Data Structure

1. What is data structure ? classification of data structure.

Ans:- The logical or mathematical model of a particular organization of data is called a data structure.

Is a technique or method of study how the data core inter related to each other logically or mathematically.

Classification of data structure.

Generally classified into Primitive and non-primitive data structure.

Primitive Data Structure are basic data type such as integer, real ,character Boolean .

Image.

2. What is Data Structure of Array ? Data Structure operation performed on arrays.

Ans:- One of the smallest type of data structure is a Linear array. A Linear array is a list of a finite number of n homogeneous data elements (data element of the same type) such that

a. The element of the array are a stored respectively in successive memory location.
b. the element of the array and represented respectively by an index set consisting of n consecutive number.

Data Structure operation performed on arrays

a. insertion:- Adding a new data in the data structure element .
b. Deletion:-Remove a data form the data structure.
c. Sorting:- Arrange data in increasing or decreasing order.
d. Searching :- find the location of data in data structure.
e. Traversing :-Ascending each is data one in the data.
f. Merging :- Combining the data of two different sorted files into a single sorted list.

3. What is array? Types of array in data Structure.

Ans:- An Array can be defined as an infinite collection of homogeneous (Similar data) elements.

Array are always store multiple values which can be referenced by a single name.

Types of array

1. single dimensional Array.

2. multiple dimensional array

1. single dimensional array :- its always known as one dimensional (1 D) array.

it's use only one script to defind the element of array.

Declaration 👍

Data - type var-name [Expression];

Ex- int num[10];

Char c [5];

Initializing one dimensional array

Data type var_name[Expression]= {values };

Ex:- int num [10] ={ 1,2,3,4,5,6,7,8,9 };

Char a [5]= {‘A’ , ’B’ , ‘C’ , ‘D’ , ‘ E’ };

2. multiple dimensional array :- Multi dimensional array is more than one sub script to describe the array element. [ ] [ ] [ ]....

Two dimensional array=> it's use two subscript ,one subscript to represent row value and second subscript to represent column value.

it mainly use more Matrix representation.

Declaration Two dimensional array

data type variable name [row] [columns];

Ex:- int num [3] [2];

Initialization 2D array

data type variable name [rows] [column]= { values };

Ex :- int num [3] [2] = { 1 , 2 , 3 , 4 , 5 , 6 };

Or

Int num [ ] [ ] = {1 , 2 , 3 , 4 , 5 , 6 };

Data Structure

Q Write a program to read & write one Dimensional array.

```
# include <stdio.h><br>
# include <conio.h><br>
Void main()<br>
{<br>
Int a [ 10 ] , i ;<br>
clrscr();<br>
Printf (“enter the array element “);<br>
for(i=0 ; i<=9; i++)<br>
{<br>
scanf(“%d”,&a[i]);<br>
}<br>
Printf(“the entered Array is “);<br>
For (i=0; i<=9; i++)<br>
{<br>
printf(“%d\n”,a[ i ]);<br>
}<br>
getch();<br>
}<br>
```

Q. Application of arrays : Sparse matrices?

Ans :- A matrix can be defined a two- dimensional array having ’ m ’ columns and ‘ n ‘ rows representing m*n matrix. sparse matrices are those Matrices are those matrices that have the majority of their element equal to zero . in other word, the sparse matrix can be defined as the matrix that has a greater number of zero elements than the non-zero elements.

Q. what is singly linked list ? Explain the algorithm.

Ans:- It is linear Collection of data item called node where each node has divided into part i.e “ data part & link part”, data part store the data item and link part store the address of next node .

linked list start with a special pointer called start pointer and terminated with Null Pointer.

Algorithm :- Step 1 : Begin

Step 2: set i ← st

Step 3: Repeat step 0 & 0 while i =/ Null

Step 4: print data [i]

Step 5: i ← link [i]

Step 6:exit

Q What is sparse matrix?

Ans:-The situation in which a matrix contain more number of zero elements , then non-zero elements , such Matrix is called sparse matrix.

Data Structure

Sparsh Matrix required if we can you the simple Matrix to a store elements.

1. Storage
2. computing time

Image 1

Representation of sparse matrix

1. Array representation
2. linked list representation

1. Array Representation:-In this 2D array will represent of sparse matrix there are three field is called array representation .

(i)Row (ii)column (iii) value

Image 2

2. linked list representation :- In this 2D array will represent of sparse matrix there are four field is called linked list representation .
(i)Row (ii)column (iii) value (iv) Next Node

Image 3

Q What is polynomial Representation and addition using Linked list ?
Ans:-The linked list used to represent polynomial of any degree .
polynomial consist of variable with coefficient and exponent.

Image 4

Struct Node
{
Int coeffi ;
Int expo ;
Struct node next ;
}

Data Structure

Additional of polynomial

1. Loot around all value of linked list.
2. if value of node exponent is Greater copy this node is result and head point it.
3. if the value of the exponent is same add coefficient add to result.
4. print result .

Image 5

6. What is stack ? Write Algorithm with suitable Example.
Ans :- Stack is a collection of data item where the insertion and deletion task place on one end called top of the stack.
In stack we Can perform two operation. I.e push and pop. push means inserting a new item into the stack. and pop means deleting an item form the stack.
Stack always perform LIFO for operation that means left in first out.
A stack can represent by the help of array and linked list

A	B	C		

0 1 2 3 4

Top = 3
N(number stack) = 4

Q Write a program for PUSH operation in stack ?
Ans ➖
Step 1 : Begin
Step 2 : if Top = N then
 print overflow & exit
Step 3 : input new item
Step 4 : top ←–Top + 1
Step 5 : Stack [top] ←-item
Step 6 : Exit

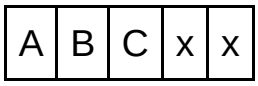
A	B	C	x	x

0 1 2 3 4

Top = 2
N = 3

Data Structure

Q Write a program for POP operation in stack ?
Ans:- Step 1 : Begin
Step 2 : if Top = -1 then
print underflow & exit
Step 3 : set item ←- stack [Top]
Step 4 : Top ←–Top – 2
Step 5 : Print deleted item
Step 6 : Exit



0 1 2 3 4

Top = 2 1 0 -1

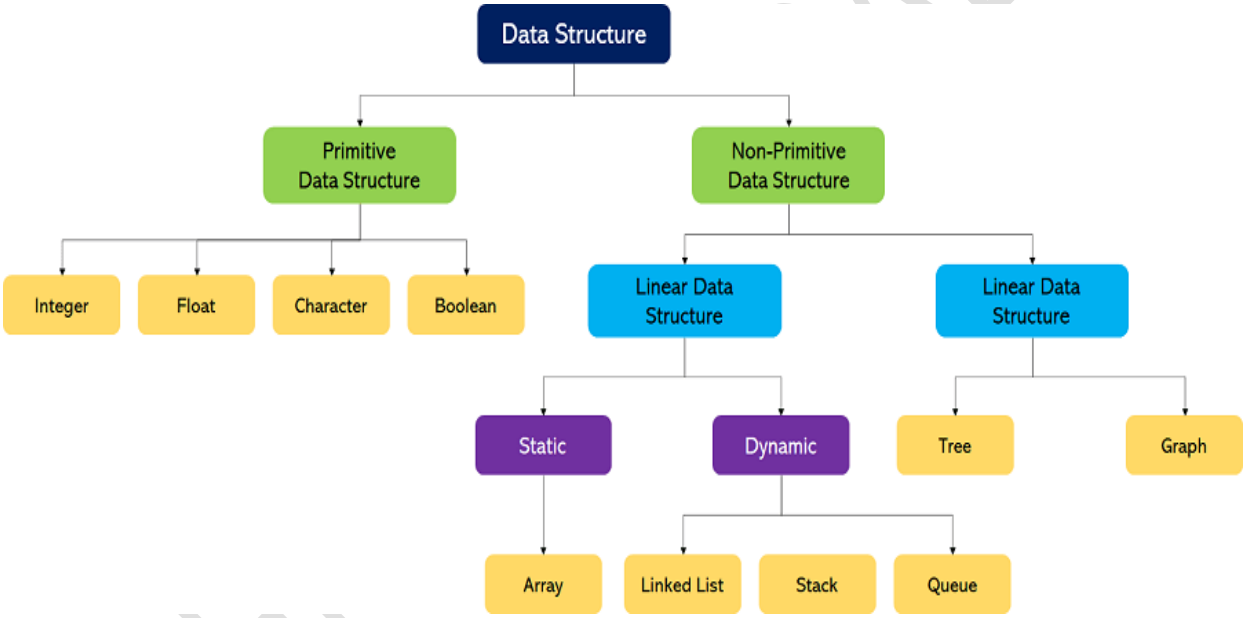
Data Structure

1.Define Data Structure.

Ans:-Organized Collection of data in a particular in a format is called data structure.
It is a technique or method of study how the data one inter related to each other logically or mathematically.

Classification of Data Structures

1. Primitive Data Structure
2. Non-Primitive Data Structure



Primitive Data Structures

1. **Primitive Data Structures** are the data structures consisting of the numbers and the characters that come **in-built** into programs.
2. These data structures can be manipulated or operated directly by machine-level instructions.
3. Basic data types like **Integer, Float, Character**, and **Boolean** come under the Primitive Data Structures.
4. These data types are also called **Simple data types**, as they contain characters that can't be divided further

Data Structure

Non-Primitive Data Structures

1. **Non-Primitive Data Structures** are those data structures derived from Primitive Data Structures.
2. These data structures can't be manipulated or operated directly by machine-level instructions.
3. The focus of these data structures is on forming a set of data elements that is either **homogeneous** (same data type) or **heterogeneous** (different data types).

2. What is abstract data type?

Ans:- An abstract data type is an abstraction of a data structure that provides only the interface to which the data structure must adhere. The interface does not give any specific details about something should be implemented or in what programming language.

Example, a List is an abstract data type that is implemented using a dynamic array and linked list. A queue is implemented using linked list-based queue, array-based queue, and stack-based queue. A Map is implemented using Tree map, hash map, or hash table.

3. Define Complexity of an algorithm.

Ans:- The complexity of an algorithm is a measure of how much computational resources, such as time and memory, it requires to solve a problem. It is usually expressed in terms of the input size of the problem, which represents the amount of data that the algorithm has to process.

There are two main types of algorithm complexity: time complexity and space complexity. Time complexity measures how much time an algorithm takes to complete as a function of the input size, while space complexity measures how much memory an algorithm requires as a function of the input size.

The complexity of an algorithm is often represented using Big O notation, which provides an upper bound on the growth rate of the algorithm's resource usage.

For example, an algorithm with time complexity O(n^2) takes no more than n^2 operations to complete, where n is the size of the input.

4. List any 5 elementary data structures.

Ans:- List are five elementary data structures:

1. Array: An array is a collection of elements of the same data type stored in contiguous memory locations. Elements can be accessed by their index position.
2. Linked List: A linked list is a collection of elements called nodes, where each node stores a value and a reference to the next node in the list.
3. Stack: A stack is an ordered collection of elements where insertion and deletion of elements are allowed only at one end, known as the top. It follows the Last-In-First-Out (LIFO) principle.
4. Queue: A queue is an ordered collection of elements where insertion is done at one end called the rear and deletion is done at the other end called the front. It follows the First-In-First-Out (FIFO) principle.
5. Tree: A tree is a hierarchical data structure consisting of nodes connected by edges. It has a single root node, and each node can have zero or more child nodes. Trees are used to represent hierarchical relationships such as in computer file systems and organization charts.

5. Define the term array.

Ans :- An array is a collection of elements of the same data type that are stored in contiguous memory locations. Each element in an array is accessed by an index, which represents the position of the element in the array. Arrays can be of fixed or dynamic size, and their size is usually determined when they are created.

6. List the most frequently used operations on Data Structures.

Ans:- List are some of the most frequently used operations on data structures:

1. Insertion - adding an element to the data structure
2. Deletion - removing an element from the data structure
3. Traversal - visiting each element of the data structure once
4. Searching - finding an element in the data structure
5. Sorting - arranging the elements in a specific order
6. Merging - combining two or more data structures into one
7. Splitting - dividing a data structure into two or more smaller data structures
8. Access - retrieving an element from the data structure by its index or key
9. Updating - modifying the value of an existing element in the data structure.

7. An Abstract Data Type is :

a. Same as an abstract Class
b. A data type that can not be instantiated
c. A data type for which only the operations defined can be used, but none else
d. All the above

Ans:-

8. A program P reads in 500 integers in the range [0,100] representing scores of 500 students. It then prints the frequency of each score above 50. What would be the best way for P to store the frequencies?

a. An array of 50 numbers

b. An array of 100 numbers

c. An array of 500 numbers

d. A dynamically allocated array of 550 numbers

Ans:- c. An array of 500 numbers

9. **An array of n numbers is given, where n is an even number. The maximum as well as minimum of these n numbers need to be determined. Which of the following is TRUE about the number of comparisons needed?**

a. At least 2n- c comparisons for some constant c

b. (3/2)n - 2 comparison

c. At least n logn comparisons

d. None of the above

Ans:- b. (3/2)n - 2 comparison

10.The minimum number of arithmetic operations required to evaluate the polynomial P(x) = x^5+4x^3+6x+5 for a given value of x using only one temporary variable is ___

Ans:- To evaluate the polynomial P(x) = x^5 + 4x^3 + 6x + 5, we can use Horner's method, which is an efficient algorithm for polynomial evaluation that requires only one temporary variable.

Horner's method involves evaluating the polynomial one term at a time, starting from the highest degree term and working down to the constant term. At each step, the result of the previous evaluation is multiplied by x and added to the next term of the polynomial. The final result is the value of the polynomial for the given value of x.

Using Horner's method, the number of arithmetic operations required to evaluate the polynomial is proportional to the degree of the polynomial. In this case, the degree of the polynomial is 5, so we will need to perform 5 multiplications and 4 additions:

1. Set the temporary variable t to 0.
2. Multiply t by x and add 5 to get t = 5.

3. Multiply t by x and add 6 to get t = 6x + 5.
4. Multiply t by x^2 and add 4 to get t = 4x^3 + 6x + 5.
5. Multiply t by x and add 1 to get t = x^5 + 4x^3 + 6x + 5.

Thus, the minimum number of arithmetic operations required to evaluate the polynomial P(x) using Horner's method with only one temporary variable is 5 multiplications and 4 additions, for a total of 9 arithmetic operations.

11. What are the different ways of expressing the complexity of an algorithm?

Ans:- There are several ways of expressing the complexity of an algorithm.

1. Time complexity: This refers to the amount of time an algorithm takes to run as a function of the input size. It is usually measured in terms of the number of basic operations (such as comparisons, assignments, and arithmetic operations) performed by the algorithm.

2. Space complexity: This refers to the amount of memory an algorithm requires as a function of the input size. It is usually measured in terms of the number of memory cells (such as bits, bytes, or words) used by the algorithm.

3. Worst-case complexity: This refers to the maximum amount of time or space an algorithm requires for any input of a given size. Worst-case complexity is often used to provide a guarantee that an algorithm will always terminate within a certain amount of time or space, regardless of the input.

4. Average-case complexity: This refers to the expected amount of time or space an algorithm requires for a random input of a given size, averaged over all possible inputs. Average-case complexity can be more informative than worst-case complexity for algorithms that are sensitive to the distribution of inputs.

5. Best-case complexity: This refers to the minimum amount of time or space an algorithm requires for any input of a given size. Best-case complexity is often not a useful measure of an algorithm's performance, since it may be achieved only for a small subset of inputs.

12. Time and Space complexities are inversely proportional. Show this relation by taking a suitable example.

Ans:- Time complexity and space complexity are not necessarily inversely proportional, but there can be a trade-off between them in some cases. A classic example of this trade-off is the Merge Sort algorithm.

Merge Sort is a divide-and-conquer sorting algorithm that uses a recursive approach to sort an array of n elements. The algorithm divides the input array into two halves, sorts each half recursively, and then merges the sorted halves to produce the final sorted array.

13. How are two dimensional arrays represented in memory?

Ans:- A two-dimensional array, also known as a matrix, is a collection of values organized in a grid with rows and columns. In computer memory, a two-dimensional array is represented as a contiguous block of memory, where each row of the matrix is stored in a consecutive block of memory addresses.

There are two main ways to represent a two-dimensional array in memory: row-major order and column-major order.

14. Explain how address of an element is calculated in two dimensional arrays.

Ans:- In a two-dimensional array, also known as a matrix, the elements are arranged in a grid with rows and columns. To access a particular element in the matrix, we need to calculate its memory address. The memory address of an element in a two-dimensional array is calculated using the following formula:

address = base_address + (row_index * number_of_columns + column_index) * element_size

- `base_address` is the memory address of the first element of the matrix
- `row_index` is the index of the row of the element, starting from 0
- `number_of_columns` is the number of columns in the matrix

- `column_index` is the index of the column of the element, starting from 0
- `element_size` is the size of each element in bytes

15. An array contains 25 positive integers. Write a module which finds all pairs of numbers whose sum is 30.

Ans:-
```
def find_pairs(arr):
    pairs = [ ]
    for i in range(len(arr)):
        for j in range(i+1, len(arr)):
            if arr[i] + arr[j] == 30:
                pairs.append((arr[i], arr[j]))
    return pairs
```

16. What is the smallest value of n such that an algorithm whose running time is 100n2 runs faster than an algorithm whose running time is 2n on the same machine?

Ans:- To find the smallest value of n such that an algorithm whose running time is 100n^2 runs faster than an algorithm whose running time is 2^n on the same machine, we can set the two running times equal to each other and solve for n:

100n^2 = 2^n

Taking the natural logarithm of both sides, we get:

2 * ln(n) = ln(100) + ln(ln(2))

Solving for n, we get:

n = e^(ln(100)/2 + ln(ln(2)))

Using a calculator, we get:

n ≈ 14.45

Therefore, the smallest value of n such that the algorithm with running time 100n^2 runs faster than the algorithm with running time 2^n is approximately 14.45.

17. Suppose we have divided n elements in to m sorted lists. Show how to produce a single sorted list of all n elements in time O(n log m).

Ans:- We can merge the m sorted lists into a single sorted list of n elements using a modified version of the standard merge sort algorithm. The basic idea is to repeatedly compare and merge pairs of sorted lists until we have a single sorted list of all n elements.

Here's how the algorithm works:

1. Divide the n elements into m sublists of approximately equal size.
2. Sort each sublist using any efficient sorting algorithm, such as quicksort or mergesort. This takes O(n log (n/m)) time.
3. Merge the m sorted sublists into a single sorted list using a modified version of the standard merge sort algorithm as follows:
   a. Create a min-heap of size m and initialize it with the smallest element from each of the m sublists. This takes O(m) time.
   c. When a sublist is exhausted, replace its element in the heap with a very large sentinel value so that it is never selected as the minimum element again. This takes O(m) time.
4. The output list is now a sorted list of all n elements.

Overall, the time complexity of this algorithm is O(n log m), which is dominated by the time needed for the merging step.

18. Let m be an n*n square matrix array. Write a module to find the sum of the elements above the principal diagonal.

Ans:-