Data Structures and Algorithms  **Linked List**

1. What is a linked list?

* A linked list is a sequence of data structures, which are connected together through links.
* Linked List is a sequence of links which contains data+link to next node.
* Linked list is the second most-used data structure after array.
* Below are the important terms to understand the concept of Linked List.
* Link − Each link of a linked list can store a data called an element.
* Next − Each link of a linked list contains a link to the next link called Next.
* LinkedList − A Linked List contains the connection link to the first link called First.
* Linked List contains a link element called first.
* Each link carries a data field(s) and a link field called next.
* Each link is linked with its next link using its next link.
* Last link carries a link as null to mark the end of the list.

|  |
| --- |
| NULL |

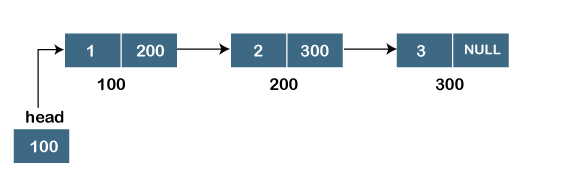
Head

Data Next Data Next Data Next Data Next Data Next

2. What are the different forms of linked lists?

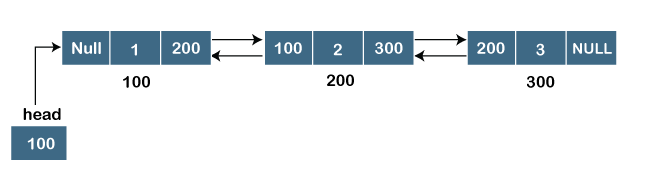
**Singly linked list**

* It is a data structure that contains two parts, i.e., one is the data part, and the other one is the address part, which contains the address of the next or the successor node. The address part in a node is also known as a pointer.
* Suppose we have three nodes, and the addresses of these three nodes are 100, 200 and 300 respectively. The representation of three nodes as a linked list is shown in the below figure:



**Doubly linked list**

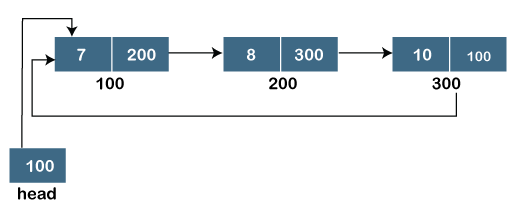
* doubly linked list contains two pointers. We can define the doubly linked list as a linear data structure with three parts: the data part and the other two address part. In other words, a doubly linked list is a list that has three parts in a single node, includes one data part, a pointer to its previous node, and a pointer to the next node.
* Suppose we have three nodes, and the address of these nodes are 100, 200 and 300, respectively. The representation of these nodes in a doubly-linked list is shown below:



In above screen shot the node in a doubly-linked list has two address parts; one part stores the address of the next while the other part of the node stores the previous node's address. The initial node in the doubly linked list has the NULL value in the address part, which provides the address of the previous node.

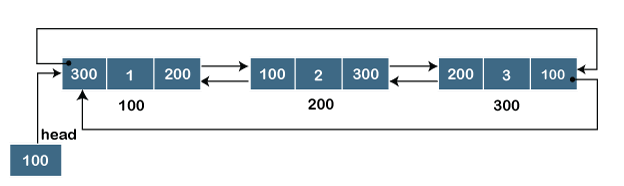
**Circular linked list**

* A circular linked list is a sequence of elements in which each node has a link to the next node, and the last node is having a link to the first node.
* A circular linked list is a variation of a singly linked list.
* The only difference between the singly linked list and a circular linked list is that the last node does not point to any node in a singly linked list, so its link part contains a NULL value.
* On the other hand, the circular linked list is a list in which the last node connects to the first node, so the link part of the last node holds the first node's address.
* The circular linked list has no starting and ending node. We can traverse in any direction, like either backward or forward. Below is the sample screen shot eg.



**Doubly Circular linked list**

The doubly circular linked list has the features of both the circular linked list and doubly linked list.



* The above sample eg in screen shot shows the representation of the doubly circular linked list in which the last node is attached to the first node and thus creates a circle. It is a doubly linked list also because each node holds the address of the previous node also.
* The main difference between the doubly linked list and doubly circular linked list is that the doubly circular linked list does not contain the NULL value in the previous field of the node. As the doubly circular linked contains three parts, i.e., two address parts and one data part so its representation is similar to the doubly linked list.

3. What is a linked list's purpose?

* Linked list can be used in implementation of stacks and queues
* Implementation of graphs : Adjacency list representation of graphs is most popular which is uses linked list to store adjacent vertices.
* Dynamic memory allocation : We use linked list of free blocks.
* Maintaining directory of names
* Performing arithmetic operations on long integers
* Manipulation of polynomials by storing constants in the node of linked list
* representing sparse matrices

4. What are the advantages of linked lists over arrays?

Linked list provides the following two advantages over arrays   
1) Dynamic size we can grow or decrease linked list size as per requirements dynamically  
2) Ease of insertion/deletion  - we can easily delete a node in between the linked list

3) They can be used as underlying data structures for search trees

4) No memory wastage

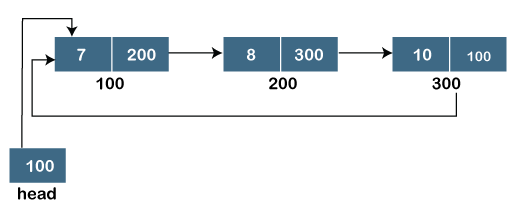
5. What is the purpose of a circular linked list?

* Its is used for the implementation of queue.
* Any node can be a starting point.
* We can traverse the whole list by starting from any point.
* We just need to stop when the first visited node is visited again.
* Circular lists are useful in applications to repeatedly go around the list.
* It saves time when we have to go to the first node from the last node.
* Reference to previous node can easily be found.
* When we want a list to be accessed in a circle or loop then circular linked list are used.

6. How will you explain Circular Linked List?

**Circular linked list**

* A circular linked list is a sequence of elements in which each node has a link to the next node, and the last node is having a link to the first node.
* A circular linked list is a variation of a singly linked list.
* The only difference between the singly linked list and a circular linked list is that the last node does not point to any node in a singly linked list, so its link part contains a NULL value.
* On the other hand, the circular linked list is a list in which the last node connects to the first node, so the link part of the last node holds the first node's address.
* The circular linked list has no starting and ending node. We can traverse in any direction, like either backward or forward. Below is the sample screen shot eg.



Below are few advantages:

* No requirement for a NULL assignment in the code.
* The circular list never points to a NULL pointer unless fully deallocated.
* Circular linked lists are advantageous for end operations since beginning and end coincide.
* Algorithms such as the Round Robin scheduling can neatly eliminate processes which are queued in a circular fashion without encountering dangling or NULL-referential pointers.
* Circular linked list also performs all regular functions of a singly linked list. In fact, circular doubly linked lists discussed below can even eliminate the need for a full-length traversal to locate an element. That element would at most only be exactly opposite to the start, completing just half the linked list.