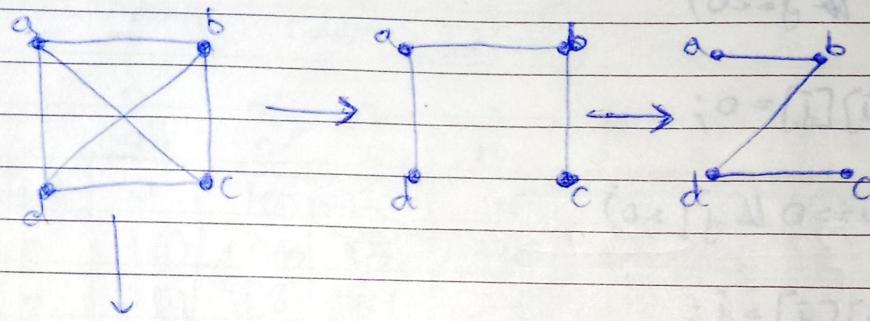
9

• All the vertices must be present.

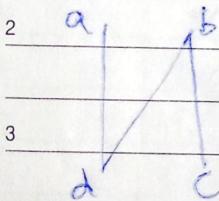
10

• Minimum edges in the tree (vertices - 1)

11

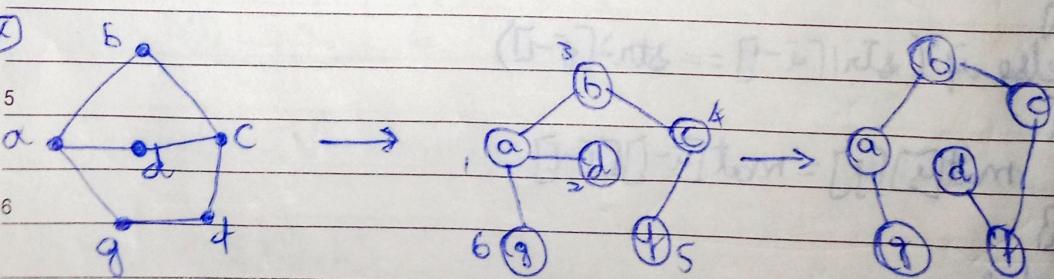


12



4

(ex)



A	1	2	3	4	5	6	7	8	9	10	11			
P	12	13	14	15	16	17	18	19	20	21	22	23	24	25
R	26	27	28	29	30									
M	T	W	T	F	S	S	M	T	W	F	S			

PRIM'S

TUESDAY

MARCH

16

WK 12 • 075-290

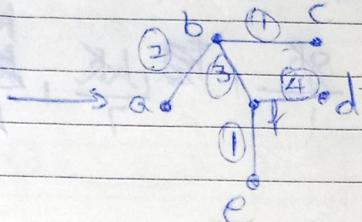
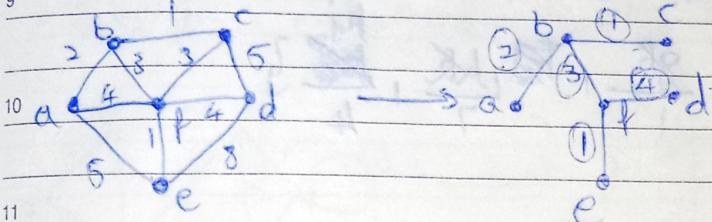
'21

Minimum Spanning Tree:

8

Each edge has its own weight

9



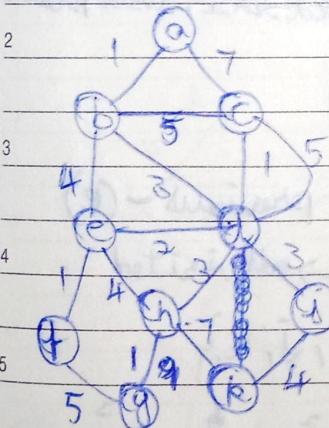
10

start from minimum weight

11

But for efficient and bigger graphs,

use algorithms.



6 Prim's Algorithm: (minimum spanning tree)

- Remove loops and parallel edges (keep minimum weight)
 - (C @ 1 and 8)

- While adding new edge, select edge with minimum weight (no cycles)
- Stop at n-1 edge.

2021

17

WEDNESDAY

MARCH

076-289 • WK 12

1	2	3	4	5	6	7	8	9	10	11	12	13	14
15	16	17	18	19	20	21	22	23	24	25	26	27	28

M	T	W	T	F	S	S	M	T	W	T	F	S
---	---	---	---	---	---	---	---	---	---	---	---	---

21												
----	--	--	--	--	--	--	--	--	--	--	--	--

closed tree
if used, we choose minimum in edges

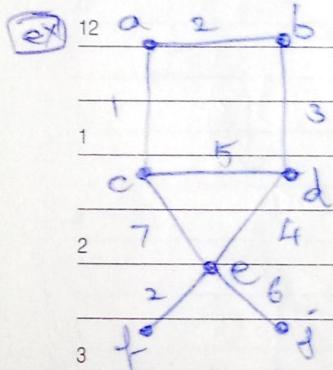
minimum

visited = {a, b, d, c, e, f, g, h, k, j}

edges = {ab, ac, bc, be, bd, ~~de~~, de, dh, di,
 1 7 5 4 3 1 2 3 3}

~~ef~~, eh, ~~fg~~, gk, ~~hk~~, lk, ~~kj~~
 1 4 5 1 7 4

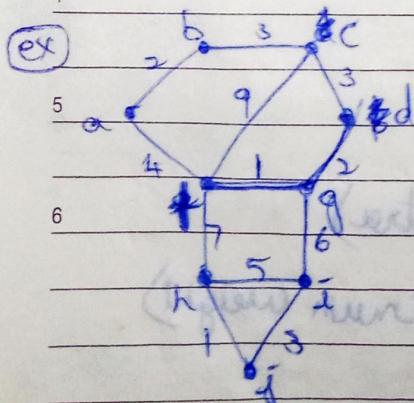
weight = 1 + 3 + 1 +



edges = {ab, ac, cd, ee, ~~dd~~, de, ef, gf,
 2 1 5 7 3 4 2 6}

visited = {a, c, b, d, e, f, g}

all visited, go to previous -(e)



visited = {f, i, g, d, e, b, a, j, h}

edges = {fg, fc, fa, fh, gd, gf,
 1 9 4 7 1 2 6 3
cb, cf, bd, ~~gh~~, ~~gi~~, ~~jh~~, ~~ji~~, ~~ik~~,
 3 9 2 5 4 3 1}

they're minimum thus ends, jobs were primitive didn't
 (ab) (aj) (ai)

A	1	2	3	4	5	6	7	8	9	10	11			
P	12	13	14	15	16	17	18	19	20	21	22	23	24	25
R	26	27	28	29	30									
M	T	W	T	F	S	S	M	T	W	T	F	S	S	

KRUSKAL'S

THURSDAY

MARCH

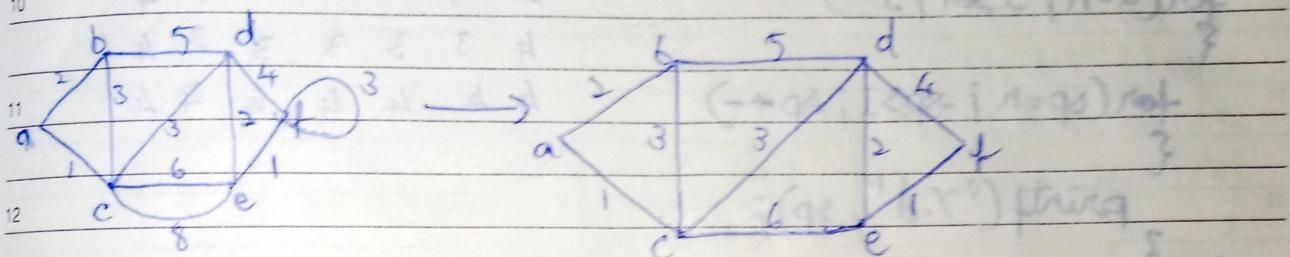
18

21

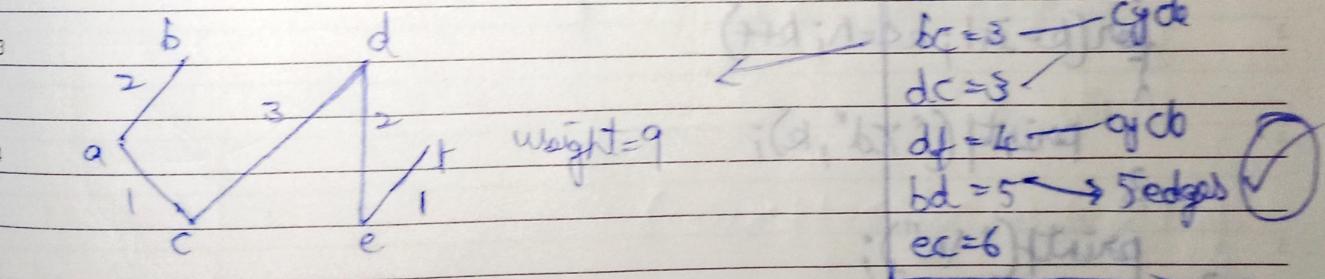
WK 12 • 077-238

Kruskal's Algorithm: (minimum spanning tree) (Union tree)

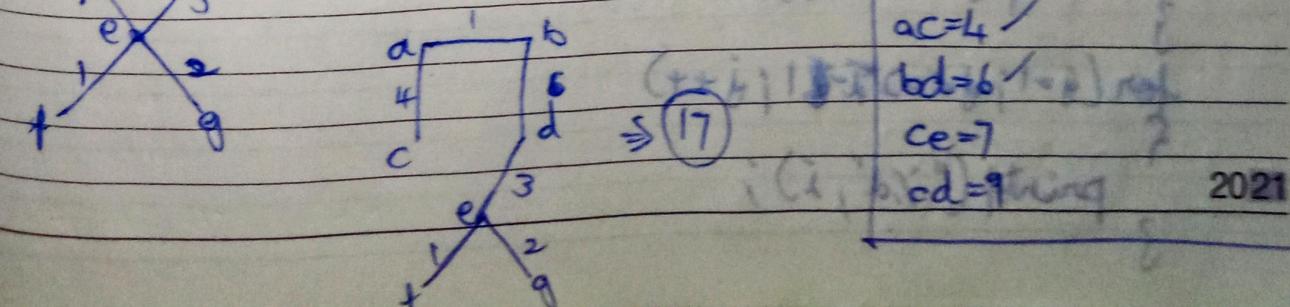
- Remove loops and parallel edges (keep min. weight)
- List all edges and sort according to weight.
- Take $(n-1)$ edges from sorted list



$$\begin{array}{llll} ab = 2 & bc = 3 & dc = 3 & ef = 1 \\ ac = 1 & bd = 5 & de = 2 & ec = 6 \\ df = 4 & & & \end{array} \xrightarrow{\text{sort with weight}} \boxed{\begin{array}{l} ac = 1 \\ ef = 1 \\ ab = 2 \\ de = 2 \\ bc = 3 \\ dc = 3 \\ df = 4 \\ bd = 5 \\ ce = 6 \end{array}}$$



$$\textcircled{1} \quad \begin{array}{llll} ab = 1 & bd = 6 & de = 3 & ab = 1 \\ ac = 4 & cd = 9 & ef = 7 & ef = 1 \\ ce = 7 & eg = 2 & eg = 2 & eg = 2 \\ & & de = 3 & de = 3 \\ & & ac = 4 & ac = 4 \\ & & bd = 6 & bd = 6 \\ & & ce = 7 & ce = 7 \\ & & cd = 9 & cd = 9 \end{array}$$



2021

26

FRIDAY

MARCH

085-280 • WK 13

M-COLOURING

1	2	3	4	5	6	7	8	9	10	11	12	13	14
15	16	17	18	19	20	21	22	23	24	25	26	27	28

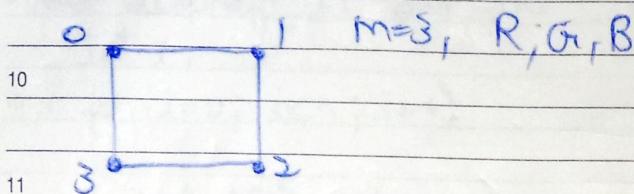
M	T	W	T	F	S	S	M	T	W	T	F	S
---	---	---	---	---	---	---	---	---	---	---	---	---

M-colouring problem:

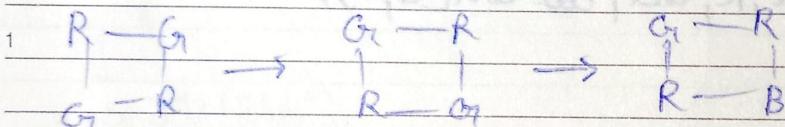
8

In a given graph, color the vertex.

9



- No same colour for adjacent vertices
- all the colors may or may not be used.



- Represent as Adjacency Matrix

	0	1	2	3
0	0	1	0	1
1	1	0	1	0
2	0	1	0	1
3	1	0	1	0

- array =

-1	-1	-1	-1
----	----	----	----

 if coloured or not

- Take a vertex, check for adjacent in the matrix

0 → 1, 3

both

1	-1
---	----

use C, or O →

1	-1	-1	-1
---	----	----	----

ind.

A	1	2	3	4	5	6	7	8	9	10	11
P	12	13	14	15	16	17	18	19	20	21	22
R	26	27	28	29	30						

21

SATURDAY

MARCH

WK 13 • 086-279

27

• $1 \rightarrow 0, 2$

 C_1

but C_1 already on 0, so use C_2 on 1 \rightarrow $\boxed{1|2|-1|-1}$

• $2 \rightarrow 1, 3$, $\boxed{1|2|1|-1}$ (from below)

 C_2

but C_2 already on 1, so use C_3 on 2 \rightarrow $\boxed{1|2|\text{ind}|1|-1}$

 C_3

• $3 \rightarrow 0, 2$

 C_1

but C_1 already on 0, 2, so use C_2 on 3 \rightarrow $\boxed{1|2|1|2} \text{ (ind)}$

 C_1 C_3

this is one solution.



SUNDAY 28

2021

M 1 2 3 4 5 6 7 8 9
 A 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 Y 24 25 26 27 28 29 30 31
 M T W T F S S M T W T F S S

21

TUESDAY

APRIL

WK 17 • 110-255

20

Binary Search Trees

	Array (unsorted)	Linked List	Array (sorted)	BST	Binary Tree
Searching	$O(n)$	$O(n)$	$O(\log n)$	$O(\log n)$	$O(n)$
Insertion	$O(1)$	$O(1)$	$O(n)$	$O(\log n)$	$O(n)$
Deletion	$O(n)$	$O(n)$	$O(n)$	$O(\log n)$	$O(n)$

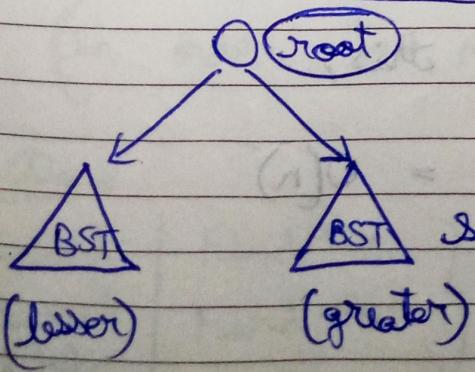
maybe..?

that's why BSTs are implemented - for lower time complexity of operations

using its sorted binary search is used.

BST conditions

$$\text{L} < \text{r} < \text{R}$$



all elements of left subtree is lesser than all elements in right subtree.

21

WEDNESDAY
APRIL

111-254 • WK 17

1	2	3	4	5	6	7	8	9	10	11	12	13	14
15	16	17	18	19	20	21	22	23	24	25	26	27	28
29	30	31										M A	21

Search, Insert, Delete

8, 10, 12, 15, 17, 20, 25

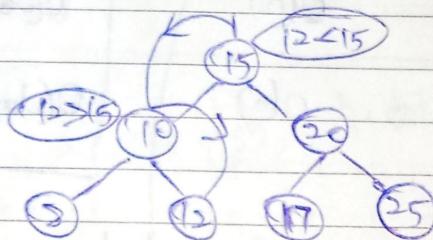
[Search for 10]

→ 8, 10, 12

$$\log_2(7) = 3 \text{ steps}$$

→ 10, ✓

→ 10 ✓

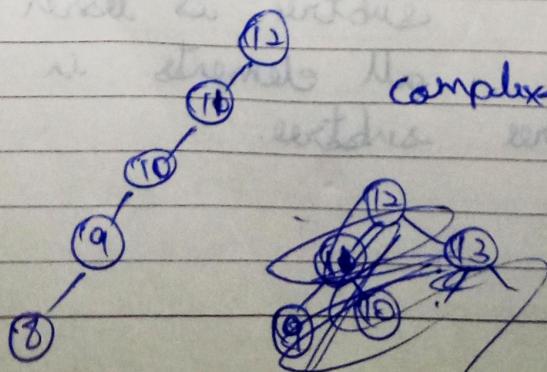


search (12)

It is a traversal where in each step we go towards left or right, in other words in each step a subtree is discarded.

- The BST is ~~not~~ always balanced in order to keep the complexity as $O(\log n)$

in case of a tree like this,



$$\text{complexity} = O(n)$$

M	1	2	3	4	5	6	7	8	9
A	10	11	12	13	14	15	16	17	18
Y	19	20	21	22	23	24	25	26	27

21

THURSDAY

APRIL

22

WK 17 • 112-253

if balanced, the complexity = $O(\log n)$

8

Always start comparing from the root node.

10

$$L \leq r < R$$

11

12

Implementation

1

- Dynamic Nodes ✓
- Array

3



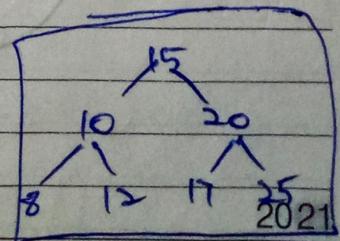
4

The main problem now is after insertion or deletion the tree may become unbalanced and we should balance the tree again

(in order to maintain $O(\log n)$)

functions

insert	FindMax	→ iterative
search	FindMin	→ recursion
delete		
print		



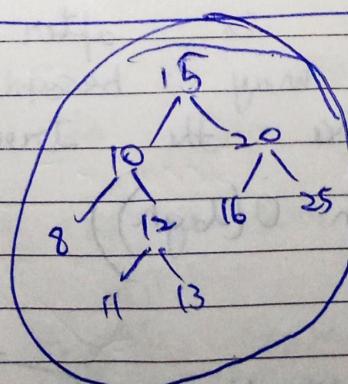
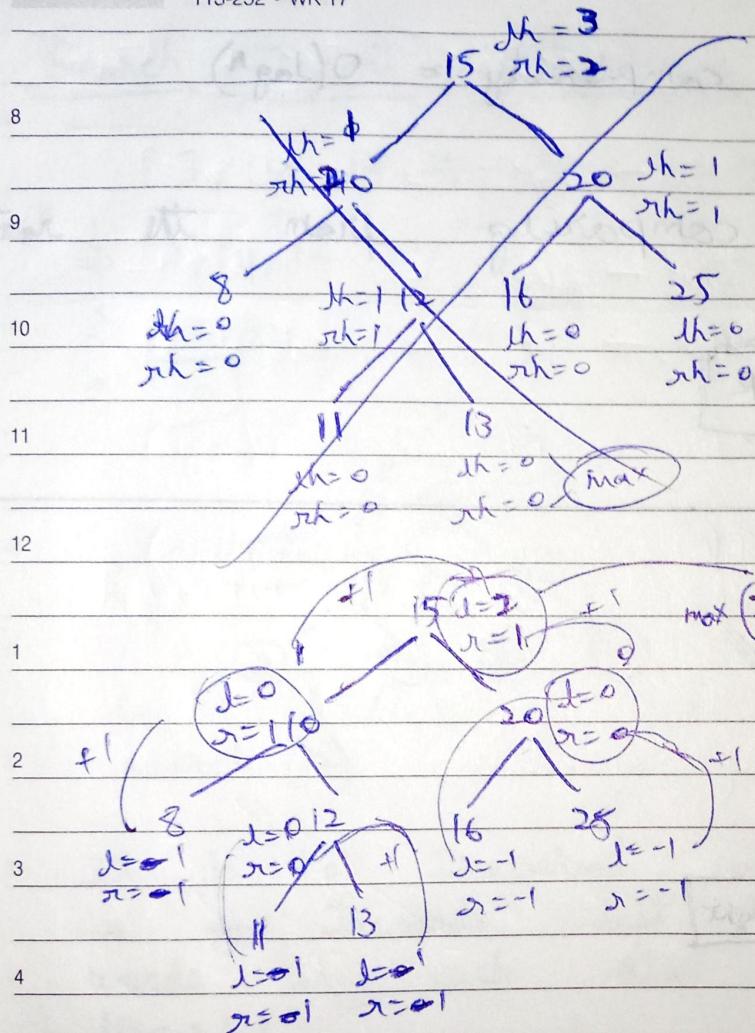
23

FRIDAY

APRIL

113-252 • WK 17

1	2	3	4	5	6	7	8	9	10	11	12	13	14
15	16	17	18	19	20	21	22	23	24	25	26	27	28
29	30	31											
M	T	W	T	F	S	S	M	T	W	T	F	S	S



M	1	2	3	4	5	6	7	8	9
A	10	11	12	13	14	15	16	17	18
Y	19	20	21	22	23	24	25	26	27

21

SATURDAY

APRIL

WK 17 • 114-251

24

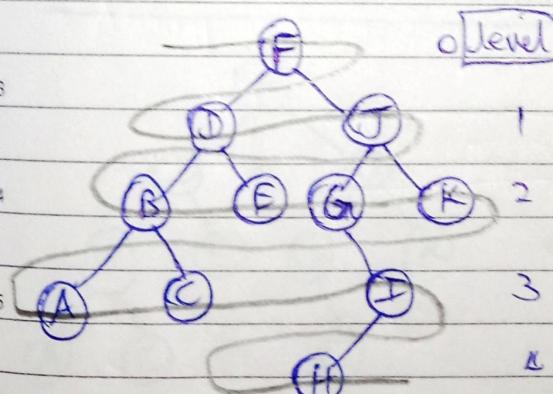
Tree Traversal

Breadth-first traversal (level order)

Depth-first traversal

Breadth-First Traversal

level order = F, D, J, B, E, G, K, A, C, I, H



0 [level]

starts from level 0, from left to right.

level - by - level

SUNDAY 25

2021

26

MONDAY

APRIL

116-249 • WK 18

1	2	3	4	5	6	7	8	9	10	11	12	13	14
15	16	17	18	19	20	21	22	23	24	25	26	27	28
29	30	31											

Depth First Traversal

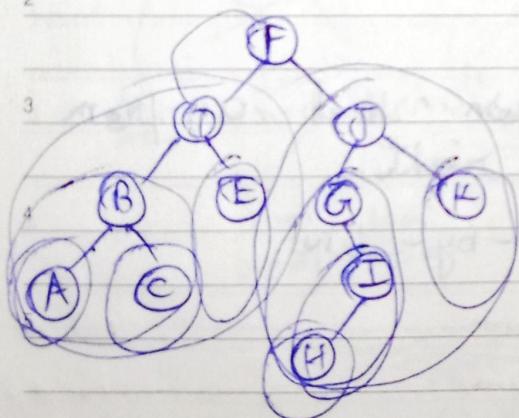
Preorder = root - left - right

Inorder = left - root - right

Postorder = left - right - root

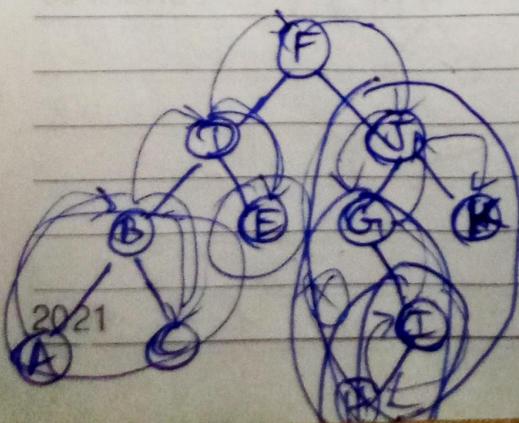
Preorder Traversal (R-L-R)

F J B A C E J G I H K



Inorder Traversal (L-R-R)

A B C D E F G H I J



M	1	2	3	4	5	6	7	8	9
A	10	11	12	13	14	15	16	17	18
Y	19	20	21	22	23	24	25	26	27

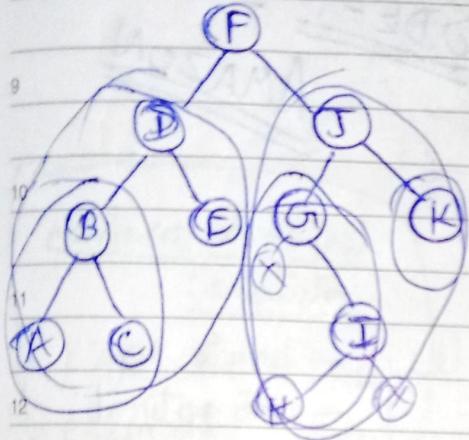
21

TUESDAY

APRIL

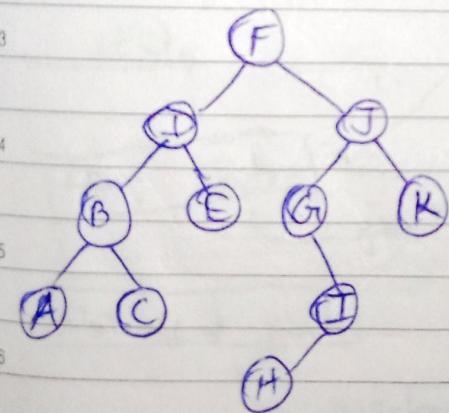
27

WK 18 • 117-248

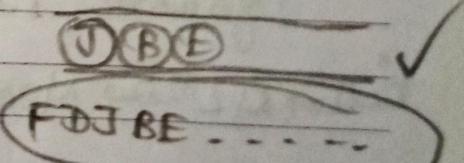
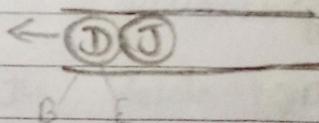
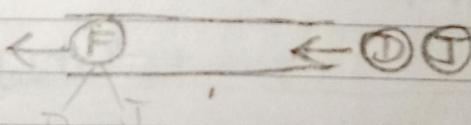
Postorder Traversal (L-R-R)

A C B E D H I G K J F

1

Level Order Traversal (Implementation)

Nodes → discovered node



A Queue is constructed where the discovered node's children are inserted. Once an element is popped out it becomes a discovered node and its children are inserted.

28

WEDNESDAY

APRIL

118-247 • WK 18

1	2	3	4	5	6	7	8	9	10	11	12	13	14
15	16	17	18	19	20	21	22	23	24	25	26	27	28
29	30	31											

MAR

21

Array ✓

Linked List ✓

Stack ✓

Queue ✓

Binary Tree ✓

BST ✓

Heap — MinHeap, MaxHeap

HashTable vs HashMap) — | dictionary in Python |

Graph (Adjacency List)

Trie (Prefix Tree)

Dynamic Programming

SQL — MongoDB

System Design — Design a Parking lot

Design an Online Book Store

Open Source Projects

FOR SDE-1 IN
AMAZON

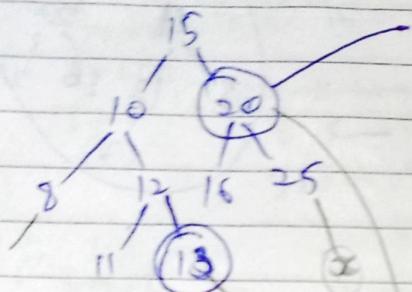
Leetcode problem

solutions:

- brute force
- sub optimal
- super optimal

M	1	2	3	4	5	6	7	8	9
A	10	11	12	13	14	15	16	17	18
Y	19	20	21	22	23	24	25	26	27

21

Delete:

$$\text{root} = 20$$

$\text{temp} = \min(\text{val} (\text{root.right}))$

$$= 25$$

$$\text{root} = 25$$

$\text{root.right} = \text{delNode}(\text{None}, 25)$

Deletion will pick and remove ~~only~~ the element and the rest will be printed in the same order.

20 30 40 50 60 70 80

del(20) → 30 40 50 60 70 80

del(50) → 30 40 60 70 80

Leaf Node just delete it

Not Leaf Node

→ replace the delete position with smallest element in the right subtree.

And delete that element (same way)

Inorder Successor

if the smallest element is it self or none

return

29

THURSDAY

APRIL

WK 18 • 119-246

2021

30

FRIDAY

APRIL

120-245 • WK 18

1	2	3	4	5	6	7	8	9	10	11	12	13	14
15	16	17	18	19	20	21	22	23	24	25	26	27	28
29	30	31											

8

9

10

11

12

1

2

3

4

5

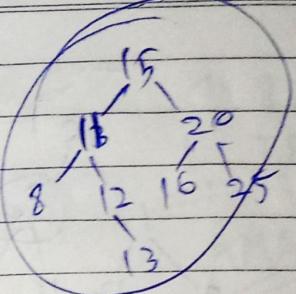
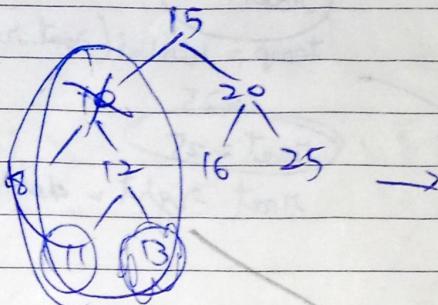
6

7

8

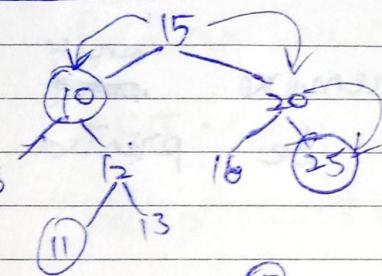
9

2021



I delete 10

Smallest element in right
subtree

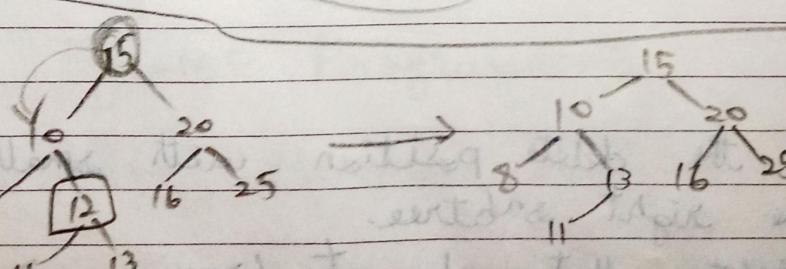


root = delete (root, 25)

↳ rootNode.right = self.delete (rootNode.right, 25)

↳ rootNode.right = self.delete (25, 25)

↳ return None



(1) key < root.val (15) → delete (10, 12)

(2) key > root.val (10) → delete (12, 12)

(3) key = root.val (12) → succPar = 12
succ = 13 ✓

succPar = rootNode

succPar.right
= succ.right

J	1	2	3	4	5	6	7	8	9	10	11	12	13	
U	14	15	16	17	18	19	20	21	22	23	24	25	26	27
N	28	29	30											

SATURDAY

MAY

01

WK 18 • 121-244

21

(15)

(10)

(L, R)

Inorder = 8 10 11 12 13 15 16 20 25

8

(12)

(16)

Preorder = 15 10 8 12 11 13 20 16 25

9

(11)

(13)

(LR)

Postorder = 8 11 13 12 10 16 25 20 15 ✓

10

Inorder (15)

Inorder (15, left)

print (15)

Inorder (15, right)

Inorder (10)

Inorder (10, left)

print (10)

Inorder (10, right)

Inorder ()

Post Case

if (root is None):
return

12

A

B

C

D

E

blue remove (4)

def remove(self, root, key)

node 24 root

2

Binary Treedeletion

4

5

6

SUNDAY 02

2021