Short Questions and Answers - 2 Marks

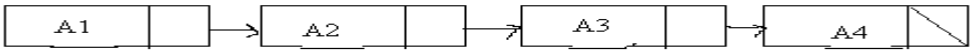
UNIT I (ARRAY AND LINKED LIST)

Q1. Define Data Structures

Data Structures is defined as the way of organizing all data items that consider not only the elements stored but also stores the relationship between the elements.

Q2. Define Linked Lists

Linked list consists of a series of structures, which are not necessarily adjacent in memory. Each structure contains the element and a pointer to a structure containing its successor. We call this the Next Pointer. The last cell's Next pointer points to NULL.



Q3. What do asymptotic notation means?

Asymptotic notations are terminology that is introduced to enable us to make meaningful statements about the time and space complexity of an algorithm. The different notations are

Big – Oh notation
Omega notation
Theta notation.

Q4. What are the basic asymptotic efficiency classes?

The various basic efficiency classes are
Constant : 1
Logarithmic : log n
Logarithmic : log n
Linear : n
N-log-n : nlog n
Quadratic : n^2
Cubic : n^3
Exponential : 2^n
Factorial : n!

Q5. List out the advantages of using a linked list

• It is not necessary to specify the number of elements in a linked list during its declaration
• Linked list can grow and shrink in size depending upon the insertion and deletion that occurs in the list
• Insertions and deletions at any place in a list can be handled easily and efficiently
• A linked list does not waste any memory space

Q6. List out the disadvantages of using a linked list

• Searching a particular element in a list is difficult and time consuming
• A linked list will use more storage space than an array to store the same number of elements

Q7. List out the applications of a linked list

Some of the important applications of linked lists are manipulation of polynomials, sparse matrices, stacks and queues.

Q8. State the difference between arrays and linked lists

Arrays: Size of an array is fixed. It is necessary to specify the number of elements during declaration. Insertions and deletions are somewhat difficult. It occupies less memory than a linked list for the same number of elements

Linked Lists: Size of a linked list is variable. It is not necessary to specify the number of elements during declaration. Insertions and deletions are carried out easily. It occupies more memory.

Arrays	**Linked Lists**
Size of an array is fixed	Size of a list is variable
It is necessary to specify the number of elements during declaration.	It is not necessary to specify the number of elements during declaration
Insertions and deletions are somewhat difficult	Insertions and deletions are carried out easily
It occupies less memory than a linked list for the same number of elements	It occupies more memory

Q9 Define an Abstract Data Type (ADT)

An abstract data type is a set of operations. ADTs are mathematical abstractions; nowhere in an ADT's definition is there any mention of how the set of operations is implemented. Objects such as lists, sets and graphs, along with their operations can be viewed as abstract data types.

Advantages:

1. Easy to debug small routines than large ones.
2. Easy for several people to work on a modular program simultaneously.
3. A modular program places certain dependencies in only one routine, making changes easier.

Q10. What are the objectives of studying data structures?

- To identify and create useful mathematical entities and operations to determine what classes of problems can be solved using these entities and operations
- To determine the representation of these abstract entities and to implement the abstract operations on these concrete representation

Q11. Explain the applications of sparse matrix.

A sparse matrix is a matrix in which most of the elements are zero. By contrast, if most of the elements are nonzero, then the matrix is considered **dense**. Sparse matrices can be useful for computing large-scale applications that dense matrices cannot handle. One such application involves solving partial differential equations by using the finite element method. The finite element method is one method of solving partial differential equations (PDEs).

Q12. What is recursive algorithm?

An algorithm is said to be recursive if the same algorithm is invoked in the body. An algorithm that calls itself is Direct recursive. Algorithm A is said to be indeed recursive if it calls another algorithm, which in turn calls A

Q13.Define the divide an conquer method.

Given a function to compute on „n‟ inputs the divide-and-comquer strategy n, yielding „k‟<k <suggests splitting the inputs in to‟k‟ distinct susbsets, 1 subproblems. The subproblems must be solved, and then a method must be found to combine subsolutions into a solution of the whole. If the subproblems are still relatively large, then the divide-and conquer strategy can possibly be reapplied.

Q14 Define optimal solution?

A feasible solution either maximizes or minimizes the given objective function is called as optimal solution

Q15.What is greedy method?

Greedy method is the most important design technique, which makes a choice that looks best at that moment. A given „n‟ inputs are required us to obtain a subset that satisfies some constraints that is the feasible solution. A greedy method suggests that one can device an algorithm that works in stages considering one input at a time.

Q16 Define dynamic programming.

Dynamic programming is an algorithm design method that can be used when a solution to the problem is viewed as the result of sequence of decisions.

Q17. What are the features of dynamic programming?

Optimal solutions to sub problems are retained so as to avoid recomputing their values. Decision sequences containing subsequences that are sub optimal are not considered. It definitely gives the optimal solution always.

Q18. What are the drawbacks of dynamic programming?

Time and space requirements are high, since storage is needed for all level. Optimality should be checked at all levels.

Unit - II (STACK AND QUEUE)

Q1. Define a stack data structure

Stack is an ordered collection of elements in which insertions and deletions are restricted to one end. The end from which elements are added and/or removed is referred to as top of the stack. Stacks are also referred as piles, push-down lists and last-in-first-out (LIFO) lists.

Q2. State the different ways of representing expressions

The different ways of representing expressions are

- Infix Notation
- Prefix Notation
- Postfix Notation

Q3. State the difference between stacks and linked lists

The difference between stacks and linked lists is that insertions and deletions may occur anywhere in a linked list, but only at the top of the stack

Q4. Mention the advantages of representing stacks using linked lists than arrays.

• It is not necessary to specify the number of elements to be stored in a stack during its declaration, since memory is allocated dynamically at run time when an element is added to the stack
• Insertions and deletions can be handled easily and efficiently
• Linked list representation of stacks can grow and shrink in size without wasting memory space, depending upon the insertion and deletion that occurs in the list
• Multiple stacks can be represented efficiently using a chain for each stack

Q5. Define a queue data structure

Queue is an ordered collection of elements in which insertions are restricted to one end called the rear end and deletions are restricted to other end called the front end. Queues are also referred as First-In-First-Out (FIFO) Lists.

Q6. Define a priority queue

Priority queue is a collection of elements, each containing a key referred as the priority for that element. Elements can be inserted in any order (i.e., of alternating priority), but are arranged in order of their priority value in the queue. The elements are deleted from the queue in the order of their priority (i.e., the elements with the highest priority is deleted first). The elements with the same priority are given equal importance and processed accordingly.

Q7. State the difference between queues and linked lists

The difference between queues and linked lists is that insertions and deletions may occur anywhere in the linked list, but in queues insertions can be made only in the rear end and deletions can be made only in the front end.

Q8. Define a Deque

Deque (Double-Ended Queue) is another form of a queue in which insertions and deletions are made at both the front and rear ends of the queue. There are two variations of a deque, namely, input restricted deque and output restricted deque. The input restricted deque allows insertion at one end (it can be either front or rear) only. The output restricted deque allows deletion at one end (it can be either front or rear) only.

Q9. What are the advantages of modularity?

• It is much easier to debug small routines than large routines
• It is easier for several people to work on a modular program simultaneously
• A well-written modular program places certain dependencies in only one routine, making changes easier

Q10. What are the types of queues?

• Linear Queues – The queue has two ends, the front end and the rear end. The rear end is where we insert elements and front end is where we delete elements. We can traverse in a linear queue in only one direction i.e. from front to rear.
• Circular Queues – Another form of linear queue in which the last position is connected to the first position of the list. The circular queue is similar to linear queue has two ends, the front end and the rear end. The rear end is where we insert elements and front end is where we delete elements. We can traverse in a circular queue in only one direction ie) from front to rear.
• Double-Ended-Queue – Another form of queue in which insertions and deletions are made at both the front and rear ends of the queue.

Q11. List the applications of stacks

• Function calling recursive or non-recursive
• Reversing a string

- Balanced parenthesis
- Recursion using stack
- Evaluation of arithmetic expressions

Q12. List the applications of queues

- Jobs submitted to printer
- Real life line
- Calls to large companies
- Access to limited resources in Universities
- Accessing files from file server

Q13. What is the need for Priority queue?

In a multiuser environment, the operating system scheduler must decide which of the several processes to run only for a fixed period of time. One algorithm uses queue. Jobs are initially placed at the end of the queue. The scheduler will repeatedly take the first job on the queue, run it until either it finishes or its time limit is up and place it at the end of the queue if it does not finish. This strategy is not appropriate, because very short jobs will soon to take a long time because of the wait involved in the run.

Generally, it is important that short jobs finish as fast as possible, so these jobs should have precedence over jobs that have already been running. Further more, some jobs that are not short are still very important and should have precedence. This particular application seems to require a special kind of queue, known as priority queue. Priority queue is also called as Heap or Binary Heap.

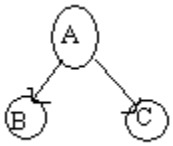
UNIT III – TREE

Q1. Define a tree

A tree is a collection of nodes. The collection can be empty; otherwise, a tree consists of a distinguished node r, called the root, and zero or more nonempty (sub) trees T1, T2,…,Tk, each of whose roots are connected by a directed edge from r.

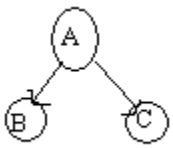
Q2. Define root AND degree of the node

This is the unique node in the tree to which further sub-trees are attached.



Here, A is the root.

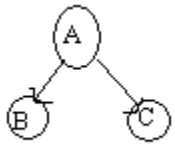
The total number of sub-trees attached to that node is called the degree of the node.



For node A, the degree is 2 and for B and C, the degree is 0.

Q3. Define leaves AND internal nodes.

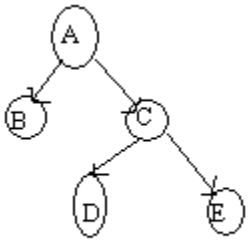
These are the terminal nodes of the tree. The nodes with degree 0 are always the leaves.



Here, B and C are leaf nodes.

internal nodes

The nodes other than the root and the leaves are called internal nodes.



Here, C is the internal node.

Q4. Define depth and height of a node

For any node n, the depth of n is the length of the unique path from the root to n. The height of n is the length of the longest path from n to a leaf.

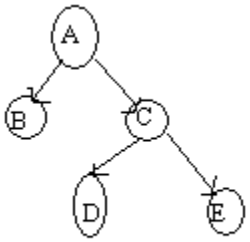
Q5. Define depth and height of a tree

The depth of the tree is the depth of the deepest leaf. The height of the tree is equal to the height of the root. Always depth of the tree is equal to height of the tree.

Q6. What do you mean by level of the tree?

The root node is always considered at level zero, and then its adjacent children are supposed to be at level 1 and so on.

Here, node A is at level 0, nodes B and C are at level 1 and nodes D and E are at level 2.



Q7. Define forest

A tree may be defined as a forest in which only a single node (root) has no predecessors. Any forest consists of a collection of trees.

Q8. Define a binary tree

A binary tree is a finite set of nodes which is either empty or consists of a root and two disjoint binary trees called the left sub-tree and right sub-tree.

Q9. Define a path in a tree

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

Q10. Define a full binary tree

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.

Q11. Define a complete binary tree

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.

Q12. State the properties of a binary tree

- The maximum number of nodes on level n of a binary tree is 2n-1, where n≥1.
- The maximum number of nodes in a binary tree of height n is 2n-1, where n≥1.
- For any non-empty tree, nl = nd+1 where nl is the number of leaf nodes and nd is the number of nodes of degree 2.

Q13. What is meant by binary tree traversal? What are the different binary tree traversal techniques?

Traversing a binary tree means moving through all the nodes in the binary tree, visiting each node in the tree only once.

- Preorder traversal
- Inorder traversal
- Postorder traversal
- Levelorder traversal

Q14. State the merits of linear representation of binary trees.

- 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

Q15. State the demerit of linear representation of binary trees.

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

Q16. State the merit of linked representation of binary trees.

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

Q17. State the demerits of linked representation of binary trees.

- 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

Q18. Define a binary search tree

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

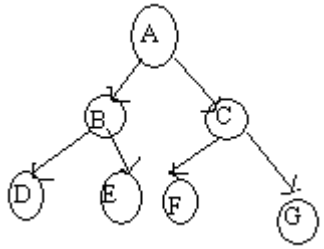
Every node has a value and no two nodes should have the same value i.e. the values in the binary search tree are distinct

- 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

Q19. What is the use of threaded binary tree?

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.

Q20. Traverse the given tree using Inorder, Preorder and Postorder traversals.



Inorder : D H B E A F C I G J
Preorder: A B D H E C F G I J
Postorder: H D E B F I J G C A

UNIT IV –GRAPHS

Q1. Define Graph.

A graph G consist of a nonempty set V which is a set of nodes of the graph, a set E which is the set of edges of the graph, and a mapping from the set for edge E to a set of pairs of elements of V. It can also be represented as G=(V, E).

Q2. What is a simple graph?

A simple graph is a graph, which has not more than one edge between a pair of nodes than such a graph is called a simple graph.

Q3. What is a weighted graph?

A graph in which weights are assigned to every edge is called a weighted graph.

Q4. Define outdegree and indegree of a graph?

In a directed graph, for any node v, the number of edges which have v as their initial node is called the out degree of the node v.

In a directed graph, for any node v, the number of edges which have v as their terminal node is called the indegree of the node v.

Q5. What is a cycle or a circuit?

A path which originates and ends in the same node is called a cycle or circuit.

Q5. What is an acyclic graph?

A simple diagram which does not have any cycles is called an acyclic graph.

Q7. What is meant by strongly connected in a graph?

An undirected graph is connected, if there is a path from every vertex to every other vertex. A directed graph with this property is called strongly connected.

Q8. When is a graph said to be weakly connected?

When a directed graph is not strongly connected but the underlying graph is connected, then the graph is said to be weakly connected.

Q9. What is an undirected acyclic graph?

When every edge in an acyclic graph is undirected, it is called an undirected acyclic graph. It is also called as undirected forest.

Q10. What are the two traversal strategies used in traversing a graph?

a.Breadthfirstsearch

b. Depth first search

Q11. What is a minimum spanning tree?

A minimum spanning tree of an undirected graph G is a tree formed from graph edges that connects all the vertices of G at the lowest total cost.

Q12. Name two algorithms two find minimum spanning tree

Kruskal'salgorithm

Prim's algorithm

Q13. Define graph traversals.

Traversing a graph is an efficient way to visit each vertex and edge exactly once.

Q14. List the two important key points of depth first search.

i) If path exists from one node to another node, walk across the edge – exploring the edge.

ii) If path does not exist from one specific node to any other node, return to the previous node where we have been before – backtracking.

Q15.DifferentiateBFSandDFS.

No.	DFS	BFS
1.	Backtracking is possible from a dead end	Backtracking is not possible
2.	Vertices from which exploration is incomplete are processed in a	The vertices to be explored are organized as a
3.	Search is done in one particular direction	The vertices in the same level are maintained

Q16. Define biconnectivity. What do you mean by articulation point?

A connected graph G is said to be biconnected, if it remains connected after removal of any one vertex and the edges that are incident upon that vertex. A connected graph is biconnected, if it has no articulation points.

If a graph is not biconnected, the vertices whose removal would disconnect the graph are known as articulation points.

Q17. What do you mean by shortest path?

A path having minimum weight between two vertices is known as shortest path,

in which weight is always a positive number.

Q18. Define adjacency list.

Adjacency list is an array indexed by vertex number containing linked lists. Each node Vi the i^{th} array entry contains a list with information on all edges of G that leave Vi. It is used to represent the graph related problems.

UNIT V – SEARCHING, SEARCH TREE AND HASHING

Q1. Define AVL Tree.

An empty tree is height balanced. If T is a non-empty binary tree with TL and TR as its left and right subtrees, then T is height balanced if

i) TL and TR are height balanced and

ii) $\left| hL - hR \right| \leq 1$

Where hL and hR are the heights of TL and TR respectively.

Q2. What do you mean by balanced trees?

Balanced trees have the structure of binary trees and obey binary search tree properties. Apart from these properties, they have some special constraints, which differ from one data structure to another. However, these constraints are aimed only at reducing the height of the tree, because this factor determines the time complexity e.g. AVL trees, Splay trees.

Q3. What are the categories of AVL rotations?

Let A be the nearest ancestor of the newly inserted nod which has the balancing factor ±2. Then the rotations can be classified into the following four categories:
Left-Left: The newly inserted node is in the left subtree of the left child of A.
Right-Right: The newly inserted node is in the right subtree of the right child of
A. Left-Right: The newly inserted node is in the right subtree of the left child of A.
Right-Left: The newly inserted node is in the left subtree of the right child of A.

Q4. What do you mean by balance factor of a node in AVL tree?

The height of left subtree minus height of right subtree is called balance factor of a node in AVL tree. The balance factor may be either 0 or +1 or -1.The height of an empty tree is -1.

Q5. Define splay tree and what is the idea behind splaying?

A splay tree is a binary search tree in which restructuring is done using a scheme called splay. The splay is a heuristic method which moves a given vertex v to the root of the splay tree using a sequence of rotations.
Splaying reduces the total accessing time if the most frequently accessed node is moved towards the root. It does not require maintaining any information regarding the height or balance factor and hence saves space and simplifies the code to some extent.

Q6. Define Heap.

A heap is defined to be a complete binary tree with the property that the value of each node is at least as small as the value of its child nodes, if they exist. The root node of the heap has the smallest value in the tree.

Q7. What is the minimum number of nodes in an AVL tree of height h?

The minimum number of nodes S(h), in an AVL tree of height h is given by S(h)=S(h-1)+S(h-2)+1. For h=0, S(h)=1.

Q8. Define B-tree of order M.

A B-tree of order M is a tree that is not binary with the following structural properties:

- The root is either a leaf or has between 2 and M children.
- All non-leaf nodes (except the root) have between ⌈M/2⌉ and M children.
- All leaves are at the same depth.

Q9. What do you mean by 2-3 tree?

A B-tree of order 3 is called 2-3 tree. A B-tree of order 3 is a tree that is not binary with the following structural properties:

The root is either a leaf or has between 2 and 3 children.

- All non-leaf nodes (except the root) have between 2 and 3 children.
- All leaves are at the same depth.

Q10. What do you mean by 2-3-4 tree?

A B-tree of order 4 is called 2-3-4 tree. A B-tree of order 4 is a tree that is not binary with the following structural properties:

- The root is either a leaf or has between 2 and 4 children.
- All non-leaf nodes (except the root) have between 2 and 4 children.
- All leaves are at the same depth.

Q11. What are the applications of B-tree?

- Database implementation
- Indexing on non primary key fields

Q12. What are the properties of binary heap?

i) Structure Property

ii) Heap Order Property

Q13. What do you mean by structure property in a heap?

A heap is a binary tree that is completely filled with the possible exception at the bottom level, which is filled from left to right. Such a tree is known as a complete binary tree.

Q14. What do you mean by heap order property?

In a heap, for every node X, the key in the parent of X is smaller than (or equal to) the key in X, with the exception of the root (which has no parent).

Q15. Define Hashing. What do you mean by hash table?

Hashing is the transformation of string of characters into a usually shorter fixed length value or key that represents the original string. Hashing is used to index and retrieve items in a database because it is faster to find the item using the short hashed key than to find it using the original value.

The hash table data structure is merely an array of some fixed size, containing the keys. A key is a string with an associated value. Each key is mapped into some number in the range 0 to tablesize-1 and placed in the appropriate cell.

Q16. What do you mean by hash function?

A hash function is a key to address transformation which acts upon a given key to compute the relative position of the key in an array. The choice of hash function should

be simple and it must distribute the data evenly. A simple hash function is hash_key=key mod tablesize.

Q17. Write the importance of hashing.

- Maps key with the corresponding value using hash function.
- Hash tables support the efficient addition of new entries and the time spent on searching for the required data is independent of the number of items stored.

Q18. What do you mean by collision in hashing?

When an element is inserted, it hashes to the same value as an already inserted element, and then it produces collision.

Q19. What are the collision resolution methods? And what do you mean by separate chaining?

- Separate chaining or External hashing
- Open addressing or Closed hashing

Separate chaining is a collision resolution technique to keep the list of all elements that hash to the same value. This is called separate chaining because each hash table element is a separate chain (linked list). Each linked list contains all the elements whose keys hash to the same index.

Q20. Write the advantage and disadvantages of separate chaining.

Advantage-

- More number of elements can be inserted as it uses linked lists.

Disadvantage-

- The elements are evenly distributed. Some elements may have more elements and some may not have anything.
- It requires pointers. This leads to slow the algorithm down a bit because of the time required to allocate new cells, and also essentially requires the implementation of a second data structure.

Q21. What do you mean by open addressing?

Open addressing is a collision resolving strategy in which, if collision occurs alternative cells are tried until an empty cell is found. The cells h0(x), h1(x), h2(x),…. are tried in succession, where hi(x)=(Hash(x)+F(i))mod Tablesize with F(0)=0. The function F is the collision resolution strategy.

Q22. What are the types of collision resolution strategies in open addressing?

- Linear probing
- Quadratic probing
- Double hashing

Q23. What do you mean by Probing?

Probing is the process of getting next available hash table array cell.

Q24. What do you mean by linear probing?

Linear probing is an open addressing collision resolution strategy in which F is a linear function of i, F(i)=i. This amounts to trying sequentially in search of an empty cell. If the table is big enough, a free cell can always be found, but the time to do so can get quite large.

Q25. What do you mean by primary clustering?

In linear probing collision resolution strategy, even if the table is relatively empty, blocks of occupied cells start forming. This effect is known as primary clustering means

that any key hashes into the cluster will require several attempts to resolve the collision and then it will add to the cluster.

Q26. What do you mean by quadratic probing?

Quadratic probing is an open addressing collision resolution strategy in which F(i)=i2. There is no guarantee of finding an empty cell once the table gets half full if the table size is not prime. This is because at most half of the table can be used as alternative locations to resolve collisions.

Q27. What do you mean by secondary clustering?

Although quadratic probing eliminates primary clustering, elements that hash to the same position will probe the same alternative cells. This is known as secondary clustering.

Q28. What do you mean by double hashing?

Double hashing is an open addressing collision resolution strategy in which F(i)=i.hash2(X). This formula says that we apply a second hash function to X and probe at a distance hash2(X), 2hash2(X),….,and so on. A function such as hash2(X)=R-(XmodR), with R a prime smaller than Tablesize.

Q29. What do you mean by rehashing?

If the table gets too full, the running time for the operations will start taking too long and inserts might fail for open addressing with quadratic resolution. A solution to this is to build another table that is about twice as big with the associated new hash function and scan down the entire original hash table, computing the new hash value for each element and inserting it in the new table. This entire operation is called rehashing.

Q30. List the limitations of linear probing.

- Time taken for finding the next available cell is large.
- In linear probing, we come across a problem known as clustering.

Q31. Mention one advantage and disadvantage of using quadratic probing.

Advantage: The problem of primary clustering is eliminated.

Disadvantage: There is no guarantee of finding an unoccupied cell once the table is nearly half full.