K.S.R. COLLEGE OF ENGINEERING (Autonomous)

Vision of the Institution

- We envision to achieve status as an excellent educational institution in the global knowledge hub, making self-learners, experts, ethical and responsible engineers, technologists, scientists, managers, administrators and entrepreneurs who will significantly contribute to research and environment friendly sustainable growth of the nation and the world.

Mission of the Institution

- To inculcate in the students self-learning abilities that enable them to become competitive and considerate engineers, technologists, scientists, managers, administrators and entrepreneurs by diligently imparting the best of education, nurturing environmental and social needs.
- To foster and maintain a mutually beneficial partnership with global industries and Institutions through knowledge sharing, collaborative research and innovation.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Vision of the Department

- To create ever green professionals for software industry, academicians for knowledge cultivation and researchers for contemporary society modernization.

Mission of the Department

- To produce proficient design, code and system engineers for software development.
- To keep updated contemporary technology and fore coming challenges for welfare of the society.

Programme Educational Objectives (PEOs)

PEO1: Figure out, formulate, analyze typical problems and develop effective solutions by imparting the idea and principles of science, mathematics, engineering fundamentals and computing.

PEO2: Competent professionally and successful in their chosen career through life-long learning.

PEO3: Excel individually or as member of a team in carrying out projects and exhibit social needs and follow professional ethics.

K.S.R. COLLEGE OF ENGINEERING (Autonomous)

Department of Computer Science and Engineering

Subject Name: Data Structures
Subject Code: 18CS311 **Year/Semester: II / III**

Course Outcomes: On completion of this course, the student will be able to

CO1: Construct the different linear data structure to solve simple problems.

CO2: Build the binary, binary search tree with its operations.

CO3: Analyze the concept of AVL tree, splay tree, B tree and B+ tree.

CO4: Apply shortest path and minimum spanning tree algorithms to solve real time problems.

CO5: Evaluate various sorting and searching techniques.

Program Outcomes (POs) and Program Specific Outcomes (PSOs)

A. Program Outcomes (PO)

B. Engineering Graduates will be able to :

PO1 **Engineering knowledge:** Ability to exhibit the knowledge of mathematics, science, engineering fundamentals and programming skills to solve problems in computer science.

PO2 **Problem analysis:** Talent to identify, formulate, analyze and solve complex engineering problems with the knowledge of computer science. .

PO3 **Design/development of solutions:** Capability to design, implement, and evaluate a computer based system, process, component or program to meet desired needs.

PO4 **Conduct investigations of complex problems:** Potential to conduct investigation of complex problems by methods that include appropriate experiments, analysis and synthesis of information in order to reach valid conclusions.

PO5 **Modern tool Usage:** Ability to create, select, and apply appropriate techniques, resources and modern engineering tools to solve complex engineering problems.

PO6 **The engineer and society:** Skill to acquire the broad education necessary to understand the impact of engineering solutions on a global economic, environmental, social, political, ethical, health and safety.

PO7 **Environmental and sustainability:** Ability to understand the impact of the professional engineering solutions in societal and Environmental contexts and demonstrate the knowledge of, and need for sustainable development.

PO8 **Ethics:** Apply ethical principles and commit to professional ethics and responsibility and norms of the engineering practices.

PO9 **Individual and team work:** Ability to function individually as well as on multi-disciplinary teams.

PO10 **Communication:** Ability to communicate effectively in both verbal and written mode to excel in the career.

PO11 **Project management and finance:** Ability to integrate the knowledge of engineering and management principles to work as a member and leader in a team on diverse projects.

PO12 **Life-long learning:** Ability to recognize the need of technological change by independent and life-long learning.

B. Program Specific Outcomes (PSOs)

PSO1 Develop and Implement computer solutions that accomplish goals to the industry, government or research by exploring new technologies.

PSO2 Grow intellectually and professionally in the chosen field.

K.S.R COLLEGE OF ENGINEERING, TIRUCHENGODE – 637 215
(Autonomous)
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

18CS311 - DATA STRUCTURES

TWO MARKS QUESTION AND ANSWER
UNIT I - Linear Structures (CO1)

PART-A

1. What is called as a Data Structure? (Remembering)

A Data Structure defines a way of representing or organizing all data that contains both the data items and their relationship with each other.

2. Differentiate between data type and data structures.(Analyzing)

Data Type: Data Type of a variable is a set of values the variable may assume.

Eg. int, char & real

Data Structures: Data structure means the way of organizing data values with the help of existing data types.

Eg. Array, stack queue & tree.

3. Name the Primary Data Structures? **(Remembering)**

- Integers
- Floating-point numbers.
- Character Constants

4. What are the types of Secondary Data Structures? (Remembering)

- Static Data structures
- Dynamic Data structures.

5. List out the static data structures? (Understanding)

- Arrays
- Structures

6. State the types of dynamic data structures? (Understanding)

- Linear Data Structures
- Non-Linear Data Structures.

7. Give examples of Linear and Non-Linear Data Structures.(Remembering)

- Linear Data Structures
    - i) Linked Lists(Singly & Doubly Linked Lists)
    - ii) CircularLinkedLists(Circular-singly&Circular-doubly Linked Lists).
- Non-Linear Data Structures
    - i) Trees

ii) Graphs.

8. Explain Abstract Data Types? (Understanding)

Classes encapsulate all the essential properties of the objects that are to be created. Since the classes use the concept of data abstraction, they are known as Abstract Data Types (ADT).

9. List the operation of set ADT? (Understanding)

Union and Find are the two operations on set ADT

10. Define Stack. (Remembering)

Stack is a list, in which insertion and deletion of elements are made at one end called top. Push and Pop are two important operations performed on stacks.

Push: The incoming element is placed on top of the stack.

Pop: Removes the topmost element from the stack.

11. Define Queue.(Remembering)

Queue is a list in which insertion takes place at one end, called the **rear** and the deletion at the other end, called the **front**.

In queue, the item inserted first is removed first. Therefore it is called a **First in First out (FIFO)** structure.

12. Define Circular Queue. (Remembering)

In array based implementation of queue, inefficiency arises only when the value of the rear pointer exceeds maximum size during insertion. This can be rectified if we logically consider the queue to be circular.

In circular queue, once the rear pointer reaches the maximum size, the next location I the first one (index 0), provided the first location is vacant. The specific location is calculated by mod (%) operator.

13. Build some applications of stack.(Applying)

- Evaluation of Expressions.
- Expression conversion.
- Balancing of symbols.

Function Calls.

14. Define Dequeue. (Remembering)

A dequeue is an ordered list in which additions and deletions may be carried out at either end.

15. Distinguish between stack and queue.(Evaluating)

STACK	QUEUE
Insertion and deletion are made at one end.	Insertion at one end rear and deletion at other end front.
The element inserted last would be removed first. So LIFO structure.	The element inserted first would be removed first. So FIFO structure.
Full stack condition: If(top==Maxsize) Physically and Logically full stack	Full stack condition: If(rear = = Maxsize) Logically full. Physically may or may not be full.

16. State the difference between cursor and pointers.(Remembering)

Pointer: Pointer is a variable, which stores the address of another variable.
Using pointers, memory can be allocated dynamically.
Cursor: Cursor is a temporary memory space. Memory allocated is static.

17. What is an ordered list. Remembering)

An ordered list is a linear list which is been ordered by some key.

Eg. An alphabetized list of students in a class, a list of exam scores in decreasing order.

18. State the difference between array and linked list. ((Understanding)

Array	Linked List
Contiguous memory location	Memory location need not be necessarily contiguous
Operations such as insertion and deletion are ineffective since the memory locations need to be moved up and down respectively.	Insertion and deletions are easier and needs only one pointer assignment.
Inefficient use of memory space.	A small amount of memory is been wasted for storing a pointer, which is been associated with each node.

19. Give the applications of linked list.(Understanding)

Applications of linked list are:

Bin Sort, radix sort, convex Hull, Offline Equivalence classes, Polynomial Manipulation.

20. What are the advantages and disadvantages of linked list?(Remembering)

Advantages:

a. Memory location for a linked list is dynamic, that is memory allocation is done during run time. So memory will not be wasted.

b. Data movements during insertion and deletion will be eliminated. So time will not be wasted.

c. Memory allocation for each node of a linked list need not be continuous.

Disadvantages:

d. Nodes of a linked list cannot be accessed directly. To access a particular node accessing should start only from the beginning.

e. To store a single data, along with data, memory must be allocated for a pointer also, which wastes memory.

21. Convert the infix (a+b)*(c+d)/f into postfix & prefix expression. (Creating)

Postfix : a b + c d + * f /
Prefix : / * + a b + c d f

22. List the steps required to evaluate the postfix expression. (Analyzing)

- Repeatedly read characters from postfix expression

- If the character read is an operand, push the value associated with it into the stack
- If it is an operator, pop the top two values from stack, apply the operator to them And push the result back onto the stack.

23. State the different ways of representing expressions. (Analyzing)

The different ways of representing expressions are
• Infix Notation
• Prefix Notation
• Postfix Notation

24. State the advantages of using infix notations (Analyzing)

- It is the mathematical way of representing the expression
- It is easier to see visually which operation is done from first to last

25. What is the advantages of using postfix notations. (Remembering)

- Need not worry about the rules of precedence
- Need not worry about the rules for right to left associativity
- Need not need parenthesis to override the above rules

26. State the rules to be followed during infix to postfix conversions. (Remembering)

- Fully parenthesize the expression starting from left to right. During parenthesizing, the operators having higher precedence are first parenthesized
- Move the operators one by one to their right, such that each operator replaces their corresponding right parenthesis
- The part of the expression, which has been converted into postfix is to be treated as single operand

27. State the rules to be followed during infix to prefix conversions. (Remembering)

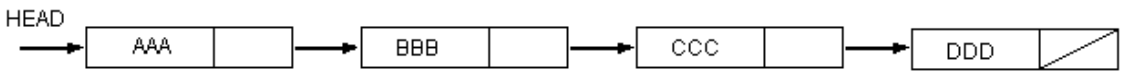
- Fully parenthesize the expression starting from left to right. During parenthesizing, the operators having higher precedence are first parenthesized
  Move the operators one by one to their left, such that each operator replaces their corresponding left parenthesis
- The part of the expression, which has been converted into prefix is to be treated as single operand
- Once the expression is converted into prefix form, remove all parenthesis

28. Difference between Abstract Data Type, Data Type and Data Structure.(Applying)

- An Abstract data type is the specification of the data type which specifies the logical and mathematical model of the data type.
- A data type is the implementation of an abstract data type.
- Data structure refers to the collection of computer variables that are connected in some specific manner. i.e) Data type has its root in the abstract data type and a data structure comprises a set of computer variables of same or different data types

29. Define Singly Linked List.(Understanding)

A singly linked list is a list structure in which each node contains a single pointer field that points to the next node in the list, along with a data field.

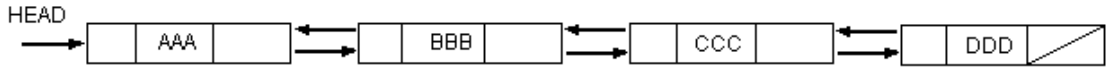


30. Define Doubly Linked List.(Understanding)

A doubly linked list is a list structure in which each node contains two pointer fields along with a data field namely,

BLINK – Points to the previous node in the list

FLINK – Points to the successive node in the list



PART-B

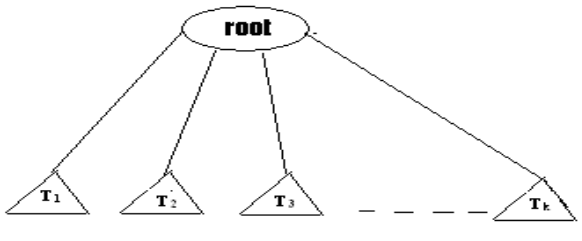
1. Write a program to print out the elements of a singly linked list.(Creating)
2. Given two sorted lists, L1 and L2, write a procedure to compute L1 L2 using only the basic list operation.(Evaluating)
3. Write a program to evaluate a postfix expression (Creating)
4. Write a program to convert postfix expression in to infix. (Creating)
5. Write a routine to implement queues using linked list (Creating)
6. Write a routine to implement queues using Array. (Creating)
7. Explain the procedure to insert a new node in the
   (a) Beginning
   (b) End of the list (Remembering)
8. Write an algorithm for the following in doubly linked list.(Analyzing)
   (i) Inserting a node to the left.
   (ii) Deleting a node.
9. Explain the cursor implementation of linked list. (Remembering)
10. Explain the following algorithm of a circular linked list. (Understanding)
    (a) Inset the node at the beginning
    (b) Delete a node from beginning
11. Write a routine to implement stack using
    (a) Array
    (b) Linked list (Creating)
12. Write a routine to implement polynomial ADT. (Creating)

UNIT II - Tree Structures (CO2)

PART-A

1. Define Tree.(Remembering)

A Tree is a collection of one or more nodes with a distinct node called the root , while remaining nodes are partitioned as T_1 ,T_2, ..,T_k , $K \geq 0$ each of which are sub trees, the edges of T_1,T_2,…,T_k are connected the root.



2. Give some applications of Trees.(Analyzing)

- Implementing the file system of several operating systems.
- Evaluation of arithmetic expression.
- Set representation.
- Gaming/Decision making problems.

3. Define node, degree, siblings, depth/height, level.(Understanding)

Node: A node is an item of information with branches to other items.
Degree: The number of subtrees of a node is called is degree.
Siblings: The children of the same parent is said to be siblings.
Level: The level of a node is defined recursively by assuming the level of the root to be one and if a node is at level l, then its children at level l+1.
Depth/Height: The depth/height of a tree is defined to be the level of a node which is maximum.

4. What is a path in a tree. (Remembering)

A path in a tree is a sequence of distinct nodes in which successive nodes are connected by edges in the tree.

5. Define terminal nodes in a tree. (Remembering)

A node which has no children is called a terminal node. It is also referred as a leaf node. These nodes have a degree as zero.

6. Define nonterminal nodes in a tree(Remembering)

All intermediate nodes that traverse the given tree from its root node to the terminal nodes are referred as terminal nodes.

7. Define a Binary Tree. (Remembering)

A Binary Tree is a tree, which has nodes either empty or not more than two child nodes, each of which may be a leaf node.

8. Define a full binary tree. (Remembering)

A full binary tree, is a tree in which all the leaves are on the same level and every non-leaf node has exactly two children.

9. Define a complete binary tree. (Remembering)

A complete binary tree is a tree in which every non-leaf node has exactly two children not necessarily to be on the same level.

10. Define a right-skewed binary tree. (Remembering)

A right-skewed binary tree is a tree, which has only right child nodes.

11. State the properties of a Binary Tree. (Understanding)

- Maximum No. of nodes on level n of a binary tree is 2^(n-1),where n>=1.
- Maximum No. of nodes in a Binary tree of height is 2^(n-1),where n>=1.
- For any non-empty tree,nl=nd+1 where nl is the number of leaf nodes and nd is the no. of nodes of degree 2.

12. What are the different ways of representing a Binary Tree? (Understanding)

- Linear Representation using Arrays.
- Linked Representation using Pointers.

13. State the merits of linear representation of binary trees. (Understanding)

- A store method is easy and can be easily implemented in arrays.
- When the location of the parent/child node is known, other one can be determined easily.
- It requires static memory allocation so it is easily implemented in all programming languages.

14. State the DISADVANTAGES of linear representation of binary trees. (Applying)

- Insertions and deletions are tougher.
- Processing consumes excess of time.
- Slow data movements up and down the array.

15. Define Traversal. (Analyzing)

Traversal is an operation which can be performed on a binary tree is visiting all the nodes exactly once.

Inorder: traversing the LST, visiting the root and finally traversing the RST.
Preorder: visiting root, traversing LST and finally traversing RST.
Post- order: traversing LST, then RST and finally visiting root.

16. What are the tasks performed while traversing a binary tree? (Understanding)

Visiting a node

- Traverse the left structure
- Traverse the right structure.

17. What are the tasks performed during preorder traversal? (Understanding)

- Process the root node
- Traverse the left subtree
- Traverse the right subtree.

Ex : +AB

18. Identify the tasks performed during inorder traversal?(Applying)

- Traverse the left subtree
- Process the root node
- Traverse the right subtree.

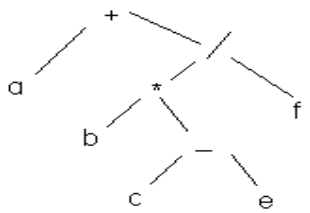
Ex : A+B

19. Identify the tasks performed during postorder traversal? (Applying)

- Traverse the left subtree
- Traverse the right subtree.
- Process the root node.

Ex : AB+

20. Give the pre & postfix form of the expression (a + ((b*(c-e))/f) (Evaluating)



21. Define a Binary Search Tree. (Understanding)

A Binary Search Tree is a special binary tree, which is either empty or if it is empty it should satisfy the conditions given below:

- Every node has a value and no two nodes should have the same value(Values should be distinct).
- The value in any left subtree is less than the value of its parent node.
- The value in any right subtree is greater than the value of its parent node.

22. What does u mean by General trees? (Understanding)

General Tree is a tree with nodes having any number of children.

23. Define Forest. (Understanding)

A forest is a collection on N(N>0) disjoint tree or group of trees are called forest. If the root is removed from the tree that tree becomes a forest.

24. State the merits of linear representation of binary trees.(Applying)

- Storage method is easy and can be easily implemented in arrays
- When the location of a parent/child node is known, other one can be determined easily
- It requires static memory allocation so it is easily implemented in all
- programming language

25. State the demerit of linear representation of binary trees.(Applying)

Insertions and deletions in a node take an excessive amount of processing time due to data movement up and down the array.

26. State the merit of linked representation of binary trees.(Analyzing)

Insertions and deletions in a node involve no data movement except the rearrangement of pointers, hence less processing time.

27. State the demerits of linked representation of binary trees. (Analyzing)

- Given a node structure, it is difficult to determine its parent node
- Memory spaces are wasted for storing null pointers for the nodes, which have one or no sub-trees
- It requires dynamic memory allocation, which is not possible in some programming language

28. Define a binary search tree. (Remembering)

A binary search tree is a special binary tree, which is either empty or it should satisfy the following characteristics:

• Every node has a value and no two nodes should have the same value i.e) the values in the binary search tree are distinct
• The values in any left sub-tree is less than the value of its parent node
• The values in any right sub-tree is greater than the value of its parent node
• The left and right sub-trees of each node are again binary search trees

29. What do you mean by general trees? (Remembering)

General tree is a tree with nodes having any number of children.

30. Define ancestor and descendant. (Remembering)

If there is a path from node n1 to n2, then n1 is the ancestor of n2 and n2 is the descendant of n1.

31. Why it is said that searching a node in a binary search tree is efficient than that of a simple binary tree? (Analyzing)

In binary search tree, the nodes are arranged in such a way that the left node is having less data value than root node value and the right nodes are having larger value than that of root. Because of this while searching any node the value of the target node will be compared with the parent node and accordingly either left sub branch or right sub branch will be searched. So, one has to compare only particular branches. Thus searching becomes efficient.

32. What is the use of threaded binary tree? (Remembering)

In threaded binary tree, the NULL pointers are replaced by some addresses. The left pointer of the node points to its predecessor and the right pointer of the node points to its successor.

33. What is an expression tree? (Remembering)

An expression tree is a tree which is built from infix or prefix or postfix expression. Generally, in such a tree, the leaves are operands and other nodes are operators.

34. Define right-in threaded tree. (Understanding)

Right-in threaded binary tree is defined as one in which threads replace NULL pointers in nodes with empty right sub-trees.

35. Define left-in threaded tree (Understanding)

Left-in threaded binary tree is defined as one in which each NULL pointers is altered to contain a thread to that node's inorder predecessor.

PART-B

1. What are the types of representation of binary tree? **(**Understanding**)**
2. What is traversal? Give an algorithm for traversal in the binary tree. (Remembering)
3. Draw a Binary search tree for the following input list 60, 25,75,15,50,66,33,44. Trace the algorithm to delete the nodes 25, 75, 44 from the tree. (Creating)
4. Explain and write routine to the various tree traversal methods. **(**Understanding**)**
5.Show the result of inserting 3,1,4,6,9,2,5,7 in to an initially empty binary search tree.
6. Show that for the perfect binary tree of height h containing 2^{h+1}-1nodes.(Applying)
7. Write a routine to implement the basic binary search tree operations. (Creating)
8.Show the result of inserting 2,1,4,5,9,3,6,7 in to an initially empty binary tree.(Applying)
9. Explain in detail about Threaded binary trees. **(**Understanding**)**

10. Write a procedure to Delete an element from a binary search tree and show the result of deleting the root. (Creating)
11. Construct an expression tree for the input ab+cde+**(Applying)
12. What is a binary search tree? Explain with example? **(**Understanding**)**
13. Explain binary tree traversals? (Understanding)
14. What is a threaded binary tree? Explain its operation with example? (Remembering)
15. Explain the expression trees? (Remembering)
16. Write the procedure to convert general tree to binary tree. (Applying)
17. Explain the various operations performed on a stack. (Remembering)
18.. Write an algorithm to convert infix expression to postfix expression. (Applying)

UNIT III – ADVANCED STRUCTURES (CO3)

PART-A

1. Define balanced search tree. (Remembering)

Balanced search tree have the structure of binary tree and obey binary search tree properties with that it always maintains the height as O(log n) by means of a special kind of rotations. Eg. AVL, Splay, B-tree.

2. Define AVL tree. (Remembering)

An empty tree is height balanced. If T is a non-empty binary tree with T_L and T_R as Its left and right subtrees, then T is height balanced if

1. T_L and T_R are height balanced.
2. $| h_L - h_R | \leq 1$.

Where hl and hr are the height of T_L and T_R respectively.

3. Name the drawbacks of AVL trees?(Understanding)

The drawbacks of AVL trees are

- Frequent rotations
- The need to maintain balances for the tree's nodes
- Overall complexity, especially of the deletion operation.

4. What is a heap? (Understanding)

A heap is a partially ordered data structure, and can be defined as a binary tree assigned to its nodes, one key per node, provided the following two conditions are met

- The tree's shape requirement-The binary tree is essentially complete, that is all the leaves are full except possibly the last level, where only some rightmost leaves will be missing.
- The parental dominance requirement-The key at each node is greater that or equal to the keys of its children

5. What is the main use of heap? (Understanding)

Heaps are especially suitable for implementing priority queues. Priority queue is a set of items with orderable characteristic called an item's priority, with the following operations

- Finding an item with the highest priority
- Deleting an item with highest priority
- Adding a new item to the set

6. Give three properties of heaps? (Analyzing)

The properties of heap are

- There exists exactly one essentially complete binary tree with 'n' nodes. Its height is equal to $\log_2 n$
- The root of the heap is always the largest element
- A node of a heap considered with all its descendants is also a heap

7. Give the main property of a heap that is implemented as an array. (Analyzing)

A heap can be implemented as an array by recording its elements in the top-down, left-to-right fashion. It is convenient to store the heap's elements in positions 1 through n of such an array. In such a representation

- The parental node keys will be in the first n/2 positions of the array, while the leaf keys will occupy the last n/2 positions
- The children of a key in the array's parental position 'i' ($1 \le i \le n/2$) will be in positions 2i and 2i+1and correspondingly, the parent of the key in position 'i' ($2 \le i \le n$) will be in position i/2.

8. What are the two alternatives that are used to construct a heap? (Understanding)

The two alternatives to construct a heap are

- Bottom-up heap construction
- Top-down heap construction

9. Give the pseudocode for Bottom-up heap construction. (Understanding)

```
ALGORITHM HeapBottomUp(H[1..n])
//Constructs a heap from the elements of the given array
//Input An array H[1..n] of orderable elements
//Output A heap H[1..n]
for I ← n/2 downto 1 do
        k ← I ; v ← H[k]
        heap ← false
        while not heap and 2*k ≤ n do
                j ← 2*k
                if j < n
                        if H[j] < H[j+1] j ←j+1
                if v ≥ H[j]
                        heap ← true
                else H[k] ← H[j]; k ← j
        H[k] ← v
```

10. What is the algorithm to delete the root's key from the heap? (Understanding)

ALGORITHM

- Exchange the root's key with the last key K of the heap
- Decrease the heap's size by one
- "Heapify" the smaller tree by sifting K down the tree exactly in the same way as bottom-up heap construction. Verify the parental dominance for K: if it holds stop the process, if not swap K with the larger of its children and repeat this operation until the parental dominance holds for K in its new position.

11. Who discovered heapsort and how does it work? (Creating)

Heapsort was discovered by J.W.J. Williams. This is a two stage process that works as follows

- ❖ Stage 1 Heap construction: construct a heap for a given array.
- ❖ Stage 2 Maximum deletions: Apply the root deletion operation n-1 times to the remaining heap

12. What is a min-heap? (Understanding)

A min-heap is a mirror image of the heap structure. It is a complete binary tree in which every element is less than or equal to its children. So the root of the min-heap contains the smallest element.

13. Define splay tree. (Remembering)

A splay tree is a binary search tree in which restructuring is done using a scheme called splay. A splay is heuristic method which moves a given vertex v to the roof of the splay tree using a sequence of rotations.

14. Define B-tree? (Remembering)

A B-tree of order m in an m-way search tree that is either empty or is of height ≥ 1 and

1. The root node has at least 2 children
2. All nodes other than the root node and failure nodes have at least m/2 children.
3. All failure nodes are at same level.

15. Define Priority Queue? (Remembering)

Priority queue is a specialized data structure in which the elements are organized according to the priorities of some key value.

16. Define Binary Heap? (Remembering)

A Binary Heap is a complete binary tree in which the key value of any node must be lesser than its children is called min heap. If the key value of any node is greater than its children is called max heap.Binary heap is also called as partially ordered tree.

17. Explain array implementation of Binary Heap.

For any element in the array position ‘i’, the left child is at the position ‘2i’, the right child is at the position ‘2i+1’ and parent is at the position ‘i/2’.

18. Define Max-heap. (Remembering)

Maxheap: A heap in which the parent has a larger key that the child’s key values then it is called Maxheap.

19. Explain AVL rotation.(Understanding)

Manipulation of tree pointers is centered at the pivot node to bring the tree back into height balance. The visual effect of this pointer manipulation so to rotate the sub tree whose root is the pivot node. This operation is referred as AVL rotation.

20. What are the different types of Rotation in AVL Tree? (Understanding)

Two types of rotation are

1. Single rotation
2. Double rotation.

PART-B

1. Show the result of inserting 10,12,1,14,6,5,8,15,3,9,7,4,11,13, and, 2, one at a time, in to an initially empty binary heap.(Evaluating)
2. Write a procedure to perform percolate up and percolate down in a binary heap. (Creating)
3. Write and test a program that performs the operation Insert, Delete Min, Build Heap, Find min, Decrease Key, Delete, and Increase Key in a binary Heap. (Creating)
4. Show the result of inserting 2, 4,1,5,9,3,6,7 in to an initially empty AVL Tree. (Evaluating)
5. Write a procedure to implement AVL single and double rotations. (Creating)
6. Write a routine to perform insertion and deletion in B-Tree. (Creating)
7. Construct the heap for the following array structure. (Evaluating)

5	19	8	37	75	55	14	22	43	4

8. Discuss the concept of AVL Tree in Detail with example. (Evaluating)

UNIT-IV GRAPHS(CO4)

1. Define Graph. (Remembering)

A Graph G, consists of a set of vertices V, and a set of edges E.V is a finite non-empty set consisting of vertices of vertices of the graph. The set of edges E consists of a pair of vertices from the vertex set.

2. What is undirected graph. (Understanding)

If an edge between any two nodes in a graph is not directionally oriented, a graph is called as undirected graph. It is also called as unqualified graph.

3**. What is directed graph. (Understanding)**

If an edge between any two nodes in a graph is directionally oriented, a graph is called as directed graph.It is also called as digraph.

4. Define a cycle in a graph. **(Remembering)**

A cycle is a path containing atleast thee vertices such that the starting and the ending vertices are the same.

5. Define a weakly connected graph. **(Remembering)**

A directed graph is said to be a weakly connected graph if any vertex doesn't have a directed path to any other vertices.

6. Define a weighted graph. **(Remembering)**

A graph is said to be weighted graph if every edge in the graph is assigned some weight or value. The weight of an edge is a positive value that may be representing the distance between the vertices or the weights of the edges along the path.

2

5 11

7. Define parallel edges (Remembering)

In some directed as well as undirected graph certain pair of nodes are joined by more than one edge, such Physically a graph can be represented as,

-adjacency matrix

-Incident matrix

-Adjacency list

-Adjacency multilist

-Circular adjacency list

8. Define Adjacency Matrix. (Remembering)

Adjacency Matrix is a representation used to represent a graph with zeros and ones.A graph containing n vertices can be represented using n rows and n columns.

9. What is meant by Traversing a Graph? (Understanding)
It means visiting all the nodes in the graph

10. Define undirected graph / directed graph. (Remembering)

If G=(V, E) is a graph. The edge between $v_1$and $v_2$is represented as (v_1, v_2). If the edges of the form (v_1, v_2) and (v_2, v_1) are treated as the same edge, then G is said to be an undirected graph.

In case of a directed graph, the edge <v_1, v_2>and <v_2, v_1>are different.

11. Define out degree of a graph. (Understanding)
In a directed graph, for any node v, the number of outgoing edges from vare called out degree of a node v. Ex : out degree of c =2

12. Define In degree of a graph. (Understanding)
In a directed graph, for any node v, the number of incoming edges to vare called In degree of a node v. Ex : In degree of c =1

13. Define total degree of a graph. (Understanding)
The sum of the In degree and out degree of a node is called the total degree of the node.Ex : total degree of a node c = 1 +2 =3

14. Define a path in a graph. (Understanding)

A path in a graph is defined as a sequence of distinct vertices each adjacent to the next,except possibly the first vertex and last vertex is different.

The path in a graph is the route taken to reach the terminal node from a starting node.
The path from ato eare

P1 = ((a,b),(b,e))

P2 = ((a,c),(c,d),(d,e))

15. What is a complete Graph? (Applying)

A complete graph is a graph in which there is an edge between every pair of vertices.

16. List out the graph traversals of graph search? (Applying)
The two methods of traversal is,

-Depth First Search (DFS)

-Breadth First Search (BFS)

17. Define minimum cost spanning tree? (Applying)

A spanning tree of a connected graph G, is a tree consisting of edges and all the vertices of G. In minimum spanning tree T, for a given graph G, the total weights of the edges of the spanning tree must be minimum compared to all other spanning trees generated from G.-Prim's and Kruskal is the algorithm for finding Minimum Cost Spanning Tree.

18. Define shortest path problem? (Applying)

For a given graph G=(V, E), with weights assigned to the edges of G, we have to find the shortest path (path length is defined as sum of the weights of the edges) from any given source vertex to all the remaining vertices of G.

19. Define topological sort? (Evaluating)

A topological sort is an ordering of vertices in a directed acyclic graph, such that if there is a path from v_i to v_j appears after v_i in the ordering.

20. What is the use of Kruskal's algorithm and who discovered it? (Evaluating)

Kruskal's algorithm is one of the greedy techniques to solve the minimum spanning tree problem. It was discovered by Joseph Kruskal when he was a second-year graduate student.

21. What is the use of Dijksra's algorithm? (Evaluating)

Dijkstra's algorithm is used to solve the single-source shortest-paths problem: for a given vertex called the source in a weighted connected graph, find the shortest path to all its other vertices. The single-source shortest-paths problem asks for a family of paths, each leading from the source to a different vertex in the graph, though some paths may have edges in common.

22. Prove that the maximum number of edges that a graph with n Vertices is n*(n-1)/2. (Analyzing)

Choose a vertex and draw edges from this vertex to the remaining n-1 vertices. Then, from these n-1 vertices, choose a vertex and draw edges to the rest of the n-2 Vertices. Continue this process till it ends with a single Vertex.

Hence, the total number of edges added in graph is (n-1)+(n-2)+(n-3)+…+1 =n*(n-1)/2.

23. Define connected and strongly connected graph.(Remembering)

Two Vertices u and v are said to be connected if there exists a path from u to v in the graph. A directed graph is said to be connected if every pair of vertices in the graph is connected.

A directed graph is said to be strongly connected if for every pair of distinct vertices v_i and v_j, there exists two disjoint paths, one from v_i to v_j and the other from v_j to v_i.

PART-B

1. Examine topological sorting of a graph G with suitable example.(Understanding)
2. Differentiate depth-first search and breadth-first search traversal of a graph with suitable examples.(Analyzing)
3. Show the algorithm for finding connected components of an undirected graph using DFS, and derive the time complexity of the algorithm.(Creating)
4. Discuss an algorithm for Breadth first Search on a graph.(Creating)
5. Discuss any two applications of Graph with example .(Creating)

6. Explain the depth first approach of finding articulation points in a connected graph with necessary algorithm. (Remembering)
7. Write short notes on Bi-connectivity. (Creating)
8. Discuss how to find Euler circuit with an example. (Creating)
9. Explain in detail about Topological Sort. (Remembering)
10. Explain in details about Application of Depth-First search. (Remembering)
11. Explain un weighted shortest-path algorithm with example. (Remembering)
12. Find the minimum spanning tree for the graph.(Evaluating)
13. Write the algorithm for Breadth-First traversal of a graph with example. .(Creating)
14. Write the algorithm for Depth-First traversal of a graph with example. .(Creating)
15. Describe the procedure for Warshal's Algorithm.
16. Write Kruskal's Algorithm with example.(Creating)
17. Show and Explain the stages of Dijkstra's algorithm for the following graph. .(Creating)
18(i) Explain Dijkstra's algorithm with an example.Dose the algorithm work for path of negative values?Explain.
(ii)What are strongly connected component?Explain.
19.(i) Explain the different ways to represent a graph. (Remembering)
(ii) Show the adjacency matrix, adjacency list and multilist representation of a given Undirected graph.
20.(i) What is Biconnectivity?Give an Example. (Remembering)
(ii) Explain how to modify Dijkstra's algorithm to produce a count of the number of different minimum paths from v to w. (Remembering)

UNIT-V SEARCHING AND SORTING(CO5)

PART-A

1. General idea of hashing and what is the use of hashing function?(Analyzing)
A hash table similar to an array of some fixes size-containing keys. The keys specified here might be either integer or strings, the size of the table is taken as table size or the keys are mapped on to some number on the hash table
from a range of 0 to table size
2. Explain Hashing (Understanding)
Hashing is a technique used to identify the location of an identifier 'x' in the memory by some arithmetic functions like f(x), which gives address of 'x' in the table.

3. Explain Hash Function. (Understanding)
Hash Function takes an identifier and computes the address of that identifier in the hash table.
4. Mention Different types of popular hash function.(Remembering)
1.Division method
2.Square method
3.Folding method
5. Define Collision. .(Remembering)
When two different keys compute in the same location or address in the hash table through any one of the hashing function then it is termed as collision.
6. Mention Different types of collision resolving techniques. .(Remembering)
The collision resolving techniques are:
1.Separate chaining.
2.Open Addressing

Linear Probing
Quadratic Probing
Double Hashing.

7. Define Separate Chaining .(Remembering)

Separate Chaining is a technique used to avoid collision, where a linked list is used to store the keys which fall into the same address in the hash table.

8. Define Open Addressing. .(Remembering)

Open addressing is an alternative to resolving collisions with linked lists.In an open addressing hashing system,if a collision occurs,alternative cells are tried until an empty cell is found.

9. Define Linear probing. .(Remembering)

In Linear Probing,F is a linear function of i,typically F(i)=i.This amount to trying cells sequentially in search of an empty cell.

10. Define Quadratic Probing .(Remembering)

Quadratic Probing is a collsion resolution method that eliminates the primary clustering problem of linear probingThe popular choice is $F(i)=i^2$.

11. Define Double Hashing. .(Remembering)

For Double Hashing,One popular choice is $F(i)=i \cdot hash_2(X)$.This formula says that we apply a second hash function to X and probe at a distance $hash_2(X)$, $hash_2(X)$…and so on.

12.What are the use of hash table?(Applying)

1.Compilers can use hash table to keep track of declared variable in source code.
2.A hash table is useful for any graph theory problem where nodes haver real names instead of numbers
3.A third use of hash table is in program that play games.
4.On line spell checkers

13.Define Equivalence relation? (Applying)

A relation R is defined on a set S if for every pair of element (a, b), a, b$\in$S, aRb is either true or false if aRb is true, then we say that a is related to b.

An Equivalence relation is a relation R that satisfies these properties,

(1) Reflexive aRa, for all a $\in$ S
(2) Symmetric aRa if and only if bRa
(3) Transitive aRb and bRc implies that aRc.

Eg:- Electrical connectivity

14. List the operation of set ADT? (Applying)

Union and Find are the two operations on set ADT

15. What is electrical connectivity? (Analyzing)

Where all connection is metal wires, in an equivalence relation. The relation is clearly reflexive, as any component is connected to itself. If a is electrically connected to b, then b must be electrically connected to a, so this relation is symmetric. Finally if a is connected to b and b is connected to c then a is connected to c. Thus electrical connectivity is an equivalence relation.

16. What is equivalence class? (Analyzing)

The equivalence class of an element a € S is the subset of S that contain the entire element that are related to a.

17. When the Disjoint set Union / Find algorithm is dynamic? (Analyzing)

The course of the algorithm, the set can change via the Union operation. The algorithm must also operate on-line: when a find is preformed, it must give an answer before continuing.

18. Define smart union algorithms. (Analyzing)

Always to make the smaller tree a sub tree of the larger, breaking ties by any method

Union by size

Union by height

19. What is path compression? (Applying)

This is the only way to speed the algorithm up, without reworking the data structure entirely. Path compression is performed during a Find operation.

20. Write the code disjoint set Find with path compression method. (Analyzing)

```
Set Type
Find(Element Type X,DisjSet S)
{
        if(S[X]<=0)
                return X;
        else
                return S[X]=Find(S[X],S);
}
```

21 **Define sorting (Understanding)**

Sorting arranges the numerical and alphabetical data present in a list in a specific order or sequence. There are a number of sorting techniques available. The algorithms can be chosen based on the following factors

- Size of the data structure
- Algorithm efficiency
- Programmer's knowledge of the technique.

22. Mention the types of sorting (Understanding)

Internal sorting

External sorting

23. What do you mean by internal and external sorting? Analyzing)

An **internal sort** is any data sorting process that takes place entirely within the main memory of a computer. This is possible whenever the data to be sorted is small enough to all be held in the main memory.

External sorting is a term for a class of sorting algorithms that can handle massive amounts of data. External sorting is required when the data being sorted do not fit into the main memory of a computing device (usually RAM) and instead they must reside in the slower external memory (usually a hard drive)

24. Define bubble sort (Understanding)

Bubble sort is a simple sorting algorithm that works by repeatedly stepping through the list to be sorted, comparing each pair of adjacent items and swapping them if they are in the wrong order. The pass through the list is repeated until no swaps are needed, which indicates that the

list is sorted. The algorithm gets its name from the way smaller elements "bubble" to the top of the list.

25. How the insertion sort is done with the array? (Analyzing)

It sorts a list of elements by inserting each successive element in the previously sorted sublist.

Consider an array to be sorted A[1],A[2],….A[n]

a. Pass 1 : A[2] is compared with A[1] and placed them in sorted order.
b. ass 2 : A[3] is compared with both A[1] and A[2] and inserted at an appropriate place. This makes A[1], A[2],A[3] as a sorted sub array.
c. Pass n-1 : A[n] is compared with each element in the sub array A[1],A[2],……A[n-1] and inserted at an appropriate position.

26. What are the steps for selection sort? (Understanding)

The algorithm divides the input list into two parts: the sublist of items already sorted, which is built up from left to right at the front (left) of the list, and the sublist of items remaining to be sorted that occupy the rest of the list.

Initially, the sorted sublist is empty and the unsorted sublist is the entire input list.

The algorithm proceeds by finding the smallest (or largest, depending on sorting order) element in the unsorted sublist, exchanging it with the leftmost unsorted element (putting it in sorted order), and moving the sublist boundaries one element to the right.

27. What is meant by shell sort? (Understanding)

Shell sort, also known as **Shell sort** or **Shell's method**, is an in-place comparison sort. It can either be seen as a generalization of sorting by exchange (bubble sort) or sorting by insertion (insertion sort).[1] The method starts by sorting elements far apart from each other and progressively reducing the gap between them. Starting with far apart elements can move some out-of-place elements into position faster than a simple nearest neighbor exchange. Donald Shell published the first version of this sort in 1959. The running time of Shell sort is heavily dependent on the gap sequence it uses

28. What are the steps in quick sort? (Understanding)

The steps are:

a. Pick an element, called a **pivot**, from the list.
b. Reorder the list so that all elements with values less than the pivot com before the pivot, while all elements with values greater than the pivot come after it (equal values can go either way). After this partitioning, the pivot is in its final position. This is called the **partition** operation.

c. Recursively apply the above steps to the sub-list of elements with smaller values and separately to the sub-list of elements with greater values.

29. Define radix sort (Understanding)

Radix Sort is a clever and intuitive little sorting algorithm. Radix sort is a non- comparative integer sorting algorithm that sorts data with integer keys by grouping keys by the individual digits which share the same significant position and value. Radix Sort puts the elements in order by comparing the **digits of the numbers**.

30. What are the advantages of insertion sort(Evaluating)

Advantages

a. Simplest sorting technique and easy to implement
b. It performs well in the case of smaller lists.
c. It leverages the presence of any existing sort pattern in the list

Disadvantages

x Efficiency of O(n) is not well suited for large sized lists x It requires large number of elements to be shifted

31. Define searching (Understanding)

Searching refers to determining whether an element is present in a given list of elements or not. If the element is present, the search is considered as successful, otherwise it is considered as an unsuccessful search. The choice of a searching technique is based on the following factors

a. Order of elements in the list i.e., random or sorted
b. Size of the list

32. Mention the types of searching (Understanding)

The types are

- Linear search
- Binary search

33. What is meant by linear search? (Understanding)

Linear search or **sequential search** is a method for finding a particular value in a list that consists of checking every one of its elements, one at a time and in sequence, until the desired one is found.

34. What is binary search? (Creating)

For binary search, the array should be arranged in ascending or descending order.

In each step, the algorithm compares the search key value with the middle element of the array. If the key match, then a matching element has been found and its index, or position, is returned.

Otherwise, if the search key is less than the middle element, then the algorithm repeats its action on the sub-array to the left of the middle element or, if the search key is greater, on the sub-array to the right.

35. Define hashing function (Remembering)

A hashing function is a key-to-transformation, which acts upon a given key to compute the relative position of the key in an array.

A simple hash function

HASH (KEY_Value)= (KEY_Value)mod(Table-size)

36. What is open addressing? (Remembering)

Open addressing is also called closed hashing, which is an alternative to resolve the collisions with linked lists. In this hashing system, if a collision occurs, alternative cells are tired until an empty cell is found.

There are three strategies in open addressing:

1. Linear probing

2. Quadratic probing

3. Double hashing

37. What are the collision resolution methods? (Remembering)

The following are the collision resolution methods
1.Separate chaining

2. Open addressing

3. Multiple hashing

38. Define separate chaining (Remembering)

It is an open hashing technique. A pointer field is added to each record location, when an overflow occurs, this pointer is set to point to overflow blocks making a linked list.

In this method, the table can never overflow, since the linked lists are only extended upon the arrival of new keys.

39. What is the use of hash table? (Remembering)

1. Compilers can use hash table to keep track of declared variable in source code.
2. A hash table is useful for any graph theory problem where nodes have real names instead of numbers
3. A third use of hash table is in program that plays games.
4. On line spell checkers

40. Explain Hash Function. (Understanding)
Hash Function takes an identifier and computes the address of that identifier in the hash table.

41. Mention Different types of popular hash function.(Applying)

1. Division method
2. Square method
3. Folding method

42. Define Collision. (Remembering)

When two different keys compute in the same location or address in the hash table through any one of the hashing function then it is termed as collision.

43. Mention Different types of collision resolving techniques. (Remembering)

The collision resolving techniques are:

1.Separate chaining.

2. Open Addressing

Linear Probing

Quadratic Probing

Double Hashing.

PART B

1. Describe about selection sort with suitable example. (Evaluating)
2. Examine the algorithm for Insertion sort and sort the following array: 77, 33, 44, 11, 88, 22, 66, and 55 (Creating)
3. List the different types of hashing techniques? Explain them in detail with Example. (Understanding)
4. Show the result of inserting the keys 2, 3, 5, 7, 11, 13, 15, 6, 4 into an initially empty extendible hashing data structure with M = 3. (Applying)
5. Write a C program to search a number with the given set of numbers using binary search (Creating)
6. Interpret an algorithm to sort a set of 'N' numbers using bubble sort and demonstrate the sorting steps for the following set of numbers: 88,11,22,44,66,99,32,67,54,10.(Evaluating)
7. Discuss the various open addressing techniques in hashing with an example. (Applying)
8. Sort the given integers and Show the intermediate results using shellsort:35,12,14,9,15,45,32,95,40,5. (Creating)
9. Write an algorithm to sort an integer array using shell sort (Creating)
10. Illustrate with example the open addressing and chaining methods of techniques collision resolution techniques in hashing. (Understanding)
11. Compare working of binary search and linear search technique with example. (Analyzing)
12. Analyze extendible hashing in brief. (Analyzing)
13. Explain in detail about separate chaining. (Remembering)
14. Formulate the rehashing technique with suitable example. (Remembering)
15. Prepare an algorithm to sort the elements using radix sort example. (Analyzing)

16. Mention the different Sorting methods and Explain about method in detailed Manner.(Understanding)

17. Sort the sequence 96, 31, 27,42,76,61,10,4 using shell sort and radix sort and prepare the required steps. (Analyzing)

18. Given input {4371, 1323, 6173, 4199, 4344, 9679, and 1989} and a hash function h(x) =x mod10 Prepare the resulting for the following: (Creating)

    a. Separate chaining hash table.

    b. Open addressing hash table using linear probing.

    c. Open addressing hah table using quadratic probing.

    d. Open addressing hash table with second hashh2(x) =7- (x mod 7).

19. Write and explain non-recursive algorithm for binary search. (Creating)

20. Using binary search, search the number 26 from the list of numbers and give the steps. 10,7,17,26,32,92 (Creating)

21. Explain about collision resolution techniques. (Remembering).