Define data structure?

Logical or mathematical model of particular organization of data

Briefly describe the notation of the space time tradeoff of algorithm.

**Time complexity** is a function describing the amount of time an algorithm takes in terms of the amount of input to the algorithm.

**Space complexity** is a function describing the amount of memory (space) an algorithm takes in terms of the amount of input to the algorithm. Big O notation is the language we use for talking about how long an algorithm takes to run. Order of magnitude is often called **Big-O** notation (for "order") and written as O(f(n)).

**Common Functions for Big-O**

| **F(n)** | **Name** |
|---|---|
| 1 | Constant |
| logn | Logarithmic |
| n | Linear |
| nlog n | Log Linear |
| n2 | Quadratic |
| n3 | Cubic |
| 2 n | Exponential |

Eg:

Linear search : O(n)

Binary search : O(log n)

Bubble sort : O(n2)

Merge sort : O(n log n)

Write different applications of data structure.

1. Dynamic memory allocation

2. Maintaining directory of names

3. Performing arithmetic operations on long integers

4. History of visited websites

5. job scheduling

What is the importance of Big-O notation?

It is used to classify algorithms according to how their run time or space requirements grow as the input size grows

8. What are the different operations that can be performed on different data structure?

Ans: Operation means processing the data in the data structure.

The following are some important operations.

a. Traversing

b. Searching

c. Inserting

d. Deleting

e. Sorting

f. Merging

**Traversing**

To visit or process each data exactly once in the data structure

**Searching**

To search for a particular value in the data structure for the given key value.

**Inserting**

To add a new value to the data structure

**Deleting**

To remove a value from the data structure

**Sorting**

To arrange the values in the data structure in a particular order.

**Merging**

To join two same type of data structure values

What are the different categories of data structure? Explain.

Ans:

Primitive

Non-primitive

Primitive data types:

These are the data structures which are directly supported by the machine.i.e,Any operation can be performed in these data items. The different primitive data types are: Integer, Float, Double

Non -Primitive data types:

These Data structures do not allow any specific instructions to be performed on the Data items directly.

The different non primitive data types are: Arrays, Structures, Unions Non-primitive data types are again divided into 2

1. Linear

2. Non – linear

Linear Data structures:

This Data Structures involve arranging the elements in linear fashion.

Eg: Stacks, Queue, Lists

Non- Linear Data structures:

This Data structures involve representing the elements in Hierarchical order.

Eg: Trees, Graphs

Define string? Explain different string operations.

Ans: Strings are defined as an array of characters. The difference between a

character array and a string is the string is terminated with a special character '\0'.

**Operations:**

1) Length: Find length of the string. [Number of characters in a string is called its length.]

Eg: s="computer'

Length(s) = 8

2) Substring: Accessing a substring from a given string.

3) SUBSTRING(string, initial,length)

Eg: S="TO BE OR NOT TO BE" , 3,4) =BE O

3) Concatenation: let S1 and S2 be strings s1//s2 is the string consisting of the characters of s1 followed by the characters of s2.

Eg: S1= 'welcome"

S2="home"

S1//s2=" welcomehome"

4) Indexing (pattern matching): find the position where a string pattern P Firstappears in a given

string T.

Eg: T=”welcome to ABC college” INDEX(T,”to”) =8

Explain Pattern matching algorithm

Ans: Pattern matching algorithm Pattern Searching algorithms are used to find a pattern or substring from another bigger string. There are different algorithms. The main goal to design these types of algorithms to reduce the time

complexity. The traditional approach may take lots of time to complete the pattern searching task for a

longer text.

Eg: Main String: “ABAAABCDBBABCDDEBCABC”, Pattern “ABC”

Output:Pattern found at position: 4

Pattern found at position:

10

Pattern found at position: 18

Algorithm:

P and T are strings with length R and S.

```
1.      set k=1 and MAX= S – R +1

2.      2. Repeat steps 3 to 5 while K<=MAX

3. Repeat for L=1 to R

If P[L]≠T[K+L-1] then goto step 5

4. Set INDEX= K and exit

5. Set K=K+1

6. Set INDEX=0

7.Exit
```

**Data structure**
Data structure is a representationof logical relationship existing between individual elements of data. In other words, a data structure defines a way of organizing all data items that considersnot only the elements stored but also their relationship to each other. The term data structure is used to describe the way data is stored.
**Primitive Data Structures:-** are the basic data structures that directly operate upon themachine instructions. They have different representations on different computers. Integers, floating point numbers, character constants, string constants and pointers come under this category.
**Non-primitive data structures**:-are more complicated data structures and are derived fromprimitive data structures. They emphasize on grouping same or different data items withrelationship between each data item. Arrays, lists and files come under this category

**Recursion:**
A function is recursive if a statement in the body of the function calls itself. Recursion is The process of defining something in terms of itself. For a computer language to be Recursive, a function must be able to call itself.
**LINKED LIST**
Linked lists and arrays are similar since they both store collections of data. Array is the Most common data structure used to store collections of elements. Arrays are Convenient to declare and provide the easy syntax to access any element by its index Number. Once the array is set up, access to any element is convenient and fast
**Advantages of linked lists:**
•Linked lists are dynamic data structures. i.e., they can grow or shrink during The execution of a program.
•Linked lists have efficient memory utilization. Here, memory is not pre-Allocated. Memory is allocated whenever it is required and it is de-allocated when it is no longer needed. •Insertion and Deletions are easier and efficient.
•Linked lists provide flexibility•In inserting a data item at a specified position and deletion of the data item From the given position.
•Many complex applications can be easily carried out with linked lists.

**ARRAY:-**
An array is a collection of similar data elements stored at contiguous memory locations. It is the simplest data structure where each data element can be accessed directly by only using its index number. … Rather, we can define an array which will store the data elements at contiguous memory locations.

**2D ARRAY**The two-dimensional array can be defined as an array of arrays. The 2D array is organized as matrices which can be represented as the collection of rows and columns. However, 2D arrays are created to implement a relational database lookalike data structure. It provides ease of holding the bulk of data at once which can be passed to any number of functions wherever required.**Syntax: data_type array_name[rows][columns];**

**ARRAY VS LINKED LIST**
**Array**:-•Size of an array is fixed •Memory is allocated from stack•It is necessary to specify the number of elements during declaration (i.e., duringcompile time).• It occupies less memory than a linked list for the same number of elements. •Inserting new elements at the front ispotentially expensive because existing elements need to be shifted over to make room•Deleting an element from an array is not possible.
**Linked List:-**•Size of a list is not fixed•Memory is allocated from heap•It is not necessary to specify the Number of elements during declaration(i.e., memory is allocated during run Time).•It occupies more memory.•Deleting an element is possible.
**SINGLY LINKED LIST:-**Linked List can be defined as collection of objects called **nodes** that are randomly stored in the memory.A node contains two fields i.e. data stored at that particular address and the pointer which contains the address of the next node in the memory.The last node of the list contains pointer to the
**TWO WAY LINKED LIST/DOUBLY LINKED LIST**
A Doubly Linked List (DLL) contains an extra pointer, typically called previous pointer, together with next pointer and data which are there in singly linked list.A doubly linked list or a two-way linked list is a more complex type of linked list which contains a pointer to the next as well as the previous node in sequence, Therefore, it contains three parts are data, a pointer to the next node, and a pointer to the previous node. This would enable us to traverse the list in the backward direction as well
**CIRCULAR LINKED LIST**
*Circular linked list* is a linked list where all nodes are connected to form a circle. There is no NULL at the end. A circular linked list can be a singly circular linked list or doubly circular linked list.**AdvantagesofCircularLinkedLists:**
**1)** Any node can be a starting point. We can traverse the whole list by starting from any point. 2)useful to implementation of queue

**CHARACTERISTICS OF DS**

• **Linear**: A linear describes data characteristics whether the data items are arranged in sequential form like an array.
• **Non-Linear**: A Non-Linear data structure describes the characteristics of data items that are not in sequential form like a tree, graph.
• **Static**: It is a static data structure that describes the size and structures of a collection of data items associated with a memory location at compile time that are fixed. Example – Array.
• **Homogenous**: It is a characteristic of data structures representing whether the data type of all elements is the same. Example- Array.
• **Non-Homogenous**: It is a characteristic of data structures representing whether the data type elements may or may not be the same
• **Dynamic**: It is a dynamic data structure that defines the shrinking and expanding of data items at the run time or the program's execution. It is also related to the utilization of memory location that can be changed at the program's run time. Example: Linked list
• It has some rules that define how the data items are related to each other
• It defines some rules to display the relationship between data items and how they interact with each other
• It has some operations used to perform on data items like insertion, searching, deletion, etc.
• It helps in reducing the usage of memory resources
•**Time Complexity**: The execution time of a program in a data structure should be minimum as possible
•**Space Complexity**: The memory usage through all data items in a data structure should be less possible.

**LINEAR SEARCH?**
A linear search, also known as a sequential search, is a method of finding an element within a list. It checks each element of the list sequentially until a match is found or the whole list has been searched.
Algorithm:-
•begin with the leftmost element of arr[] and one by one compare x with each element.
•x matches with an element then return the index.
•if x does not match with any of the elements then return -1.
**PROGRAM:-**
```
•#include<stdio.h>
•Int main(){
•Int a[20],I,x,n;
•Printf("How many elements?");
•Scanf("%d",&n);
•Printf("Enter array elements:n");
•For(i=0;i<n;++i)• Scanf("%d",&a[i]);
•Printf("nEnter element to search:");
•Scanf("%d",&x);
•For(i=0;i<n;++i)
•If(a[i]==x)
•Break;
•If(i<n)
•Printf("Element found at index %d",i);
•Else
•Printf("Element not found");
•Return 0;}
```

**Application of data structures**
Data structures are integral part of programming languages and when you write a program and execute it, you will

be using a data structure in some or the other context. Whether you have used data structures implicitly or explicitly doesn't matter. If you have not used any data structure explicitly in your program, operating system will use one while executing your program

•**Stacks** are used in applications like:UNDO/REDO options in your text editorStoring function calls in recursion, result that each function returns in recursion is stored in stack and when function returns the result if popped from stack ,Storing browser history, when you click back and forward buttons you visit the previous or next page that you visited respectively

• **Queues:-**are used in Scheduling processes like LRU (Least Recently Used)In asynchronous programming
For storing your keystrokes as you type on your keyboard

• **Graphs** are used in Network/ request routing In algorithms like Djikstra's algorithm (whose modified version is used in Google ,Social networking sites

• **Trees:-**A tree is also one of the data structures that represent hierarchical data. Suppose we want to show the employees and their positions in the hierarchical form.

•General tree is a tree in which each node can have many children or nodes
•The subtree of a general tree do not hold the ordered property
•In data structure, a general tree can not be empty

**Expression Trees**

An expression tree is a representation of expressions arranged in a tree-like data structure. In other words, it is a tree with leaves as operands of the expression and nodes contain the operators. Similar to other data structures, data interaction is also possible in an expression tree. Expression trees are mainly used for analyzing, evaluating and modifying expressions, especially complex expressions.

**BINARY SEARCH TREE(BST)**

Binary Search Tree is a node-based binary tree data structure which has the following properties:
• The left subtree of a node contains only nodes with keys lesser than the node's key
• The right subtree of a node contains only nodes with keys greater than the node's key.
• The left and right subtree each must also be a binary search tree.

The binary search tree is considered as efficient data structure in compare to arrays and linked lists. In searching process, it removes half sub-tree at every step. Searching for an element in a binary search tree takes o(log2n) time. In worst case, the time it takes to search an element is 0(n).

**Adding New Node BST:-**

•Create a new BST node and assign values to it.
•Insert(node, key)If root == NULL,
•Return the new node to the calling function.If root=>data < key
•Call the insert function with root=>right and assign the return value in root=>right.
• Root->right = insert(root=>right,key)
• If root=>data > key
•Call the insert function with root->left and assign the return value in root=>left.
•Root=>left = insert(root=>left,key)
•Finally, return the original root pointer to the calling function.

**APPLICATIONS OF TREES**

• **Storing naturally hierarchical data:** Trees are used to store the data in the hierarchical structure. For example, the file system. The file system stored on the disc drive, the file and folder are in the form of the naturally hierarchical data and stored in the form of trees.
•**Organize data:** It is used to organize data for efficient insertion, deletion and searching. For example, a binary tree has a logN time for searching an element.
•**Trie**: It is a special kind of tree that is used to store the dictionary. It is a fast and efficient way for dynamic spell checking.
•**Heap**: It is also a tree data structure implemented using arrays. It is used to implement priority queues.
•**B-Tree and B+Tree**: B-Tree and B+Tree are the tree data structures used to implement indexing in databases.
•**Routing table:** The tree data structure is also used to store the data in routing tables in the routers.

**Θ Notation**: The theta notation bounds a function from above and below, so it defines exact asymptotic behavior. A simple way to get Theta notation of an expression is to drop low order terms and ignore leading constants

• **Big O Notation: T**he Big O notation defines an upper bound of an algorithm, it bounds a function only from above. For example, consider the case of Insertion Sort. It takes linear time in best case and quadratic time in worst case. We can safely say that the time complexity of Insertion sort is O(n^2). Note that O(n^2) also covers linear time.

• **Ω Notation:** Just as Big O notation provides an asymptotic upper bound on a function, Ω notation provides an asymptotic lower bound. Ω Notation can be useful when we have lower bound on time complexity of an algorithm

**CONVERT INFIX TO POSTFIX**

•Push "(" onto stack, and add ")" to the end of X.
•Scan X from left to right and repeat Steps 3 to 6 for each element of X until the STACK is empty :
•If an operand is encountered, add it to Y.
•If a left parenthesis is encountered, push it onto STACK.
•if an operator is encountered, then**(a)** Repeatedly pop from STACK and add to Y each operator (on the top of STACK) which has the same precedence as or higher precedence than operator.**(b)** Add operator to STACK./*End of If structure */
•if a right parenthesis is encountered, then :
**(a)** Repeatedly pop from STACK and add to Y each operator (on the top of STACK) until a left parenthesis is encountered.
**(b)** Remove the left parenthesis. [Do not add the left parenthesis to Y]./* end of If structure *//* end of Step 2 loop */
•exit.

**MEMORY ALLOCATION IN TWO DIMENSIONAL ARRAY**

**1)using a single pointer:-** A simple way is to allocate memory block of size r*c and access elements using simple pointer arithmetic.
**2)using an array of pointers:-** We can create an array of pointers of size r. Note that from C99, C language allows variable sized arrays. After creating an array of pointers, we can dynamically allocate memory for every row.
**3)using pointer to a pointer:-** We can create an array of pointers also dynamically using a double pointer. Once we have an array pointers allocated dynamically, we can dynamically allocate memory and for every row like method 2. **4) Using double pointer and one malloc call;**

**GRAPH**

A graph can be defined as group of vertices and edges that are used to connect these vertices. A graph can be seen as a cyclic tree, where the vertices (Nodes) maintain any complex relationship among them instead of having parent child relationship

**WEiGHTED GRAPH**

A WG is a graph with the property that its edges have a number associated with it.the number is called weight of the edge.The WG can be undirected or directed

**DIRECTED GRAPH**

A DG is a graph with the property that it edges have directions.They are also called digraph

**DATA STRUCTURE CATEGORY:-**

**Data structure can be divided into following categories• Primitive data structure• Non-primitive data structure**
**Primitive data structure**

Primitive data structure are the basic structure and they are directly operated upon by the machine instructions.
The important primitive data structure are:Integer,Float,,Character,Pointer

**Non-primitive data structure**

Non-primitive data structure are derived from the primitive data structures.They can allow structuring of homogeneous or heterogenous data items.The important non-primitive data structure are:Arrays,Lists,Files

**PATTERN MATCHING ALGORITHM:-**

Pattern matching finds whether or not a given string pattern appears in a string text. Commonly used pattern matching algorithms are Naïve Algorithm for pattern matching and pattern matching algorithm using finite automata.

```
1.      SET K=1 and MAX=S-R+1.

2. Repeat Step 3 to 5 while K<=MAX:

3.  Repeat for L=1 to R:  If TEXT[K+L-1] ≠ PAT[L], then: Go to Step 5.

4.  SET INDEX=K, and EXIT.

5.  K=K+1.6. SET INDEX=0.

7. Exit
```

**INSERTION INTO A QUEUE**

```
Step 1: IF REAR = MAX-1  Write OVERFLOW Goto step 4 [END OF IF]
Step 2: IF FRONT=-1 and REAR=-1  SET FRONT = REAR = 0 ELSE SET REAR = REAR+1 [END OF IF]
Step 3: SET QUEUE[REAR] = NUM
Step 4: Exit
```

**APPLICATION OF STACKS**

• **Evaluation of Arithmetic Expressions**

A stack is a very effective data structure for evaluating arithmetic expressions in programming languages. An arithmetic expression consists of operands and operators

•In addition to operands and operators, the arithmetic expression may also include parenthesis like "left parenthesis"and"rightparenthesis.Stacks can be used to check parenthesis matching in an expression.

•Stacks can be used for Conversion from one form of expression to another.

•Stacks can be used for Memory Management.

•Stack data structures are used in backtracking problems

• **Backtracking:**-Backtracking is another application of Stack. It is a recursive algorithm that is used for solving the optimization problem.

•**Delimiter Checking:-T**he common application of Stack is delimiter checking, i.e., parsing that involves analyzing a source program syntactically. It is also called parenthesis checking.

• **Reverse a Data:-**To reverse a given set of data, we need to reorder the data so that the first and last elements are exchanged, the second and second last element are exchanged, and so on for all other elements.

**Stack using Linked List**

In linked list implementation of stack, the nodes are maintained non-contiguously in the memory. Each node contains a pointer to its immediate successor node in the stack. Stack is said to be overflown if the space left in the memory heap is not enough to create a node

**Push an element to stack using Linked List**

**Step 1** - Create a **newNode** with given value.

**Step 2** - Check whether stack is **Empty** (**top** == **NULL**)

**Step 3** - If it is **Empty**, then set **newNode→ next** = **NULL**.

**Step 4** - If it is **Not Empty**, then set **newNode→ next** = **top**.

**Step 5** - Finally, set **top** = **newNode**

**DELETE DUPLICATE ELEMENTS FROM ARRAY**

• #include<stdio.h>• Intremove_duplicate_elements(intarr[], int n){• If (n==0 || n==1)• Return n;• Int temp[n];• Int j= 0;• Int I;• For (i=0; i<n-1; i++)• If (arr[i] != arr[i+1])• Temp[j++] = arr[i];• Temp[j++] = arr[n-1];• For (i=0; i<j; i++)• Arr[i] = temp[i];• Return j;}• int main(){

• int n;

• scanf("%d",&n);

• intarr[n];

• inti;

• for(i = 0; i< n; i++){

• scanf("%d",&arr[i]);}

• n = remove_duplicate_elements(arr, n);

• for (i=0; i<n; i++)

• printf("%d ",arr[i]);

• return 0;j

Define circular linked list.

Circular linked list is a linked list where all nodes are connected to form a circle.

There is no NULL at the end.

What is linear array? What are the different operations on array.

*Ans:*

*An array is a list of finite number of n homogeneous data elements.(i.e. data elements of the same type) such that*

*The elements of an array are referenced by an index.*

*Elements of array stored at contiguous memory locations.*

*This makes it easier to calculate the position of each element by simply adding an offset to a base value.*

*Traversal*

*Insertion*

*Deletion*

***Traversal***

*It prints all the array elements one after another.*

*Algorithm*

*LA is a linear array with lower bound LB and upper bound UB.*

```
step 1: set k=LB

step 2: Repeat steps 3 and 4 while k<=UB
step 3: Apply visit to LA[k]

step 4: set k=k+1

step 5: exit
```

***Insertion***

*It adds an element at given index.*

*Algorithm*

*INSERT(LA,N,K,ITEM)*

*LA is a linear array with N elements and K is a positive integer K<=N. This algorithm inserts an element ITEM into*

*the Kth position in LA.*

```
Step 1: set J=N

step 2: Repeat steps 3 and 4 while J>=K

step 3: set LA[J+1]=LA[J]

step 4: set J=J-1

step 5: set LA[K]=ITEM

step 6: set N=N+1

step 7: exit
```

***Deletion***

*Operation of removing one of the elements from LA.*

*Algorithm*

**P**
**Li**

*DELETE(LA,N,K,ITEM)*

*LA is a linear array with N elements and K is a positive integer ,K<=N. This algorithm deletes the Kth element*

*from LA.*

```
step 1: set ITEM=LA[K]
step 2: Repeat for J=K to N-1
step 3: set LA[J]=LA[J+1]
step 4: set N=N-1
step 5: exit
```

What is the main difference between array and linked list?

*:An array is a list of finite number of n homogeneous data elements.(i.e. data elements of the same type) such that*

*The elements of an array are referenced by an index. Elements of array stored at contiguous memory locations.*

*This makes it easier to calculate the position of each element by simply adding an offset to a base value.Linked List is an ordered collection of elements of same type, which are connected to each other using pointers.*

*Linked List supports **Sequential Access**, which means to access any element/node in a linked list, we have to*

*sequentially traverse the complete linked list, upto that element.*

*Linked list can be **Linear(Singly)**, **Doubly** or **Circular** linked list.*

What are the different types of notations?

***Infix** :*

*An expression is called the Infix expression if the operator appears in between the operands in the expression.*

*Example : A+B*

*(A+B) * (C-D)*

***Prefix** :*

*An expression is called the prefix expression if the operator appears in the expression before the operands.*

*Prefix notation is also known as Polish Notation.*

*Example :*

| *infix* | *prefix* |
|---|---|
| *A+B* | *+AB* |
| *(A+B) * (C-D)* | **+AB-CD* |

***Postfix** :*

*An expression is called the postfix expression if the operator appears in the expression after the operands.*

*Postfix notation is also known as Reverse Polish Notation or Suffix.*

*Example :*

| *Infix* | *postfix* |
|---|---|
| *A+B* | *AB+* |
| *(A+B)*(C-D)* | *AB+CD-** |

?write down stack operations algorithms.

*Two basic operations:*

*a) Push : Insert an element into a stack.*

b) *Pop : Delete an element from a stack.*

🗣 *Algorithm:PUSH*

```
PUSH(STACK,TOP,MAXSTK,ITEM)
1: if TOP = MAXSTK then print:
   OVERFLOW and return.
2: set TOP = TOP + 1
3: set STACK[TOP] = ITEM
4: return
```

🗣 *Algorithm:POP*

```
POP(STACK,TOP,ITEM)
1: if TOP = 0 then print: UNDERFLOW
   and return
2: set ITEM = STACK[TOP]
3: set TOP = TOP - 1
4: return
```

Write an algorithm to perform insertion operation in a queue?

*Ans: Insertion :Algorithm*

***QINSERT( Queue, N, front, rear, item)***

***Step 1** : Check if the queue is full.*

***Step 2** : If the queue is full, produce overflow error and exit.*

*[ rear = N ]*

***Step 3** : If the queue is not full,*

*(a) If rear=0 then set rear=front=1 otherwise*

*(b) increment **rear** pointer to point the next empty space.*

*[rear = rear + 1]*

***Step 4** : Add data element to the queue location, where the rear is pointing.*

*[ Queue[rear] = item ]*

***Step 5** : return success.*

*Deletion : Algorithm*

***QDELETE( Queue, N, front, rear, item)***

***Step 1** : Check if the queue is EMPTY.*

***Step 2** : If the queue is EMPTY, produce underflow error and exit.*

*[ front=0 ]*

***Step 3** : If the queue is not empty, then set item = Queue[front]*

*and perform (a) or(b)*

*(a) If front=N, then set rear=front=0 otherwise*

*(b) increment **front** pointer to point the next empty space.*

*[ **front** = **front** + 1]*

***Step 5** : return success.*

*CIRCULAR QUEUE*

*Circular Queue is a linear data structure in which the operations are performed based on FIFO (First In First Out) principle and the last position is connected back to the first position to make a circle.It is also called 'Ring Buffer'*

What is priority queue?

*Ans: It is a collection of elements such that each element has been assigned a priority and such that the*

*order in which elements are deleted and processed comes from the following rules: An element of higher priority is processed before any element of lower priority. Two elements with the same priority are processed according to the order in which they were added to*

*the queue.*

**Define forest. .**

Forest is a collection of disjoint trees. In other words, we can also say that forest is a collection of an acyclic graph which is not connected. Here is a pictorial representation of a

forest.

**What are the applications of tree?**

1. Store hierarchical data, like folder structure, organization structure, XML/HTML data.

2. Binary Search Tree :is a tree that allows fast search, insert, delete on a sorted data. It also

allows finding closest item

3. Heap: is a tree data structure which is implemented using arrays and used to implement priority

queues.

4. B-Tree and B+ Tree : They are used to implement indexing in databases.

5. Syntax Tree: Used in Compilers.

6. K-D Tree: A space partitioning tree used to organize points in K dimensional space.

7. Trie : Used to implement dictionaries with prefix lookup.

8. Suffix Tree : For quick pattern searching in a fixed text.

9. Spanning Trees and shortest path trees are used in routers and bridges respectively in computer networks

10.As a workflow for compositing digital images for visual effects.

**How to represent a tree using array? Write a program to delete a number from binary tree?**

• Because an array's length is fixed at compile time, if we use an array to implement a tree we have to set a limit on the number of nodes we will permit in the tree. Our strategy is to fix the

maximum *height* of the tree (H), and make the array big enough to hold any binary tree of this height

(or less). We'll need an array of size (2**H)-1. Here is the biggest binary tree of depth 3: root of the

tree (A): array position 1

• root's left child (B): array position 2

• root's right child (C): array position 3

• ...

• left child of node in array position K: array position 2K

• right child of node in array position K: array position 2K+1

• So D is in position 2*2 (4), E is in position 2*2+1 (5), F is in position 2*3 (6), G is in position 2*3+1

(7). root of the tree (A): array position 1

• root's left child (B): array position 2

• root's right child (C): array position 3

• ...

• left child of node in array position K: array position 2K

• right child of node in array position K: array position 2K+1

So D is in position 2*2 (4), E is in position 2*2+1 (5), F is in position 2*3 (6), G is in position 2*3+1

(7).

```
// C program to Delete a Tree

#include<stdio.h>

#include<stdlib.h>

/* A binary tree node has data, pointer to left child

and a pointer to right child */

struct node

{

int data;

struct node* left;

struct node* right;

};

/* Helper function that allocates a new node with the

given data and NULL left and right pointers. */

struct node* newNode(int data)

{
```

**What is an undirected graph?**

An undirected graph is **a set of nodes and a set of links between the nodes**. Each node is called

a vertex, each link is called an edge, and each edge connects two vertices. The order of the two connected vertices is unimportant. An undirected graph is a finite set of vertices together with a finite

set of edges.