2078

1) How can you use stack to convert an infix expression to postfix? Convert infix expression (A+B)*(C-D) to postfix using stack.

⇒ Algorithm to convert infix to postfix expression using stack is given below:

i) Scan the infix string from left to right.

ii) Initialize an empty stack

iii) If the scanned character is operand, add it to postfix string.

iv) If the scanned character is operator, push the character to stack.

v) If the top operator in stack has equal or higher precedence than scanned operator then POP the operator in stack and add it to postfix string else push the scanned character to stack.

vi) If left parenthesis '(' is scanned. Push it to stack.

vii) If right parenthesis ')' is encountered, Pop and add it to postfix stack until '(' is encountered. Ignore '(' ')'.

viii) Repeat step iii) to vii) till all characters are scanned

ix) After all character are scanned POP the characters and add to postfix string from stack until it is not empty

Infix expression = (A+B) * (C-D)

Character	stack	Postfix
(	(	
A	(	A
+	(+	A
B	(+	AB
)		AB+
*	*	AB+
(	*(	AB+
C	*(	AB+C
-	*(-	AB+C
D	*(-	AB+CD
)	*	AB+CD-
		AB+CD-*

Postfix expression: AB+CD-*

2) Explain concept of divide of and conquer algorithm. Hand test quick algorithm with array of number (78, 34, 21, 43, 7, 18, 9, 56, 38, 19). What is complexity of quick sort algorithm

→ Divide and conquer algorithm breaks a problem into sub problems that are similar to the original problem, recursively solves the problem, and finally combines the solution to the subproblems to solve the original problem. Divide and conquer algorithm consist of following three steps

i. Divide : Divide original problem into set of subproblem
ii. Conquer : Solve every subproblem individually, recursively.
iii. Combine : Put together the soln of subproblem to get solution.

78, 34, 21, 43, 7, 18, 9, 56, 38, 19

78 as pivot

34, 21 43 7 18 9 56 38 19 [78]

34 as pivot

21 7 18 9 19 [34] 43 56 38

21 and 43 as pivot

7 818 9 19 [21] [38] [43] [56]

7 as pivot

[7] 18 9 19

18 as pivot

[9] [18] [19]

Sorted number : 7, 9 18, 19, 21 34, 38, 43, 56, 78

Time complexity

Best case : $T(n) = O(n \log n)$

Worst case : $T(n) = O(n^2)$

Average case : $T(n) = O(n \log n)$

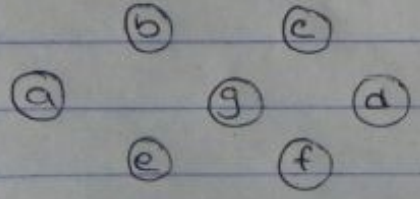
3 Discuss depth first and breadth first traversal of a graph with suitable example.

→ Traversing a graph means visiting all the vertices in a graph exactly once. In graph all nodes are treated equally. So, the starting node can be chosen arbitrarily.

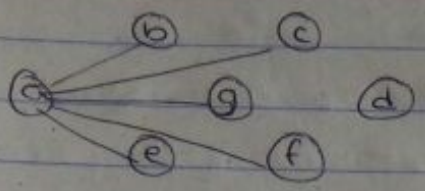
Breadth first search

The traversal starts at a node 'n', after marking the node the traversal visits all the incident edges to node 'n' and then moving to an adjacent node and repeating the process. The traversal continues until all unmarked nodes in the graph has been visited. A queue is maintained in the technique to mark node an incident edges.

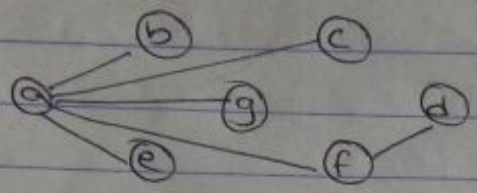
eg:



Choosing a as initial vertex

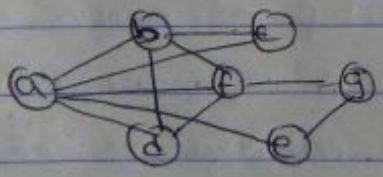


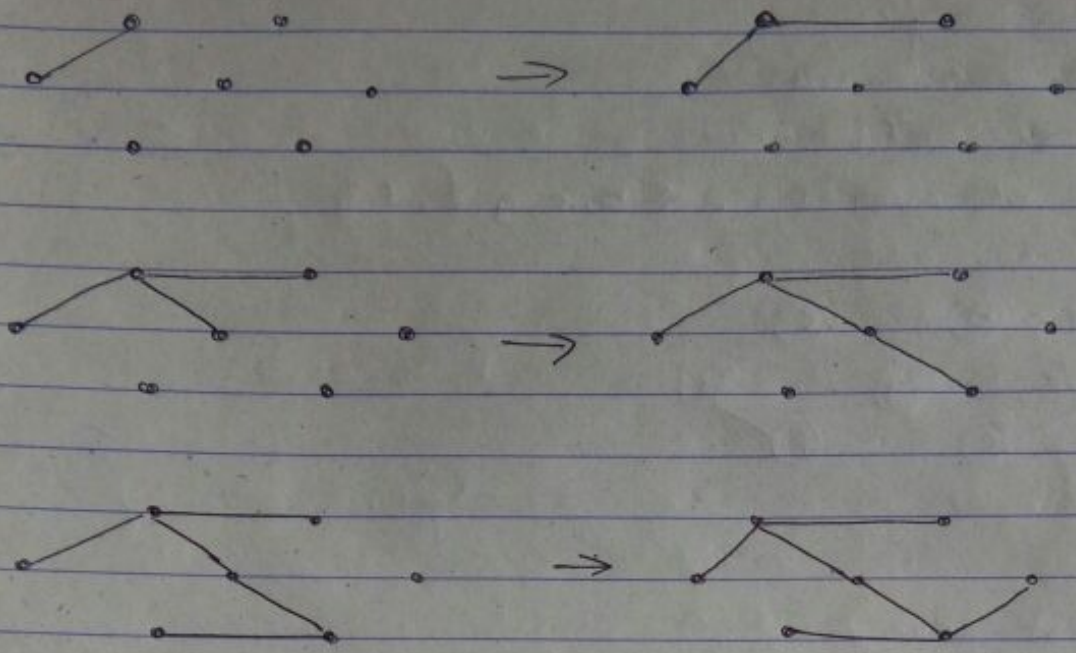
order of level d i-e $\{b, c, g, e, f\}$



Depth first scan

This technique picks up a node and marks it. An unmarked adjacent node to previous node is then selected and marked, becoming the new start node, possibly leaving the previous node with unexplored edges for the present. The traversal continues recursively until all marked nodes of the current path is visited. The process is continued for all the paths of the graph. If this cannot be done, move back to another vertex and repeat the process. The whole process is continued until all the vertices are met





4) What do you mean by complexity of algorithm? How do you find time complexity?

→ The complexity of a function f(n) which measures the time and space used by an algorithm in terms of input size n. The complexity of an algorithm is a way to classify how efficient an algorithm is, compared to alternative ones.

The process of finding complexity of given algorithm by counting number of steps and number of memory reference used by using RAM model is called detailed analysis. For detailed time complexity algorithm we count the number of steps and finally add all steps and calculate their big Oh notation.

eg:

```
#include <stdio.h>
printf ("Enter a number of to calculate factorial");
scanf ("%d", &n);
for ( i = 1; i <= n; i++)
    fact = fact * i ;
Printf (" Factorial of %d = %d", n, fact);
```

Time complexity = 1 + 1 + 1 + 1 + n + 1 + n + n + 1 + 1

= 3n + 7

= O(n)

5) Compare stack with queue. How is linear queue different from circular queue?

	Stack	Queue
1	Objects are inserted and removed from same end called top of stack	Object are inserted from rear and removed from fron
2	It follows LIFO principle i.e. Last In First Out.	It allows FIFO principle i. First In First Out.
3	Only one pointer is used which points top of stack.	Two pointers are used for front and rear.
4	Stack are visualized as vertical collection.	Queue are visualized as horizontal collection.
5	eg: collection of plates.	eg: line at bus stop.

Linear Queue	Circular queue
1 A linear data structure that stores data as a sentence of element similar to a real world queue.	A linear data structure in which the last item connects back to the first item forming circle.
2 Possible to enter new items from rear and remove from front.	Possible to enter and remove elements from any position.
3 Requires more memory.	Requires less memory.
4 Less efficient	More efficient.

5 What is ADT ? Discuss stack as ADT.

→ Abstract Data Types are a set of data values and associated operations that are precisely independent of any particular implementation. Generally the associated operations are abstract methods. eg: stack queue etc.

Stack as an ADT

A stack of elements of type T is a finite sequence of elements of T together with the operations.

- Create Empty Stack (S): Creates an empty stack.
- Push (S, x): Inserts x at top of stack.
- Pop (S): Deletes the element from top of stack.
- Top (S): Retrives top element from stack.
- IsFull (S): Checks if the stack is full.
- IsEmpty (S): Checks if the stack is empty.

7) Define recursive algorithm. How do you implement recursive algorithm while writing computer programs?

→ A recursive algorithm is an algorithm which calls itself.

We can implement recursive algorithm by using following steps:

i) Initialize the algorithm. Recursive programs often need a seed value to start. This is accomplished either by using a parameter passed to the function or by providing a gateway function that is non recursive but sets the the seed value for recursion.

ii) Check whether the current value being processed match the base case. If so, process and return the value.

iii) Redefine the answer in simpler sub-problem.

iv) Run the algorithm on sub problem.

v) Combine the results in the formula of the answer.

vi) Return the result.

8) What are the benefits of using linked list over array? How can you insert a node in a singly linked list?

→ The benefits of using linked list over array are:

i) Linked list are dynamic data structure. i.e. they can grow or shrink during the execution of program.

ii) They have efficient memory utilization. Since memory is not pre allocated, it allocates memory when required and deallocates when task is finished.

iii) Insertion and Deletion are easier and efficient.

iv) Complex applications can be easily carried out with

linked list.

v) There is no need to define initial size of linked list.

vi) It reduces the access time.

Inserting a node at the beginning:

i Create a new node using malloc function

New Node= (Node Type*) malloc (size of (Node type));

ii) Assign data to info field.

NewNode→info = newItem;

iii) Set next of new node to head

New node → next= head;

iv) Set the head pointer to new node

head = Newnode.

v) End

g) How do you implement binary search algorithm? What is time complexity of this algorithm.

→ Binary search is an extremely efficient search technique that searches the given item in minimum possible comparison in already sorted list of given element.

The logic behind this technique is

1) First find the middle element of the list.

2) Compare the mid element ~~of~~ within searched item. There are three cases:

a) If it is desired element then search is successful.

b) If it is less than desired element then search is in first half.

c) If it is greater than desired item then search in second half.

The time complexity of this algorithm is

$T(n) = T(n/2) + O(1)$

$= O(\log n)$

Q) What is hashing? Discuss rehashing with example.

→ Hashing is the process of mapping large amount of data into a smaller table. We store key value in hash table after applying the hash function.

Rehashing is a technique in which the table is resized i.e the size of table is doubled by creating a new table.

Rehashing is performed when

- table is completely full
- quadratic probing when table is half full.
- when insertion fails due to overflow.

Eg: 37, 90, 55, 22, 17, 49, 87 at size 10

H(key) = key mod tablesize

37 % 10 = 7

90 % 10 = 0

55 % 10 = 5

22 % 10 = 2

17 % 10 = 7

49 % 10 = 9

Now this table is almost full and if we insert more element collision will occur and eventually further

insertion will fail. Hence we will rehash by doubling table size. The old table size is 10 so we will double the table size. Since 20 is not prime we prefer 23 as new size.

11 How do you traverse a binary tree? Discuss.

→ The tree traversal is a way in which each node in the tree is visited exactly once in a symmetric manner. The binary tree can be traversed in 3 ways:

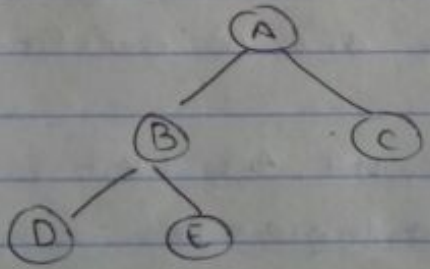
i) Pre-order traversal

ii) In-order traversal

iii) Post order traversal

i) Preorder traversal

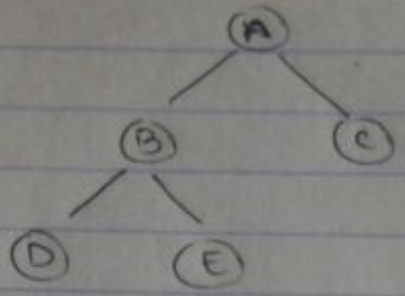
In pre-order traversal, the root node is ~~vite~~ visited f then its left child and then right child.



Preorder: A B D E C

ii) Inorder traversal

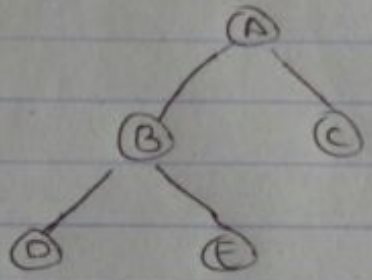
In this traversal the left node is visited first, then root node and then right child.



Inorder : D B E A C

iii) Post order traversal

In this traversal the left child is visited first, then the right child and then root node.



Post order : D E B C A

12) Write short notes on

a) Dynamic memory allocation

Dynamic memory allocation is the process of assigning memory space during the execution time or run time. There are four library functions malloc( ), calloc( ), free( ) and realloc( ) for memory management.

2074

1) Illustrate the algorithm for Binary Search tree with example.

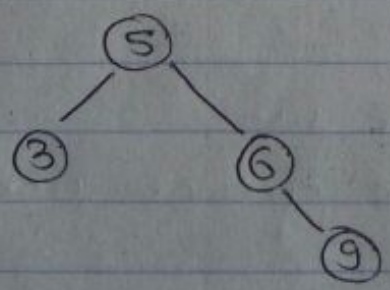
⇒ A binary search tree is a binary tree in which every node contains a key value and satisfies following conditions:

- All keys in the left subtree of root are smaller than root node.
- All keys in the right subtree of root are greater than root node.
- The left and right subtree of root are again binary search tree.

Algorithm for insertion

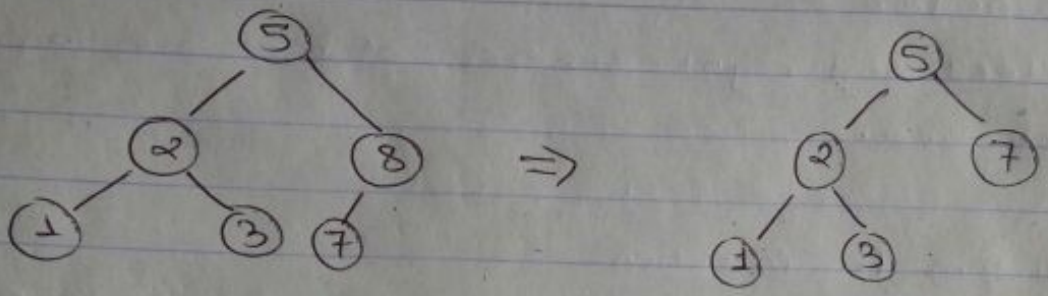
- If the tree is empty, create a new node n storing the key x and make the new node n the root of this tree.
- Otherwise, compare x to the key stored in root R.
- If x is identical to key in root R, then do not change the tree.
- If x is smaller than key in root R, insert the new item into the subtree of the root (recursively)
- Otherwise insert the new item into the right subtree of the current node.

eg: inserting 9



Deletion from binary search tree

- If the node is a leaf, it can be deleted immediately.
- If the node has one subtree, the node can be deleted by replacing the node with the root of subtree.
- If the node has two subtree, we replace the node with the rightmost node of left subtree or leftmost node of right subtree.



Deleting 8 from the graph.

Ans 2 What do you mean by circular list? Differentiate between stack as a circular list and queue as circular list.

→ A circular linked list is a list where the link field of last node points to the very first node of the list.

Stack as a circular list

Structure for the circular linked list

```
struct node
{ int info;
  struct node *next;
};
```

```
typedef struct node NodeType;
NodeType *pstack = NULL;

Push function
void Push (int item)
{ NodeType newnode;
  newnode = (NodeType*) malloc (sizeof(NodeType));
    newnode -> info = item;
    if (pstack == NULL)
    { pstack = newnode;
      pstack -> next = pstack;
    }
    else
    { newnode -> next = pstack -> next
      pstack -> next = newnode;
    }
}

Pop function
void POP ()
{ NodeType *time;
  if (pstack == NULL)
  { printf (" Stack underflow ");
    exit (1);
  }
  else if (pstack -> next == pstack)
  { printf (" popped item %d", pstack -> info);
    pstack = NULL;
```

```
    }
    else
    { temp = pstack -> next;
      pstack -> next = temp -> next;
      printf(" popped item = %d ", temp -> info);
      free (temp)
    }
  }
```

Queue as a circular list

Insertion function

```
void insert (int item)
{ NodeType *nnode;
  nnode = (NodeType*) malloc (sizeof (NodeType));
  nnode -> info = item;
  if (pq == NULL)
    pq = node;
  else
  { nnode -> next = pq -> next;
    pq -> next = nnode;
    pq = nnode;
  }
}
```

Deletion function

```
void delete (int item)
{ NodeType * temp;
  if (pq == NULL)
  { printf ("void deletion");
    exit (1);
  }
  else if (pq -> next == pq)
  { printf ("popped deleted item = %d", pq -> info);
    pq = NULL;
  }
  else
  { temp = pq -> next;
    pq -> next = temp -> next;
    printf ("deleted item = %d", temp -> info);
    free (temp);
  }
}
```

3 Explain the procedure for construction of Huffman algorithm with example.

→ Huffman algorithm is a technique of compressing data to reduce its size without losing any of the details. It is generally useful to compress the data in which there are frequently occuring characters.

Procedure for creating Huffman tree

i) Calculate the frequency of each string.

ii) Sort the characters in increasing order of the frequency. These are stored in priority queue.

iii) Make each unique character as a leaf node.

iv) Create an empty node z. Assign the minimum frequency to the left child of z and assign second minimum frequency to the right child of z. Set the value of z as the sum of two above minimum frequency.

v) Remove two minimum frequency from Q and add the sum into list of frequency.

vi) Insert node z into tree.

vii) Repeat step (iii) to (iv) for all the characters.

viii) For each non-leaf node, assign 0 to the left edge and 1 to right edge.

Eg: let us take four characters and frequency

Character	Frequency
A	06
B	03
C	02
D	01

Sorting them according to frequency

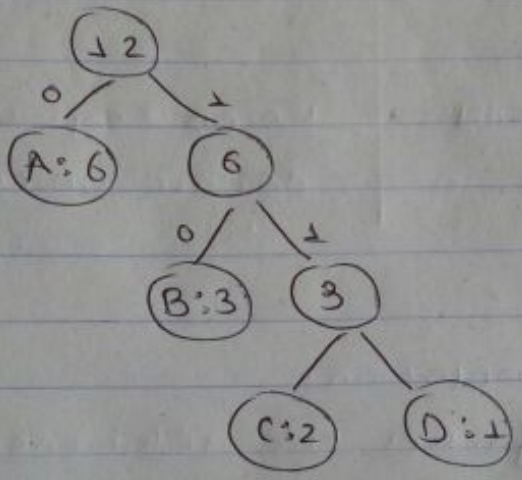
D	01
C	02
B	03
A	06

i) (D : 1) (C : 2) (B : 3) (A : 6)

ii) (3) (B : 3) (A : 6)

iii) (6) (A : 6)

iv)



4 Difference between structure and union.

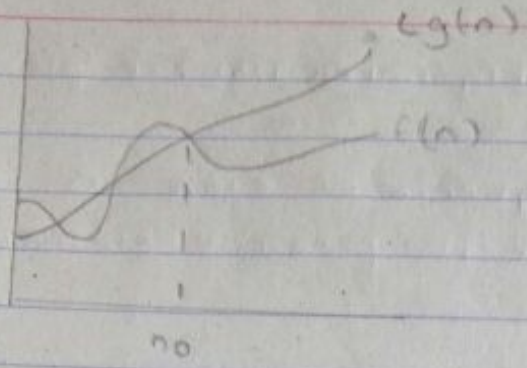
→

Structure	Union
We can access all the members of structure at anytime.	Only one member of union can be accessed at anytime.
Memory is allocated for all variable.	Allocates memory for variable which variable requires more memory.
All member of structure can be initialized.	Only the initial member of union can be initialized.
Struct keyword is used to declare structure	Union keyword is used to declare union.

Describe the Big 'O' notation.

→ Big 'O' notation signifies the relationship between the input to the algorithm and the steps required to execute the algorithm.

A function f(n) is said to be big-Oh of g(n) and we write f(n) = O(g(n)) or simply f = O(g) if there are two positive constants c and n_0 such that $f(n) \leq c * g(n)$ for all $n >= 0$.



$f(n) = O(g(n))$

7) Explain the Tower of Hanoi (TOH) with practical example.

→ Tower of Hanoi (TOH) is a mathematical puzzle which consist of three pegs named as origin, intermediate and destination and more than one disc. These disks are of different sizes and smaller one sits over the larger one.

Algorithm for TOH

lets consider we need to move 'n' disks from peg A to peg C using intermediate peg B.

i) Assign three pegs A, B & C.

ii) If n == 1

Move single disk from A to C and stop.

iii) If n > 1

a) Move the top (n-1) disks from A to B

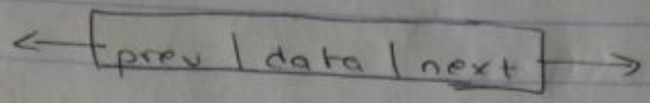
b) Move the remaining disks from A to C

c) Move the (n-1) disks from B to C.

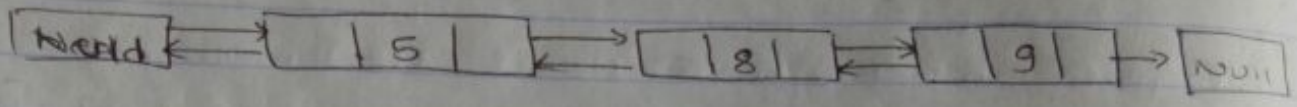
iv) End.

8 What do you mean by doubly linked list? Explain with example.

→ Doubly linked list is a sequence of elements in which every element has links to its previous element and next element in sequence.



Eg:



- In doubly linked list, the first node must be always pointed by head.
- Always the previous field of the first node must be NULL.
- Always the next field of last node must be NULL.

9 What are the types of binary tree? Compare between them.

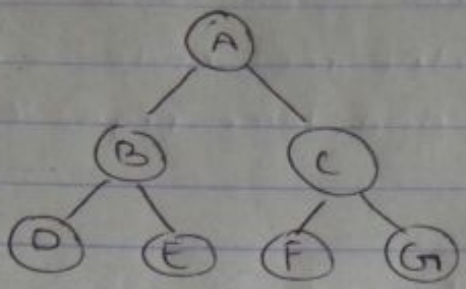
→ A binary tree is a finite set of elements that are either empty or partioned into three disjoint ~~set~~ subs
- The first subset contains a single element called root of tree.
- The two other subsets are left and right sub-tre of original tree.

There are three types of binary tree.

1) Complete binary tree

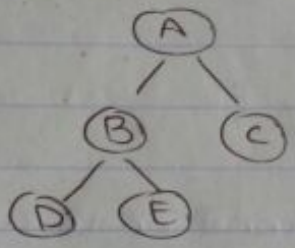
A binary tree in which every internal node has exactly two child and all leaf nodes are at

same level is called complete binary tree.



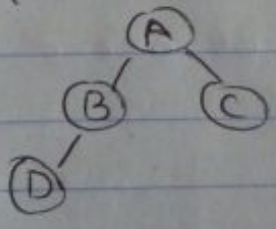
2) Strictly binary tree

A binary tree in which every node has either two or zero number of child is called strictly binary tree.



3) Almost completely binary tree

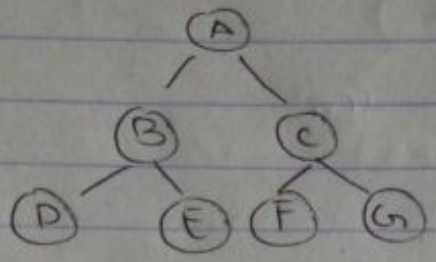
Almost completely binary tree is a binary tree in which every level of tree is completely filled except last level. Also, the last level nodes should be attached from left most position.



Q. Differentiate between pre-order & inorder traversal.

Pre-order traversal

In pre order traversal, the root node is visited at first, then left child and later right child.

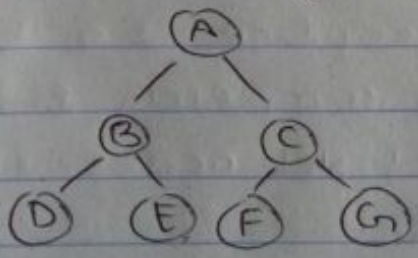


The pre order traversal of above tree is

A B D E C F G

Inorder traversal

In inorder traversal root node is visited between left and right child. Here left node is visited first then root node and right node.



The inorder traversal of above tree is

D B E A F C G

11) What do you mean by sorting? Explain the Bubble sort with example.

→ Sorting is a technique to arrange the items of list in any specific order which is either ascending or descending.

Bubble sort is a sorting method where each element in the list is compared to its successor and they are interchanged if they are not in proper order.

eg:

5, 1, 4

= 1, 5, 4

= 1, 4, 5

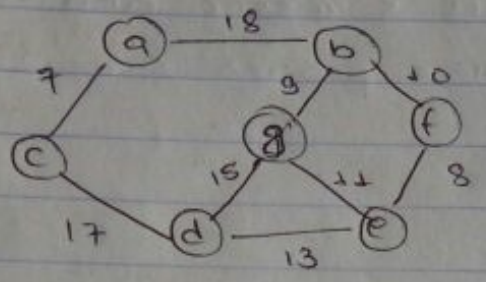
12) Differentiate between sequential searching & binary searching.

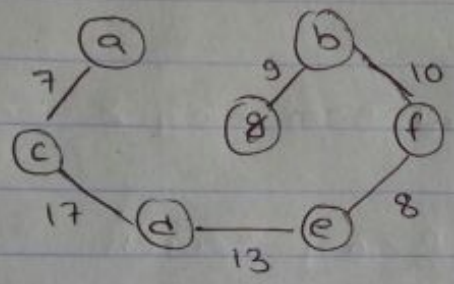
→

Sequential search	Binary search
• Time complexity: O(n)	Time complexity: O(log n)
• Eliminates only one element from search per comparison.	Eliminates half of the array elements per comparison
• Doesn't require element to be sorted.	Require elements to be in sorted order.
• Easy to program but takes too much time for execution.	Easy to program and execution is faster.

13) Discuss the Kruskal's algorithm with example.

→ Kruskal's algorithm is used to find the minimum spanning tree for a connected weighted graph. The main target of this algorithm is to find the subset of edges by using which, we can traverse every vertex of the graph.





2072

3) What is circular queue? Write an algorithm and C function to implement circular queue.

→ Circular queue is a linear data structure in which the operations are performed based on FIFO principle and last position is connected to first position to make a circle.

Algorithm for insertion in circular queue
Assuming that rear and front are initially set to Maxsize -1

1. If (front == (rear + 1) % Maxsize)
   print queue is full and exit
   else
   rear = (rear + 1) % Maxsize
2. cQueue [rear] = item
3. end

```
void enqueue (int value)
{ if ((front == 0 & rear == size-1) || (front == rear+1))
   printf (" Circular queue is full");
   else
    if ( rear == size-1 && front != 0)
    { rear = -1;
      cQueue[++rear] = value;
      printf (" Insertion success");
      if (front == -1);
        front = 0;
    }
}
```

Algorithm for deletion

Assuming that rear and front are initially set to Maxsize - 1.

1) if (rear == front)
   print queue is empty and exit
2) front = (front + 1) % Maxsize
3) item = cqueue [front];
4) return item;
5) end

```
void dequeue ( )
{ if (front == -1 && rear == -1 )
  printf(" queue is empty" );
  else
  { printf (" deleted item : %d ", cQueue [front++]);
    if (front == size)
        front = 0;
    if (front - 1 == rear)
        front = rear = -1;
  }
}
```

4) What is Recursion ? write a recursive algorithm to implement binary search.

→ Recursion is the process by which a function calls itself repeatedly, until some specific condition has been satisfied.

Recursive algorithm for binary search

```
int binSearch (int A[], int low, int high, int x)
{ if (low > high)
   { return -1;
   }
   int mid = (low+high) /2;
    if(x == A[mid])
    { return mid;
    }
   else if (x< A[mid])
   { return binsearch (A, low, mid-1, x);
    }
   else
   { return binSearch (A, mid+1, high, x);
   }
}
```

5) Differentiate between array and pointer

→

Array	Pointer
• It is a collection of homogenous data element.	It is a variable which stores the address of another variable.
• Array can be initialized at definition.	Pointers can't be initialized at definition.
• They are static in nature. Once memory is allocated, it cannot be resized or freed dynamically	They are dynamic in nature. The memory allocation can be resized or freed dynamically.

6 What is an algorithm?. Write down the feature of algorithm..

→ An algorithm is a sequence of instructions used for solving a problem, which can be implemented on a computer.

The features of algorithm are:

- It takes input from external.
- It produce output as result.
- Every statement in an algorithm must be clear and unambigious.
- For all different cases, the algorithm must produce finite number of steps.
- Correct set of output value must be produced.

9 Write an algorithm and c function for merge sort.

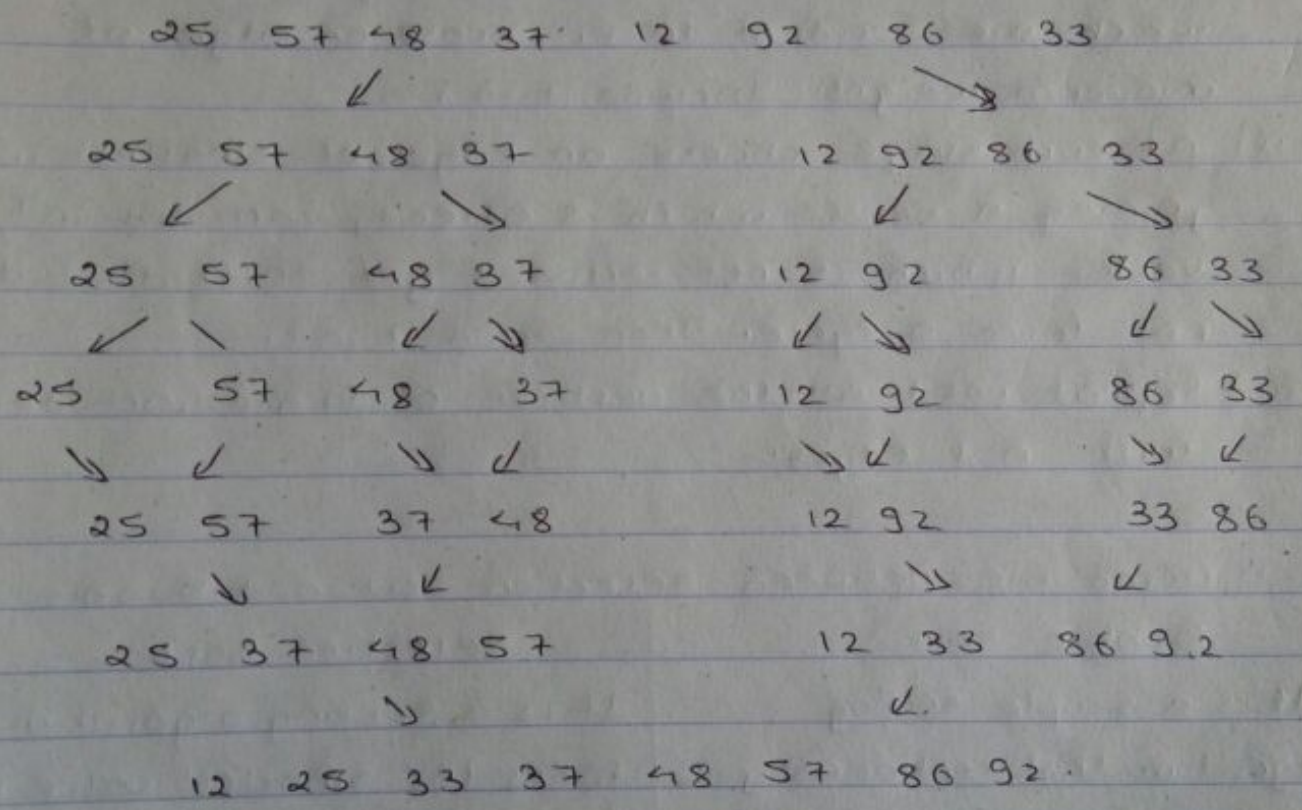
→ Algorithm

i) If the no. of item is 0 or 1, return. Else

ii) Divide the array into two half

iii) Recursively mergesort both half separately

iv) Merge the two sorted half.



10. what do you mean by graph traversal? Explain prim's algorithm with example.

→ Traversing is a ~~gra~~ process where each node is visited only once.

Prim's Algorithm is used to find minimum spanning tree from a graph.

i) Select any vertex ~~G~~ in G. Among all the edges which are incident to it, choose an edge of minimum weight. Include it in T.

ii) At each stage, choose an edge of smallest weight joining a vertex which is already included in T and a vertex which is not included yet so that it does not form a cycle. Then includ it in T.

iii) Repeat until all the vertices of G are included with n-1 edges.

11) Differentiate between selection sort and bubble sort.

→

Bubble sort	selection sort
i) It is a simple sorting algorithm that continously steps through the list & ~~searches~~ compares the adjacent pairs to sort the element.	It is a sorting algorithm that takes the smallest value in the list and moves it to proper position.
ii) Less efficient	More efficient
iii) Exchange of item is done.	Items are selected and placed
iv) It is slower.	It is faster.

2071

2) What is linked list? Explain the process of inserting and removing nodes from a linked list.

⇒ A linked list is a collection of nodes, where each node consists of two parts

info: the actual element to be stored in the list. It is also called data field.

link: one or two links that points to next and previous node.

A node can be inserted in a linked list at the beginning and end.

1) Inserting a node at the beginning.

i) Create a new node using malloc function
ii) Assign data to the info of new node
iii) Set next of new node to pHead.
iv) Set the head pointer to point new node
v) End

2) Inserting at the end

i) Create a new node using malloc function.
ii) Store data in the newly allocated node.
iii) Set next of new node to NULL.
iv) If pHead = NULL then set pHead = pNewNode and exi
v) Set loc = *pHead
vi) while (loc -> next != NULL)
loc = loc -> next
vii) Set loc -> next = pNewNode
viii) End

Deleting a node

1) Deleting first node.
   i) If link list is empty print "Empty list" and exit.
   ii) Store the address of first node in a temporary pointer ptemp
   iii) Set phead to next of phead
   iv) Free the memory reserved by ptemp.
   v) Return phead.

2) Deleting last node.
   i) if *phead == Null Print "Empty list" and Exit.
   ii) if phead → next == Null then Set P = phead
       phead = NULL
       print 'item'.
       Free the node (P)
   iii) Set p = p head
   iv) while (p → next != NULL)
       Set loc = p;
       Set p = p → next
       Set loc → next = NULL
       Free (p)
   v) End.

Sno 4 Discuss array as an ADT.

→ Let A be an array of type T and has n elements then it satisfies the following operations.

- Create (A) : Create an array A.
- Insert (AX): Insert an element x into array A in any location
- Delete (A, x): Delete an element x from array A.
- Traverse (A): Access all the element of an array A.
- Merge (A,B) : Merge elements of A and B into third array C.

Thus we can use array to perform above operation as an ADT.

5 Transform the postfix expression AB-C+DEF-+$ to infix.

→

Character	Stack	~~Infix~~ Operation
A	A	
B	A, B	
-		(A-B)
C	C	(A-B)+C
+		
D	D	
E	D, E	
F	D, E, F	
-		(A-B)+C $ (E-F)
+		(A-B)+C $ D+(E-F)
$		((A-B)+C) $ (D+(E-F))

6 What is recursion? write a recursive program to find factorial of a number.

→ Recursion is a process by which a function calls itself repeatedly until some specific condition has been satisfied.

Program to find factorial

```
#include <stdio.h>
void main()
{ int factorial (int);
  int n, f;
  printf ("Enter a number:");
  scanf("%d", &n);
  f= factorial (n);
  printf("Factorial of %d is %d", n, f);
}
int factorial (int n)
{ int f;
  if(n==1)
    return 1;
  else
    f= n * factorial (n-1);
    return f;
}
```

Q.8 Write a program to sort an array using selection sort.

→

```
#include <stdio.h>
int main()
{ int a[10], i,j, temp, min, loc, n;
  printf(" Enter no of elements to sort: ");
  scanf("%d", &n);
  printf(" Enter the elements");
  for (i=0; i<n; i++)
  { scanf("%d", &a[i]);
  }
  for (i=0; i<=n-1; i++)
  { min = a[i]
    loc = i;
    for (j=i+1; j<=n-1; j++)
    { if (a[j] < min)
      { min = a[j];
        loc = j;
      }
    }
    if (loc != i)
    { temp = a[i];
      a[i] = a[loc];
      a[loc] = temp;
    }
  }
  printf("The sorted array is: ");
  for (i=0; i<n; i++)
  {
```

```
    printf ("%d ", a[i]);
  }
  return 0;
}
```

10) Why do we need Hashing? Discuss linear probing in detail.

→ Hashing is a well known technique to search any particular element among several elements.
It minimizes the number of comparisons while performing the search. In hash table, data is stored in an array format, where each data value has its own unique index value. Access of data becomes fast if we know the index of desired data. Thus, using hashing insertion and searching are very fast irrespective of the size of the data. So we need hashing.

Linear probing is a method of handling collision. When collision occurs i.e when two data demand for the same place in the hash table, then collision can be solved by placing the second data linearly down when an empty place is found.

12 Hand test the insertion sort algorithm with following array of numbers.

16 7 31 2 9 41 -10

i) 7 16 31 2 9 41 -10

ii) 7 16 31 2 9 41 -10

iii) 2 7 16 31 9 41 -10

iv) 2 7 9 16 31 41 -10

v) 2 7 9 16 31 41 -10

vi) -10 2 7 9 16 31 41

2077

1) What is stack? What are the different applications of stack? Explain stack operation with example.

→ A stack is an ordered collection of items into which new items may be inserted and deleted from only one end which is called the top of stack.

Stack can be used in following fields:

- To evaluate the expression (postfix/prefix)
- To keep the page visited history in a web browser.
- To perform undo sequence in a text editor.
- Used in recursion
- To pass the parameter between the function in a C program.
- Can be used as a component of other data structure.
- Can be used as an auxiliary data structure for implementing algorithm.

The following operations can be performed on stack:

1) Push operation

The push operation is used to add elements in a stack. When we add an item to stack it is pushed into the stack. The last item put into the stack is at the top.

```
void push ()
int item;
if (top == Maxsize - 1)
   printf("\n The stack is full");
else
$ printf(" Enter the element : ");
```

```
        scanf("%d", &item);
        top+= 1;
        stack[top]= item;
        }
    }
```

ii) Pop operation

The pop operation is used to remove or delete the top element from the stack. When an item is popped from the stack, it is always the top item which is removed.

```
    void pop()
    { int item;
      if (top < 0)
        printf("The stack is empty");
      else
      { item = stack[top];
        top= top-1;
        printf("The popped item is %d", item);
      }
    }
```

ii) Is Full operation

The isfull operation is used to check whether the stack is full or not (i.e. stack overflow)

```
    int isempty(stack *s)
    { if(s → top<0)
        return 1;
```

```
else
    return 0;
}
```

iv) IsFull

```
int isfull (stack *s)
{ if (s -> top == Max - 1 )
    return 1;
  else
    return 0;
  }
```

iv) Is Empty operation

This operation is used to check whether the stack is empty or not (i.e stack underflow).

```
int isempty (stack *s)
{ if (s -> top < 0)
    return 1;
  else
    return 0;
  }
```

2) Difference between singly linked list and doubly linked list. How do you insert and delete a node from doubly linked list? Explain

Singly linked list	Doubly linked list
1) Singly linked list has one way direction.	Doubly linked list has two way direction.
2 It uses less memory	It uses more memory.
3 In singly linked list each node contains data and link to next node.	In doubly linked list each node contains data and link to next and previous node.
4 Complexity of insertion & deletion at known position is O(n).	Complexity of insertion & deletion at known position is O(1)
5 eg: stack	eg: tree, heap

Inserting a node in doubly linked list

1) Inserting a node at the beginning
   i) Allocate memory for new node.
   ii) Assign value to info field of new node.
   iii) Assign Left and Right link to null.
   iv) Make the right link of the new node to point to head node of the list and make left link of head node to point to the new node.

v) Finally reset head pointer i.e make it point to the new node which is inserted at the beginning.

2) Inserting a node at the end.
   i) Allocate memory for new node
   ii) Assign value to the info field of the new node
   iii) Set left and right links to the new link.
   iv) If list is not empty then traverse to the end and make right link of last node to point to the new node and left link of new node to point the last node.

Deleting a node in doubly linked list

1) Deleting a node from the beginning
   i) If the list is empty then display the message "Empty list" and exit.
   ii) Otherwise, Make the head pointer to point the second node and if the second node is not null then make its left ~~node~~ link to point to null.
   iii) Free the first node.

2) Deleting a node from the end.
   i) If the list is empty then display the message "Empty list" and exit.
   ii) Otherwise, traverse the list till the last but not one and make the right list of the last but one node to point null.
   iii) Free the node.

3) What is shortest path? Explain Dijkstra algorithm for finding shortest path using suitable example.

→ In graph theory, the shortest path is a method of finding a path between two vertices (or nodes) in a graph such that the sum of the weight of its constituent edges is minimized.

The algorithm is as follows

i) we assume all the vertices ~~of~~ are at ∞ distance from the source.

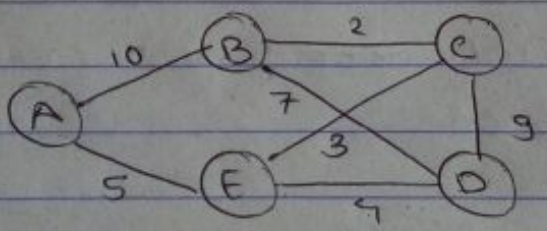
ii) Then if $(d(u) + edge) < d(v))$

$d(v) = d(u) + edge$; where u is source
v is destination

iii) we find the distance for all adjacent nodes then choose min $\{d(u_1)\, d(u_2) \cdots d(u_n)\}$

iv) Then repeat for adjacent vertex that is selected until we reach the destination.

eg:



	A	B	C	D	E
A	[0]	∞	∞	∞	∞
E		10	8	9	[5]
C		10	[8]	9	
D		10		[9]	

Shortest path from A to D : AED = 5+4 = 9.

6) Explain queue as an ADT.

→ A queue of elements of type T is a finite sequence of elements of T together with the operation.

- Make Empty (Q) - Create an empty queue Q
- Is Empty (Q) - Determine if the queue Q is empty or not. Returns true if Q is empty, false otherwise.
- Is Full (Q) :- Determine whether Q is full or not. Return true if Q is full, false otherwise.
- Enqueue (Q, n) - Insert element n at the end of queue Q
- Dequeue (Q) - If queue Q is not empty, remove the element at the front of the queue.
- Front (Q) - Retrive the element at the front of the queue Q, without deleting it.

7) Write a recursive program to find GCD of two number.

→
```
#include <stdio.h>
int gcd (int n1, int n2)
int main ()
{ int n1, n2;
  printf ("Enter two positive integers: ");
  scanf ("%d %d", &n1, &n2);
  printf ("GCD of %d and %d is %d", n1, n2,
          gcd (n1, n2));
  return 0;
}
```

```
int gcd (int n1, int n2)
  { if (n2 != 0)
      return gcd (n2, n1%n2);
      else
      return n1;
   }
```

8) What is linked list? How is it different from array?

→

Array	Linked list
• Size of an array is fixed.	Size of a list is not fixed.
• Memory is allocated from stack	Memory is allocated from heap.
• It occupies less memory than linked list for same number of elements.	It occupies more memory.
• It is necessary to specify the number of elements during declaration.	It is not necessary to specify the number of elements during declaration because it can be during run time.
• Insertion and deletion of element from given position is difficult.	Insertion and deletion is easy as only link needs to be managed.

9 Hand test bubble sort with array of number 53, 42, 78, 3, 5, 2, 15 in ascending order.

→ 53 42 78 3 5, 2, 15

42 53 78 3 5 2 15
42 53 78 3 5 2 15
42 53 3 78 5 2 15
42 53 3 5 78 2 15
42 53 3 5 2 78 15
42 53 3 5 2 15 78
42 53 3 5 2 15 78
42 3 53 5 2 15 78
42 3 5 53 2 15 78
42 3 5 2 53 15 78
42 3 5 2 15 53 78
3 42 5 2 15 53 78
3 5 42 2 15 53 78
3 5 2 42 15 53 78
3 5 2 15 42 53 78
3 5 2 15 42 53 78
3 2 5 15 42 53 78
2 3 5 15 42 53 78

10. What is hashing? Explain the concept of hash table and hash function with example.

→ The process of mapping large amounts of data into a smaller table is called hashing.

A hashing table is a data structure where we store a key value after applying the hash function. It is arranged in the form of an array that is addressed via a hash function. The hash table is divided into number of bucket and each bucket is in turn capable of storing a number of records.

A function that transformes a key into a table index is called a hash function. If h is a hash function and key is a key h(key) is called the hash of key is the index in which a record with the key should be placed.

eg. key = 8, 15 at mod 10

$h(k) = 8 \% 10 = 8$

$h(k) = 15 \% 10 = 5$

11. What is minimum spanning tree? Explain.

→ Given a ~~minimum~~ connected and undirected graph, a spanning tree of that graph is a subgraph that is a tree and connects all the vertices together. A graph can have many different spanning tree.

A minimum spanning tree for weighted, connected and undirected graph is a spanning tree with a weig less than weight of all other spanning tree.