Page No. [ ] Date :

Aim : To study implement sparse matrix algorithm and program.

Theory :

A matrix is a two-dimentional data object made of 'm' rows and 'n' columns, therefore having total mxn values. If most of the elements of the matrix have '0' values, then it is called a sparse matrix.

Advantage of using sparse matrix instead of simple matrix:

Storage : There are lesser non-zero elements than zeros and thus lesser memory can be used to store only those elements.

2. Computing Time : computing time can be saved by logically designing a data structure traversing only non-zero elements.

e.g

0	0	3	0	4
0	0	5	7	0
0	0	0	0	0
0	2	6	0	0

PAVAN

Date :

Page No. [ ]

Sparse matrix representation can be done in many ways:

1. Array representation
2. Linked list representation

1] Using Array

2D array is used to represent a sparse matrix in which there are three rows named as "Rows, Columns, Value"

- Row : Index of row, where non-zero element is located.
- Column : Index of column, where non-zero elements are located.
- Value : Value of the non-zero element located at index - (row, column)

$$\begin{bmatrix} 0 & 0 & 3 & 0 & 4 \\ 0 & 0 & 5 & 7 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 6 & 0 & 0 \end{bmatrix} \rightarrow$$

Row	0	0	1	3	3
column	2	4	2	1	2
Value	3	4	5	2	6

PAVAN

Date :

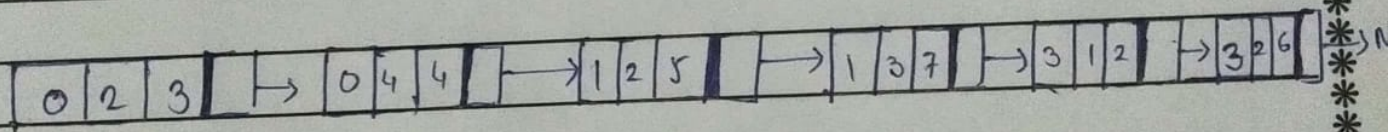
Page No. [ ]

2) Using linked list

In linked list each node has four fields. These four fields are "Row, column, value, next-node".

- Row : Index of row, where non-zero element is located.
- column : Where the non-zero elements located
- nex node : Address of the next node

0	0	3	0	⌀
0	0	5	7	0
0	0	0	0	0
0	2	6	0	0



Row	column	value	Address of next node

Node structure

Result: Hence, we have studied and implemented the operation on sparse matrix by using algorithm and progedm.

**********(PAVAN)**********

Page No. [ ]

Date :

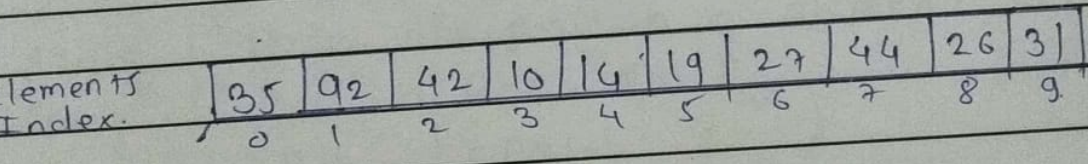
Aim: To study & implemented the operate of array.

Defination of array:

An array is a collection of data items of the same data type. accessed using a common name

A dimensional array is like a list and two dimensional array is like table, the language place ii limits on the no. of dimensions in array through specific implementation may.

Representation of array:



(size)(10)

Operation on array:

(1) Creat ()
(2) insert ()
3) delete ()
4) display ()

Vipul

Page No. [ ] Date :

1) create ()

```
create ()
{
Print ("Enter no. of element of array");
Scanf ("%d", &n);
printf ("enter the elements");
for (i=0; i<n; i++);
{
scanf ("%d", a[i]);
}
}
```

2) Insert()

```
Insert()
{
printf ("Enter the position of element");
Scanf ("%d", &pos, &n);
for (i=n-1; i>pos-1; i--)
{
a[i+1] = a[i]
}
a[pos-1] = b;
n = n+1;
}
```

Vipul

3) delete ()

```
delete ()
{
printf ("enter the position to delete")
scanf ("%d", &pos);
b = a[pos];
for (i = pos; i<n-1; i++)
{
a[i] = a[i+1];
}
n = n-1;
}
```

4] Display ()

```
main ()
{ {
printf ("enter the elements");
for (i=0; i<n; i++)
{
scanf ("%d", a[i]);
printf ("%d" \n", a[i]);
```

Conclusion :-

Hence we have studied and implemented the operation on array.

Aim : To study and implement operation on binary search.

Theory :

Binary Search is the most popular search algorithm. It is efficient and also one of the most commonly used technique that is used to solve problems.

- Binary search works only on a sorted list or set of elements. To use binary search on a collection, the collection must first be sorted.

Syntax :

low = mid + 1

mid = low + (high - low) / 2

How binary search works :

Let us assume that we need to search the location 31 using binary search.

10	14	19	26	27	31	33	35	42	44
0	1	2	3	4	5	6	7	8	9

- By using the formula we will determine half of the given array

mid = low + (high - low) / 2

So here 4 is the mid of the array

- Now we compare the value stored at location 4, with value being searched i.e '31'
- We find 27 at 4th location. which is not match
- As the value is greater than mid i.e '27' and we have sorted array, so we can also know the target value that must be upper position or portion of the array.

- We change our low to mid+1 and find new mid value again.

low = mid + 1
mid = low + (high - low) / 2

- Now mid becomes '7' again it does not match.
- So it should be lower portion
- We calculate the mid again. This time it become 5 i.e '31' is sorted at 5th location.

Pseudocode :-

The pseudocode of binary search algorith :-

```
Procedure binary-search
A ← sorted array
n ← size of array
x ← value to be searched

  Set lowerBound = 1
  set upperBound = n

while x not found
    if upperBound < lowerBound
        Exit : x doest not exists

  Set midpoint = lowerBound + (upperBound - lowerBound) /2

    if A[midPoint] < x
      Set lowerBound = midPoint + 1

    if A[midPoint] > x
      set upperBound = midpoint - 1

    if A[midpoint] = x
      Exit : x found at location midpoint
end while
end procedure
```

PAVAN

Result : Hence, we have studied and implemented binary search operation.

**********(PAVAN)**********

Aim : To study the different sorting Techniques by using algorithm

1] Insertion sort:

Analysis of algorithm

```
For j = 2 to A length
key = A [j]
// Insert A [j] into sorted sequence
  i = j - 1
while i > 0 and A [i] > key
  A [ i+1 ] = A [i]
  i = i - 1
  A [i+1] = key
```

Theory

Def^n :- Insertion sort is a sorting algorith in which the elements are transfered are at a time to the right position

Insertion sort is a simple sorting algorithm that builds the final sorted array (or list)

one item or a time. It is much less efficient on large lists than more advanced algorithms such as quicksort. heapsort or meaze sort.

2. Selection Sort.

Algorithm.

```
Void sort (int list [ ], int n)
  {
    int i, j, min, temp;
    for (i = 0; i < n-1; i++)
    {
      min = i;
      for (j = i+1; j < n; j++)
        if (list [j] < list [min])
          min = i;
      Swap (list [j], list [min], temp)
    }
    Swap (x, y, t)
    {
      t = x;
      x = y;
      y = t;
    }
  }
```

Theory :-

The selection sort is a combination of searching and sorting. During each pass the unsorted element with the smallest (or largest) value is made to its proper position in the array.

The number of times the sort passes through the array is one less than the number of items in the array.

Selection sort is a simple sorting algorithm initially the sorted part is empty and unsorted part is the entire list.

The smallest element is selected from the insorted array and swapped with the leftmost element. and that element becomes a part of the sorted array.

EX :-

14	33	27	10	35	19	42	44

14	33	27	10	35	19	42	44

10	33	27	14	35	19	42	44

10	33	27	~~33~~ 14	35	19	42	44

Page No. [ ] Date :

10	33	27	14	35	19	42	44

10	14	27	33	35	19	42	44

10	14	27	33	35	19	42	44

10	14	27	33	35	19	42	44

10	14	19	33	35	27	42	44

10	14	19	33	35	27	42	44

10	14	19	27	35	33	42	44

10	14	19	27	35	33	42	44

10	14	19	27	35	33	42	44

10	14	19	27	33	35	42	44

10	14	19	27	33	35	42	44

Conclusion:

Hence We have studied and Implimented the different sorting techniques.

Aim : To study and implement programs on singly linked list

Theory :

Linked list is a linear data structure. It is a collection of data elements, called nodes pointing to the next node by means of a pointer.

In linked list each node consists of its own data and the address of the next node and forms a chain.

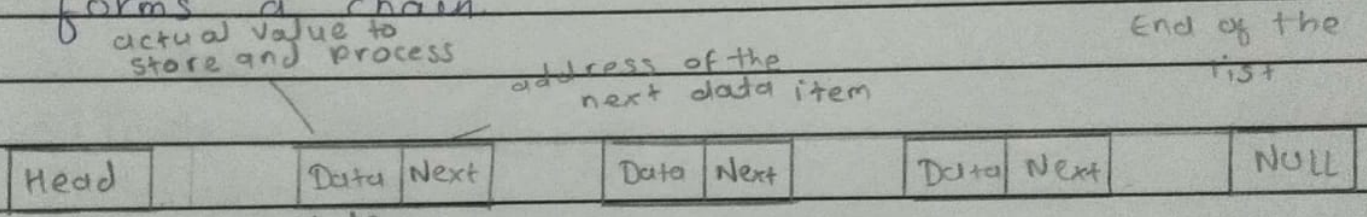


fig linked list

Linked list contains a link element called first and each link carries a data item. Entry point into the linked list is called the head of the list.

A node is declare as below:

```
Typedef struct node
{
  int data;
  struct node *next;
  } node;
  node *head = NULL;
```

Following are the operations that can be performed on a linked list:

1) creat
2) Insert
3) Delete
4) Traverse
5) Search
6) Concatenation
7) Display

1) Creating first node

```
head = (node*) malloc (sizeof(node));
head -> data = 20;
head -> next = NULL;
```

Inserting an element

```
node *nextnode = malloc (sizeof(node));
nextnode -> data = 22;
nextnode -> next = NULL;
head -> next = nextnode;
```

Deleting the element

```
int delete (node **head, node* n );
```

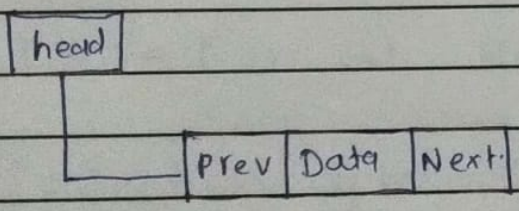
Traversing a node in a linked list:

```
Void traverse (node *head)
{
  if (head != NULL)
  {
    traverse (head -> next);
    printf ("%d", head -> data);
  }
}
```

Result : Hence we have studied and implemented all operations on linked list.

Aim : To study and implement progeams on doubly linked list.

Doubly linked list is a complex type of linked list in a node contains a pointer to the previous as well as the next node in the sequence. Therefore in a doubly link list, a node consist of three parts: node data, pointer to the next node in sequence (next pointer), pointer to the previous node (previous pointer). A sample node in a in a doubly linked list is shown in fig.



Node.

In data structure structure of a node in doubly linked list can be given as.

```
Struct node
{
    Struct node *prev;
    int data;
    Struct node *next;
}
```

Operations on doubly linked list

like single linked list operations are also can be operates on doubly linked list.

1) Insertion at beginning.
2) Insertion at end.
3) Insertion after specified node
4) Deletion at beginning
5) Deletion at the end
6) Deletion of the node having given data.
7) Searching.
8) Traversing.

**********(PAVAN)**********

Result: Hence we have studied and implemented all operations on doubly linked list.

Date :

Aim : To study and implementation of Queue

Theory :

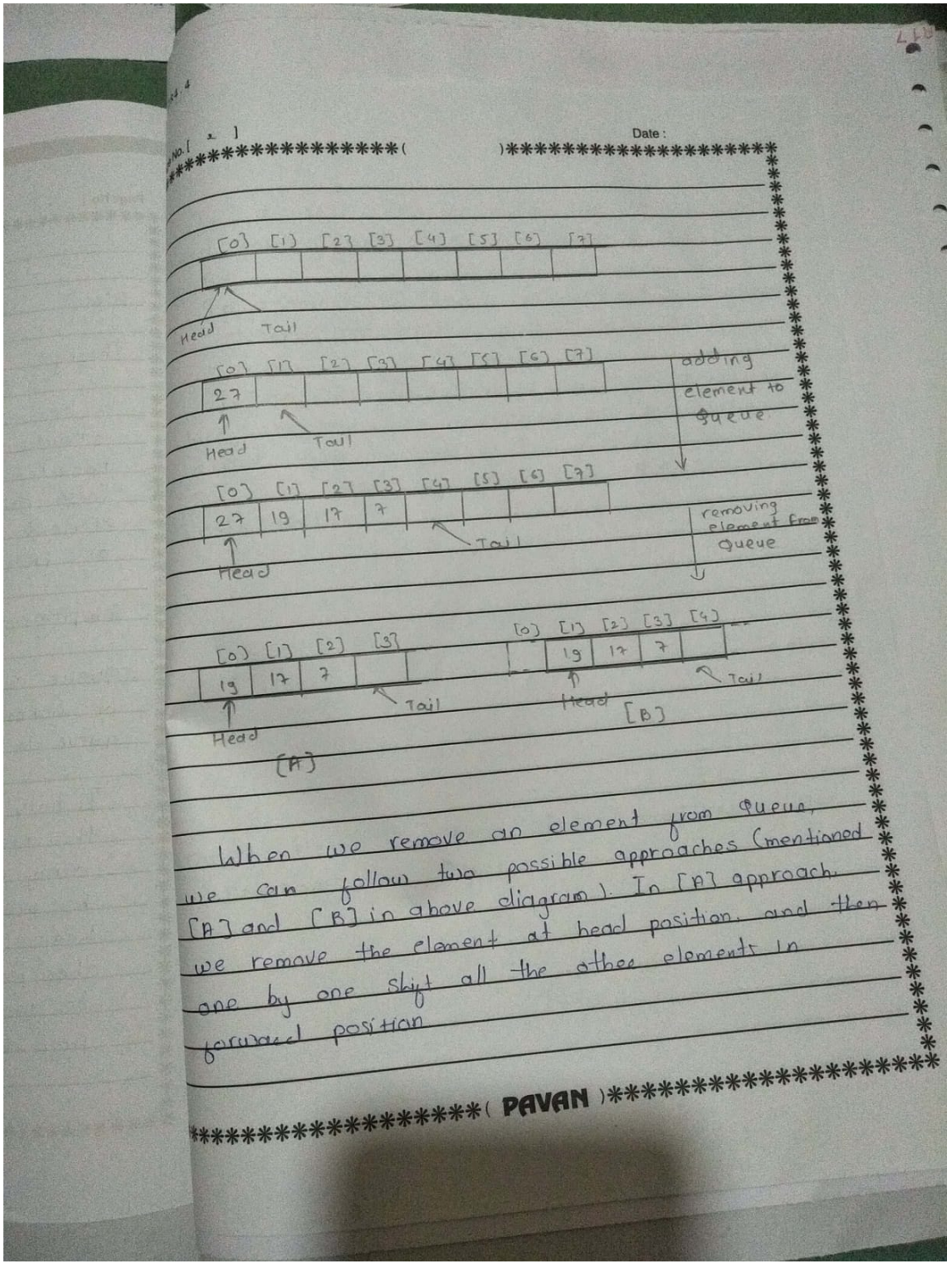
Queue is an abstract data type or a linear data structure, just like stack data structure in which the first element is inserted from one end called the REAR (also called tail), and the removal of existing element takes place from the other end called as FRONT (also called head).

Implementation of Queue

Queue can be implemented using an Array, stack or linked list. The easiest way of implementing a queue is by using an Array.

Initially the head (FRONT) and the tail (REAR) of the queue points at the first index of the array (starting the index of array from 0). As we add elements to the queue, the tail keeps on moving ahead always pointing to the position where the next element will be inserted, while the head remains at the first index.

PAVAN



When we remove an element from queue, we can follow two possible approaches (mentioned [A] and [B] in above diagram). In [A] approach, we remove the element at head position, and then one by one shift all the other elements in forward position

PAVAN

Page No. 1 > 1

Date :

In approach [B] we remove the element from head position and then move head to next position.

In approach [A] there is an overhead of shifting the elements one position forward every time we remove the first element.

In approach [B] there is no such overhead but whenever we move head one position ahead, after removal of first element, the size of Queue is reduced by one space each time.

result : Hence we have studied and implemented programs on Queue.

PAVAN

NO. [ ]

Date :

Aim : To study and implement the programs on tree

Theory :

Tree represents the nodes connected by edges. A Binary Tree is a special datastructure used for data storage purposes. A binary tree has a special condition that each node can have a maximum of two children. A binary tree has the benifits of both an ordered array and a linked list as search is as quick as in a sorted array and insertion or deletion operation are as fast as in linked list.

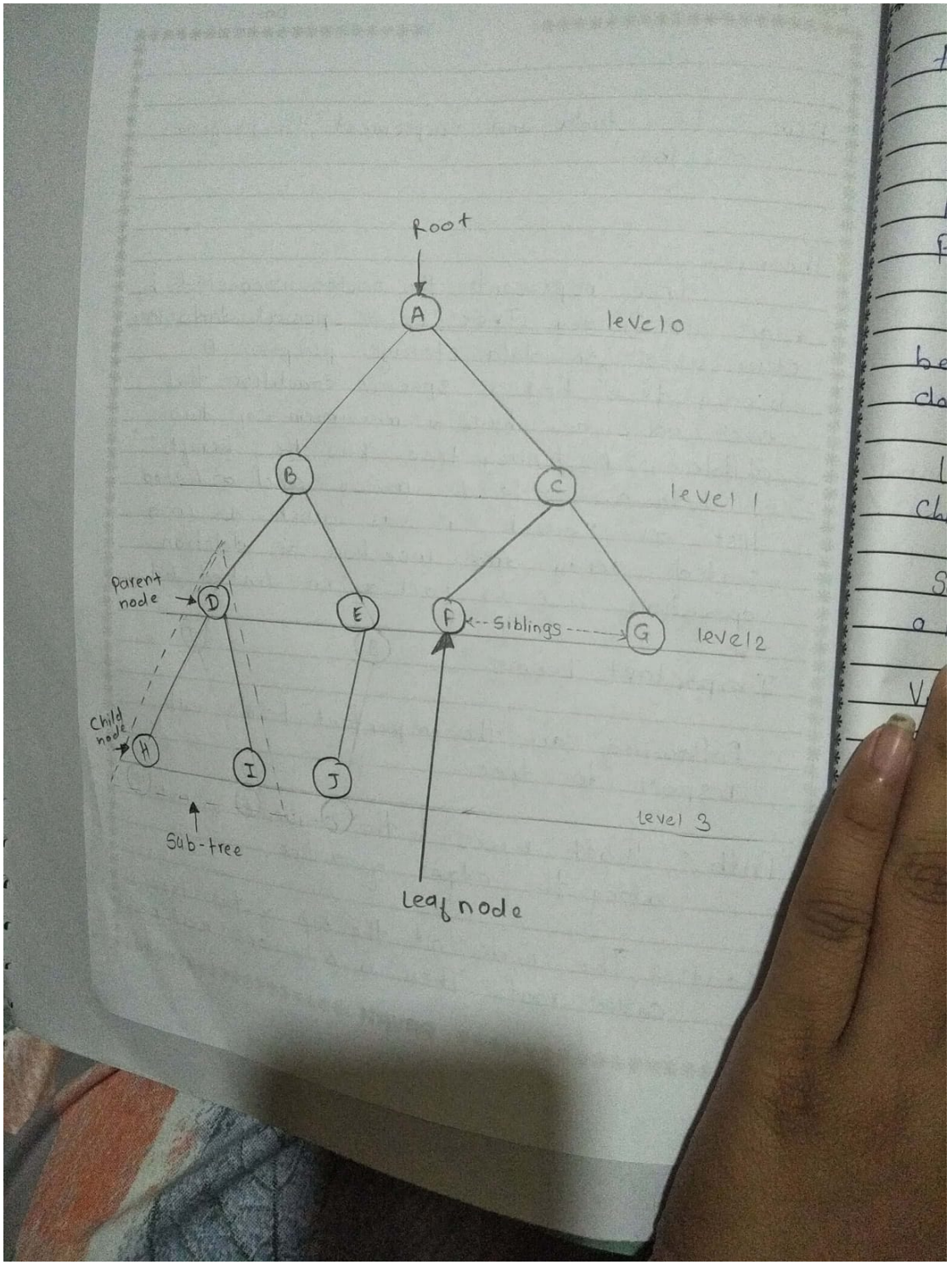
Important Terms

Following are the important terms with respect to tree.

Path : Path refers to the sequence of nodes along the edges of a tree

Root : The node at the top of the tree is called root. There is only one root per

PAVAN



Page No. [ ] Date :

tree and one path from the root node to any node.

Parent : Any node except the root node has one edge upward to a node called Parent.

Child : The node except the root node has below a given node connected by its edge downward is called its child node.

Leaf : The node which does not any child node is called the leaf node.

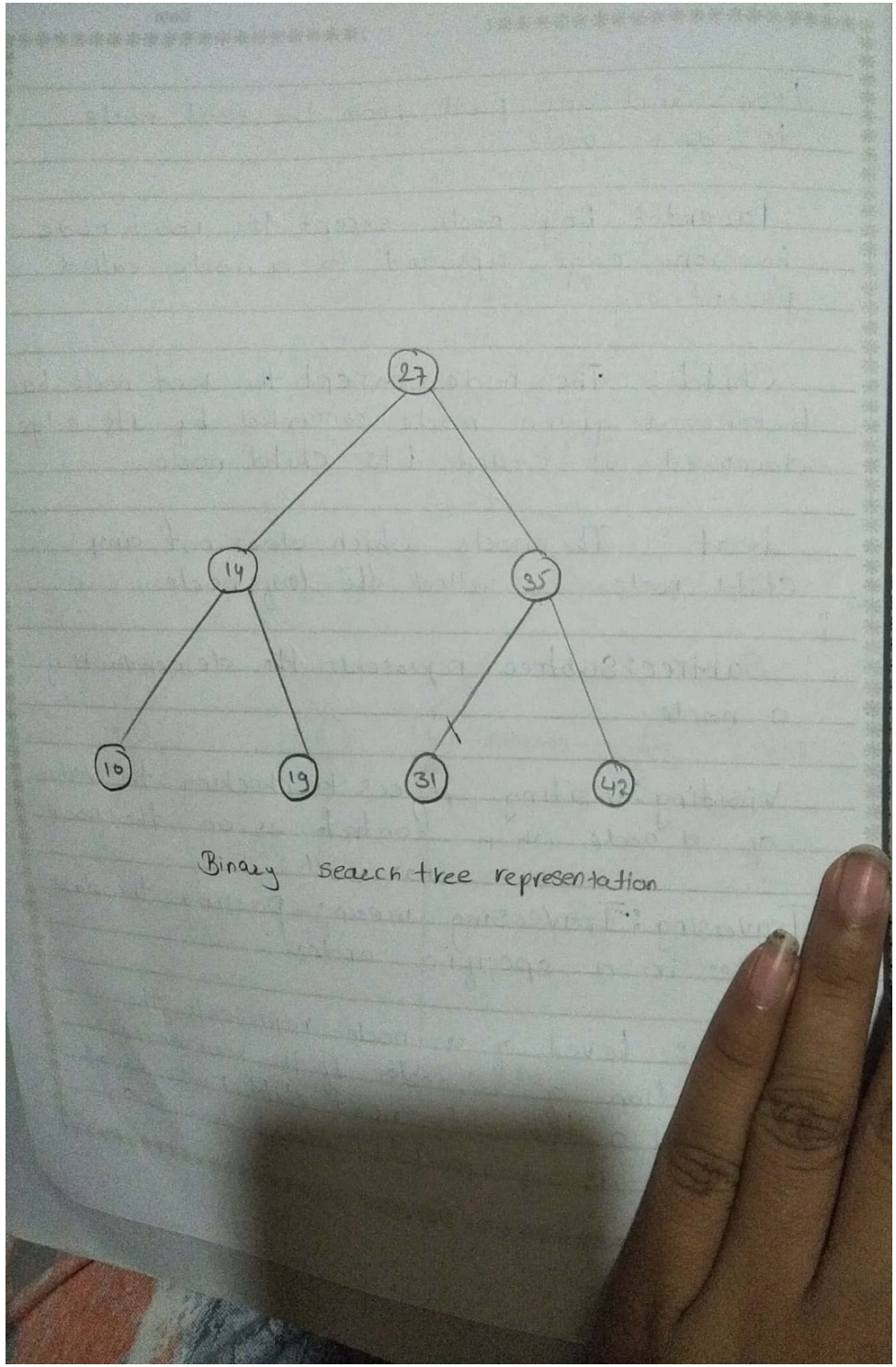
Subtree : Subtree represents the descendants of a node.

Visiting : visiting refers to checking the value of a node when control is on the node.

Traversing : Traversing means passing through nodes in a specific order.

Levels : Level of a node represents the generation of a node. If the root node is at level 0, then its next child node is at level 1, its grandchild is at level 2, and

PAVAN



Binary search tree representation

so on.

keys :- key represents a value of a node based on which a search operation is to be carried out for a node

Binary Search Tree Representation

Binary Search tree exhibits a special behavior. A node's left child must have a value less than its parent's value and the node's right child must have a value greater than its parent value.

Tree node.

The code to write a tree node would be similar to what is given below. It has a data part and references to its left and right child nodes.

```
struct node {
    int data;
    struct node * leftchild;
    struct node * rightchild;
};
```

PAVAN

Date :

Basic operations on Tree.

The basic operations that can be performed on a binary search tree b:data structure, are the following.

1] Insert 2] Search
3] Preorder Traversal 4] Inorder Traversal
5] Postorder Traversal.

result :- Hence we have studied and implemented the operation on ~~space~~ tree by using algorithm and program.