Unit ⇒ 1

Date: Page: 01

Data Structures

Data may be organized in many different ways. The logical or mathematical model of a particular organization of the data is called data structures.

Operations ⇒:

1. Insert ⇒ Adding a new record to D.S.
2. Delete: ⇒ Removing a Record from the D.S.
3. Searching: ⇒ Finding the location of the record with a given key value or finding the location of all records which satisfy one or more condition.
4. Traversing: ⇒ Excessing each record exactly once so that certain items in the record may be proceed.
5. Sorting ⇒ Arrange the Record in some logical order.
6. Merge ⇒ Combine the data of two different sorted files into a single sorted file.

address Array → adv. & disa., write in alag assignment copy. How
— Linear & non-li d.s

Data Structure

- Primitive D.S. (Built-in D.S)
  - int
  - float
  - char
  - Pointer
- Non-Primitive D.S. (User-defined D.S)
  - Array
  - List
    - Linear List
      - Stack
      - Queue
      - Linked List
    - Non-linear list
      - Tree
      - Graph
  - file

1. Primitive D.S. ⇒ is directly operated by the machine instruction and the size of this data structures is pre-defined. Ex ⇒ Int, float, char, string

2. Non-Primitive D.S ⇒ is a user-defined data structures and these are derives from the Primitive D.S.

The non Primitive D.S. emphases on structuring of group of homogenous & heterogenous data items.

For ex ⇒ Array, Stack, Queue, Linked List etc.

Date: ⇒

Linear List: ⇒

Linked-List: ⇒

Linked list overcome the disadvantage of array data structure the size of array cannot be changed after it's declaration i.e. it's size has to be known at compilation time.

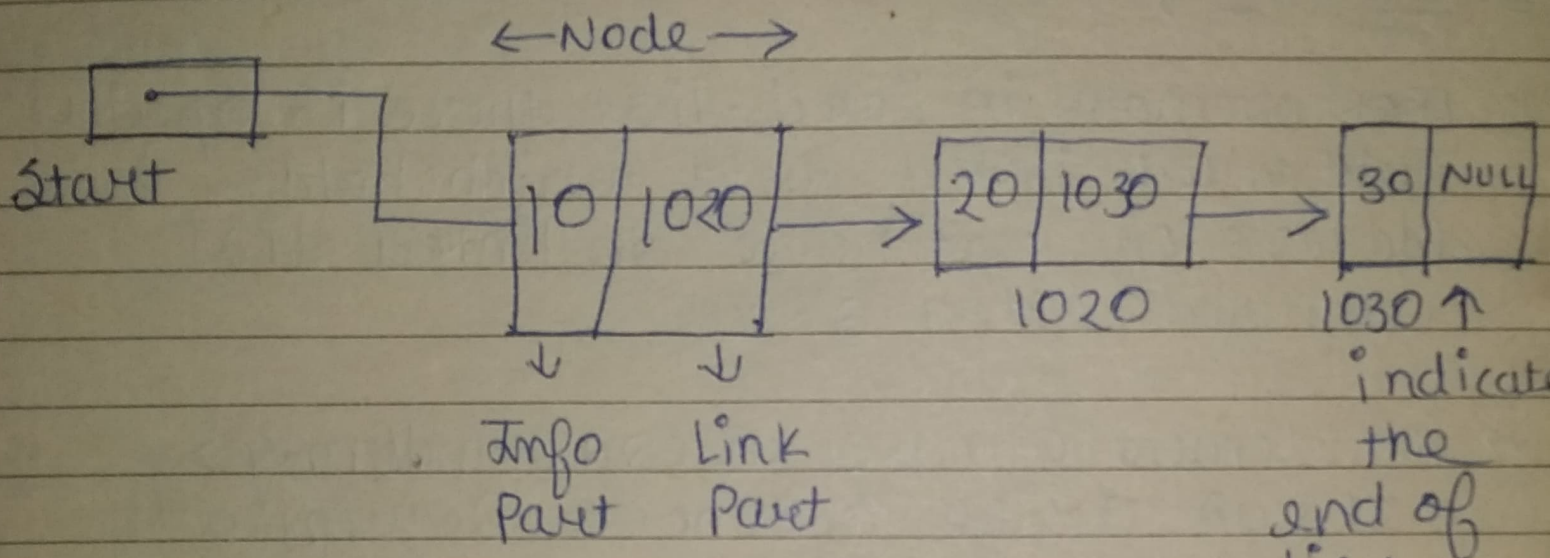
The element of linked list are not stored in adjacent location as in array.

Linked list is defined as ordered set of element which may increase or decrease as when required. In Linked list memory space located for the element of the list can be extended at any time.

A Linked list is an ordered collection of homogenous data elements called nodes where the linear order is maintained by means of links of pointer.

Each node is divided into two parts. The first part contain information and second part contain the address of next node in first.

The number of Pointer is maintained depending on the requirement & user.



Advantage :=>

1. Linked list are dynamic data structure.
2. They can grow & shrink during the execution of program
3. Efficient memory utilization → memory is not are alloated & memory is alloacted whenever required.
4. Intili Insertion & delotion of node are efficient → Linked list provide in insertion data item have specified posn or deletion of data item for

given Position.

* Disadvantage ⇒ :

1. More memory in linked-list there is special field called linked field which hold address of next node so linked list required extra space.

2. No element can be access randomly ⇒ It has access each node sequentially.

3. Reverse traversing is difficult in linked list.

Application ⇒

i. Linked list is used to represent Polynomial expression.

2. Linked list is used to symbolic table, Symbol table is data structure used in compiler for keeping the value of variable & constant data used in appl. of our program.

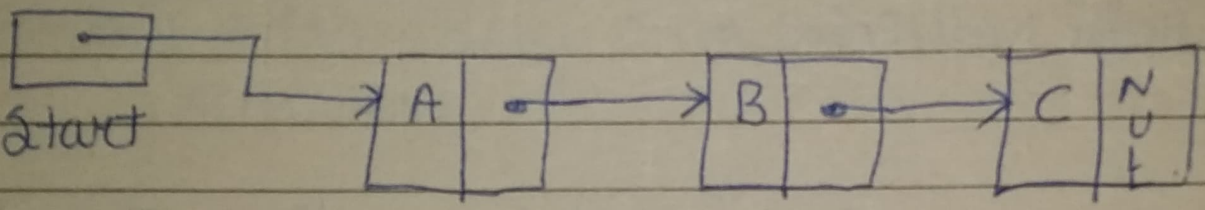
3. Linked list are used to represent sparse matrix.
   ↳ no. of elements non-zero more

L.P. → Recursive Node
R.P → Successive node

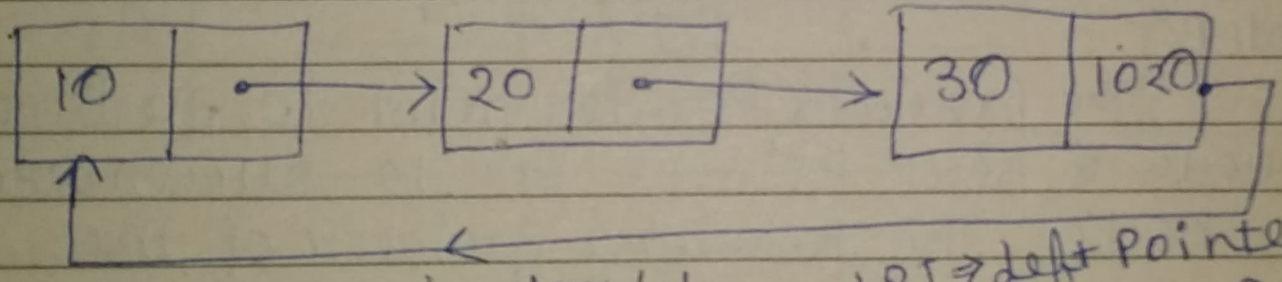
Date:⇒ 23/07/18.

Types of Linked List ⇒:

1. Singly Linked list: ⇒



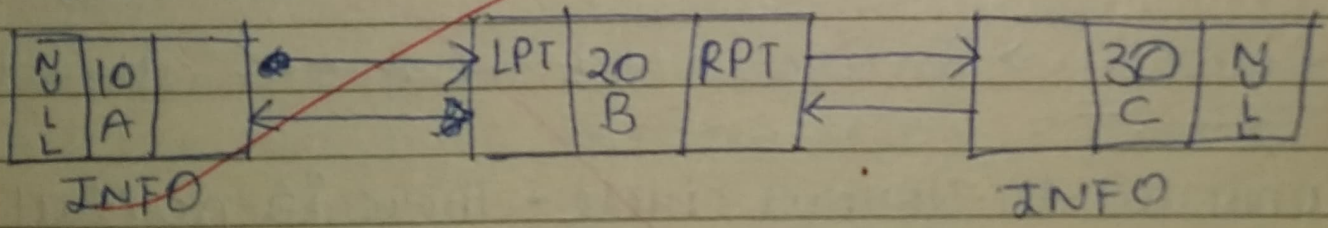
2. Circular L.L. ⇒



Circular L.L.

LPT ⇒ left Pointer
RPT ⇒ Right Pointer

3. Doubly L.L. ⇒



1. Singly - Linked list :⇒ In this linked list each node is divided into two parts. The first part contain the information & IInd part contain address of the next node in the list. Each node has a single pt. to next node. and the last node contain null Pointer.

2. Doubly L.L.⇒ are also called tobay list. In this L.L each node has two pointers previous & next Pointer.

The Previous Pointer point to predecessor node & next Pointer pt. to successor node. The previous pointer of the first node & next pointer of last node contain the null-Pointer.

3. Circular L.L.⇒ In circular L.L ~~the first~~ all nodes are connected to form a circle. There is no Null Pointer at the end of the list the elements Point to each other in a circular way which form a circular chain & the last node contain the address of the 1st Node.

16.

Date:- 28/06/20

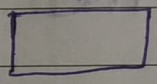
Introduction to linear Data Structures:->

why -> array concept was introduced:->

Example ->

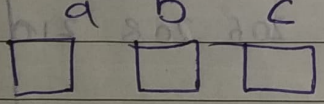
we create a int variable in C/C++

1 -> int a;



2 bytes

-> 3 -> int a, b, c;



2b 2b 2b

-> For reduce time the array concept was introduced

int a [100]

-> If we want to make same

1.

Array ->

-> Array is defined as

-> a collection of similar type elements.

-> This means an array can store either all integers or all floating Point, all characters, but all of same data type.

Some Points about array:->

-> Array are always stored in consecutive memory locations

-> Array name is actualy a Pointer to the first location of the memory blocked alloacted to the name of array

Types:→

Array

1D 2D multiple Dimensional

int a[5]; int a[2][5]; int a[2][3][3];

one-dimensional array

→ declaration →:

data_type var_name [Expression];

data type → is a type of element s to be stored in array

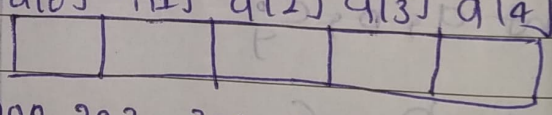
var-name → simple variable name

The Expression / Subscript specifies the no. of values stored in the array.

Ex → → → → → → → →

→ only one subscript specification is needed to specify a particular element of the array.

int a[5];



memory size of array = 5 x 2 = 10 bytes

Integer = 2 bytes
Float = 4 bytes
Char = 1 bytes

memory size of array / space allocated to array = size of * size of array

Index of first element → 0, 1, 2

Index of last element 1 less than size of element = 1

Ex → int a[2]

a[0] = 5
a[1] = 10

→ Index of last element
→ 2 − 1 = 1

Array num size =

Size = (upperbound − lowerbound) + 1

→ a[0]
a[1] → upperbound

⇒
Size = (1 − 0) + 1
= 1 + 1 = 2.

Advantages →:

→ We can directly access the data values from array using index number.

→ We can use an array to group related variable under a single name

Disadvantages →:

(1). It's necessarily to declare in advance the amount of memory to be utilized by array.

↳ Phele se hee batana padega kii kyaa size bogi array ka.

Ex → int a[n] → ; (x)

(2). It requires contactive memory locations there may be not possible if array size is large.

Formulae of Array chart from JavatPoint

Date:⇒ 29/06/20

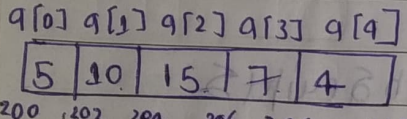
⇒

Why-linked list concept introduced:⇒

Date:⇒ 01/07/20

For Ex →:

int a[5]; → Constant size (bcoz Compiler allocate memory on compile time).



base address → 5 x 2 → 10 bytes (memory Allocate).

→ Access value of index 2

index = 2

value = a[2]

Address = Base address + Index x size of integer (In case float so float ichega)

= 200 + 2 x 2

= 204

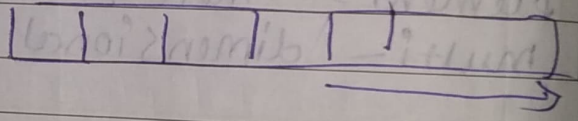
Access 3rd element of array

= 3 - 1 = 2

Drawback of array →:

(i) We should know the size of array in advance.

int a[1000];



wastage of memory.

(2) Contiguous memory allocation used in memory. → Ex → 20,000 memory location available but not at One Place

(3) Insertion & deletion are time consumed.
↓
So to remove the above drawbacks of array. Linked list concept is introduced

Date:→ 05/07/20

Linked list →

→ We store data in the form of Node. We can't tell size to create One by one node to store data.

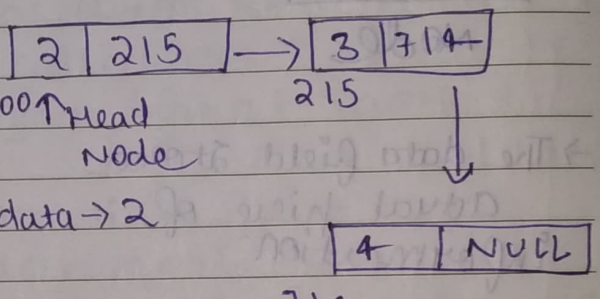
Node

data	Address

→ You can create any no. of nodes without giving size of LL in advance

Advantages:→

(i) Random memory allocation →

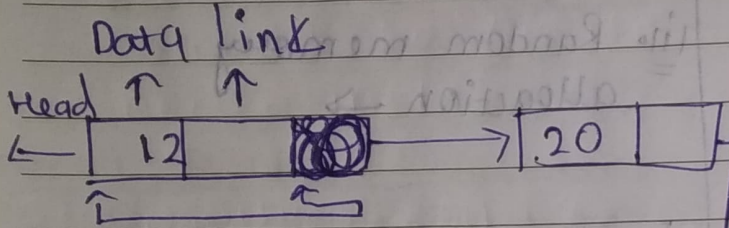


→ 2 ke baad 3 aayega Ye kese pata chalega ki jo 3 ki address hai vo 2 mae store hojayegi

Pointer → They are used to store the address of another Node.

Date:- 06/07/20

→ L.L. can be defined as collection of objects called nodes that are randomly stored in the memory.



→ The data field stores actual piece of information.

→ Link field is used to store the address of next node. (Pointer)

Advantages :→

(i). Empty node cannot be present in the linked list.

Types of L.L →:

(i). Singly L.L.

2. Doubly L.L

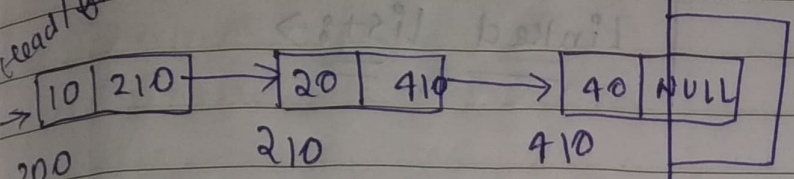
3. Circular L.L

Singly Circular L.L

Doubly Circular L.L

i) Singly L.L →:

Ex →:



Node

data	Link

→ Collections of node which contains two fields data & Link.

→ Traversal →
One direction means you can go in forward direction. not move from 20 to 10 return. For removing this doubly L.L. introduced

ii) Doubly L.L →

→ It's called singly bcoz this list consits of one link to point to next node or element.

→ The first node of list is called as Head / start / first node.

→ The link field of last node is NULL which represents end of linked list.

→ Traversal is only one direction i.e. forward direction

→ This drawback can be overcome by doubly L.L.

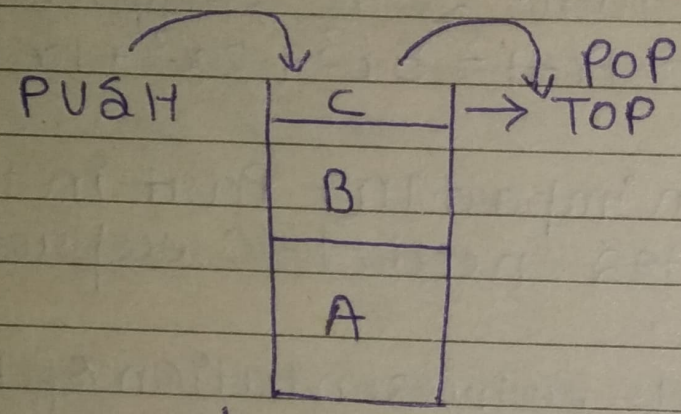
Print Reverse order → Stack mae.

Date:→ 06/08/18.

Date : Page :

Unit→2

Stacks



→ Non-Primitive D.S.
→ linear D.S.

↳ operation ⇒ Last In first out

(i). PUSH ⇒ Insert element

(ii). POP ⇒ Delete element

3. Peek or TOP ⇒ Return karega last element
4. is Empty → True → else false.

True → koi element Nahi hae
False → element Hae.

Array Representation⇒:

Implementation⇒
→ static → array
→ Dynamic → Linked list

Date: Page:

(i). linear

Stack ⇒ is a non-Primitive D.S.

(ii). Stack is a list of element in which an element may be inserted or deleted only at one ~~point~~ called top of the stack, the last ~~end~~ added element will be the first to be remove from the stack that is the reason the stack is also called "LIFO".

Two basic operation associate with stack⇒

(i). PUSH operation → is used to insert an element into a stack.

(ii). POP operation → is used to delete an element from a stack.

And ~~other~~ other operation

3. Peek or TOP ⇒ Return Top element of stack.

4. Is empty ⇒ Return True if stack is Empty as false

Advantages ⇒

1. Stack provides unique way to work with continuous memory.

2. Stack provide a way to access different of continuous data in a "LIFO" manner

Disadvantages ⇒

1. Inflexible

2. Unable to Copy & Paste

Applications of Stacks ⇒

1. Reverse a string

2. Recursion

3. Polish Notation

4. Expression evaluation

5. Back tracking & two operation.

Date :⇒ 08/08/18

Applications of Stack :⇒

(i). Polish Notation :⇒
- → Infix Notation
- → Prefix Notation
- → Postfix Notation

1. Infix Notation :⇒ Operator operant ke baad
   x + y * m

2. Prefix ⇒ Operator operatant ke Bhele
   + xy

3. Postfix ⇒ xy +

Remember ... return when the function has to return.

Recursion make use of system stack for storing the written address of the function call.

Queue:⇒

⇒ linear D.S
→ Non-primitive D.S
→ FIFO
→ Rear (Insert → store)
→ Front (Delete → Process)

A
Queue is an linear list of elements in which deletion can take place only at one end called the "front end" and insertion can take place only at the other end called "Rear end".

Queue follows the "FIFO" methodology that is the data item stored

4. Static & dyn implementation of Queue-2

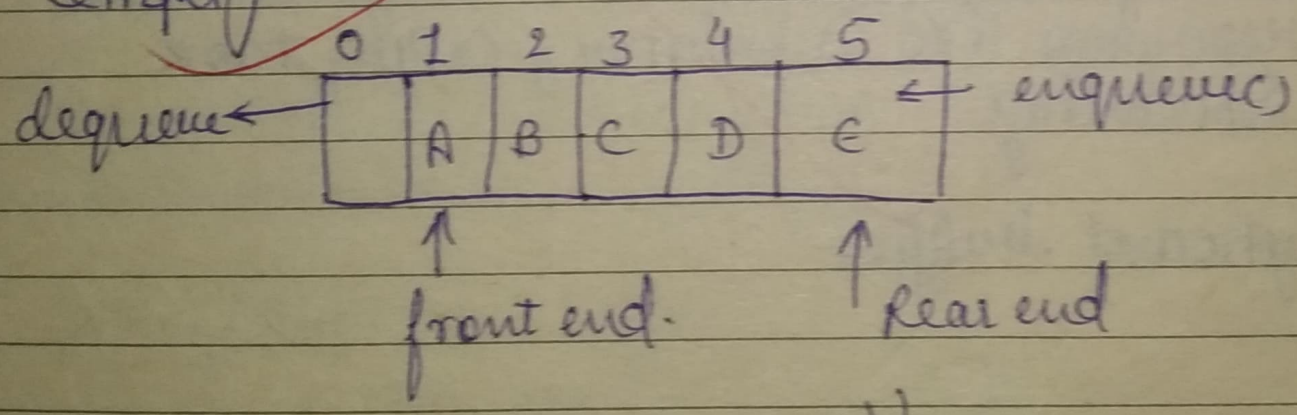
Date: Page:

2.

first will be access first.

Basic operations of Queue ⇒

- enqueue ⇒ The process to add an element into Queue is called enqueue.
- Dequeue ⇒ The process of removal of an element from the Queue is called Dequeue
- Peek() ⇒ Gets the element add the front of the queue without removing it.
- isfull() ⇒ check if the Queue is full.
- isempty() ⇒ check if the Queue is empty.



Binary search
sorting
searching
Red black tree
} etc.

5. following

Application of Queue →

(i). Real life

(ii). Computer Science

(i). Real life :→

(i). Call center Phone System

(ii). Waiting in line.

2. Computer Science related →:

→ Round Robin Scheduling
→ Handle of Interupttts
→ Serving a request on a single shared resources like a printer, C.P.U., task Scheduling
→ Keyboard buffer.

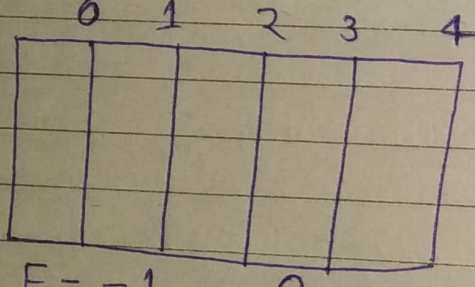
Date :⇒ 17/09/18

Types of Queue⇒

1. Simple or linear Queue
2. Circular queue
3. Deque (double ended queue)
4. Priority queue.

1. Simply or linear queue⇒:

(i).



F = -1
R = -1

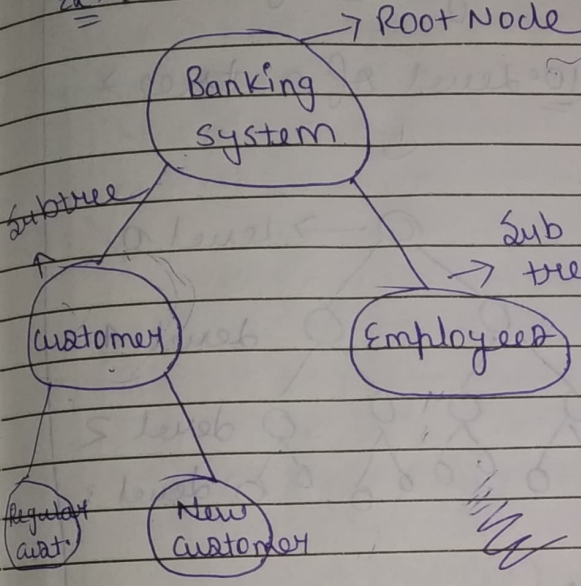
Queue empty

Simple queue defined the simple operation of the queue in which insertion occurs at the rare end of list & deletion occurs at the front of the list

Tree

→ Trees are basically used to represent the data object in hierachical manner.

Ex→:



Definition:→

→ Tree is a finite set of one or more data items(nodes) such that.

1. There is special data item called the root of the tree.

2. And its remaining data items are Partitioned into no. of mutually exclusive subsets, each of which is itself a tree. They are called subtree.

Tree Terminology:→

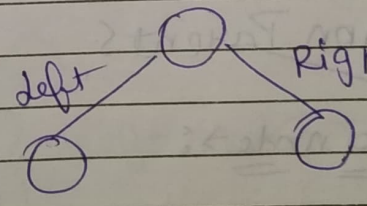
→ There are no. of terms associated with trees which are listed below:→

(i) Root

→ It is a unique node in the tree to which further subtrees are attached.

(ii) Node → Each data item in a tree is called node. It specifies the data information & links (branches) to other data items.

3. Parent Node →: The node having further subbranches.



4. Child Node →: The node having no further subtrees.