Name :- Zubayer Ahmed Zidhan Laskar , Roll :- 200710007062

Branch :- Computer Science & Engineering , Semester : 3rd

Subject :- Data Structures & Algorithms , Date :- 11/12/2021

CIE – II

1. Ans :-

Algorithm delete (start)

Begin

1. If (start = NULL) Then
2. Write ("Underflow!")

~~Return~~

~~Else If (start → next == start)~~

3. Else If (NEXT(start) = start)
4. start = NULL
5. Else
6. temp ← start
7. Repeat step 8 ~~through~~ while NEXT(temp) <> start
8. temp ← NEXT(temp)
9. NEXT(temp) ← NEXT(start)
10. PREV(NEXT(start)) ← temp
11. Start ← NEXT(temp)
12. End If
13. Return

For the above algorithm "delete", the worst-case scenario is when there are 'n' number of elements in the list and $n > 1$. When n is very large, $O(n) = 1 + 1 + 1 + (n-1) + 1 + 1 + 1$

$\therefore$ $O(n)$ is of the order of n i.e. $\boxed{O(n) = n}$

This is because ther is only one loop present in the algorithm.

—

2) Ans:- Algorithm move (start, element)

Begin

1. If (Start = NULL) Then
2. Write ("Underflow!")
Return
3. Else
4.
5. temp ← start
6. If INFO(start) = element
Return
EndIf
7. Repeat step 8 while INFO(temp) <> element
If (NEXT(temp) = start && INFO(temp)<>element)
Write ("Element not present").
Return
EndIf
8. temp → NEXT(temp)
9. temp2 ← start
10. Repeat step 11 while NEXT(temp2) <> temp
11. temp2 = NEXT(temp2)
12. If (NEXT(temp) = Start) Then

2) Ans:- Algorithm move (start, element).

Begin

1. If (start = NULL) Then
2. Write ("Underflow!")
3. Return
4. Else
5. temp ← start
6. If INFO (start) = element
7. Return
8. EndIf

9. Repeat <u>10 to 14</u> while INFO(temp) <> ~~tes~~ element
10. If (NEXT(temp) = Start && INFO(temp) <> element)
11.
12. Write ("Element not present")
13. Return
14. EndIf

temp → NEXT(temp)

15. temp2 ← Start
16. Repeat step 17 while NEXT(temp2) <> temp
17. temp2 ← NEXT(temp2)

~~temp2 → nex~~

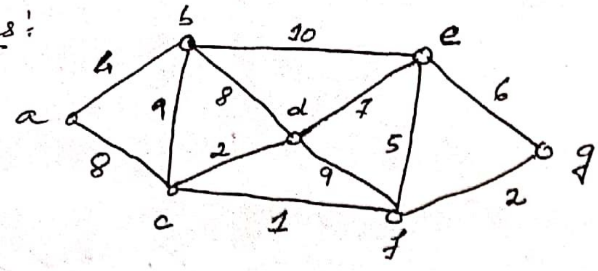
18. NEXT(temp2) ← NEXT(temp)
19. NEXT(temp) ← Start
20. temp ← Start
21. EndIf
22. Return

Worst-case time, $O(n) = 1 + 1 + 1 + 1 + 2(n-1) + (n-2) + 1 + 1 + 1$

$\therefore O(n) = n$ [of the order of n, because there are no internal loops]

The worst-case occurs when the element is present at the ~~last~~ end of the list, and the entire list has to be traversed ≈ twice for getting temp & temp2.

3. Ans:



(i)

	a	b	c	d	e	f	g
a	0	4	8	0	0	0	0
b	4	0	9	8	10	0	0
c	8	9	0	2	0	1	0
d	0	8	2	0	7	9	0
e	0	10	0	7	0	5	6
f	0	0	1	9	5	0	2
g	0	0	0	0	6	2	0

Adjacency Matrix

a → [b | 4] → [c | 8 | \]

b → [a | 4] → [c | 9] → [d | 8] → [e | 10 | \]

c → [a | 8] → [b | 9] → [d | 2] → [f | 1 | \]

d → [b | 8] → [c | 2] → [e | 7] → [f | 9 | \]

e → [b | 10] → [d | 7] → [f | 5] → [g | 6 | \]

f → [c | 1] → [d | 9] → [e | 5] → [g | 2 | \]

g → [e | 6] → [f | 2 | \]

Linked adjacency list

3. (ii) Ans:- For Breath first search :-

Step - 1

Q [~~g~~ | e | f | | | |]

Vaited : g

Step - 2

Q [~~g~~ | ~~e~~ | f | b | d | |]

Vaited :- ge

Step - 3 :

Q [~~g~~ | ~~e~~ | ~~f~~ | b | d | c |]

Visited : gef

Step - 4 :

Q [~~g~~ | ~~e~~ | ~~f~~ | ~~b~~ | d | c | a]

Visited : gefb

Step - 5 :

Q [~~g~~ | ~~e~~ | ~~f~~ | ~~b~~ | ~~d~~ | ~~c~~ | ~~a~~]

Visited : gefbdca

∴ The reqd. order in which the nodes were visited is :

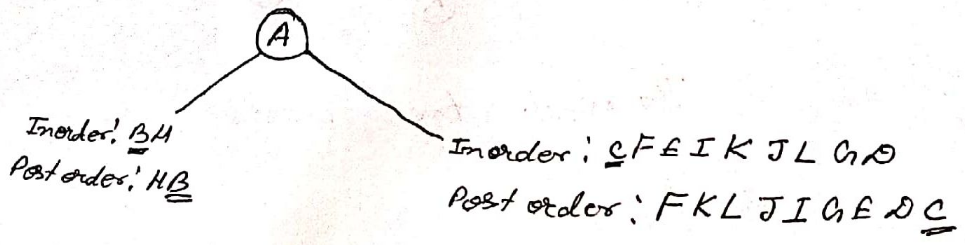
gefbdca.

4. Ans :- Given,

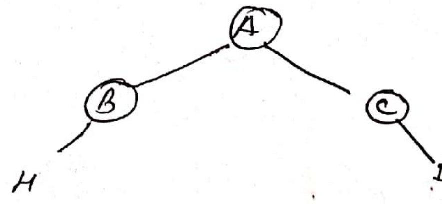
Inorder: B H A C F E I K J L G D

Post order: H B F K L J I G E D C A

Step - 1

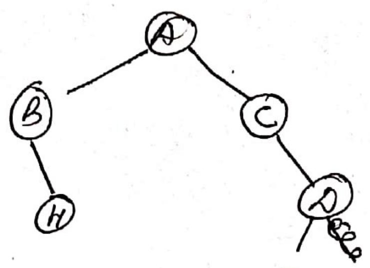


Step 2



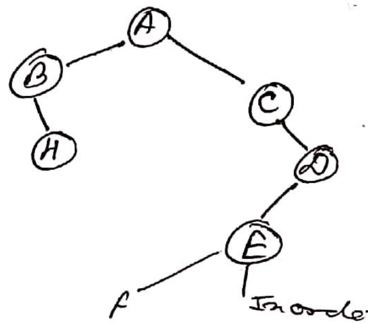
Inorder ; FEIKJL GD
Postorder; FKL JIGED

Step-3



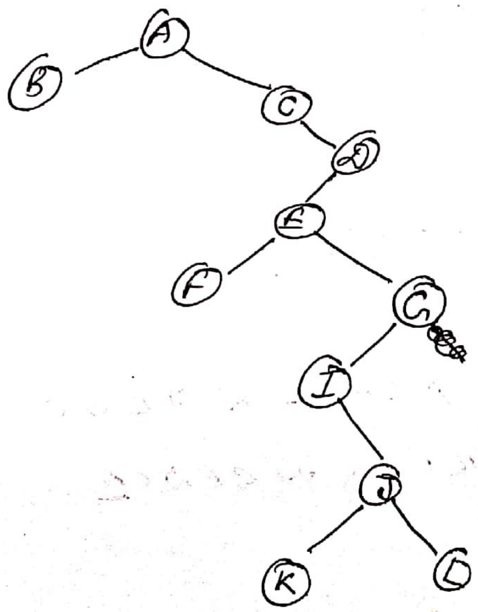
Inorder : FEIKJLG
Postorder : FKL JIGE

S-4



Inorder : IKJLG
Postorder : KL JIG

Step 5



The reqd. tree is as above