Data Structure

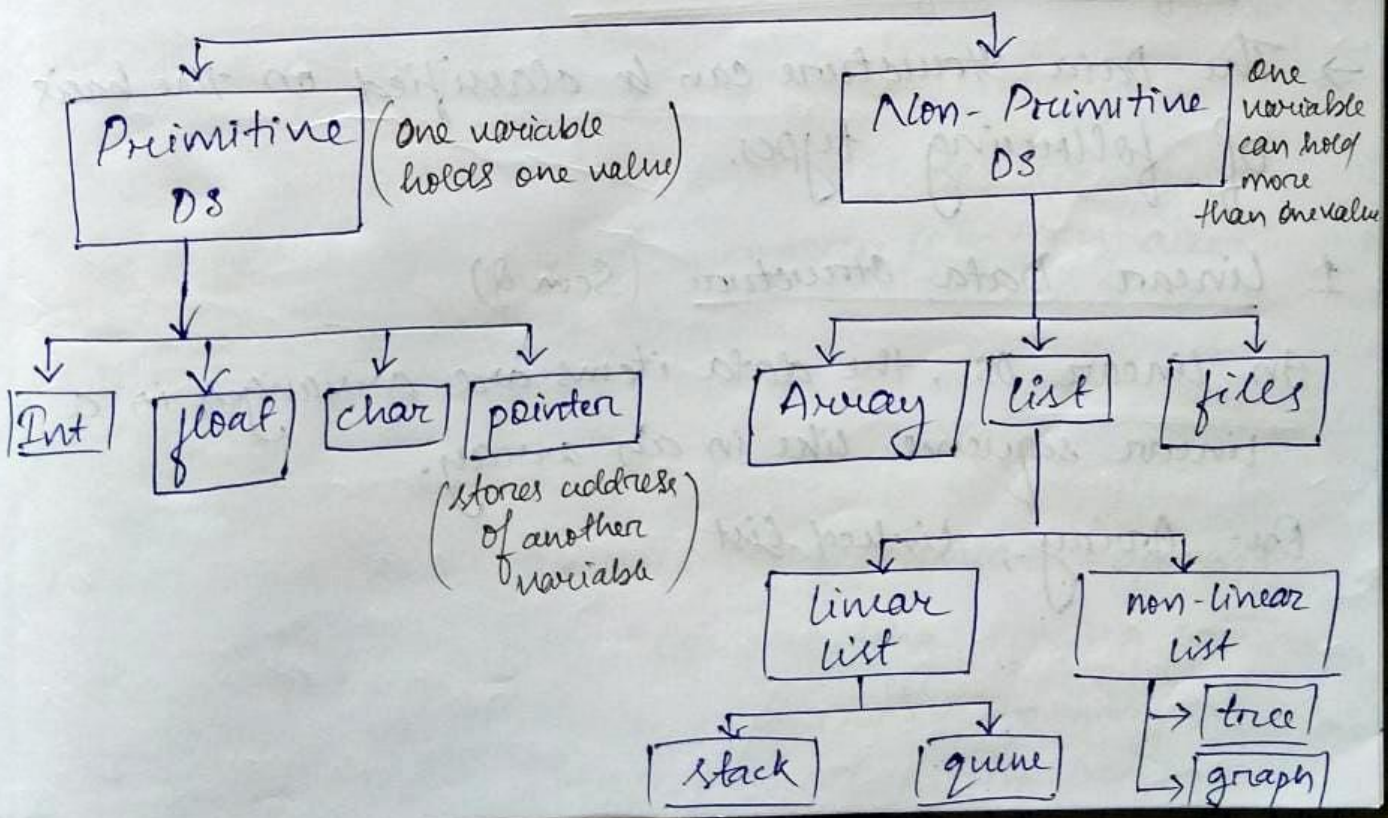
→ Data is raw fact.

→ Data Structure is the particular organisation of data either in a logical or mathematical manner.

→ Data Structure is the way of collecting and organising data in such a way that we can perform operation on this data in an effective manner.

→ Data Structure = Organised Data + Allowed Operation

Data Structure



→ To store the data in an organised manner, to access the data more quickly and to manage the data efficiently is called Data Structure.

2 marks
→ A datatype can be defined as an abstract concept that defines an internal representation of data in a memory.

→ A method of interpreting a bit pattern is often called a datatype.

→ A set of values on which some operation is performed is called as datatype..

* Classification of Data Structure

→ The Data Structure can be classified on the basis of following types.

1. Linear Data Structure (Sem 8)

In linear DS, the data items are arranged in a linear sequence like in an array.

Ex- Array., Linked list

2. Non-Linear DS

In non-linear DS, the data items are not in sequence.

Ex - Tree, Graph.

Homogenous Data Structure

In homogenous DS, all the elements are of same type. Ex - Array.

Non-Homogenous Data Structure

In non-homogenous DS, the elements may or may not be of same type. Ex- Record.

→ **Static Structures**

Static Structures are ones whose size and structures associated memory locations are fixed at compile time.

→ **Dynamic Structures**

Dynamic Structures are ones which expands or shrinks, also required ~~reading~~ during program execution and their associated memory location is changed. Ex: Linked list created using pointer.

List

→ A list can be defined as a collection of variable no. of items.

→ Lists are the most commonly used data structure.

→ An element of list must contain two fields.
  (i) Storing data / information
  (ii) Storing address of next element.

Array

→ An array is defined as a set of finite no. of homogenous elements or data items.

→ It means an array can contain one type of data only, i.e, either integers or floating point numbers or all characters.

For ex:- a[10]

Linked List

→ A linked list is a linear data structure like arrays where each element is a separate object.

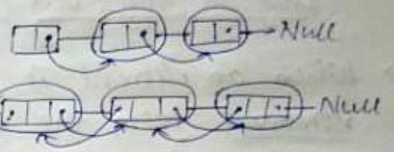
→ Each element, that is, node of a list comprises of two items :- the data and the address of the next node.

Types of Linked List

① Single Linked List

② Double Linked List

③ Circular Linked List



Stack

Common Operations Performed on Array (2 marks)

1. Creation of an Array
2. Traversing an array (Accessing an array element)
3. Insertion of new element.
4. Deletion of required element.
5. Modification of element
6. Merging of arrays

Stack

→ A stack is also an ordered collection of elements like arrays but it has a special feature that deletion and insertion of elements can be done only from one end called the top of the stack.

→ Due to this property, it is also called as Last In First Out (LIFO) type of data structure.

→ When an element is inserted into a stack or removed from the stack, its base remains fixed while the top of the stack changes.

→ Insertion of elements into stack is called Push and Deletion of elements from stack is called Pop.

→ The Stack can be implemented in two ways:-
1. Using Arrays (Static Implementation)
2. Using Pointers (Dynamic Implementation)

Queue

→ Queues are First In First Out (FIFO) type of data structure.

→ In Queue, the new elements are added to the queue from one end called the Rear End and the elements are always removed from other end called the Front End.

→ Queues can also be implemented in two ways:
1. Using Arrays
2. Using Pointers

→ Two Operations
1. Enqueue (Insertion)
2. Dequeue (Deletion)

Operations on Data Structure Imp (5 marks)

1. Creating
   - A data structure is created.
2. Inserting
   - New items are added to the data structure.
3. Modifying
   - The values of a data structure are modified by inserting more values.
4. Transversing
   - Each item in the data structure is visited for processing purpose.
5. Searching
   - A data item is searched in the structure, the item may or may not exist in the data structure.

6. Deleting
- It is a process of removing an item from the data structure.

7. Sorting
- Data items are sorted in ascending or descending order.

8. Merging
- Data items in more than one sorted data structure are merged together to produce a single sorted data structure.

9. Copying
- Data items from one structure are copied to another structure.

10. Concatenating
- Data items of a structure are ~~added~~ appended at the end of another same type of structure.

11. Spliting
- Data items in a very big structure are split into smalled structures for processing.

01.05.23

Q. Differentiate between single linked list and double linked list. (2 marks)

2. What is linked list? Mention the types of linked list. (2 mark)

3. Explain ~~ADT~~ ADT. (Abstract Data Type)

4. List the linear and non-linear data structure like example.

5. What are primitive datatypes? (2 marks)

6. What is stack?

7. What is array?

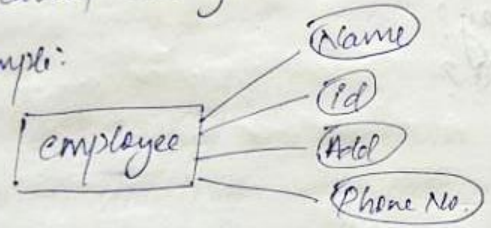
8. What is queue?

Basic Terminology or Elementary Data Organisation

1. Data :- It is a simple value or a set of values.
   - → It is a raw fact.
   - → It is present in system.

2. Data Item :
   - → Only one thing is present, single unit of value.

3. Entity
   - → Anything which has attributes and some properties is called entity.
   - → Example:



4. Entity Set
   - → Set of different - different entities which are combined together.
   - → Entities ~~have~~ with similar attributes or properties is called entity set.

5. Range
   - → which has a minimum and maximum value is called range.
   - → Each attributes of entity set has a range of values.
   - → Example:

6. Information
   - → Meaningful or processed data is called information.
   - → Example: Name :- Padmini

7. Field
   - → Field is the column of any table.
   - → Single elementary unit of information that represents attributes of entity.

8. Record : Given the information of a specific person.
   - → Collection of field values of given entity.
   - → The rows of any table.

9. File

→ All the records and fields together is called file.

→ Collection of records of entities in given entity set.

10. Primary Key

→ Any off key which we can uniquely define or iddentify the record is called Primary key.

→ Uniquely determined record in file.

11. Fixed-length record

→ Record contains same ~~lengt~~ amount of space assigned to each item.

→ Example. Phone no.,

12. Variable-length record

→ Record contains different lengths and the size is not fixed.

13. Database Management System (DBMS)

→ It is used to manage the database.

→ Multiple files together is called a database.

→ Managing the different-different files is called DBMS.

03.05.23

Analysis of an Algorithm

Introduction to Algorithm

→ An algorithm is a well defined computational procedure that takes some value or set of values as input and produces some value or set of values as output.

→ An algorithm is a finite sequence or finite set of instructions to solve a problem.

→ It is a sequence of computational steps that transfer the input into the output.

→ An algorithm is a tool for solving a well-defined specified computational problem.

→ The algorithm describes a computational procedure for achieving input or output relationship.

Q. What is algorithm? (2mark).

→ Every algorithm must satisfy the following caiteria

i) Input
ii) Output
iii) Definiteness
iv) Finiteness
v) Effectiveness

→ Input

→ There are some input data which are externally supplied to the algorithm 0 or more inputs

→ Output

→ There will be atleast one output.

→ Definiteness

→ Each instruction or steps of the algorithm must be clear or unambiguous.

→ Finiteness

→ Every algorithm must have finite no. of steps. The steps must be countable.

→ If we trace out the instructions or steps of an algorithm then for all case the algorithm will terminate after a finite no. of steps.

→ Effectiveness.

→ The steps of an algorithm must be sufficiently basic that it can be carried out by a person mechanically using pen & paper.

OR

→ It should not contain any unnecessary statement.

08.05.23

Q. Write an algorithm for the sum of two nos.

Ans. Step 1: Start
Step 2: Read a, b
Step 3: Sum = a + b
Step 4: Stop / End

Q. Write an algorithm for the greatest of 3 integers.

Step 1: Start
Step 2: Read a, b, c
Step 3: If a > b and a > c then a is greatest
Step 4: If b > a and b > c then b is greatest
Step 5: If c > a and c > b then c is greatest.
Step 6: Stop.

Q. Difference between Algorithm & Program.

Algorithm	Program
i) Required at design phase.	i) Required at implementation phase.
ii) Can be written in natural language / English language.	ii) Can be written in programming language like C, C++, Python, java, javascript.
iii) The person should have domain knowledge.	iii) It is written by programmer.
iv) Analyze the algorithm.	iv) Compiling the program.
v) Hardware, software independent.	v) Hardware, software dependent.

Analysis of an Algorithm

→ Degree of difficulty of an algorithm is called complexity or degree of efficiency of an algorithm is called complexity.

Time Complexity

Rule 1: For single statement

Ex: $x = x + 1$ its time complexity is $O(1)$.

Rule 2: For loop

$i \leftarrow 1$ to $n \rightarrow O(n)$

$x = x + 1$ its time complexity is $O(n)$.

Rule 3: For nested loop

$i \leftarrow 1$ to n	$= n$	n
$i \leftarrow 1$ to n	$= n.n$	n^2
$x = x + 1$	$= 1.n.n$	n^2
		$O(n^2)$

Rule 4: For consecutive statement

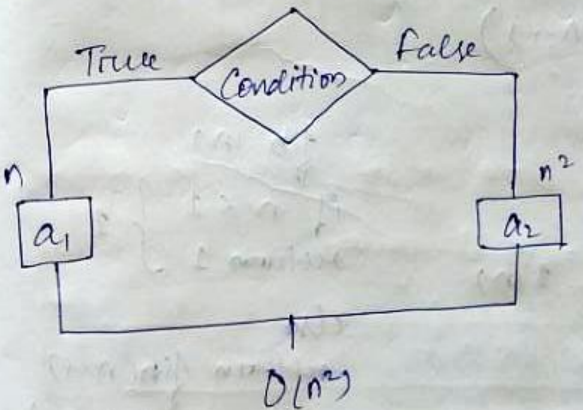
for $\left.\begin{array}{l} i \leftarrow 1 \text{ to } n \\ x = x + 1 \end{array}\right\}$ a $\quad O(n)$

$\left.\begin{array}{l} i \leftarrow 1 \text{ to } n \\ i \leftarrow 1 \text{ to } n \\ x = x + 1 \end{array}\right\}$ b $\quad O(n^2)$

max (a, b)

max $(n, n^2) = O(n^2)$

Rule 5: If-else statement



Rule 6: For while loop

```
while (n>0)
{
    i ← i+1
    n ← n/2
}
```

$O(\log_2 n)$

```
while (n>0)
{
    i ← i+1
    n ← n-1
}
```

$O(n)$

* `while (n<0) { i ← i+1 n ← n-1 }`

$O(T(n-1))$

Rule 7: Recursion

```
fact(n)
if n ≤ 1 } — 1(n)
return 1
else
return n * fact(n-1)
```

n

$O(n)$

```
fib(n)
if n < 1 } 1
return 1
else
return fib(n-1)
       + fib(n-2)
```

$T = 1 + T(n-1) + T(n-2)$

$= T(n-1) + T(n-2)$

29.05.23

Binary Search

→ It is a searching algorithm for finding an element in a sorted array.

→ In this approach, the element is always searched in the middle of a portion of an array.

→ It can be implemented only on a sorted list of items.

→ If the elements are not sorted, we need to sort them first.

0	1	2	3	4	5	6	7	8	9
5	9	17	23	25	45	59	63	71	89

l = 0, mid = 4, r = 9, n = 10

$mid = \left[\frac{l+r}{2}\right]$; data = 59

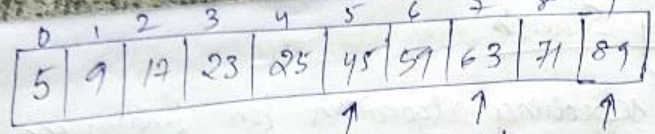
l	r	mid
0	9	4
5	9	7
5	6	5
6	6	6 → stop

1) data == a[mid]

2) data < a[mid]

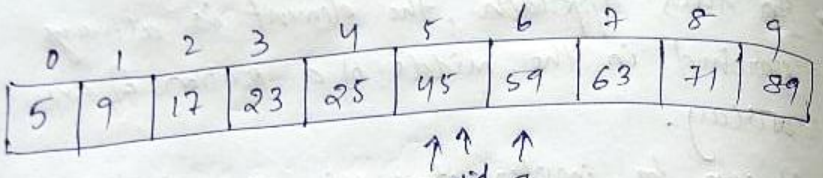
3) data > a[mid]

→ data > a[mid]



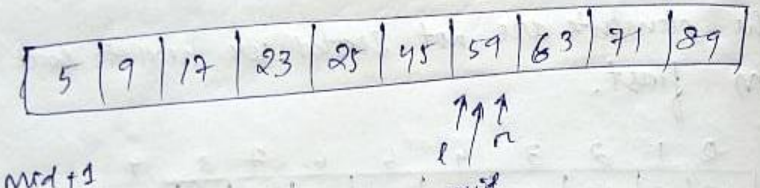
→ data < a[mid]

l = mid + 1



r = mid - 1

→ data > a[mid]

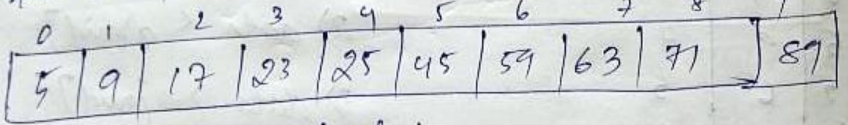


l = mid + 1

→ data == a[mid]

⇒ when l > r, data is not present in the list.

→ Eg ex, data = 60.



data > a(mid) → l = mid + 1
= 7
r = 6

l > r ⇒ data doesn't exist.

Q.

Ans. (data = 72)

l	r	mid	a[mid]	
0	9	4	50 < 72	l = mid + 1
5	9	7	72 = 72	

(ii) data = 75

l	r	mid	a[mid]	
0	9	4	50 < 75	l = mid + 1
5	9	7	72 < 75	l = mid + 1
8	9	8	80 > 75	r = mid - 1
8	8	8	80 > 75	r = mid - 1
8	7			

Since l > r, ⇒ data doesn't exist

(iii) data = 12

l	r	mid	a[mid]	
0	9	4	50 > 12	r = mid - 1
0	3	1	8 < 12	l = mid + 1
2	3	2	12 == 12	

→ stop.

```
int BinarySearch(a, n, data)
{
  l = 0, r = n-1,
  while (l < r)
  { mid = (l+r)/2;
    if (data == a[mid])
      return mid;
    else if (data < a[mid])
      r = mid-1;
    else
      l = mid + 1;
  }
  return -1;
}
```

Time complexity :- ~~O (log n)~~

O (log n)