

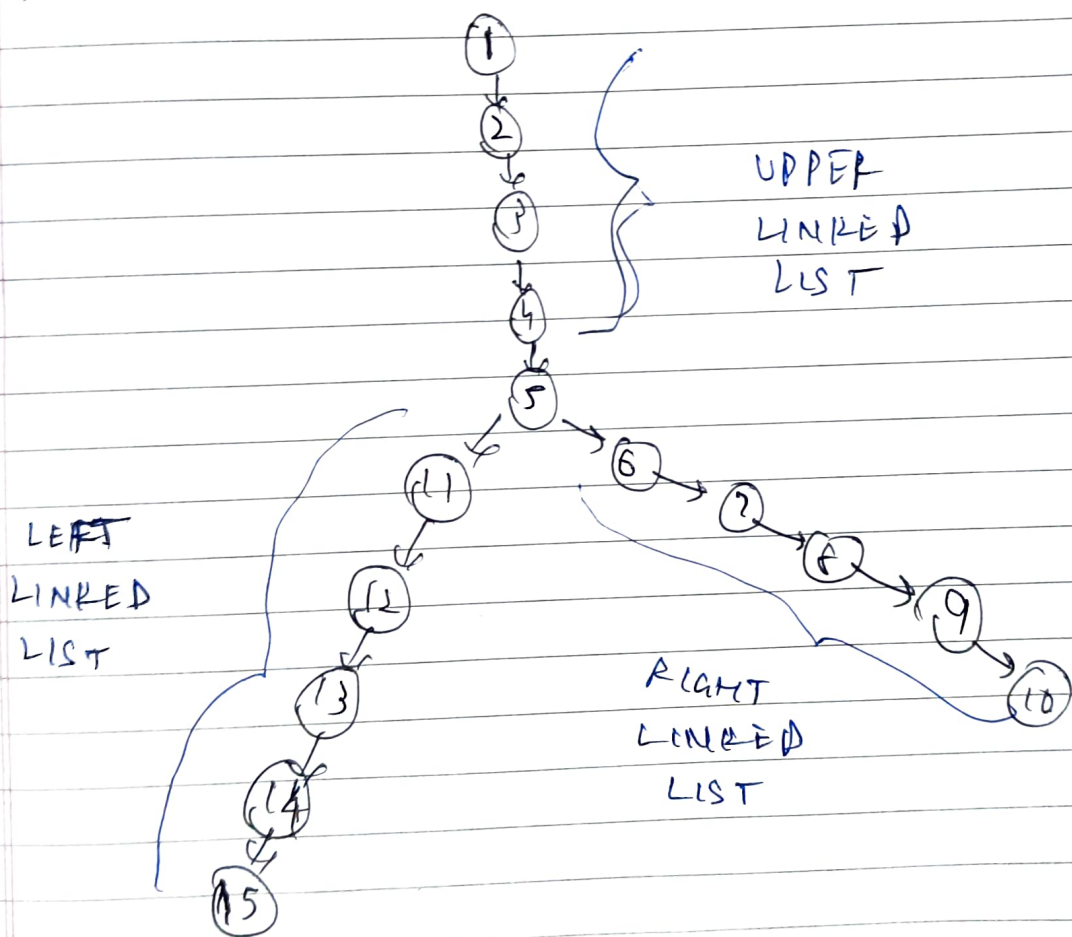
SACHIN PRASANNA

2111T058

Nodes

~~links~~ are indexed in the following way.

Count from top and when the root (middle) node is encountered, count the right part of linked list, and then left part.



Insertion

In an already existing 15 node linked list, in the worst case, insertion will happen in 15th node

(as numbered in diagram).

r If we insert a new node on the 15th (left leaf node), then we have traversed all the elements.

r In that case, the time complexity will become $O(n)$ where n is the size of linked list.

Worst case Time Complexity $\rightarrow O(n)$

★ Deletion

r Similarly as in insertion, we take the position of the node to be deleted, which ~~can~~ be the leaf node in left linked list.

r Again, all the nodes are traversed, which makes the time complexity $O(n)$, where n is the size of the linked list.

Worst case Time Complexity $\rightarrow O(n)$

★ Search

r Similarly as in insertion and deletion, we traverse Upper linked list, then Right linked list, then finally

SACHIN PRASANNA

2111T058

Left linked list.

If the element to be searched is the last element of the left linked list, then all elements will be traversed in worst case.

So,

Worst case Time Complexity $\rightarrow O(n)$ 

A diagram showing a left-linked list. It consists of three nodes, each represented by an 'x'. The nodes are connected by horizontal lines pointing from right to left. The first node on the right has an arrow pointing to the second node, and the second node has an arrow pointing to the third node on the left.