



Mohammad Ali Jinnah University
Chartered by Government of Sindh - Recognized by
HEC

FINAL ASSIGNMENT

Name: Muhamad Fahad

Id: FA19-BSSE-0014

Subject:

Section: AM

Teacher: Awais Ahmed

Date: Saturday, January 23, 2021

FINAL ASSIGNMENT

Dear Students,

Take your own two examples (input size of $n \geq 8$) for each of the topics we covered and show proper working (dry run).

Take pictures and make a single pdf file

1.) Asymptotic Notation

Code:

```
int i;  
for (int j=0; j<i; j++) {  
    for (k=0; k<i; k++) {  
        for (int l=0; l<i; l++) {  
            k++;  
            i--;  
        }  
    }  
}
```

Output:

The time complexity of the algorithm is (n for the first loop $\times n$ (for the first inner loop) $\times n$ (for last loop))

$$n \times n \times n \Rightarrow n^3$$

Best case: n^3

Avg case n^3

Worst case: n^3

Example 2:

Code:

```
int i=n;  
for (int a=i; a<i*i; a++) {  
    while (a%2==1) {  
        a++  
    }  
}
```

Output:

- first loop will run n time
 - second while loop will run's $n/2$ time
- so,

$$n \times (n/2) = n^2/2 \Rightarrow n^2$$

Best case: n times.

Worst Case: n^2 times.

2.) Searching:

A.) Linear Search:

Code:

```
int Linear(int arr[], int key) {  
    int index = -1;  
    for (int i = 0; i < arr.length; i++) {  
        for (int j = 0; j < arr[i].length; j++) {  
            if (arr[i][j] == key) {  
                index = i;  
                return index;  
            }  
        }  
    }  
    return -1;  
}
```

Example 1:

array = {1, 8, 2, 82, 67, 32, 22, 37}

key = 37

i	arr[i]	key	condition
0	1	37	false
1	8	37	false
2	2	37	false
3	82	37	false
4	67	37	false
5	32	37	false
6	22	37	false
7	37	37	True

Time Complex

Best Case: $O(1)$

Avg Case: $O(n/2)$

Worst Case: $O(n)$

Example (output = 2)

The given array = {7, 27, 33, 52, 1, 82, 15}

And the key = 33

i	arr[i]	key	arr[i] == key	Condition
0	7	33	7 == 33	false
1	27	33	27 == 33	false
2	33	33	33 == 33	true

B) Binary Search:

Time Complexity

Best Case: $O(1)$

Avg Case: $O(\log n)$

Worst Case: $O(n)$

Example 1:

if array $\{1, 2, 3, 4, 5, 6, 7, 8\}$

Key = 6

start	end	mid	arr[mid]	key	cond
0	7	3	4	6	False
4	7	5	6	6	True

Example 2:

$A = \{37, 52, 72, 78, 79, 80, 81, 82\}$

Key = 37

start	end	mid	arr[mid]	key	cond
0	7	3	78	37	F
0	2	1	52	37	F
0	0	0	37	37	T
0	0				

REPAIRING

Time Complexity

Best Case: $O(1)$

Avg Case: $O(n^2)$

Wrong Case: $O(n)$

Example 1:

if array { 1, 2, 3, 4, 5, 6, 7, 8 }

Key = 6

Start end mid arr[mid] key cond

0 7 3 4 6 fuba

4 7 5 6 6 true

Example 2:

$$A = \{37, 52, 72, 78, 79, 80, 81, 91\}$$

Key = 37

Start end mid arr[mid] key cond

0 7 3 78 37 F

0 2 1 50 37 F

0 0 0 37 37 T

0 0 0 37 37 1

0 0

• SORTING:

•) Bubble Sort:

Time Complexity:

Best case: $O(1)$

Avg case: 01

Worst Case: $O(n^2)$

Example 1:

$$A = \{7, 77, 22, 52, 78, 5, 8, 17\}$$

$i \quad j \quad arr[i] \quad arr[j] \quad arr[i] < arr[j] \quad \text{Condition(b)}$
True

0 0 7 77 7677 True

0	2	7	52	7(52)	True
0	2	7	52	7(52)	
0	2	7	52	7(52)	

0	4	7	78	7, 78	True
0	5		5	7, 5	True

0 6 8 7.8 True

08. $\{1, 77, 22, 52, 78, 5, 9\}$ false

...

7 0 78 1 2.1 false

7 1 78 5 78 5 false

7 2 78 8 78,8 False

73 78 52 79.52 False

75 78 77 78, 77 False

76 78 78 78, 78 False

7 7 5 6 7 8 22 11 21 203

100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200, 210, 220, 230, 240, 250, 260, 270, 280, 290, 300, 310, 320, 330, 340, 350, 360, 370, 380, 390, 400, 410, 420, 430, 440, 450, 460, 470, 480, 490, 500, 510, 520, 530, 540, 550, 560, 570, 580, 590, 600, 610, 620, 630, 640, 650, 660, 670, 680, 690, 700, 710, 720, 730, 740, 750, 760, 770, 780, 790, 800, 810, 820, 830, 840, 850, 860, 870, 880, 890, 900, 910, 920, 930, 940, 950, 960, 970, 980, 990, 1000, 1010, 1020, 1030, 1040, 1050, 1060, 1070, 1080, 1090, 1100, 1110, 1120, 1130, 1140, 1150, 1160, 1170, 1180, 1190, 1200, 1210, 1220, 1230, 1240, 1250, 1260, 1270, 1280, 1290, 1300, 1310, 1320, 1330, 1340, 1350, 1360, 1370, 1380, 1390, 1400, 1410, 1420, 1430, 1440, 1450, 1460, 1470, 1480, 1490, 1500, 1510, 1520, 1530, 1540, 1550, 1560, 1570, 1580, 1590, 1600, 1610, 1620, 1630, 1640, 1650, 1660, 1670, 1680, 1690, 1700, 1710, 1720, 1730, 1740, 1750, 1760, 1770, 1780, 1790, 1800, 1810, 1820, 1830, 1840, 1850, 1860, 1870, 1880, 1890, 1900, 1910, 1920, 1930, 1940, 1950, 1960, 1970, 1980, 1990, 2000, 2010, 2020, 2030, 2040, 2050, 2060, 2070, 2080, 2090, 2100, 2110, 2120, 2130, 2140, 2150, 2160, 2170, 2180, 2190, 2200, 2210, 2220, 2230, 2240, 2250, 2260, 2270, 2280, 2290, 2300, 2310, 2320, 2330, 2340, 2350, 2360, 2370, 2380, 2390, 2400, 2410, 2420, 2430, 2440, 2450, 2460, 2470, 2480, 2490, 2500, 2510, 2520, 2530, 2540, 2550, 2560, 2570, 2580, 2590, 2600, 2610, 2620, 2630, 2640, 2650, 2660, 2670, 2680, 2690, 2700, 2710, 2720, 2730, 2740, 2750, 2760, 2770, 2780, 2790, 2800, 2810, 2820, 2830, 2840, 2850, 2860, 2870, 2880, 2890, 2900, 2910, 2920, 2930, 2940, 2950, 2960, 2970, 2980, 2990, 3000, 3010, 3020, 3030, 3040, 3050, 3060, 3070, 3080, 3090, 3100, 3110, 3120, 3130, 3140, 3150, 3160, 3170, 3180, 3190, 3200, 3210, 3220, 3230, 3240, 3250, 3260, 3270, 3280, 3290, 3300, 3310, 3320, 3330, 3340, 3350, 3360, 3370, 3380, 3390, 3400, 3410, 3420, 3430, 3440, 3450, 3460, 3470, 3480, 3490, 3500, 3510, 3520, 3530, 3540, 3550, 3560, 3570, 3580, 3590, 3600, 3610, 3620, 3630, 3640, 3650, 3660, 3670, 3680, 3690, 3700, 3710, 3720, 3730, 3740, 3750, 3760, 3770, 3780, 3790, 3800, 3810, 3820, 3830, 3840, 3850, 3860, 3870, 3880, 3890, 3900, 3910, 3920, 3930, 3940, 3950, 3960, 3970, 3980, 3990, 4000, 4010, 4020, 4030, 4040, 4050, 4060, 4070, 4080, 4090, 4100, 4110, 4120, 4130, 4140, 4150, 4160, 4170, 4180, 4190, 4200, 4210, 4220, 4230, 4240, 4250, 4260, 4270, 4280, 4290, 4300, 4310, 4320, 4330, 4340, 4350, 4360, 4370, 4380, 4390, 4400, 4410, 4420, 4430, 4440, 4450, 4460, 4470, 4480, 4490, 4500, 4510, 4520, 4530, 4540, 4550, 4560, 4570, 4580, 4590, 4600, 4610, 4620, 4630, 4640, 4650, 4660, 4670, 4680, 4690, 4700, 4710, 4720, 4730, 4740, 4750, 4760, 4770, 4780, 4790, 4800, 4810, 4820, 4830, 4840, 4850, 4860, 4870, 4880, 4890, 4900, 4910, 4920, 4930, 4940, 4950, 4960, 4970, 4980, 4990, 5000, 5010, 5020, 5030, 5040, 5050, 5060, 5070, 5080, 5090, 5100, 5110, 5120, 5130, 5140, 5150, 5160, 5170, 5180, 5190, 5200, 5210, 5220, 5230, 5240, 5250, 5260, 5270, 5280, 5290, 5300, 5310, 5320, 5330, 5340, 5350, 5360, 5370, 5380, 5390, 5400, 5410, 5420, 5430, 5440, 5450, 5460, 5470, 5480, 5490, 5500, 5510, 5520, 5530, 5540, 5550, 5560, 5570, 5580, 5590, 5600, 5610, 5620, 5630, 5640, 5650, 5660, 5670, 5680, 5690, 5700, 5710, 5720, 5730, 5740, 5750, 5760, 5770, 5780, 5790, 5800, 5810, 5820, 5830, 5840, 5850, 5860, 5870, 5880, 5890, 5900, 5910, 5920, 5930, 5940, 5950, 5960, 5970, 5980, 5990, 6000, 6010, 6020, 6030, 6040, 6050, 6060, 6070, 6080, 6090, 6100, 6110, 6120, 6130, 6140, 6150, 6160, 6170, 6180, 6190, 6200, 6210, 6220, 6230, 6240, 6250, 6260, 6270, 6280, 6290, 6300, 6310, 6320, 6330, 6340, 6350, 6360, 6370, 6380, 6390, 6400, 6410, 6420, 6430, 6440, 6450, 6460, 6470, 6480, 6490, 6500, 6510, 6520, 6530, 6540, 6550, 6560, 6570, 6580, 6590, 6600, 6610, 6620, 6630, 6640, 6650, 6660, 6670, 6680, 6690, 6700, 6710, 6720, 6730, 6740, 6750, 6760, 6770, 6780, 6790, 6800, 6810, 6820, 6830, 6840, 6850, 6860, 6870, 6880, 6890, 6900, 6910, 6920, 6930, 6940, 6950, 6960, 6970, 6980, 6990, 7000, 7010, 7020, 7030, 7040, 7050, 7060, 70

2) Insertion Sort:

Time Complexity

Best Case:

Avg Case:

Worst Case:

Example #1:~

A = {52, 77, 18, 19, 11, 44, 48, 84}

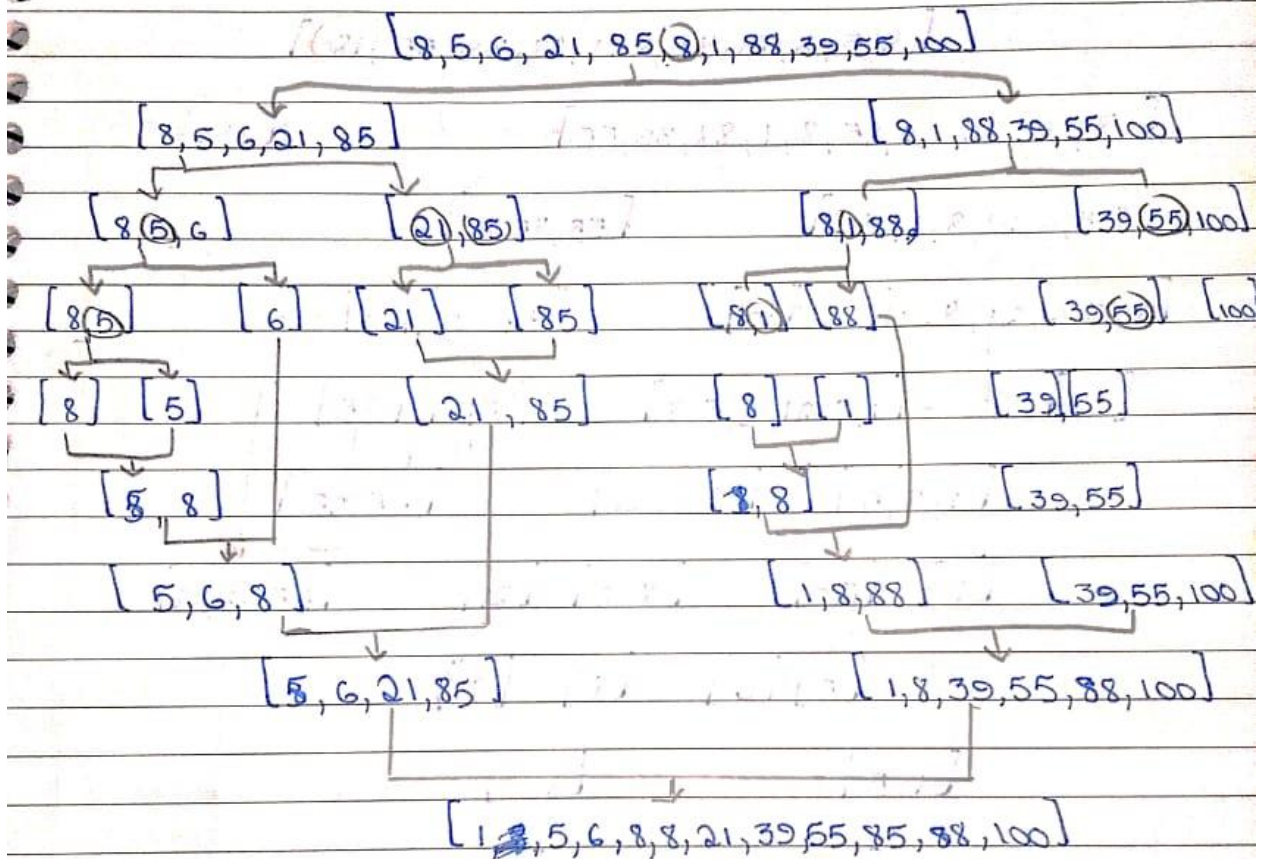
i	j	Key	arr[j]	Condition	arr
1	0	77	52	false	{52, 77}
2	1	78	77	True	{52, 77, 77}
2	0	18	52	T	{52, 52, 77}
----- i=2 {18, 52, 77} -----					
3	2	19	72	T	{18, 52, 77, 77}
3	1	19	52	T	{18, 52, 52, 77}
3	0	19	18	T	{18, 19, 52, 77}
----- i=3 {18, 19, 52, 77} -----					
4	3	11	77	T	{18, 19, 52, 77, 77}
4	2	11	52	T	{18, 19, 52, 52, 77}
4	1	11	19	T	{18, 19, 19, 52, 77}
4	0	11	18	T	{18, 11, 19, 52, 77}
----- i=4 {11, 18, 19, 52, 77} -----					
5	4	44	77	T	{11, 18, 19, 52, 77, 77}
5	3	44	52	T	{11, 18, 19, 52, 77}
----- i=5 {11, 18, 19, 44, 52, 77} -----					
6	5	48	77	T	{11, 18, 19, 44, 48, 52, 77}
6	4	48	52	T	"
6	3	48	44	F	"
----- i=6 {11, 18, 19, 44, 48, 52, 77} -----					
----- i=7 {11, 18, 19, 44, 48, 52, 77, 84} -----					

Example 2:

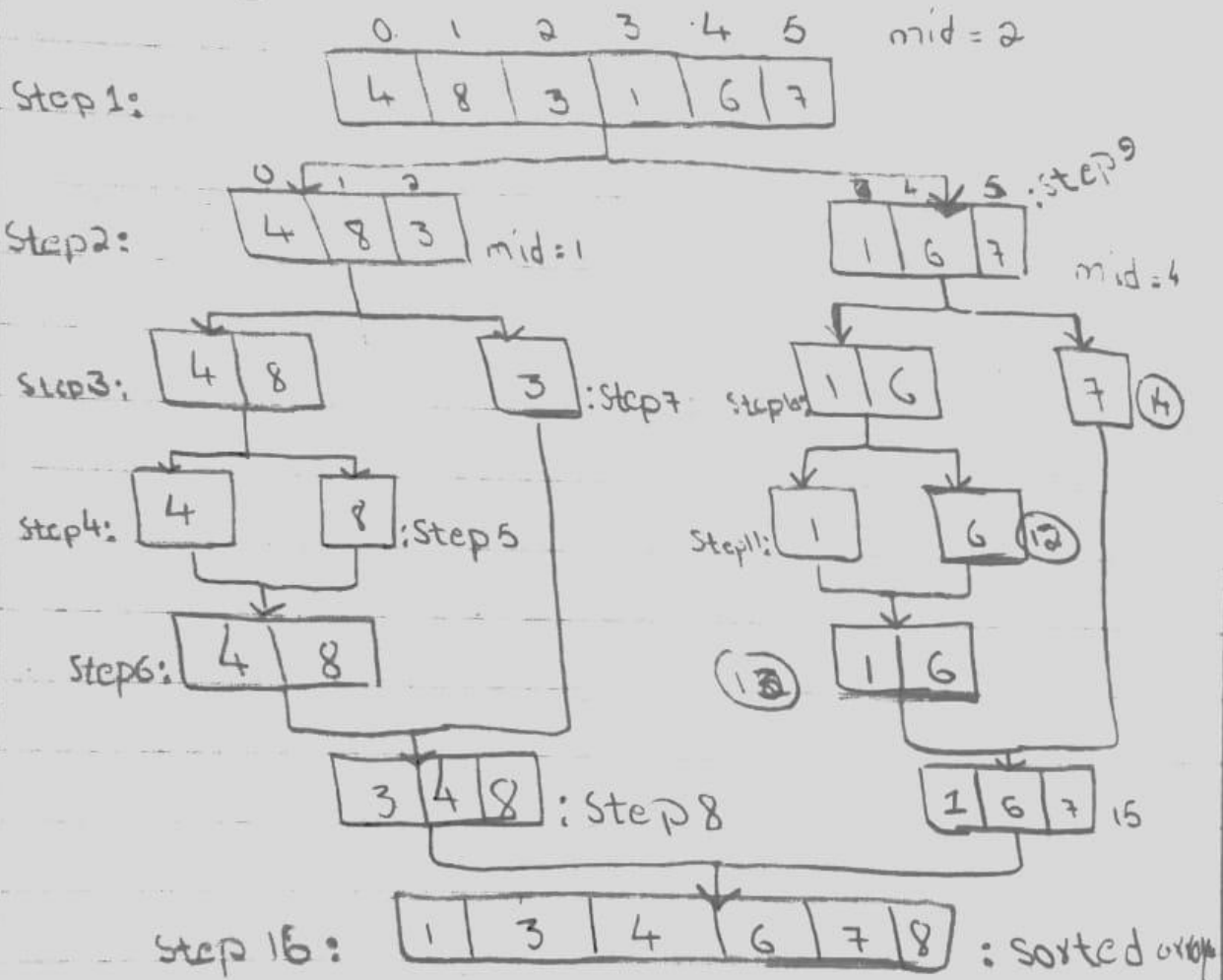
A = {1, 8, 32, 4, 55, 58, 78, 94}

i	j	Key	arr[j]	Condition	arr
2	1	8	32	false	{1, 8, 32}
3	2	32	8	false	{1, 8, 32, 32}
4	2	4	8	false	{1, 8, 8, 32}
4	0	4	1	false	{1, 4, 8, 32}
5	4	55	32	false	{1, 4, 55, 32}
6	5	58	55	false	{1, 4, 58, 55}
7	6	78	58	false	{1, 4, 78, 58}
8	7	94	78	false	{1, 4, 94, 78}

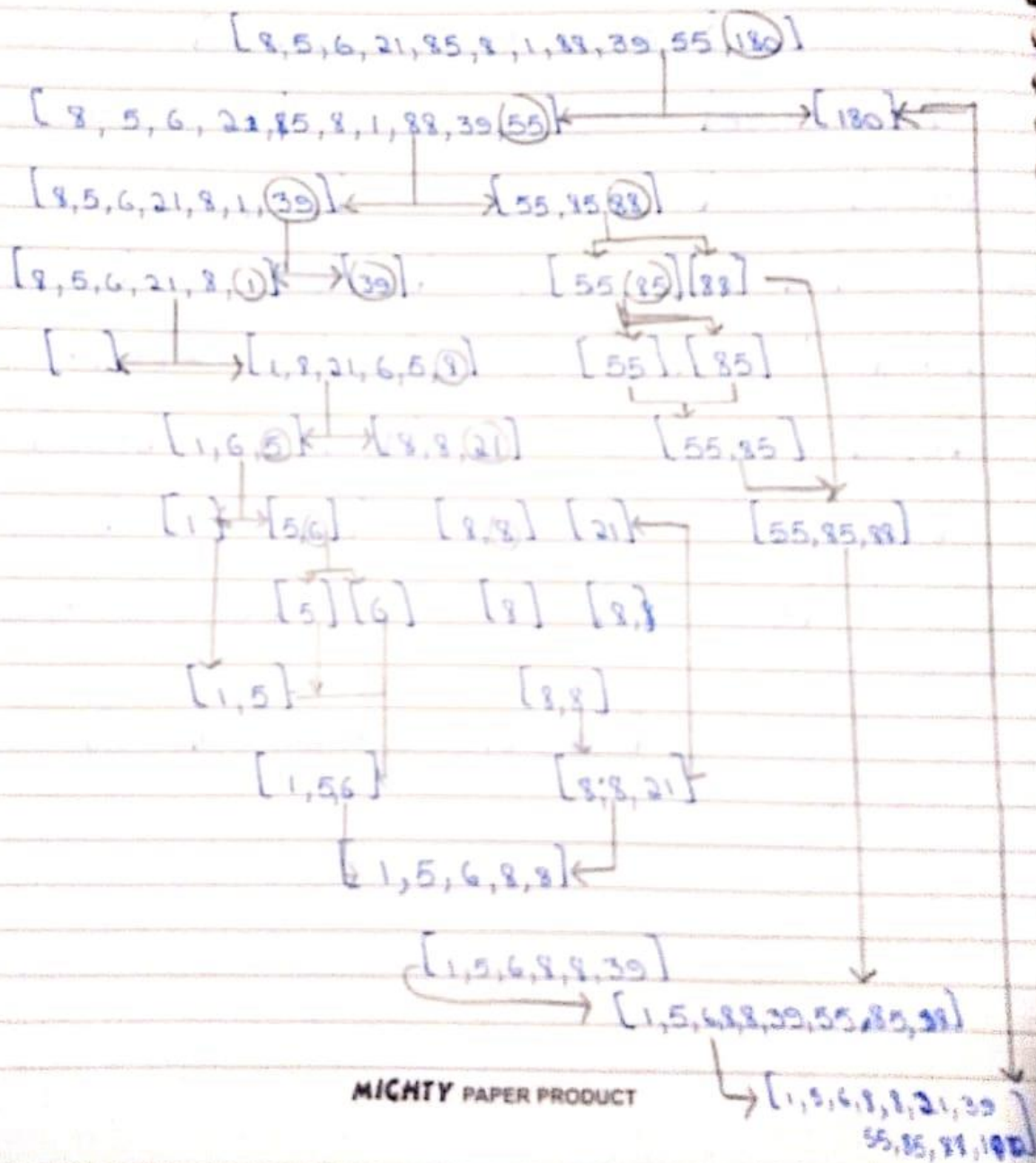
• Merge Sort:-



Merge - Sort:



Q2 Quick-Sort:



MIGHTY PAPER PRODUCT

Quick - sort:

i	Pivot	condition	status	Array
(ii)	(high)	(pivot)	(True/False)	({Array, testing})
1	7	(7, 4)	True	{4, 8, 3, 1, 6, 7}
1	7	(7, 8)	False	{4, 8, 3, 1, 6, 7}
2	7	(7, 3)	True	{4, 3, 8, 1, 6, 7}
3	7	(7, 1)	True	{4, 3, 1, 8, 6, 7}
4	7	(7, 6)	True	{4, 3, 1, 6, 8, 7}
4	7	(7, 7)	False	{4, 3, 1, 6, 7, 8}

Now we call sort of low=0, high=3;

0	1	(1, 4)	False	{4, 3, 1, 6, 7, 8}
0	1	(1, 3)	False	{4, 3, 1, 6, 7, 8}
1	1	(1, 1)	False	{1, 3, 4, 6, 7, 8}

And after that all the value/sort function return/change the index of value not found.

3) Stack

2.)

Example:-

Q1: Postfix is given $abc^*+de^*f+g^*$ we have to find infix and prefix and proved by putting constant 1 value.

Input (Postfix)	Postfix	Infix	Prefix
$abc^*+de^*f+g^*$	g	a	
$abc^*+de^*f+g^*$	g^*	ab	
$+$	g^*+f	$a(bc)$	
$+$	g^*+fed	$a+(b*c)$	
$+$	g^*+fed^*	$a+(b*c)d$	
$+$	g^*+fed^*+a	$a+(b*c)d+e$	
$+$	$g^*+fed^*+cba^*$	$(a+(b*c))(d+e)+f$	
$+$	$g^*+fed^*+cba^*+a$	$(a+(b*c))((d+e)+f)g$	
$abc^*+de^*f+g^*+a+bc^*+fed^*+d((a+(b*c))((d+e)+f))g$			

Calculate:

Put all the value of the variable 1 such as

$$\therefore a=b=c=d=e=f=g=1;$$

Infix Notation:

$$((a+(b*c))+(d+e)+f)*g \Rightarrow ((1+1)+1)+((1+1)+1)*1$$

$$((1+1)+((1+1)*1)) \Rightarrow 4$$

Postfix Notation:

$$abc^*+de^*f+g^* \Rightarrow (1)(1)(1)^*+(1)(1)^*(1)+(1)^*$$

$$= 4$$

Prefix Notation:

$$^*+abc+^*fed+^*d$$

$$= 4$$

Proved.

$$(A * B) * C * (D * E) + F$$

Soln-

$$\text{fully parenthesized: } (((a * b) * c) * ((d + e) + f))$$

Input	Postfix	Prefix
$a * b) * c * (d + e) + f$	A	f
$* b) * c * (d + e) + f$	AB	fE
$* b) * c * (d + e) + f$	AB*	FED
"	AB*	FED+
"	AB*C*	FED+C
"	AB*C*D	FED+CB
"	AB*C*DE	FED+CB*
"	AB*C*DE+	FED+CB*++
$a * b) * c * (d + e) + f$	AB*C*DE+f+	+++ABC+DEF

Calculate: ~

$$\text{Infix: } (((a * b) * c) * ((d + e) + f)) \Rightarrow ((1 * 1) * 1 * 1 * ((1 + 1) + 1)) \Rightarrow 3.$$

$$\text{Postfix: } ab * c * de + f + \Rightarrow 11 * 1 * 11 + 1 + \Rightarrow 3.$$

$$\text{Prefix: } +++ABC + DEF \Rightarrow +++ABC + DEF \Rightarrow 3.$$

4.) Queue: ~

Time Complexity:

Insert = $O(1)$

Space = $O(n)$

Delete = $O(1)$

Search = $O(n)$

Overall Case: (for all the case of insert, delete & search.)

Best Case = $O(1)$

Avg Case = $O(n)$

Worst Case = $O(n)$

Example 1:

Q. A = {1, 2, 3, 4, 5, 6} and delete two of them.

6					
6 5					
6 5 4					
6 5 4 3					
6 5 4 3 2					
6 5 4 3 2 1					
5 4 3 2 1					
4 3 2 1					

Output: "1, 2, 3, 4"

insert(5);
insert(4);
insert(3);
insert(2);
insert(1);
delete();
delete();
print();

Q. A = {B, C, D, A, F, G}

B					
C B					
D C B					
A D C B					
F A D C B					
G F A D C B					
F A D C B					

del();
Print();

Output:

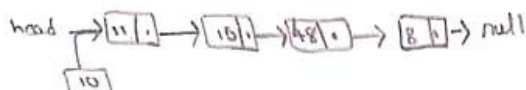
"F, A, D, C, B"

5.) Linklist:~

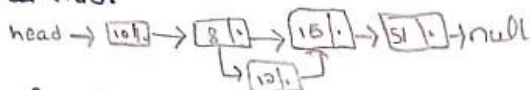
Single linked list:~

• Insertion:

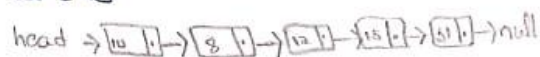
1. Insert at start:



2. Insert at mid.

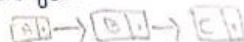


3. Insert at end



• Deletion:

Same as insert just broke the link between two



• Search:

is simple and linear search, time complexity is $O(n)$ for

n size inputs.

Double Linked list:

• Insertion: same as just linklist.

• Deletion: same as but just handle the prev point.

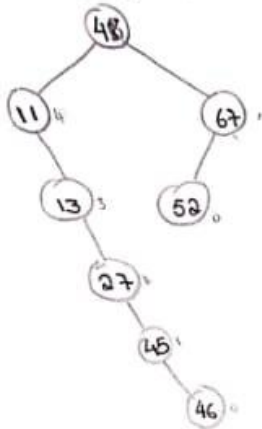
• Search: same as this case just easy to travel the list.

6. Tree:-

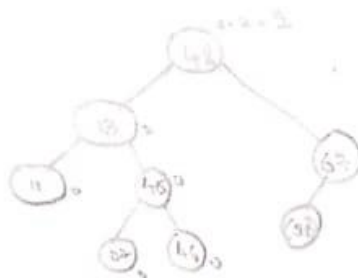
Binary Search tree:-

Example 1:

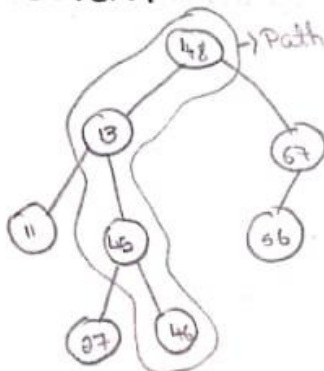
Array = {11, 13, 27, 45, 67, 46, 52}, ~~46~~ Search (46)



L-R rotation:-



Search:

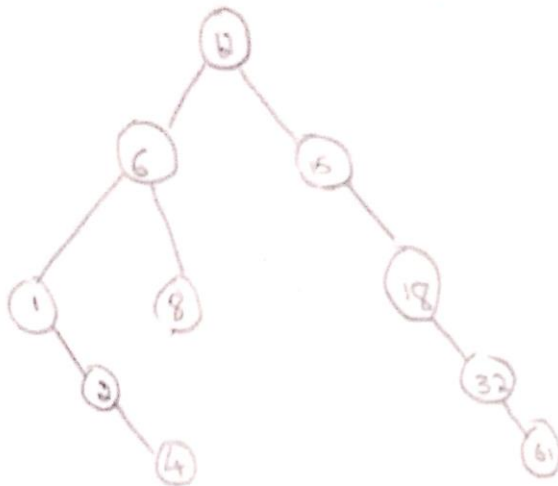


hit = 3 hit.

index = 7 index.

Example 2:

Array = {12, 6, 8, 15, 18, 32, 61, 1, 2, 4}

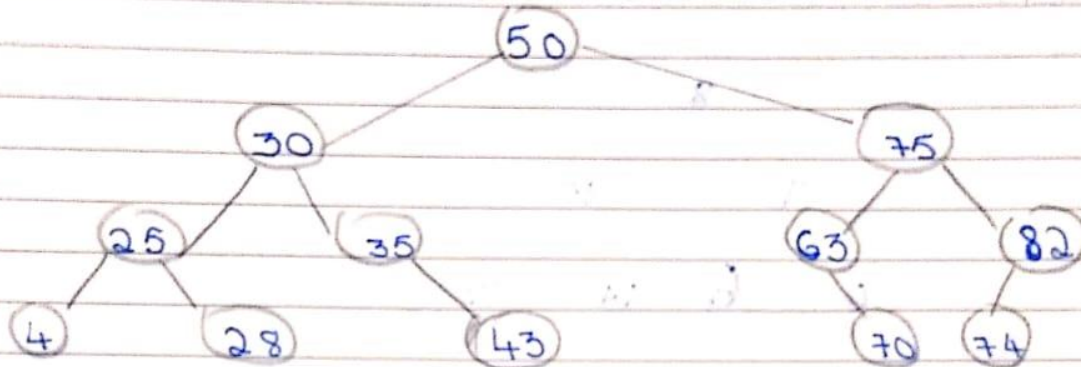


InOrder: 1, 2, 4, 6, 8, 12, 15, 18, 32, 61

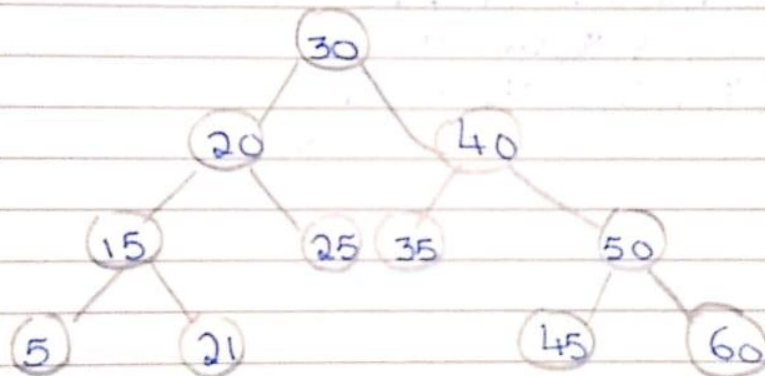
PostOrder: 4, 2, 1, 8, 6, 32, 18, 15, 12

PreOrder: 12, 6, 1, 2, 4, 8, 15, 18, 32, 61

Q₂



Q₃



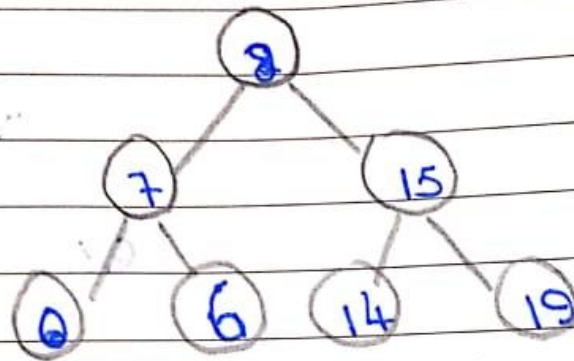
Inorder: 5, 15, 25, 20, 26, 35, 40, 45, 50, 60, 30.

Postorder: 30, 20, 15, 5, 21, 25, 40, 35, 50, 45, 60.

Preorder: 5, 25, 15, 21, 20, 40, 35, 45, 60, 50, 30.

D

Example:-



InOrder: [0, 7, 6, 8, 15, 14, 19]

PostOrder: [0, 6, 7, 14, 19, 15, 8]

PreOrder: [8, 7, 0, 6, 15, 14, 19]

Ans

4.) AVL tree:

Example 2:

Array = {1, 2, 3, 4, 5, 6, 7}

Step 1 (insert 1)



Step 2 (insert 4) and balance



Unbalance

(RR rotation)

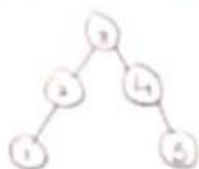


Unbalance

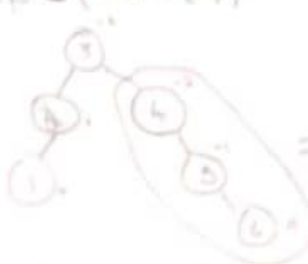
(RL rotation)



Step 3 (insert 5)



Step 4 (insert 6)

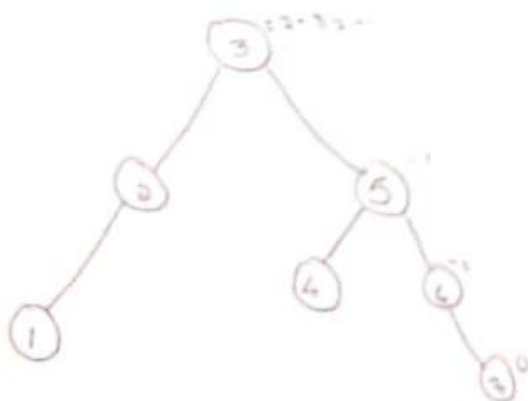


Unbalance

(RL rotation)



Step 5 (insert 7)

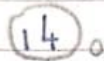


(Balance)

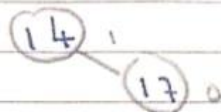
Insertion :

14, 13, 11, 7, 53, 4, 13, 2, 8, 60, 19, 16, 20

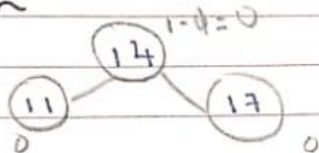
Step #1:-



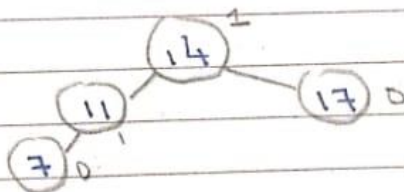
Step #2:-



Step #3:-

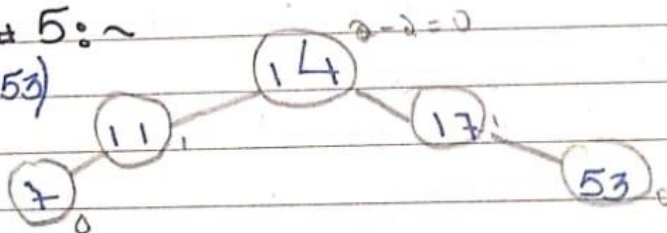


Step #4 (insert 7)



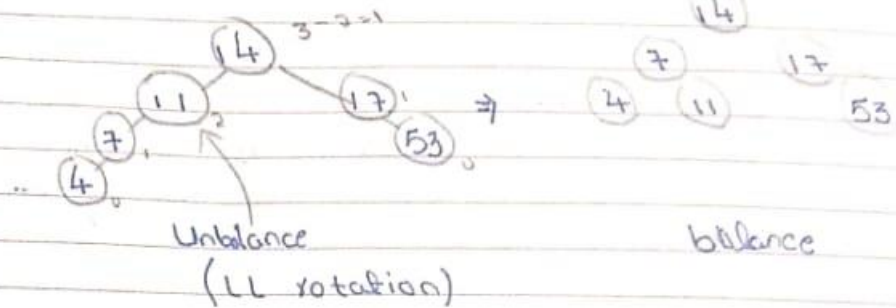
Step #5:-

(insert 53)

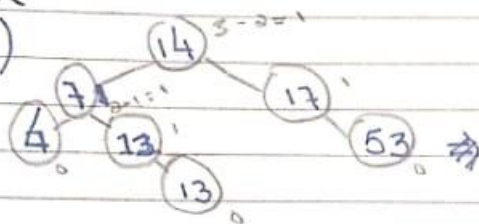


Date:

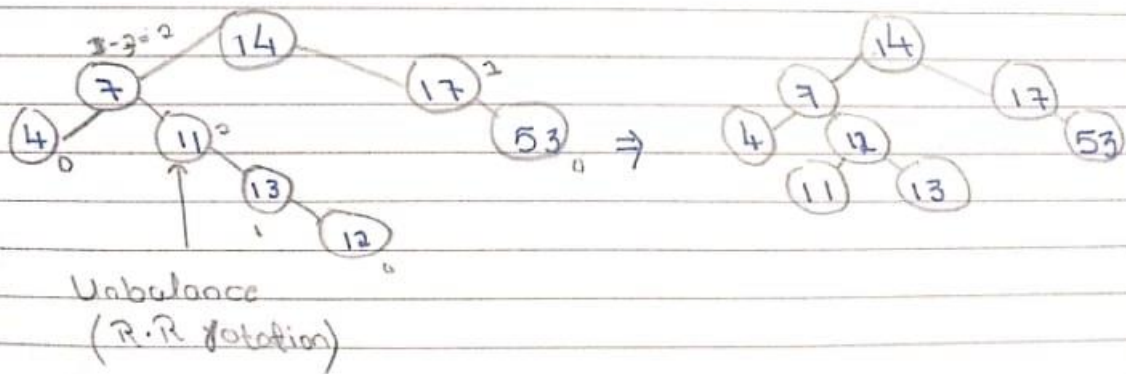
Step #6:-
(insert 4)



Step #7:-
(insert 13)



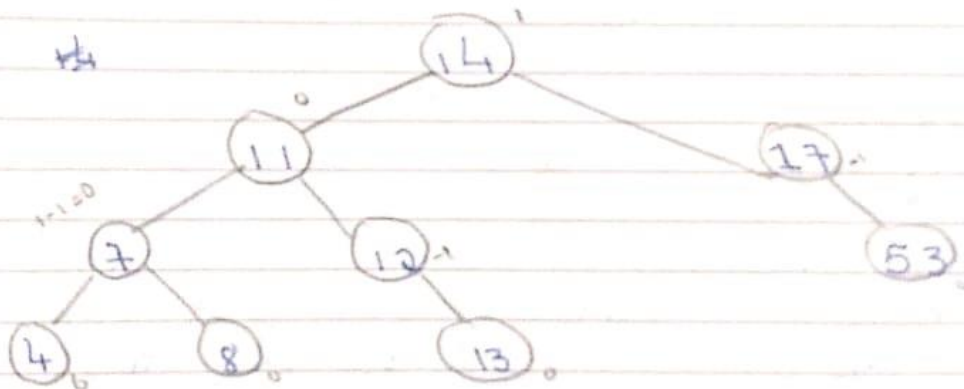
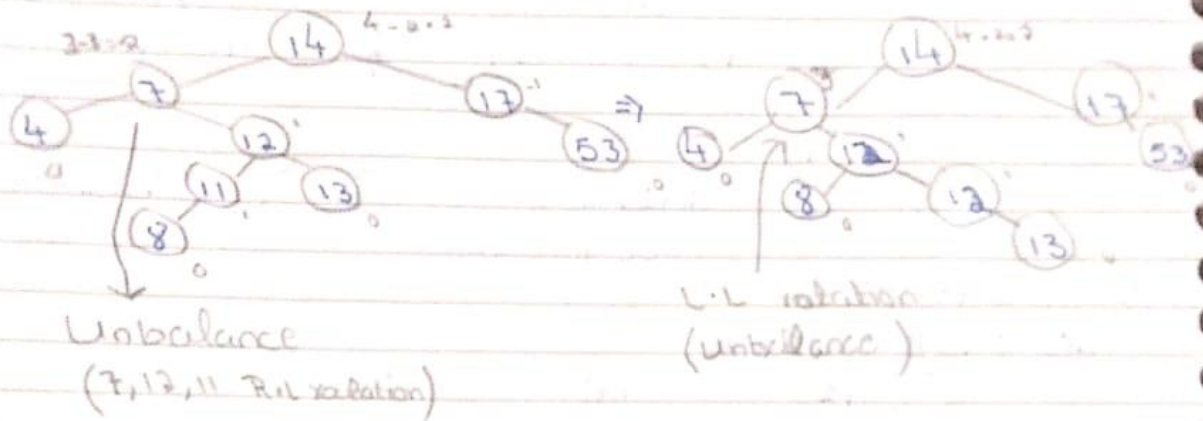
Step #8:-
(insert 12)



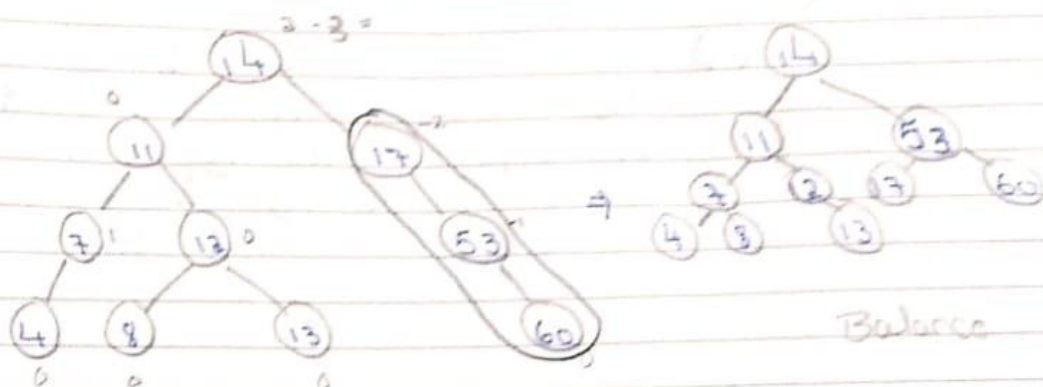
MIGHTY PAPER PRODUCT

Date: _____

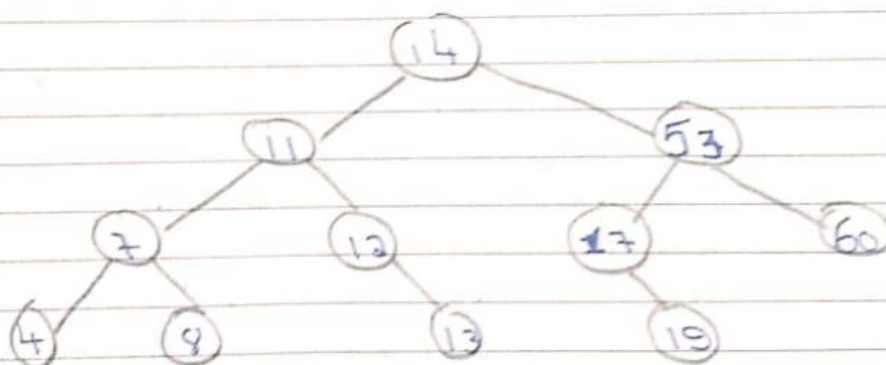
Step 9 in
(insert 8)



Step # 10
(insert 60)



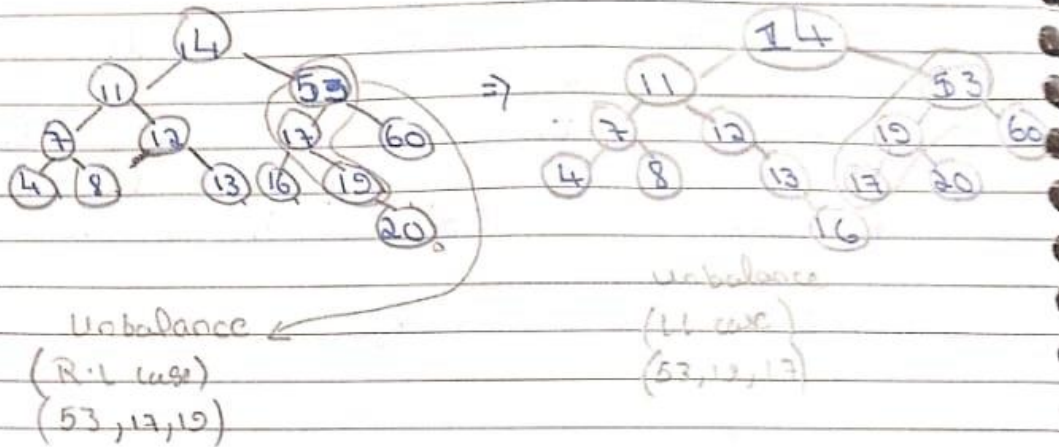
Step # 11
(insert 19)
Unbalance (R.R rotation)



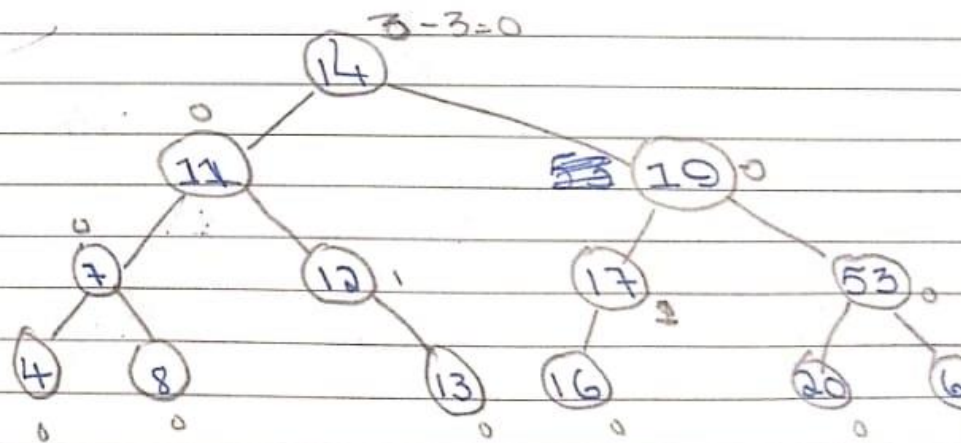
MIGHTY PAPER PRODUCT

Date: _____

Step #12
(insert 20)



Now final product will be



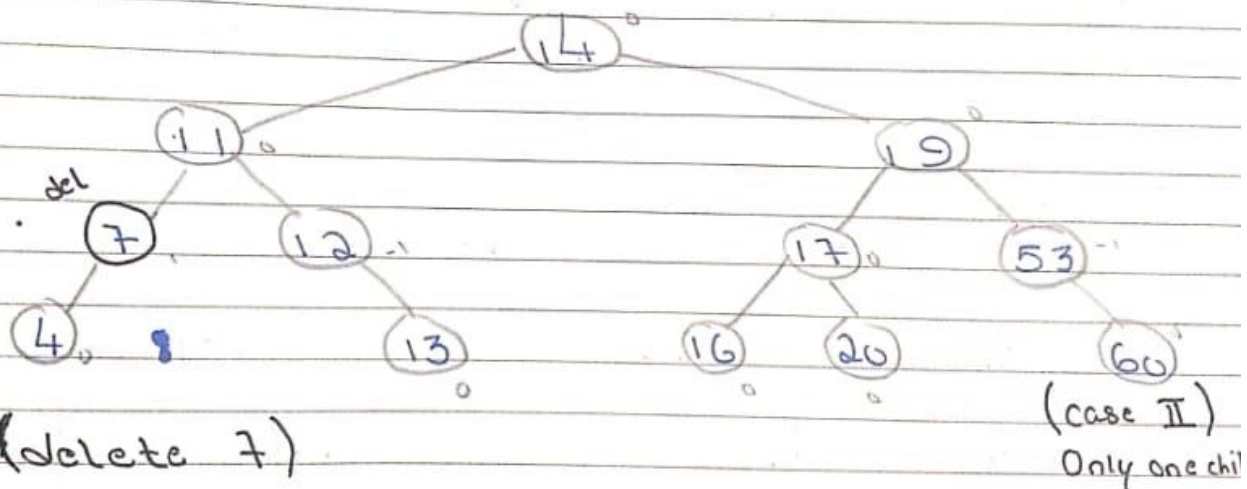
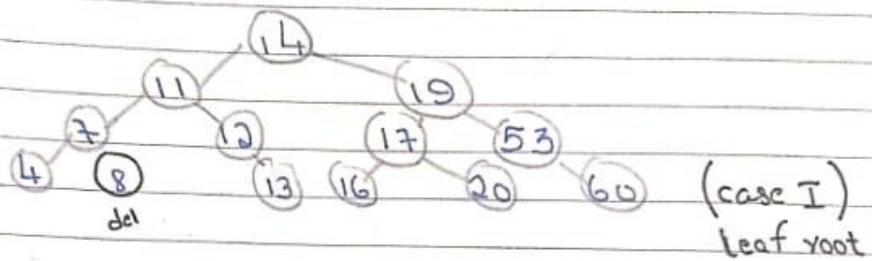
MIGHTY PAPER PRODUCT

Date: _____

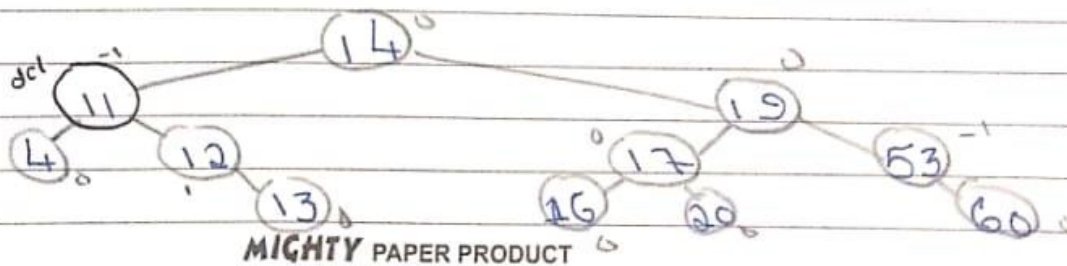
Delete:-

8, 7, 11, 14, 17

Step # 1
(delete 8)

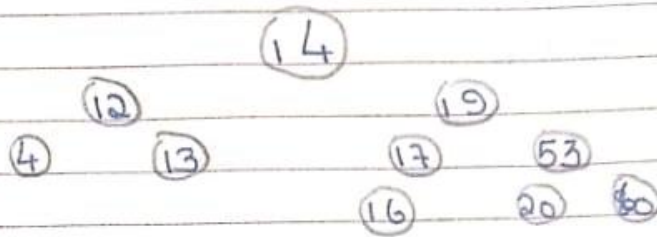


(delete 7)



Date: _____

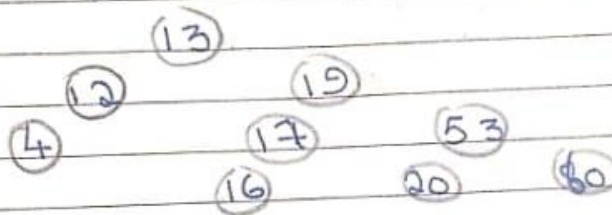
(delete 11)



(del 14)

(Case III)
two child
min of right

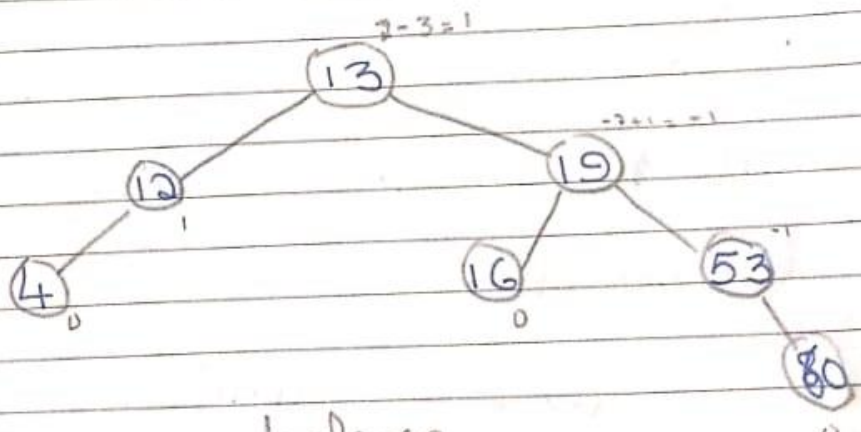
Case III
max from
left become
right.



(del 17)

Case II

Only one
child

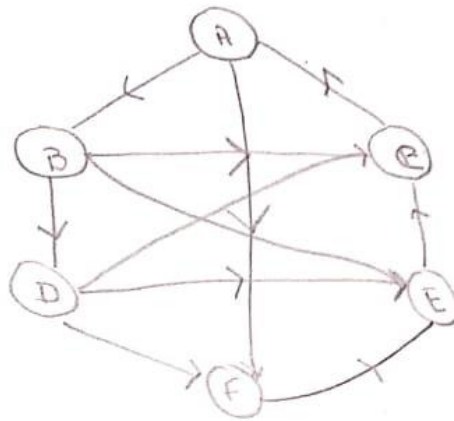


balance

7. Graph:

Representation:-

Example. I:



Matrix:

$$M = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Linklist:

A = B, F;

B = C, D, E;

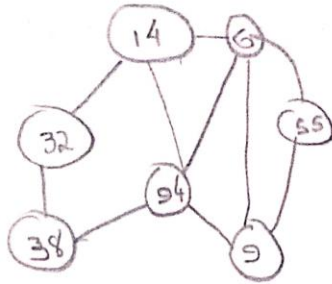
C = A;

D = C, E, F;

E = C;

F = E;

Example 2:



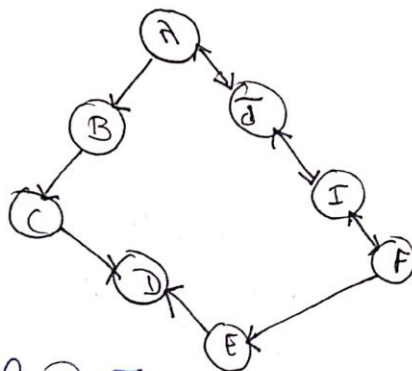
BFS:

14, 32, 94, 38, 9, 55, 6.

DFS:

14, 32, 38, 94, 6, 9, 55.

Example 3:



BFS: A, B, ~~J~~, J, C, D, I, F, E.

DFS: A, B, C, D, J, I, f, E.

h