

IT33- DATA STRUCTURES AND ALGORITHMS

Questions and Answers - 2 Marks

UNIT I –LINEAR STRUCTURES

1. 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.

2. 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 theNext Pointer. The last cell'sNext pointer points to NULL.



3. State the different types of linked lists

The different types of linked list include singly linked list, doubly linked list and circular linked list.

4. List the basic operations carried out in a linked list

The basic operations carried out in a linked list include:

- Creation of a list
- Insertion of a node
- Deletion of a node
- Modification of a node
- Traversal of the list

5. 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

6. 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

7. 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.

8. State the difference between arrays and linked lists

| 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 |

9. Define a stack

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.

10. List out the basic operations that can be performed on a stack

The basic operations that can be performed on a stack are

- Push operation
- Pop operation
- Peek operation
- Empty check
- Fully occupied check

11. State the different ways of representing expressions

The different ways of representing expressions are

- Infix Notation
- Prefix Notation
- Postfix Notation

12. State the rules to be followed during infix to postfix conversions

- 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

13. 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

14. 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

15. Define a queue

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.

16. 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.

17. 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.

18. Define a Dequeue.

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.

19. 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.

20. 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

21. 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

22. 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 ie) 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.

23. List the applications of stacks

- Towers of Hanoi
- Reversing a string
- Balanced parenthesis
- Recursion using stack
- Evaluation of arithmetic expressions

24. 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

25. Why we need cursor implementation of linked lists?

Many languages such as BASIC and FORTRAN do not support pointers. If linked lists are required and pointers are not available, then an alternative implementation must be used known as cursor implementation.

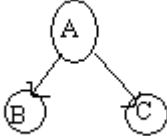
UNIT II – TREE STRUCTURES

1. 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 T_1, T_2, \dots, T_k , each of whose roots are connected by a directed edge from r .

2. Define root

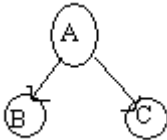
This is the unique node in the tree to which further sub-trees are attached.



Here, A is the root.

3. Define degree of the node

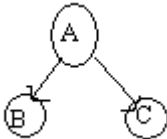
The total number of sub-trees attached to that node is called the degree of the node.



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

4. Define leaves

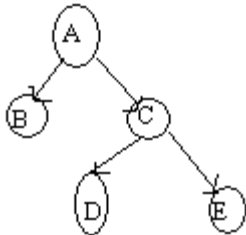
These are the terminal nodes of the tree. The nodes with degree 0 are always the leaves.



Here, B and C are leaf nodes.

5. Define internal nodes

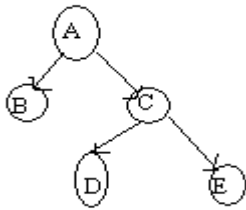
The nodes other than the root and the leaves are called internal nodes.



Here, C is the internal node.

6. Define parent node

The node which is having further sub-branches is called the parent node of those sub-branches.



Here, node C is the parent node of D and E

7. Define depth and height of a node

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

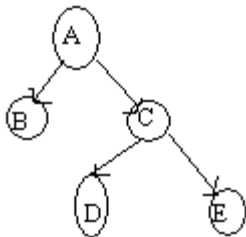
8. Define depth and height of a tree

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.

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

The root node is always considered at level zero, 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.



10. 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.

11. 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.

12. 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.

13. 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.

14. 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.

15. State the properties of a binary tree

- The maximum number of nodes on level n of a binary tree is 2^{n-1} , where $n \geq 1$.
- The maximum number of nodes in a binary tree of height n is $2^n - 1$, where $n \geq 1$.
- For any non-empty tree, $n_l = n_d + 1$ where n_l is the number of leaf nodes and n_d is the number of nodes of degree 2.

16. What is meant by binary tree traversal?

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

17. What are the different binary tree traversal techniques?

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

18. What are the tasks performed during inorder traversal?

- Traverse the left sub-tree
- Process the root node
- Traverse the right sub-tree

19. What are the tasks performed during postorder traversal?

- Traverse the left sub-tree
- Traverse the right sub-tree
- Process the root node

20. 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

21. 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.

22. 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.

23. 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

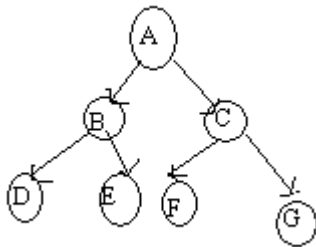
24. Define a binary search tree

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

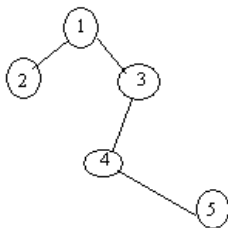
- The values in any left sub-tree is less than the value of its parent node
- The values in any right sub-tree is greater than the value of its parent node
- The left and right sub-trees of each node are again binary search trees

25. 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

27. In the given binary tree, using array you can store the node 4 at which location?



At location 6

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | - | - | 4 | - | - | 5 |
|---|---|---|---|---|---|---|---|---|

28. 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

- TL and TR are height balanced and
- $hL - hR \leq 1$

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

29. 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.

Eg: AVL trees, Splay trees.

30. What are the categories of AVL rotations?

Let A be the nearest ancestor of the newly inserted node 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.

31. 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.

32. 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.

33. 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 $h=0$, $S(h)=1$.

34. What are the properties of binary heap?

- i) Structure Property
- ii) Heap Order Property

35. 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.

36. 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).

UNIT III-HASHING AND SETS

1. Define Hashing.

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.

2. What do you mean by hash table?

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 $\text{tablesize}-1$ and placed in the appropriate cell.

3. 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 $\text{hash_key} = \text{key} \bmod \text{tablesize}$.

4. 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.

5. 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.

6. What do you mean by separate chaining?

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.

7. Write the advantage of separate chaining.

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

8. Write the disadvantages of separate chaining.

- 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.

9. 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 $h_0(x)$, $h_1(x)$, $h_2(x)$,... are tried in succession, where $h_i(x) = (\text{Hash}(x) + F(i)) \bmod \text{Table size}$ with $F(0) = 0$. The function F is the collision resolution strategy.

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

- Linear probing
- Quadratic probing
- Double hashing

11. What do you mean by Probing?

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

12. 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.

13. 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.

14. Uses of Disjoint set ADT.

1. Show how it can be implemented in minimal coding effort.
2. Greatly increase its speed, using just simple observations.
3. Analyze the running time of a fast implementation.
4. See a simple applications.

15. Define equivalence relation.

An equivalence relation is a relation R that satisfies 3 properties.

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

16. Example for equivalence relation.

1. Electrical connectivity. Where all connections are by metal wires, is an equivalence relation.
2. Two cities are related if they are in the same country.

17. Define Smart union algorithm.

The unions were performed by making the second tree the subtree of the first. To make a smaller tree a subtree of the larger breaking ties by any method we call this approach union-by-size.

18. Define union-by-size.

To make a smaller tree a sub tree of the larger breaking ties by any method we call this approach as union-by-size.

19. Define Path Compression.

Path Compression is that every node on the path from X to the root has its parent changed to the root. i.e. after every union it should made as the child nodes of a single root.

20. Define union-by-height.

To make a smaller tree a sub tree of the larger breaking ties by any method. Instead of representing size represent the height of the tree. we call this approach as union-by-height.

21. Applications of Set.

1. Used in network computers with bidirectional connections.
2. File transferring.

UNIT IV –GRAPHS

1. Define Graph.

A graph G consists 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 $G=(V, E)$.

2. Define adjacent nodes.

Any two nodes which are connected by an edge in a graph are called adjacent nodes. For example, if an edge $x \in E$ is associated with a pair of nodes (u,v) where $u, v \in V$, then we say that the edge x connects the nodes u and v .

3. What is a directed graph?

A graph in which every edge is directed is called a directed graph.

4. What is an undirected graph?

A graph in which every edge is undirected is called a directed graph.

5. What is a loop?

An edge of a graph which connects to itself is called a loop or sling.

6. 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.

7. What is a weighted graph?

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

8. Define outdegree 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 .

9. Define indegree of a graph?

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 .

10. Define path in a graph?

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

11. What is a simple path?

A path in a diagram in which the edges are distinct is called a simple path. It is also called as edge simple.

12. What is a cycle or a circuit?

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

13. What is an acyclic graph?

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

14. 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.

15. 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.

16. Name the different ways of representing a graph?

- a. Adjacency matrix
- b. Adjacency list

17. 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.

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

- a. Breadthfirst search
- b. Depth first search

19. 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.

20. Name two algorithms to find minimum spanning tree

- Kruskal's algorithm
- Prim's algorithm

21. Define graph traversals.

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

22. 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.

23. What do you mean by breadth first search (BFS)?

BFS performs simultaneous explorations starting from a common point and spreading out independently.

24. Differentiate BFS and DFS.

| 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 |

25. What do you mean by tree edge?

If w is undiscovered at the time vw is explored, then vw is called a tree edge and v becomes the parent of w .

26. What do you mean by back edge?

If w is the ancestor of v , then vw is called a back edge.

27. Define biconnectivity.

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.

28. What do you mean by articulation point?

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

29. 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.

30. Define Activity node graph.

Activity node graphs represent a set of activities and scheduling constraints. Each node represents an activity (task), and an edge represents the next activity.

31. Define adjacency list.

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

UNIT V ALGORITHM DESIGN AND ANALYSIS

1. Define an algorithm.

The algorithm is defined as a collection of unambiguous instructions occurring in some specific sequence and such an algorithm should produce output for given set of input in finite amount of time.

2. What is greedy algorithm?

In greedy method, the solution is constructed through a sequence of steps, each expanding a partially constructed solution obtained so far, until a complete solution to the problem is reached. At each step the choice made should be,

- **Feasible** – It should satisfy the problem's constraints.
- **Locally Optimal** – Among all feasible solutions the best choice is to be made.
- **Irrevocable** – Once the particular choice is made then it should not get changed on subsequent steps.

3. What are the applications of greedy method?

- a) Knapsack Problem.
- b) Prim's algorithm
- c) Kruskal's algorithm
- d) Finding shortest path
- e) Job sequencing with deadlines
- f) Optimal storage on tapes

4. Write general method of divide and conquer.

- In divide and conquer method, a given problem is,
 - i. Divided into smaller sub problems.
 - ii. These sub problems are solved independently.
 - iii. Combining all the solutions of sub problems into a solution of the whole.
- If the sub problems are large enough then divide and conquer is reapplied.
- The generated sub problems are usually of same type as the original problem.
- A control abstraction for divide and conquer is as given below using control abstraction a flow of control of a procedure is given.

5. Define Dynamic Programming

Dynamic Programming is typically applied to optimization problem. For each given problem, we may get any number of solutions from which we obtain for optimum solution (i.e. minimum value or maximum value solution.) And such an optimal solution becomes the solution to the given problem.

6. Comparison between divide and conquer and dynamic programming.

| Sl. No. | Divide and conquer | Dynamic programming |
|---------|---|--|
| 1. | The problem is divided into small sub problems. These sub problems are solved independently. Finally all the solutions of sub problems are collected together to get the solution to the given problem. | In dynamic programming many decision sequences are generated and all the overlapping sub instances are considered. |
| 2. | In this method duplications in sub solutions are neglected, i.e., duplicate sub solutions may be obtained. | In dynamic computing duplications in solutions is avoided totally. |
| 3. | Divide and conquer is less efficient because of rework on solutions. | Dynamic programming is efficient than divide and conquer strategy. |
| 4. | The divide and conquer uses top down approach of problem solving. | Dynamic programming uses bottom up approach of problem solving. |
| 5. | Divide and conquer splits its input at specific deterministic points usually in the middle. | Dynamic programming splits its input at every possible split points rather than at a particular point. After trying all split points it determines which split point is optimal. |

7. Comparison between greedy algorithm and dynamic programming.

| Sl. No. | Greedy Method | Dynamic Programming |
|---------|--|---|
| 1. | Greedy method is used for obtaining optimum solution. | Dynamic programming is also for obtaining optimum solution. |
| 2. | In greedy method a set of feasible solutions and the picks up the optimum solution. | There is no special set of feasible solutions in this method. |
| 3. | In greedy method optimum selection is without revising previously generated solutions. | Dynamic programming considers all possible sequences in order to obtain the optimum solution. |
| 4. | In greedy method there is no as such guarantee of getting optimum solution. | It is guaranteed that the dynamic programming will generate optimal solution using principle of optimality. |

8. What are the steps of dynamic programming?

- Characterize the structure of optimal solution. That means develop a mathematical notation that can express any solution and sub solution for the given problem.
- Recursively define the value of an optimal solution.
- By using bottom up technique compute value of optimal solution. For that you have to develop a recurrence relation that relates a solution to its sub solutions, using the mathematical notation of step 1.
- Compute an optimal solution from computed information.

9. State the principle of backtracking.

Backtracking is a method in which the desired solution is expressed as n tuple $(x_1, x_2, x_3, \dots, x_n)$ which is chosen from solution space, using backtrack formulation. The solution obtained i.e. $(x_1, x_2, x_3, \dots, x_n)$ can either minimize or maximize or satisfies the criteria function.

10. What is state space tree?

A state space tree is a rooted tree whose nodes represent partially constructed solutions to given problem. In backtracking method the state space tree is built for finding the solution. This tree is built using depth first search fashion.

11. How to write an algorithm?

Algorithm is basically a sequence of instructions written in simple English language. The algorithm is broadly divided into two sections.

| |
|---|
| <p style="text-align: center;">Algorithm heading</p> <p>It consists of name of algorithm, problem description, input and output.</p> |
|---|

| |
|---|
| <p>Algorithm body</p> <p>It consists of logical body of the algorithm by making use of various programming constructs and assignment statement.</p> |
|---|

12. What are the steps of algorithm design?

1. Understand the problem.
2. Decision making
 - a. Capabilities of computational devices
 - b. Select exact/approximate method
 - c. Data structures
 - d. Algorithmic strategies
3. Design of algorithm
4. Algorithm verification
5. Analysis of an algorithm
6. Coding of algorithm

13. What are the types of asymptotic notation?

- a. Big oh notation.
- b. Omega notation
- c. Theta notation

14. Define Big oh notation?

Let $F(n)$ and $g(n)$ be two non negative functions.

Let n_0 and constant c are two integers such that n_0 denotes some value of input and $n > n_0$ similarly c is some constant such that $c > 0$. We can write $F(n) \leq c * g(n)$. then $F(n)$ is big oh of $g(n)$. It is also denoted as $F(n) \in O(g(n))$. In other words $F(n)$ is less than $g(n)$ is multiple of some constant c .

15. Define Recurrence equation.

The recurrence equation is an equation that defines a sequence recursively. It is normally in following form

$$T(n) = T(n-1) + n \dots\dots 1 \text{ for } n > 0$$

$$T(0) = 0 \dots\dots 2$$

Here equation 1 is called recurrence relation and equation 2 is called initial condition. The recurrence equation can have infinite number of sequence. The general solution to the recursive function specifies some formula.

16. Define solving recurrence equation.

The recurrence relation can be solved by following methods

- a. Substitution method
- b. Master's method

17. What is the general plan for mathematical analysis of non recursive algorithm?

1. Decide the input size based on parameter n .
2. Identify algorithm's basic operation(s).
3. Check how many times the basic operation is executed. Then find whether the Execution of basic operation depends upon the input size n . Determine worst, average, and best cases for input of size n . If the basic operation depends upon worst case, average case, and best case then that has to be analyzed separately.
4. Set up a sum for the number of times the basic operation is executed.
5. Simplify the sum using standard formula and rules.

18. What is the general plan for mathematical analysis of non recursive algorithm?

1. Decide the input size based on parameter n .
2. Identify algorithm's basic operation(s).
3. Check how many times the basic operation is executed. Then find whether the Execution of basic operation depends upon the input size n . Determine worst, average, and best cases for input of size n . If the basic operation depends upon worst case, average case, and best case then that has to be analyzed separately.
4. Set up the recurrence relation with some initial condition and expressing the basic operation.
5. Solve the recurrence relation.

19. Distinguish between backtracking and branch and bound techniques.

| Sl. No. | Backtracking | Branch and Bound |
|---------|--|--|
| 1. | Solution for backtracking is traced using depth first search. | In this method, it is not necessary to use depth first search for obtaining the solution, even the breadth first search, best first search can be applied. |
| 2. | Typically decision problems can be solved using backtracking. | Typically optimization problems can be solved using branch and bound. |
| 3. | While finding the solution to the problem bad choices can be made. | It proceeds on better solutions. So there cannot be a bad solution. |
| 4. | The state space tree is searched until the solution is obtained. | The state space tree needs to be searched completely as there may be chances of being an optimum solution any where in state space tree. |
| 5. | Applications of backtracking are M coloring, knapsack. | Applications of branch and bound are job sequencing, TSP. |

20. Define P and NP.

Problems that can be solved in polynomial time” (“P” stands for polynomial).

Eg. Searching of key element, sorting of elements, All pair shortest path

NP stands for “non deterministic polynomial time”. Note that NP does not stand for “non polynomial”

Eg. Traveling salesman Problem, graph coloring problem, Knapsack problem, Hamiltonian circuit problem.

21. Two stages of non deterministic algorithm.

- a. **Non deterministic (Guessing) stage** → Generate an arbitrary string that can be thought of as a candidate solution.
- b. **Deterministic (“Verification”) stage** → In this stage it takes as input the candidate solution and the instance to the problem and returns yes if the candidate solution represents actual solution.