Name - Muhammed Raihan

Course - MCA 2C

Subject - DFS

DFS Theory Assignment 1

Solved Question Paper 2023

Questions	Fully Attempted	Partially Attempted	Not Attempted	Total questions
Q-1 a	✓			
Q-1 b	✓			
Q-1 c	✓			
Q-1 c	✓			
Q-1 f		✓		
Q-1 g			✓	
Q-1 h	✓			
Q-1 i	✓			
Q-1 j			✓	
Q-2 a	✓			
Q-2 b	✓			
Q-3 a	✓			
Q-3 b	✓			
Q-4 a	✓			
Q-4 b			✓	
Q-5 a	✓			
-5 b	✓			

(Please write your Exam Roll No.)

END TERM EXAMINATION

SECOND SEMESTER [MCA] JULY 2023

Paper Code: MCA-102 **Subject: Data and file Structures**

(BATCH 2020 ONWARDS)

Time: 3 Hours **Maximum Marks: 75**

Note: Attempt five questions in all including question Q.No.1 which is compulsory. Select one question from each unit.

Q1 Answer the following briefly: **(2.5×10=25)**

a) What do you mean by "sort in place"? Write an algorithm for selection sort.

b) Given In order traversal: BQRHEADLKMC and Pre order traversal: ABEHQRCDKLM of a tree. Find the tree

c) What are pitfalls encountered in singly link list.

d) How graphs are represented in computer memory? Give relative merits and demerits of each representation scheme.

e) Given the following arithmetic expression in infix notation as 12/(7-3) + 2 * (3 + 8) - 7. Translate this expression into postfix and then evaluate it.

f) If there are 27 nodes in a complete binary tree, what will be its height? Also tell how many nodes will be at last level.

g) Here is an array, which has just been partitioned by the first step of quick sort in order to sort it in the ascending order: 3,0,2,4,5,8,7,,6,9. Which of these could be pivot?

h) Write a algorithm for reversing a string using a stack.

i) Compare and contrast B tree and B+ trees? When you might prefer to use B+ tree instead of B trees?

j) What is hash table? What are the advantages of hash table over direct address table?

UNIT-I

Q2 a) Compare and contrast linear list with linear arrays. Create a pseudo code which moves the last element of link list to first element of the link list. **(6.5)**

b) Formulate an algorithm that detects and removes a cycle in link list. **(6)**

OR

Q3 a) What will be the complexity to the operation to add an element in the beginning of a linear linked list? Formulate an algorithm to merge 2 sorted linked lists **(6.5)**

P.T.O.

Create functions in C language for Insertion and deletion in circular queue. When would a circular queue be full? (6)

UNIT-II

Q4 a) Construct a binary search tree with following numbers that are inserted in an empty binary search tree 30, 40, 35, 25. 50, 45. Formulate an algorithm to insert a new node in binary search tree. (6.5)

b) The sequence <70, 15, 50, 5, 12, 35, 10> represents a max heap. Show the heap- after the following operations: i) Insertion of element 25. ii) Deletion from root iii) Insertion of element 75. (6)

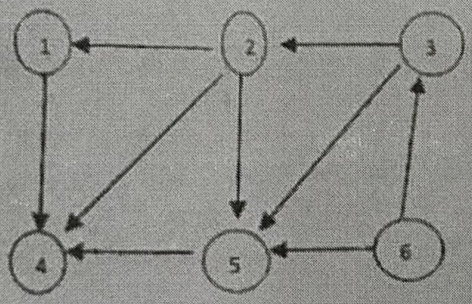
OR

Q5 a) State a few merits of AVL trees. Construct an AVL tree by inserting following elements one by one and count the total number of left and right rotations after inserting all the elements 35, 50, 40, 25, 30, 60, 78, 20, 28. (6.5)

b) Discuss the application of priority queue. Write a function in C that deletes root element from max heap. (6)

UNIT-III

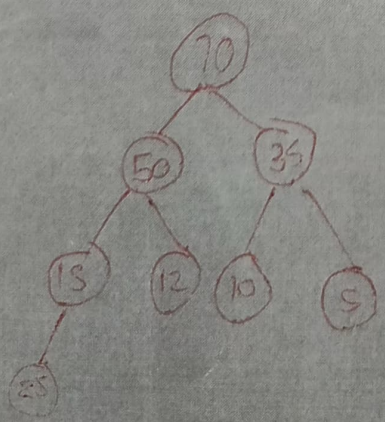
Q6 a) Elaborate topological sort. Find topological sort of the following graph (6.5)



b) Formulate an algorithm for Floyd - Warshall's algorithm to find the shortest paths between all pair of vertices (6)

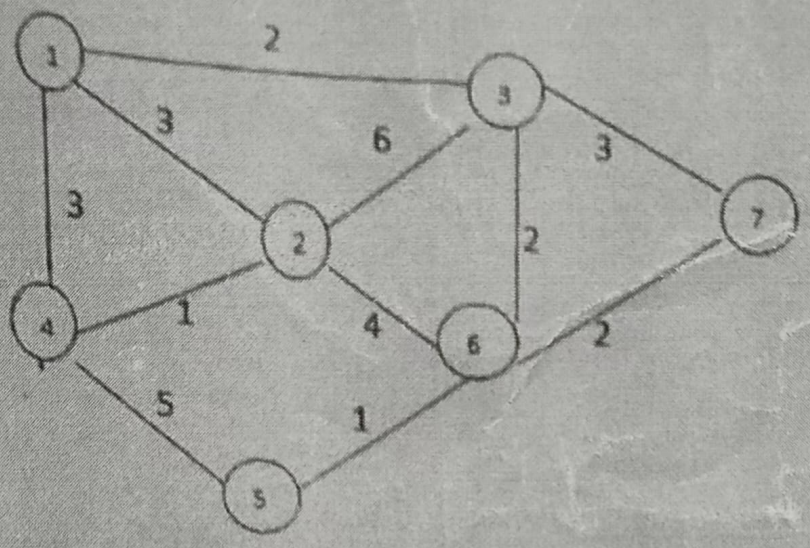
OR

Q7. (a) Show the progress of Dijkstra's algorithm to find the shortest path from vertex 4 to vertex 7 in a graph shown below (6.5)



P.T.O.

MCA-102
P2/3



b) What is a spanning tree? When it is called a minimum spanning tree? Create an algorithm to find minimum spanning tree by Prim's algorithm. (6)

UNIT-IV

Q8 a) Demonstrate the insertion of keys 73, 45, 37, 65, 77, 88, 100, 121 into a hash table with 11 slots using i) Separate chaining hash table ii) Open addressing hash table using linear probe iii) Open addressing hash table using quadratic probing with c1=0 and c2=2 (6.5)

b) Explain how error handling of files is done in C. Write a program in C to copy a file to another name (6)

OR

Q9 a) Discuss how sequential files are handled in C. Write a program in C to count the number of words and characters in a file. (6.5)

b) Explain the model for external sorting. Use k-way merge and 2 way merge sorting techniques to merge the following data on tape 81, 94, 11, 96, 12, 35, 17, 99,28,58, 41,75,15. Assuming internal memory can store 3 records. (6)

****

Q-1a) What do you mean by sort in place, write an algorithm for selection sort

An in-place sorting algorithm uses constant space for producing the output (modifies the given array only). It sorts the list only by modifying the order of the elements.

selection sort is a effective & efficient sort algorithm based on comparison operations

Algorithm of Selection Sort

Start with the first element: Let the first element be the current minimum.

Find the minimum element: Iterate through the rest of the array to find the minimum element.

Swap the minimum with the first unsorted element: Swap the minimum element found with the first unsorted element.

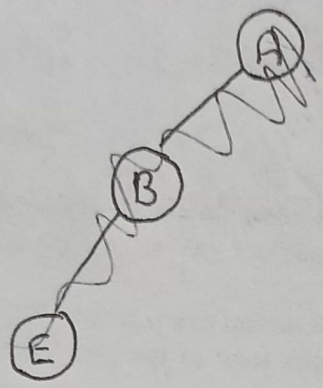
Repeat the process: Continue this process for the remaining unsorted portion of the array, each time select the next minimum element & swapping it with the appropriate position.

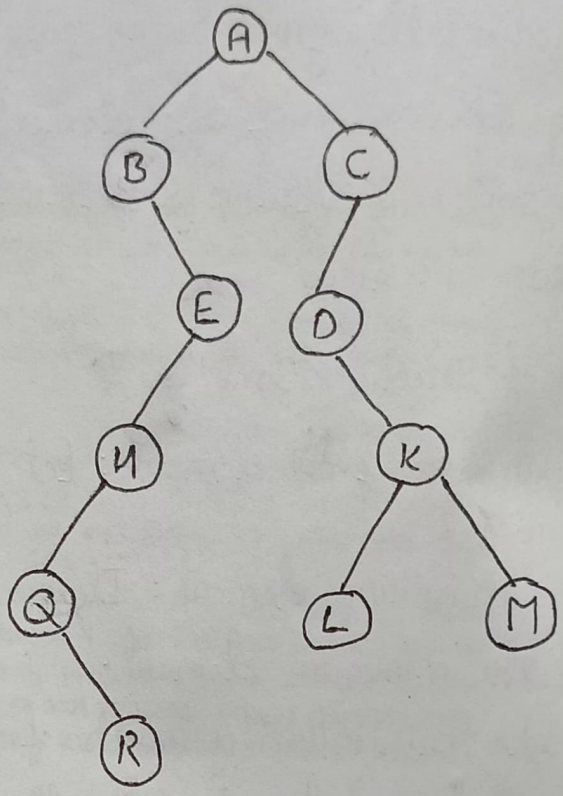
Continue until sorted: Repeat steps 2-4 until the entire array is sorted.

0	1	2	3	4	= Index
8	0	7	1	3	Length of array is 5
0	8	7	1	3	4 comparisons, Swap - 8 & 0
0	1	7	8	3	3 comparisons, Swap - 8 & 1
0	1	3	8	7	2 comparisons, Swap - 7 & 3
0	1	3	7	8	1 comparisons, Swap - 7 & 8

Q-1 b) Inorder Traversal BQRHEADLKMC LNR

Preorder Traversal ABEHQRCDKLM NLR





Inorder Traversal

B Q R H E A D L K M C

Preorder Traversal

A B E H Q R C D K L M

Preorder is used to select the next element to be inserted
Inorder is used to align elements into left & right position, used find the position into left & right

Q-1 c) What are the pitfalls encountered in singly linked list.

1) Traversal - Sequencial traversal from the head is required to access elements
2) Memory Overhead - Additional memory is needed for storing pointers to the next node
3) Insertion & deletion - Efficient at beginning O(1), but O(n) for arbitrary positions.
4) No Random Access - Cannot directly access element by index
5) No Backward Traversal - Only supports forward traversal

2-1 c) Expression 12/(7-3) + 2 *(3+8) - 7 . Convert infix to postfix & then evaluate it.

12/(7-3) + 2*(3+8) - 7

Character	Stack	Expression
12		12
/	/	12
(	/(	12
7	/(	127
-	/(-	127-
3	/(-	1273
)	/	1273-
+	+	1273-/
2	+	1273-/2
*	+*	1273-/2
(	+*(	1273-/2
3	+*(	1273-/23
+	+*(+	1273-/23
8	+*(+	1273-/238
)	+*	1273-/238+
-	+-	1273-/238+*+
7	+-	1273-/238+*+7
		1273-/238+*+7-

Ans

~~Postfix Expression - 1273-/238+*7+-~~

Postfix Expression 1273-/238+*+7-

Q-1 f) If there are 27 nodes in a complete binary tree, what will be its height? Also tell how many nodes will be at last level.

height = 5 ✓ 12 nodes at last level

Q-1 g)

Q-1 h) Write a algorithm of reversing a string using stack

Algorithm

1) Create an Empty stack
2) Loop through each character in the input string - Push each character onto the stack
3) Create an empty output string
4) Loop Until the stack is empty - Pop a character from the stack & append it to the output string
5) Return the output string as reverse string

Q-1 i) Compare & Contrast B tree & B+ tree? When you might prefer to use B+ trees instead of B tree?

B tree

- In a B tree, each node can have multiple children & multiple keys
- Keys & pointers to child nodes are stored in non leaf nodes.
- In B trees, data may be stored in both internal nodes & leaf nodes
- B tree are well suited for databases & file systems where both random & sequential access to data are important.
- B tree typically store data in sorted order, making range queries efficient.

+ Trees

In B+ Trees, only keys are sorted in non leaf nodes, while actual data entries are sorted in leaf nodes.

Leaf nodes are linked together in a linked list to support range query efficient

B+ Tree guarantee that data can be found in leaf nodes only, which simplifies searches.

B+ Tree are commonly used in databases where range queries are frequent & performance is critical.

B+ tree typically have a higher fanout (number of children per node) compared to B trees, leading to shallower trees

Because leaf nodes contain actual data entries, B+ trees can support sequential access more efficient than B trees.

When to use B+ tree instead of B tree

Range Queries

Memory Access Patterns

Leaf Node storage

Ordered data retrieval

-1) What is hash table? What are the advantages of hash table over direct access table.

Name - Mohammed Raihan
Course - MCA 2C
Subject - DFS
Solved Question Paper - 2023

UNIT - 1

12 a) Compare & contrast linear list with linear arrays. Pseudo code which moves the last element of list to first.

Arrays	Linked List
n array is a collection of elements of similar data type	A linked list is a collection of objects known as a node consists of 2 parts data & address.
rray elements store in a continous emory location	Linked list elements can be stored anywhere in the memory or randomly stored.
rray works with a static memory. ere static memory means that the emory size is fixed & cannot be hanged at the run time.	The Linked List works with dynamic memory. Here, dynamic memory means that the memory size can be changed at the run time according to our requirements.
rray takes more time while performing ny operation like insertion, deletion tc.	Linked list takes less time while performing any operation like insertion, deletion etc.
n array memory is allocated at compile time	In ~~array~~ linked list memory is allocated at run time.

Memory Utilization is inefficient in the array. For ex, if the size of the array is 6, & array consists of 3 elements only the rest of the space will be unused.

Irrespective of size, an array takes constant time for accessing an element Time complexity is O(1)

Takes less memory as compared to linked list

Implementation of arrays is easy

Random access is possible

Memory Utilization is efficient in the case of a linked list as the memory can be allocated or deallocated at the run time according to our requirements

Random access is not possible in or to access an element we have traver all the previous nodes

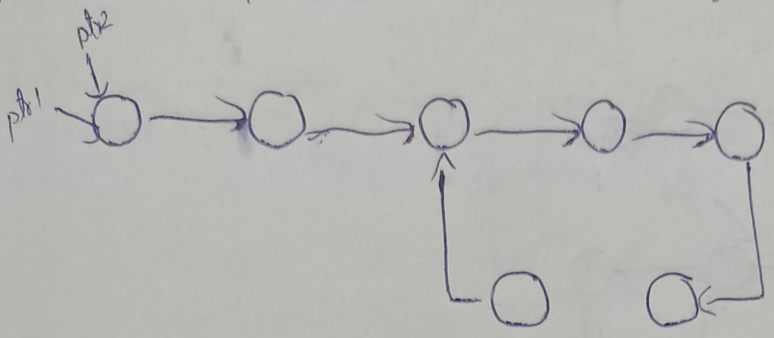
There is no unsued memory but extra memory is occupied by th pointer varibles.

Implementation is difficult

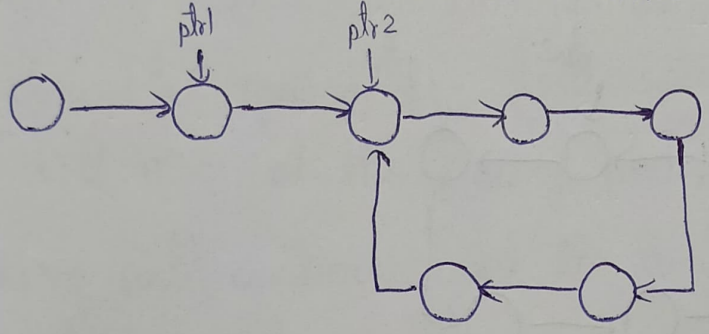
- Traverse the linked list till last node & initialize two pointers to store the address of the last node & second last node
- Then follow these three steps to move the last node to the front
- Make second last as last (secLast → next = NULL)
- Set next of last as head (last → next = head)
- Make last as head (head = last)

L-2 b) Formulate an algorithm that detects & remove a cycle in linked list.

Step 1 - Both the pointers (ptr1 & ptr2) begin by pointing at the 1st Node



Step 2- The two pointers start traversing the list, ptr1 one node at a time, and ptr2 two nodes at a time



Step 3 - Since ptr2 moves faster than ptr1, it will reach the end of the linked list sooner.

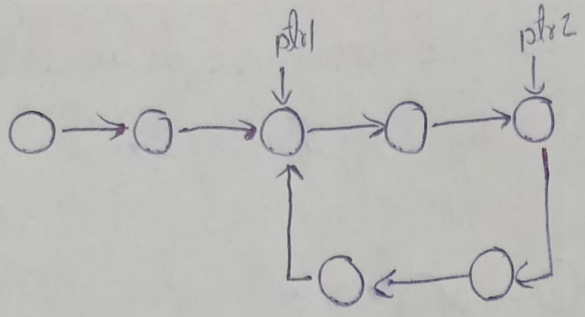
After reaching the end of the list, there are two possible scenarios

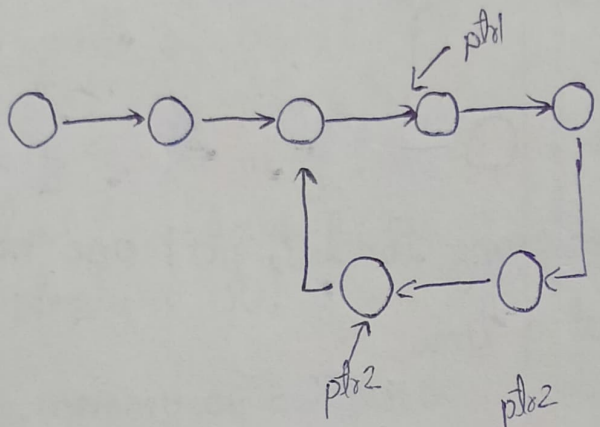
In the first scenarios, there will be no loop in the list & therefore ptr2, will point to NULL since ptr2 moves two nodes at a go.

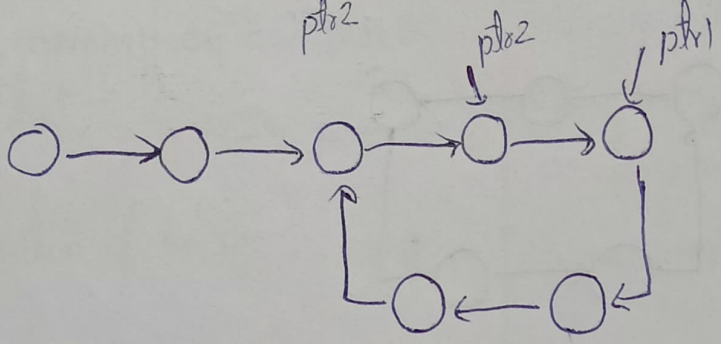
This means that ptr2 will become null even if there are several nodes. In this case we stop the execution & demonstrate the output of the code as 'no loop'.

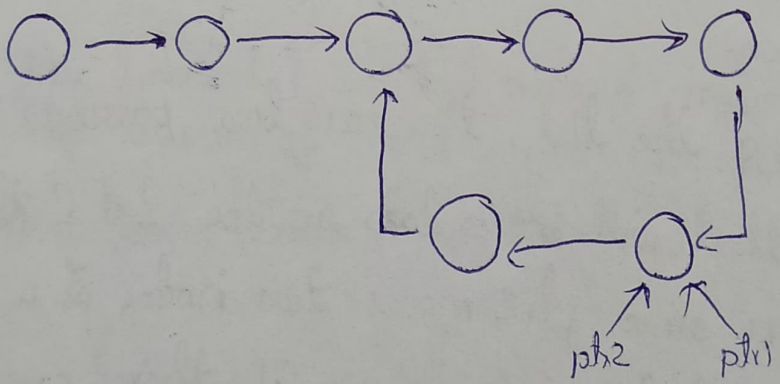
The second scenario is the presence of loop. The pointer will not reach NULL. Instead, they will enter the loop & cross over there.

Additionally, at some stage during traversing the list, the two pointers will eventually meet at a particular node.





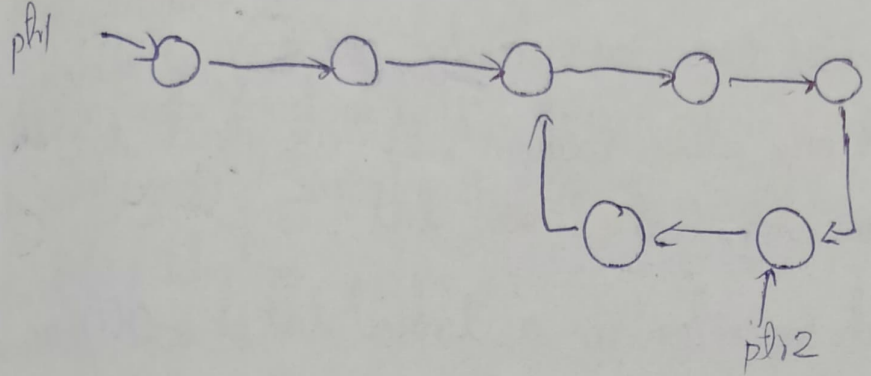




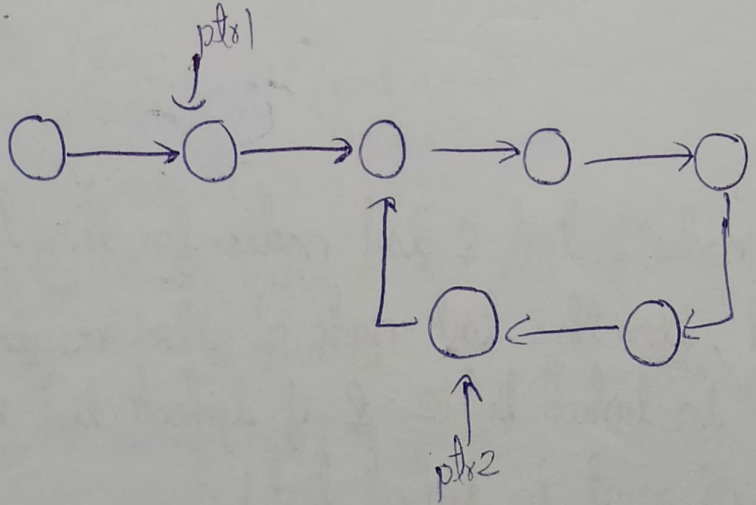
How to remove a loop

step1- As Mentioned, ptr1 & ptr2 will finally point to the same node in the loop. so, we have to find the last element in the loop & make it point to null.

Step 2- Leave ptr2 in its current location while making ptr1 point to the head of the loop



Step 3- Next we traverse the two pointers through the linked list. Rather than Ptr2 moving at a faster speed than ptr1, both pointers will move at the pace of ptr1, one node at a time. The traverse will continue until the two pointers eventually meet at one node.



Step 4- We will make ptr2 point to the null value.

Q-3 a) 1st Part Time complexity of adding an element in the beginning of linked list. Formulate an algorithm to merge 2 sorted linked list

For insertion on linked list, the time complexity is O(1) if done on the head, O(N) if done at any other location, as we need to reach that location by traversing that linked list.

Space complexity for each operation in a linked list is O(1) as no extra space is required for any operation.

Merge 2 sorted linked list

Ex Linked List A - 1→5→8→10

Linked List B 3→6→9→12

After merging 1→3→5→6→8→9→10→12

Brute Force Approach

Pseudocode

1) Create two temporary nodes, start & tail nodes for the linked li

2) If linked list 1 is null, then the tail node of the resulting linked list will point to linked list 2 & if linked list 2 is nu then the tail node will point to linked list 1.

3) Compare the data from both nodes of linked list 1 & Linked list 2 & make the tail node point to the node with higher (smaller) value

Move the tail pointer to tail→next

Move the pointer of the list from which the node was taken to its next node

4) Continue the comparison & keep pointing the tail node to one of the node from the given linked lists till one of the linked l is completely traversed.

If one of the given linked lists is completed, then add the remaining elements to the tail node of the resulting linked list & display the output.

3 b) Create functions in C for insertion & deletion in circular queue. When would circular queue be used

```
#include <stdio.h>
#define SIZE 5
int cqueue_array [SIZE];
int front = -1, rear = -1, value;

void EnQueue(){
  if ((rear+1)%size == front){
    printf("Queue Overflow");
  }
  else{
    if(front == -1)
      front = 0;
    rear = (rear+1) % SIZE;
    printf("Enter value of the new node : ");
    scanf("%d", &value);
    cqueue_array[rear] = value;
  }
}
```

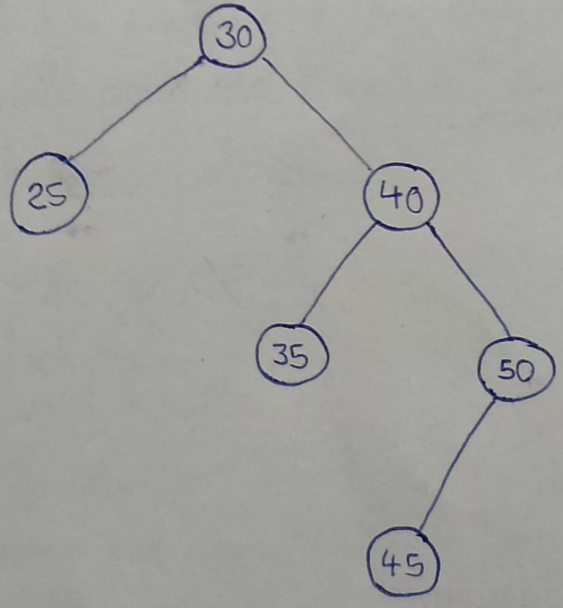
Applications of circular queue

- CPU scheduling
- Inter process communication
- Resource allocation & management

```
void DeQueue(){
    if(front==-1){
        printf("Queue Underflow");
    }
    else{
        printf("Element deleted from the queue : %d\n", cqueue_array[
        if(front==rear){
            front=-1;
            rear=-1;
        }
        else{
            front=(front+1)%SIZE;
        }
    }
}
```

UNIT - II

Q-4a) Nodes to be inserted 30, 40, 35, 25, 50, 45 Algo for inserti



Algorithm for insertion operation in a binary search tree

```
if node == NULL
    return createNode(data)
if (data < node->data)
    node->left = insert(node->left, data);
else if (data > node->data)
    node->right = insert(node->right, data);
return node;
```

step 1 - Start at the root of the tree

step 2 - If tree is empty, create a new node & make it the root

step 3 - If the value of the new node is less than the value of the current node, move to the left child of the current node

step 4 - If the value of the new node is greater than the value of the current node, move to the right child of the current node.

step 5 - Repeat step 3 & 4 until a leaf node is reached

Q-4 b)

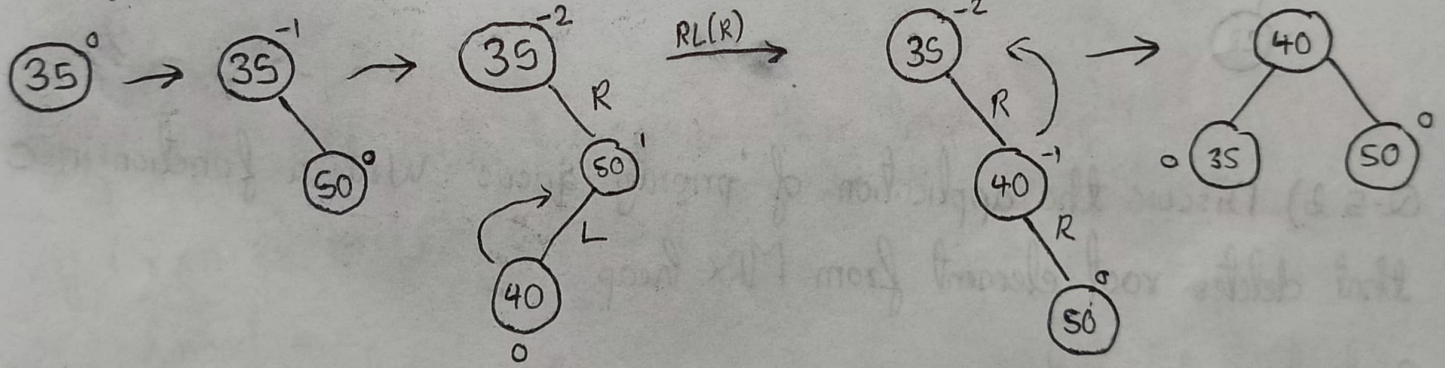
Q-5 a) Merits of AVL Tree

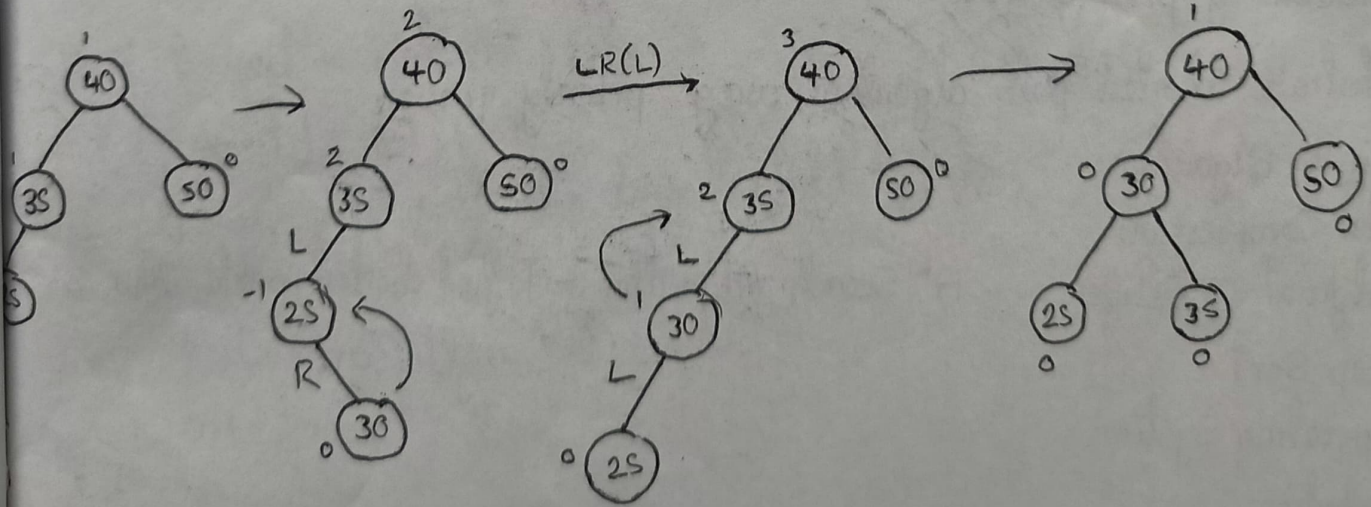
The AVL tree is always height balanced, and its height is always O(log(N)), where N is the total number of nodes in the tree.

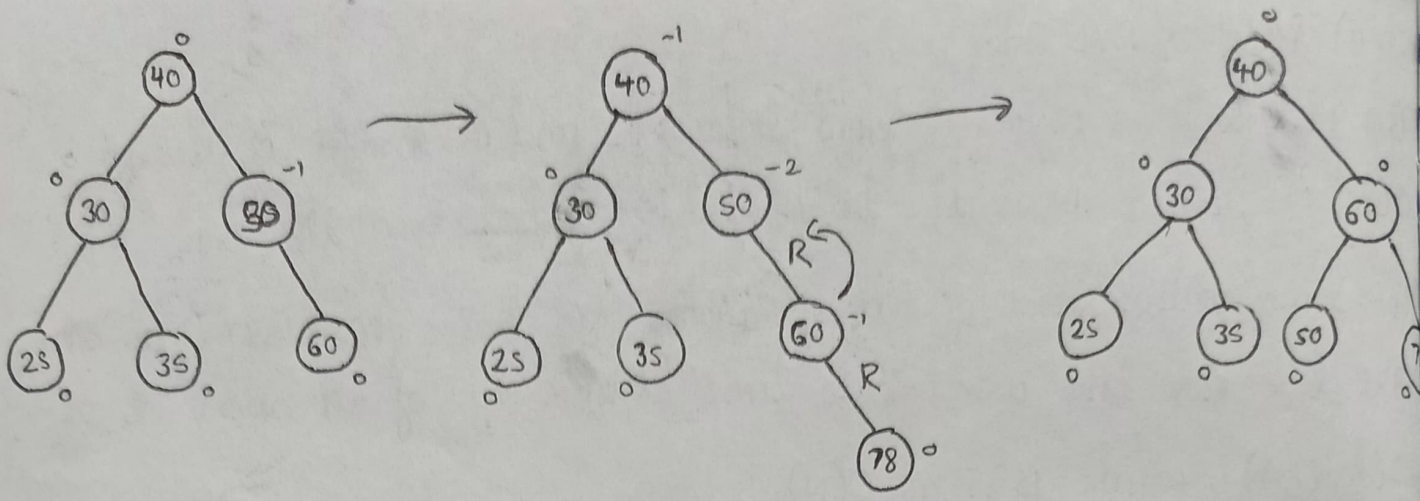
The time complexities of all operations are better than BST's as the AVL tree has the worst case time complexity of all operations as O(log(N)), while BST, is O(N).

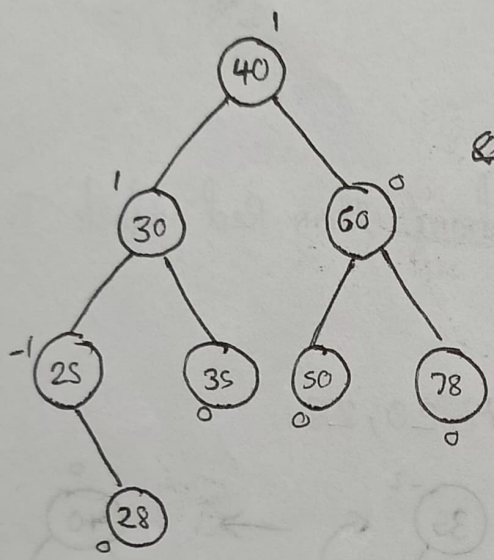
- AVL tree have self balancing properties
- AVL trees are not skewed
- AVL trees perform better searching operations than Red Black Trees
- It shows better search time complexity

Nodes for insertion 35, 50, 40, 25, 30, 60, 78, 20, 28









Ans

Left Rotations = 3

Righ Rotations = 2

Q-5 b) Discuss the application of priority queue. Write a function in C that deletes root element from MAX heap

Applications of priority queue

- Dijkstra's Shortest path algorithm using priority queue
- Prim's Algorithm
- Data Compression
- Artificial Intelligence - A* Search Algorithm - To find shortest path b/w 2 weighted graphs
- Heap Sort
- Operating Systems
- Robotics
- Medical Systems

```
#include<stdio.h>
// Function to heapify a subtree rooted with node i in the max heap
// arr[]: array representing the heap
// n: size of the heap
// i: index of the node to heapify
void heapify(int arr[], int n, int i) {
    int largest = i;  // initialize largest as root
    int l = 2*i+1;  // left child index
    int r = 2*i+2;  // right child index

    // If left child exists and is greater than largest so far root
    if (l<n && arr[l] > arr[largest]
        largest = l;
    // If right child exists & is greater than largest so far
    if (r<n && arr[r] > arr[largest])
        largest = r;
    // If largest is not the root
    if (largest != i) {
        // swap root with the largest child
        int temp = arr[i];
        arr[i] = arr[largest];
        arr[largest] = temp;
        heapify(arr, n, largest);
    }
}
```

```
// Function to delete the root from the max heap
// arr[]: array representing the heap
// n: pointer to the size of the heap

void deleteRoot(int arr[], int *n){
    // get the last element of the heap
    int lastElement = arr[*n-1];
    // replace the root with the last element
    arr[0] = lastElement;
    // decrease the size of the heap
    (*n--);
    // heapify the root node to maintain the heap property
    heapify(arr, *n, 0);
}

// Utility function print the array of size n
void printArray(int arr[], int n){
    for(int i=0; i<n; ++i)
        printf("%d", arr[i]);
    printf("\n");
}

int main(){
    int arr[] = {10, 5, 3, 2, 4};
    int n = sizeof(arr)/sizeof(arr[0]);
    deleteRoot(arr, &n);
    printArray(arr, n);
    return 0;
}
```

$$A^0 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ 1 & 0 & 3 & \infty & 5 \\ 2 & 2 & 0 & \infty & 4 \\ 3 & \infty & 1 & 0 & \infty \\ 4 & \infty & \infty & 2 & 0 \end{array}$$

$$A_1 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ 1 & 0 & 3 & \infty & 5 \\ 2 & 2 & 0 & 9 & 7 \\ 3 & \infty & 1 & 0 & 8 \\ 4 & \infty & \infty & 2 & 0 \end{array}$$

$$A_2 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ 1 & 0 & 3 & \infty & 5 \\ 2 & 2 & 0 & 9 & 4 \\ 3 & 3 & 1 & 0 & 5 \\ 4 & 5 & \infty & & 0 \end{array}$$

9
6