Top of Form

**Linked List:**

A linked list is a sequence of data structures, which are connected together via links.

Linked List is a sequence of links which contains items. Each link contains a connection to another link. Linked list is the second most-used data structure after array. Following 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 Representation**

Linked list can be visualized as a chain of nodes, where every node points to the next node.

As per the above illustration, following are the important points to be considered.

* 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.

**Types of Linked List**

Following are the various types of linked list.

* Simple Linked List − Item navigation is forward only.
* Doubly Linked List − Items can be navigated forward and backward.
* Circular Linked List − Last item contains link of the first element as next and the first element has a link to the last element as previous.

**Basic Operations**

Following are the basic operations supported by a list.

* Insertion − Adds an element at the beginning of the list.
* Deletion − Deletes an element at the beginning of the list.
* Display − Displays the complete list.
* Search − Searches an element using the given key.
* Delete − Deletes an element using the given key.

**Insertion Operation**

Adding a new node in linked list is a more than one step activity. We shall learn this with diagrams here. First, create a node using the same structure and find the location where it has to be inserted.



Imagine that we are inserting a node B (NewNode), between A (LeftNode) and C (RightNode).

Then point B.next to C −

NewNode.next −> RightNode;

It should look like this −



Now, the next node at the left should point to the new node.

LeftNode.next −> NewNode;



This will put the new node in the middle of the two. The new list should look like this −



Similar steps should be taken if the node is being inserted at the beginning of the list. While inserting it at the end, the second last node of the list should point to the new node and the new node will point to NULL.

**Deletion Operation**

Deletion is also a more than one step process. We shall learn with pictorial representation. First, locate the target node to be removed, by using searching algorithms.



The left (previous) node of the target node now should point to the next node of the target node−

LeftNode.next −> TargetNode.next;



This will remove the link that was pointing to the target node. Now, using the following code, we will remove what the target node is pointing at.

TargetNode.next −> NULL;

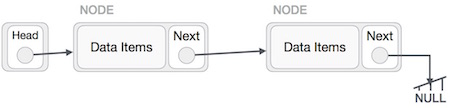


We need to use the deleted node. We can keep that in memory otherwise we can simply deallocate memory and wipe off the target node completely.

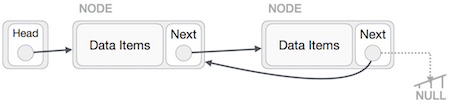
****

**Reverse Operation**

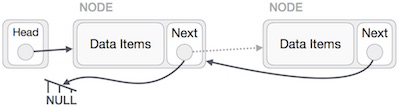
This operation is a thorough one. We need to make the last node to be pointed by the head node and reverse the whole linked list.



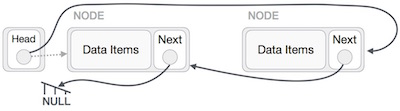
First, we traverse to the end of the list. It should be pointing to NULL. Now, we shall make it point to its previous node −



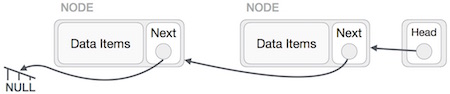
We have to make sure that the last node is not the last node. So we'll have some temp node, which looks like the head node pointing to the last node. Now, we shall make all left side nodes point to their previous nodes one by one.



Except the node (first node) pointed by the head node, all nodes should point to their predecessor, making them their new successor. The first node will point to NULL.



We'll make the head node point to the new first node by using the temp node.



The linked list is now reversed.

**Doubly Linked List:**

Doubly Linked List is a variation of Linked list in which navigation is possible in both ways, either forward and backward easily as compared to Single Linked List. Following are the important terms to understand the concept of doubly 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.
* Prev − Each link of a linked list contains a link to the previous link called Prev.
* LinkedList − A Linked List contains the connection link to the first link called First and to the last link called Last.

**Doubly Linked List Representation**



As per the above illustration, following are the important points to be considered.

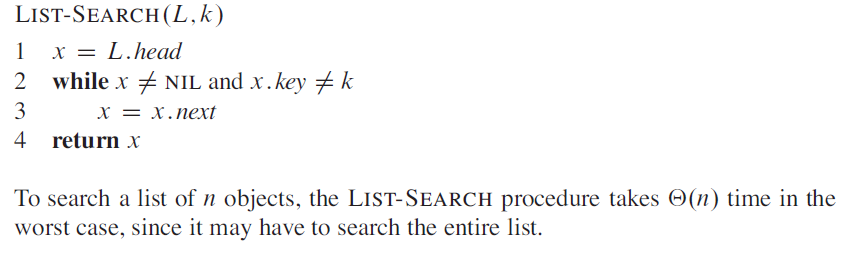
* Doubly Linked List contains a link element called first and last.
* Each link carries a data field(s) and two link fields called next and prev.
* Each link is linked with its next link using its next link.
* Each link is linked with its previous link using its previous link.
* The last link carries a link as null to mark the end of the list.

**Basic Operations**

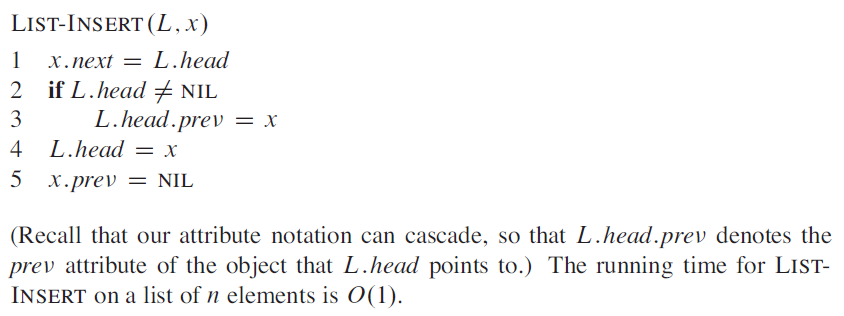
Following are the basic operations supported by a list.

* Insertion − Adds an element at the beginning of the list.
* Deletion − Deletes an element at the beginning of the list.
* Insert Last − Adds an element at the end of the list.
* Delete Last − Deletes an element from the end of the list.
* Insert After − Adds an element after an item of the list.
* Delete − Deletes an element from the list using the key.
* Display forward − Displays the complete list in a forward manner.

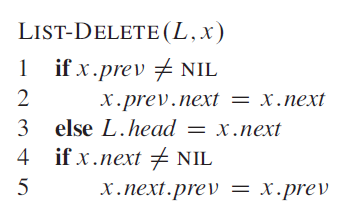
**Searching a linked list:**



**Inserting into a linked list**



**Deleting from a linked list**



Bottom of Form

Top of Form

**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.

https://media.geeksforgeeks.org/wp-content/uploads/CircularLinkeList.png

**Advantages of Circular Linked Lists:**

**1)**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.

**2)**Useful for implementation of queue. Unlike this implementation, we don’t need to maintain two pointers for front and rear if we use circular linked list. We can maintain a pointer to the last inserted node and front can always be obtained as next of last.

**3)** Circular lists are useful in applications to repeatedly go around the list. For example, when multiple applications are running on a PC, it is common for the operating system to put the running applications on a list and then to cycle through them, giving each of them a slice of time to execute, and then making them wait while the CPU is given to another application. It is convenient for the operating system to use a circular list so that when it reaches the end of the list it can cycle around to the front of the list.

**4)** Circular Doubly Linked Lists are used for implementation of advanced data structures like Fibonacci Heap.

**RELEVANT READING MATERIAL AND REFERENCES:**

**Source Notes:**

1. <https://www.tutorialspoint.com/data_structures_algorithms/linked_list_algorithms.htm>
2. <https://www.tutorialspoint.com/data_structures_algorithms/doubly_linked_list_algorithm.htm>
3. <https://www.geeksforgeeks.org/circular-linked-list/>

**Lecture Video:**

1. <https://www.youtube.com/watch?v=QaWkT-ZUwxA>
2. <https://www.youtube.com/watch?v=IKrNp3yqZh8>
3. <https://www.youtube.com/watch?v=CxOtQ11_qPM>
4. <https://www.youtube.com/watch?v=7I5eg7lyMYk>

**Online Notes:**

1. <http://www.crectirupati.com/sites/default/files/lecture_notes/ds%20ln.pdf>
2. <http://www.vssut.ac.in/lecture_notes/lecture1428550942.pdf>

**Text Book Reading:**

1. Cormen, Leiserson, Rivest, Stein, “*Introduction to Algorithms*”, Prentice Hall of India, 3rd edition 2012. problem, Graph coloring.
2. Lipschutz, S., “*Data Structures, Schaum's Outline Series*”, Tata McGraw Hill.

**Online Notes:**

1. <http://www.crectirupati.com/sites/default/files/lecture_notes/ds%20ln.pdf>
2. <http://www.vssut.ac.in/lecture_notes/lecture1428550942.pdf>

**Text Book Reading:**

1. Cormen, Leiserson, Rivest, Stein, “*Introduction to Algorithms*”, Prentice Hall of India, 3rd edition 2012. problem, Graph coloring.
2. Lipschutz, S., “*Data Structures, Schaum's Outline Series*”, Tata McGraw Hill.

**Online Book Reference:**

1. <https://www.edutechlearners.com/download/books/DS.pdf>
2. <https://www.edutechlearners.com/download/books/DS.pdf>

**In addition: PPT can be also be given.**