Converted By SP GAMER HAND

ACADEMY OF TECHNOLOGY

AEDCONAGAR, HOOGHLY – 712121

Question Bank of Data Structure and Algorithms (PCC-CS301)

Department of CSE and IT

Semester: 3rd

GROUP-A

Multiple Choice Questions (each question carries 1 mark) (40 X1) =40

1. The memory address of the first element of an array is called
   a) Floor address
   b) Foundation address
   c) First address
   d) Base address
2. The expression that accesses the (i, j) th entry (i=0,1,…m-1,j=0,1,…n-1 of an m×n matrix (stored in column major order) is
   a) n × (i-1)+j b) m×(j-1)+I c) n × (i-1)+ (j-1) d) m×(j-1)+ (i-1)
3. In C language arrays are stored in which representation?
   a) Column major b)Row major c)Layer major d)None of these
4. An abstract data type (ADT) is
   a) Same as an abstract class
   b) A data type that cannot be instantiated
   c) A data type for which only the operations defined on it can be used but none else
   d) None of the above
5. Stack is used in
   a) Recursion b) Invoking functions c) All of the above d) None of the above
6. The other name for prefix notation is
   a) Reverse polish b) Polish c) Infix d) None of the above
7. Stack can be implemented using
   a) Arrays b) Linked lists c) both a & b d) None of the above
8. In Stack data structure, insertions and deletions are made at
   a) both ends b) one end only c) middle of the stack d) None of the these

9. Which of the following stack operations could result in stack underflow?
   a) Is_empty b) Pop c) Push d) Two or more of the above answers.
10. Which data structure is needed to convert infix notations to postfix notations?
    a) Linear list b) Queue c) Tree d) Stack

11. Stack is ……
    a) a linear data structure b) a nonlinear data structure c) None of the above

12. Stack is also called as
    a) FIFO b) LIFO c) both (i) and (ii) d) None of the above

13. The number of elements that can be removed from the stack at any time is ……
    a) 3 b)4 c)1 d) 0

14. Which data structure is used in evaluating mathematical expressions with parentheses?
    a)Stack b) Queue c) Tree d) Graph

15. Which of the following applications may use a stack?
    a) A parentheses balancing program. b) Keeping track of local variables at run time.
    c) Syntax analyzer for a compiler. d) All of the above.

16. Suppose you opened a notepad, a music player, an excel sheet, and also you are doing your data structure programming simultaneously. Your OS implements which data structure for it.
    a) Stack b) Queue c) Tree d) Linked List
17. Simulations are implemented using _____.
    a) Stack b) Queue c) Linked List d) Tree

18. Insertion in stack is done in
    a) front b) rear c) top d) bottom

19. Which data structure is used to manage Printer Buffer?
    a) Stack b) Queue c) Linked List d) Tree
20. Which of the following is essential for converting an infix expression to postfix notation?
    a) A parse tree b) An operand stack
    c) An operator stack d)None of these

21. Ascending priority queue is one in which the item removed is
    a) the smallest item b) the largest item c) any item

22. Descending priority queue is one in which the item removed is
    a) the smallest item b) the largest item c) any item

23. One difference between a queue and a stack is:
    a) Queues require linked lists, but stacks do not.
    b) Stacks require linked lists, but queues do not.
    c) Queues use two ends of the structure; stacks use only one.
    d) Stacks use two ends of the structure, queues use only one.

24. The deque can be used
    a) as a stack b) as a queue c) both as a stack and as a queue d) None of the above

25. Priority queue can be implemented using
    a)Array b)Linked list c)Heap d)All of these

26. If data is a circular array of CAPACITY elements, and rear is an index into that array, what is the formula for the index after rear?
    a) (rear % 1) + CAPACITY b) rear % (1 + CAPACITY)
    c) rear + 1) % CAPACITY d) rear + (1 % CAPACITY)

27. Which of the following data structure may give overflow error, even though the current number of elements in it, is less than its size.
    a) Simple queue b) Circular queue c) Stack d) None of the above.

28. A linear list in which elements can be added or removed at either end but not in the middle is known as
    a)Stack b) Queue c) Dequeue d) Heap

29. The rear and front end of a linear queue is used for
    a) deletion, insertion
    b) searching, sorting
    c) insertion deletion
    d) none of these

30. Recursion may be implemented by
    a) linked-list b) stack c) queue d) dequeue

31. In external sorting technique all data reside in
    a) Primary memory b) Secondary memory c) both (a) and (b)
    d) None of these

32. Which one is the best time among the following algorithms?
    a) O(n) b) $O(\log_2 n)$ c)$O(2^n)$ d) $O(n\log_2 n)$

33. Average case time complexity of quick sort
    a) $O(N \log_2 N)$ b) $O(N \log N)$ c) $O(N^2)$ d)$O(N^3)$

34. The complexity of merge sort algorithm is
    a) O(n) b) $O(n\log n)$ c) $O(n^2)$ d) $O(\log n)$

35. Which of the following sorting methods requires extra space, than the data to be sorted?

a) Selection sort b) Heap sort c) Bubble sort d) None of these

36. The worst case complexity of binary search is

a) O(log n) b) O(nlog n) c) O(n) d) $O(n^2)$

37. The best case time complexity of bubble sort technique is

a) O(n) b) $O(n^2)$ c) O(nlog n) d) O(log n)

38. What will be the time complexity for selection sort for an array of n elements?

a) O(log n) b) O(nlog n) c) O(n) d) $O(n^2)$

39. Best case time complexity of insertion sort is

a) O(1) b) O(n) c) O(nlog n) d) $O(n^2)$

40. The time complexity of binary search is

a) $O(n^2)$ b) O(n) c) O(log n) d) O(n log n)

GROUP-B

Multiple Choice Questions (each question carries 2 marks) (30 X2) =60

41. Each element of an array arr [20][50] requires 4 bytes of storage. Base address of arr is 2000. The location of arr[10][10] when the array is stored as column major fashion is
    a) 2820 b) 2840 c) 4048 d) 4840

42. Base address of a floating point 2D array A is 2000.a is stored in row- major order in memory, lower limit is adapted as 0 and the dimension of a is 4 and 5 respectively. What will be the address of A[2][3] ?
    a) 2022 b)2052 c)2026 d)2044

43. For a strictly lower triangular matrix the element a_{ij}, where i is the row and j is the column position respectively, is 0 for
    a) i≠ j b) i<= j c) i> j d) i< j

44. The following sequence of operations is performed on a stack: push(1,) push(2),pop, push(1), push(2), pop, pop, pop, push(2),pop. The sequence of popped out values are
    a) 2,2,1,2,1 b) 2,2,1,1,2 c) 2,1,2,2,1 d) 2,1,2,2,2

45. A stack is implemented using an array with the following declaration:
    int stack [100]; int stacktop=0;
    To perform the POP operation, which of the following is correct?
    a) x=stack [stacktop++] b) x= stack [++stacktop]
    c) x=stack [stacktop--] d) x= stack [--stacktop]

46. What is the value of the postfix expression 6 3 2 4 + - *:
    a) Something between -15 and -100 b) Something between -5 and -15
    c) Something between 5 and -5 d) Something between 5 and 15

47. The postfix equivalent of the prefix *+ab-cd is
    a) ab+cd-* b) abcd+-* c) ab+cd*- d) ab+-cd*

48. A reverse polish expression for the infix expression a+b*(c+d)/f+d*e
    a) ab+cd+*f/d+e* b) abcd+*f/+de*+ c) a*b+cd/f*de++ d) None of these

49. A prefix expression for the infix expression a*(b+c)/e-f
    a)/*a+bc-ef b) -/*+abcef c) -/*a+bcef d) None of these

50. Translating the infix expression(P) into postfix notation, we get
    [P = A + (B* C – (D / (E + F)) * G) * H]
    a) ABC * DEF / + G *– H * +
    b) ABC * + DEF + / – G * H – +
    c) ABC * DEF + / G *– H * +
    d) None of these

51. Suppose we have an array implementation of the stack structure, with ten items in the stack stored at data[0] through data[9]. The CAPACITY is 42. Where does the push method place the new entry in the array?
    a) data[0] b) data[1] c) data[9] d) data[10]

52. The five items A, B, C, D, E are pushed in a stack, one after the other starting from A. The stack is popped four times and each element is inserted in a queue. Then two elements are deleted from the queue and pushed back on the stack. Now one item is popped from the stack. The popped item is
    a) A b) B c) C d) D

53. The evaluation of the postfix expression 3,5,7,*,+,12,% is
    a)2 b)3 c)0 d)3.17

54. If we evaluate the following postfix expression 23 5 7 * - 12 +
    The result will be
    a) 12 b) 0 c) -12 d) 35

55. The integers 1,2,3,4 are pushed into the stack in that order. They may be popped out of the stack in any valid order. The integers, which are popped out produce a permutation of the numbers 1, 2, 3, 4.Which of the following permutation can never be produced in such a way?
    a) 1,2,3,4 b) 4,2,3,1 c) 4,3,2,1 d) 3,2,4,1

56. If the characters 'D', 'C', 'B', 'A' are placed in a queue (in that order), and then removed one at a time, in what order will they be removed?
    a) ABCD b) ABDC c) DCAB d) DCBA

57. Let P be the queue of integers defined as follows

```
#define MAX 50
struct queue {
        int items[MAX];
        int front, rear;
} q;
```

    To insert an elements in the queue we can use
    a)++q.items[q.rear]=x; b) q.items[++q.rear]=x; c) q.items[++q.rear]++=x;

58. Suppose we have a circular array implementation of the queue structure, with ten items in the queue stored at data[2] through data[11]. The current capacity is 42. Where does the insert method place the new entry in the array?
    a) data[1] b) data[2] c) data[11] d) data[12]

59. The initial configuration of queue a,b,c,d('a' is at the front).To get the configuration d,c,b,a one needs a minimum of
    a) 2 deletions and 3 additions
    b) 3 deletions and 2 additions
    c) 3 deletions and 3 additions
    d) 3 deletions and 4 additions

60. Queue is implemented with a circular array, keeping track of front, rear, and many Items (the number of items in the array). Suppose front is zero, and rear is one less than the current capacity. What can you tell me about many Items?
    a) Many Items must be zero.
    b) Many Items must be equal to the current capacity.
    c) Count could be zero or the capacity, but no other values could occur.
    d) None of the above.

61. A machine needs a minimum of 100 sec to sort 1000 names by quick sort. The minimum time needed to sort 100 names will be approximately
    a) 72.7 sec b) 11.2 sec c) 50.2 sec d) 6.7 sec

62. Which of the following shows the correct relationship among some of the more common computing times for algorithm?
    a) $O(\log n) < O(n) < O(n*\log n) < O(2^n) < O(n^2)$
    b) $O(n) < O(\log n) < O(n*\log n) < O(2^n) < O(n^2)$
    c) $O(n) < O(\log n) < O(n*\log n)) < O(n^2) < O(2^n)$
    d) $O(\log n) < O(n) < O(n*\log n) < O(n^2) < O(2^n)$

63. What is the Big Oh notation of the following expression $F(n)=n \log n^2 + n^2 + e^{\log n}$
    a) O(n) b) $O(n^2)$ c) $O(n \log n^2)$ d) $O(e^{\log n})$

64. The running time of an algorithm T(n),where n is the input size is given by
    $T(n) = 8\,T(n/2) + qn$, if $n>1$
    $= p$, if $n=1$
    where p and q are constants. The order of this algorithm is
    a) n^2 b) n^n c) n^3 d) n

65. Consider the following queue of characters, where queue CQ is a circular array which is allocated 5 memory cells: Front = 2, Rear = 3, CQ: _, P, Q, _, _.
    The content of CQ after following operations is:
    i. R is added to the queue
    ii. Two letters are deleted from the queue.
    iii. S, T, U are added to queue.
    a) T,U,_,R,S b) U,T,_,R,S c) S,T,U,R,_ d) T,U,_,Q,_

66. To implement a stack using simple queue, the required number of simple queue is
    a) 1 b) 2 c) 3 d) 4

67. A single array A [1.....MAXSIZE] is used to implement two stacks which grow from opposite ends of array. Variables top1 and top2 (top1<top2) point to the location of topmost element in each of the stacks. If the space is to be used efficiently, the condition of "stack-full" is
    a) top1=(MAXSIZE)/2 and top2=(MAXSIZE/2)+1
    b) top1+top2=MAXSIZE
    c) top1=(MAXSIZE)/2 or top2=MAXSIZE
    d) top1=top2-1

68. The required number of steps for solving Tower of Hanoi having 5 disks is
    a) 32 b) 15 c) 31 d) 16

69. The given array is arr = {11, 22, 44, 33}. Bubble sort is used to sort the array elements. How much iteration will be done to sort the array?
    a) 4 b) 2 c) 1 d) 0

70. A machine needs 200 sec to sort 200 names by bubble sort. In 800 sec,it can approximately sort _____ names.
    a) 400 b) 800 c) 750 d) 600

ACADEMY OF TECHNOLOGY

AEDCONAGAR, HOOGHLY – 712121

MCQ Question Bank of Data Structure & Algorithms (PCC-CS301)

Department of CSE and IT
Semester: 3rd

GROUP-A

Multiple Choice Questions (each question carries 1 mark)

1) malloc returns _____ pointer.
   a) void
   b) null
   c) integer
   d) a structure typed
2) calloc returns _____ pointer.
   a) void
   b) null
   c) integer
   d) a structure typed
3) Function used for the deallocation of memory is ___.
   a) malloc
   b) calloc
   c) free
   d) realloc
4) Generic pointer is ___ pointer.
   a) void
   b) wild
   c) null
   d) integer
5) Time complexity of TOH game (with n number of discs) is ___.
   a) $O(n^2)$
   b) $O(\log_2 n)$
   c) $O(n^3)$
   d) $O(2^n)$
6) Recursion may be implemented by
   a) linked-list
   b) stack
   c) queue
   d) dequeue

7) In Linked list, the logical order of elements
   a) is same as their physical arrangement
   b) is determined by their physical arrangement
   c) is not necessarily equivalent to their physical arrangement
   d) none of these

8) Address field-part of last node of a singly linked list is ___.
   a) address of the node positioned immediate before last node
   b) address of the first node of the singly linked list
   c) null pointer
   d) wild pointer

9) Previous address field-part of first node of a doubly linked list is ___.
   a) address of the last node of the doubly linked list
   b) address of the node positioned immediate after the first node
   c) null pointer
   d) wild pointer

10) A singly linked list can be converted into a circular linked list if ___.
   a) address of the first node is the content of address field-part of last node
   b) reversing the entire list
   c) address field-part of last node is null pointer
   d) cannot be converted in any circumstance

11) Underflow condition in linked list may occur when attempting to :
   a) insert a new node when there is free space for it
   b) delete a nonexistent node in the list
   c) delete a node in empty list
   d) none of these

12) Which is similar to a stack operation?(with respect to a singly linked list)
   a) insertion of node at the beginning of the list
   b) insertion of node at any position of the list
   c) deletion of node from any position of the list
   d) both of a),c)

13) Which is the correct notation to delete the last node p from a doubly linked list? (prev is the pointer to the previous node and next is the pointer pointing to the next node)
   a) p->next=NULL
   b) p=NULL
   c) p->prev->next=NULL
   d) none of these

14) Find the correct option for getting an ascending order sorted sequence of data from a doubly linked list.
(x is a pointer pointing to first node of corresponding doubly linked list)
a) x->data > x->next->data
b) x->data<x->prev->data
c) x->data<x->next->data
d) both of a),b)

15) Find appropriate data structure with respect to the operations of singly linked list.
Operation-A: Insertion at beginning of the list and at the end of the list.
Operation-B: Deletion from beginning of the list and from the end of the list.
a) stack
b) queue
c) circular queue
d) double ended queue

16) Default initial value of memory spaces allocated by calloc is ___.
a) garbage value
b) 0
c) 1
d) can't say

17) Default initial value of memory spaces allocated by malloc is ___.
a) garbage value
b) 0
c) 1
d) can't say

18) Find the appropriate operation of singly linked list similar with a priority queue.
a) deletion from any position of the list.
b) deletion from only beginning of the list.
c) deletion from only end of the list
d) both of b),c)

19) Find out correct alternative of C of a while loop used for displaying all values of nodes of a non-empty singly linked list.
(q is a pointer pointing initially to the first node of list)

```
while(C)
{
  printf ("%d",q->data);
   q=q->next;
}
```

a) q->next==NULL
b) q!=NULL
c) q->next!=NULL
d) q== NULL

20) Inserting a new node after a given node in a doubly Linked list requires
a) 4 pointers exchange
b) 2 pointers exchange
c) 1 pointer exchanges
d) no pointer exchanges

21) In binary tree each node has ___ 2 children.
a) exactly
b) minimum
c) at most
d) less than

22) Number of child/children for root of a tree:
a) 0
b) 1
c) 2
d) More than 2

23) Maximum nodes for a binary tree with 3 levels are ___
a) 5
b) 6
c) 7
d) 8

24) For a non-empty binary tree if 9 nodes are present in the tree, then total edges for corresponding binary tree is ___.
a) 7
b) 8
c) 9
d) 10

25) What will be the root of a max-heap if we create max-heap with following numbers: 25,15,20,30,12,32,16?
a) 25
b) 32
c) 16
d) 12

26) What will be the root of a min-heap if we create min-heap with following numbers: 25,15,20,30,12,32,16?
a) 25
b) 32
c) 16
d) 12

27) The node ___ will be the root of the tree if we find following pre-order traversal sequence: D C A B E F G
a) D
b) G
c) B
d) A

28) The node _____ will be the root of the tree if we find following post-order traversal sequence: D C A B E F G
a) D
b) G
c) B
d) A

29) BST is ____ binary tree.
a) superset of
b) subset of
c) equal to
d) not related with

30) AVL tree is ____ BST.
a) superset of
b) subset of
c) equal to
d) not related with

31) AVL tree allows Balance-Factor(BF) of any node
a) 0
b) -1
c) +1
d) all of these

32) AVL tree is _____ height-balanced tree.
a) not
b) almost
c) perfectly
d) different from

33) AVL tree conducts double rotation if ____.
a) BF of a node : +2 and BF of its any child : +1
b) BF of a node : +2 and BF of its left child : +1
c) BF of a node : -2 and BF of its right child : -1
d) Both of b),c)

34) Which of following traversals is/are sufficient to create a BST from given traversal:
i) Preorder ii) inorder iii) postorder
a) Any one of given 3 traversals
b) Either inorder or postorder
c) inorder and postorder
d) preorder and postorder

35) In B+ tree all keys are distributed ____
a) among root ,internal, leaf nodes
b) only among leaf nodes
c) only among internal nodes
d) none of these

36) BFS algorithm is implemented by ____
a) stack
b) queue
c) run-time stack
d) all of these

37) For b tree of order 4 maximum children for each internal node (including root) is __.
a) 2
b) 3
c) 4
d) 5

38) For b tree of order 5 maximum keys for each node is __.
a) 2
b) 3
c) 4
d) 5

39) For a Hash-Table of size 7 the element 14 will stored in an index ____.(Divison-Remainder method)
a) 0
b) 1
c) 2
d) 3

40) In ___index, an index entry appears for only some of search key values.
a) dense
b) sparse
c) straight
d) continuous

GROUP-B

Multiple Choice Questions (each question carries 2 marks)

1. What will be steps for implementing TOH game (with 3 stands) with 4 disks and 5 disks?
   a) 16,31
   b) 16,32
   c) 15,32
   d) 15,31

2. To delete a node from a sorted (in ascending order) singly linked list we require ___ updation.
   a) 0 pointer
   b) 1 pointer
   c) 2 pointers
   d) 3 pointers

3. Match followings:
   A) recursion
   B) null Pointer
   C) typedef
   i) stack
   ii) redefinition
   iii) invalid memory address
   a) A-i;B-ii;C-iii
   b) A-i;B-iii;C-ii
   c) A-ii;B-i;C-iii
   d) A-ii;B-iii;C-i

4. Pointer responsible for corrupting memory is known as _____.
   a) void pointer
   b) generic pointer
   c) null pointer
   d) wild pointer
5. Find similarity between a singly linked list and a circular linked list.
   a) both are unidirectional
   b) both are bidirectional
   c) both contain two beginning nodes
   d) none of these
6. How do you make a doubly linked list into a singly linked list?
   a) removal of all nodes
   b) removal of previous pointer field-part for a node in doubly linked list
   c) making previous pointer field-part for a node in doubly linked list as null
   d) both of a),c)

7. Which is not an advantage of tree?
   a) Hierarchical structure
   b) Faster search
   c) Router Algorithm
   d) UNDO/REDO operations in a notepad

8. In a full binary tree if number of internal nodes is 16, then the number of leaf nodes is ____.
   a) 32
   b) 17
   c) 15
   d) 31

9. In a full binary tree if there are 32 leaf nodes, then total number of nodes is ___.
   a) 64
   b) 33
   c) 31
   d) 63

10. How many binary trees can be formed using 3 nodes?
    a) 20
    b) 30
    c) 120
    d) 5

11. For a BST we find pre-order traversal sequence: 40 30 35 42 and post-order traversal sequence: 35 30 42 40.What will be right child of 35?
    a) 30
    b) 42
    c) 40
    d) No Child

12. What will be inorder traversal sequence of BST if we insert 22,55,11,33,44 into that BST with the corresponding order mentioned above:
    a) 11 22 33 44 55
    b) 11 22 44 55 33
    c) 22 11 33 44 55
    d) 22 33 11 55 44

13. We have inserted 18, 24,21,10,12 into an empty BST following this order. If we delete 18 after the creating BST, then pre-order traversal of BST (after deleting 18) is ___.
    a) 12 10 24 21
    b) 21 10 12 24
    c) Either a) or b)
    d) None of these

14. How many BSTs can be formed using 3 nodes?
    a) 20
    b) 30
    c) 120
    d) 5

15. If we insert 45,50,75,60 into an empty AVL tree with this order, then what will be the root of AVL tree after its creation?
    a) 45
    b) 50
    c) 75
    d) 60

16. Match followings:

i. Heap	p) No child
ii. BF=+2 or -2	q) AVL tree rotations
iii. Leaf node	r) Complete Binary Tree

    a) i.-p) ; ii.-r) ; iii.-q)
    b) i.-r);ii.-q);iii.-p)
    c) i.-p);ii.-q);iii.-r)
    d) i.-r);ii.-p);iii.-q)

17. If we insert 40,55,72,63 into an empty AVL tree with this order, then value of right child of 55(after the creation of AVL tree) will be _____.
    a) 63
    b) 40
    c) 72
    d) No Child

18. If we insert 7,4,8,5 into an empty AVL tree with this order, then the number of unbalanced nodes for that newly created AVL tree in case of inserting 6 into the corresponding AVL tree is ___.
    a) 0
    b) 1
    c) 2
    d) 3

19. Consider a binary max-heap implemented using an array. Which one of the following array represents a binary max-heap?
    a) 25,12,16,13,10,8,14
    b) 25,14,13,16,10,8,12
    c) 25,14,16,13,10,8,12
    d) 25,14,12,13,10,8,16

20. Consider the data given in above question (Q19). What is the content of the array after two delete operations on the correct answer to the previous question?
    a) 14,13,12,10,8
    b) 14,12,13,8,10
    c) 14,13,8,12,10
    d) 14,13,12,8,10

21. For a b tree of order 3 with height 3 maximum keys are present in the corresponding b tree are ___.
    a) 39
    b) 26
    c) 27
    d) 24

22. Five node splitting operations occurred when an entry is inserted into a B-tree (of order m). Then how many nodes are written for that B-tree (of order m)?
    a) 10
    b) 11
    c) 12
    d) 14

23. For a b-tree of order 5 the minimum and maximum key entries of a root are ___ and ___ respectively.
    a) 2,4
    b) 2,5
    c) 1,4
    d) 1,5

24. For a b-tree of order 7 the overflow and underflow occur for a leaf node when key entries of that leaf node are ___ and ___ respectively.
    a) 3,6
    b) 6,3
    c) 6,1
    d) 7,3

25. Match followings:

i) B+ tree	x) MST
ii) DFS	y) Secondary Memory Device
iii) Prim's Algorithm	z) Stack

a) i)-x);ii)-y);iii)-z)
b) i)-y);ii)-z);iii)-x)
c) i)-z);ii)-x);iii)-y)
d) i)-y);ii)-x);iii)-z)

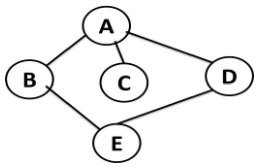
26. If we insert 14,41,30,53 in a hash table with size 6 in this order using linear probing, then hash index of 53 will be ___.
a) 0
b) 1
c) 4
d) 5

27. If we insert 14,41,30,53 in a hash table with size 6 in this order using quadratic probing, then hash index of 53 will be ___.
a) 1
b) 2
c) 3
d) 4

28. Clustered index is ___ and for clustered index the parent file must be ____.
a) sparse, ordered
b) dense, ordered
c) dense, unordered
d) sparse, unordered

29. A B+ tree can contain a maximum of 6 child pointers in a node. What is the minimum number of keys in leaves?
a) 0
b) 1
c) 2
d) 3

30. Find DFS traversal sequence for corresponding graph mentioned below:



a) A D E C B
b) A B E C D
c) A C B E D
d) All of these

INTRODUCTION

Chapter at a Glance

- **Data** (plural of **"datum"),** refers to qua rtat1ve or quant1'tative attributes of a variable or set of variables. Typically, data re the re ults of measurements and can be t e as1s of graphs images or observations of a set ofvanable_s. . of other data. Earlier term for metadata is ancillary data.
- **Metadata** is a data, containing a descnption
- **Data structure:** Data may ;be ?rgamzed in an different ways; The logical or mathematical model of a particular orgamzat1on of data is ca . d ll:d data structure.
- **Classification:** Data structure can e c assi.1 .d b tied as ***linear*** if its element forms a sequence or linear list. Line r data structures organize e ir datae emdeta elements are not organized in a sequential fashion.A data item ma no mea_ h and all the data items cannot be traversed in a single run.
- Various operations can be performed on data structure. List o some equen y use operations are given below:

1. Traversing: sometimes also called as visiting, means accessing the reqmred record so as to process the data.

2. Searching: finding the location of record to be processed.

3. Inserting: adding a new record to the struc_ture

4. Deleting: removing the record from the structure.

5. Sorting: arranging the record in some logical order.

6. Merging: combining two different sets of records to form a final set of records.

- **Abstract data type (ADT):** Abstract Data Types are a set of data values and associated operations that are precisely independent of any particular implementation. The term abstract signifies that the data type will only set the rule of it's usage but how it will be used depends on the implementation. For example stacks and queues are called abstract data types. The stack data type defines two abstract methods PUSH and POP.
- **The basic properties of an algorithm** are:

Input: Each algorithm is supplied with a zero or more external quantities.

Output: Each algorithm must produce atleast one quantity.

Definiteness: Each instruction must be clear and unambiguous.

Finite ess: The algo_rithm ust terminate afte a finite num er of steps within finite time.

Effectiveness: Each mstruct1on must be sufficiently basic and also be feasible.

- **Analysis of an algorithm** m ans, to deter:mine the amount of resources (such as time and storage) necessary to ex cute 1t. Most lgo thms are designed to work with inputs of arbitr len . U$ ally the *efficiency or runmng hme* of an algorithm is stated asa function relating the m put length to the number of steps (time complexi·ty) t t· ns (space compIex1ty). or s orage 1oca
- In theoretical analysis of algorithms it is common to est· th t the *asymptotic* sense, 1.e., to estimate the complexity function 1mate e1r comp1ext 10 c-: b. t **Big0** ,or ar 1tran1y large mpu •

notation, **omega** notation and **theta** notation are used for this Usually asymptotic esti ates are used because different implementations of the same algorithm may differ in *efficiency*. However the efficiencies of any two "reasonable" implementations of a given algorithm are related bya constant multiplicative factor calle d a *hiddencom*

Complexity: There are two types of complexities of an algorithm, time complexity and space complexity.

The ***time complexity*** of an algorithm is the amount of time the computer requires to execute the algorithm.

The ***space complexity*** of an algorithm is the amount of memory space the computer re9uires, completing the execution of the algorithm.

Multi ple Choice Type Questions

1. Each element of an array arr[20][50] requires 4 bytes of storage. Base address of arr is 2000. The location of arr[101[1O] when the array is stored as column major is
a) 2820 **b) 2840** c) **4048** **d) 4840** **[WBUT 2008)**
Answer: (b)

2. In C **language,** malloc( ) returns **[WBUT 2009]**
a) integer poi nter
b) structure pointer
c) null pointer
d) void pointer
Answer: (c)

3. Which of the following is the best time for an algo ithm? [WBUT 2010]
a) O(n) b) $O(\log_2 n)$ c) $0(2^n)$ d) $O(n\log_2 n)$
Answer: (b)

4. Four algo.s do the same task. Which algo. should execute the slowest for **large** values Qf n? [WBUT 2010]
a) $o(n^2)$ b) O(n) c) o(i g2) d) 0(2n)
Answer: (d)

5. In C language, arrays are stored in which representation? [WBUT 2011]
a) Column major b) Row major c) layer major d) None of these
Answer: (b)

6. Which of the following algorithm should execute the slowest for large values of N? [WBUT 2012]
a)O (N) b) O (N^2) c) O($\log_2$N) d) None of these
Answer: (b)

7. Which is better computing time (in **analysis of algorithm)?** [WBUT 2013]
a) O(n) **b)** 0(2n) c) O(log2n) d) None of these
Answer: (c)

8.A dynamic data structure where we can search for desired records in $Ot1$ time is

a) heap bl binary search tree 1'2.01
c) circularly linked 11st d) array

Answer: (b)

9. For_ Column Major, what is the **address** of [3, 21 th element ofa 3 x 4 a (contaans integer number)? It Is given that the •ease Address Is 2000'. [WBOT 20 13

a) 2010 b) 2012 c) 2016 d) 2018

Answer: should be 2020

10. Which of the following **expressions access** th_e (i, j)th entry ofa (mx n) matri **stored** in column major **order?** [WBUT 20

a) n x (i - 1) + J b) m x 0 - 1) + i
c) m **x (n** - j) + j d) n x (m - i) + j

Answer: (b)

1"1. Which of tt,e following is non-linear data structure? [WBUT 201

a) Stacks b) List c) Strings d) Tf e$

Answer: (d)

Short Answer Type Quest ions

1. Let the size of the eterraents stored iri an 8 ,c 3 matrix be 4 bytes each. If the ba **address** of the matrix is 3500, then find the address of A (4, 2] for both row majc column major **cases.** (WBUT 2007, 20

What is sparse matrix? (WBUT 20

What are sparse matrices? How such a matrix is represented in memory? Wha t the **types** of **sparse** matripes? [WBUT 20

Answer:

1st **Part:**

In row major order the address will be

$$Loe(A(i,j)) = Lo + ((i-1) \cdot n + G-i)) \cdot c$$

Therefore

$$Loe(A(4,2)) = 3500 + ((4- .1) \cdot 3 + (2\ \ 1))\ A = l5QO + (1.0) \cdot 4$$
$$= 3500 + 40 = 3540$$

In column-major order the address will be

$$Loe(A(i,j)) = Lo + (G - i) \cdot m + (i - 1)) \cdot c$$

Therefore

$$Loe(A(4,2)) = 3S00 + ((2 -1) \cdot 8 + (4- 1)) \cdot 4 = 3S00 + (11) \cdot 4$$
$$= 3500 + 44 = 3544$$

STACKS & QUEUES

er Chapter at a Glance

- **Stacks:** A st ack is an abstract data type, which declares two methods PUSH and POP. Stacks are implemented either by an array or by a linked list.
- The PUSH oper tion allows us to insert data at the end of an array or linked list.
  The POP o rat1on allows us to re ove data from the end of an array or linked list.
  A **STACK** 1s called a last in first out (LIFO) system.
- A stack may be represented by a linked list. The first node of the list will be the topmost element of the stack and is pointed by a top pointer. This type of stack representation is called **linked stack.** The stack can be declared as follows:

```
typedef struct linked_lis
{
   int data;
   struct linked_list *next;
} lstack;
lstack *top;
```

- **Various types of expression:** A mathematical expression involves constants (operands) and operations (operators).
  Infix notation: operandi *operator* operand2, **A+ B**
  Prefix notation: *operator* operand1 operand2, + **A B**
  Postfix notation: operand1 operand2 *operator*, **AB+**
- **Conversion from INFIX to . POSTFIX . expression:** In order to convert the infix to its corresponding postfixes fonn; we need to do the following steps:
  - Fully parenthesize the expression according to the priority of different operators.
  - Move all operators so that they replace their corresponding ght parentheses.
  - Delete all parentheses.

  The priorities of different operatots ate given below:

Operators	Priori
Unary -, unary +, not (!)	4
*, /, %, and (& / &&}	3
+, - , or (I / II	2
<, <=, >, >=, = =, !=	1

- **Priority Queue:** A priority queue is essentially a list of items in which each item has associated with it **a *priority.*** In general, different items may have different priorities and we speak of one item having a higher priority than another. Given such a list we can determine which is the highest (or the low st) priority item in the list. Items are inserted into a priority queue in any, arbitrary order. However, items are withdrawn from a priority queue in order of their priorities starting with the highest priority item first. Two elements with the same priority are processed according to the order in which they were added to the queue.
- **De-queue:** De-queue is a linear data structure, where insertions and deletions are made to or from either end of the structure.

ueetlons J

[M ultiple Choice·- 1.r.

.1 A linear 11st In which elemente can be added or removed at either end bu the middle Is known •• [WBUT 2007, 2o8oinot1
•l **queue** bl **deque** c) **stack** **d) tre** '2G1a,

Answer: (b) 1

.2 **The prefix expression** for the Infix **expressiom** • **(b+** c)/ **e-** f **is** lWBUT 2
a) ra+ be f bl -/*+ **abcef** c) **--l*a+ bc-1** **d) None** of th O

Answer: (a) **114**

3: The number of stacks required to implement mutual recursion is [WBUT2 0
•l3 b)2 c) 1 d) none of the

Answer: (c)

4. Priority **queue** can be implemented using [WBUT 20
a) **array** **b)** linked list c) heap d) all of the

Answer: (d)

5. **Reserve** Polish notation is often known as [WBUT 20
a) Infix -) Prefix c) Postfix d) none of th

Answer: (c)

6. The **evaluation** of the postfix eJpre sion [WBUT 2
3, 5, 7, *, +, 12, % is
a) 2 b) 3 c) O d) 3.17

Answer: (a)

7. The operation for adding an entry t a stack is traditional,y called [WBUT 2
a) **Add** **b)** Append c) lo ert d) Push,

Answer: (d)

8. The **best data** structure to evaluate an arithmetic expression in os tfix fo rm
[WBUT
a) Queue b) .Stack c) Tree d) Lin ked

Answer: (b)

9. Theh **ea (represented** by an array) constructed from the list of numbers 80, 60, 15, 55, 17 is - [WBUT
a)60,80,55,30,10,17,15 b)SO 55 60 15 10 30 17
c) 80, 60, 30, 17, 55, 15, 10 d) n ne f **thes** • •

Answer: (b)

10. The postfix :qulvalent of the prefix * + *ab* -. *cd* is [WBUT 016,
a) ab+ c-d b) *abed* + - • c) *ab+ cd•* _ d) *ab+ _cd* 20141 20151 2

Answer: (a)

11. The following sequence of operations la performed ona stack: push (1), puah (2), pop, push (1), push (2), pop, pop, pop, push(2) pop. The sequence of popped out values are [WBUT 2014, 2019)
a)2,2, 1, 1,2 b)2,2, 1, 2,2 c)2,1,2,2,1 d)2,1,2,2,2
Answer: (a)

12. Stack cannot be used to [WBUT 2015)
a) evaluate an arithmetic expression in postfix form
b) implement recursion
c) allocate resources (like CPU) by the operating system
d) convert infix expression to its equivalent postfix expression
Answer: (c)

13. The f-0llowing sequence of operations is performed ona stack [WBUT 2015]
push(1), push(2), pop(), push(1), push(2), pop(), pop(), pop(), push(2), pop(), the sequence of popped out values are
a) 2, 2, 1, 2,1 b) 2, 2, 1, 1, 2 c) 2, 1, 2, 2, 2 d) 2, 1, 2, 2, 1
Answer: (b)

14. The deque can be used [WBUT 2016)
a) as a stack
b) as a queue
c) both as a stack and as a queue
d) none of these
Answer: (b)

15. Inserting an item into the stack when stack is not full is called ..... operation and deletion of item from the stack, when stack is not empty is called ..... operation. [WBUT 2017]
a) push, pop
b) pop, push
c) insert, delete
d) delete, insert
Answer: (a)

16. To make a queue empty, elemerits can be deleted till [WBUT 2018)
a) front= rear+ 1
b) front= rear -1
c) front = rear
d) None of these
Answer: (a)

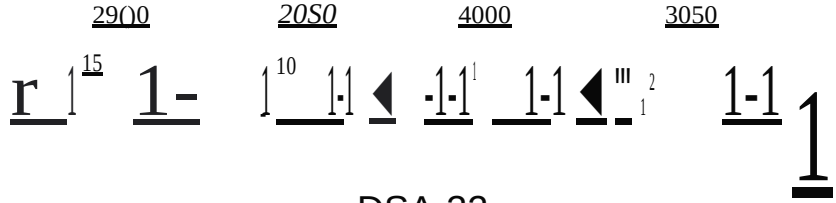
17. Which among the following are applicatiqns of queues? [WBUT 2019]
a) QueuQs **keep** track of events waiting to be handled, like multiple button clicks
b) Queues are uted for evaluation of arithmetic expressions
c) Queues are used in parsing
d') None of these
Answer: (a)

LINKED LIST

Cir'" Chapter at a Glance

- Sin gly linked lists: A li nked list is a data structure where d ta can be represented as a chain of nodes. Each node of a li nked list contains two parts: one 1 th address part an ther is tht data part. The address part holds the address of the node which is next to or previous to th, c urrent node.
  Typically the first node of a linked list is called the HEAD nod. e If the he d node is destroyed then th e entire list gets destroyed as weH. Depending on the traversal requirement_a linked list can be designed as circular, bi-directional etc. In it's simplest orm t e foll wm diaararn shows the structure of a uni-directional linked list also known as smgly lmked hst.
- Several operations: (Insertion, deletion, Traversing, Searching)
- **Insertion:** The algorithm for inserting in the above manner is **give** below:
  . Let us assume that x is data value to be inserted after the node containing the data value y. p initially contains the address of the head node.
  Step 1: start traversing the linked list from the head node
  Step 2: if the data value of the head node is not equal toy goto **step 8**
  Step 3: create a new node
  Step 4: place x to the data field of the new node
  Step 5: place the next address of the head node to th e next address of the new node
  Step 6: change the next address of the head node by the new node address
  Step 7: Goto Step 13
  Step. 8: Identify the address f the node containing data value y as followS-
  Step 8a) if the data value of the next node of pis equal toy
  Step 8b) else change p by its next node address
  Step 9: create a new node
  Step 10: place y to the data field of the new node
  Step 11: place the next address of p node to the next address of new node
  Step 12: change the next address of the p node by the new node address
  Step 13: End
- Circular linked list: the circular linked list, the address of the last node contains the addr$ of the first node, as shown in the figure below:

- Doubly linked list: The node of a doubly linked list contains two link fields, instead of o-n One link is used to point to the predecessor node, i.e., the previous node& the other link 15 used to point to the successor node, i.e., the next node. So, the two links are directed, one towards the front and another towards the back.



inked representation of poly nomial: In encra t any polyyom ia l A(x) can bewr it ten as

- L th A(x) = Ro + a, x + R2X + ----- + Rnl " n •r+ AnX."

Each a x ' is i te rn, off he p o t r nomial, where x is variable, a, is itscoe ffic en & e rs the exponent. If a = 0 then t he te nn 1s zero term, otherwise ti is nonzero term.

Multiple Choice Type Questions

1. In a circularly linked list organization, Insertion of a record involves the rnodification of [WBUT 2008, 2018]
a) no pointer b) 1 pointer c) 2 pointers d) 3 pointers
Answer: (c)

2. Linked list are not suitable for [WBUT 2012, 2016]
a) Stack b) Deque c) AVL Tree d) Binary search
Answer: (d)

3. Dynamic memory allocation use [WBUT 2012]
a) Calloc b) Malloc c) Free d) all of these
Answer: (d)

4. Which type of linked List contains a pointer to the next as well as previous node in the sequence? [WBUT 2013]
a) Singly Linked List b) Circular linked list
c) Doubly Linked List d) All of these
Answer: (c)

5. linked list is not suitable data structure for which one of the following problems? [WBUT 2014]
a) insertion sort b) radix sort
c) binary search d) polynomial addition
Answer: (c)

6. **Self-referential** pointer is used in defining [WBUT 2015]
a) an array **b)** a node of linked-list
c) **a queue** d) all of the •
Answer: (d)

7. Inserting a node after a given node in a doubly linked list requires [WBUT 2016, 2018]
a) four pointer exchanges b) two pointer exchanges
c) one pointer exchange d) no pointer exchange
Answer: (b)

B. Binary search is not possible for [WBUT 2017]
a) array b) linked 11st c) stack d) queue
nswer: (b)

9. A linear link list can be traversed using **[WBUT 2017]**

a) recursion **b) both (a) and (c) are correct**
c) stack **d) both (a) and (c) are wrong**

Answer: (a)

10. The data structure used to solve recur$ive problem is **[WBUT 2017]**

a) Linked list **b) Queue** **c) Stack** **d) none of these**

Ans,ver: (c)

11. linked list is a **[WBUT 2019]**

a) Linear data structure **b) Dynamic data structure**
c) Self referential data structure **d) all of these**

Answer: (d)

12. One limitation of linked list is **[WBUT 2019]**

a) it re·quir es huge memory space
b) it requires contiguous memory space
c) nodes can only be accessed sequentially
d) nodes can only be accessed randomly

Answer: (c)

Short Answer Type Questions

1. Write an algorithm 'to add two polynomials. **[WBUT 2007]**

Answer:

Let us assume that the two polynomials are represented using linked list and the resultant is also using a linked list.

Let phead I, phead2 and phead3 represent the pointers of the three lists under consideration.

Let each node contain .two integers: exp and coff.

Let us assume that the two linked lists already contain relevant data about the two polynomials.

Also assume that we have got a function append to insert a new node at the end of the gi ven list.

pl = pheadl;
p2 = phead2;

Let us call malloc to create a new node p3 to build the third list

p3 = phead3;

/* now traverse the lists till one 1ist gets exhausted */

while ((pl!=; NULL) || (p2 != NULL))

{

* if the exponent ofp 1 is higher than that of p2 then the next term in final list is gomg to be the node of p1* /

```
while (pl ->exp      p2  ->exp)
   {
```

TREES

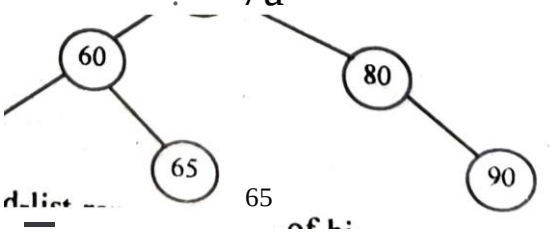
Chapter at a Glance

- **Trff:** A tree is a finite set of one or more nodes connected by edges such that
  1) There is a specially designated node called the root
  ii) The remaining nodes are partitioned into mutually exclusive disjoint sets
- Basic Terminology:
  i) ***Node:*** Each data item in a tree is called a node. It specifies the data information & links (branches) to other data items. In the example of the tree (1) as 13 nodes.
  ii) ***Root:*** It is the parent of all other nodes of a tree. A root node does not have any parent
  iii) ***Dqne of a node:*** The number of sub trees of a node is called its degree. In our example the root node has three sub trees (B, C, D). Hence, the degree of the root node A is 3. Degree of B is 2. Degree of C is 1. And so on,
  iv) ***Degree of a tree:*** It is the maximum degree of nodes in a given tree. In our example the **root node A has** the highest degree. Hence, the degree of the tree is 3.
  v) ***Terminal node(s):*** **A** node with degree zero is called a terminal node(s) or leaf node(s) or **external node(s).** In our example E, F, G, I are leaf nodes.
  vi) ***Non Terminal node(s):*** Any node whose degree is not zero is called non-terminal node(s) or internal nodes or interior node(s). In our example A, B, C, D & H are the non-terminal node(s).
  vii) ***Path:*** It is a sequence of consecutive edges from the source node to the destination node. In our example the path between A & I is given by the edges (A, D) (D, H) & (H, I).
- **Binary tree:** A binary tree is either empty or consists of one single vertex called the root together with two disjoint binary trees called left sub-tree & right sub tree. Number of branches of any node is either zero or one or at most two.
- Binary search tree: A binary search tree (BST) is a binary tree that is either empty or every data node that forms the tree satisfies the following conditions.

  The data in the left child of a node is less than the data in its parent node.

  ii) **The data in the right child of a node is greater than the data in its parent node.**

  The left and right sub trees of the root are also binary search trees. The fig. shown below is **Binary search tree.**



65

- **Threaded Binary** tree: In linked-list representation of binary trees, additional storage is required to store the two links of each node. The null pointers just results in wastage memory. To overcome this drawback, the null pointers are replaced by appropriate values called *threads.*

Mul ti nle Choice ueatl ona

1. If binary tree is threaded for In-order traveraal a right NULL llnk of any node I• ed by the **address** of Its [WBUT 2007, 2016)

pl:successor b) predecessor C) root d) own

4 ns -er: (3)

In 8 height balanced tree the heights of two sub-trees of every node never differ 1110 re than [WBUT 2007, 2009, 2018]

bY a) 2 b) 0 c) 1 d) - 1

A nswet : (c)

3 Maximum possible height of an AVL **Tree** with 7 nodes is [WBUT 2008, 2013, 2016, 2018]

a) 3 b) 4 c) 5 **d) 6**

Answer: (a)

4. A B-tree is [WBUT 2008, 2018]

a) Al ays balanced b) An ordered tree

c) A direct tree d) All of these

Answer: (d)

5. In array representation of Binary tree, if the **index** number of a child node **is 6** then the index number of its present node is [WBUT 2010, 2014]

a} 2 b) 3 c) 4 d) 5

Answer: (d)

6. The depth of a complete binary tree with n nodes [WBUT 2012]

a) log(n.+1)-1 b) log (n) c) log(n-1)+1 d) log(n)+1

Answer: (a)

7. In a binary search tree , if the number of nodes of **a tree** is 9, then the minimum height of the tr ee is [WBUT 2012]

a) 9 b) 5 c) 4 d) None of these

Answer: (d)

8 Which method of **traversal does** not **stack** to hold nodes that **are waiting** to **be processed:** [WBUT 2012]

a) Breadth - first **b)** Depth - first

c) 0- **search** d) None of these

Answer: (a)

:· Which of the following **la essential** for converting an Infix expression to the oatflx **expreaalon** efficiently? [WBUT 2013]

a) An operator stack b) An operand stack

A c) An operand stack and operator stack d) A **parse** tree

nswer: (a)

10. Number of all possible binary trees with 4 nodes is- [WBUT 2014]
a) 13 b)12 c)14 d) 15

Answer: (c)

11. If the inorder and preorder traversal of a binary tree are D, B, F, E, G, H and A, B, D, E, F, G, H, C respectively then, the postorder traversal of that tree is
a) D, F, G, A, B, C, H,E b) F, H, D, G, E, B, C,A [WBUT 2014]
c) C,G, H, F, E, D, B,A d) D, F, H, G, E, B, C,A

Answer: (d)

12. The number of possible distinct binary trees with 12 nodes is [WBUT 2015]
a) 4082 b) 4084 c) 3082 d) 3084

Answer: (b)

13. A binary tree has *n* leaf nodes. The number of nodes of **degree** 2 in this tree is [WBUT 2015]
a) logn b) *n* -1 c) *n* d) cannot be said

Answer: (d)

14. The minimum height of a binary tree of n nodes is [WBUT 2016]
a) $\lfloor n \rfloor$ b) $\lceil n \rceil$ c) $2^{n}-2$ d) $\log_2(n+1)$

Answer: (d)

15. The number of edges in a full binary tree of height *h* is [WBUT 2017]
a) $2^{h+1}-1$ b) $2^{h}-1$ c) $2^{h+1}-2$ d) $2^{h}-2$

Answer: (c)

16. Minimum number of nodes required to make a complete binary tree of height *h* is [WBUT 2017]
a) $2^{h}-1$ b) $2h$ c) $2h+1$ d) 2^{h-1}

Answer: (b)

17. Which one is required to reconstruct a binary tree? [WBUT 2017]
a) Only inorder sequence
b) Both preorder and postorder sequences
c) Both inorder and postorder sequences
d) Only postorder sequence

Answer: (c)

18. Number of nodes in a complete binary tree of depth *k* is [WBUT 2018]
a) $2k$ b) 2^{k} c) $2^{k}-1$ d) None of these

Answer: (c)

19. A full binary tree with n non-leaf nodes contains [WBUT 2019]
a) log 2n nodes b) n+1 nodes c) 2n nodes d) 2n+1 nodes

Answer: (d)

CiRAPHS

Chapter at a Glance

- A graph is a mathematical tool used to represent a physical problem. It is also used to model networks, data structures, scheduling, computation and a variety of other systems where the relationship between the objects in the system plays a dominant role.
- **Types of graph:** A graph often symbolized as G can be of two types:
  1. **Undirected graph:** where a pair of vertices representing an edge is unordered.
  2. **Directed graph:** where a pair of vertices representing an edge is ordered.
- **Graph Traversal: A** graph can be traversed in two ways:
  Depth first search traversal: often analogous to pre-order traversal of an ordered tree.
  Breadth first search traversal: often analogous to level-by-level traversal of an ordered tree.
- **Spanning Tree:** Given a connected, undirected graph, a spanning tree of that graph is a sub graph which is a tree and connects all the vertices together. A single graph can have many different spanning trees. We can also assign a *weight* to each edge, which is a number representing how unfavourable it is, and use this to assign a weight to a spanning tree by computing the sum of the weights of the edges in that spanning tree.
- **Shortest Path:** In grap theory, the shortest path problem is the problem of finding a path between two vertices in a weighted graph such that the sum of the weights of its constituent edges is minimized.
  The most important algorithms for solving this problem are:
  Dijkstra's algorithm solves the single-source shortest path problems.
  Bellman-Ford algorithm solves the single source problem if edge weights may be negative.
  A* search algorithm solves for single pair shortest path using heuristics to try to speed up the search.

Multiple Choice Type Questions

1. The vertex, removal of which makes a graph disconnected is called [WBUT 2007]
a) Pendant vertex **b) bridge**
c) articulation point **d) colored vertex**
Answer: (c)

2. A vertex of in-degree zero in a directed graph is called [WBUT 2007, 2018)
a) articulation point b) sink
c) **isolated vertex** **d)** root **vertex**
Answer: (c)

3. Adjacency matrix of a digraph is
a) identity b) symmetric[WBUT 2007, 2012, 2016)
c) asymmetric **d)** none of **these**
Answer: (b)

.4. Which **data** structure is used for breadth first traversal ofa graph? [WBUT 2008]
a) Stack b) Queue
c) Both stack and queue d) None of these
Answer: (b)

5. The adjacency matrix of an undirected graph is [WBUT 2008, 2010]
a) UnU matrix b) Asymmetric matrix
c) Symmetric matrix d) None of these
Answer: (c)

&. B FS [WBUT 2008]
a) scans all incident edges before moving to the other vertex
b) scans adjacent unvisited vertex as soon as possible
c) is same as backtracking
d) none of these
Answer: (b)

7. A non-planar graph with minimum number of vertices has [WBUT **2008,** 2018]
a) 9 edges, 6 vertices b) 6 edges, 4 vertices
c) 10 edges, 5 vertices d) 9 edges, 5 vertices
Answer: (c)

8. Any connected gfaph with x vertices must have at least [WBUT 2009]
a) x+ 1 edges b) x-1.edges c) x edges d) x/2 edges
Answer: (b)

9. Maximum number of edges in an-node undir.ected grapti without self loop is [WBUT 2010]
a) n^2 b) $\frac{n(n-1)}{2}$ c) $n...-2$ d) $\frac{(n+1)(n)}{2}$
Answer: *(b)*

[WBUT 2010, 2014, 2018]
10. BFS constructs
a) a minimal cost spanning **tre e** of a graph
b) a depth first spanning tree of a graph
c) a breath first spanning tree ofa graph
d) none of these
Answer: (a)

[WBUT 2011]
11.A complete directed graph of 5 nodes has..... d) 25
a)5 b) 10 c) 20
Answer: (b)

[WBUT 2012]
12.**A vertex** with **degree** one in a graph is called d) none of these
a) leaf b) pendant vertex c) end vertex
Answer: (b)

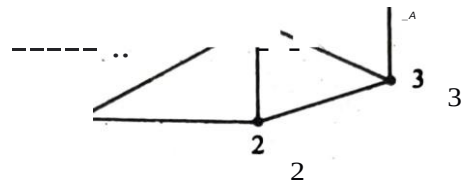
13. To implement DFS which **data** atructure **Isgenerally used?** [WBUT 20131
a) Stack b) Queue c) Both (a)& **(b)** d) None of tt111,

Answer: (a)

14. Adjacency matrix for a digraph is - [WBUT 2014]
a) unit matrix **b)** symmetric matrix
c) asymmetric matrix **d)** none of **these**

Answer: (b)

15. What is the sum of the **degrees** of all the vertices In the following **graph?** (WBUT 2017]



a) 19
b) 20
c) 5
d) none of these

Answer: (a)

16. The adjacency matrix of an undirected graph is [WBUT 2017]
a) Unit matrix b) Asymmetric matrix
c) Symmetric matrix d) none of these

Answer: (c)

17. A path is **[WBUT** 2017}
a) **a** closed walk with no vertex repetition
b) an open walk with no vertex repetition
c) an open walk with no edge repetition
d) a closed walk with no **edge** repetition

Answer: (c)

Short Answer Type Questions

1. Describe Kruskal's minimal spanning tree algorithm. [WBUT 2008, 201

Answer:

Kruskal's algorithm is an algorithm in graph theory that finds a minimum spanning t for a connected weighted graph. This means it finds a subset of the edges that form tree that includes every vertex, where the_ total weight of all the edges in the tre minimized. lf the graph is not connected, then it finds a .minimum spanning for s minim um spanning tree for each connected component). Kruskal's algorithm is example of a greedy algorithm.

It works as follows:

- create a forest *F* (a set of trees), where each vertex in the graph **is a** separate tree
- c rea te a setS containing all the edges in the graph
- while Sis nonempty
- remove an edge with minimum weight from S

SORTING & HASHING

Chapter at a Glance

- A **hash table** data structure is Just ... rke an array. Data is stored into this array at specific ind ... hashes (converts) a num er m a large ran ... generated by a hash function. A **hash function** ... into a number in a smaller range. ... ge,
- **Linear search:** Lmear or equ ... sential Search is a me ... thod where the search begins at one end ... the list, scans the elements of the last frome ... ft to right (if the search begms from left) until ... of th ... desired record is found. ... d ed li'st or an unordered list. For order ... 1 1l11's type o searc m g can e bperfor med ma n or er ked lists or files with variable-Jen h lists that must be -accessed sequentially, sue as m d b th gt records lacking an index the average performance can be improve Y givm pat e first element which is greate; than the unmatched target value, a er an exammmg the entire
- **:ary Search:** In Binary Search the entire sorte list is divided i' to two pa . We first compare our input item with the mid element of the list and t_hen r stnct our attention o only the first or second half of the list depending on whether the mput item comes left or nght of the mid-element. In this way we reduce the length of the list to be searched by h lf.
- **Hashing:** In hash' tables, there is always a possibility that two data elements will hash to the same integer value. When this happens, a collision occurs i.e. two data members try to occupy the same place in the hash table array. There are methods to deal with such situations like **Open Addressing and Chaining:**
- **Bubble sort:** Given an array of unsorted elements, s ubbl e sort performs a sorting operation on the first two adjacent elements in the array, then between the second & third then between third & fourth & so on.
- **Insertion sort:** In insertion sort data is sorted data set by identifying an element that is out of order relative to the elements around it. It removes that element from the list, shifting all other elements up one place.

  Finally it places the removed element in its correct location.

  For example, when holding a hand of cards, players wi'II often scan th e1 r car d s fr om J eft to gh t, ookmg for the first card that 1s out of place If the first th d f ree car s o a p layer's h an dare 4, S, 2, h e wi ll often be satisfied that the 4 and the 5 are i d 1 . th . but, upon getting to the 2 desires to 1 't b c. n or er re at1ve to eacho er, typically removes the 2 from the list, shift e, ore the 4 and t e .5 In that case, t e Pr8yer s I s t e 4 and the 5 . h d then places the 2 into the first slot on the left. one spot to the ng t, an
- **Qu'ick sort:** In Quick-Sort we divide the arr . (normally the middle element of the array) :y mto two halves. We select a pivot element elements to the left of the pivot element anl performa . sorting in such a manner that all e greater than the pivot element. Thus we get two su an It all the elements to it's ng 1s esser th . & . ht 1s
- **Merge sort:** In this method we d1'v1'de th b arrays. ?earIY equal as possible and then sort them **e array** r o I1'st . into two sub arrays or sub h's a5 mto another sub arrays. separately. Then the sub-arrays are again divided
- **Heap** sort: A heap takes the fonn of b' t . mm1mum e ement is placed in the root a De ana ry, ree w ith the feature that the maxirnurtt or pen mg up on h t is feature the heap is calle d rn8%'

heap or min-heap respectively. A er the heap construction the elements from the root ar taken out from th e tree and the heap structure is reconstructed. This process continues until the heap is empty.

Multiple Choice Type Questions

1 The ratio of the number of **Items** In a hash **table,** to the table **size is** called the [WBUT 2007, 2009, **2016]**

a) load factor **b) item** factor c) balanced factor d) all of these

Answer: (a)

2. Which of the f l owing is not a requirement of good hashing function?

a) Avoid colhs1on b) Reduce the storage space

c) Make faster retrieval d) None of these (WBUT 2008, 2015]

Answer: (b)

3. Stability of Sorting Algorithm is important for [WBUT 2007]

a) Sorting records on the basis of multiple k eys

b) Worst case performance Qf sorting a!gorithm

c) Sorting alpha numeric keys as they are l kely to be the same

d) None of these

Answer: (a)

4. Which of the following is the best time for an algorithm? [WBUT 2007]

a) O(n) b) O(log2 n) c) O(2n) d) O(n log2 n)

Answer: (b)

5. The Linear Probing Technique for collision resolution can lead to [WBUT 2009]

a) Primary lusteri n.g b) Secondary clustering

c) Overflow d) Efficiency storage utilization

Answer: (a)

6. The fastest sorting algorithm for an almost already sorted array is [WBUT 2009]

a) q1,.1i k sort b) merge sort c) se ec ion sor.t d) insertion sort

Answer: (d)

7- The time complexity qf binary search is [WBUT 2009]

a)O(n 2 b.) O(n) .c) O(log n) d) O(n log n)

Answer: (c)

8 The best case time complexity of Bubble sort technique is [WBUT 2010]

a) 0(n) b) $O(n^2)$ c)' O(n logn) d) O(logn)

Answer: (a)

-9 Which of the following sorting procedures is the slowest? CVVBu,- 0,n,
a) Quick sort b) Heap sort c) **Merge** sort d) Bubble i l

Answer: (d)

10. Which of the following traversal techniques lists the elements of search tree in ascending order? [WBUT20 1 b2lna
a) pre-order b) Post-order c) lnorder d) None of' 01

Answer: (c) th"-

11. Binary search cannot be used in linked lists. CWBUT 20111
a) True b) False

Answer: (b)

12. Breadth-first-search algorithm uses.....**data** structure [WBUT 20111
a) stack b) queue c) binary tree d) none of these

Answer: (b)

13. The best case complexity ofinsertion sort is - [WBUT 2011]
a) o{n^2)) O(n $\log_2$ n) c-) (n^3) d) *O(n)*

Answer: (d)

14. Which of the following is not related to hashing? [WBUT 2011]
a) Synonyms b)'Collision c) Balance d) Load factor

Answer: (c)

15. A machine needs a minimum of 100sec to sort 100 **names by** quick sort. The minimum time needed to sort 100 nam es will be approximately [W UT 201
a) 72.7 sec b) 11.2 sec c} 50.2 sec d) 6.7 sec

Answer: (d)

16. What **will be** the time complexity for selection sort to sort an array ofn elements? [WBUT 2012, 201ij
a) O(log n) b) O(n log n) c) O(n) cl) Q(n^2)

Answer: (d)

17. The best sorting technique when the **data** is almost sorted is [WBUf 2013)
a) Selection sort b) Bubble sort c) Quick sort d) Insertion so

Answer: (d)

18. Which of the following is a hash function? [WBUT 2014)
a) Quadratic probing b) chaining
c) open addressing d) folding

Answer: (a)

19 The number of swapping needed to sort numbers 8, 22, 7, 9, 31, 19, 5, 13 in 1 endin g order using bubble sort Is [WBUT 2014)
as a)1-1 b)12 c)13 d)14
Answer: (d)

20. Binary search uses [WBUT 2014)
a) divide and reduce strategy b) divide and conquer strategy
c) heuristic search d) both (a) and (b)
Answer: (b)

21. Merge sort uses [WBUT 2015)
a) divide and conquer strategy b) backtracking approach
c) heuristic search d) greedy approach
Answer: (a)

22. The prerequisite condition of Binary search is [WBUT 2017,2019)
a) unsorted array b) ascending order array
c) descending order array d) sorted array
Answer: (d)

23. The worst case complexity of binary search for a list having n elements is [WBUT 2019)
a) log 2" b) n log 2 c) n d) n A2
Answer: (b)

24. Consider that n elements are to be stored. What is the worst case time complexity of Bubble sort? [WBUT 2019]
a) 0(1) b) O(log 2n) c) 0(n) d) 0(nA2)
Answer: (d)

25. What is the worst case performance of Selection sort algorithm? [WBUT 2019]
a) 0(log n) b) ()(n*n) c) 0(n) d) 0(n log n)
Answer: (b)

26. A sort which relatively passes through a list to exchange the first element with any element less than it and then repeats with a new first element is called [WBUT 2019]
a) insertion sort b) selection sort c) heap sort d) quick sort
Answer: (a)

Short Answer Typo g ue sd ons

1. Explain the advantages of binary search over sequential search. [WBUT 2007, 2013)

MISCELLANEOUS

Multiple Choice Type Questions

1. Recursion uses more memory **space** than iteration **because** **[WBUT 2019]**
   a) it uses stack instead of queue
   b) every recursive call has to be stored
   c) both (a) and (b)
   d) none of these

Answer: (c)

Short **Answer Type Questions**

1. a) Describe a string reversal algorithm. **[WBUT 2012, 2015]**
b} What is difference between Union & Structure? **[WBUT 2012]**

Answer:

a) The C code isgiven below:

```
void reverse(char *str) {
    char* end= str;
    char tmp;

    if (str) {
        while (*end) {
            ++end;
        }
        --end;
        while (str < end) {
            tmp = *str;
            *str++ = *end';
            *end--= tmp;
        }
    }
```

Inthe above code, in the innermost while loop in each iteration, the characters pointed to by str and end get swapped, str gets incremented to point to the next character, and end is decremented to point to the previous one.

) Th differences between structure and union:

Union allocates the memory equal to the maximum memory required by the member of the union but structure allocates the memory equal to the total memory required by the members.

In union, one block is used by all the member of the union but in case of structure, ach member has their own memory space.

lJESTION 2015

GROtlp _ A

(Multiple Cho ic r Type Questions)

_1 Answer any ten qu s tions:

i) Which of the following traversal techniques hsts the nodes of a binary search tree in ascending order?

a) Post-order b) In-order c) Pre-order d) None of these

ii) The number of possible distinct binary trees with 12 nodes is

a) 4082 b) 4084 c) 3082 d) 3084

iii) Which of the following expressions access the (i, j)th entry of a (rn x n) matrix stored in column major order?

a) n x (i - 1) + j b) m x (j - 1) + I c) m x (n - j) + j d) n x (m - i) + j

iv) Stack cannot be used to

a) evaluate an arithmetic expression in postfix form

b) implement recursion

c) allocate resources (like CPU) by the operating system

d) convert infix expression to its equivalent postfix expression

v) The postfix equivalent of the prefix• + ab - cd is

a) ab+ cd - • b) abed+ - • c) ab+ cd* - d) ab+ -e ed..,

vi} Merge sort uses

a} divide and conquer strategy b} backtracking approach

c} heuristic search d)•greedyapproach

vii} The following sequence of.operations is performed on **a stack**

push(1), push(2), pop(), push(1), push(2), pop(), pop(), pop(), ps h(2) pop(),

the sequence of popped out values are

a) 2, 2, 1, 2, 1 b) 2, 2, 1, 1, 2 c) 2, 1, 2, 2,2 d) 2, 1, 2, 2, 1

viii) Which of the following is not a requirement of good hashing function?

a) Avoid collision b) Reduce the storage space

c) Make faster retrieval d) None of these

ix} Self-referential pointer is used in defining

a) an array b) a node of linked-list

c) a queue d) all of these

x) A binary tree with *n* leaf nodes The number of nodes of degree 2 in the tree is

a) logn b) *n* - 1 c) *n* d) cannot be said

GROUP-B

(Short Answer Type Questions)

2 What is the difference between array and linked-list? What is the primary criterion of perforrning binary search technique on a li st of data?

1st **Part: Sec Topic: LINKED LIST, Short Answer Type Question No. 11.**

l"d **Part: See Topic: SORTING & HASHING, Short Answer Type Question No. 9.**

3. Write a recursive algorithm to solve tower 9f Hanoi problem.

See Topic: MISCELLANEOUS, Long Answer Type Question No. l(b).

4. Deduce the average time complexity of Quicksort algorithm.

Sec Topic: SEARCHING & SORTING, Short Answer Type Question No. 8.

5. Suppose a queue is implemented by an array. Write an algorithm to insert a new element at the kth position oUhe array.

See Topic: STACKS & QUEUES, Short Answer Type Question No. 8.

6. Write an algorithm to delete the last node of a linked-list.

See _Topic: LINKED LIST, Sho_rt Answer Type Question No.

GROUP-C

(Long Answer Type Questions)

7. a) The in-order and pre-order traversal sequence of nodes in a binary tree are given below:

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

Construct the tree.

b) Define Hashing.

c) Describe a String reversal Algorithm.

d) Write an algorithm for inserting an element into a Binary tree with example.

a) & d) See Topic: TREES, Long Answer Type Question No. **1,(a)** & (b).

b) Sec Topic: **SORTiNG & HASHING, Short Answer Type Question No.** 2

c) See Topic: **MISCELLANEOUS, Short Answer Type Question No. l(a).**

a. a) Convert the following infix expression into equivalent postfix expression using stack.

(A + B) * C - (D - E)/(F + G)

b) What is dequeue?

c) How can a polynomial such as $6x^6 + 3x^3 - 2x + 10$ be represented bya linked list?

d) For the following expression draw the corresponding expressiontree:

$a+b*c-d/e$

e) Write an algorithm to insert an element in the middle of a linked list.

a) See Topic: STACKS & QUEUES, Short Answer Type Question No. l(a).

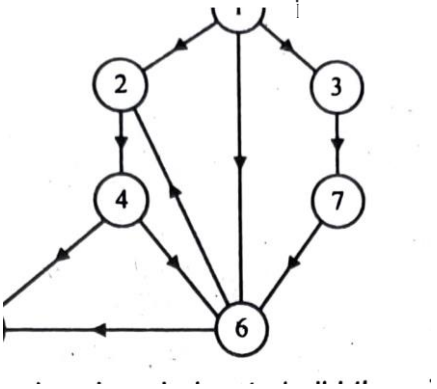
b) See Topic: STACKS & QUEUES, Short Answer Type Question No. 3.

c) **Sec Topic: LINKED LIST, Short Answer Type Question No. J.**

d) **See Topic: TREES, Short Answer Type Question No. 9.**

e) Sec Topic: LINKED LIST, Short Answer Type Question No. JO.

g For the following graph find the BFS and DFS traversal with proper algorithm.



b) insert the following **keys** in the order given below to build them into an AVL-tree, 12, 11, 13, 1O, 09, 15, 14, 18, 7, 6, 5, 4. Clearly mention different rotations used and balance factor *of* each node.

a) Se e Topic: GRAPHS, Short Ans er Type Question No. *5.*

b) See Topic: TREES, Short Answer Type Question No. 7.

10. a) What is hashing? Describe any three methods *of* defining . a hash function.

b) Discuss different collision resolution tectiniques .

a) See Topic: SORTING & HASHING, Long Answer Type Question No. l l.

b) See Topi_c: SORTING & HASHING, Long Answer Type Question No. 3(c).

11. Write short notes on any thr e of the following:

a) ADT

b) AVL Tree

c) Circular link list

d) Threaded binary trees

e) Heap

a) See Topic: INTRODUCTION, ong Answer ype Question No. I.

b) See Topic: TREES, Long Answ r Type Question No. 7(a).

c) See Topic: LINKED LIST, Long Answer Type Question No. 5(b)

d) See Topic: TREES, Long Answer Type Question No. l?(b).

e) See Topic: SORTING& HASHING, Long Answer Type Question No. 7(d).

QUESTION 2016

GROUP-A

(Multiple Choice Type Questions)

1 **Ans**

i) The wer any te questions:

postfix equivalent of the prefix*+ ab cd is

a) $ab+cd-*$ b) $abcd+-*$ c) $ab+cd*-$ d) $ab+-cd*$

ii) If a binary tree is threaded *tor* inorder traversal a right NULL link of any node is replaced by the address of its

a) successor b) predecessor c) root d) own

iii) Adjacency matrix of a digraph Is
a) Identity matrix b) Symmetric matrix c) Asymmetric matrix d) None of thsee

iv) Linked lists are not suitable for
a) Stack b) Dequeue c) AVL tree d) Binary Searcn

v) The ratio of items present in a has.h table of the total **size is** called
a) balance factor b) load factor c) item factor d) weight factor

vi) Maximum possible height of an AVL tree with 7 nodes is
a) 3 b) 4 c) 5 d) 6

vii) The deque can be used
a) as a stack b) as a queue
c) both as a stack and as a queue d) none of the e

viii) Inserting a node after a given node in a doubly linked iist reqL:Jires
a) four pointer exchanges b) *twp* pointer exchanges
c) one pointer exchange d) no pointer exchange

ix) The minimum height of a binary tree of n nodes is
a) n b) n/2 c) n/2- 2 d) $\log_2(n+1)$

x) What will be the time complexity for selections sort to sort an array of n elements?
a) O(Iogn) b) o(nlogn) c) O(n) d) $O(n^2)$

GROUP-B
(Short Answer Type Questions)

2. Show tt,at the function $f(n)$ defined by

$f(n) = 1; \quad n = 1$

$f(n) = f(n-1) + 1/n, \quad n > 1$

has complexity O(ln n).

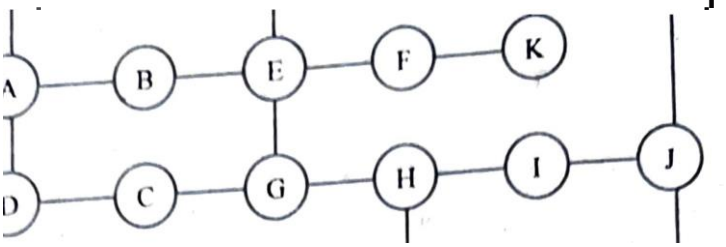
Sec Topic: INTRODUCTION, Short Answer Type Question No. 2.

3. a) Do·es a B tree grow at Its leave or at its r<;>ot? Why?
b) In deletinga key from a B tree, When it is necessary to combine nodes?
c) For what purposes are B trees especially appropriate?
a)& **b) See Topic: TREE, Short Answer Type Question No.** IO.
c) S e Topic: TREE, Long Answer Type Question No. 4(b).

4. The post--order and in-order traversal sequences of codes ina binary tree are given below.
Post-order: D G E B H I F C A
Pre-order: D B G E A C H F I
Construct the binary tree.

1.0 a) Describe BFS algorithm

b) Find out the DFS traversal of the following graph **starting at node A**



c) Define Prim's algorithm for minimum cost spanning tree with example

a) See Topic: GRAPHS, Long Answer Type Question No. **1(a).**

b) See Topic: GRAPHS, Short Answer Type Question No. 6.

c) See Topic: GRAPHS, Short Answer Type Question No. 4.

QUESTION 2017

GROUP–A
(Multiple Choice Type Questions)

1. Choose the correct alternatives for any ten of the following:

i) Which of the following is non-linear **data** structure?

a) Stacks b) List c) Strings d) **Trees**

ii) Binary search is not possible for

a) array b) linked list c) stack d) queue

iii) The prerequisite condition of Binary search is

a) unsorted array b) ascending order array

c) descending order array d) sorted array

iv) Inserting an item into the stack when stack is not full is called ..... operation and deletion of item from the stack, when stack is not empty is called ..... operation.

a) push, pop b) pop, push c) insert, delete d) **delete,** insert

v) The number of edges in a full binary tree of height h is

a) $2^{h+1} - 1$ b) $2^h - 1$ c) 2h+I 2 d) $2h - 2$

vi) Minimum number of nodes required to make a complete binary tree of height h is

a) $2h - 1$ b) $2h$ c) $2h+1$ d) 2h-I

vii) A linear link list can be traversed using

a) recursion

c) stack

b) both (a) and (c) are correct

d) both (a) and (c) are wrong

\lli0 •A'hst is the sumo f h e degrees of ell tho v rt, ces in the following gr nh?

8) 19 b) 20 c) d) none of the 9--,

n,e adjacency matrix of an undirected graph is
a) Unit matrix
b) Asymmetric matn
c) Symmetric matrix
d) none of these

A path is
a) a dosed **walk** with no vertex repetition
b) an open walk with no vertex repetition
c) an open walk with no edge repetition
d) a closed walk with no edge repetition

xi) The data structure used to solve recursive problem is
a) Linked list b) Queue c) Stack d) none of th ese

xii) Which one is required to reconstruct a binary tree?
a)On inordersequence
b) Both preorder and postorder sequences
c) Both inorder and postorder sequences
d) Only postorder sequence

Group-B
(Short Answer Type Questions)

2. How a polynomial such as $8x^5 + 4x^3 - 9x^2 + 2x - 17$ can be represented using a linked list? **What are** the advantages and disadvantages oflinked list over an array?
Sec Topic: LINKED LIST, Sliort Answer Type Question No. 13.

3. If $T(n) = a_0 + a_1x + a_2x^2 + a_3x^3 + ... + a_nx^n$ then prove $T(n) = e^{(n)}$.

See Topic: INTRODUCTION, Short Answer Type Question No. 14.

- Why circular queue is used over simple queue? Write an algorithm to insert an element in a Circular queue.

t• **part: See** Topic: STACKS & QUEUES, Short Answer Type Question No.5(b)
part: Su Topic: STACKS & QUEUES, Long Answer Type Question No.5(iii)

5. **The inorder and preorder** tree traversals are given. Dr w thebinary tree
Inorder: ABCOEFGHI
Preorder: FBAOCEGIH
Is it P<>&sible to build upa unique binary tree when its preorder and postorder traversal is given?

Stt Topic: TREES, Short Answer Type Que11tlon No. 18.

What is Hashing? Write two hash functions What is collision
tlopic:: SORTING& HASHING, Short Answer Type QucStionNo l.

QUESTION 2018

Group-A
(Multiple Choice Type Questions)

_ 1 Choose the co ect al_tematives fot the following:

i) Maximum possible height of an AVL Tree With 7 node is

a) 12 b) 4 c) 5 d) 3

ii) In a circularly !"inked list organization, insertion of a record involves the modification of

a) no pointer b) 1 pointer c) 2 pointers d) 3 pointers

iii) A 8-tree is

a) always balanced b) an ordered tree c) a directed tree d) All of these

iv) Number of nodes in a complete binary tree of depth *k* is

a) $21c$ b) $2k$ c) $21c$-1 d) None of these

v) To make a queue empty, elements can be deleted till

a) front= rear+ 1 b) front= rea - 1 c) front = rear d) None of these

vi) BFS constructs

a) a minimal cost spanning tree of a graph b) a depth first spanning tree of a graph

c) a breadth first spanning tree d) None of these

vii) A vertex of in-degree zero in a directed g(aph is called

a) Articulation point b) Sink

c) Isolated matri.x d) Root yertE:x

viii) In a h ight balanced tree the height f two sub-:-tr es of every node never differ by more than

a) 2 b) O c) 1 d) -1

ix) Inserting a new node after s cific node in a doubly linked requires

a) tout pointer exchanges b) two pointer exchanges

c) one pointer exchanges d) no pointer exchanges

x) A non-planar graph with minimum number of vertices has

a) 9 edges, 6 vertices b) 6 edges, 4 vertices

c) 10 edges,5 vertices d) 9 edges,5 vertices

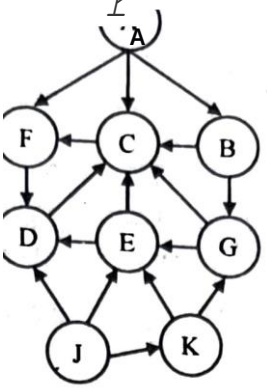
Group-8
(Short Answer Type Questions)

- Write an algorithm for in-order traversal of a threaded binary tree

te Top·ic. **T REES, Short Answer Type Questi·on No. 2.**

9. What is expressio tree? Draw the expression tree and write the In, Pre & Post-Order tra ersa s the given expression tree: E = (2x +y) (5a - b)^3. Prove that the number of odd degree vertices in tor raph is always even . Apply BFS/DFS Algorithms and find out the path of the given graph:

a



1^{11} & **2nd Part: See Topic: TREES, Short Answer Type Question No. 16.**
3rd **Part: See Topic: GRAPHS, Long Answer Type Question No. l(c).**
4• Part: See Topic: GRAPHS, Short Answer Type Question No. 7.

10. a) Define circular queue.
b) Write an algorithm to insert an item in circular queue.
c) What is input restricted dequeue?
d) Write an algorithm to convert an infix expression to postfix using stack.
a), b) & **c) See Topic: STACKS & QUEUES, Long Answer Type Question No. J(a), (b) & (c).**
d) See Topic: STACKS & QUEUES, Long Answer Type Question No. 2.

11. Write short notes on *any three* of the following:
i) AVL Tree
ii) Heap Sort
iii) DFS
iv) Tail recursion
v) Binary Search Tree

i) See Topic: TREES, Long Answer Type Question No. 17(a).
ii) See Topic: SORTING & HASHING, Long Answer Type Question No. 17(d).
iii) See Topic: G PHS, Long Answer Type Question No. 4(d).
iv) See Topic: MISCELLANEOUS, Long Answer Type Question No. l(b}.
v) See Topic: TREES, Long Answer Type Question No. 17(d).

QUESTION 2019

G roup - A

1 C **(Multiple Choice Type Questions)**

i) ·Thoose the correct alternatives or *any ten* of thefo llowing:

e worst case complexity of binary search for a list havingn elements 1s

a) log 2" b) *r*/log 2n c) *n* d) n'' 2

ii) Which among the following are applications of 1u eues?

a) Queues keep track of events waiting to ue handled, like multiple button clicks

b) Queues are used for evaluation of arithmetic expressions

c) Queues are used in parsing

d) None of these

iii) A full binary tree with n non-leaf nodes contains

a) log 2n nodes b) n+1 nodes c) 2n nodes d) 2n+1 nodes

iv) Consider that n elements are to be stored. What is the worst case time complexity of Bubble sort?

a) 0(1) b) 0(log 2n) c) 0(n) d) 0(n"2)

v) The pre-requisite condition of binary search is

a) Unsorted array b) Ascending order,array

c) Descending order array d) Sorted array .:

vi) The following sequence of operations are performed on a stack: push (1), push(2), pop, push (1) , push(2), pop, pop, pop, push(1) pop. The sequence of popped out values are

a) 2, 2, 1, 1, 1 b) 2, 2, 1, 2, 2 c) 2, 1, 2, 2, 1 d) 2, 1, 2, 2, 2

vii) The postfix equivalent of the prefix expressio,n -* +ab - cd is

a) ab + cd - * b) abed +- * c) ab + cd * - d) ab+ - cd *

viii) Linked list is a

a) Linear data -structure b) Dyn mic data structure

c) Self referential **data** structure d) all of these

ix) One limitation of linked list is

a) it requires huge memory space b) it require contiguous memory space

c) nodes can only be accessed sequentially d) nodes can only be accessed randomly

x) What is the worst case performance of Selection sort algorithm?-

a) 0(log n) b) 0(n*n) c) 0(n) d) 0(n log n)

xi) A sort which relatively passes through a list to exchange the first element with any element less than it and then rep.eats with a new first element is called

a) insertion sort b) selection sort c) heap sort d) quick sort

xii) Recursion uses more memory space than iteration because

a) it uses stack instead of queue b) every recursive call has to be stored

c) both (a) and (b) d) none of these

notation. notation and theta notation are used for this. Usually asymptotic estimates used because different itnpletnentations of the same algorithm may differ in efficiency. However the efficiencies of any two "reasonable" ilnplementations of a given algorithm are I clated by a constant multiplicative factor called a hidden conslanl.

complexity: There are two types of complexities of an algorithm, time complexity and space con

The time complexity of an algorithlll is the alnount of' time the computer requires to execute the algorithm.

Thespace complexity of an algorithlll iq the atnount of memory space the computer requires, cornpleting the execution of the algoritlun.

Multi le Choice e uestions

1. Each element of an array arr[20][50] requires 4 bytes of storage. Base address of arr is 2000. The location of arrC101[101 when the array is stored as column major is
a) 2820 b) 2840 c) 4048 d) 4840 CWBUT 2008).

Answer: (b)

2. In C language, malloc( ) returns [WBUT 20091
a) integer pointer b) structure pointer
c) null pointer d) void pointer

Answer: (c)

3. Which of the following is the best time for an algorithm? CWBUT 20101

b) $O(\log_2 n)$ c) $O(2^n)$ d) $O(n \log_2 n)$

Answer: (b)

of n? PUBUT 20101

b) o(n) c) $O(\log_2 n)$ d) $O(2^n)$

Answer: (d)

5. In C language, arrays are stored in which representation?

a) Column major b) Row majorc) Layer major [WBUT 20111
Answer: (b)
d) None of these

4. Four algo.s do the same task. Which algo. should execute the slowest for large values

6• Which of the following algorithm should execute the slowest for large values of N?

c) O(log2N) **b) O (N^2)** Answer: (b) **U T** 20121 d) None of these

7. Which is better computing time (in analysis of algorithm)?

b) o(2n) c) O(log2n) [WBUT 20131
Answer: (c)
d) None of these

DSA-3

8. A dynamic data structure where we can search for desired records [WBUT in 2013) time is b) binary search tree
a) heap d) array
c) circularly linked list
Answer: (b)

9. For Column Major, what is the address of [3, 2] th element of a
(contains integer number)? It is given that the c) 'Base 2016Address is 2000'. [WBUT
a) 2010 b) 2012

Answer: should be 2020

10. Which of the following expressions access the (i, j)th entry of a (m [WBUT x n) matrix20151 stored in column major order? b) m X O—I) + i

a) n × (i – 1) + j
d) n × (m – i) + j
c) m × (n – j) + j

Answer: (b)

(WBUT 2017)

11. Which of the following is non-linear data structure? d) Trees b) List c) Strings
a) Stacks

Answer: (d)

Short Answer e uestions

1. Let the size of the elements stored in an 8 x 3 matrix be 4 bytes each. If the base address of the matrix is 3500, then find the address of A (4, 2] for both row major & column major cases. [WBUT 2007, 2018)

What is sparse matrix? [WBUT 20071

OR,

What are sparse matrices? How such a matrix is represented in memory? What are the types of sparse matrices? [WBUT 2018] Answer: r t Part:

In row major order the address will be

Loc (A(i,j)) = Lo + ((i-l) * n + (j-i)) * c

Therefore

Loc = 3500 + l) * 3 + (2 - l)) * 3500 + (1 0) * 4
= 3500 + 40 = 3540

In column-major order the address will be

Loc (A(i,j)) = Lo + ((j — i) *m + (i − l)) * c

Therefore

Loc (A(4,2)) = 3500 + ((2 -l) * l)) = 3500 + (l l) * 4
= 3500 + 44 = 3544

ligrt:

is tvptesented in as 21) array, like A (i. jl. where i is a row and j is a

and n colutnns. then total cletncnts in a matrix or f,ize of a matrix is m cletncnts. This x n elenlents tv•quires x n units of Qtorage.

that is filled mostly zeroes (tnore than 2/3 elements are zero). is called as a sparse tnatrix.

Now, if there is a sparse Inatrix of order x n. and if wc use a 20 array of order m n to this store tnatri.x. then there is basically wastage of large amount of memory.

Now, a sparse nnattåx can be stored as a list of three tuples of the form (i. j. value).

In this can store only non-zero elements.

0 0 0 0 1 ()

2 0 0 0 0 0

0 0 0 7 0 0

0 0 0 0 0 30 Let us consider a matrix M(5x6)

Here the number of non-zero elements t = 5.

The array A [0. . .t, I ...31 can store all the non zero elements as given below:

	1	2	3
ALO)	5	6	
Alll	I	5	I
		1	2
A[31	3	2	—3
A[44		4	7
A[51	5	6	30

The element A [0, Il and A [0, 21 contain the number of rows and columns of the matrix. ALO, 31 contains total number of non-zero items.

2. Show that the function f(n) defined by: [WBUT 2007, 2010, 2016, 201B)

f (n) = —1)+— for n > I has the complexity O (log n).

Define Big — O, Q, 0 notations. Explain the conceptual differences among these three representatives. CWBUT 20071

And, Define Big O notation. (WBUT 2010, 20121

OSA-5

Answer:

I St Part:

f(n) = f(n- I)+ l/n

— f(n-2)+l/(n- I)+ l/n

= f(n-3)+1/(n-2)+1/(n-1)+1/n

.....

.....

= f(2-1)+1/2+1/3+.....+1/(n-1)+1/n

= 1/1 + 1/2 +1/3 +.....+ 1/11

These types of numbers are called Harmonic numbers.

We can evaluate this type of series just by integrating l/x from h to n + 1/2.

After integrating, we get the result as In(n + 1/2) In

1/2 z In n -0.7

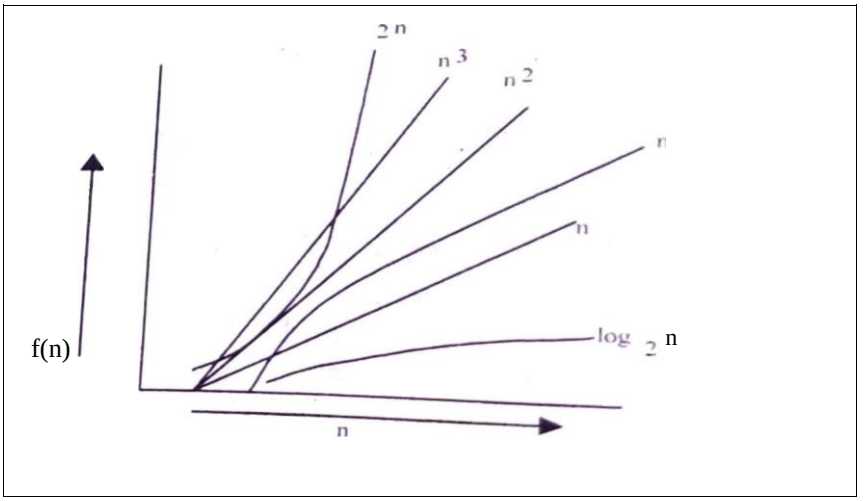
Hence, we can write the complexity as O(log n).

Big O notation:

The Big Oh (the "O" stands for "order Of') notation is used to classify functions by their asymptotic growth function and hence finding the time complexity of an algorithm.

There are two types of complexities of an algorithm, time complexity and space complexity. The time complexity of an algorithm is the amount of time the COtnP11ter requires to execute the algorithm. The space complexity of an algorithm is the amount of memory the computer needs to run to complete the execution of the algorithnu The algorithms are compared based on their performances. The performance is measured in terms of time complexity & space complexity. Based on the nature of the input, time complexity can be of three different types: best case, average case and worst case. The different computing functions are measured as:

n, n , n•, log 211, nlogy, 2". The rate of growth of all these functions is shown below:


2 n
n 3
n 2
n
log 2 n
f(n)
n

Let an) and g(n) be functions, where n is a positive integer. We write /(n) = Q (g(n)) if and only if gin) $= O(f(n)$

Let t(n) and g(n) be functions, where n is a positive integer. We write /(n) = O(g(n)) if and only it' g(n) = and g(n) = O(/(n)).

3. Derive values related to the average case and worst case behaviour of Bubble Sort algorithm. Also, confirm that the best case behavior is O(n). PNBUT 2007]

OR,

Write an algorithm for sorting a list numbers in ascending order using bubble sort technique and find its time complexity. [WBUT 20171

Answer:

The algorithm for bubble sort is given as follows: procedure:

```
BubbleSort (List [ ] )
 Inputs: List [ ] - A list of numbers Locals: i, j
        integers
 Begin :
       For i              0 to List. Size-I
            For j              i +1 to List. Size-I
                If List [i] > List [j ] , Then
                      Swap List [i] and List [j ]
                End If
           Next     j
 Next i Analysis:
```

The size of the sot-ting problem is related to the size N of the list. As N increases, we expect the execution time to increase as well.

The innermost instruction is to swap two list elements. Regardless of how long the list is. this always takes the same amount of time, and so the swap instruction is 0(1) with respect to N. Surrounding that instruction is an if statement, which either executes its body, or does not; in the worst case, it will execute its body. Again. the if statement is

The if statement is executed as many times as the inner loop iterates, and the inner loop executes as many times as the outer loop iterates. During the first iteration (i = 0), the inner loop executes N — I times, during the second iteration (i = l), IV— 2 times. and so on.

$$O\Big(\underbrace{\sum_{i=0}^{N-1}(N-i-1)}_{\text{num ber-of-loops}} \times \big(\underbrace{1}_{if} + \underbrace{1}_{swap}\big)\Big) \; 0(N^2 \qquad \text{IV}).$$

time-per-loop

So the complexity is O(N2) in the average case as well worst case.

The best case would be if the list were already sorted.

There will be comparisons as it is but no exchanges and execution tinie is in O(N2). But if we keep a track of exchanges in each pass and terminate the program checking if no there are no exchanges, then the prograrin would require only one pass and max. (N-l)

comparisons are required in that single pass and we can say that the complexity i order of O(N).

[WBUT

4. What is Linear data structur

Answer:

A data structure is said to be *linear* if its elements form a sequence or a linear list.

Exatvples:

- Linked lists
- Stacks. Queues

The Operations that can be done on linear structures are:

- Trenvrsal: Travel thmugh the data structure
- Search: Trausrsal through the data structure for a given element
- Insertion: Adding new eletnents to the data structure • Deletion: Renioving an element from the data structure
- Sorting: Arranging the elements in some type of order

Aferging: Combining two similar data structures into one.

. Prove that, + O(g(x)) = O (max(f(x),g(x)) [WBUT 2009]

Answer:

Suppose q(x) is in O(f(x) + g(x)) then q(x) is in O(max(f(x), g(x)) because two times the greatest off(x) and g(x) is greater than or equal to the sum. More precisely: q(x) <= c(f(x) + g(x)) <= g(x)).

Next. suppose q(x) is in g(x)) then q(x) is in O(f(x) + g(x)) becallse the greatest of f(x) and g(x) is less than or equal the sum of the two (assuming the two functions are positive valued). In formulas:

q(x) <= c max (f(x),g(x),h(x)) <= c (f(x) + g(x)). Hence proved.

6. What is an Abstract Data Type? What do you mean by a Dynamic Data

Ansvt er: [WBUT 2011] An abstract data type (ADT) is an object with a generic description independent Of implementation details. If for a panicular collection of data only the structure of data and the functions to be performed on the data is defined but the implementation is not defined, then such a collection of data is called Abstract data type.

In dynamic Data Structure inemory allocation for the data structure takes place at the run time and only required amount of memory is allocated. E.g.:
Link lists, Stacks, Queues, Trees

7. T(n) = 4n2 + 3n log n, express T(n) in Big (O) notations.

Answer: In Big (O) notation the complexity is

[WBUT 20121

$O(n^2$

POPULAR PUBLICATIONS

1)SA-8

write the difference between at I [ ] and **a. [WBUT 20121 ßnswer:

difference between multidimensional arrays **(a[][])** and arrays of pointer **(**a)**

refers to an array of particular datatype arrays (a pointer to pointers to that datatatype)

is fact that multidimensional arrays are rectangular data structures, with the same dimension. This constraint necessarily present in an array of pointers. a comparison among Data Type, Abstract Data Type and Data structure.

9 Do [WBUT 2013]

or Answer:Data character type consists data types of the are values found it represents in nearly all and computers, the operations and defined they have upon well-definedit.. Integer

operations that can be done with them. For example. the integer data type has operations for addition, subtraction, multiplication, and division. The computer knows how to perform these arithmetic operations with any two integers because these operations are

part of the integer data type.

A data type can be considered abstract when it is defined in terms of operations on it, and its implementation is hidden (so that we can always replace one implementation with another for, e.g., efficiency reasons, and this will not interfere with anything in the program). Thus, speaking about such a type, we leave its implementation aside considering it irrelevant to the topic, unless we directly discuss the implementation.

A data structure is an arrangement of data in a computer's memory or even disk storage.

An example of several common data structures are arrays, linked lists, queues, stacks, binary trees, and hash tables. Data Structure is an implementation of ADT. Many ADT can be implemented as the same Data Structure.

10. What are the differences between linear and non-linear data structures?

[WBUT 2013, 2014]

Answer:

Data structure can be classified as linear and non-linear.

Linear data structure: The data structure is said to be linear, if its element forms a sequence or linear list. There are two basic ways of representing such structure in memory. Here the elements are traversed sequentially starting from the beginning.

(l) Linear relationship between data elements is represented by means of sequential memory locations. Ex: Arrays.

(2) Linear relationship between data elements is represented by means of pointers and links. Ex: Linked list.

Non-linear data structure: They are used to represent the data having a hierarchical relationship between elements. Here the elements are not traversed sequentially, rather they are traversed in a non linear fashion. For example, in case of the tree, we have to start from the root but we have to traverse either left subtree or right subtree, but not the both.

Ex: Graphs and Trees.

11. What are the characteristics of algorithm? Answer:

- Precision — the steps are precisely stated(defined).

[WBUT 20141

- Uniqueness — results of each step are uniquely defined and only depend on the input
- Finiteness the algoritlun stops after a finite number of instructions are executed.
- Input the algorithlll receives input.
- Out i) u t the algorithnn produces output.
- Generality --- the algorithm applies to a set of inputs.

12. a) Suppose one 2-D array is initialized as int a Base address is 4000. Find the location of element a in row major form and column major form.

b) Define Sparse Matrix. [WBUT 20141 Answer:

a) Assume that the size of each element is 4 bytes.

In row rnajor order the address will be

Loc (A(i,j)) = Lo + ((i—l) * n + 0—1)) *c

Therefore,

Loc (A(2,4)) = 4000 + l) * 7+ (4- l)) * 4000 + (7 + 3) * 4 = 4000 + 4040

b) Refer to Question No. 1(2'1'1 Part) ofShort Answer Type Questions.

13. What is the default return type of malloc( )? Why do we need to typecast it?

[WBUT 20161 Answer:

mal loc returns a void pointer (void *), which indicates that it is a pointer to a region of unknown data type.

The (char*) or (int*) is an explicit type conversion, converting the pointer returned by malloc from a pointer to anything, to a pointer to char or integer. This is unnecessary in C, since it is done implicitly, and it is recommended not to do this, since it can hide some errors.

14 4. If T (n) = ao + al x + +c13x3 + + anx'l , then prove T (n) =) .[WBUT 2017]

Answer:

To prove that f(n) @(g(n)) we have to find c l, c2 > 0 and no such that 0 Cig(n) S f(n) S C2g(n).

Now and c2 are two valid constants för

=>n₀·

1 = (2 — 21) ak

where

$$2 \bullet \max_{k \quad 1} \left\{ \sqrt[i]{\frac{|a_i|}{a_k}} \right| \quad 0 \quad i \text{ no}$$

Hence proved.

Lon Answer Type uestions

Data Type. [WBUT 2012, 2015] I. Write short note on Abstract

Answer: Refer to Question No. 6 ofShort Answer Type Questions.

2. Define pseudo-code. [MODEL QUESTION]

Answer:

Actual purpose of using pseudo-code is that it is easier for humans to understand than conventional programming language code, and that it is an efficient and environmentindependent description of the key principles of an algorithm.

pseudo-code is an artificial and informal language that helps prograrnmers develop algorithms. It is a "text-based" detail (algorithmic) design tool.l)seudo-code is a compact and informal high-level description of a computer programming algorithm that uses the structural conventions of a programming language, but is intended for human reading rather than

machine reading. Pseudo-code typically omits details that are not essential for human linderstanding of the algorithm, such as variable declarations, system-specific code and subroutines.

Examples:

1 . If student's grade is greater than or equal to 80 Print "passed"

Print "failed "

2. Set total to zero

Set grade counter to one

While grade counter is less than or equal to ten

Input the next grade

Add the grade into the total

Set the class average to the total divided by ten Print the class average .

3. Initialize total to zero

Initialize counter to zero Input the first grade while the user has not as yet entered the sentinel add this grade into the running total add one to the grade counter

POPULANPUBV'CATJON$

input the next qrade (possibly the sentinel)

counter

set t-he avevaqe the total divided by the counter print the average

no grades were entered'

4.

initialize passes to zero initialize failures to zero

initialize student

while student counter is less than or equal to ten input the next exam result if the student passed add one to passes

add one to failures add one to student counter print the number of passes print the number of failures if eight or more students passed print " raise tuition"

STACKS & QUEUES

& chapter at a Glance

stacks: A stack is an abstract data type, which declares two methods PUSH and POP. Stacks are implemented either by an array or by a linked list.

The pt SH operation allows us to insert data at the end of an array or linked list.
The POP operation allows us to remove data from the end of an array or linked list.
A STACK is called a last in first out (LIFO) system.

A stack may be represented by a linked list. The first node of the list will be the topnwst element of the stack and is pointed by a top pointer. This type of stack representation is •illed linked stack. The stack can be declared as follows: typedef struct linked_ list

```
    int data; struct linked_ list •next ;
) I stack;
1 stack *top;
```

- Various types of opre«ion: A mathematical expression involves constants (operands) and operations (operators).
  Infix notation: operand I operator operand2. A + I
  Prefix notation: operator operand I opcrand2, 4 A B
  Postfix notation. operand I operand2 operator. A B +
- Conversion from INF I \ to POS V HN expression: In order to convert the infix to its corresponding postfixes fortn•, we need to do the following <eps:

r Fully parenthesize the expression according to the priority of different operators.
Move all operators so that they replace their corresponding right parentheses. Delete all parentheses.
'Hie priorities of different o erators are L'isen below:

O era tors	Priority
Unarv - , unarv + , not (.	4
0/0 / and (& /	
1)	2
<, <=, >, >=, = =, !=	1

- Priority Queue: A priority queue is essentially a list of items in which each item has associated ssith it a priority. In general, different items may have different priorities and we speak of one item has ing a higher priority than another. Given such a list we can determine which is the highest (or the lowest) priority item in the list. Items are inserted into a priority queue in any, arbitrary order. However, items are withdrawn from a priority queue in order of their priorities starting with the highest priority item first. Two elements with the same Priority are processed according to the order in which they were added to the queue.
- De-queue: De-queue is a linear data structure, where insertions and deletions are made to or from either end of the structure.

Multiple Choice Type uestions

I. A linear list in which elements can be added or removed at either end but not in the middle is known as [WBUT 2007, 2009, 20101 a) queue b) deque c) stack d) tree

Answer: (b)

2. The prefix expression for the infix expression a * (b + c) / e — f d) is None

[WBIJT of these20081 a) /*a + bc –ef b) —l* + abcef c) —l*a + bcef

Answer: (a)

d) none of these

Answer: (c)

4. Priority queue can be implemented using 20111

a) array b) linked list c) heap d) all of these

Answer: (d)

5. Reserve Polish notation is often known as [WBUT 2012)

a) Infix b) Prefix c) Postfix d) none of these

Answer: (c)

6. The evaluation of the postfix expression [WBUT 20121

3, 5, 7, +, 12, $^{0}/0$ is

3. The number of stacks required to implement mutual recursion is CWBUT 20111

d) 3.17

Answer: (a)

7. The operation for adding an entry to a stack is traditionally called PABLJT 20131

a) Add b) Append c) Insert d) Push

Answer: (d)

8. The best data structure to evaluate an arithmetic expression in postfix form is

[WBUT 20131

a) Queue b) Stack c) Tree d) Linked List

Answer: (b)

9. The heap (represented by an array) constructed from the list of numbers 30, 10'

80, 60, 15, 55, 17 is - [WBUT 2014)

a) 60, 80, 55, 30, 10, 17, 15 b) 80, 55, 60, 15, 10, 30, 17

c) 80, 60, 30, 17, 55, 15, 10 d) none of these

Answer: (b)

10. The postfix equivalent of the prefix * + *ab*—cd is PWBIJT 2014, 2015' 2016]

a) alp + cd * b) abcd — * c) ab + cd *d) ah + cd *

Answer: (a)

11. The following sequence of operations is performed on a stack: push (1), push (2), pop, push (1), push (2), pop, pop, pop, push (2), pop. The sequence of popped out values are [WBUT 20141

a) 2, 2, 1, 1, 2 b) 2, 2, 1, 2, 2 c) 2, 1, 2, 2, 1 d) 2, 1, 2, 2, 2

Answer: (a)

12. Stack cannot be used to [WBUT 2015]

a) evaluate an arithmetic expression in postfix form

b) implement recursion

—NC) allocate resources (like CPU) by the operating system

d) convert infix expression to its equivalent postfix expression

11 Answer: (c)

13. The following sequence of operations is performed on a stack [WBUT2015] push(l), push(2), pop( ), push(l), push(2), pop( ), pop( ), pop( ), push(2), pop( ) the sequence of popped out values are

a) 2, 2, 1, 2, 1 b) 2, 2, 1, 1, 2 c) 2, 1, 2, 2, 2 d) 2, 1, 2, 2, 1

Answer: (b)

14. The deque can be used [WBUT 2116]

a) as a stack b) as a queue

c) both as a stack and as a queue d) none of these

Answer: (b)

15. Inserting an item into the stack when stack is not full is called .....operation and deletion of item from the stack, when stack is not empty is callecl ..... .operation. [WBUT 20171

a) push, pop b) pop, push

c) insert, delete d) delete, insert

Answer: (a)

16. To make a queue empty, elements can be deleted till [WBUT 20181

a) front = rear + 1 b) front = rear — 1

c) front = rear d) None of these Answer: (a)

Short Answer Type uestions

l. a) Convert the following infix expression into equivalent postfix expression using stack:

(A + B) * C – (D – E) / (F + G). [WBUT 2009, 2010, 20151

b) What is a Max Heap? [WBUT 20091

Ansm•r:

a)

Symbol scanned	Stack	Output
(	(	-
A	(	A
+	(+	A
B	+	AB
)	-	AB+
*	*	AB+
C	*	AB+C
-	-	AB+C*
(	-(	AB+C*
D	-(	AB+C*D
-	-(-	AB+C*D
E	-(-	AB+C*DE
)	-	AB+C*DE-
/	-/	AB+C*DE-
(	-/(	AB+C*DE-
F	-/(	AB+C*DE-F
+	-/(+	AB+C*DE-F
G	-/(+	AB+C*DE-FG
)	-/	AB+C*DE-FG+
NONE	EMPTY	AB+C*DE-FG+/-

b) A heap is a tree-based data structure that satisfies the h.eap property: if B is a child rude of A, then key(A) key(B). This implies that an element with the greatest key is aways in the root node, and so such a heap is sometimes called a max-heap.

2 What is a priority queue? [WBUT 2009, 2012, 20131 Nention the different design options for priority queue. [WBUT 2009] Answer:

A priority queue is essentially a list of items in which each item has associated with it a priority. general, different items may have different priorities and we speak of one item having a higher priority than another. Given such a list we can determine which is the highest (or the lowest) priority item in the list. Items are inserted into a priority queue in any, arbitrary order. However, items are withdrawn from a priority queue in order Of their priorities starting with the highest priority item first.

Two elements with the same priority are processed according to the order in which they were added to the queue.

Often a type of binary tree called a heap is used to store the items in a priority queue• A heap has a root node (usually draw at the top) and it has two children, a left child and a right child. Each node (parent, and left or right child) has a value associated with it, with he property that the parent's value is more than either of its children. When we add an

item to the heap we check to see if the parent's value is more or less than it. If it is more, then we swap the child with the parent. We check again, and maybe swap

again, until the tree is "balanced," (i.e. that it obeys the heap property).

There are some other methods of implementing priority queue which are not so efficient.

They are

Sorted list implementation:

unsorted list implementation: Keep a list of elements as the queue. To add an element, append it to the end. To get the next element, search through all elements for the one with the highest priority. .

3. Define dequeue?

[WBUT 2010] OR, What is dequeue?

[WBUT2015]

Answer:

DequeUe is a linear data structure, where insertions and deletions are made to or from either end of the structure.

4. Convert the following infix expression to postfix notation by showing the operator stack and output string after reading each input token:

$A*B+C*(D-E)-F*G$ CWBUT 20111 Answer:

S m bol scanned	Stack	Out ut
		AB
		AB*
C		AB*C
		AB*C
		AB*C
		AB*CD
		AB*CD

		AB*CDE
		AB*CDE-
		AB*CDE-*
		AB*CDE-*F
		AB*CDE-*F-
		AB*CDE-*FG
NONE	NONE	AB*CDE-*FG*-+

5. a) Consider the array int a [10] [1 0] and the base address 2000, then CalCUlate th address of the array a [2] [3] in the row and column major ordering. C'NBIJT 20121 Answer:

In row major ordering the linear offset from the beginning of the array to any given element A[row][column] can then be computed as:

offset = row*NUMCOLS + column Thus

will have address 2000 + 10+3 = 2023

In column major ordering the memory offset could then be computed as:

offset = row + column*NUMROWS Thus

will have address 2000 + 2+3* 10 = 2032

b) Write the

OR,

Why circular queue is better than simple queue? OMBUT 20161

OR,

Why circular queue is used over simple queue . [WBUT 20171

Answer:

Refer to Question No. 5 of Long Answer Type Questions.

6. Evaluate the following postfix expression: [WBUT 20131

advantage of **4, 5, 4, 2, ∧, +, *, 2, 2, ∧, 9, 3, /, *, –** circular queue over linear queue. [WBUT 20121

Write pseudo code for evaluating postfix expression.

Answer:

1 st Part:

4, 5, 4, 2, ∧, +, *, 2, 2, ∧, 9, 3, /, *, –
≡ 4, 5, 16, +, *, 2, 2, ∧, 9, 3, /, *, –
≡ 4, 21, *, 2, 2, ∧, 9, 3, /, *, –
≡ 84, 2, 2, ∧, 9, 3, /, *, –
≡ 84, 4, 9, 3, /, *, –
≡ 84, 4, 3, *, –
≡ 84, 12 –
≡ 76

2nd Part: Refer to Question No. 2 ofLong Answer Type Questions.

7. How many types of priority queues are there? Can you consider stack as a priority queue? If yes, how? [WBUT 2013]

Answer:

Mainly there are two kinds of priority queue: l) Static priority queue 2) Dynamic priority queue.

Stack can be considered as a priority queue where the priorities of each element being inserted are considered higher than the previous one. This will let it behave e like a LIFO structure (i.e., stack).

---

8. suppose a queue is implemented by an array. Write an algorithm to insert a new element at the kth position of the array. [WBUT 20151

Answer:

```
static int head, tail; // Declare global indices to head and
tail of queue static char theQueue [MAX_ SIZE] ; / / The queue

// Function: InitQueue( )
//  purpose:  Initialize  queue  to
empty.   //   Returns:   void   void
InitQueue ( )

       head = tail= -1;
}
//-----------------------------------------
// Function: Enqueue ( )
// purpose: Enqueue an item into the kth position of queue .
// Returns: TRUE if enqueue was successful
//          or FALSE if the enqueue failed.
//-----------------------------------------
int Enqueue (char ch, k)

       int c;
       // Check to see if the Queue is full
       if (isFu11 ( ) ) return FALSE;

       // Increment tail index tail + + ;
     for (c = MAX_SIZE     I; c--)array
     [C+I ]  array [C] ;
```

```
 array [posi t ion—I]    ch;
return TRUE;
```

Lon Answer Type uestions

1. Write the difference between stack and queue and implement the operations of priority queue. [WBUT 2007) Answer:

1 st Part:

A Stack is a (ordered) collection of items, where all insertions are made to the end of the sequence and all deletions always are made from the end of the sequence. In principle a stack is a container of data items, from which we get data items out in reverse order COmpared to the order they have been puf into the container. We can also say that the item that has been put last in is coming first out. That's why a stack is also called LIFO.

(Last In First Out list). we can as well say that the item, which is put first in the container is get last out (First In Last Out: FILO).

A Queue is a (ordered) collection of items, where all insertions are made to the end ofth sequence and all deletions always are made from the beginning of the sequence. In principle a queue is Container from which data items are retrieved out in the same order they are put in. This means that the queue is a container that preservesÄhe order of items put there. We can also say that the item that is put last into the queue is taken last out from the queue. we can also say that the item which is put first into the queue is taken i first out. (First In First out: FIFO).

2 nd Part:

A priority queue is an abstract data type that supports the following three operations: insenWithPriority: add an element to the queue with an associated priority getNext: remove the element from the queue that has the highest priority, and return it

/* C implementation using array of size MAX. Assume structure is as below*/ struct data

```
int      int val,p,o;
       }d[MAX];
```

front=rear=—l; / *initial values

/ * insertWi thPriority*/ void insert (data dl)

*/

if rear==MAX-1) print f ("Priority Queue is Full") •

rear + + .

d [rear] =dl; if (front==front=0 • data temp ; for (int i=front;i<=rear; i++) for (int j=i+l;j<=rear;j++)

```
if(d[i].p
{

    d[i]=d[j];
```

temp=d [i J ; d

[j] -temp;

if (d [i] . p)

if(d[iJ .o > d [j J . o)

```
temp=d [ i ] ;
d[i]=d[j];
d [j ] =temp;
```

```
data getNext ( )

     data dl; if ( front 1 ) print f
     ("Priority Queue is Empty") ;

        d1=d[front];
        if(front==rear)
             front=rear=-

     l; front++ ; return

     d1;
```

2, Write an algorithm to convert an infix expression to postfix using stack.

[WBUT 2008, 2018]

OR,

Write an algorithm to evaluate a postfix expression [WBUT 20171

Answer:

- Scan the Infix string from left to right.
- Initialise an empty stack.
- If the scannned character is an operand, add it to the Postfix string. If the scanned character is an operator and if the stack is empty Push the character to stack.
- If the scanned character is an Operand and the stack is not empty, compare the precedence of the character with the element on top of the stack (topStack). If topStack has higher precedence over the scanned character Pop the stack else Push the scanned character to stack. Repeat this step as long as stack is not empty and topstack has precedence over the character.
- Repeat this step till all the characters are scanned.

(After all characters are scanned, we have to add any character that the stack may have to the Postfix string.) If stack is not empty add topStack to Postfix string and Pop the stack. Repeat this step as long as stack is not empty. Return the Postfix string.

The C-function to an infix expression to it's equivalent postfisform is as given below:

```
void postfix (char x [ J )

  int i o; while (x
  [i] ! = '\0

    if (isalpha (x [i] )
    ) printf ( , x[i])
    else if (x [i]== ')

      while (s (top] != '(print f
      ( " %c", pop() ) ; pop ( )
      ; // delete ' symbol


        while )  >= icp(x print  f  Cc",
        pop() ) ; push (x (i] ) ;


      while   (top   >   0)
        printf(" %c", pop()
        ) ;


 int i sp (char a)

   if (a == 'return
(3) ; else if (a
 == 'return (2) ;
else if (a == '
return (1) ; else
if (a == 'return (
     1 ) ; else if
(a   t )  'return
(0) ; int icp(char
a)

  if  (a == '*
 return (3) ;
 else   if == '
 (a    return
 (2)     ; == '

           == '
```

```
else  if  (a
return (4) ;
else  if  (a
return (0) ;
```

3. a) Define circular queue. CWBUT 2008, 20181 Answer:
Refer to Question No. 5(lSt Part) of Long Answer Type Questions.

b) Write an algorithm to insert an item in circular queue. [WBUT 2008, 2018)
Answer:

```
void Qlnsert (int item)

    if (rear            N-l)

     print f ( " Queue Is Full " ) • return;

  CQ [ ++rear]      i t em; if
  (front     1) front         o;
```

c) What is input restricted Dequeue? [WBUT 2008, 20181 nswer:
An input-restricted Dequeue is one where deletion can be made from both ends, but input can only be made at one end.

4. a) What is a Stack ADT?
b) Write a C function of popping an element from a stack implemented using linked &Jist.
c) Explain three uses of stack data structure. [WBUT 2009)
Answer:
a) A Stack ADT is a (ordered) collection of items, where all insertions are made to the end of the sequence and all deletions always are made from the end of the sequence. In principle a stack is a container of data items, from which we get data items out in reverse

order compared to the order they have been put into the container. We can also say that the item that has been put last in is coming first out. That's why a stack is also called LIFO ((Last In First Out list). We can as well say that the item, which is put first in the container is get last out (First In Last Out: FILO).

b) Assume that the list is defined as below: typedef struct node *nptr; struct node

int data; nptr next;

/ * function pop, • * / int pop (nptr s) / *Function to pop the elements* /

nptr temp ; int Y;

POPULAR PUBLICATIONS

```
if(s->next==NULL)
        {
printf("Underflow on Pop");
return(-1)

Else
{
y=s->next->data;
temp= s — >next ;
s->next=temp->next;
free  (  temp)  ;
return (y) ;
```

c) Use of stack

Reversing Data: We can use stacks to reverse data.

(example: files, strings). It is very useful for finding palindrotnes. Consider the following pseudocode:

```
1)    read (da ta)
2)    loop (data not EOF and stack not full)
            1) push (data)
            2) read (data)
3)    Loop (while stack notEmpty)
            1) pop (data)
            2) print (data)
```

```
Converting Decimal to Binary :
Consider the following pseudocode
Read (number )
Loop (number > 0)
    1) digit = number modulo 2
    2) print (digit)
    3) ) number  number / 2
```

The problem with this code is that it will print the binary number backwards. (ex: 1 9 becomes 1 1001000 instead of 0001001 1.)

To remedy this problem, instead of printing the digit right away, we can push it onto the stack. Then after the number is done being converted, we pop the digit out of the stack and print it.

Evaluating arithmetic expressions.

In high level languages, infix notation cannot be used to evaluate expressions. We must analyze the expression to determine the order in Which we evaluate it. A common technique is to convert a infix notation into postfix notation, then evaluating it. This is done using stack.

5. i) What is Circular queue? ii) Write Q-insert algorithm for the circular queue. [WBUT 2010]

[WBlJT 20101

OR,

Write an algorithm to insert an element into circular queue. 20171

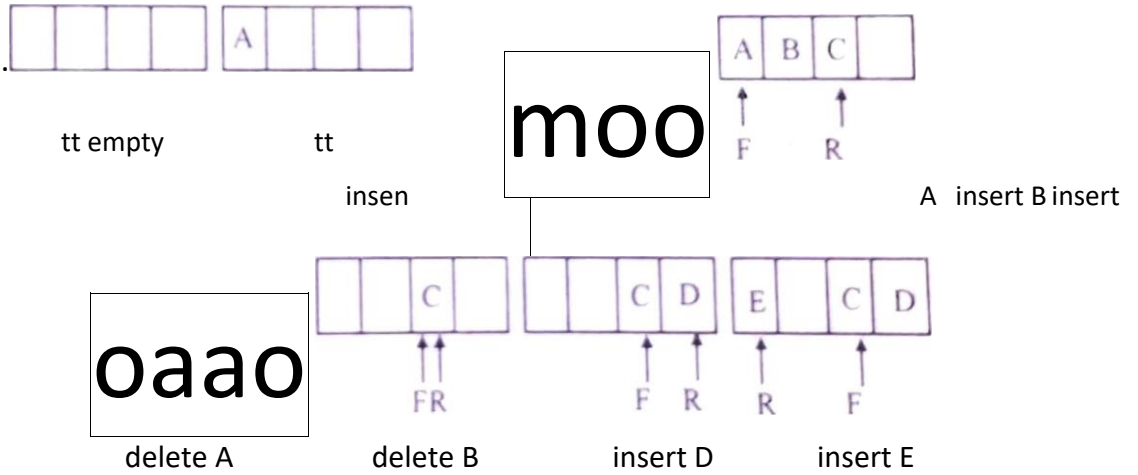
Answer:

i) The two algorithms QINSERT & QDELETE can be very wasteful of storage if the front pointer F never manages to catch up the rear pointer. Actually an arbitrary large amount of memory would be required to accommodate the elements. The method of performing operations on a Queue should only be used when the queue is emptied at certain intervals.

Let us consider the following sequence of operations. F and R represents the front and rear pointers.

To avoid this problel)l can think an alternatise representation of a queue, which prevents an excessive use of memory. In this representation elements are arranged as in a circular fashion where the rear again points to the front. This type of queue is known circular queues

So the above sequence of operations can be represented are shown below (in circular queue).



As we can see that the overflow issue in the previous case while inserting E is resolved by redirecting the rear to the front. This is the essence of circular queue.

ii) Let us now write the insert and delete functions of the circular queue implementation. Let us also assume that F and R represents front and rear pointers respectively void CQInsert (int item) if (rear == N-1print f ("Queue Is Full") ;

```
    return ;

  CQ [ + + rear]  i
  tem ; if ( front1)
  front  o ;

int CQDe1ete()

  int x; if ( front
  1)

    print f ( " Queue Is Empty" )
    ; exit (O) ;

  x = CO [front] ; CQ [
  front] 1 ; if ( front
    rear ) front  rear
    1 ;

    front   ++   ;
  return x;
```

In linear queue the condition for queue full is QREAR==MAXLIMIT

Suppose maxlimit is ten and queue is full now if we delete 9 element from queue then inspite of queue is empty we cannot insert any element in the queue. This wastage of memory is solved through circular queue where queue full condition is QREAR==Qfront+1

6. Define the ADT for stack. Show the implementation of the stack data structure using linked list.

[WBUT 20111 Answer:

The Stack ADT has two main functions:

Push input: a stack data object; an element of the base type output: a boolean indicating whether the operation was successful effect: the element is added to the sequence in the stack, in the top position

Pop input: a stack data object output: a boolean indicating whether the operation was successful effect: the element in the top position in the sequence in the stack is removed

Stack Implementation

First, let us define the linked list data structure. typedef struct linked_list

```
    int data;
```

```
    struct linked_ list *next;
) Node;
```

For us to be able to test our code, we need to define a way to display our stack. The following code uses recursion to display stack.

```
// recursively display the contents
// of the stack void DisplayStack
(Node* currentNode)

    // recursive termination condition
    if (currentNode NULL )

        return •

    // the node is not null // display the
 data print f ( " -s , currentNode->data)
 ; . // recursively call the display to //
 display the next element in the stack
    DisplayStack ( currentNode->next) ;
```

The code for pushing an element onto the stack is as given below.

```
// push item on the stack  //
this is same as adding a node
// at the top of the list void
Push ( int dataToAdd)

    // assumption: head is already defined elsewhere in the
    program
    // 1. create the new node
    Node * temp    new Node;
    temp->data dataToAdd ;

    // 2. insert it at the first position
    temp->next — head;

    // 3. update the head to point to this new node
    head = temp ;
```

The code for popping an element from stack is given below.

```
// pop an element from the stack
```



```
return NULL;
```

```
// this is same as removing the first element
// from the list
Node* Pop ( )

    // check for empty list if (head
    -- NULL) printf ( "Stack is
    empty\n") ;
```

```
 // get the top node
Node *firstNode = head;
// move the head head =
head->next; //
disconnect the node //
from the list firs
tNode->next = NULL;

 // return the top node
 return firstNode ;
```

7. What are the differences between stack and Queue? Write the algorithm for insertion and deletion in a circular Queue. [WBUT 2013]

Answer:

1st Part: Refer to Question No. I(lSt Part) of Long Answer Type Questions. 2nd Part: Refer to Question No. 5 of Long Answer Type Questions.

8. a) Evaluate the postfix expression using stack: [WBUT 20161

3, 16, 2, +, 12, 6, 1, -

b) Convert the infix expression into its equivalent prefix expression using stack:

$a+b*c+(d*e+f)*g.$

Answer:

a) The steps are as shown below:

3, 16, 2, +, *, 12, 6, /, -

Input	Operation	Stack contents	Details token (top on the right)
3	Push on the stack	3	
16	Push on the stack	3, 16	
2	Push on the stack	3, 16, 2	
+	Add	3, 1 8	
*	Multiply	54	
12	Push on the stack	54, 12	
6	Push on the stack	54, 12, 6	
/	Divide	54, 2	
-	Divide	54, 2	
(End of tokens)	(Return the result)	52	

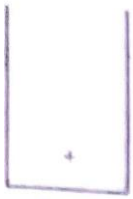
Pop two values: 16 and 2 and push the result 18 on the stack Pop two values: 3 and 18 and push the result 54 on the stack

Pop two values: 12 and 6 and push the result 2 on the stack

Pop two values: 54 and 2 and push the result 52 on the stack

Pop the only value 52 and return it

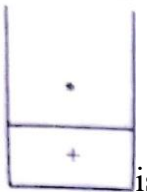
b) the symbol "a" is read, so it is to the output I ben is read and shed onto the Ktaek Next pu "b" is rend And pasqeqt through to the output i he state of affairs at this junctur• is as foll



Stack

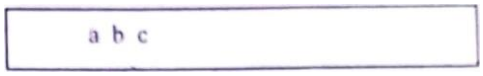
Scyt a is read. The top entry on the operator •tack has lower precedencc than nothing is output and is put on the stack. Next.

"c" is rea



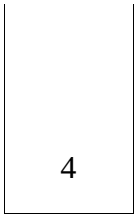
is t•ad and output. Thus far, have

Stack



Output

The next symbol is a Checking the stack, we find that we will pop a and place it on the output, pop the other which is not of lower but equal priority, on the stack, and then push the '+'.

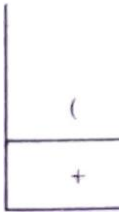


Stack

a b c . +

Output

The next symbol read is an which, being of highest precedence is placed on the stack. Then "d" is read and output.

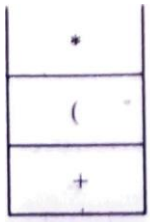


Stack

a b c *+d

Output

We continue by reading a Since open parentheses do not get remosed except when a closed parenthesis is being processed, there is no output. Next, "e" is read and output.



Stack

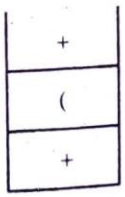
( )utput

a b e * + d e

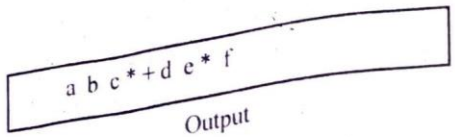
1)SA-29

POPULAR PUBLICATIONS

The next symbol read is a '+'. We pop and output '*' and then push '+'. Then we read and output.



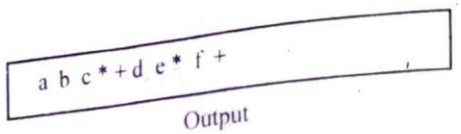
Stack



Output

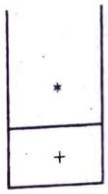
Now we read a ')', so the stack is emptied back to the '('. We output a '+'.

Stack

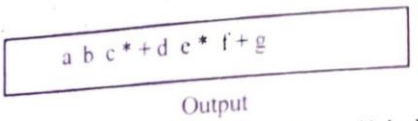


Output

We read a '*' next; it is pushed onto the stack. Then "g" is read and output.



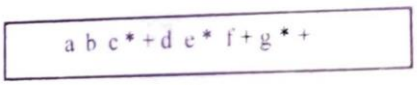
Stack



Output

The input is now empty, so we pop and output symbols from the stack until it is empty.

Stack



Output

9. Convert the infix expression 9+5*7−6^2+15/3 into its equivalent postfix expression and evaluate that postfix expression, clearly showing the state of the stack. [WBUT 2017]

Answer:

Symbol scanned	Stack	Output
9		9
+	+	9
5	+	95
*	+*	95
7	+*	957

Now we read a 'Y, so the stack is emptied back to the we output a 'V.

We read a '*' next; it is pushed onto the stack. Then "g" is read and output.

Output

a b c * + d e* f + g

()utput

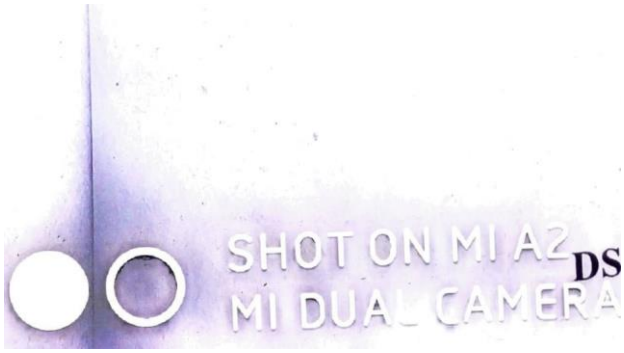
Stack

Symbol scanned	Stack	Output
-	-	957*+
6	-	957*+6
^	-^	957*+6
2	-^	957*+62
+	+	957*+62^-
15	+	957*+62^15
/	+/	957*+62^15
3	+/	957*+62^153
NONE	NONE	957*+62^153/+

read

ty.

stfix
f the
71


SHOT ON MI A2
MI DUAL CAMERA
DS

LINKED LIST

☞ Chapter at a Glance

- **Singly linked lists:** A linked list is a data structure where data can be represented as a chain of nodes. Each node of a linked list contains two parts: one is the address part another is the address of the node which is next to or previous to the

- Singly linked lists: A linked list is a data structure where data can be represented as a chain of nodes. Each node of a linked list contains two parts: one is the address part another is th data part. The address part holds the address of the node which is next to or previous to the

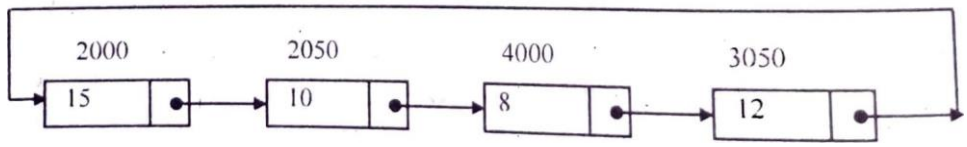
urrent no

Typically the first node of a linked list is called the HEAD node. If the head node is destroyed then the entire list gets destroyed as well. Depending on the traversal requirement a linked list can be designed as circular, bi-directional etc. In it's simplest form the following diagram shows the structure of a uni-directional linked list also known as singly linked list. e Several opera tions: (Insertion, deletion, Traversing, Searching)

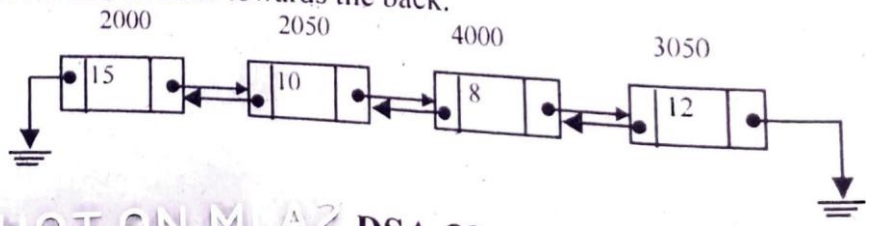
- Insertion: The algorithm for inserting in the above manner is given below:
  Let us assulne that x is data value to be inserted after the node containing the data value y. p initially contains the address of the head node.

  Step l: start traversing the linked list from the head node
  Step 2: if the data value of the head node is not equal to y goto step 8 Step
  3: create a new node
  Step 4: place x to the data field of the new node
  Step 5: place the next address of the head node to the next address of the new node
  Step 6: change the next address of the head node by the new node address Step
  7: Goto Step 13
  Step 8: Identify the address of the node containing data value y as follows
  Step 8a) if the data value of the next node of p is equal to y
  Step 8b) else change p by its next node address
  Step 9: create a new node
  Step 10: place y to the data field of the new node
  Step I l: place the next address of p node to the next address of new node
  Step 12: change the nex't address of the p node by the new node address
  step 13: End

• Circular linked list: the circular linked list, the address of the last node contains the address of the first node, as shown in the figure below:



- Doubly linked list: The node ofa doubly linked list contains two -link fields, instead of one One link is used to point to the predecessor node, i.e., the previous node & the other link is used to point to the successor node, i.e., the next node. so, the two links are directed, one towards the front and another towards the back.



Linked representation of polynomial: In general any polynotnial A(z) can be written as

Each a, xl is Ith term of A(x) the = polynonlial, ao+ alX + a2Xwhere ^{2} + - x - is + variable, an-tx + a, anx'.is ts coefficient & e is the exponent. $f\ a_i=0$,then the term is zero term, otherwise it is nonzero erm.

Multiple Choice Type uestions

1. In a circularly linked list organization, insertion of a record involves the modification of WBUT 2008, 20181

a) no pointer b) 1 pointer c) 2 pointers d) 3 pointers

Answer: (c)

2. Linked list are not suitable for BUT 2012, 2016]

a) Stack b) Deque c) AVL Tree d) Binary search

Answer: (d)

3. Dynamic memory allocation use BUT 2012]

a) Calloc b) Malloc c) Free d) all of these

Answer: (d)

4. Which type of linked List contains a pointer to the next as well a; previous node in the sequence? [WBUT 2013]

a) Singly Linked List b) Circular Linked List

c) Doubly Linked List d) All of these

Answer: (c)

5. Linked list is not suitable data structure for which one of the following problems? [WBUT 20141

a) insertion sort b) radix sort

c) binary search d) polynomial addition

Answer: (c)

is used in defining [WBUT 2015]

6. Self-referential pointer b) a node of linked-list a) an array

d) all of these

c) a queue

Answer:(d)

7. Inserting a node after a given node in a doubly linked list requires[WBUT 2016, 2018] exchanges b) two pointer exchange:

a) four pointer d) no pointer exchange
c) one pointer exchange

Answer: (b)

[VBUT 2017]

8. Binary search is not b) possible linked listfor c) stack d) (ueue
a) array

Answer: (b)

[WBUT 20171

9. A linear link list can be traversed using
a) recursion b) both (a) and (c) are correct
c) stack d) both (a) and (c) are wrong

Ans»er: (a)

[WBUT 20171

10. The data structure used to solve recursive problem is d) none of these
a) Linked list 5) Queue c) Stack

Ans»cr: (c)

Short Answer e uestions

1. Write an algorithmto add two polynomials. [WBUT 2007)

Answer:

Let us assume that thr tsso polynomials are represented using linked list and the reSUltant
is also using a linked ist.

Let pheadl, phead: and phead3 represent the pointers of the three lists under consideration.

Let each node contai- two integers: exp and coff

Let us assume that the two linked lists already contain relevant data about the two polynomials.

Also assume that ve have got a function append to insert a new node at the end of the given list.

```
pl = phea€l;
p2 =
phead2;
```

Let us call malloc o create a new node p3 to build the third list

```
p3 = pheld3;
```

```
/* now traverse tle lists till one list gets exhausted */
while ((pl NUuL) Il (p2 NULL))

 / * if the expolent of pl is higher than that of p2 then the next term in final list is going to be the nde of PI* / while —>exp > p2 - > exp )

   1^3 exp      pl -> exp; 13 -> coff = pl -> coff .cppend (p3, phead3) ; , now move to the next term in list 1* / pl = pl  next;

/ * if p2 expcnent turns out to be higher than make p3 same as p2 and append to final list* /
while (pl ->exp <                p2 -> exp )

- > exp __        p2 - > exp ;
13 - > coff __   p2 ->                                    coff .
```



```
append (PA , pheadj) ; p2 = p2 -s next; consider the possibility that both exponents are same , then we must add the coctlicients to get the tern) for the final liq 4/ while (pl - »e,xp
* now  p2 exp )

p3-> exp = pi - > exp; p 3-»coff = pi-scoff p2-> coff

          append (p3, phead3) .PI      pl -snext: .p2 = p2-»next
                                ;
```

* now consider the possibility that list2 gets exhausted , and there are terms remaining only in listl e So all those terms have to be appended to end of list3. However, one does not have to do it term by term, as pl is already pointing to remaining terms. so simply append the pointer pl to phead3 */ if (pl ! = NULL) append (PI, phead3) .else append (p2, phead3) ; .

2. Write the key features of circular linked list and state why it is important in case of Josephus problem. [WBUT 20071

Answer:

A circular list node is identical to a singly-linked list node. However, the circular list has a reference to its tail node instead of its head node. The tail node has a reference to the head node. This makes it possible to get to both ends of the list in constant time.

Josephus problem is stated as:

There are people standing in a circle waiting to be executed. After the first man is executed, certain number of people are skipped and one nnan is executed. Then again, people are skipped and a man is executed. The elimination proceeds around the circle (which is becoming smaller and smaller as the executed people ate removed). until only the last Inan remains, who is given freedom.

The task is to choose the place in the initial circle so thal you sun ise (are the last one remaining).

The easiest and most logical way to solve the problem is to simpl} sinutlate it using a circular link list as shown below. / *linked list structure * / struct node { int item; . node* next;

function* / .int simple_ simulation (int N, int M)

```
int i, node •t=new node
( ) ; node •x=t;    t;
t->item = I; t-snext
//Constvuct the list for
(i

  x      (x-»next = new
  node x->item i; x-»next
   t;

/ / Find the
survivor while (x !
= x->next)

  for (i x->next ; x->next =
    x = x->next->next; N--

return x->item;
```

```
if
(p-
>data x)

b
r
e
a
k
;
```

[WBUT 2008]

3. What are the advantages of linked list over an array? Write an algorithm to insert a data X after a specific data item CWBUT Y in a 2008, linked 2016, list.20181

Answer:

1 st Part:

The advantages of linked list over an array are:

Refer to Question No. 5 ofShort Answer Type Questions.

2nd Part:

Assuming the linked list already contains n integer elements. The algorithm to insert a data X after a specific data item Y in a linked list is as follows:

```
node *insertXafterY (node *p, int y, int x)

 node *q,  *r,  •km;


 r (node* )malloc (sizeof (node) )
 ; while (p->next ! = NULL)
```

POPULAR

}

P = p->next;

r->7
r->n

11es

HOW c list? snswer:
We can re ointer t ap

5 What
array? Answer: Linked li
1. Linke store 2. Linke

3. Insert
or rer need

Linked Il l. Arræ betw
2.
3. Fixe

mem

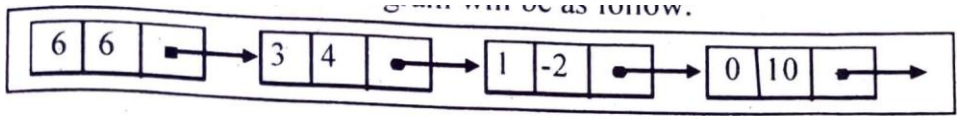
4. Inse
5. Ace

6. write conditi(Answer struct

```
r->data = Y;
        = m > next ;
 ->next
return (q) ;
```

4. How can a POIynomial such as 6x6 + 4x3 —2x+10 be represented by a linked list? [WBUT 2010, 2015] Answer:

We can represent the polynomial by a linked list, where each node contains deg, coef and a pointer to the next node. So, the diagram will be as follow:



5. What are the advantages and disadvantages of linked list data structure over array?CWBUT 2010] Answer:

Linked list advantages over array

l. Linked lists do not need contiguous blocks of memory; extremely large data sets stored in an array might not be able to fit in memory.
2. Linked list storage does not need to be preallocated (again, due to arrays needing contiguous memory blocks).
3. Inserting or removing an element into a linked list requires one data update, inserting or removing an element into an array requires n (all elements after the modified index need to be shifted).

Linked list disadvantages over array:

l. Arrays have contiguous memory allocation which makes it easy to access elements in between.
2. As memory is allocated during compilation makes the program faster
3. Fixed in size so if we are aware of the exact size of datas then there can be no memory wastage which is also an advantage linked list has.
4. Insertion and deletion at the end of the array is easy but not in between.
5. Accessing data is easy. Example, a[2].

6. Write a C language function to delete nth node of a singly-linked list. The error conditions are to be handled properly. [WBUT 20111

Answer:

```
struct node* DeleteNode (struct node* head, int n) {
    struct node* temp = head; int length =
   LinkedListLength( temp) ; int i; if (n <= 0
   | n > length) { print f ( "ERROR: Node does
   not exist! \ntt ) ; if (n== 1)
```

POPULAR PVOUtCATtONS t h od (ise node) to

f

/ through list

node
}else{
for(i

•next. ;

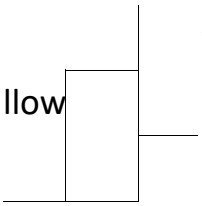
a) Hew the polynomial 4x 10x2 +3 can be represented using a linked list?

7.

b) Compare and contrast between an array and a single linked list. [WBIJT 20121

Answer:

a) We can rcptxncnt the polynonlial by a linked list. VOIere each node contains deg, and a pointer to the next node. So. the diagram will be as



b) Rcfcr to Question No. S ofShort Answer Type Questions.

8. What is a self referential structure? [WBUT 20121

Ansucr:

A self-referential structure is one of the data structures which refer to the pointer to (points) to another structure of the sarne type. For exatnple, a linked list is supposed to be a self-referential data structure. The next node of a node is being pointed, which is of the same struct type. For exatnple, typede± struct listnode { •eel d •da t a; st Y uet 2 istnode *next;) hnked_list;

In the above exarnples the listnode is a self-retZ•rential structure — because the *next is of the type sturct listnode.

9. write an algorithm to find the largest and smallest element in a single linear list.

Ansuer: LWBUT 2014)

Linked List definition in C:

typedef struct linked_list

int data; struct linked_ list *next;

) Node;

(•ode to find largest and smallest elements
//finds the and smallest in the list
// assumes that head pointer is defined elsewhere

```
   MaxMinInList ( int *max, int *min)
int

   //  start  at  the  root  Node
   *currentNode  =  head;  if  (
   currentNode NULL ) return 0;
   // list is empty.
   // initialize the max and min values to the first node

   * max  min  currentNode->data ; // loop through the
list for (currentNode    currentNode->next; currentNode !
= NULL ; currentNode     curren tNode- >next )

       if (currentNode->data > *max)
            max     currentNode- >data ;
       el  sé  if  (currentNode->data  <
       *min)
          •k min currentNode->data ;

    //  we  found  our  answer
    return 1 ;
```

10. Write an algorithm to delete the last node of a linked-list. [WBUT 20151

Answer:

C function to delete last node is as given below:

```
struct node *delete ( struct node *head)

  struct node *temp
  =head; struct node *t;
     (head->next==NULL
  if                 )
    free     (head)     ;
    head=NULL ;


     while ( temp->next ! NULL )

        t= temp  ;  temp=

     temp- >next ; free

     (t->next) ;
```

```
    t->next=NULL;

return head;
```

11. Write an algorithm to insert an element in the middle of a linked list.[WBUT 20151

Node Structure for singly Linked List:

```
struct node ( int data; struct node
        *next;
    NULL;
```

• Singly Linked List

Insen node at middle position .

voxd insert mid ( )

```
insert_mid()

int pos,i;
struct node *new_node,*current,*temp,*temp1;
new_node=(struct node *)malloc(sizeof(struct
printf("nEnter the data : ");
scanf("%d",&new_node->data);
scanf ( " , &      ; new_node - > next =NULL
;

printf ( "nEnter the position scanf ( "
        , &pos) ; if (pos>Z (length( )
+1) )
        {
        printf("nError
                        : pos > length n ) ;
    goto st;
    }
if(start==NULL)
    {
    start=new_node;
    current=new_node;
    }
else
    {
    temp = start;
            for(i=l; i< pos-I;i++)

            temp = temp->next;
            }
    temp1=temp->next;
    temp->next = new_node; new_node->next=templ ;
```

node)) ;

12. What is the difference between array and linked-list? tWBlJT 20151 Answer:

The primary difference between an array or ArrayList and a linked list is that the array 01 ArrayList permits random access into the list structure using a subscript; each data ite has a "named" storage location (named by the subscript). In contrast, with a linked list a locations are relative to the next or previous locations. With an array or ArrayList. we ca access the 10th data item, say, with one probe to subscript 9 (indexing zero-up). with

linked list, we cannot access the 1 0th item without first traversing the first nine items. searching through a linked list is thus inherently a sequential process.

On the other hand, because all locations are relative, additions and deletions to a linked list can be made without affecting any of the data items already stored in the list. insertions and deletions require only that the next and previous pointers be changed exactly as needed; no data needs to be moved. When a deletion takes place, no hole is left. The deleted node becomes unused memory.

13. Show how linked list can be used to add the following polynomials:

514 $5x^3 + 10x^2 + 8x$ + 513 + 3

$2x^2 + 7x + 8.$ [WBUT 2016]

31^3 +

Answer:

Let us assume that the two polynomials (pl and p2) are represented usin} linked list and the resultant is also using a linked list.

Let each node contain two integers: exp and coff.

The two linked lists already contain relevant data about the two polynomials. We will create a linked list list3 (p3) with a single node to begin with.

Steps: We traverse, the lists till one list gets exhausted

l) if the exponent of pl is higher than that ofp2 then the next term in final list is going to be the node of pl

2) if p2 exponent turns out to be higher than make p3 same as p2 and append to final list

3) now consider the possibility that both exponents are same , then we must add the coefficients to get the term for the final list

4) now consider the possibility that list2 gets exhausted , and there are terms remaining only in listl . So all those terms have to be appended to end of list3. The final solution would be $5x^A4 + 8x^A3 + 12x^A2 + 1\ 5x + I$ l based on the above algorithm.

14. How a polynomial such as 8x5 +413 —912 +21—17 can be represented using a linked list? What are the advantages and disadvantages of linked list over an array? CWBUT 20171

Answer:

1st part:

We can represent the polynomial by a linked list, where each node contains deg, coef and a pointer to the next node. So, the diagram will be as follow:

5|8 →3|5→ 2|-9→1|2→0|-17

2nd part: Refer to Question No. 5 of Short Answer Type Questions.

ONS

15. Compare and contrast linked list with static and dynamic array. [WBUT 2018]

Answer:

In contrast, linked lists are dynamic and flexible and it can expand and contract is size In an static array, memory is assigned during compile time. While in a dynamic array and a linked list, it is allocated during execution or runtime. Elements are stored consecutively in arrays whereas it is stored randomly in linked lists.

Compare

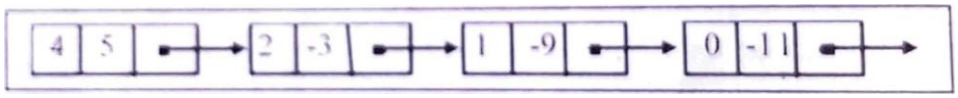
1. An array is the data structure that contains a collection of similar type data elements whereas the linked list is considered as non–primitive data structure contains a collection of unordered linked elements known as nodes.
2. Accessing an element in an array is fast, while linked list takes linear time, so it is quite a bit slower
3. Arrays are of fixed size. Linked lists are dynamic and flexible and can expand and

Linked lists arc dynatnic and flexible and can expand and eon:tact 4 tn addition memory utilization is inefficient in an array. Conversely. memor..v zation iscfVtctent tn thc linked list.

Lon Answer e uestions

Is a) How can a polynomial such as $5x^4$ -3x2 +9x—I l be represented by a linked [WBUT 20071 list?

We can tvpresent the poly nonual by a linked list. here each node contains deg. eoetf:ctent and a pointer to the next node. s;o, the diagratn ill be as töllow:



b) Write an algorithm to delete a node from a doubly linked list, where a node contains one data and two address (prev & next) portion. LWBUT 20071

OR,

Write an algorithm for deletion of a node from a doubly-linked list. (WBUT 2011)

Answer:

```
/ • C $ unction*/ vo•a
deletenode (node*n
```

```
        np =
        nn = n->prev;
n- >next; np->next
= n->
nn> prev n —prev;
```

1)SA-42

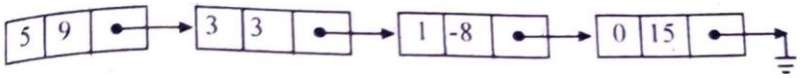
2. a) DO you COnsider the following data-structure as linear? Circular doubly linked .

b) list.Represent the following polynomial by linked list (show the diagram only).

9x5 — + 1 .

c) Write an algorithm to delete all nodes having value greater than X CWBUT from a 2008)given singly linked list. answer:

a) Linear. because traverse all the list of elements sequentially.

b) 

c)

node *removeNodeAfteri (node *p, int x)

// r holds the address of the previous node

node *q,r , m;

while (q! = NULL)

if (q->next == Nif (q->data

> x)

r - >next = NUfree

(q) ; break; else if (q->data >

x)

r->next = m->next; free (g);

=r;

r g;

=g->next ;

return g;

DSA43

2014)

3. a) Represent the given polynomial using a link list.

$3x^4 + x^2 - 5x + 2$

b) Write the pseudo code / C code for adding two polynomials (already given by user, no need to take input). Also comment on the complexity Of your algorithm

Answer:

a) We can represent the polynomial by a linked list, where each node contains deg, coet and a pointer to the next node. So, the diagram will be as follow:

b) Refer to Question No. I ofShort Answer Type Questions.

Running is @(max(x,y)•(x+y)) where x and y are the number of terms in the polynomials Polyl and Poly2 respectively.

4. a) Write an algorithm for creating a linked-list with n nodes. [WBUT 2017)

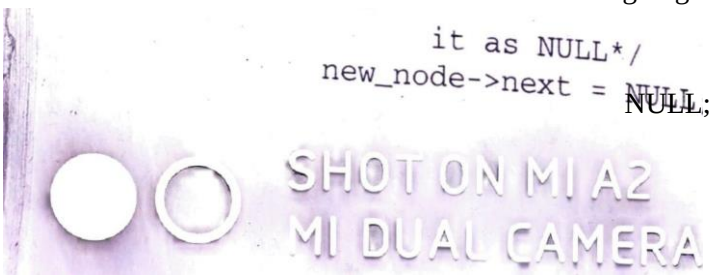
b) How it can be made a circular linked-list? Write a function for that purpose.

Answer:

a) //C functions showing how to create a linked list with n nodes with values I to n in sequence

```
# include<stdio. h>
#include<stdlib.h>

// A linked list node struct Node

   int   data;   struct   Node
   *next;


/* Given a reference (pointer to pointer) to the head of a list and an int, appends a new
node at the end * void append (struct Node* * head_ref, int new_data)

      / * 1. allocate node * /
      struct Node* new_node
Node) ) ;=(struct Node* ) mal loc (sizeof (struct
       struct Node *last

                                    =
                                                *head_ref ; / * used in step 5* /
    / * 2. put in the data * new_node->data
   =new_data ;
```

3. This new node is going to be the last node, so make next of



```
    * 4.If the Linked List is empty, bead * / then make the new
node as

   if ( *head.ref   NULL )

       *head_ref   =   new_node   ;
       return;


    /*5. Else traverse till the last node *
    / while (last->next ! = NULL) last
      last->next;

    /*6. Change the next of last node *
    / last->next = new_node ; return;

/*this    creates    a
list* / int createList
( )

  / * Start with the empty list *
/ struct Node* head = NULL ; for
(i =0; i<n; i++)
//  Insert  data  at  the  end
  append ( &head, i) ;


b) // Function that convert singly linked list
// into circular linked list.
struct Node* circular (struct Node* head)

    // declare a node variable start and
    // assign head node into start node
    .
    struct Node* start = head;

    // check that while head->next not equal
    // to NULL then head points to next node
    .
    while  (head->next  !  =  NULL)
        head = head->next;
```

```
// if head->next points to NULL then
// start assign to the head->next node
.
head- >next     start;
return start;
```

sAtnesp"ll 5. How a linked-lists can be used to implement stack?following things before

To implement stack using linked list, we need to set the
a Node' structure with two members data and next.

step 2: Define a Node pointer 'top' and set it to NULL.Menu with list of operations and make
Step 3: ltnplement the main method by displaying suitable function calls in the main method.

we can use the following steps to insert a new node into the stack...

Step l: Create a newNode with Empty given value.(top -NULL)

Step 2: Check whether stack set is newNode -e next NULL.
Step 3: If it is Empty, then
Step 4: If it is Not Empty, then set newNode —+ next top. Step 5:
Finally, set top = newNode.

pop() - Deleting an Element from a Stack we can use the
following. steps to delete a node from the stack...
Step 1: Check whether stack is Empty (to NULL).

step 2: If it is Empty, then display "Stack is Empty!!! Deletion is not possible!!!" and
terminate the function
Step 3: If it is Not Empty, then define a Node pointer 'temp' and set it to 'top'.
Step 4: Then set 'top = top —i next'.
Step 7: Finally, delete 'temp' (free(temp)).

6. Write short notes on the following:
a) Linear Lists
b) Circular link list [WBUT 20131 Answer: [WBUT 2015]

a) Linear Lists:

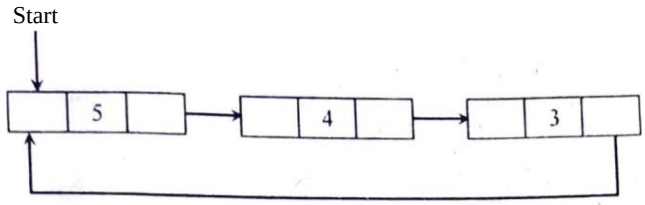
A linear list stores a collection of objects of a certain type, usually denoted as the elements of the list. The elements are ordered within the linear list in a linear sequence. Linear lists are usually simply denoted as lists.

Unlike an array. a list is a data structure allowing insertion and deletion of elements at an arbitrary position of the sequence. If the position in question is given, for example by a reference, such a modification takes only a constant number of operations, that is, no effortful copying of entries is necessary and all insertion and deletion operations take an equally short time. Conversely, however, one cannot access a single element via an (integral) index in constant time, as in the case of an array, without having searched for it before and having received a reference to it. Furthermore, lists are not limited to a certain maximum number of elements from the beginning on (like an array).

b) Circular link list:

circular Linked List is Divided into 2 Categories. o Singly Circular Linked List
o Doubly Circular Linked List

- In Circular Linked List Address field of Last node contain address of "First Node".
- In short First Node and Last Nodes are adjacent.
- Linked List is made circular by linking first and last node, so it looks likecircular chain as shown in following diagram.



- Two way access is possible only if we are using "Doubly Circular Linked List" • Sequential movement is possible • No direct access is allowed.

Chapter at a Glance

TREES

- Tree: JA trvc is a finite set of one or tnore nodes connected by edges such that
  i) is a specially deQgnated node called the root
  i)) retnaining nodes atws paflitioned into n (3=0) mutually exclusive disjoint sets
- Basic Terminology:

  i) Vodc: Each data itetll in a tree is called a node. It specifies the data information & links (branches) to other data iterns. In the e.xarnple of Fig (l) the tree has 13 nodes.

  ii) Root: It is the parent of all other nodes of a tree. A root node (loes not have any parent iii) Degrcc era node: The nunlber of sub trees of a node is called its degree. In our example the root node has three sub trees (B, C, D). Hence, the degree of the root node A is 3. Degree of B is 2. Degree of C is l . And so on.

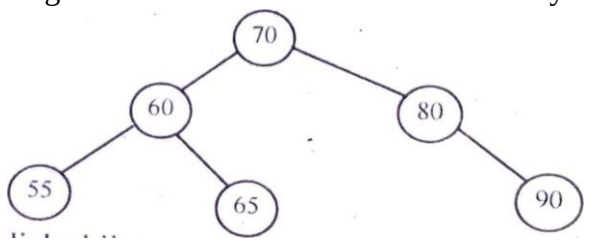
  iv) Dcgrcc Ma tree: It is the ma,ximum degree of nodes in a given tree. In our example the root node A has the highest degree. Hence, the degree of the tree is 3.

  v) Terminal node(s): A node with degree zero is called a terminal node(s) or leaf node(s) or external node(s). In our exannple E. F, G, I are leaf nodes.

  pi) Non Tenninal node(s): Any node whose degree is not zero is cal led non-terminal node(s) or internal nodes or interior node(s). In our exatnple A, B, C, D & H are the non-terminal node(s).

  vii) Path: It is a sequence of consecutive edges from the source node to the destination node. In our example the path between A & I is given by the edges (A, D) (D, H) & (H, I).
- Binary tree: A binary tree is either empty or consists of one single vertex called the root, together with two disjoint binary trees called left sub-tree & right sub-tree. Number of branches of any node is either zero or one or at most two.
- Binary search tree: A binary search tree (BST) is a binary tree that is either empty or every data node that forms the tree satisfies the following conditions.
  i) The data in the left child of a node is less than the data in its parent node.
  ii) The data in the right child of a node is greater than the data in its parent node.

The left and right sub trees of the root are also binary search trees. The fig. shown below is a Binary search tree.



- Threaded Binary tree: In linked-list representation of binary trees, additional storage space is required to store the two links of each node. The null pointers just results in vs astage Of tnemory. To overcojne this drawback, the null pointers are replaced by appropriate pointer values called *threads*.

Multi le Choice e uestions

1. If a binary tree is threaded for in-order traversal a right NULL link of any node is replaced by the address of its

a) successor b) predecessor c) root [WBUT d) own2007, 20161

Answer: (a)

2. In a height balanced tree the heights of two sub-trees of every node never differ by more than

a) 2 CNBUT 2007, 2009, 2018]

Answer: (c)

3. Maximum possible height of an AVL Tree with 7 nodes is

[WBUT 2008, 2013, 2016, 20181

Answer: (a)

4. The in-order and post-order traversal of a binary tree are DBEAFC and DEBFCA respectively. What will be the total number of nodes in the left subtree of the given

[WBUT 20081

d) None of these

Answer: (d)

5. A binary tree is a special type of tree tWBUT 20081

a) That is ordered

b) Such that no node has degree more than 2

c) For which both (a) and (b) above are correct

d) In which non-leaf nodes will have degree 2

Answer: (b)

B-tree is BUT 2008, 20181

6. A

a) Always balanced b) An ordered tree

c) A direct tree d) All of these

Answer: (d)

7. Which tree structure is used for efficient access of records residing [WBUT in 20091 disc memory? b) B Tree c) 2-3 Tree d) Binary Tree

a) AVL Tree

Answer: (b)

8• Which of the following is essential for converting an infix expression [WBUT to postfix 20091 notation? b) An operand stack a) A parse tree d) None of these

c) An operator stack

Answer: (c)

9. The values in a BST can be sorted in ascending order by using which of th following traversals? (WBUT 20091

a) Pre — order b) In - order c) Post - order d) Level - order

Answer: (b)

10. The prefix expression for the infix expression a (b+C) / e f is (WBtJT 20091

a) / + bc — ef + abcef

c) a + bcef d) None of these

Ansyter: (c)

Il. In array representation of Binary tree, if the index number of a child node 6 then the index number of its present node is (WBUT 2010, 2014)

Answer: (d)

12. The depth of a complete binary tree wrth n nodes (WOUT 20121

a) log(n+1)–1 b) Jog (n) c) log(n–1)+1 d) log(n)+1

Ansvser: (a)

13. In a binary search tree . if the number ot nodes of a tree 9, then the minimum height of the tree is2012) d) None of these

Ans"er: (d)

14. Which method of traversal does not to hold nodes that are waitin to be processed. (WEBUT 20121

a) Breadth first b) Depth ...,,, first

c) D. search d) None of these

.Anssser: (a)

15. Which of the foljowjng is essential for converting an infig eapressi00 to the postfix expression efficjentiy? ('ABUT 20131

a) An operator stack b) An operand stack

c) An operand stack and operator stack d) A parse tree

-Ari"ser: (a)

16. Number of ail possubje binary trees with 4 nodes js CNBUT 20141

a) 13 b) 12

An•ser: (c)

17. If the Inorder and preorder traversal of a binary tree are D, B, F, E, G, H, A, C and A, B, D. E, F. G. H. C respectively then, the postorder traversal of that tree IS

a) D, F, G, A, B, C, H, E

c) C, G, H, F, E, D, B, A d) D, F, H, G, E, B, C, A

Answer: (d) b) F, H, D, G, E. B, c, A 20141

.18. The a) 4082number of possible b) 4084distinct binary trees c) 3082with 12 nodes is d) 3084 2015)

Answer: (b)

19. A binary tree has n leaf nodes. The number of nodes of degree 2 in this tree is CWBUT 20151

a) logn d) cannot be said

Answer: (d)

20. The minimum height of a binary tree of n nodes is BU T 20161

b) ni2 c) ni2 -2 d) log2(n+1)

Answer: (d)

21. The number of edges in a full binary tree of height h is CWBUT 20171

- I

c) 2^{n+1}-2

Answer: (c)

22. Minimum number of nodes required to make a complete binary tree of height h is CWBUT 20171

Answer: (b)

23. Which one is required to reconstruct a binary tree? CWBUT 20171

a) Only inorder sequence

b) Both preorder and postorder sequences

c) Both inorder and postorder sequences

d) Only postorder sequence

Answer: (c)

24. Number of nodes in a complete binary tree of depth k is [WBUT 20181

b) 2k C) 2^k-1 d) None of these

Answer: (c)

Short Answer e uestions

1. Prove that for any non-empty binary true T, if no is the number of leaves and n
   be the number of nodes having degree 2, then no = +1. [WBUT 20071

Answer:

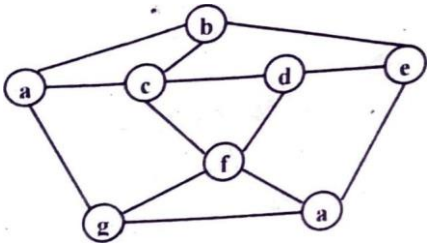
Let p and B denote the total number of nQdes & branqhes
in T.

Let no, n l, n2 represent the nodes with no children. single child, and two childr
respectively.

n= nå+n I +n2,

$B=n$I +2n2 ==>n I
+2172+1 n, n I +2112+ I I +n2
==> $n0=n2+1$

2. Explain spanning tree and create a spanning from the following graph.[WBUT 20071

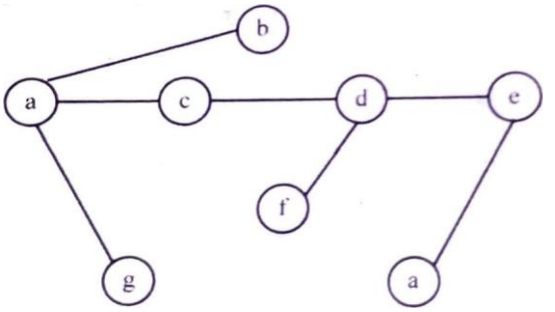


Answer:

A spanning tree of a graph is a subset of edges that form a tree. Exannples of various spanning trees from their graphs are shown below: ncnsuNZSS

z • 77xxxx

The spanning tree for the given graph is as follows:



3. Do you consider the following data-structure as linear? [WBUT 20081 . Binary tree Answer:

Nonlinear, traversing the nodes in a nonlinear pattern, root, then left or right, any one like that. So in worst case the traversing will be the height of the tree.

4. Show how the following integers can be inserted in an empty binary search tree in the order they are given:

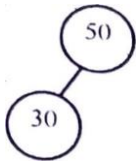
50, 30, 10, 90, 100, 40, 60, 20, 110, 5.

Draw the tree in each step.

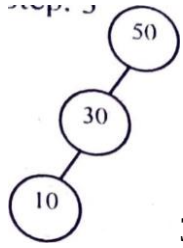
[WBUT 20091

Answer:

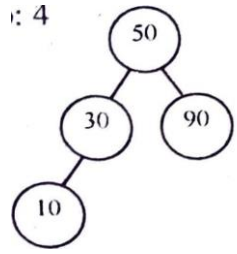
step l: 50 Step: 2



Step:

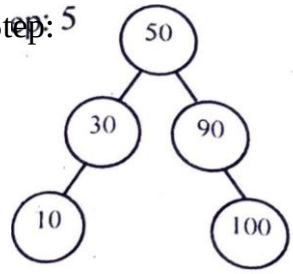


3

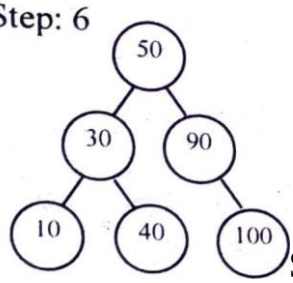


Step: 4

Step: 5

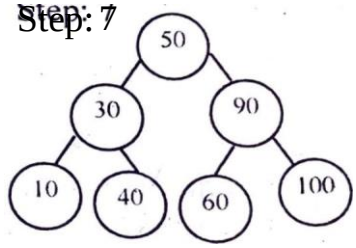


Step: 6

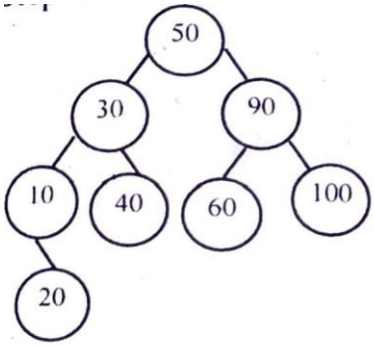


Step: 6

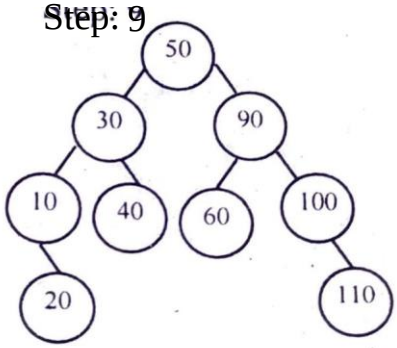
Step: 7



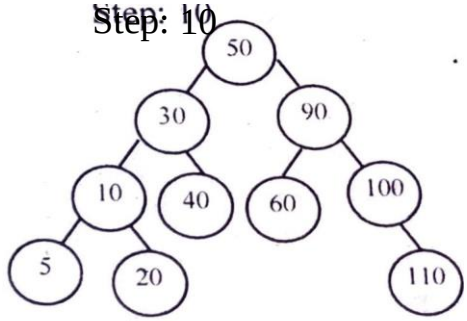
Step:



Step: 9



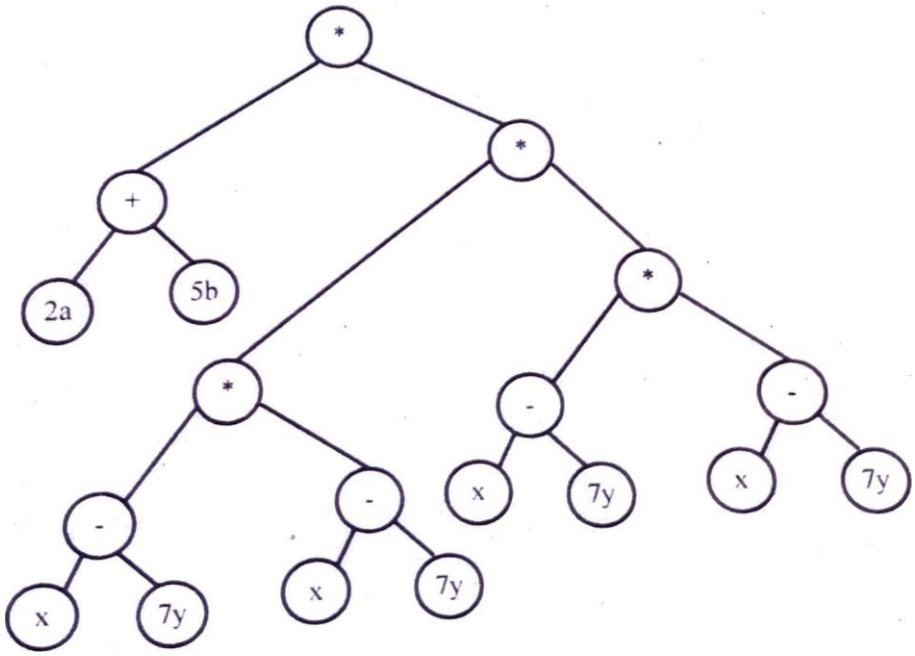
Step: 10



8

5. Construct the expression tree for the following expression: (WBUT 20101

$$E = (2a + 5b)(x - 7y)^4$$



Answer:

6. Write an algorithm for non-recursive in-order traversal of a threaded binary tree.

[WBUT 2011, 20181 Answer:

Step-I: For the current node check whether it has a left child which is not there in the visited list. If it has then go to step-2 or else step-3.

Step-2: Put that left child in the list Of visited nodes and make it your current node in consideration. Go to step-6.

Step-3: For the current node check whether it has a right child. If it has then go to step-4 else go to step-5

Step-4: Make that right child as your current node in consideration. Go to step-6.

Step-5: Check for the threaded node and if it's there make it your current node. Step-6: Go to step-I if all the nodes are not over otherwise quit

7. Write a C language function to find the in-order successor of the root of a binary tree. [WBUT 20111 Answer:

Let the structure of any node be and let d be the root node. struct

Node

```
        int data;
Node*                                                   successor (Node * d)
        Node* Iptr;             // pointer to the left subtree
        Node* rptr ;            // pointer to the right subtree
      if (d   NULL) return NULL ;
```

```
// If d has a Right Child if (d—>rptr ! =
NULL)

    // Move to Right Node d = d->rptr;

    // Move to the extreme left while
(d->lptr ! = NULL) d = d—>lptr;

    return d;
```

Below are two functions — first one to find out the minimum key in the binary tree and the second one to find the inorder successor of the root which uses first function.

```
Node *treeminimum (Node *root, Node *nil)

    / * returns a pointer to the minimum key in a NONEMPTY tree * / while (root->left ! = nil).
```

```
    root    =    root->left;
  return (root) ;

   *treesuccessor (Node *root, Node *nil)
```

Node

```
  /*returns pointer to inorder successor of root, and to
    nil if root is the maximum of the tree. This code
    assumes that root ! = nil. * /

  Node *treeminimum (Node * , Node * ) • Node
  Y; if  (root->right  !  =  nil)  return  (
  treeminimum ( root->right , nil) ) ; y = root-
  >par'ent;  while (Y nil && root Y- >right )

      root = Y;
      Y = Y — > parent;

  return (y) ;
```

8. Write an algorithm to test whether a given binary tree is a binary search tree.

[WBUT 20111

Answer:

The solution is to check, for every node in the tree, the min and max key values of the nodes in its left sub tree is less than the value of its key and the min and max key values in its right sub tree is greater than the value of its key. This can be achieved by recursion.

The idea is to "bubble up" the min and max values of a node's sub tree, after satisfying the conditions above, to its parent node, which will in turn repeat the same process. The algorithm is designed around pre-order tree traversal and solves the problem in O(n) (linear) time.

Algorithm

For the current node:

Step]: Get the min and max values for left sub tree if left child exists

Step2: If the min and max values are greater or equal to current node's key, return "Not BST"

Step 3: If there is no left child, set the min value to current node's key value

Step 4: Get the min and max values for right sub tree if right child exists

Step 5: If the min and max values are lesser or equal to current node's key, return "Not BST"

Step 6: If there is no right sub tree, set the min value from step I and the max value to current node's key value
Step 7: Return the min value from step I and max value from step 6

9. Write an algorithm to left rotate a binary tree. [WBUT 2011]

Answer:

Input: A binary tree u. (u may be a subtree of a larger tree).

Output: A (new) binary tree obtained from u by performing a left rotation about u. If empty or has an empty right subtree, an empty tree is returned. is Algorithm:

Bin Tree rotateLeft(BinTree u)

if (u == nil or rightSubtree(u) == nil) return nil; v rightSubtree(u); return buildTree(v,rightSubtree(v),huildTree(u, leftSubtree(v), leftSubtree(u)));

10. What is binary tree? Construct a binary tree using the Inorder and Postorder traversal of the iven node below;[WBUT 20121

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

HC A Answer:

I St Part:

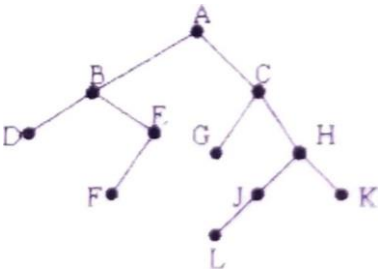
A binary tree is either empty or consists of one single verte.x called the root touether with two disjoint binary trees called left subtree & right subtree. Number of brancheq any node is either zero or one or at most two.

Binary trees are used to implement binary search trees and binary heaps.

2 nd Part:

Based on the postorder traversal. we find A is the root. Now. DBF E is the left child and GCLJHK is the right child of A.

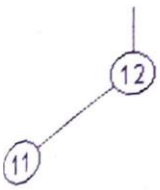
The root of right subtree would be the second last element in preorder i.e., C We can now further divide the right subtree(with C as root), giving {G} as right subtree and {L J H K} as left. In this way the final trees is given below:

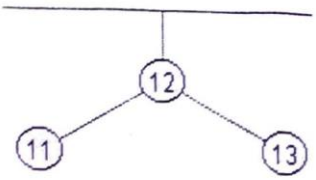


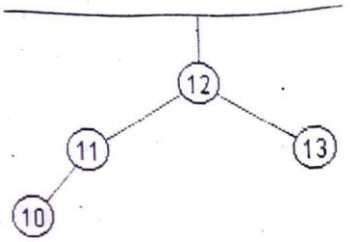
11. Construct an AVL tree using the below list. Show all the steps 12, 11, 13, 10, 09, 15, 14, 18, 7, 6, 5, 4 NVBUT 2012, 2015]

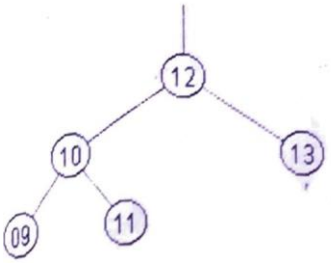
Answer:

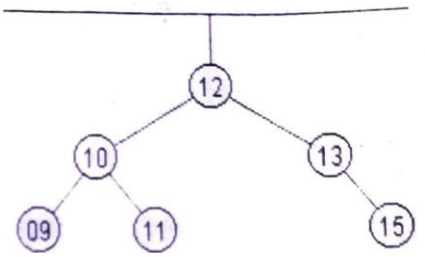
The steps are as shown below:

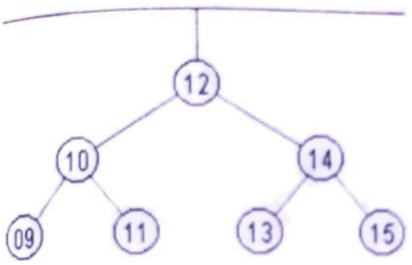

12
11

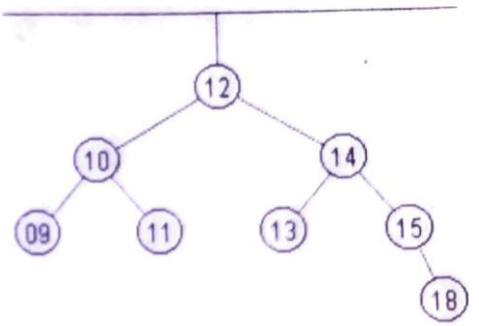

12
11
13

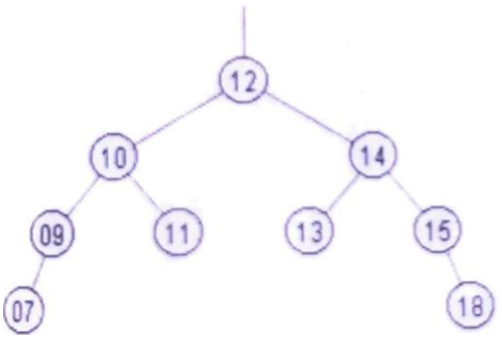

12
11
13
10

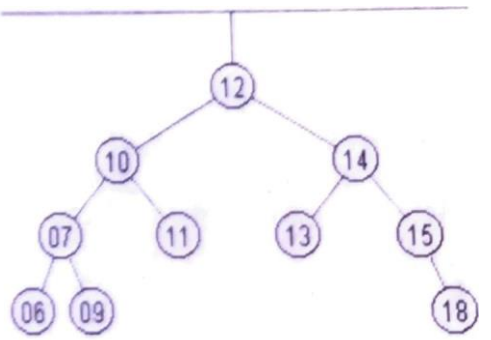

12
10
13
09
11

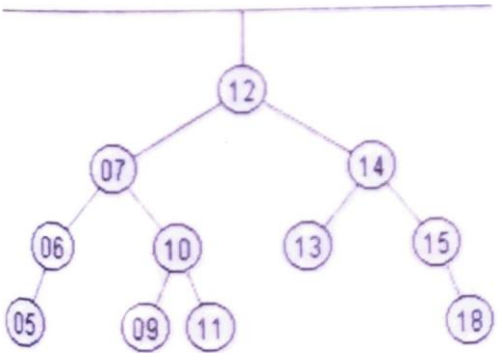

12
10
13
09
11
15

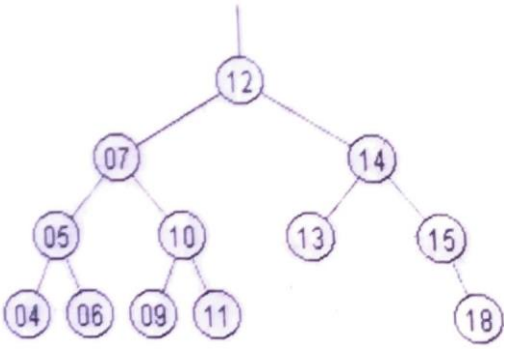

12
10
14
09
11
13
15


12
10
14
09
11
13
15
18


12
10
14
09
11
13
15
07
18


12
10
14
07
11
13
15
06
09
18


12
07
14
06
10
13
15
05
09
11
18


12
07
14
05
10
13
15
04
06
09
11
18

12. Prove that the maximum no. of nodes in a binary tree of depth k is 2k — 1

[WBUT 2014)

Answer:

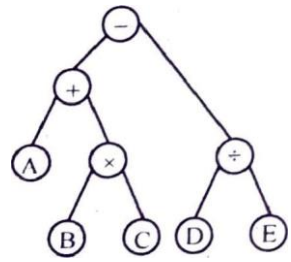
The maximum number of nodes in a binary tree in level i is 2i (according theorem) Hence the max'imum number of nodes in a binary tree of depth k is , summation of Inaximum number of nodes from level 0 upto level k — l, i.e.,

$$= 20 + 21 + 22 +$$

$$= 2^0 * \left(\frac{2^k - 1}{2 - 1} \right) \left[a \cdot \left((r^n - 1)/(r - 1) \right) \right.$$

13. For the following expression draw the corresponding expression tree:

a+b*c—d/e [WBUT 20151 Answer:



14. a) Does a B tree grow at its leave or at its root? Why?

b) In deleting a key from a B tree, when it is necessary to combine nodes?

[WBUT 2016]

Answer:

a) B Tree grows at its root (bottom up). B-trees are balanced search trees designed to work well on magnetic disks or other direct-access secondary storage devices. In a Btree, new nodes are inserted into the leaf level however, if the B-tree must increase in height, it is the root level or top of the tree that changes. This automatically preserves the balance of the tree and no rotation is necessary.

b) If a node has the minimum number of keys, then deleting a key from the node will cause an underflow and it would violate the B Tree property. It needs to be merged to another node to fix the B Tree back.

15. The post-order and in-order traversal sequences of codes in a binary tree are given below:

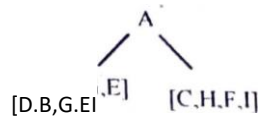
Post-order: D G E B H F CA

Pre-order: DB G EAC H F I

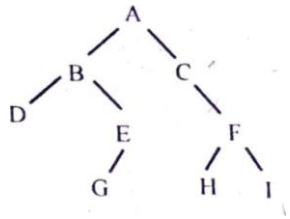
Construct the binary tree. [WBUT 2016]

snswer:

1) We first find the last node in post-order. The last node is "A", we know this value is root as root always appear in the end of postorder traversaL
2) We search "A" in in-order to find left and right subtrees Of root• Everything on left of "A" in inorder is in left subtree and everything on right is in right subtree.

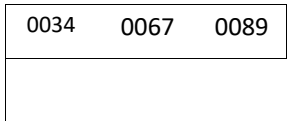


3) We recur the above process for two subtrees.

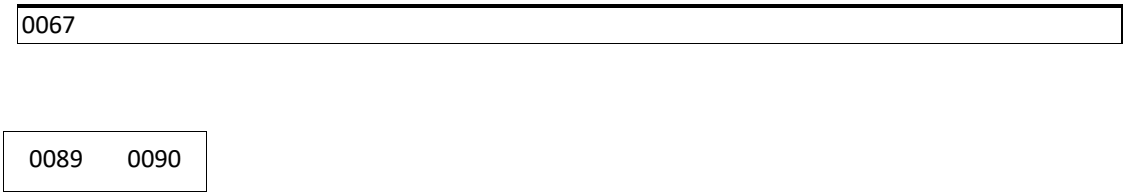


16. construct one B-Tree of order 4 with the following data. 34, 67, 89, 90, 100, 2, 36, 76, 53, 51, 12, 10, 77, 69. [WBUT 20161 Answer:

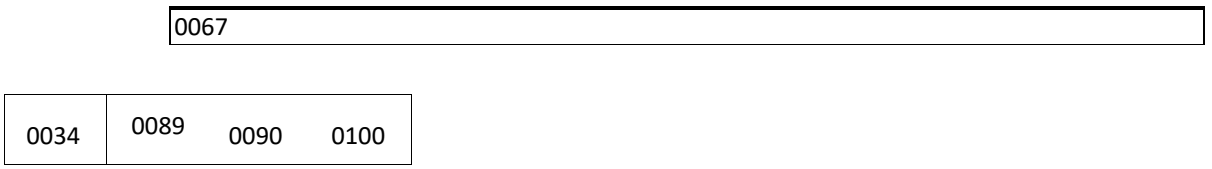
After inserting 34, 67, 89



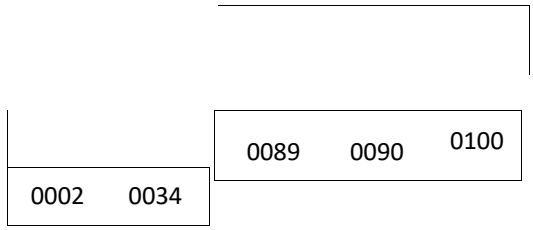
After inserting 90



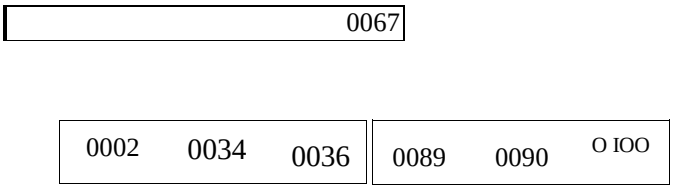
After inserting 100



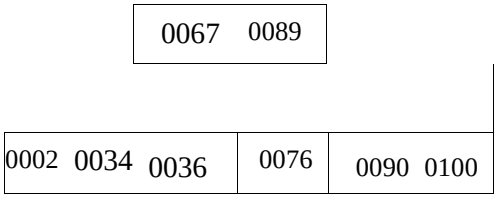
After inserting 2


0089
0090
0100
0002
0034

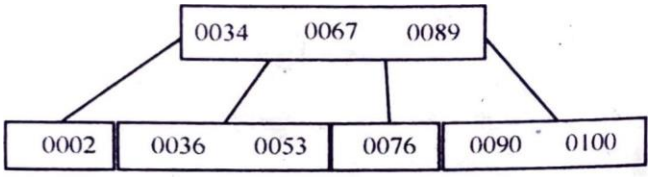
After inserting 36



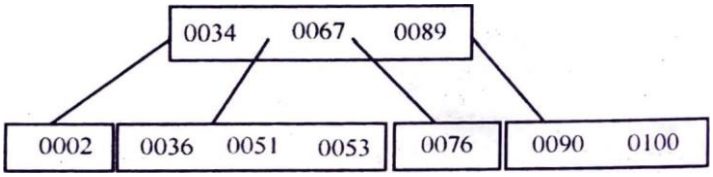
After inserting 76



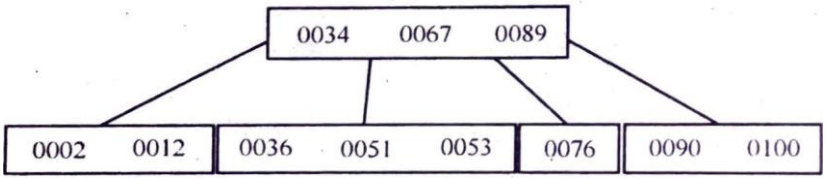
After inserting 53



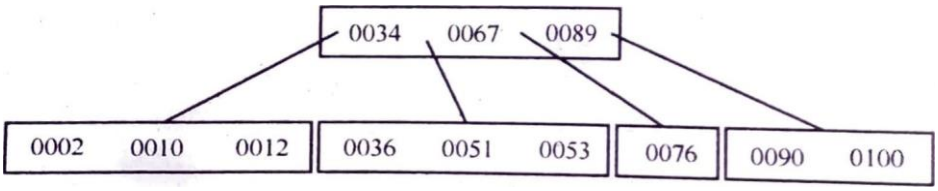
After inserting 51



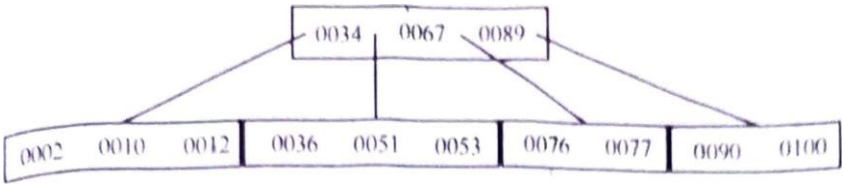
After inserting 12



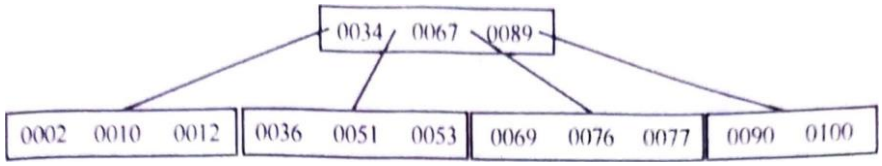
After inserting 10



inserting 77

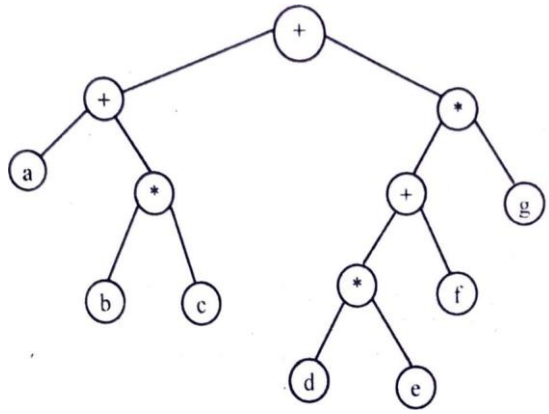


After inserting 69



construct a tree from the given postfix expression abc* +de* f + g* +

[WBUT 2016]



Answer:

Postcode traversal: ab c * + d e *f + g * +

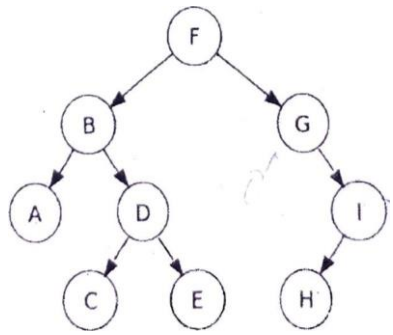
18. The inorder and preorder tree traversals are given. Draw the binary tree.

Inorder: ABCDEFGHI

Preorder: FBADCEGIH

Is it possible to build up a unique binary tree when its preorder and postorder traversals are given? [WBUT 20171

Answer: 1 st part:



2nd part: Refer to Question No. 13(b) (Pt Part) of Long Answer Type Questions.

19. Insert the following keys into a B-Tree of given order mentioned belo

a, f, b, k, h, m, e, s, r, c. (Order 3)

a, g, r, b, k, d, h, m, j, e, g, l, r, x, c, I, n, t, u, p. (Order 5) Answer:

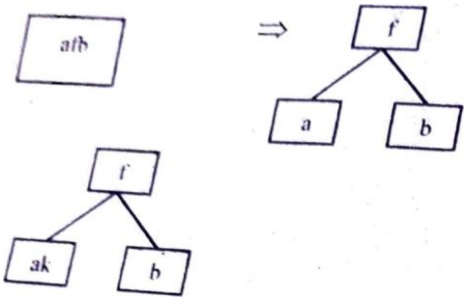
I st part:

Construction of B-tree: (of order 3)

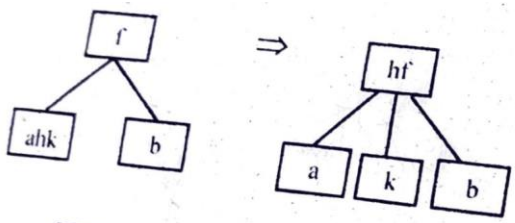
a, f, b, k, h, m, e, s, r, c

a:

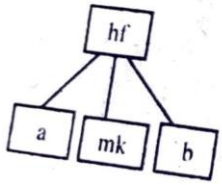
f:



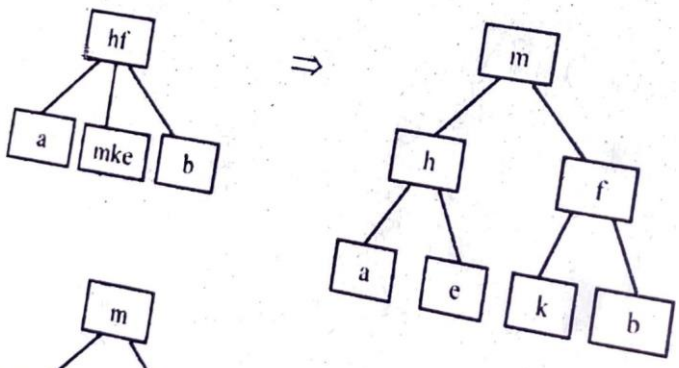
h:



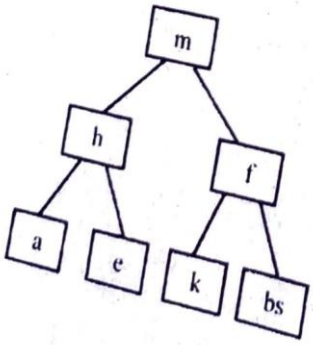
m:

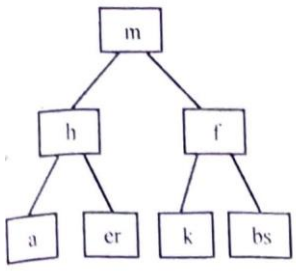


e:

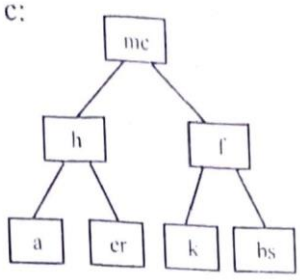


s:



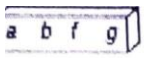


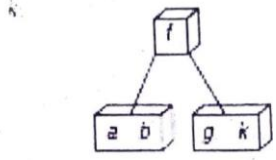
c:

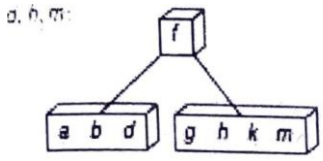


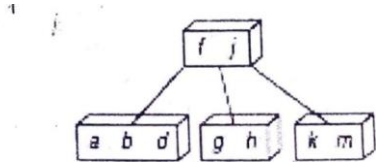
2Ü'l l'grt:

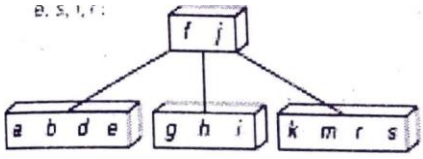
The insettion steps are n in the figure below:

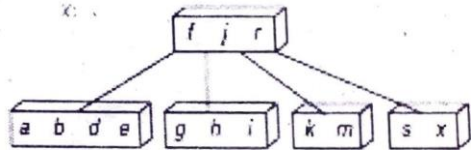

a b f g

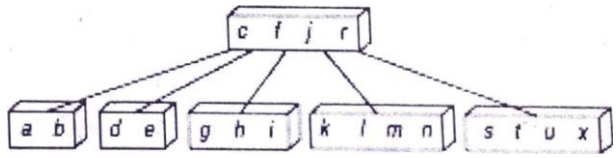

f
a b
g k

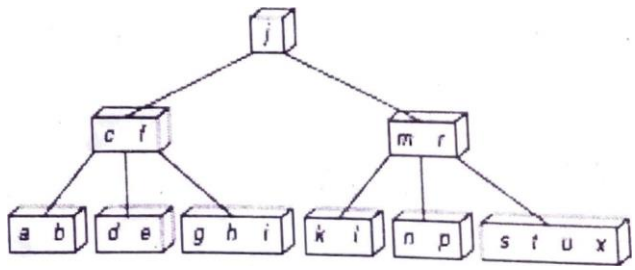

f
a b d
g h k m


f j
a b d
g h
k m


f j
a b d e
g h i
k m r s


f j r
a b d e
g h i
k m
s x


c f j r
a b
d e
g h i
k l m n
s t u x


j
c f
m r
a b
d e
g h i
k i
n p
s t u x

20. What is expression tree? Draw the expression tree and write the In, Pre & Post. Order traversals for the given expression tree: E = (2x +Y) (5a — b)3. [WBUT

Answer: 1

Part:

An expression tree is a representation of expressions arranged in a tree like data . Structure It is a tree with leaves as operands of the expression and nodes contains the operators Data interaction is also possible in an expression tree.

2nd Part: Refer to Question No. 2(c) of Long Answer Type Questions.

Lon Answer Type uestions

I. Write an algorithm to insert a node in a binary search tree. PNBIJT 2007, 20141

Answer:

Implementing binary search tree.

We assume that a tree node has left and right pointers and a key element.

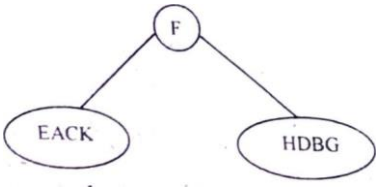
```
SearchTree Insert( ElementType X, SearchTree T )

     if( T    NULL )

       / * Create and return a one-node tree * /
           T = mal loc( sizeof( struct TreeNode )
            );if( T - NULL )
                Fatal Error( "Out of space...


             T—>E1ement = X;
            T->Left = T->Right = NULL;


      if( X < T->E1ement )
          T—>Left   Insert( X, T->Left ) ,

     if( X > T—>E1ement )
          T->Right   Insert( X, T->Right ) ;
   / * Else X is in the tree already; we'll do nothing * /
     return T; / * Do not forget this line! ! * /
```

2. a) The inorder and preorder traversal sequence of nodes in a binary tree are given below.

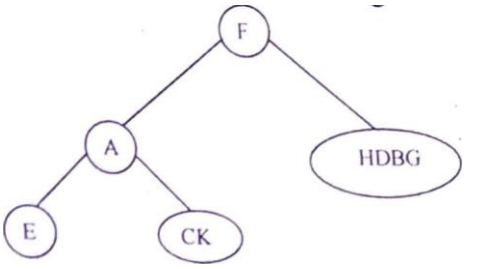
[WBUT 20071

Inorder: E A C K F H D B G
Preorder: F A E K C D H G B
Draw the binary tree. State briefly the logic used to construct the tree.

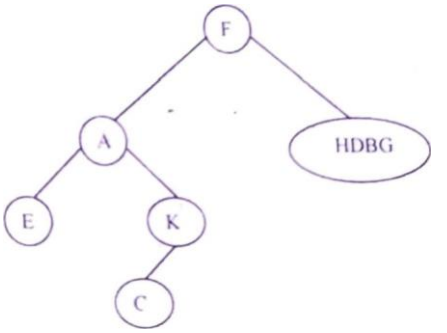
ALGORITHM snswer: from the preorder traversal, we know that the first node is always the root node of the tree. So, F is the root node of the binary tree. Now, from the inorder traversal, we binary have to find the position of F. All the nodes of F are in the left subtree and all the nodes in right will constitute the right subtree. So, the tree looks like:



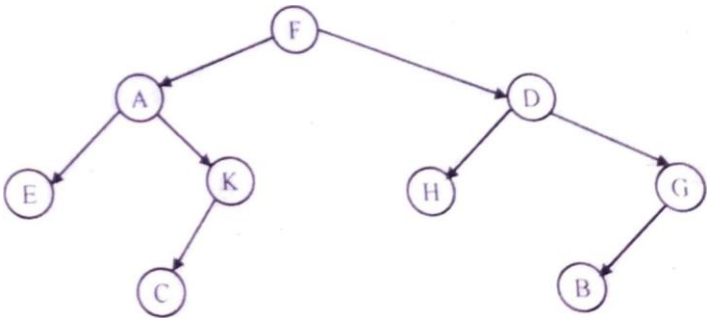
Again from the preorder traversal, next element is A. So, A is the root of the left subtree. Again if we find the position of A in the inorder traversal, left element is E & CK are right elements. So, the left subtree consists only E and right subtree consists CK.



Again for CK, K occurs before C in preorder traversal, so K is the root of right subtree and from the inorder traversal C is at left of K, so, C is the left child of K.



Now, for the right subtree, HDBG of F, the first element in the preorder traversal is D and from the inorder traversal, we say that H is the left child and BG is the right subtree. Again among BGs G is the first element, so G is the root and from the

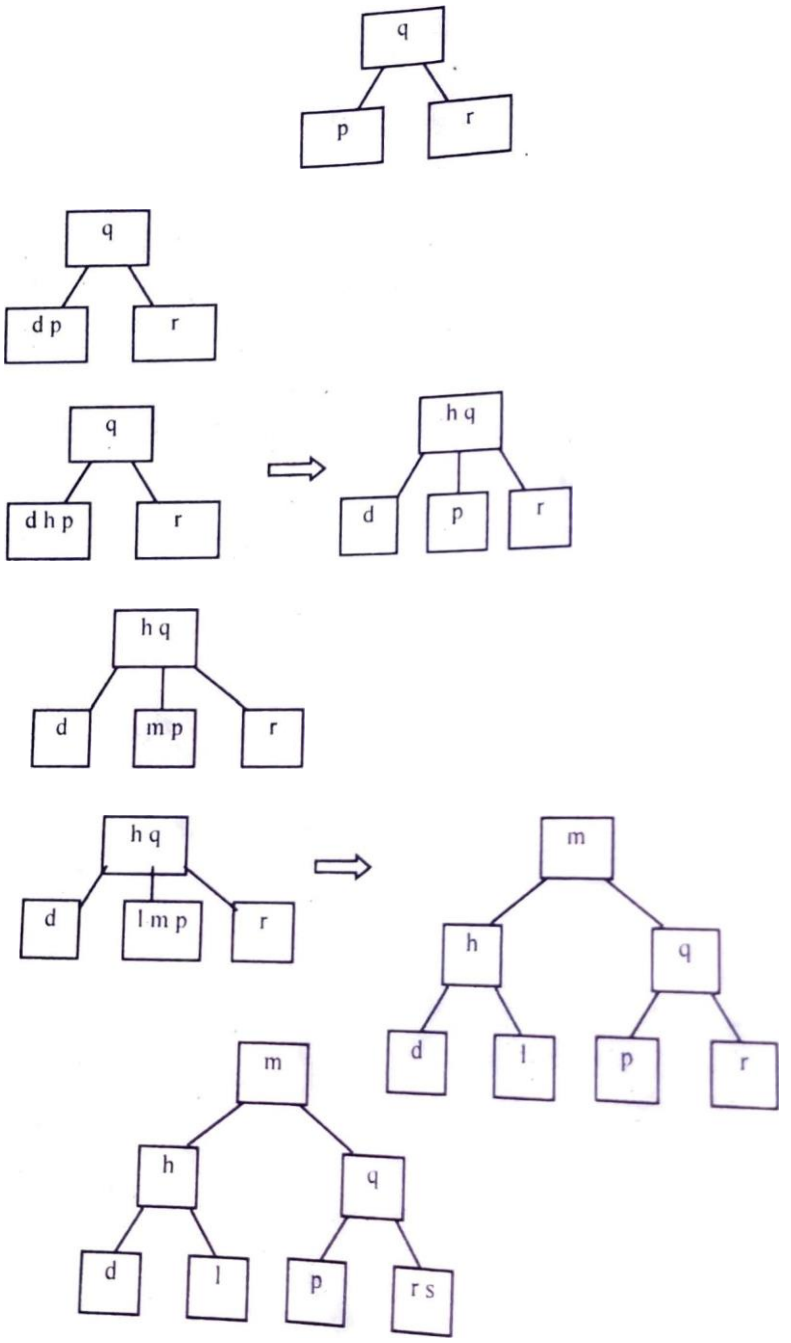


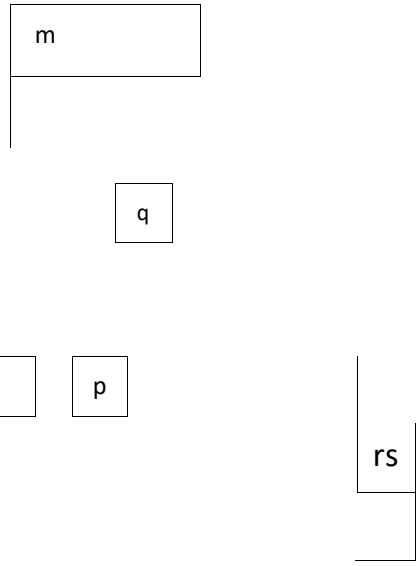
inorder traversal B is the left child. So, the final tree is

b) Insert the following keys into a B-Tree of order 3:
p, q, r, d, h, m, l, s, k, n.

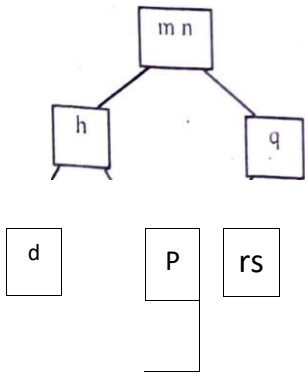
CWBUT 20071

Answer:

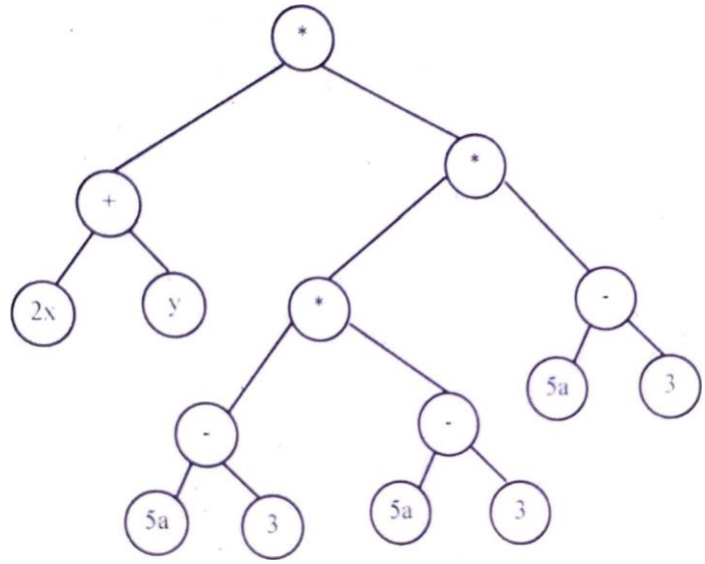




n:



c) construct an expression tree for the expression E = (2x + y) * (5a — .



[WBUT 2007] Answer:

d) Write a non-recursive algorithm for in-order traversal of a binary tree.

[WBUT 2007, 2010, 2016]

Answer:

ASSIllne that a stack ADT exists. The following is the non recursive C function whose input is the root pointer of the binary tree stored as a linked list with left. right and key as the elements.

```
print_inofder (Node *root )

 Node   temp   ;   while
(left  (root)  !  -null)
push    (stack,    left
(root) ) ;
```

```
while (isEmpty (stack) )
    temp = pop (stack) ; print temp->key;
push (stack, right (temp) ) ;
```

be)3. a) the In number a 2-tree, of if vertices E be the that external are not path leaves, length, then P be prove the that internal E = path length and Q P + 2QCWBUT 2008)

Ifa tree contains only its root and no other vertices, then E P = Q = O, and the first case of the above proof is trivially correct. Now let us consider a larger tree, & let v be some vertex that is not a leaf, but for which both the children of the vertex v are leaves. Let k be the number of branches on the path from the root to v.

Now let us delete the two children of v from the 2-tree. Since b is not a leaf but its children are, the number of non-leaf nodes goes down from Q to Q — I. The internal path length I is reduced by the distance to v, that is, to P - k. The distance to each child of v is k + l, so the external path length is reduced from E to E —2 * (k+ I), but v is now a leaf, so its distance, k, must be added, given a new external path length of

$$E - 2 * (k+1) + k = E - k - 2.$$

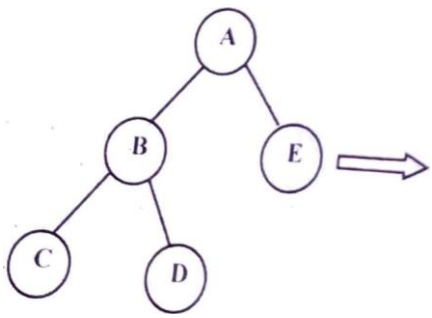
Since the new tree has fewer vertices then the old one; by the induction hypothesis we know that

$$E - k - 2 = (P - k) + 2 * (Q - 1)$$

or, E-k-2=P-k+2q -2 or, E= P+2Q. Proved.

Answer:b) What is threaded binary tree? (WBUT 2008, 2011]

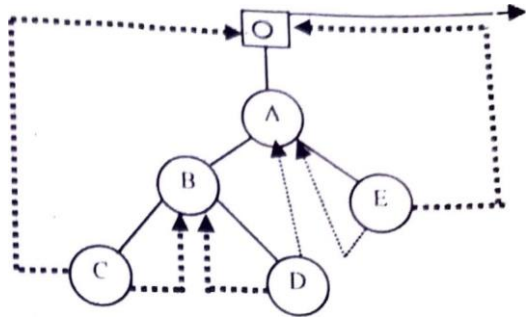
A Tlrreaded Binary Tree is a binary tree in which every node that does not have a right child has a THREAD (in actual sense, a link) to its INORDER successor. By doing this threading we avoid the recursive method of traversing a Tree, which makes use of stacks and consumes a lot of memory and time.



"l he tree has 6 null links
2 each tor the leaf nodes In-order traversal of the tree IS
C B D A E

I consider the tree show above.

imnwdiate predecessor of C will be the head node. So the left child of C. which is a null link, will point to theI lead node head node. (as per in-order traversal)



The right null link of C will point to the inunecliate successor which is B (as per inorder traversal).

- The left null link of D will point to the immediate predecessor which is B (as per in-order traversal).
- The right null link of D will point to the immediate successor which is A (as per inorder traversal).

The above procedure is followed for the node E also and we get the resultant threaded binary tree.

The node structure for a threaded binary tree varies a bit and its like this -struct NODE

```
struct NODE *leftchild; int node_value ; struct NODE *rightchild; struct NODE *thread;
```

c) Write an algorithm to delete a node from a binary search tree. (WBUT 20081
Answer:

In order to delete a node from a binary search tree there may be three possible scenarios:

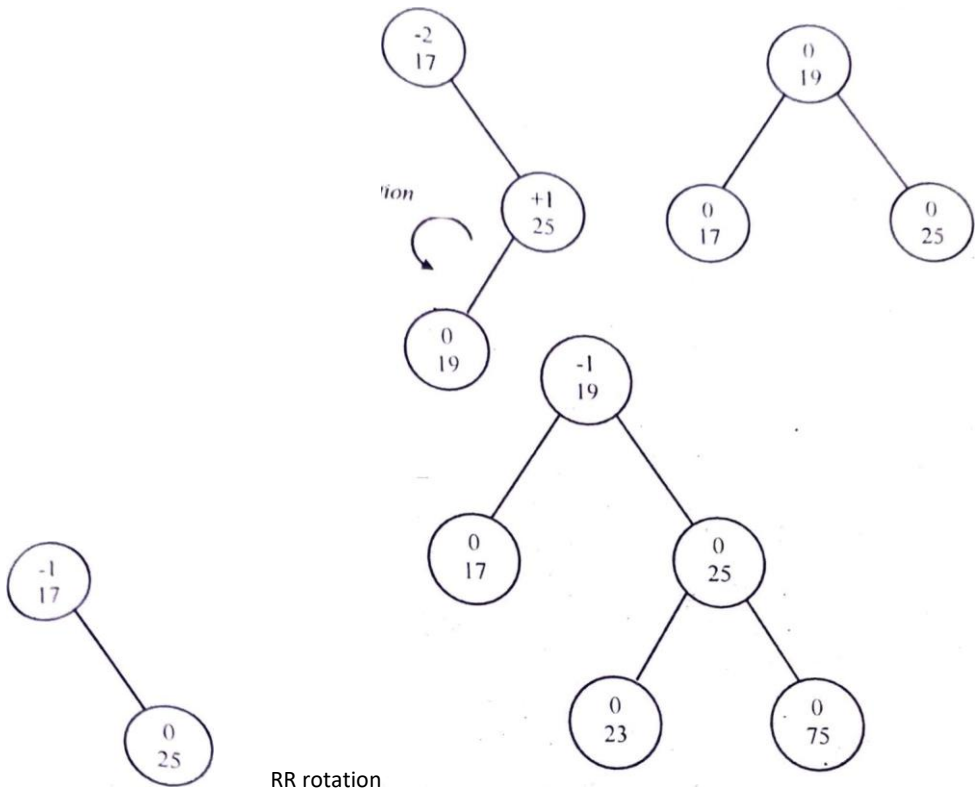
l. Node to be deleted has no children
2. The node to be deleted has only one sub tree
3. The node to be deleted has both left & right subtrees node *delete (node *p, int x)

```
node *q, * rp , *root; q = NULL; root = P; while
(p! = NULL & &x ! = p->data)

    if (x < p->data)
        P       P->lc;
```

```
p =p->rc;

if (P - NULL)

print f ( " key not present " ) ;

exit (0) •

if (p->lc  NULL) / * node p has no left child* ;
  rp = p->rc; else if (p->rc == NULL) 7* p has no
right child* / rp = p->lc;

a = p; / * a contains the parent node p * / rp = p->rc; b
= rp->lc; 7* b is always left child of rp * / while (b !
  NULL) / * to find the inorder successor wi th no
. left child * /

rp = b; b =
rp->lc;

  / * At this point rp is the inorder successor of p f;
if (a ! p)

  a->lc    rp->rc
; . rp->rc = p->rc;

if (q    NULL )
  nroot  rp; else if
(p q->lc)

q->lc  rp; rp->lc =
p->lc;

q->rc  rp; rp->lc =
p->lc;

free (p) ; / : delete the node
return (root) .
```

d) create a AVI- tree by inserting the following numbers in the order in which they are given: 17 25 19 23 75. Draw figure for each step.

Answer: (WBUT 20081



RR rotation

4. a) prove that, the height of a binary tree that contains n elements, n20, is at most n and at least [log (n + 1)].

b) The order of nodes of a binary. tree in Preorder and in order traversal are as under:

In order: D B F E G H I AC

Pre-order: A B D E F G H I C

Draw the corresponding binary tree.

c) How does static allocation differ from dynamic allocation of memory?

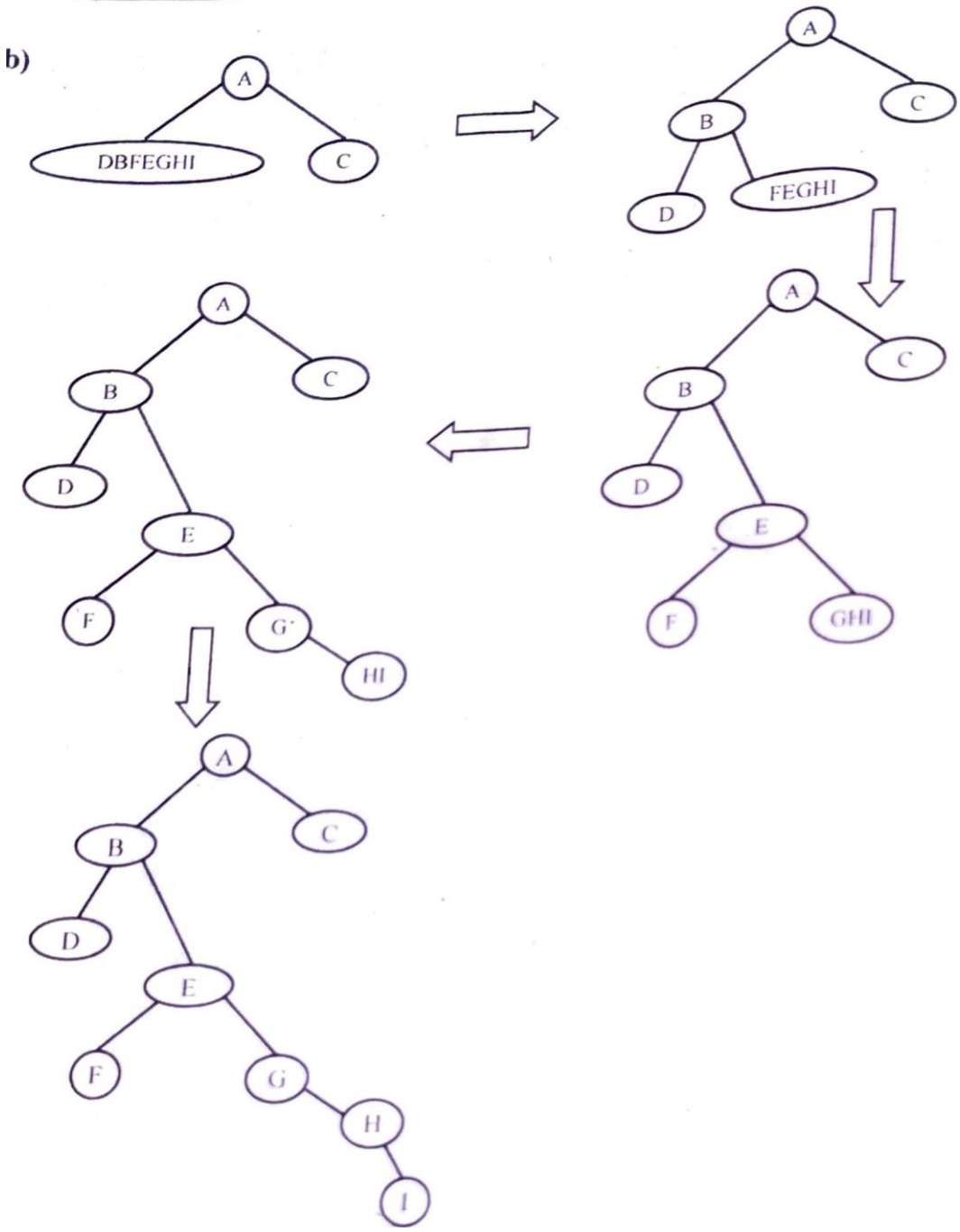
(WBUT 20091

Answer:

a) The worst case is a linear tree with only one leaf, and thus has n + I nodes, that is h S O(n+l). The best base is a binary tree, with 2^k internal nodes at each level, except for the bottom level (they are all leaf nodes). However, the completion of nodes at the bottom level can effect the number of external nodes from $2^{b\,l}$ (the bottom level has

only the left-most pair of nodes) to 2h (a complete binary tree) external nodes, where h is the height of the tree. Thus, we can have this inequality for external nodes for a true with n internal nodes: 2^{11}- < n + I S 211 h- I < lg (n + l) S h. Thus, lg n ShSn+ l .

b)



c) Static allocation is done at the time the program is Mitten. The programrner reserves a set amount of menujry for each use. That setting ssill be used each time the program runs. Dynalnic allocation is done hile the program runs, adjusting thc amount of memory for each use according to how much memory required at that time.

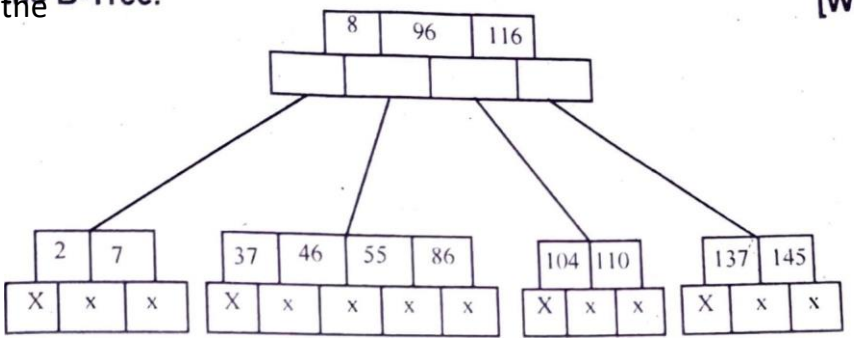
5. a) Show the steps in creation of a height balanced binary AVL TREE using insertion of items in the following order — Show the balancing steps required• (March, May, November, August, April, January, December, July, February, June, October, September)

b) What do you mean by a B-Tree and what are the gses of such a tree in data structures? [WBUT 2009]

OR,

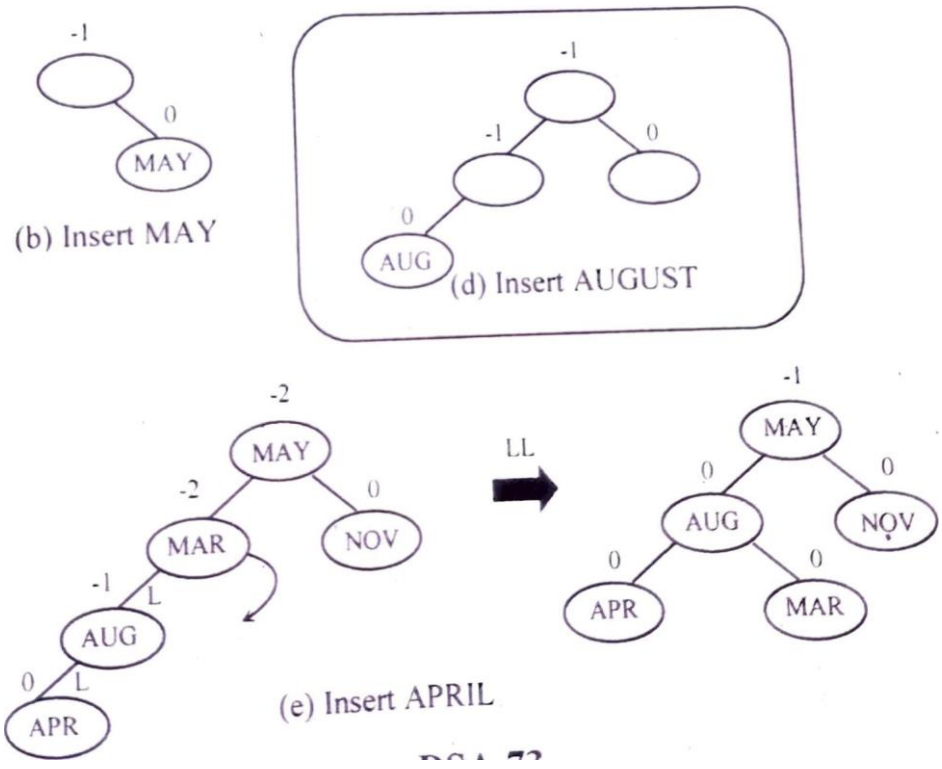
For what purposes are B trees especially appropriate? cWBUT 20161

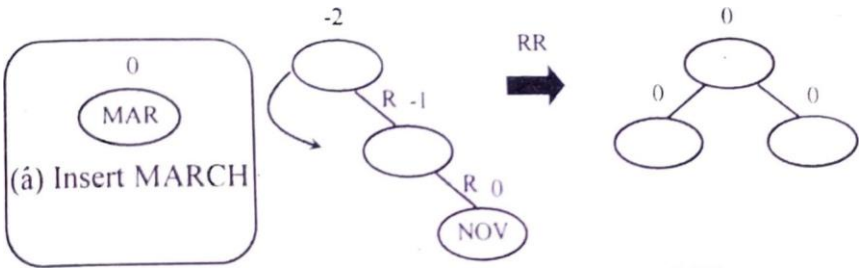
this order inthe B-Tree. [WBUT 2009]



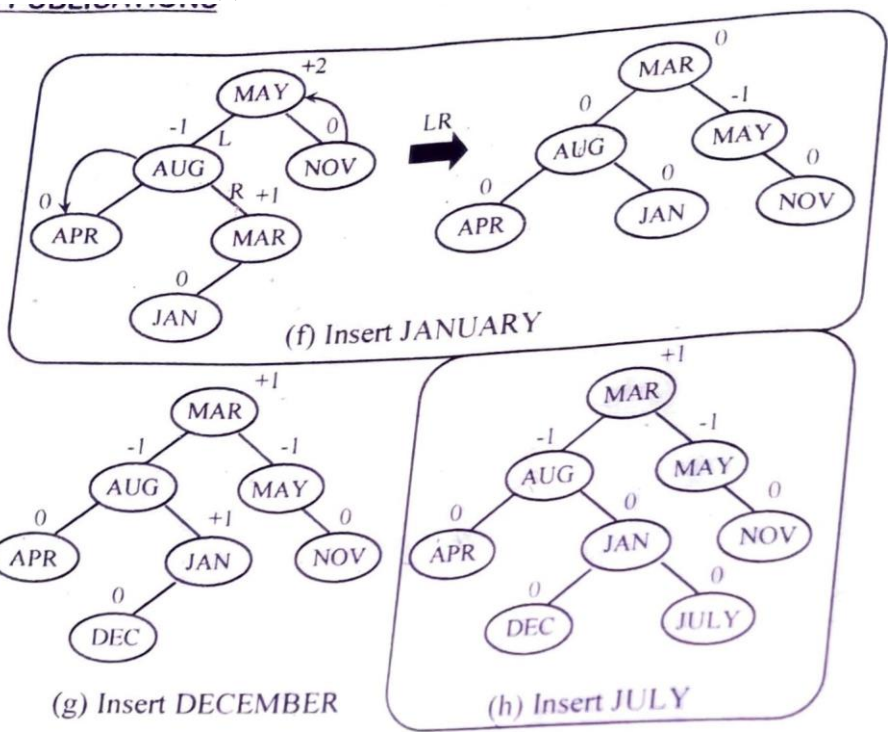
c) consider a B-Tree of order 5 as shown below — insert the elements 4, 5, 58, 6 in

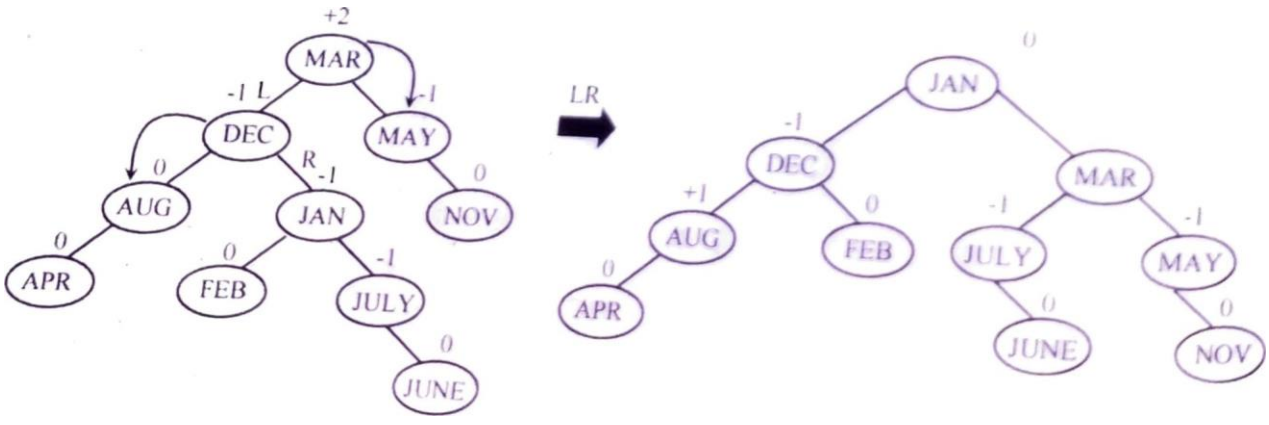
Answer:

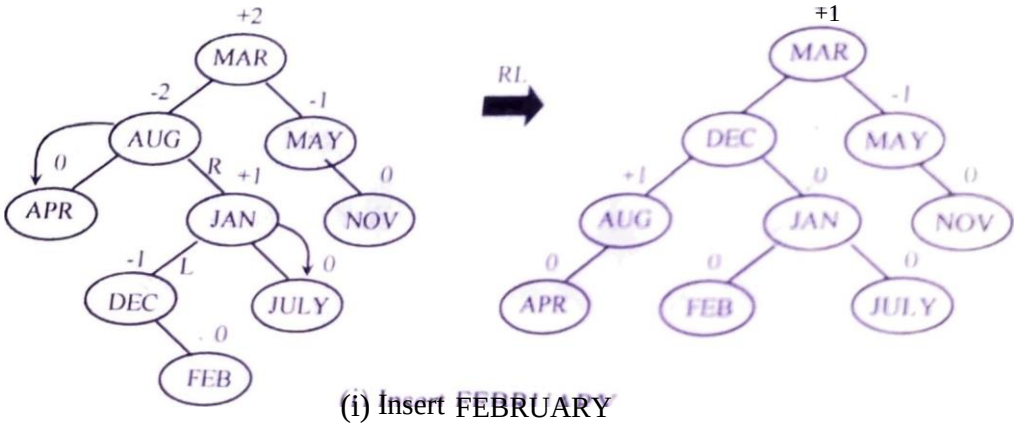




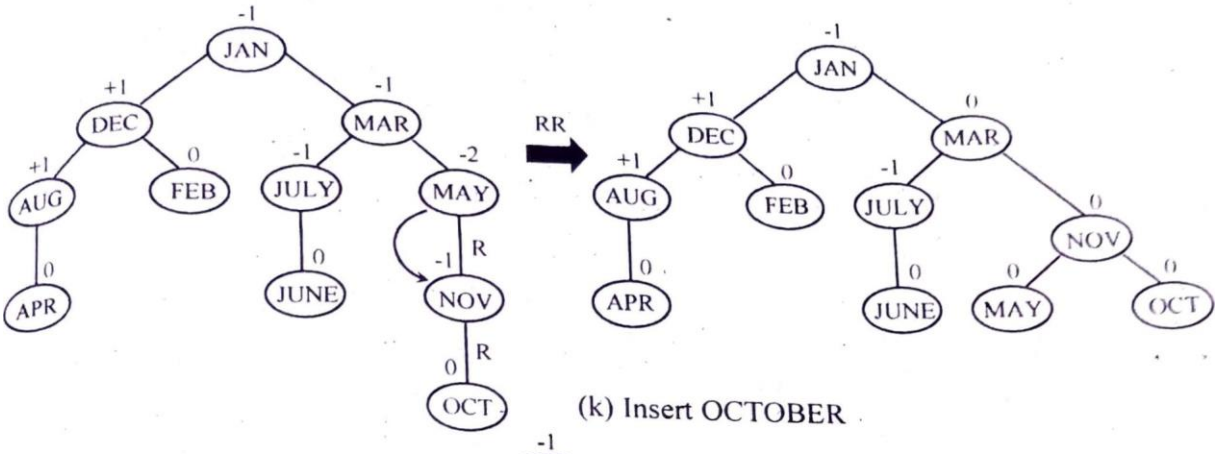
(c) Insert NOVEMBER



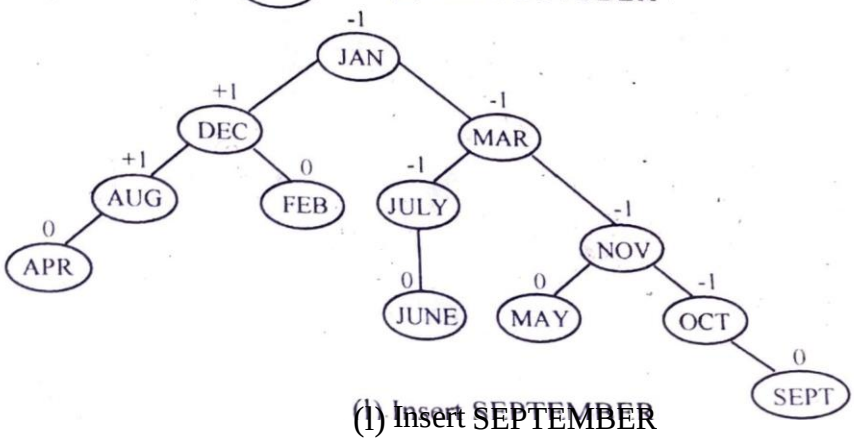




(i) Insert FEBRUARY



(k) Insert OCTOBER



(l) Insert SEPTEMBER

b) B-trees are balanced trees that are optimized for situations when part or the entire tree must be maintained in secondary storage such as a magnetic disk. Since disk accesses are expensive (time consuming) operations, a B-tree tries to minimize the number of disk accesses. Each node of a b-tree may have a variable number of keys and children. The keys are stored in non-decreasing order. Each key has an associated child that is the root of a subtree containing all nodes with keys less than or equal to the key but greater than the preceding key. A node also has an additional rightmost child that is the root for a subtree containing all keys greater than any keys in the node.

Magnetic disks (secondary memory) are cheaper than RAM and have higher capacity. But they are much slower because they have moving parts. B-trees try to read as much information as possible in every disk access operation.

6. a) Show the stages in growth of an order — 4 B-tree when the following keys are inserted in the order given:

74 72 19 87 51 10 35 18 39 60 76 58 19 45 [WBUT 20101 Answer:

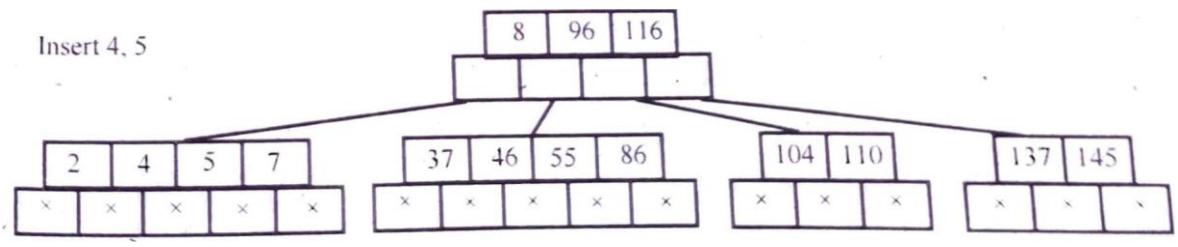
74 72 19 87 51 10 35 18 39 60 76 58 19 45

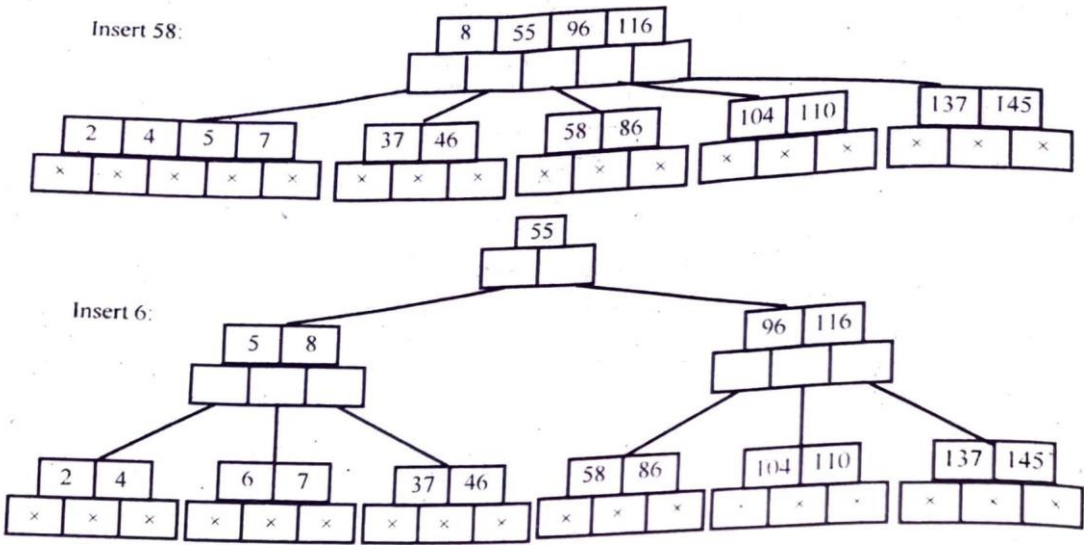
Insert 74: 74

Insert 72: 72 , 74

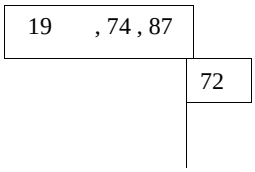
Insert 19: 19.72 , 74

c)

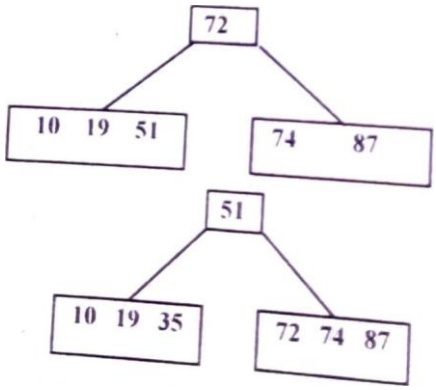


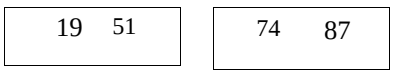


Insert 87:



Insert 5 1 :





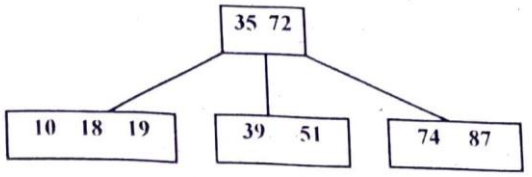
Insert 10:

POPULAR
PUBLICATIONS

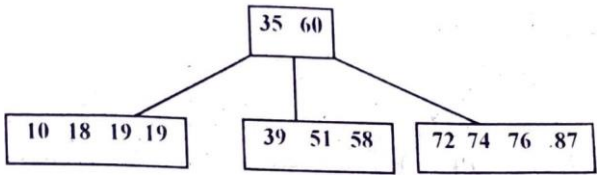
Insert 35:

Insert 1 8: 35 51

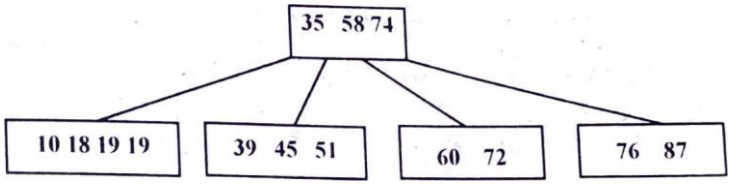
Insert 39:



Insert 19:



Insert 45:



b) How an AVL tree differs from binary search tree? [WBUT 20101

OR,

What are the differences between AVL Tree & Binary Search Tree? . [WBUT 20121

Answer:

A binary search tree is a binary tree that is either empty or in which each node satisfies the following conditions:

i) The left child has a value smaller than the parent ii)
The right child has a value greater than the parent.

Now, an AVL tree is a height-balanced tree. If T is a non-empty binary tree with Tv & TR as its left and right subtrees respectively, then T is height balanced if

i) Tl. & TR are height balanced ii) lhl_ — h R I <= l, where hi. & hR are the heights of TL & TR reSpectively.

Now, an AVL tree is also a binary search tree, but it is a height balanced binary search tree. In case of binary search tree, if all the data are already sorted, then the height will be O(n), where n is the number of elements in tree. So. then the worst case search time will be O(n). But since A VL tree is height balanced, this type of case will never occur. Here we can perform searching in O (Lgn) time. Now, to construct an A VL tree, we have to perform some extra operations to get the tree height balanced.

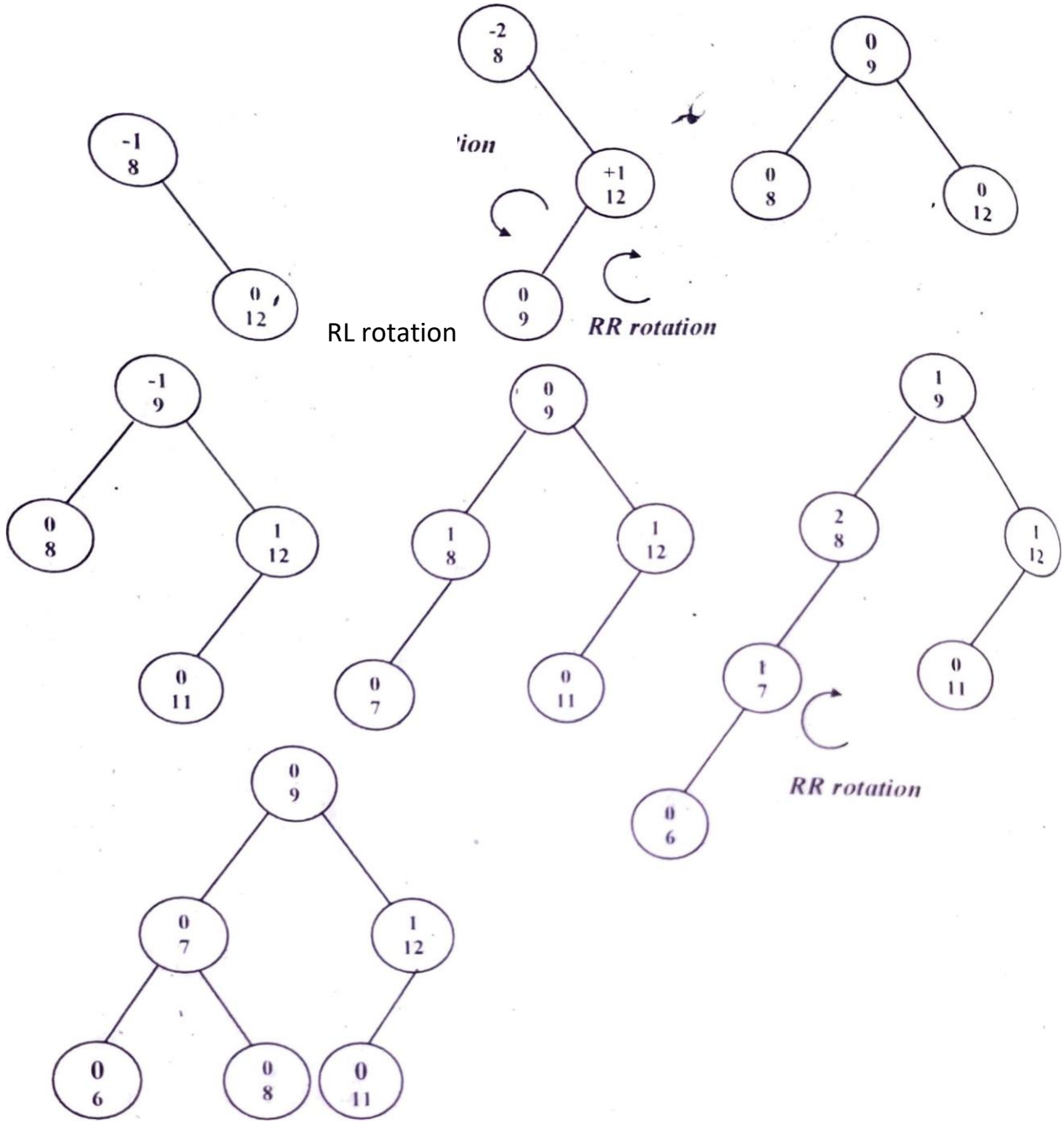
c) Insert the following keys in the order given below to build them into an AVL tree.

8 12 9 11 7 6 .

Clearly mention different rotations used and balance factor of each node.

[WBUT 20101

Answer:



7. a) Given the preorder and inorder sequence and draw the resultant binary tree and write its postorder traversal:

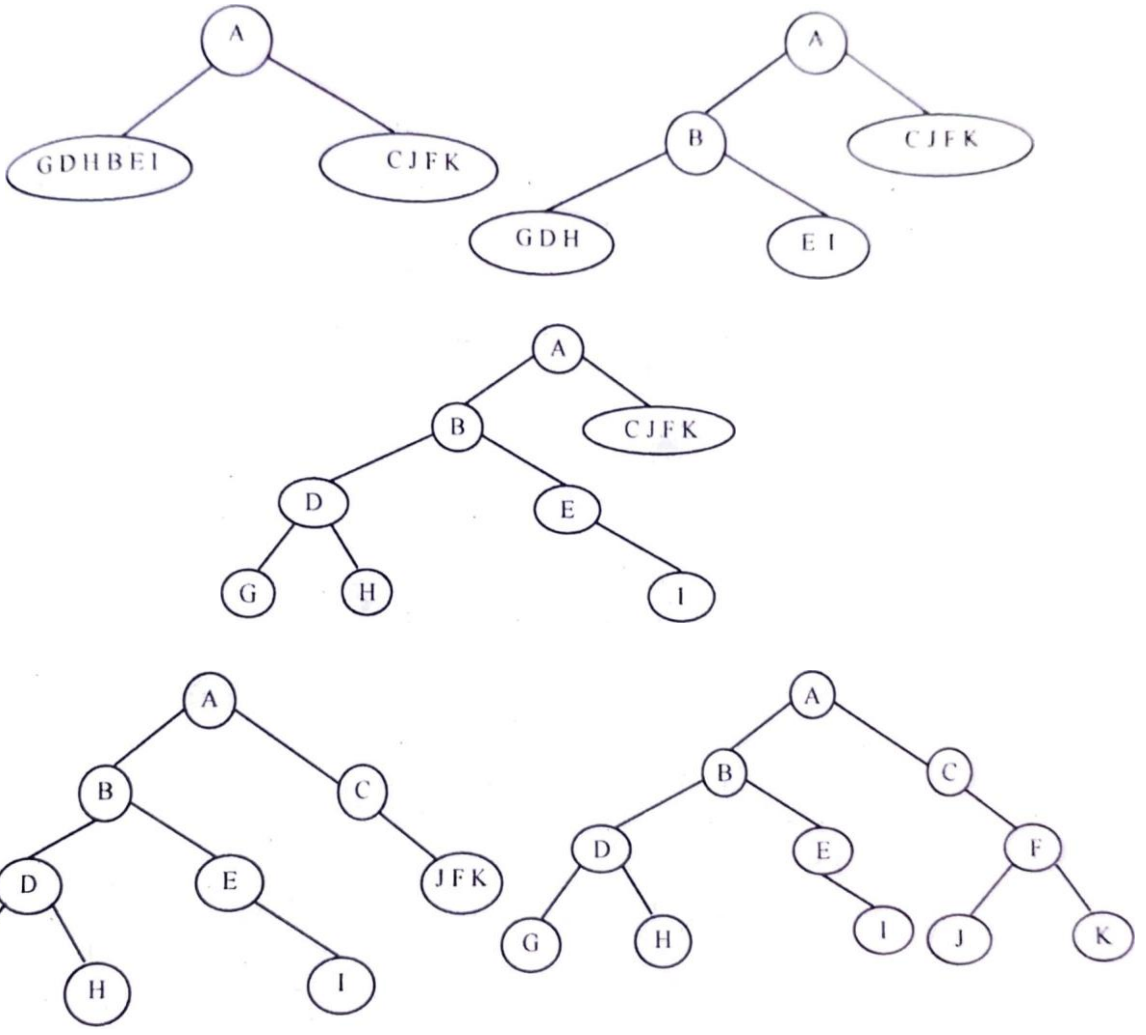
Pre-order: A B D G H E I C F J K In-order: G D H B E I A C J F K

b) Write an algorithm to search a node in a binary search tree. [WBUT 2010)

A no'cr:

B In preorder traversal. the first node is always the root the binary tree. Here it is A.

for the position of' A in the corresponding in-order traversal. In order the root is a pivot around which the left and right f,ubtreeq arc constructed. So in our all nodes lying to the left of A will represent the left subtree. & all other nodes to the right of A will tepresent the right sub-tree. The trve looks like.



The Post Order traversal of the above tree is: G , H , D , I , E , B , J , K , F , C A

b) p is either the root node or the node from where the search has to begin void search (node *p, int digit)

```
if (p  NULL) print f ( " Error! Tree structure not found \ n" ) ; else
if (digit        p->data) print f ('Number=%d\n", digit) ; else if
(digit < p->data)
```

```
                                    -- left child
    search(p->lc, digit) ; //lc right child
    search(p->rc, digit) ; // = rc
```

8. What are the problems of binary tree? Explain the improvement of performance by the use of height-balanced tree.

Explain how a height-balanced tree can be formed by inserting the following elements in the given order:

1, 2, 3, 4, 5, 6, 8, 9, 10, 7, 11 [WBUT 20111

Show the root element the can be deleted from the above tree. Answer:

Most operations on a binary search tree (BST) take time directly proportional to the height of the tree, so it is desirable to keep the height small. Since a binary tree with height h contains at most $2^0+2^1+\cdots+2^h$ = 2h+l—l nodes, it follows that the minimum height of a tree with n nodes is log2(n), rounded down; that is, [log2nJ. However, the simplest algorithms for BST item insertion may yield a tree with height n in rather common situations. For example, when the items are inserted in sorted key order, the tree degenerates into a linked list with n nodes. The difference in performance between the two situations may be enormous: for n = 1,000,000, for example, the minimum height is

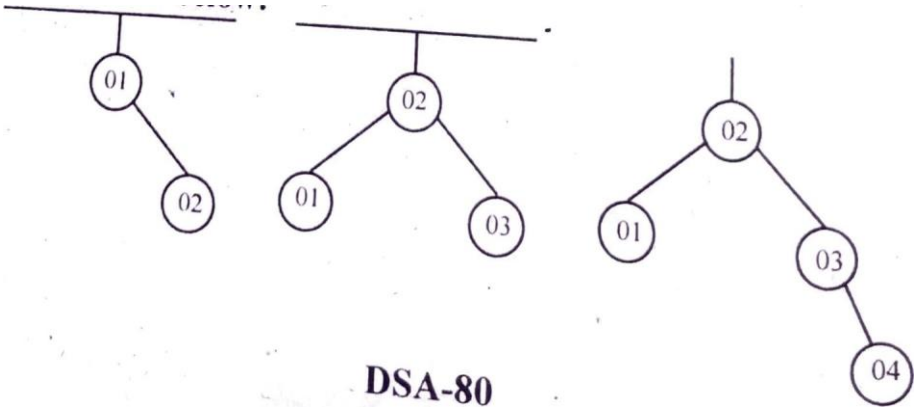
Self-balancing • binary trees or height balanced tree solve• this problem by performing transformations on the tree (such as tree rotations) at key times, in order to keep the height proportional to log2(n). Although a certain overhead is involved, it may be justified in the long run by ensuring fast execution of later operations.

Maintaining the height always at its minimum value [log2(n)] is not always viable; it can be proven that any insertion algorithm which did so would have an excessive overhead. Therefore, most self-balanced BST algorithms keep the height within a constant factor of this lower bound. In the asymptotic ("Big-O") sense, a self-balancing BST structure containing n items allows the lookup, insertion, and removal of an item in O(log n) worst-case time, and ordered enumeration of all items in O(n) time. For some implementations these are per-operation time bounds, while for others they are amortized bounds over a sequence of

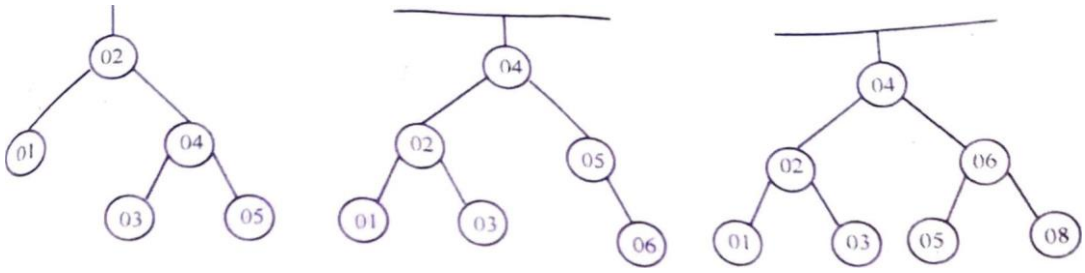
operations. These times are asymptotically optimal among all data structures that manipulate the key only through comparisons.

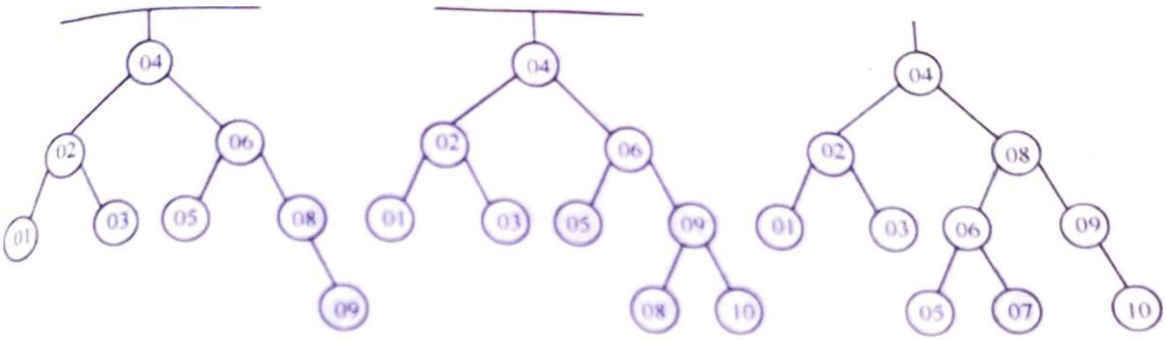
2 nd part:

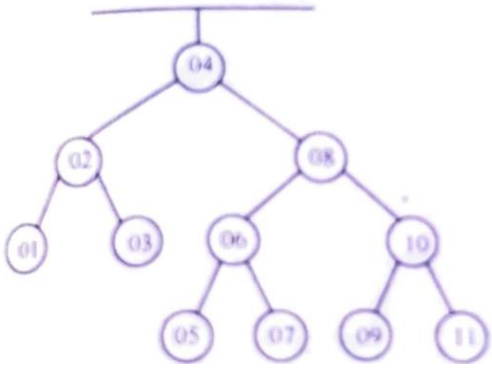
The steps are as shown below:



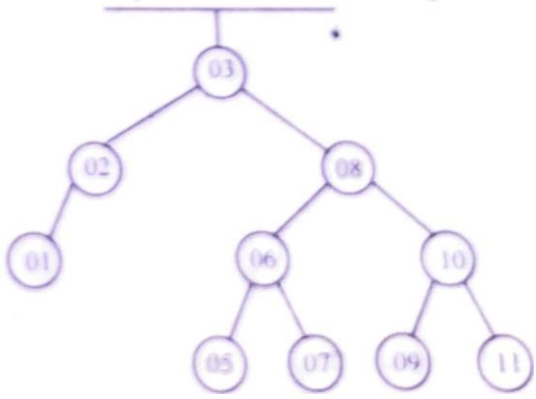
DSA-80





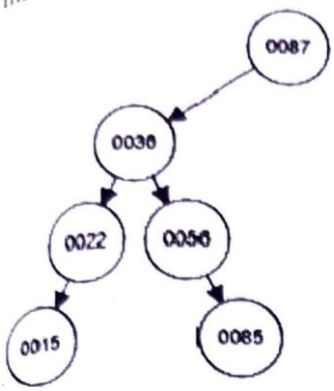


The root element 4 when deleted will produce the following foliose tree:

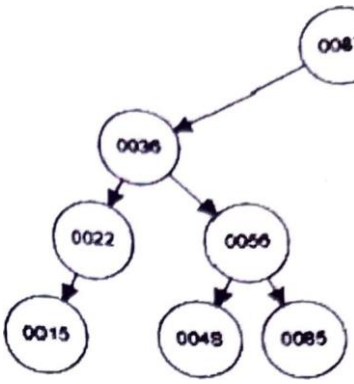


9. What is a B-tree? Show how the letters A to P of English alphabet can be entered into a B-tree of order 4 (WBUT 20111

Insert 56. 85

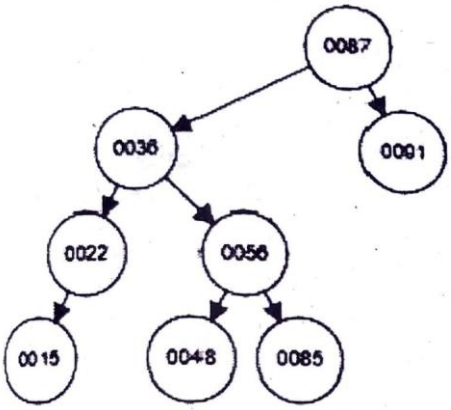


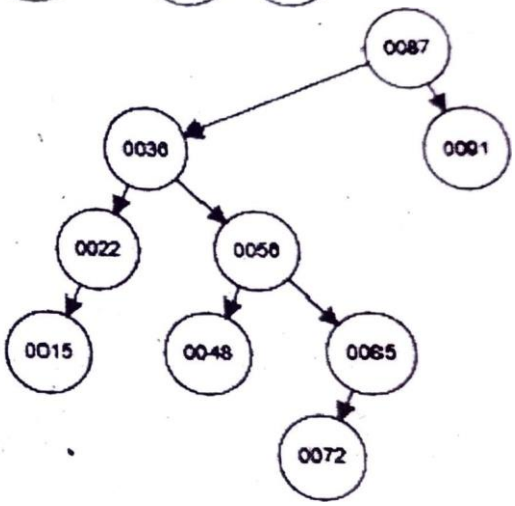
Inscrt 48



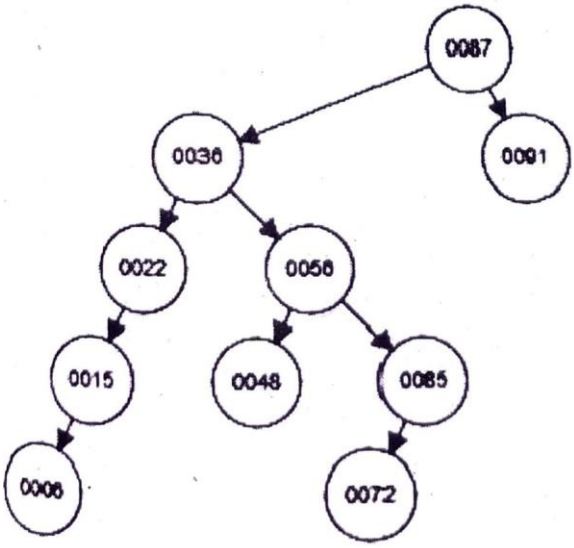
Insett 91

Insert into 72

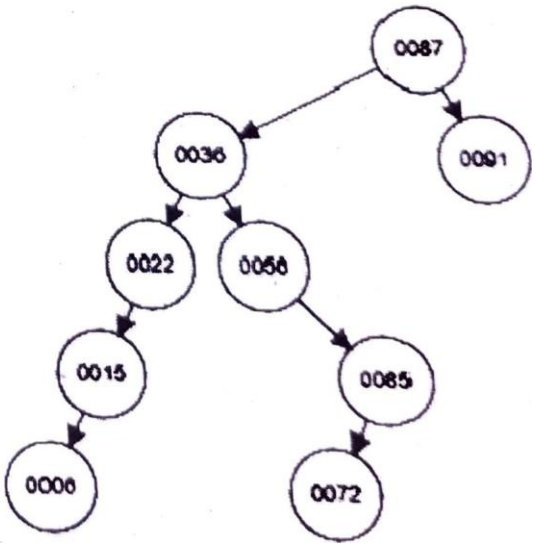




Insert 6

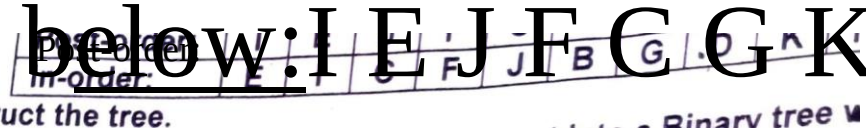


Delete 48



DSA-83

12. a) The in-order and pre-order traversal sequence of nodes in a binary tree given below: I E J F C G K L H D B A

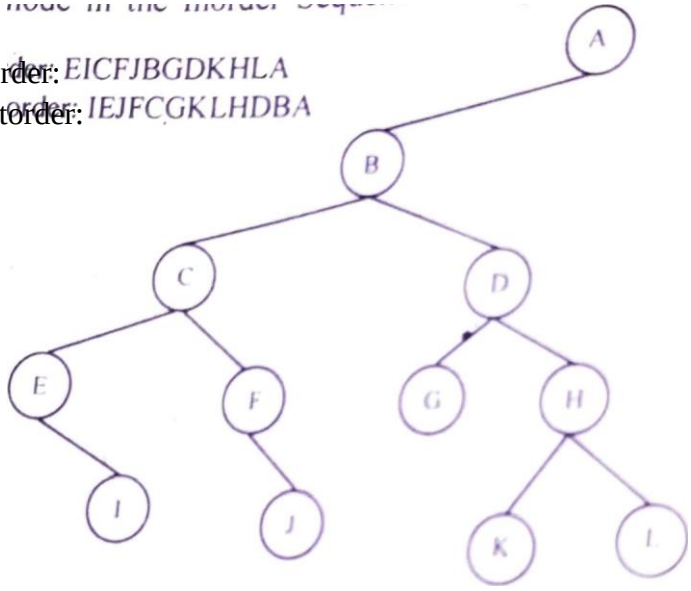


b) Write an algorithm for inserting an element into a Binary tree with example.[WBUT 201

Answer:

a) The last node in the postorder sequence is the root. We traverse right to left in th postorder sequence, finding each node's position (left or right) with respect to previously located node in the Inorder Sequence. Using this we construct the tree shown below.

inorder: EICFJBGDKHLA
postorder: IEJFCGKLHDBA



b) The following C prograjn does the insertion into a binary tree /* Structure */ struct bin_tree { int data; struct bin_tree * right, * left;

typedefstruct bin_tree node;
/*insert function*/
I void insett(node **

```
2 node *temp = NULL;tree, int , {
 if(!(*tree)) { temp = (node
 *)malloc(sizeof(node)):
 temp->letl =temp->right = NULL; temp->data =
 Val;
*tree = temp; return•
```

```
Il    if(val    <    (    *tree)->data)    {
      insert(&(*tree)->left, val);
12
13                              } else if(val > ( *tree)->data) {
14                              val);
15                              }
16                              }
```

Explanation:

[Lines [Line , I 3-91 l] a. Check [Line Check 121 if first node Call if value insert() tree is to empty, function be inserted then recursively insert is lesser node than while a} root root.there node is non-NULL value, then left node

- b. [Lines 3-91 When reached to leftmost node as NULL, insert new node.

[Line 13] Check if node value to be inserted is greater than root node value, then

, a. [Line 14] Call insert() function recursively while there is non-NULL right node

- b. [Lines 3-9] When reached to rightmost node as NULL, insert new node.

13. a) Write a C function to find out the maximum and the minimum elements in a binary search tree.
b) Given the pre-order sequence and the post-order sequence, why cannot you reconstruct the tree? [WBUT 2016] Answer:

a) Assume the binary search tree nodes are defined as follows: struct node

```
struct node *Ichild; int info;
struct node *rchild;
```

```
/*Min function -recursive */ struct node *Min
(struct node *ptr)
```

if ptr==NULL) return NULL ; else
if tr->lchild==NULL)return
ptr; else return Mio (ptr->lchild) ;
l / *End of min( ) */

Max function -recursive */struct node *Max
(struct node *ptr)


```
else                    if
(ptr->rchild==NULL)
return ptr; else return
Max (ptr->rchild) ; ) /
*End of max() */
```

b) It is not possible to construct a binary tree just from pre and post order traversals of a binary tree because only inorder traversals give the position of the sub-tree elements with Ireese If one ebetcausheethe has inorder pre/post along with order pre information or post order gives one you can the always root of construct the tree. Once a unique the root binaryof

the tree is known one can recursively construct the left and right sub-trees which are binary trees themselves thereby making the final product unique.

14. a) What do you mean by a binary search tree?
b) Construct a binary search tree by inserting the list of elements one by one:

13, 10, 3, 5, 18, 15, 14

c) Write an algorithm for pre-order traversal of a tree represented by a linked-list.

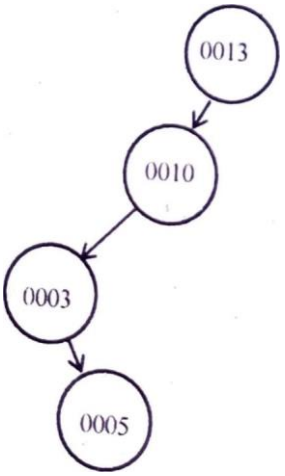
[WBUT 20171 Answer:

a) A binary search tree is a binary tree that is either empty or in which each node satisfies the following conditions:

i) The left child has a value smaller than the parent

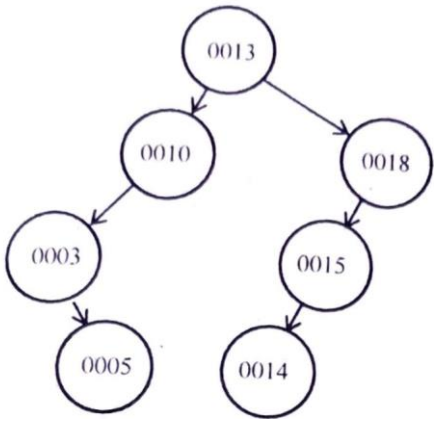
ii) The right child has a value greater than the parent.

b) After inserting 13, 10, 3, 5 the tree looks like



DATA

after inserting 18, 15, 14 the tree looks like



c)//C program implementation

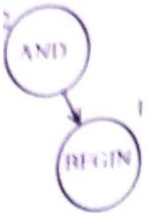
```
/*  A binary tree node has data, pointer to left
child and a pointer to right child * / struct node

    int  data;  struct
    node* left; struct
    node* right;


/*Given a binary tree, print its nodes in preorder*
/ void printPreorder (struct node* node) if (node NULL
) return;
   / * first print data of node * /
    print f ( " %d node->data) ;

   /  *  then  recur  on  left  subtree  *  /
   printPreorder (node->left) ;
   /  *  now  recur  on  right  subtree  *  /
   printPreorder (node->right) ;
```

a) What is an AVL tree? 2017] Construct an AVL search tree for the data list:
AND, BEGIN, CASE, DO, END, FOR GOTO.

For the AVL tree you have constructed delete the following keys in the order
Do, FOR END. ;wer:

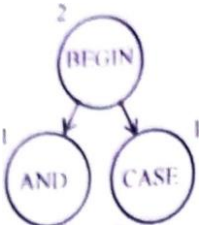
efer to Question No. 16(a) ofLong Answer Type Questions.

inectong AND and BEGIN:

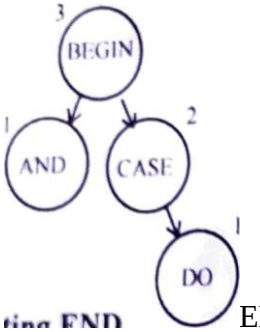


b) After

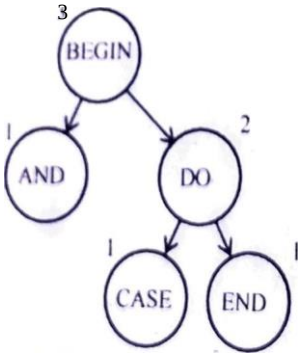
After inscHing CASE



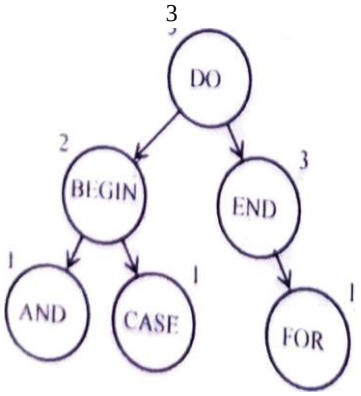
Attcr in«crting DO



After inserting ting END END

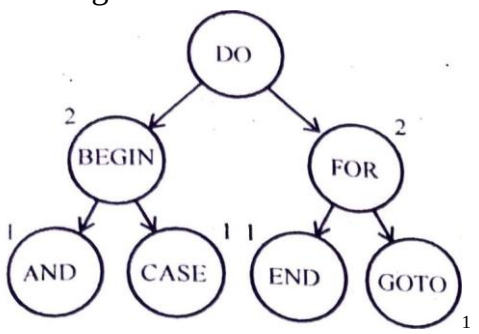


After inserting FOR

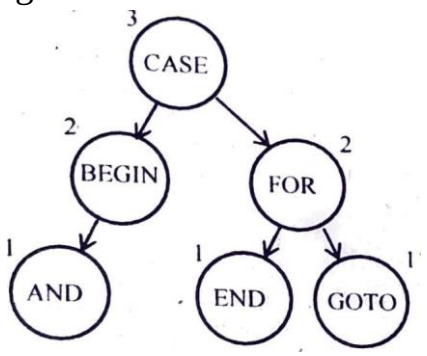


1)SA-92

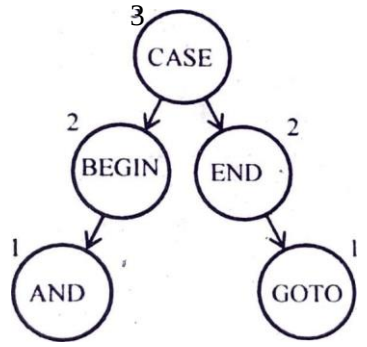
After inserting GOTO



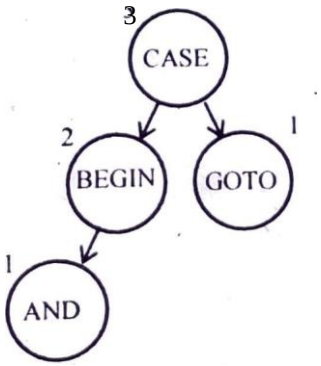
c) After removing DO



After removing FOR



After removing END


3
CASE
2
BEGIN
1
GOTO
1
AND

16. Write short notes on the following:

a) AVL Tree CWBUT 2013, 2015, 20181

b) Threaded Binary Tree [WBUT 2010, 2014, 2015)

c) B — tree [WBUT 2012, 20131

d) Binary Search Tree [WBUT 20181 Answer:

a) AVL Tree:

An A VL Tree is a form of binary tree, however unlike a binary tree. the worst case scenario for a search is O (log n). The A VL data structure achieves this property by placing restrictions on the diftu•ence in height between the sub-trees of a given node, and re-balancing the tree if it violates these restrictions.

The complexity of an A VL Tree comes from the balance requirements it enförces on each node. A node is only allowed to possess one of three possible states (or balance factors):

Left-High (balance factor -1)

The left-sub tree is one level taller than the right-sub tree

Balanced (balance factor 0)

The left and right sub-trees are both the same heights Right-

High (balance factor +1)

The right sub-tree is one level taller than the left-sub tree.

If the balance of a node becomes -2 (it was left high and a level was lost frorn the left sub-tree) or +2 (it was right high and a level was lost from the right sub-tree) it will require re-balancing. This is achieved by performing a rotation about this node.

b) Threaded Binary Tree:

Refer to Question No. 3(b) ofLong Answer Type Questions.

c) B — tree:

Definition: A B-Tree of order m is an m-way tree in which

i) All leaves are on the same level.

ii) All internal nodes except the root have at most m nonempty children, and at least rm/2] nonempty children.

iii) The number of keys in each internal node is one less than the nutnber of its nonempty children, and these keys partition the keys in the children in the fashion of a search tree.

iv) The root has at most m children, but may have as few as 2 if it is not a leaf, or none if the tree consists of the root alone.

There are some disadvantages also

- The nodes of a B-tree do not normally contain the same number of value entries.
- They normally do contain a certain amount of free space.
- Difficulty in traversing the keys sequentially.

Note: B-tree is not a binary tree. A B-tree is also known as the balanced sort tree.

d) Binary Search Tree:

Refer to Question No. 14(a) & (b) of Long Answer Type Questions.

GRAPHS

-chapter at a Glance

graph is a tnathetnatical tool used to represent a physical problem. It is als,o used to model networks, data structures, scheduling, computation and a variety of other systems „hct.e the relationship between the objects in the system plays a dominant role. O,pcs of graph: A graph often synnbolized as G can be of two types:

l. t n dircctcd graph: where a pair of vertices representing an edge is unordered.

2. Directed graph: where a pair of vertices representing an edge is ordered.

Graph Traversal: A graph can be traversed in two ways:

pepth first search traversal: often analogous to pre-order traversal of an ordered tree.

Breadth first search traversal: often analogous to level-by-level traversal of an ordered tree.

spanning Tree: Given a connected, undirected graph, a spanning tree of that graph is a sub graph which is a tree and connects all the vertices together. A single graph can have many different spanning trees. We can also assign a weight to each edge, which is a number representing how unfavourable it is, and use this to assign a weight to a spanning tree by computing the sum of the weights of the edges in that spanning tree.

shortest Path: In graph theory, the shortest path problem is the problem of finding a path between two vertices in a weighted graph such that the sum of the weights of its constituent edges is minimized.

The most important algorithms for solving this problem are:

Dijkstra's algorithm solves the single-source shortest path problems.

Bellman-Ford algorithm solves the single source problem if edge weights may be negative.

A* search algorithm solves for single pair shortest path using heuristics to try to speed up the search.

Multiple Choice Type uestions

l. The vertex, removal of which makes a graph disconnected is called PNBUT 20071

a) Pendant vertex b) bridge

c) articulation point d) colored vertex

Answer: (c)

2. A vertex of in-degree zero in a directed graph is called PNBUT 2007, 20181

a) articulation point b) sink

c) isolated vertex d) root vertex

Answer: (c)

3. Adjacency matrix of a digraph is [WBUT 2007, 2012, 20161

a) identityb) symmetric

c) asymmetric d) none of these

Answer: (b)

4. Which data structure is used for breadth first traversal of a graph? PNBIJT 20081

a) Stack b) Queue c) Both stack and queue d) None of these

Answer: (b)

5. The adjacency matrix of an undirected graph is 2008, 20101

a) Unit matrix b) Asymmetric matrix

c) Symmetric matrix d) None of these

Answer: (c)

6. BFS [WBUT 20081

a) scans all incident edges before moving to the other vertex
b) scans adjacent unvisited vertex as soon as possible
c) is same as backtracking
d) none of these

Answer: (b)

7. A non-planar graph with minimum number of vertices has PNBUT 2008, 2018)

a) 9 edges, 6 vertices b) 6 edges, 4 vertices

c) 10 edges, 5 vertices d) 9 edges, 5 vertices

Answer: (c)

8. Any connected graph with x vertices must have at least [WBUT 20091

a) x+ 1 edges b) X-I edges c) x edges d) x12 edges

Answer: (b)

9. Maximum number of edges in a n-node undirected graph without self loop is [WBUT 20101

b) $\frac{\quad}{2}$ d) $\frac{(n+1)(n)}{2}$

Answer: (b)

10. BFS constructs [WBUT 2010, 2014, 20181

a) a minimal cost spanning tree of a graph
b) a depth first spanning tree of a graph
c) a breath first spanning tree of a graph
d) none of these

Answer: (a)

11. A complete directed graph of 5 nodes .. has ..... LWBUT 2011]

b) 10 c) 20 d) 25

Answer: (b)

12. A vertex with degree one in a graph is called PUBUT 20121

a) leaf b) pendant vertex c) end vertex d) none of these

Answer: (b)

TO implement OFS which data structure is generally used? tWBlJT 2013)

13,) stack b) Quouo c) noth (a) & (b) d) None of these

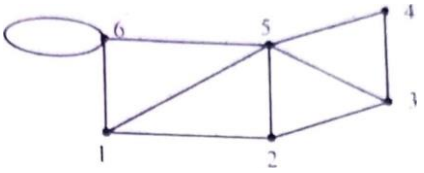
tWBUT 20141

14. Adjacency matrix for a digraph is _

a) unit matrix b) symmetric matrix

c) asymmetric matrix d) none of these \ øs"cr: (b)

5. What is the sum of the degrees of all the vertices in the following graph?



CWBUT 20171

a) 19

b) 20

d) none of these xns»er: (a)

16. The adjacency matrix of an undirected graph is 20171

a) Unit matrix b) Asymmetric matrix

c) Symmetric matrix d) none of these

.Ånswer: (c)

17. A path is CWBUT 20171

a) a closed walk with no vertex repetition

b) an open walk with no vertex repetition

c) an open walk with no edge repetition

d) a closed walk with no edge repetition

Ans"er: (c)

Short Answer e uestions

1. Describe Kruskal's minimal spanning tree algorithm. [WBUT 2008, 20181

Answer:

Kruskal's algorithm is an algorithm in graph theory that finds a minimum spanning tree for a connected weighted graph. This means it finds a subset of the edgeq that forms a tree that includes every vertex, where the total' weight of all the edges in the tree is minimized. If the graph is not connected, then it finds a minimum spanning forest (a

minimum spanning tree for each connected component). Kruskal's algorithm is an example of a greedy algorithnl.

It works as follows:

- create a forest F (a set of trees), where each vertex in the graph is a separate tree
- create a set S containing all the edges in the graph while S" is nonempty remove
- an edge with minimum weight from S
-

1)SA-97

if that edge connects two different trees, then add it to the forest, combining two trees into a single tree

Otherwise discard that edge.

At the termination of the algorithm, the forest has only one component and forms a Illinirnum spanning tree of the graph.

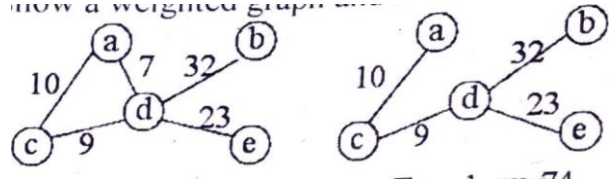
2. What is a minimum spanning tree? Describe Huffman's Algorithm. WBUT 20121

Ans»cr:

Part:

A Minimum Spanning Tree in an undirected connected weighted ra is a spanning tree

unique. if the weights of all the edges are pairwise distinct, it is indeed unique. The following figures show a weighted graph and its minimum spanning tree.



weighted graph Tree l . w=74

2 nd Part:

Huffman's algorithm is a method for building an extended binary tree with a minimum weighted path length from a set of given weights. Initially construct a forest of singleton trees, one associated with each weight. If there are at least two trees, choose the two trees with the least weight associated with their roots and replace them with a new tree, constructed by creating a root node whose weight is the sum of the weights of the roots of the two trees removed, and setting the two trees just removed as this new node's children.

This process is repeated until the forest consists of one tree.

Pseudocode

Huffman(W, n)

Input: A list W of n (positive) weights.

Output: An extended binary tree T with weights taken from W

that gives the minimum weighted path length. Procedure:

Create list F from singleton trees formed from elements of W

WHILE (F has more than one element) DO

Find T l, T2 in F that have minimum values associated with their roots

Construct new tree T by creating a new node and setting Tl and T2 as its children

Let the sum of the values associated with the roots of Tl and T2 be associated with the root of T

Add T to F

End DO
Huffman := tree stored in F

3. What is the adjacency matrix representation of a graph? [WBUT 2013]

Answer:

The adjacency matrix of a graph is

	1	2	3	4	5	6	7
1	0	1	1	0	0	1	0
2	0	0	0	1	0	0	0
3	0	0	0	0	0	0	1
4	0	0	0	0	1	1	0
5	0	0	0	0	0	0	0
6	0	1	0	0	1	0	0
7	0	0	0	0	0	1	0

Write the Prim's algorithm for finding MST from a graph. CWBt.JT 2014, 20161

Answer:

Prim's algorithlll is a Greed\ algorithm. It starts with an empty spanning tree. The idea is maintain two sets of vertices. The first set contains the vertices already included in the MST. the other set contains the vertices not yet included. At every step, it considers all edges that connect the two sets, and picks the minimum weight edge from these edges. After picking the edge, it nnoves the other endpoint of the edge to the set containing MST. Alaorithtn

l) Create a set mstSet that keeps track of vertices already included in MST.

2) Assign a key value to all vertices in the input graph. Initialize all key values as INFINITE. Assign key value as 0 for the first vertex so that it is picked first.

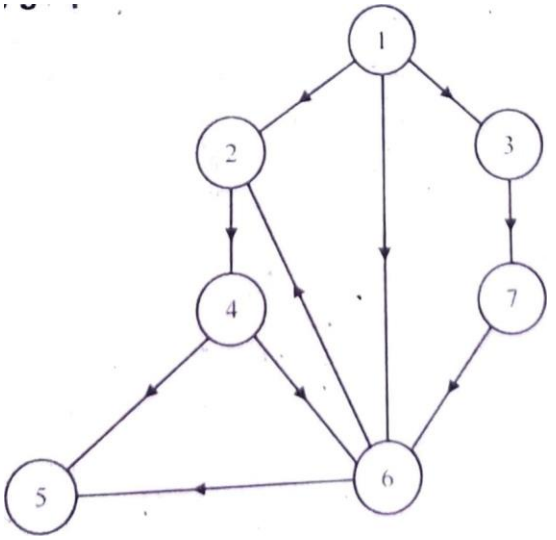
3) While mstSet doesn't include all vertices

.a) Pick a vertex u which is not there in mstSet and has minimum key value.

b) Include u to mstSet.

.c) Update key value of all adjacent vertices of u. To update the key values, iterate through all adjacent vertices. For every adjacent vertex v, if weight of edge "-v is less than the previous key value of v, update the k'ey value as weight of u-v

5. For the following graph find the BFS and DFS traversal with proper algorithm.



[WBUT 20151

1)SA-99

Answer:

a) (i) The BFS traversal of the ra h is:

Event	ueue (Front to Rear)	BFS
Visit I		1
Visit 2	2	12
Visit 3	23	123
Visit 6	236	1236
Remove 2	36	12364
Visit 4	364	
Remove 3		123647
Visit 7	647	
Remove 6	47	1236475
Visit 5	475	
Remove 4	75	
Remove 7	5	
Remove 5		
Done		

Required BPS is 1 2 3 6 4 7.5

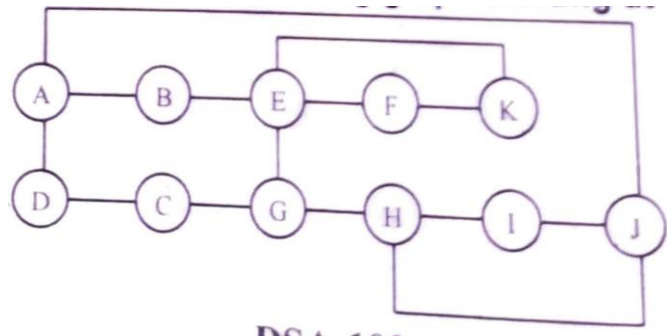
(ii) The DFS traversal of the ra h is:

Event	Stack	DES
Visit I		
Visit 2	12	12
Visit 4	124	124
Visit 5	1245	1245
PO	124	
Pop 4	12	
isit 6	126	12456
	12	
isit 3	3	124563
Visit 7	37	1245637
Remove 7	3	
Remove 3		
Done		

Required DES is 1 2 4 5 6 3 7.

2

6. Find out the DFS traversal of the following graph starting at node A.



DSA-100

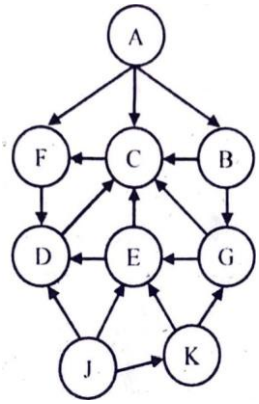
Answer:
DFS traversal we employ the following rules using stack for Rule I Visit the adjacent unvisited vertex. Mark it as visited. Display it. push it in a stack.

- Rule 2 — If no adjacent vertex is fi)und, pop up a vertex from the stack. (It will pop up all the vertices fronl the stack, which do not have adjacent vertices.)
- Rule 3 Repeat Rule I and Rule 2 until the stack is empty.

the above rules we get can get the following traversal when starting with A:

using
ABEGH IJFKCD

7, Apply BFS/DFS Algorithms and find out the path of the given graph:[WBUT 2018]



Answer:
The steps involved in breadth first traversal are as follows: Search

Steps

Step 1 -> Node
A I step 2 Node B
I step 3 Node C I
step 4 Node F
I Step 5 -> Node G
I step 6 Node D
I step 7 Node E

The steps involved in depth first traversal are as follows:

I Step I -> Node A I
Step 2 -> Node B

I Step 3 -> Node
C
I Step 4 -> Node F
step 5 Node D I
Step 6 -> Node G
I step 7 Node E

algorithm. [MODEL QUESTION)

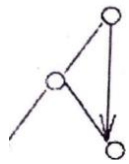
8. Describe the Edge classification of DFS Answer:

Consider a directed graph G = (V, E). After a DFS of graph G we can put each edge into one of four classes:

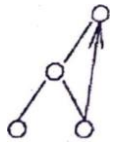
1. A tree edge is an edge in a DES-tree.
2. A back edge connects a vertex to an ancestor in a OFS-tree. Note that a self-loop is a back edge.
3. Aforward edge is a non-tree edge that connects a vertex to a descendent in a DFs_ tree.
4. A cross edge is any other edge in graph G. It connects vertices in two different tree or two vertices in the same DES-tree neither of which is the ancestor of the other.

l. A Tree Edgeto a descendant

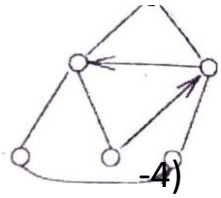
3. A Forward Edge



o



4 A Cross Edge to a



2. A Back Edgedifferent node O to an ancestor -4)

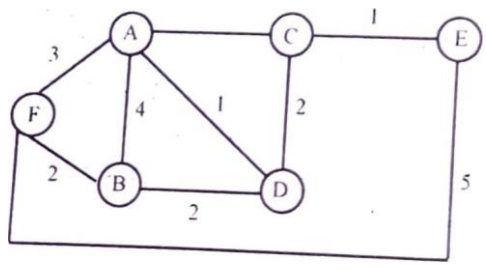
Any particular DFS or BFS of a directed or un-directed graph, each edge gets classified as one ofthe above.

In a DFS of an undirected graph, every edge is either a tree edge (Jr back edge. Here no cross edge goes to a higher numbered or rightward vertex. The reason behind the DFS algorithm is so important is that it defines a very nice ordering of edges of the graph.

Lon Answer e uestions

1. a) Compare BFS and DFS. Discuss the two different ways of representing a graph.

PNBUT 2009]



3

5

OR,

Describe BFS algorithm.

[WBUT 2016]

110' er:
A

1'flrt: t Search: DES of a graph is analogous to the per-order traversal of an For a given graph the DI'S will have the following rules.

gule 1: If possible, visit an adjacent unvisited vertex, mark it visited, and push it on the stack.

2: If Rule I fails, then if possible pop a vertex off the stack follow Rule I fronl lt.

3: Repeat Rule I and Rule 2 until the all the vertices are visited.

Breadth First Search: BPS of a graph is analogous to level-by-level traversal of an fdered tree.

For a given graph the BF S will have the following rules.

Rule 1: Visit all the next unvisited vertices (if any) that is adjacent to the current vertex, and insert them into a queue one at a time on every visit.

Rule 2: If there is no unvisited vertex, remove a vertex from the Queue and make it the current vertex and then follow Rule l.

Rule 3: Repeat Rule I and Rule 2 until the all the vertices visited.

Breadth-first search (BFS) is a graph search algorithm that begins at the root node and explores all the neighboring nodes. Then for each of those nearest nodes, it explores their unexplored neighbor nodes, and so on, until it finds the goal.

The time complexity can also be expressed as IE I + IV I) since every vertex and every edge will be explored in the worst case.

Depth-first search (DES) is an algorithm for traversing or searching a tree, tree structure, or graph. Intuitively, one starts at the root (selecting some node as the root in the graph case) and explores as far as possible along each branch before backtracking. The time complexity is also given by 0(| E l + I VI).

2nd Part:

There are three ways by which graphs can be represented in memory.

i) Adjacency matrices. ii)

Adjacency list.

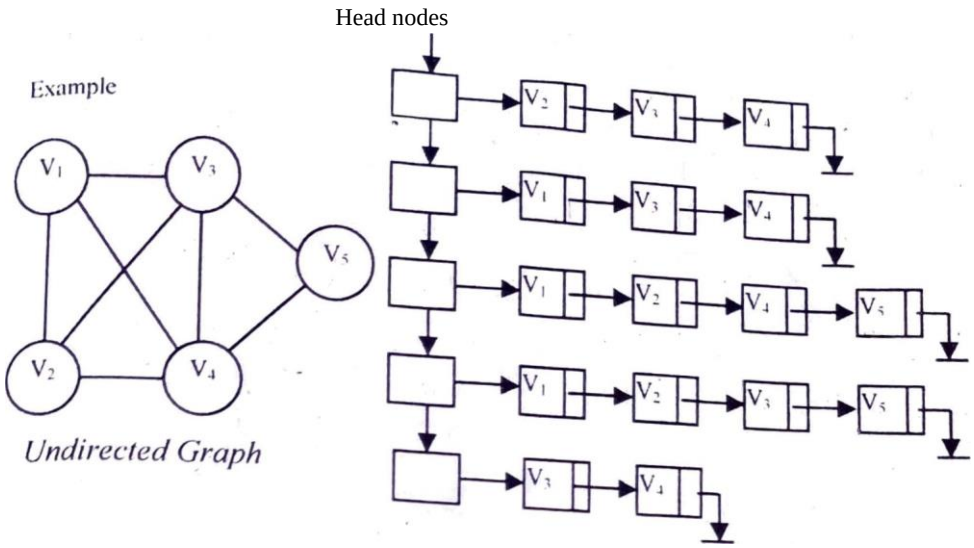
iii) Adjacency multilists

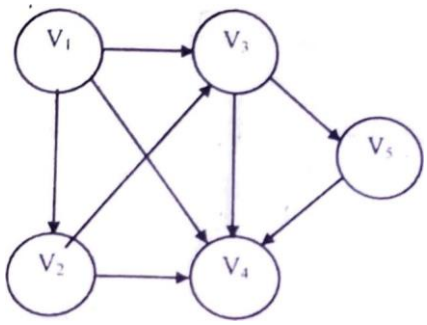
i) Adjacency nmtrices: For a graph G with n vertices the adjacency matrix is a 2D array with n x n element.

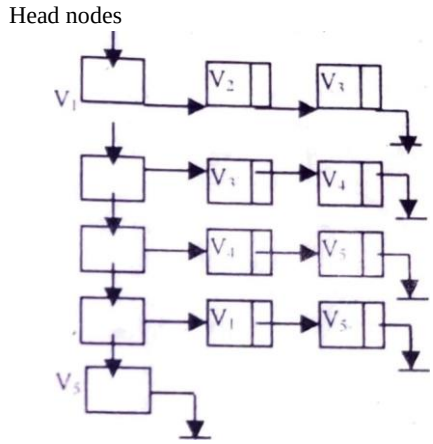
For undirected graphs

= l, if there is an edge between i & j

[i, j] = 0, if there is no edge between i & j.







Directed graph

Drawbacks: When a graph is large, it is necessary to handle to many numbers of pointers, which can be complex.

iii) Adjacency Multilist: From the adjacency list representation of an undirected graph, it is clear that each edge (V i, Vj) is represented by two entries, one is list Vi and another is list Vj. This is unnecessary wastage of the memory as the information present is same at both the nodes ((2, I).

POPUUAR

TO avoid this, the adjacency multilist is used. Using this representation, the nodes are shared amongst the several lists.

In this method for each edge of the graph, there will be exactly one node. But this node will provide the information about two more nodes to which it is incident. he node structure with this re resentation will be:

m				

-- tagfield

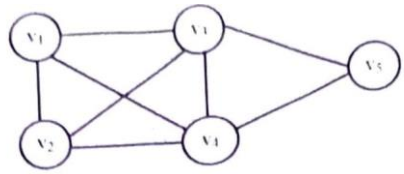
l. V2 -Vertex [1] '

Incident paths

PUBLICATIONS

n) is a one bit Inark ,tield that is used to indicate whether or not the edge has b exatnined.

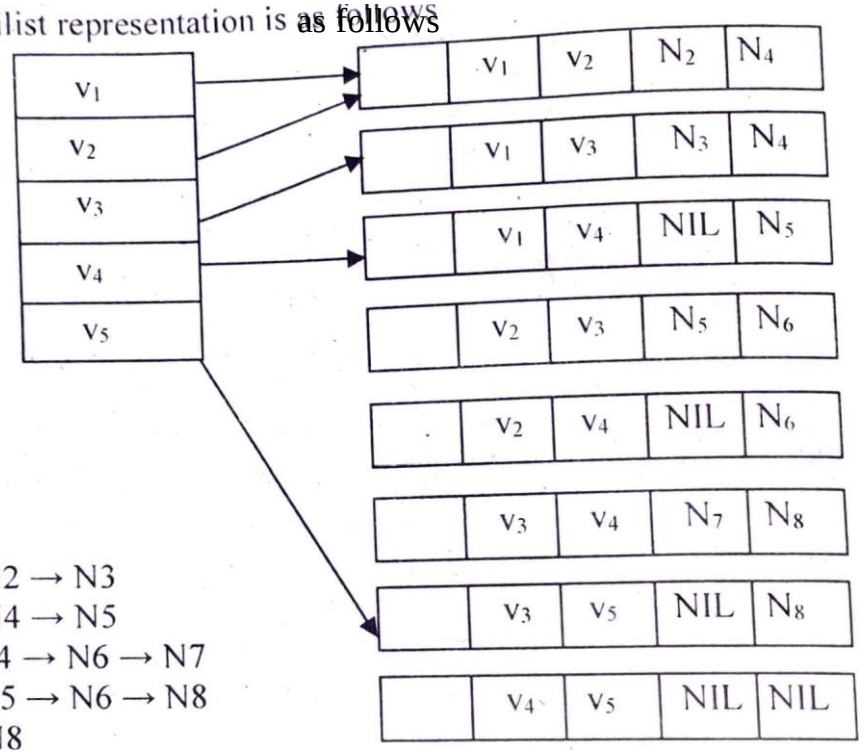
Consider the following graph.



Total distinct edges are:

{v1, v3}, {v1, v4}, {v2, v3}, {v2, v4}, {v3, v4}, {v3, v5} & {v4, v5}. All 0th edges represent the san)e infornnation, because this is an undirected graph.

The adjacency multilist representation is as follows



NS

e:
N1 → N2 → N3
N1 → N4 → N5
N2→ N4 → N6 → N7
N3 → N5 → N6 → N8
N7 → N8

Vertex 5 : N

The lists are:

Vertex I :

NI Vertex 2

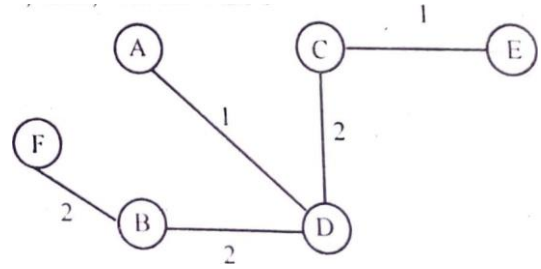
: NI Vertex

3 :

Vertex 4 : N 3

b) Draw the minimum cost spanning tree for the graph given below and also find its cost. [WBUT 20091 Answer:

The tree has edges AD, [3D, CD, CE and BF.



Cost is 8.

Note: Since the cost of A to C is not given in the question it is assunned that the cost 25 .

AT [URE&

What is Complete Graph? (WBIJT 20091

c)that the sum of degree of all the vertices in a graph is always even• tWBIJT 2009)

POPUUAR

OR, show that the number of vertices of odd degree in a finite graph is even•tWBUT 20171

OR, prove that the number of odd degree vertices in a graph is always even•

(WBUT 20181

Arts" part: er: A graph is said to be complete if there exist an edge between every pair of

ettices.

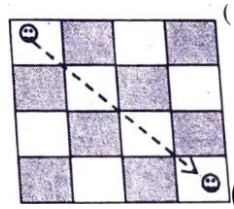
part: The number of odd degree vertices in a graph is always even:

Degree of a vertex is defined as no. of edges incident on a particular vertex with the self loop counted twice. Now, let us take the sum of degree of all the vertices. Now. this summation is an even number, because each edge contributes twice when we are calculate degree Of different vertices. Now, if we take the summation of all the degrees that is nothing but the twice the number of edges Yd(v,) = 2e even number

2. A rat has entered a checkerboard maze through one corner, whose the white boxes are open and black boxes represent obstacles. Develop an algorithm by which the rat can exit the maze through the opposite corner. Clearly explain the representation of the maze and any specific data structure you have used for the algorithm. CNBUT 2011]

Answer:

Let the checkerboard maze be represented by the figure below. For simplicity we show a



4x4 checkerboard. A(iO) (3,3)

(0,0) (3,0)

The maze can be represented by a two dimensional array. Since black boxes are obstacles, the rat will only travel through white boxes. So the best path to follow will be the diagonal, The algorithm will be as follow: Steps:

$i \leftarrow 0; j+-0$

l. Start with A(i, j)

2. While (i < n, j < n)

3. Final position A(n, n)

PUBLICATIONS

short notes on the following: [WBUT 20101

a) BFS vs DFS

b) BFS PNBUT 20121

[WBUT 20131

c) Dijkstra's Algorithm CNBUT 2014, 20181

3. Write

d) DFS in graph

Answer:

a) BFS vs. DFS: Refer to Question No. I(a) of Long Answer Type Questions.

b) BFS: Refer to Question No. I(a) of Long Answer Type Questions.

c) Diikstra's Algorithm:

Dijkstra's algorithm works on the principle that the shortest possible path from the source has to come from one of the shortest paths already discovered. A way to think about this is the "explorer" model--starting from the source, we can send out explorers each travelling at a constant speed and crossing each edge in time proportional to the weight of the edge being traversed. Whenever an explorer reaches a vertex, it checks to see if it was the first visitor to that vertex: if so, it marks down the path it took to get to that vertex.

This explorer must have taken the shortest path possible to reach the vertex. Then it sends out explorers along each edge connecting the vertex to its neighbors.

It is useful for each vertex of the graph to store a "prev" pointer that stores the vertice from which the "explorer" came from. This is the vertex that directly precedes the

POPUUAR

CUrrent vertex on the path from the source to the current vertex.

The pseudocode for Dijkstra's algorithm is fairly simple and reveals a bit more about what extra information needs to be maintained. Vertices will be numbered starting from 0 to simplify the pseudocode.

Given a graph, G, with edges E of the form (VI, 72) and vertices V, and a source vertex, s di st array of distances from the source to each vertex prev array of pointers to preceding vertices :loop index :list of finished vertices :list or heap unfinished vertices

```
/ * Initialization: set every distance to INFINITY until we
discover a path * / for i  0 to IVI  1 dist [i]     INFINITY
prev[i] = NULL end
/ * The distance from the source to the source is defined to be
zero * / dist [s]= 0
/ * This loop corresponds to sending out the explorers walking
the paths, where
```

step of picking " the vertex, v, with the shortest path to

* thecorresponds
5. an explorer arriving at an unexplored vertex * / to

wbiIe(F is missing a vertex) pi Ck the vertex, v, in U with the shortest path to s
add v to F for each edge of v, (VI, v2)

/ *The next step is sometimes given the confusing name
" relaxation " if (dist (VI] + length(vl, v2) < dist) dist [v2] — dist [v 11
+ length(vl, v2) prev [v2] = VI possibly update U, depending on implementation
end if end for end while

d) DFS in graph: Refer to Question No. I(a) of Long Answer Type Questions.

POPULAR

&

PVßLICATIONS

SORTING & HASHING

Chapter at a Glance

- A hash table data structure is just like an array. Data is stored into this array at specific index generated by a hash function. A hash function hashes (converts) a number On a large range, into a number in a smaller range.
- Linear search: Linear or Sequential Search is a method where the search begins at one end of the list, scans the elements of the list from left to right (if the search begins from left) until the desired record is found.

  This type of searching can be perfortned in an ordered list or an unordered list. For ordered, lists that must be accessed sequentially, such as linked lists or files with variable-length records lacking an index, the average performance can be improved by giving up at the first element which is greater than the unmatched target value, rather than examining the entire list.
- Binary Search: In Binary Search the entire sorted list is divided into two parts. We first compare our input item with the mid element of the list and then restrict our attention to only the first or second half of the list depending on whether the input item comes left or right of the mid-element. In this way we reduce the length of the list to be searched by half.
- Hashing: In hash tables, there is always a possibility that two data elements will hash to the
- same integer value. When this happens, a collision occurs i.e. two data members try to occupy the same place in the hash table array. There are methods to deal with such situations like Open Addressing and Chaining.
- Bubble sort: Given an array of unsorted elements, Bubble sort performs a sorting operation on the first two adjacent elements in the array, then between the second & third, then between third & fourth & so on.
- Insertion sort: In insertion sort data is sorted data set by identifying an element that is out of order relative to the elements around it. It removes that element from the list, shifting all other elements up one place.

  Finally it places the removed element in its correct location.

  For example, when holding a hand of cards, players will often scan their cards from left to right, looking for the first card that is out of place. If the first three cards of a player's hand are 4, 5, 2, he will often be satisfied that the 4 and the 5 are in order relative to each other, but, upon getting to the 2, desires to place it before the 4 and the 5. In that case, the player typically removes the 2 from the list, shifts the 4 and the 5 one spot to the right, and then places the 2 into the first slot on the left.
- Quick sort: In Quick-Sort we divide the array into two halves. We select a pivot elem[ent] (normally the middle element of the array) and perform a sorting in such a manner that all the elements

&

to the left of the pivot element is lesser than it & all the elements to it's right is greater than the pivot element. Thus we get two sub arrays.

e Merge sort: In this method, we divide the array or list into two sub arrays or sub lists as nearly equal as possible and then sort them separately. Then the sub-arrays are again divided into another sub arrays.

• Heap sort: A heap takes the form of a binary tree with the feature that the maximum or minimum element is placed in the root. Depending upon this feature the heap is called max-

Ileap or min-heap respectively. After the heap construction the elements from the root are taken out fronl the tree and the heap structure is reconstructed. This process continues until (he heap is etnpty.

Multi le Choice e uestions

The ratio of the number of items in a hash table, to the table size is called the [WBUT 2007, 2009, 2016]

a) load factor b) item factor c) balanced factor d) all of these gns»er: (a)

2. Which of the following is not a requirement of good hashing function?

a) Avoid collision b) Reduce the storage space

c) Make faster retrieval d) None of these [WBUT 2008, 2015]

Answer: (b)

3. Stability of Sorting Algorithm is important for [WBUT 2007]

a) Sorting records on the basis of multiple keys
b) Worst case performance of sorting algorithm
c) Sorting alpha numeric keys as they are likely to be the same
d) None of these

Answer: (a)

4. Which of the following is the best time for an algorithm? [WBUT 2007]

a) O(n) b) O(log2 n) c) 0(2n) d) O(n log2 n)

Answer: (b)

5. The Linear Probing Technique for collision resolution can lead to [WBUT 20091

a) Primary clustering b) Secondary clustering
c) Overflow d) Efficiency storage utilization

Answer: (a)

6. The fastest sorting algorithm for an almost already sorted array is [WBUT 20091

a) quick sort b) merge sort c) selection sort d) insertion sort

Answer: (d)

7. The time complexity of binary search is PUBUT 2009]

a) $O(n^2)$ b) O(n) c) O(log n) d) O(n log n)

Answer: (c)

8. The best case time complexity of Bubble sort technique is CWBUT 20101

b) O(n ^{2})c) ()(nlogn) d) $O(\log n)$

Answer: (a)

DSA-III

(WBUT 20101

9. Which of the following sorting procedures is c) the Merge slowest?sort d) Bubble sort
a) Quick sortb) Heap sort

Answer: (d)

10. Which of the following traversal techniques lists the elements [WBUT Of 2011, a binary20151 search tree in ascending order? c) Inorder d) None of these

The best case complexity of insertion sort is — [WBUT 2011]

a) $O(n^2)$ b) log: n) c) $O(n^3)$

Answer: (d)

Which of the following is not related to hashing? [WBUT 2011]

a) pre-order b) Post-order

Ans"ct•: (c) linked lists. [WBUT 2011)

11. Binary search cannot be used in a) True b) False

Answer: (b) data structure [WBUT 20111

12. Breadth-first-search algorithm uses ..... tree
a) stack b) queue c) binary d) none of these

Answer: (b)

a) Synonymsb) Collision c) Balance d) Load factor

Answer: (c)

15. A machine needs a minimum of IOOsec to sort 1000 names by quick sort. The minimum time needed to sort 100 names will be approximately [WBUT 20121

a) 72.7 sec b) 11.2 sec c) 50.2 sec d) 6.7 sec

Answer: (d)

16. What will be the time complexity for selection sort to sort an array Of n elements? [WBUT 2012, 2016]

a) O(log n) b) O(n log n) d) $O(n^2)$

Answer: (d)

17. The best sorting technique when the data is almost sorted is [WBUT 2013]

a) Selection sort b) Bubble sortc) Quick sort d) Insertion sort

Answer: (d)

18. Which of the following is a hash function? [WBUT 20141

a) Quadratic probing b) chaining

c) open addressing d) folding

Answer: (a)

&

The number order of using swapping bubble needed sort is to sort numbers 8, 22, 7, 9, 31, [WBUT 19, 5, 2014113 in

ascending b) 12 c) 13 d) 14

[WBUT 20141

20. Binary search uses
a) divide and reduce strategy b) divide and conquer strategy
c) heuristic search d) both (a) and (b)

(WBUT 20151

21. Merge sort uses
a) divide and conquer strategy b) backtracking approach
c) heuristic search d) greedy approach

Ans»cr: (a)

PWBI.JT 20171

22. The a) unsorted prerequisite arraycondition of Binary search b) isascending order array
c) descending order array d) sorted array Answer: (d)

Short Answer Type uestions

1. Explain the advantages of binary search over sequential search.

[WBUT 2007, 20131

Define Hashing. (WBUT 201

What .is collision? tWBUT 201

Explain Linear Probing & Quadratic Probing with example. IWBUT 200

Answer:

A sequential search of either a list, an array, or a chain looks at the first item, the second item, and so on until it either finds a particular item or determines that the item does not occur in the group. Average case of sequential search is O(n)

A binary search of an array requires that the array be sorted. It looks first at the middle of the array to determine in which half the desired item can occur. The search repeats this strategy on only this half of the array.

The benefit of binary search over linear search becomes significant for lists over about 100 elements. For smaller lists linear search may be faster because of the speed of the simple increment compared with the divisions needed in binary search.

Thus for large lists binary search is very much faster than linear search. but is not worthwhile for small lists.

Binary search is not appropriate for linked list structures (no random access for th middle term).

2. What is hashing? [WBUT 2007, 2012, 2014, 201

OR,

DSA-113

Define 'Hashing'. Explain with addressing.a suitable example the collision resolution schem using linear probing with open

Define Hashing. Explain one collision resolution scheme citing one example.[WBUT

20131

OR,

Write two hash functions [WBUT 20171 Answer:

yqu

Hashing is a method for storing and retrieving records from a database. It lets insert deletae. a•nd search for records based on a search key value. When properly innplemented these operations can be performed in constant time. In fact, a properly tuned hash system typically looks at only one or two records for each search, insert, or delete operation. This is far better than the O(log n) average cost required to do a binary search on a sorted array of n records, or the O(log n) average cost required to do an operation on a binary search tree. However, even though hashing is based on a very simple idea, it is Stit•prisingly difficult to implement properly. Designers need to pay careful attention to all of the details involved with implementing a hash system.

A hash system stores records in an array called a hash table, which we will call HT . Hashing works by performing a computation on a search key K in a way that is intended to identify the position in HT that contains the record with key K. The function that does this calculation is called the hash function, and is usually denoted by the letter h Since hashing schemes place records in the table in whatever order satisfies the needs of the address calculation, records are not ordered by value. A position in the hash table is also known as a slot. The number of slots in hash table HT will be denoted by the variable M with slots numbered from 0 to M- l .

2 nd Part:

A collision between two keys K & K' occurs when both have to be stored in the table & both hash to the same address in the table.

3 rd Part:

Open addressing: It is a general collision resolution scheme for a hash table. In case of collision, other positions of the hash table are checked (a probe sequence) until an empty position is found.

The different types of Open addressing scheme includes

a) Linear Probing (Sequential Probing)
b) Quadratic Probing
c) Double Hashing (Re Hashing)

Linear probing is a used for resolving hash collisions of values of hash functions by sequentially searching the hash table for a free location. This is accomplished using two values - one as a starting value and one as an interval between successive values in modular arithmetic. The second value, which is the same for all keys and known as the

is repeatedly added to the starting value until a free space is tound. or the entire epsize.

table is traversed.

The function for the rehashing is the following:

ehash(key)=(n+1) % k; For example, we have a hash table that could accommodate 9 information, and the data to be stored were integers. To input 27. we use hash(key)= 27 % 9=0. Therefore, 27 is stored at O. If another input 18 occurs, and we know that 18 % O, then a collision would occur. In this event. the need to rehash is needed. Using linear probing, we have the ehash(key)= (18+1) % 9= l . Since I is empty, 18 can be stored in it.

Quadratic probing:

In order to prevent collision we use quadratic probing scheme. ,

In quadratic probing.

We start from the original hash location i

- If a location is occupicd. 'esc chcck the locations i' I ' i +22 i+4² ...
- We wrap around from thc last table location to the first table location if necessary.

Let us take thc followving cxamplc•

Table Size is I l (0 10) _ flash Function:

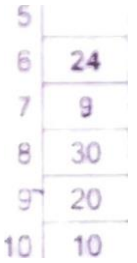
h(x) x rnod I l _ Insert (20.

30,2.13.25,24.10,9):

- 2() mod
- 30 mod 11 = 8
- 2 mod 11 = 2
- 13 mod 11 = 2 → 2+1²=3
- 25 mod 11 = 3 → 3+1²=4
- 24 mod 11 = 2 → 2+1², 2+2²=6
- 10 mod 11 = 10
- 9 mod 11 = 9 → 9+1², 9+2² mod 11, 9+3² mod 11 =7

I 9

The tieure shosss the correspondillé entries•in the hash table.



DSA-115

3. Prove that, the best case time complexity for quick sort is O(nlogn) [WBUT for input 20081size of n. Answer:

The analysis of the procedure QUICK SORT is given by

T(N) = P(N) + T(J-LB) + T(UB - J) where P(N), T(J-LB), and T(UB-J) denote the times to partition the given table, sort the left subtable, and sort the right subtable, respectively. Note that the time to partition a table is O(N). The worst case occurs when, at each invocation of the procedure, the current table is partitioned into two subtables with one of them being empty (that is J LB or J = UB). Such a situation, for example, occurs when the given key set is already sorted.

The worst case time analysis, assuming J = LB, then becomes

Tw = P(N) + Tw(0) + Tw(N-l)

= + Tw(N-l)

= + + Tw(N-2)

= + + + Tw(N-3)

-c * {(N+1)(N)}/2 = O(N2)

The best case analysis occurs when the table is always partitioned in half. that is, J= [(LB+UB)/2]. The analysis becomes:

Tb = P(N) + 2Tb(N/2)

= c*N + 2Tb(N/2)

= + 2c(N/2) + 4Tb(N/4)

= + 2c(N/2) + 4c(N/4) + 8Tb(N/8)

= (log2N)*c*N + 2log2N*Tb(1)

= O(Nlog2N)

4. Give an algorithm to search for an element in an array using binary search. [WBUT 2008]

OR,

What is the precondition of performing binary search in an array? Write the Binary Search algorithm.

[WBUT 20101

Answer:

Let us consider an array A of size N = 10. The left index and right index are left = 0 right = (N—l) = 9 int bin_ search (int A [J , int n, int item)

int left 0 , right = n - 1; int flag = 0; ///a flag to indicate whether the element is found

•URL

```
(left    right) // till left and right cross each other

                          (left    right) / 2; : /
divicie tho array -into two halve.5
```

A(mid)) set flag 4 åf the element is found

```
mid = (
if (item ==
{
flag = 1; //
break;
}
else if
right = mid
else
left = mid +
flag;
```

(item S

A(mid))

```
    - 1; / • compute new right from the right gub
                                              array • /

                          + 1; compute new left from
the left sub array* /
```

S. "Binary search technique cannot be implemented using Linked list. " Justify the validity of the statement. [WBUT 2009]

The statetnent is not true as it can Binary Search can be implemented using Linked List — houoer it would be less efficient than array.

Binary search on an array so fast and efficient because of its the ability to access any element in the array in constant tinw. Thus one can get to the middle of the array just by saying array[nliddle].

The same cannot be done with a linked list. One needs to write an algorithm to get the value of the middle node of a linked list. In a linked list, one loses the ability to get the ialue of any node in a constant time.

One solution to the inefficiency of getting the middle of the linked list during a binary search is to have the first node contain one additional pointer that points to the node in the middle. Decide at the first node if one needs to check the first or the second half of the linked list. Continue doing that with each half-list.

6. Draw a minimum heap tree from the below list: [WBUT 2012, 2014)

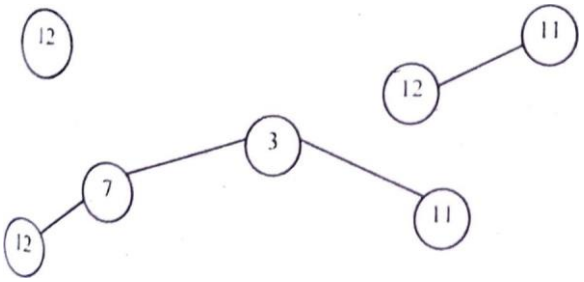
12, 11, 7, 3, 10, -5, o, 9, 2

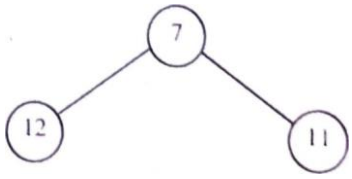
Now do the heap sort operation over the heap tree which you have formed. Write the insertion sort algorithm.

Answer:

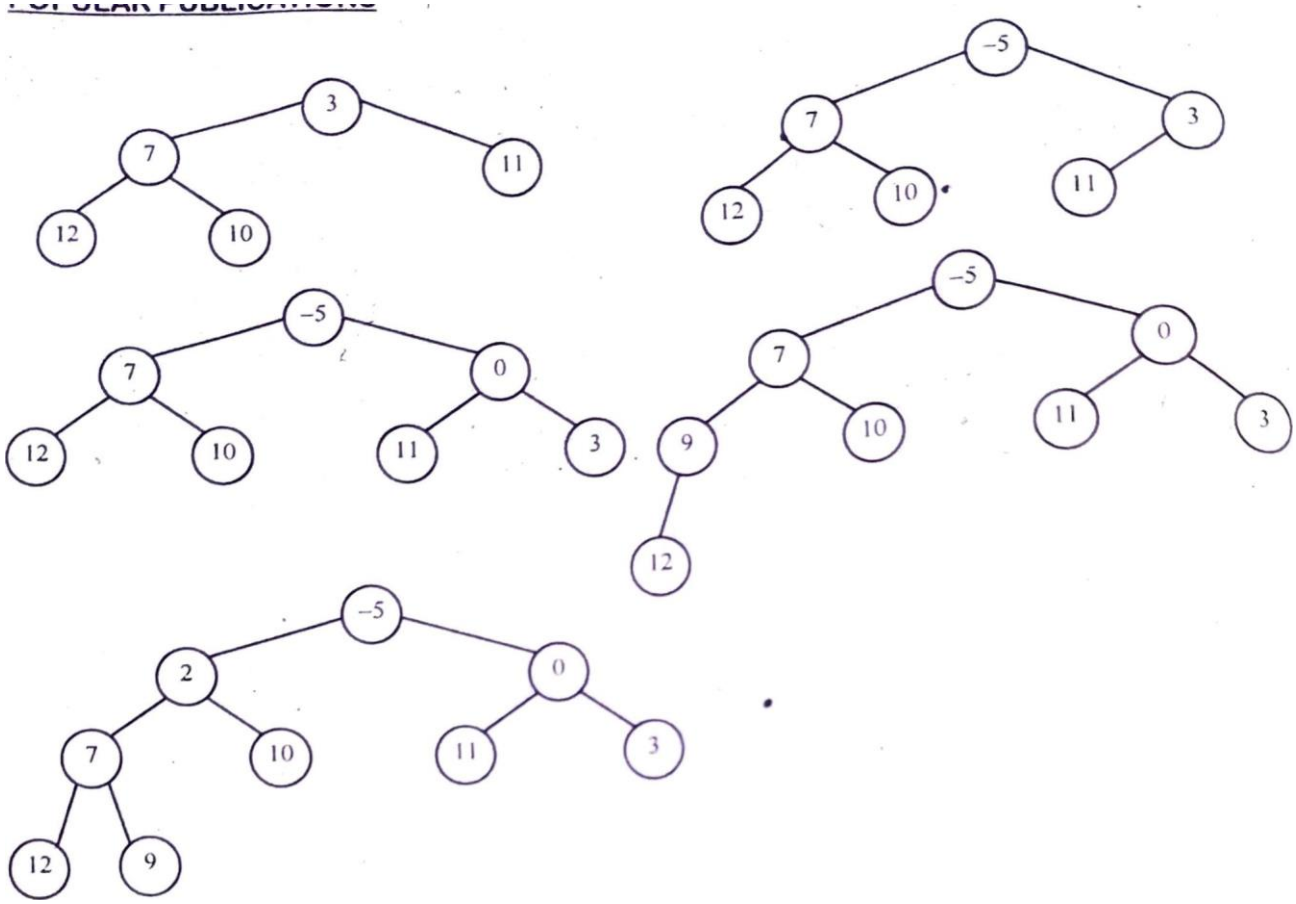
I ^{S}! Part:

The steps are as shown below:





l)SA-117



Performing sorting in min heap will result in the array being sorted in descending order. The array now looks like: -5, 2, 0, 7, 10, I l, 3. 12. 9

The steps for sorting are:

Tree nodes: 9, 2, 0, 7, 10, I l , 3, 12 Sorted Array: -5

Tree nodes: 0, 2, 9, 7, 10, I l , 3, 12 Sorted Array: -5

Tree nodes: 12, 2, 9, 7, 10, I l . 3 Sorted Array: 0, -5

Tree nodes: 3, 2, 9, 7, 10, I l , 12 Sorted Array: 0, -5

Tree nodes: 2, 3, 9, 7, 10, I l , 12 Sorted Array: 0, -5

Tree nodes: 3, 9, 7, 10, I l , 12 Sorted Array: 2, 0, -5 Tree nodes:

9, 7, 10, I l , 12 Sorted Array: 3, 2, 0, -5

Tree nodes: 7,9, 10, I l , 12 Sorted Array: 3,2, 0, -5

Tree nodes: 9, 10, I l , 12 Sorted Array: 7, 3.2, 0, -5

Tree nodes: 10, I l , 12 Sorted Array: 9, 7, 3,2, 0, -5 Tree nodes: I l , 12 Sorted Array: 10, 9, 7, 3,2, 0, -5 Tree nodes: 12 Sorted Array: I l, 10, 9, 7, 3,2, 0, -5

Tree nodes: Sorted Array: 12, I l , 10, 9, 7, 3,2, 0, -5

2nd Part:

If the first few objects are already sorted, an unsorted object can be inserted in the sorted set in proper place. This is called insertion sort. An algorithm consider the elements one at a time, inserting each in its suitable place among those already considered (keeping

them sorted). Insertion sort is an example of an incremental algorithm; it builds the sequence one number at a time. pseudocode we a procedure INSERTION SORT. It takes as parameters an array All.. n] and the length n of the array. The array A is sorted in place: the numbers are rearranged within the array, with at most a constant number outside the array at any time. INSERTION_SORT (A)

l . FOR j 4--= 2 TO length[A1
DO key 2. $\leftarrow A[j]$

{Put AliI into the sorted sequence All . . j — Il)

3.

4.

WHILE i > O $4[i$ and
> key 5. $\leftarrow A[i]$

DO A [i 4-11 6.

i — i- l

8. A [i + Il key

7. Write the pseudo code for Heap sort. (WBUT 20131

Answer:

The Algorithm for heap-sort is as follows.

```
#include <stdio .h>
#include <conio . h> #define N 6 void
buildheap ( int (l , int) ; void heapsort
(int [ l , int) ; int main (void) int
heapArr [N) = (15, 19, 10, 7 , 17, 6) ;

  int i; print f ( " \nBefore
  Sorting: \ n" ) ; for (i
  print f ( , heapArr[il ) ;
  buildheap (heapArr, N) ;
  heapsort (heapArr, N) ;
  print f ( " \nAfter Sorting:
  \ n" ) ; for (e i print f (
  " \ t" , heapArr [i) ) ;
  getch() ; return 0 ;

void buildheap (int , int n)
  int i, val, s, f;
  for (i= 1:
```

```
Val  =  x i l:s
    i;
```

DSA-119

```
   while (s > 0    x(fl < val)


      f = (s - 1)

    xls) = val;


void heapsort (int xl) t int n)

  int i, s, f, ivalue; for
  (i 1; i > 0; i--)

    ivalue = x lil ;

    f =o; if
     (i    -
     1) s  1 ;

       s = if (i > 2 && x | 2 ) >
     xlll ) while (s 0 && ivalue <
     xls) )

       x[f]- xls]
       ; =s; s=

       if (s          1 && x(s) < xls + 11 )

        if (s   1) s
           1;

      x[f] =ivalue ;
```

8. Deduce the average time complexity of Quicksort algorithm. [WBUT 20151

Answer:

The basic idea of Quicksort is as given below:

1 . Pick one element in the array, which will be the pivot.

2. Make one pass through the array, called apartilion step, re-arranging the entries so that:

- the pivot is in its proper place.

u entries smaller than the pivot are to the left of the pivot.
• entries larger than the pivot are to its right.

3. Recursively apply quicksort to the part of the array that is to the left of the pivot, and to the right pan of the array.

Analysis

T(N) = T(i) + T(N - i -1) + cN

The time to sott o the the time file is to equal sort the to left partition with i elements, plus o the time to sort the right partition with N-i-l elements, plus o the time to build the partitions

The average value ofT(i) is I/N times the sum ofT(0) through T(N-I)

S TO), j = 0 thru N-l

Multiply by N

NT(N) TO)) + cN*N

To remove the sumrnation, we rewrite the equation for N-l :

-1)T(N-1) = TO)) + c(N-l j O thru N-2 and

subtract:

N) - (N-1)T(N-1) = 2T(N-l) + 2cN -c prepare for telescoping.

Rearrange terms, drop the insignificant c:

NT(N) = (N+l + 2cN

Divide by N(N+I):

/(N+1) = +

Telescope: 2c/(N+l)

T(N)/(N+1) = T(N-1)/N + 2c/ + 2c/(N+l)

T(N-1)/(N) = T(N-2)/(N-1)+ 2 2c/(N)

T(N-2)/(N-1) = T(N-3)/(N-2) + 2c/(N-l)

....

T(2)/3 = T(1)/2 + 2c /3

Add the equations and cross equal terms:

+1) = T(1 +2c S = 3 to N+l

N+1)(1/2 + 2c S(l/j))

The sum S (l/j), j =3 to N-l , is about LogN

Thus T(N) = O(NlogN)

9, What is the primary criterion of performing binary search technique on a list of data? [WBUT 2015) Answer:

A binary search algorithm is one method of efficiently processing a sotted list to determine rows that match a given value of the sorted criteria. It does so by "cutting" the set Of data in half (thus the term binary) repeatedly, with each iteration comparing the supplied value with the value where the cut was made. If the supplied value is greater than the value at the cut, the lower half of the data set is ignored, thus eliminating the need to compare those values. The reverse happens when the skipped to value is less than

the supplied search criteria. This cornparison rcpcats until there are no more values to

The binary searvh algorithll) able to elitninate the need to do a comparison on each of tChe mcordss and in doing so reduced the overall computational complexity of our request tor the database server. Using the snulllcr set of sotted weight data, we are able to avoid needing to load all the record data into nrtnory in order to compare the product weights to our searvh criteria.

10. What is Load Factor? Why do we need hashing? How does a hash table allow 0(1) searching? Why is prime number chosen for computing a hash function?[WBUT 2018]

Arts" cr.' 1'

Part:

Loads factor refers to the ratio of the number of records to the number of addresses within a data structure. 2nd Part:

Hashing is used to inbox and retrieve items in a database because it is faster to find the item using the shorter hashed key than to find it using the original value.

3 rd Part:

A hash table is an array containing all of the keys to search on. The position of each key in the array is determined by the hash function, while can be any function which always maps the same input to the same output. So, we shall assume the hash function as I).4 th Part:

The prime numbers are used to minimize collisions when the data exhibits some particular pattern. The reason prime numbers are used to neutralize the effect of patterns in the keys in the distribution of collisions of a hash function.

Lon Answer e uestions

1. a) Explain with an example the Merge sort algorithm. b) Write an algorithm for Merge sort. [WBUT 2007]

[WBUT 2007]

OR,

Show the operation of merge sort with an example.

c) Compare the best case time complexity of selection sort and insertion sort.2013] [WBUT

Answer: [WBUT 2007]

a) Merge-sort is based on the divide-and-conquer paradigm. The Merge-sort algorithm can be described in general terms as consisting of the following three steps:

1. Divide Step

Ifgiven array A has zero or one element, return S; it is already sorted. Otherwise, divide into two arrays, Al and 742, each containing about half of the elements ofA.

Recursively sort array Al and A2.

conquer Step

the elenwnts back in A by tnerging the Qortecl arrays Al and A' into a sorted

Five Mer-oe Sort alqovi thm• /

```
(int a l ) , int low, int high)

mid;
( Iowanigh)

      mid- ( low•high) / 2;
mergesovt (a, 10w,mid) ;
mergesovt (a, mid+l , high)
; merge (a, low, high, mid)
; return (0) ;

verge function* /
ace(int , int low, int high, int mid)

int i,assume size of array is / i -low; j=mid+l; k:
low; while( (i<=mid) ( j<=high) )
      if(a[i]<a[j])
      {
            c[k]=a[i];
            k++;
            i++;
```

```
            c[k]=a[j];
            k++;
            j++;


whilei<=mi
            c[k]=a[i];
            k++;
            i++;

while ( j < -high)
            c[k]=a[j];
            k++;
```

1)SA-123

```
for ( i = low; i < k; i++)
        a[i]=c[i];
```

Let the original unsorted array is: 8 4 5 2 5
In the first step it gets split into
8 4 Ill 5 2 5 (Ill denotes the divider)
Further splitting and merging is shown in the next steps: 8
Ill 4 Ill 5 2 5 (split)
8 Ill 4 Ill 5 2 5 (can't split, merges a single-element array and returns)
8 Ill 4 Ill 5 2 5 (can't split, merges a single-element array and returns)
4 8 Ill 5 2 5 (merges the two segments into the array) 4
8 Ill 5 Ill 2 5 (split)
4 8 Ill 5 Ill 2 5 (can't split, merges a single-element array and returns) 4
8 Ill 5 Ill 2 Ill 5 (split)
4 8 Ill 5 Ill 2 Ill 5 (can't split, merges a single-element array and returns)
4 8 Ill 5 Ill 2 Ill 5 (can't split, merges a single-element array and returns)
4 8 Ill 5 Ill 2 5 (merges the two segments into the array)
4 8 Ill 2 5 5 (merges the two segments into the array) 2
4 5 5 8 (merges the two segments into the array)

c) Selection sort Performance: Best Case is O(n2) because even if the list is sorted, the same number of selections must still be performed.

Insertion Sort Performance: The hest-case time complexity is when the array is already sorted, and is O(n).

2. a) Explain with a suitable example, the principle of operation of Quick sort. [WBUT 2007, 2009, 2010] Answer:

In Quick-Sort we divide the array into two halves. We select a pivot element (normally the middle element of the array) & perform a sorting in such a manner that all the elements to the left of the pivot element is lesser than it & all the elements to it's right is greater than the pivot element. Thus we get two sub arrays. Then we recursively call the quick sort function on these two sub arrays to perform the necessary sorting.

Let us consider the following unsorted array:

a [ ]= 45 26 77 14 68 61 97 39 99 90

step 1:

We choose two indices as left = 0 and right = 9. We find the pivot element by the formula alletl + right] / 2 = a [0 + 9] / 2 = a [4]. So the pivot element is a [41 68. We also start with a [left] = a [0] = 45 and a [right] = a [9] = 90

n,pare all the elernents to the left of the pivot element. We increase the value of left I each tirne, the latest when value an eletnent of left. In is less our exanople than 68. the We left stop value when is there 2 i.e., is left=2•no such element.

Illpare all the elennents to the right of the pivot element. We decrease the value of right cOby I each titne when an elennent is greater than 68. We stop when there is no such We tecord the latest value of right. In our example the right value is 7 i.e., oghts7.

Stcpsince left < = right we swap between a [left] and a [right]. That is between 77 and 39. fie array now looks like

45 26 39 14 68 61 97 77 99 90.

Stcp We further f: increment the value of left and decrement the value of right by I respectively

ie.. left=2+ 1 = 3 and right = 7 - 1 = 6.

We repeat the above steps until left < = right.

We will see that at one stage that the left and right indices will cross each other and we ill find that our away has been subdivided. That all the elements lying to the left of the pivot element is less than it and all to it's right are greater. We then make recursive calls ofquick sort on this two sub arrays.

b) Find the complexity of Quick sort algorithm. WBUT 2007, 2013]

Answer:

Refer to Question No. 8 ofShort Answer Type Questions.

3. a) Why is hashing referred as a heuristic search method? [WBUT 2008]

Answer:

A heuristic algorithm, is an algorithm that is able to produce an acceptable solution to a problem in many practical scenarios, in the fashion of a general heuristic, but for which there is no formal proof of its correctness. Alternatively, it may be correct, but may not be proven to produce an optimal solution, or to use reasonable resources. Heuristics are typically used when there is no known method to find an optimal solution, under the given constraints (of time, space etc.) or at all. It is a technique whereby or items are placed into a structure based on a key to-address transformation. We use Hashing for

l) Performing optimal searches & retrieval of data at a constant time i.e. l)
2) It increases speed, betters ease of transfer, improves retrieval, optimizes searching of data and reduces overhead.

That's why hashing are sometimes referred as heuristic search method.

b) What is the primary advantage of hashing over deterministic search algorithms? [WBUT 2008]

Answer:

In deterministic search method we try to explore all the data items sequentially, one by one which gives different polynomial time complexity. But hashing is the searching method where the searching time is always 0(1). Is this sense later is always advantageous.

c) Define collision. Discuss two collision resolution techniques and compare their performances. [WBUT 20081

OR,

Discuss different collision resolution techniques. [WBUT 2015, 2017] Answer:

A collision between two keys K & K' occurs when both have to be stored in the table & both hash to the same address in the table.

The two collision resolution techniques are:

- Open addressing: It is a general collision resolution scheme for a hash table. In case of collision, other positions of the hash table are checked (a probe sequence) until an empty position is found.

  The different types of Open addressing scheme include:

- a) Linear Probing (Sequential Probing)
  b) Quadratic Probing
  c) Double Hashing (Re Hashing)

- Chaining: It is a collision resolution scheme to avoid collisions in a hash table by making use of an external data structure. A linked list is often used.

Linear Probing:

Refer to Question No. 2 ofShort Answer Type Questions.

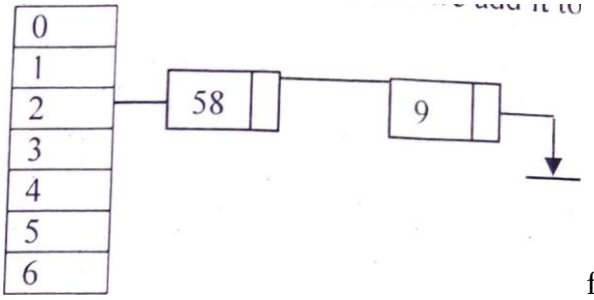
Chaining: In open addressing, collisions are resolved by looking for an open cell in the hash table. A different approach is to create a linked list at each index in the hash table. A data item's key is hashed to the index in the usual way, and the item is inserted into the linked list at that index. Other items that hashes to the same index are simply added to the linked list at that index. There is no need to search for empty cells in the primary hash table array. This is the chaining method. Let us consider the following elements

89, 18, 49, 58, 9, 7,

H(89) 5 (Using Division — Remainder Method) H(1 8) 4 (Using Division — Remainder Method) H(49) 0 (Using Division — Remainder Method) H(58) 2 (Using Division — Remainder Method)

H (9) = 2 (Using Division — Remainder Method)

Now here, already there is one element in the position 2, which is 58 in our example. But 9 is also hashed to position 2, which is occupied by 58 in our example. When this kind Of situation occurs we say that a collision has taken place.

collision avoided by chaining method is an adjacency list representation. Whenever a collision takes place we just add to the adjacency list to the corresponding header where the our collision example occurred.collision occurred at header node 2. So we just add 9 and 58 to it as an adjacency list. IV any 

further collision occurs at 2 we add it to our existing list.

hash table uses hash functions to compute an integer value for searching a data. This integer value can then be used as an index into an array, giving us a constant tinie complexity to find the requested data. However, using chaining, we cannot always achieve the average time complexity of O (l) . If the hash table is very small or the hash function is not good enough then the elements can start to build in one index in the array. And hence all n elements could end up in the same linked list associated to that index. Therefore, to do a search in such a data arrangement is equivalent to looking up a data element in a linked list, something we already know to be O (n) time.

d) Why the hash functions need to be simple? [WBUT 2008] Answer:

The reason is that, the computation itself will consume less amount of time.

4, What do you mean by hashing? What are the applications where you will prefer hash tables to other data structures? What do you mean by collision? How is it handled? [WBUT 2011] Answer:

Hashing is a method for storing and retrieving records from a database. It lets you insert, delete, and search for records based on a search key value. When properly implemented, these operations can be performed in constant time. In fact, a properly tuned hash systenn typically looks at only one or two records for each search, insert, or delete operation. This is far better than the O(log n) average cost required to do a binary search on a soiled array Ofn records, or the n) average cost required to do an operation

on a binary search tree. However, even though hashing is based on a very simple idea. it is surprisingly difficult to implement properly. Designers need to pay' careful attention to all of the details involved with implementing a hash system.

hash system stores records in an array called a hash table (HT). Hashing works by krforming a computation on a search key K in a way that is intended to identify the)ositjon in HT that contains the record with key K. The function that does this calculation called the hash function, and is usually denoted by the letter h. Since hashing schemes lace records in the table in whatever order satisfies the needs of the address calculation. rds are not ordered by value. A position in the hash table is also known as a slot. he

number of slots in hash table HT will be denoted by the variable M with slots nUmbered from 0 to M - 1.

and The goal for a hashing system is to arrange things such that, for any key value K some hash function h, i h(K) is a slot in the table such that 0 i < M, and we have the key of the record stored at HT [i] equal to K.-

Hash tables are commonly used to implement many types of in-memory tables. They are used to implement associative arrays (arrays whose indices are arbitrary strings or other complicated objects), especially in interpreted programming languages like A WK, perl and PHP. Hash tables can be used to implement caches, auxiliary data tables that are used to speed up the access to data that is primarily stored in slower media. In this application hash collisions can be handled by discarding one of the two colliding entries—usually erasing the old item that is currently stored in the table and overwriting it with the new item, so every item in the table has a unique hash value.

A collision between two keys K & K' occurs when both have to be stored in the table & both hash to the same address in the table.

Linear probing is a used for resolving hash collisions of values of hash functions by sequentially searching the hash table for a free location. This is acconnplished using two values - one as a starting value and one as an interval between successive values in modular arithrnetic. The second value, which is the same for all keys and known as the stepsize, is repeatedly added to the starting value until a free space is found, or the entire table is traversed.

The function for the rehashing is the following:

rehash(key)=(n+l) % k;

For example, we have a hash table that could accommodate 9 information, and the data to be stored were integers. To input 27, we use hash(key)= 27 % 9=0. Therefore, 27 is stored at 0. If another input 18 occurs, and we know that 18 % 9= 0, then a collision would occur. In this event, the need to rehash is needed. Using linear probing, we have the rehash(key)= (18+1) % 9= l . Since I is empty, 18 can be stored in it.

5. a) What do you mean by external sorting? How does it differ from internal sorting? [WBUT 2008, 2017, 20181 Answer:

External sorting is the method when the sorting takes place with the secondary memory. The time required for read and write operations are considered to be significant e L'. sorting with disks, sorting with tapes.

Internal sorting on the other hand is defined as a sorting mechanism where the sorting takes place within the main memory. The time required for read and write operations are considered to be insignificant e.g. Bubble Sort, Selection sort, Insertion sort etc.

Internal sorting is faster than external sorting because in external sorting we have to consider the external disk rotation speed, the latency time etc. Such operations are costly than sorting an array or a linked list or a hash table.

b) write an algorithm for sorting a list numbers in ascending order using selection sod technique. OR, CWBUT 2008, 2018] Write a C function for selection sort and also calculate the time complexity for

selection sort. [WBUT 2010] Answer:

In selection sorting we select the first element and compare it with rest of the elements. The minimum value in each comparison is swapped in the first position of the array. puring each pass elements with minimum value are placed in the first position, then second then third and so on. We continue this process until all the elements of the array

are soiled.

```
////n denotes the no.  of  elements  in
the  array  a  void  int  selecsort (int a
[ j , int loc, min, temp, i, j ; for (i= 0; 1

   min = a[1loc  i  ;
   for (j = 1 +if (a
   [j J < min)

       min    = a[j]
       loc    j ;


  if (loc    i )
     temp = a[1
     a[i] = a[l    a
     [loc] ; a [loc]
         t emp ;


print f ( " \nThe sorted array is:
\ n" ) ; for (i = 0; iprint f ( Ed",
a [i] ) ;
```

Time Complexity of Selection Sort

For first pass the inner for loop iterates n - I times. For second pass the inner for loop iterates n - 2 times and so on. Hence the time complexity of selection sort is

$(n-1)+(n-2)+(n-3)+\ldots\ldots\ldots\ldots\ldots\ldots$
$=[(n-1)*(n-1+1)]/2$

+2+

1

[n(n-l)J/2
= 0 (112)

The selection sort minimizes the number of swaps. Swapping data items is time consuming compared with comparing them. This sorting technique might be useful when the amount of data is small.

b) The basic idea of Bubble Sort is to:

l . Data elements are grouped into two sections: a sorted section and an Un-sorted section.

2. Go through every element in the un-sorted section and re-arrange its position with its neighbour to put the element with higher order on the higher position. At the end, the element With the highest order will be on top of the un-sorted section, and moved to the bottom of the sorted section.

3. Repeat step 2 until no more elements left in the un-sorted section.

Performance: Best Case: O(n), Worst Case: 001^2)

The basic idea of Selection Sort:

l . Data elements are grouped into two sections: a sorted section and an un-sorted section.

2. Assuming the sorting order is from low to high, find the element with the lowest comparable order from the un-sorted section.
3. Place the found element to the end of the sorted section.
4. Repeat step 2 and 3 until no more elements left in the un-sorted section. Performance: Best Case: $O(n^2)$, Worst Case: $O(n^2)$

7. Explain the merge sort algorithm. Why does it run faster than bubble sort in most of the cases? Show how the merge sort algorithm will sort the following array in increasing order:

100, 90, 80, 70, 60, 50, 40, 30, 20

Analyze the time complexity of the merge sort algorithm. [WBUT 20111 Answer:

The Mergesort algorithm is based on a divide and conquer strategy. First, the sequence to be sorted is decomposed into two halves (Divide). Each half is sorted independently (Conquer). Then the two sorted halves are merged to a sorted sequence (C'ombine) as shown in Figure below.

The following procedure mergesort sorts a sequence a from index 10 to index hi.

```
void mergesort ( int 10, int hi )

    if (10<hi)
```

```
 ms  (Io+hi)  /  2  ;
mergesort  (10,  m)  ;
mergesort  (m+l,  hi)
;  merge  (10,  m,  hi)
;
```

in the middle between 10 and hi is determined. Then the first of the (from 10 to m) and the second pan (from m+l to hi) are sorted by recursive calls

Then the two soiled halves are merged by procedure merge. Recursion

auxiliary ctive next-greatest array b. Then element the at two each halves time is are copied scanned back by to array pointers a . i and j and the

```
resped merge (int 10, int m,

int hi) { int i, j , k;

  // copy both halves of a to auxiliary array
  b for (i = 10 ; i<=hi; i++)
```

element at each time

```
    b[i]=a[i];

i=lo; j=m+1; k=lo;
// copy back next-greatest
while (i<=m && j<=hi)
    if (b[i]<=b[j])
        a[k++]=b[i++];
    else
        a[k++]=b[j++];

// copy back remaining
while (i<=m)
    a[k++]=b[i++];
}
```

elements of first half (if any)

MergeSort has the average case tin
and there is no worst-case scenario.
as $O(n^2)$. Hence MergeSort is bette
Sorting of array 100, 90, 80, 70, 60

better

MergeS01t has the average case time of O(nlog(n)). Moreover, Mergesort is a stable sort, and there is no worst-case scenario. Bubble sort as the average and worst case complexity

than Bubble sort in most of the cases.
80, 70, 60, 50, 40, 30, 20

Step l: the array is divided into 2 parts 50,recursively40, 30, 20

100, 90, 80, 70 60, 40, 30,20

100.90 80,70 60,50 30,20

40

IOO 90 80 70 60 50 30 20

IOO 90 80 70 60 50 40

100 90 80 70 60 50 40 Step 2: the arrays are now merged20,30

90, 1 00 70, 80 50, 60

40 5 0 60,

70.80.90100

20,30,40,50,60,70,80,90I OO - final sorted array

Complexity of MergeSort

If the running time of merge sort for a list of length n is T(n), then the recurrence T(n) = 2T(n/2) + cn where c is a constant. This follows from the definition of the algorithm (apply the algorithm to two lists of half the size of the original list, and add the n steps taken to merge the resulting two lists).

$4.\left[2.T\left(\frac{n}{8}\right)+c.\frac{n}{4}\right.$ (4) $8.T\left(\frac{n}{8}\right)+3.cn$

$= 2.T$ +cn=2. 2.T - +c.- +cn=4.T - +2.cn=4.

2.T + $\left(\frac{n}{2^k}\right)$ - 2.cn

$= 2^1$ .7'+ k.cn

$\left(\frac{n}{2^{\lg n}}\right)$

- 21gnT+ $= n \Rightarrow k$ Ign.cn(2^klog2 n)

=

= n.T(l) + cn. lg(n)

$= \Theta(n.\lg n)$

8. a) Radix Sort the following list:

Write 189, the 205, Radix 986, sort 421, algorithm.97, 192, 535, 839, 562, 674 WBIJT 2012]

b) Find the time complexity of Binary Search Algorithm.

& ALCORN

INPUT	1 st pass	2nd pass	3rd pass (sorted)
189	421	205	
205	192	421	097
986	562	535	189
421	674	939	192
097	205	562	205
192	535	674	421
535	986	986	535
			562
839	097	189	674
562	189	192	
			939
674	939		
		097	986

Suppose we have a list of n records each with a key that's a number from I to k (we gneralize the problem a little so k is not necessarily equal to n).

We can solve this by making an array of linked lists. we move each input record into the list in the appropriate position of the array then concatenate all the lists together in order. bucket sort (L)

```
list Ytk+l] for (i = 0; i <= k; i++) Y[i] =
empty while L nonempty

    let X = first record in L move X to Y [key
    (X) ]

for (i = 0; iconcatenate Y [i] onto end of L
```

What to do when k is large? Think about the decimal representation of a number

x = a + 10 b + 100 c + 1000 d + . . .where a,b,c etc all in range 0..9. These digits are easily small enough to do bucket sort.

radix sort (L) :

bucket sort by a bucket sort by
b bucket sort by c

more simply radix sort (L) :

while (some key is nonzero)

bucket sort (keys mod 10)

```
keys = keys / 10
```

In radix sorting, the last pass of bucket sorting is the one with the most effect on the overall order. So we want it to be the one using the most important digits. The previous bucket soning passes are used only to take care of the case in which two items have the same key (mod 10) on the last pass.

b) The time complexity of Binary Search is as follows:
In each iteration, the array is split into two halves. Thereby we can say that the binary search takes the form of a binary tree. The time complexity is thus O(log2 n) in worst case also.

9. When will interpolation search be more appropriate than binary search? How does an interpolation search work? Write an algorithm for interpolation search. Show with an appropriate example that worst case time complexity of interpolation search is ()(n) . What is the average case time complexity of interpolation search? [WBUT 20131

Answer:

Interpolation search works better than Binary Search for a sorted and uniformly distri buted array.

Interpolation search is an algorithm for searching for a given key value in an indexed array that has been ordered by the values of the key. It parallels how humans search through a telephone book for a particular name, the key value which the book's entries are ordered. In each search step it calculates where in the remaining search space the sought item might be, based on the key values at the bounds of the search space and the value of the sought key, usually via a linear interpolation. The key value actually found at this estimated position is then compared to the key value being sought. If it is not equal, then depending on the comparison, the remaining search space is reduced to the part before or after the estimated position. This method will only work if calculations on the size of differences between key values are sensible.

Algorithm: INTERPOLATON SEARCH

/* Given: X, and al < ... < an

Return: i such that x = co

0 if no such i exists */

i := 1

j := n

LO := a_i

HI := a_j

if x < LO then return 0 if x 2 HI then i :=j
loop invariant: x 2 LO and x HI while (i<j)
do m floor(i + j-i)*(x-LO)/ (HI-LO))

```
    MID) then
    >
if(s
    1-0 :- MID else
if(s < MID) then
    Ill MID clse
    return MID
if(s a,) then i := 0
return i
```

On average the interpolation search makes about log(log(n)) comparisons (if the elements are uniforn11y distributed), where n is the number of elements to be searched. In the worst case (for instance where the numerical values of the keys increase exponentially) it can make up to O(n) comparisons (e.g., 1,2,...,999, searching for 1000 in 1 000, I OA9 will take 1000 accesses).

10, a) Define sorting.

b) What is a stable sorting? What is In-Place sorting? CWBUT 20141

c) Write the Pseudocode for Merge sort implementation. What is its time complexity?

d) If the existing array is sorted and you want to insert a new element in the list without disrupting the sortedness then which sorting technique you should use? Why?

Answer:

a) Sorting is a process by which a collection of items is placed into which is typically based on a data field called a key. Sorting refers to ordering data in an increasing or decreasing fashion according .to some linear relationship among the data items.

b) Stable sort:

Ilhen sorting some kind of data, only part of the data is examined when determining the sort order. The data being sorted can be represented as a record or tuple of values, and the part of the data that is used for sorting is called the key. A sorting algorithm is stable if uhenever there are two records R and S with the same key, and R appears before S in the original list, then R will always appear before S in the sorted list.

When equal elements are indistinguishable. such as with integers, or more generally. any data where the entire element is the key, stability is not an issue. Stability is also not an issue if all keys are different. In-place sorting:

A sort algorithm in which the sorted items occupy the same storage as the original ones. These algorithms Iliay use O(n) additional memory for bookkeeping, but at most a constant number of items are kept in auxiliary memory at any time.

c) Part: Refer to Question No. I(a) of Long Answer Type Questions.

2nd Part:

ity of merge sort:

Merv at every splits step. the Inerge-sofl considers only one array. In the next step, the algorCitChl)1 the array into halves and then sorts and Inerges them. In the kth iteration the algorithtn splits the arrays into sub-arrays, which are 2 the nun)ber of steps required to break the array into sub-arrays of single elements is

is O(nlog.'n) (in best. average as well as in worst case also). A drawback of mergesort is that it needs an additional space of for the tenlporary array h. There are different possibilities to ilnplement function nwrge. The most efficient of these is variant b. It requires only half as tnuch additional space, it is faster than the other variants, and it is stable.

d) Insertion Sort as its best-case time connplexity is when the array is already sorted which is O(n).

11. What is hashing? Describe any three methods of defining a hash function.

[WBUT 2015, 2017]

Ans» er:

rt part: Refer to Question No. 2 ofShort Answer Type Questions.

2 nd part:

A hash function maps keys to small integers (buckets). An ideal hash function maps the keys to the integers in a random-like nnanner, so that bucket values are evenly distributed even if there are regularities in the input data. This process can be divided into two steps:

- Map the key to an integer.
- Map the integer to a bucket.

The following functions map a single integer key (k) to a snnall integer bucket value h(k).

m is the size of the hash table (number of buckets).

Division method (Cormen) Choose a prime that isn't close to a power of 2. h(k) = k mod m. Works badly for many types of patterns in the input data.

Knuth Variant on Division h(k) = k(k+3) mod m. Supposedly works much better than the raw division Inethod.

Multiplication Method (Cormen). Choose m to be a power of 2. Let A be sonle random looking real number. Knuth suggests M = 0.5*(sqrt(5) - l). Then do the following:

x = fractional part of s . h(k) = floor(m*x)

&

write a C function to sort positive integers that does not compose the array 12.

elements.

[WBUT 2016]

```
\ include < s tdio . h>
€ include < std Iib . h>
structure for a node * /

*truct Node

int data; struct Node *next;

Function to insert a node at the begining of a linked I sit * / insertAtTheBegin (struct Node **
Function 'to bubble sort the given startlinked _ref, 1 sit int * /data) ; bubbleSort (struct
Node *start) ;
void
F Function to swap data of two nodes a and b* / void swap
(struct Node *a, struct Node *b) •
Function .to print nodes in a given linked list * / void printList (struct Node
*start) ;

ant main ( )

int arr[ ] {12 , 56, 2 , 11 , 1 , 90) ; int
list_size, i;

/ * start with empty linked list * / struct Node *start =
NULL;

/ * Create linked list from the array arr[] .
Created linked list will be 1->11->2->56->12 * / for (i = 0;
insertAtTheBegin (&start, arr [i] ) ;

/ * print list before sorting * / print f ( " \ n Linked list before
sorting " ) ;printList (start) ;

/ * sort the linked list * / bubbleSort ( start) ;
```

```
/ * print list after sorting * / print f ( " Linked list after
sorting " ) ; printList (start) ;

getchar ( ) ; return 0;
```

FUnction to insert a node at the * *start_ref, begining of int a linked data) 1 sit * / void insertAtTheBegin (struct Node

```
{                                                                    ;
    struct Node *ptrl (struct Node* ) mal loc (sizeof (struct
    Node) ) ptrl->data = data; ptrl ->next * start_ref ;*start_ref
    = ptrl;

 / * Function to print nodes in a given linked list * /
void printList (struct Node *start)

    struct Node *temp = start;
    print f ( " " ) ; while
    (temp ! =NULL) .

        print f ( " temp—>data) ; temp
        = temp- >next;


/ * Bubble sort the given linked 1 sit * /
void bubbleSort (struct Node *start)

    int swapped, i; struct
    Node *ptrl ; struct Node
    *Iptr = NULL;

    / * Checking for empty list *
    / if (ptrl NULL) return ; do

        swappedo; ptrl    start ; while (ptrl-
        >next ! Iptr) if (ptrl->data > ptrl-
        >next->data)

              swap    (ptrl,    ptrl-
           >next) ; swapped = 1; ptrl
           = ptrl->next;

       Iptr   =   ptrl;

   while (swapped) ;

/ * function to swap data of two nodes a and b* /
void swap (struct Node *a, struct Node *b)
                                                    STRUCT RE
. int temp = a—>data;
```

```
> data = b—
>data; da ta t
emp ;
```

b) 43. a) Write an algorithm of the complexity for linear search.of your algorithm. CWBUT 2017] Give an outline snswer:

```
Vinear 1: Search Set i to (Array 1 A, Value x)

  Step 2 : if i > n then go to step 7

stepStep 4: 3 : Set if A i [i)to i x + then 1     go to step 6

 Step 5: Go to Step 2

  Step 6: print Element x Found at index i and go to step 8
  Step 7: print element not
      found
  Step 8: Exit
  Step
```

b) Linear search executes in O(n) time where n is the number of elements in the array.

obviously, the best case of linear search is when VAL is equal to the first element of the array. In this case, only one comparison will be made.

Likewise, the worst case will happen when either VAL is not present in the array or it is equal to the last element of the array. In both the cases, n comparisons will have to be made.

14. Find the time complexity of merge sort technique using the recurrence relation assuming the size of the list n = 2k . [WBUT 20171 Answer:

Refer to Question No. 7 of Long Answer Type Questions.

15. Write short notes on the following:

a) Radix sort [WBUT 2010, 20141

b) Merge Sort [WBUT 20121

c) Interpolation search [WBUT 20141

d) Heap CWBUT 2015, 20181

Answer:

a) Radix sort:

Radix sorting is a technique for ordering a list of positive integer values. The values are Sliccessively ordered on digit positions, from right to left. This is accomplished by copying the values into "buckets." where the index for the bucket is given by the position Of the digit being sorted. Once all digit positions have been examined, the list must be sorted.

The following table shows the sequences of values found in each bucket during the four steps involved in sorting the list 624 852 426 987 269 146 415 301 730 78 593. During Pass I the ones place digits are ordered. During pass 2 the tens place digits are ordered.

retaining the relative positions of values set by the earlier pass. On pass 3 the hundreds place digits are ordered, again retaining the previous relative ordering. After three passes the result is an ordered list.

bucket	ass I	ass 2	ass 3
	73 0	3 0 1	0 78
	30 1	4 1 5	| 46
2	85 2	62 4, 4 2 6	2 69
3	59 3	7 3 0	3 01
4	62 4	1 46	4 15, 4 26
5	41 5	8 5 2	5 93
6	42 6 , 14 6	2 6 9	6 24
7	98 7	7 8	7 30
8	7 8	9 8 7	8 52
9	26 9	5 9 3	9 87

The C-program of the radix-sort is as shown below:

```
#       include <stdio.h>
#       include <conio .h>
#define N 11
#define SHOWPASS

void radixsort (int [ ] , int) ; void print (int [J , int) ;

int main (void)

 int unsortedArr [N] = {624, 852, 426, 987, 269 , 146 , 415 , 301, 730, 78, 593} ; int i; print f ( " \nBefore Sorting: \n\n") ; for (i = 0;print f ( " %5d", unsortedArr [i] ) ; radixsort (unsortedArr, N) ; printf ( " \n\nAfter Sorting: " ) ; for (i = 0;print f ( " %5d" , unsortedArr [i] ) ; getch() ; return 0;

void radixsort (int a [ ] , int n)

 int i, b [N] , m = 0, exp = 1;

 int count o ; for
 (i= 0;
```

```
    if (a [i] >
      m) m= a

while (m / exp > 0)
```

```
    { count ++ ; bucket ( 10
] ={0};    int
        (i = 0;      forbucket
(a [i] / exp  10] +4;
        (i 1; i < 10; i + + )
     forbucket [i] += bucket [i
     - 11:(i = n -for
    b [ - -bucket (a [i] / exp
              ' % 101 = a[:(i = 0:for

  exp -

    #ifdef SHOWPASS printf ( "
    \nPASS%d : ",count) ; print (a,
    n) ;
    # endi f


 void print (int a [ ] , int n)

  int 1, printf ( " \ n"
  ) ; for (i = 0;printf (
  " %5d" , a [ i] ) ;


 /* Output of the above program
 Beforg Sorting:
624 852 426 987 269415 301 730 78 593
 PASS 1 •
 730 301 852 593 624 415 426 146 987 78 269
 PASS 2:

 301 415 624 426 730 146 852 269 78 987
 593 USS 3 .
 78 146 269 301 415 426 593 624 730

 852 987 After Sorting:

78 146 269 301 415593 624 730 852 987
```

Time complexity of radix sort:
The time complex'ity of the algorithm is as follows: Suppose that the n input numbers have maximum k digits. Then the Counting Sort procedure is called a total of k times. Counting Sort is a linear, or O(n) algorithm. So the entire Radix Sort procedure takes O(n) time.

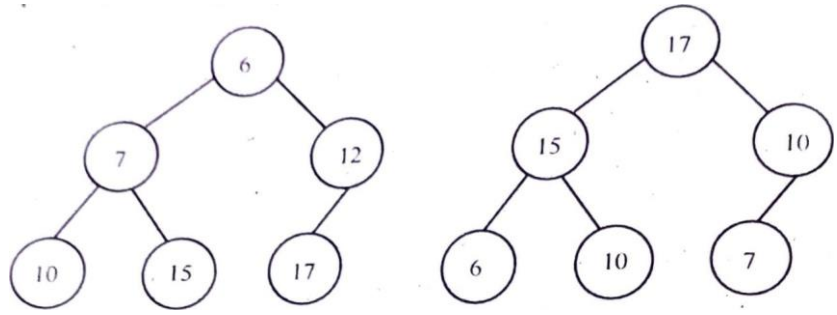
b) Merge Sort: Refer to Question NO. / of Long Answer Type Questions.

c) Interpolation search: Refer to Question No. 9 of Long Answer Type Questio

d) Heap Sort:

A binary heap is a cornpletc binaty tree which satisfies the heap ordering property ordering can be one of two tvpes:

- the min-heap propcrt) t: the value of each node is greater than or equal to value of its parent. ith the tninitnurn-value element at the root.
- the ma.v-heap property: the value of each node is less than or equal to the of its parent. ith the rna,xirnunl-value element at the root.



In a heap the highest (or lowest) priority element is always stored at the root, hence the name "heap". A -heap is not a sorted structure and can be regarded as partially ordered. As it can be seen from the picture, there is no particular relationship among nodes on an} given level, even among the siblings.

Since a heap is a complete binary tree, it has a smallest possible height - a heap with N nodes always has O(log N) height.

lowest) A heap priority. is useful A data common structure use of when a heap we is need to implement to remove a the priority object queue.with the highest (orl

MISCELLANEOUS

Short Answer uestions Type

a) Describe a string reversal algorithm. [WBUT 2012, 2015]

b) What is difference between Union & Structure? [WBUT 2012] Ans code is given below:

```
g) The' d reverse (char
 *str) { 1* end = str;
 char char tmp ;

   if (str) { while
      ( *end) {
          + + end ;

        - -end; while (str < end) { tmp
          *str;
        *str++  * end ;
        - end-- tmp ;
```

In the above code, in the innermost while loop in each iteration, the characters pointed to by str and end get swapped, str gets incremented to point to the next character, and end is decremented to point to the previous one.

b) The differences between structure and union:

1. Union allocates the memory equal to the maximum memory required by the meniber of the union but structure allocates the memory equal to the total memory required by the members.

2. In union, one block is used by all the member of the union but in case of structure, each member has their own memory space.

While structure enables us treat a number of different variables stored at different in memory, a union enables us to treat the same space in memory as a number of different variables. That is a Union offers a way for a section of memory to be treated as a variable Of one type

on one occasion and as a different variable of a different type on another

Occasion. There is frequent requirement while interacting with hardware to access a byte or group Of bytes simultaneously and sometimes each byte individually. Usually union is the answer.

Long Answer Type Questions

I. a) Write an algorithm to solve the Tower of Hanoi problem. [WBUT Also calculate 2007, 2013)the time complexity of your algorithm.

OR,

Write the Pseudocode or C code to implement Tower of Hanoi problem. LWBUT Also 20141fin d the complexity of your procedure.

Answer:

```
Tower of Hanoi (n, source, auxiliary, destination)

       If n=l move disk from source to destination; (base case)
       Else,

                  Tower of Hanoi (top n—l, from, to, using) ;
                                                          ;
                   Move the nth disk from from' to to'
                  Tower of Hanoi (n—l, using, from, to) ,
```

First recursive call moves n-l disks from 'from' to 'using' using 'to'. SO after that call nI disks are in 'using ' peg int order of size and the 'from' peg contains the nth disk i.e., the largest one. So, now move that glisk from 'from' peg to 'to' peg. Then again by the 2nd recursive call move n-l disk from 'using' peg to 'to' peg using 'from' peg. So, after all these, 'to' peg contains all disks in order of size.

Let the time required for n disks is T(n). There are 2 recursive call for n-l disks and one constant time operation to move a disk from 'from' peg to 'to' peg . Let it be kl .Therefore,

$T(n) = 2\,T(n-l) + kl$

T(O) = k2, a constant.

$T(1) = 2\,k_2 + k_1$

$T(2) = 4\,k_2 + 2k_1 + k_1$

T(2) = 8 k2+ 4k! + 2kI + kl

Coefficient of kl =2')

Coefficient of $=2^{n}$- I

Time complexity is 0(211) or O(a'l) where a is a constant greater than l .

So it has exponential time complex'ity.

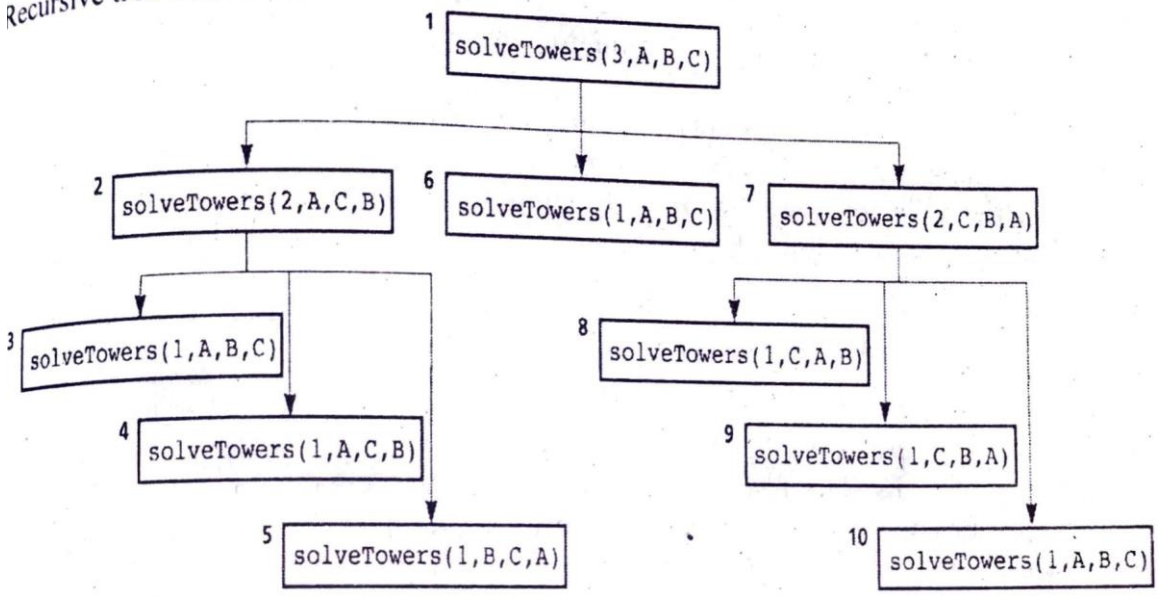
b) Write the recursive function for the Tower of Hanoi problem. Also draw the recursion tree for any set of initial values.

[WBUT 2007, 20101 Write a recursive algorithm to solve tower OR,of Hanoi problem.

D 144

```
soswer.fünction * / /*.. solveTowers (int count, char
source v„oJchar destination, char spare) (
            1 ) {
  'f (count
  ..rf("Move top disk from pole" + source +
                    'to pole" + destination) ;

e
solveTowers (count-I, source, spare, destination) ; // X
solveTowers (1, source, destination, spare) • solveTowers
                                          // Y
(count-I, spare, destination, source) ; // Z// end if

} // end solveTowers
```

Recursive tree with n =3.



2. Write short notes on the following:

a) Index sequential file ordering [WBUT 2010, 20141

b) Tail recursion [WBUT 20181

Answer:

a) Building Indexed-Sequential Files:

The additional entries required on the File specification for an output Indexed-Sequential file (as compared to an ordinary sequential output file) are: Length of Key Field (columns 29-30)

Record Address Type (column 31) should contain K to specify that records are accessed with record keys

Type Of File Organization (column 32)-should contain I to specify Indexed-Sequential organization

The records written to the Indexed-Sequential tile being built must be written in ascending key sequence. If an attempt is made to write a record with a key equal to or less than the key of the record previously written to the file, HO will be set on and the program will terminate abnorlnally.

Processing Indexed-Sequential Files Sequentially

Phere are no additional entries required beyond those listed above (for building Indexed_ Sequential tiles) in order to process an Indexed-Sequential file in its entirety. A sequentially' processed tile nvay be used as input (I in column 15 on the File specification) or update (U in coltill)ll 15). If processed as an update file, records from the file may be updated at either detail or total tinne. The record available for updating is the record read on the prev IOUS input cycle.

Procesqno Indexed-Sequcntial Files by Chaining

Fronl one to nine Indexed-Sequential files may be processed by the use of record key fields specified as chaining fields. A data field in an input record is designated as a chaining field b} placing the chaining field identifier (C l, C2, C 3, ... C 9 in columns 61 _ 62 of the Input specification). Each chaining field must also be included on an Extension

specification which functions to link the chaining file to the chained file.

During the input cycle, the record key(s) contained in one or more chaining fields are used to retrieve the corresponding record(s) from the Indexed-Sequential file specified on the Extension specification. Chaining is the only circumstance in which more than one record identifying indicator may be on during a single cycle.

b) Tail recursion:

A function call is said to be tail recursion if there is nothing to do after the function returns except return its value. Since the current recursive instance is done executing at that point, saving its stalk frame is a waste. Specifically, creating a new stack frame on top of the current, finished, frame is a waste. A compiler is said to implement Tail recursion if it recognizes this case and replaces the caller in place with the cal lee, so the instead of nesting the stack deeper, the current stack frame is reused. This is equivalent in effect to a "GOTO", and lets a programmer write recursive definitions without worrying about space inefficiency during execution. Tail recursion is as efficient as iteration.

DSA-146