Analysis of Linked Lists

Linked list is a dynamic data structures that allows for fast insertion and rearrangement of data. It also avoids allocating extra memory, as is in the case for array based list implementation. However, it requires extra space for next/previous pointers and needs careful book-keeping.

Singly Linked List

Singly Linked List is a linked list where linkage is only one directional. Each node has a pointer that points to the next node. The table below shows complexities of different operations:

Operations	Complexity
insert	$O(1)$
find_next	$O(1)$
is_empty	$O(1)$
find	$O(N)$
find_prev	$O(N)$
delete	$O(N)$
length	$O(N)$
last	$O(N)$
kth	$O(N)$
make_empty	$O(N)$

Doubly Linked List

Doubly Linked List is a linked list where linkage is bi-directional. Each node has two pointers, one pointing to the next node and the other pointing to the previous node. The table below shows complexities of different operations.

Operations	Complexity
insert	$O(1)$

Operations	Complexity
find_next	$O(1)$
is_empty	$O(1)$
find	$O(N)$
find_prev	$O(1)$
delete	$O(1)$
length	$O(N)$
last	$O(N)$
kth	$O(N)$
make_empty	$O(N)$

Due to the availability of the pointer $previous$, $delete$ and $findprev$ operations now run in constant time.

Cirularly Linked List

Cirularly Linked list could be Singly or Doubly Linked List. The distinction is in that the last node's $next$ pointer points to the first node and first node's $previous$ pointer points to the last node (if Doubly Linked). Thus, $last$ operation now runs in constant time $O(1)$. Circularly Linked List is particularly useful when you need to iterate over all nodes starting from any node.