DATA STRUCTURES AND ALGORITHMS [CS 2103]

CLASS TEST

Name: Abhirup Mukherjee

Enrolment Number: 510519109

G-Suite ID: 510519109.abhirup@students.iiests.ac.in

Date of Examination: 05/01/2020

No. of Sheets Uploaded: 5

1) Disadvantage of Linear Queue is as follows:

i) We can only remove data from one side, if by mistake wrong data is sent to queue, it ~~is~~ is hard to rectify it [like delete the inserted node]

Solution: We can use double ended queue, what is allows us to do is that we can insert and delete elements from both side of the queue.

2) Following are a prerequisite of performing binary search on ~~the~~ a list of element.

i) Element should be sorted [either ascending or descending]

ii) Data should be easily addressable [not possiple with ~~lined~~ linked structure].

3) void print_range (node * ~~root~~, ~~node~~ temp , int k_1 , int k_2

{

// this function prints all values between k_1 & k_2

// node has , data, lchild and rchild.

// assume data is int

~~if (node -> data > k_1 && node -> data < k_2)~~

~~printf ("%d", node -> data);~~

if (~~node~~ temp == NULL)

return;

if (~~tep~~ temp -> data > k_1 && temp -> data < k_2)

printf ("%d", temp -> data);

print_range (temp -> lchild, k_1, k_2);

print_range (temp -> rchild, k_1 , k_2);

}

1) Let there be a full Binary tree with l no. of levels.

* Eg.: $l = 3$

→ have total no. of nodes $= n$ (let)

and $n = 2^l - 1$

→ no. of internal nodes will be every nodes except leaves

→ no. of leaves $= 2^{l-1}$

∴ no. of internal nodes $= 2^l - 1 - 2^{l-1}$

---

4) let total no. of nodes be n

$\therefore n = n_0 + n_1 + n_2$

where n_0 = node with zero child
n_1 = node with one child
n_2 = node with two child.

→ Total Branches $= n_0 \times 0 + n_1 \times 1 + 2 \times n_2$

also in a Full Binary tree with n nodes, $n-1$ branches are present

$$\therefore \quad n_0 + \cancel{n_1} + \cancel{n_2} - 1 = n_0 \times 0 + \cancel{n_1 \times 1} + n_2 \times \cancel{2}$$

or $n_0 - 1 = n_2$

or $\boxed{n_0 = 1 + n_2}$

∴ n_0 → node with zero child = leaves

n_2 → have two child = internal nodes

5) given $L_1 \to n$ nodes $(n \geq m)$

$L_2 \to m$ nodes

~~insert~~

~~void inser~~

node* insert_alternate (node* L1, node* L2)

{

node* temp 1 = L1;

node* temp 2 = L2;

while (temp 1 != NULL || temp2 != NULL)

{

~~node* next = temp1 → next;~~

~~temp L2 = L2 → next;~~

~~temp2 → next = next;~~

~~temp 1 → next = next;~~

node* next = temp 1 → next;

L2 = L2 → next;

~~L1 = t1 → next;~~

temp 1 → next = temp 2;

temp2 → next = next;

temp1 = next;

temp2 = L2;

}

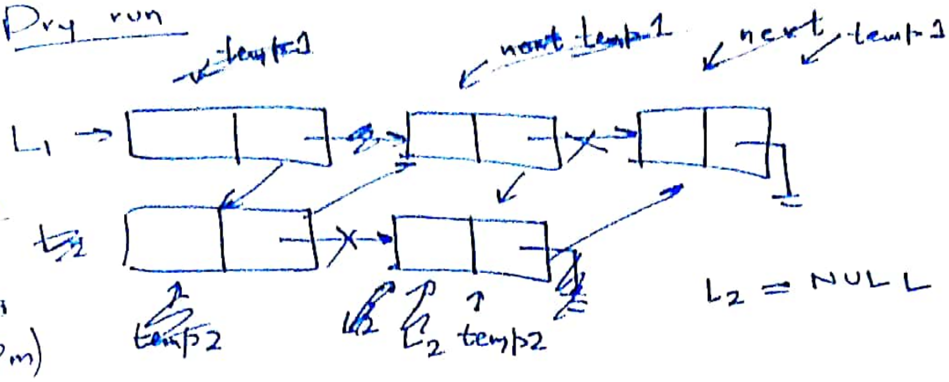
return L1;

}

Dry run

Eg 1

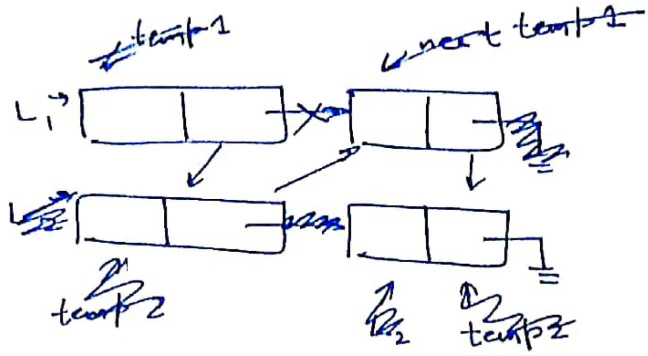
(n > m)



L_2 = NULL = temp2

Eg 2

(n = m)



temp1 = NULL

next = NULL

L_2 = NULL

temp2 = NULL