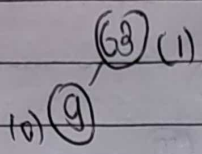
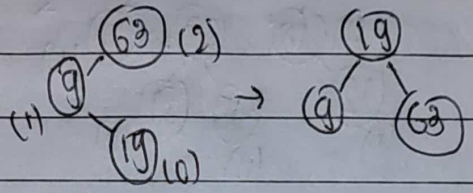
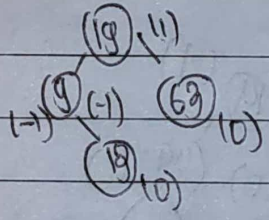
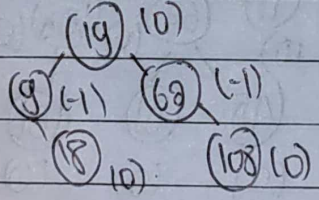
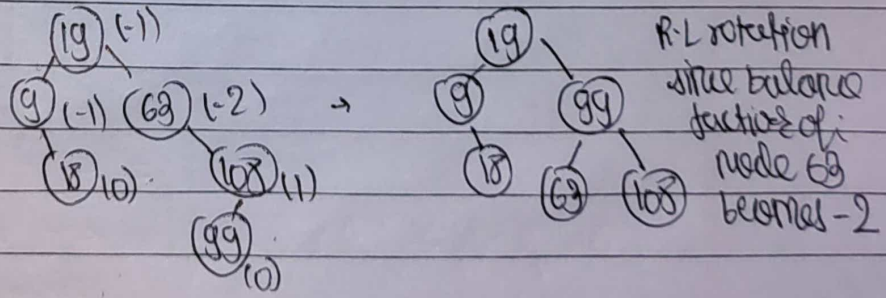
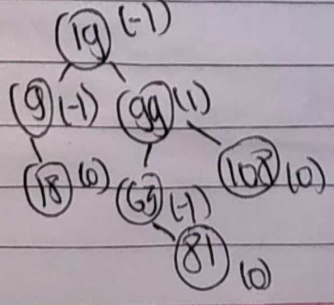
Harsh Kasliwal
2003085
C21

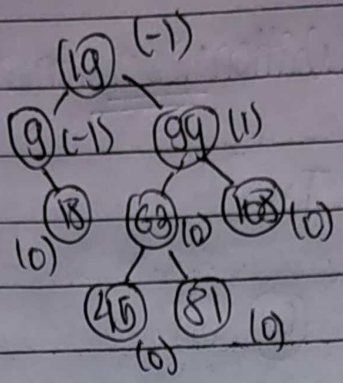
Page No.
Date

Written Assignment 2

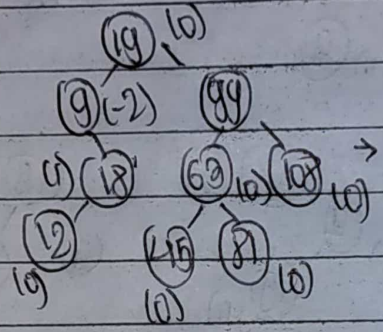
Element	Tree
63	
9	 No rotation required.
19	 L-R rotation required since balance of node 63 becomes 2.
18	
108	
99	 R-L rotation since balance factor of node 63 becomes -2
81	

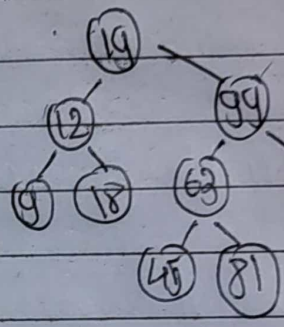
Harsh Kasliwal (2)
2003085

45



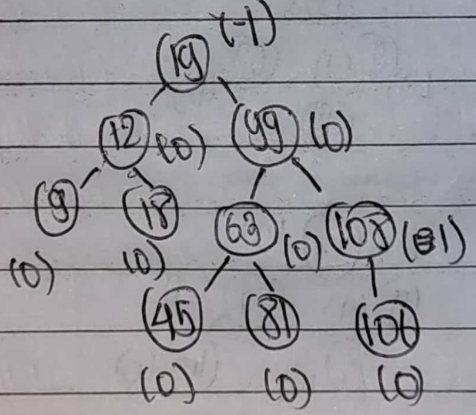
12





R-L rotation
BF of node 9 becomes -2.

106



No rotation required.

Harsh
085

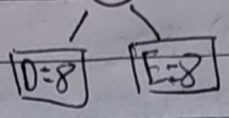
(3)

Page No.
Date

(2) Symbol A B C D E
Frequency 24 12 10 8 8

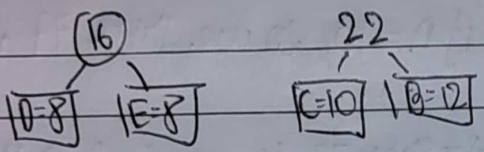
D=8, E=8, C=10, B=12, A=24

i=0 [C=10] [B=12] (16) [A=24]



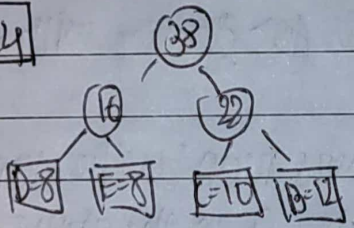
Adding D & E to the tree as they are lowest

i=1 (16) 22 [A=24]



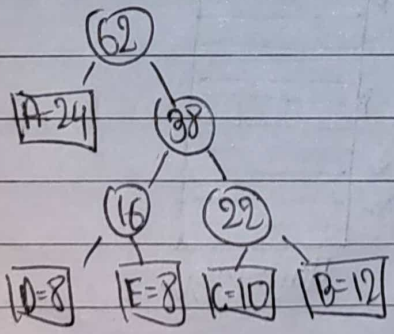
Now (10) & B(12)

i=2 [A=24]



Now 16 & 22 are the two lowest nodes, so we ~~do~~ add them.

i=3



Combine A(24) & 38 node.

⇒ A=0, B=111, C=110, D=100, E=101;

are the Huffman codes for the given frequency table.

Page No.
Date

(3) Hashing is the process of converting a given key into a code, smaller than the input key using a hash function. The hash function maps data of arbitrary size to a fixed range. The values generated by hash function are called hash codes and are used to index a table called the hash table.

→ Hashing is done in order to have faster access to arbitrary data elements usually in O(1) time

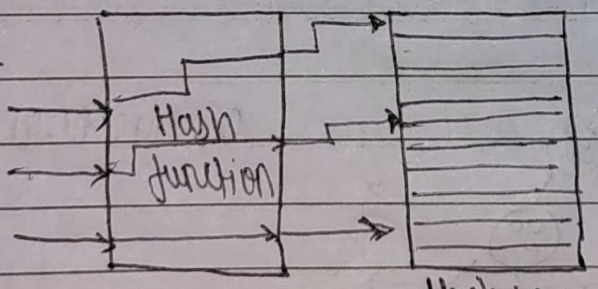
→ There are various hash functions available like modulo division folding, mid-square

→ Hash code of 77 will be 7 if n=10, so we store 77 at index 7 in the hash table

→ Now if we want to check whether 77 is contained in the hash table we just have to check the index 7.

→ There are also many methods for dealing with collisions, like linear probing quadratic probing.

Collisions occur when two keys map to the same index, like 77 and 87 both mapping to index 7 in the above example.



Hash 085

(5)

Page No.
Date

0 : 10	Linear probing.
1 : 44	i> 63 %. 10 = 3
2 : 82	ii> 82 %. 10 = 2
3 : 63	iii> 94 %. 10 = 4
4 : 94	iv> 77 %. 10 = 7
5 : 53	v> 53 %. 10 = 3
6 : 23	vi> 87 %. 10 = 7
7 : 77	vii> 23 %. 10 = 3
8 : 87	viii> 55 %. 10 = 5
9 : 55	ix> 10 %. 10 = 0
	x> 44 %. 10 = 4

Probes

collision final location = 5 (3)
Collision final location = 8 (2)
Collision final location = 6 (4)
Collision final location = 9 (5)
Collision final location = 1 (8)
Total collisions = 5

Quadratic probing.

0	10
1	
2	82
3	63
4	94
5	55
6	
7	77
8	87
9	53

i> 63 %. 10 = 3.
ii> 82 %. 10 = 2
iii> 94 %. 10 = 4.
iv> 77 %. 10 = 7
v> 53 %. 10 = 3. collision. ✓
$(3+1^2)$ %. 10 = 4.
$(3+2^2)$ %. 10 = 7
$(3+3^2)$ %. 10 = 2.
$(3+4^2)$ %. 10 = 9 ✓
vi> 87 %. 10 = 7 collision ✓
$(7+1^2)$ %. 10 = 8 ✓
vii> 23 %. 10 = 3 collision ✓
$(3+1^2)$ %. 10 = 4
$(3+2^2)$ %. 10 = 7
$(3+3^2)$ %. 10 = 2.
$(3+4^2)$ %. 10 = 9
$(3+5^2)$ %. 10 = 8
$(3+6^2)$ %. 10 = 9
$(3+7^2)$ %. 10 = 2
$(3+8^2)$ %. 10 = 7
$(3+9^2)$ %. 10 = 4.

⇒ 23 cannot be placed in table ✓

Harsh
085

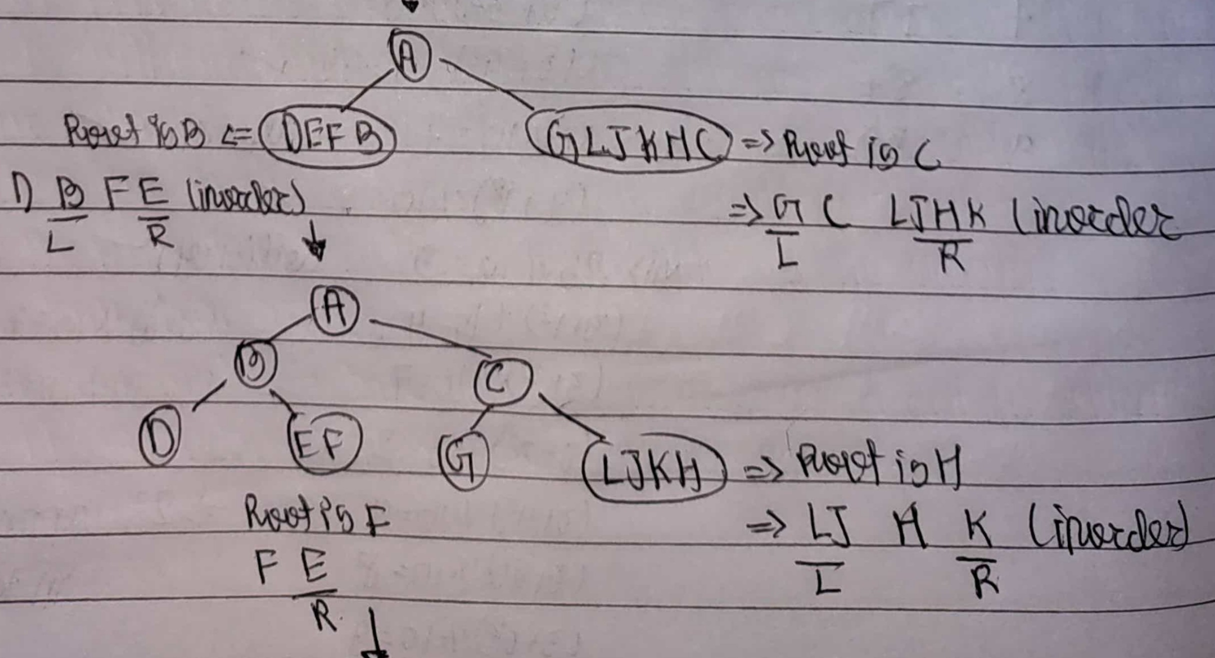
viii> 55 % 10 = 5.
ix> 10 % 10 = 0
x> 44 % 10 = 4 collision ✓

$(4+1^2)\ \%\ 10 = 5$.
$(4+2^2)\ \%\ 10 = 8$
$(4+3^2)\ \%\ 10 = 3$.
$(4+4^2)\ \%\ 10 = 0$.
$(4+5^2)\ \%\ 10 = 9$.
$(4+6^2)\ \%\ 10 = 0$.
$(4+7^2)\ \%\ 10 = 9$.
$(4+8^2)\ \%\ 10 = 8$.
$(4+9^2)\ \%\ 10 = 5$.
cannot be placed in table.

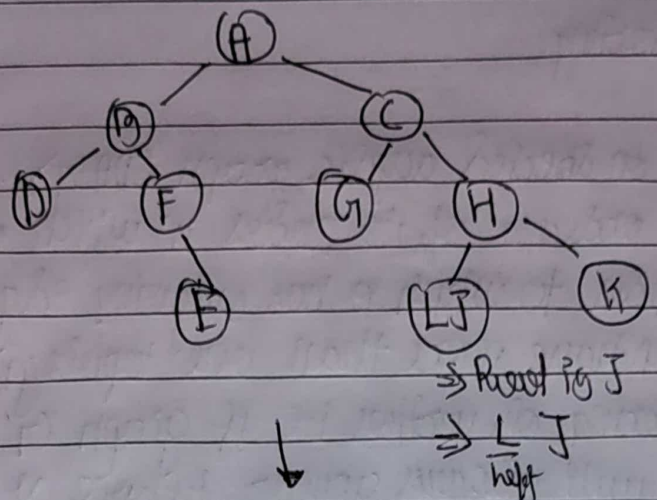
=> No. of collisions = 4.

(4) Post: D E F B G L J K H C A → left right root => root at end
In: D B F E A G C L J H K → left root right => root in middle.

DEFBGLJKHCA => Root is A.

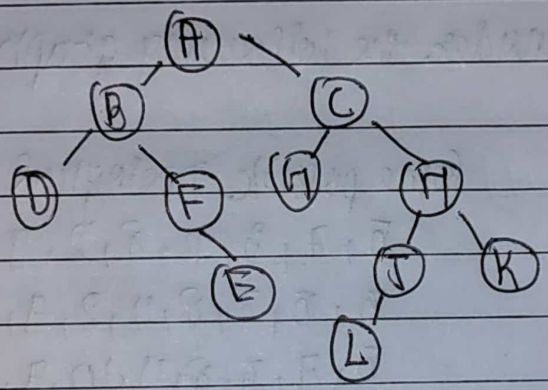


(7)



⇒ Root is J

⇒ L J
left



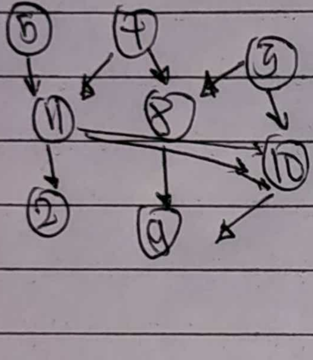
Harsh Kaslical
2003085 C21. (8)

Page No. | Date

⑤ Topological Sorting

→ It is applied on directed acyclic graph (DAG).
→ It is linear ordering of its nodes in which nodes comes before all nodes to which it has outgoing edges
→ Every DAG can have more than one topological sort.
→ It is an ordering of vertices i.e if graph G contains an edge (u, v) then u will always appear before v in the ordering.

For example, consider the following graph.



Some possible topological sortings are.
5, 7, 3, 11, 8, 2, 9, 10.
3, 5, 7, 8, 11, 2, 9, 10
5, 7, 3, 8, 11, 10, 9, 2

→ A simple algorithm to implement topological sorting can be written using stack & recursion.

→ We start from nodes that has no incoming edges and push them into a stack without printing them, and set visited for those nodes as true. Then we pick any node and keep recursively calling the function for all adjacent nodes. Again we do not print any nodes. Also, the node is only pushed in stack after all the adjacent nodes have been pushed
[Also set visited = true for any node accessed and do return if an already visited node is visited again.