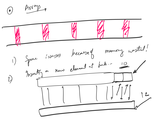
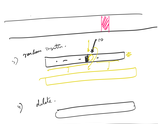
① Linked List.

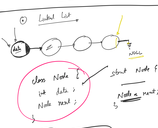
① Array.



1) Space: wasteful of memory unused!
2) Insertion = have allocated it first.

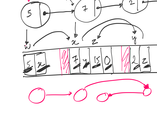


3) random insertion.



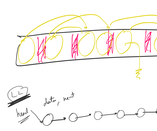
4) deletion.

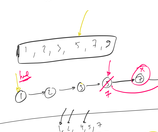
② Linked List

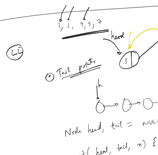


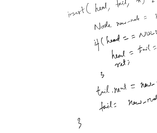
class Node {
int data;
Node next;
};

struct Node {
Node * next;
};





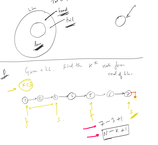




① Tail pointer.

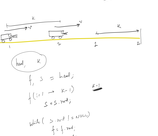
Node head, tail = NULL;

insert(head, tail, n) {
Node new_node = new node(n);
if(head == NULL) {
head = tail = new_node;
};
tail.next = new_node;
tail = new_node;
}

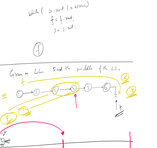


void(head, previous) {

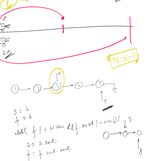
previous.next = previous.next.next;
}

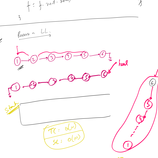


② Given a LL. Find the length of the LL.



③ Given a LL. Find the nth node from end of LL.



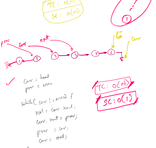


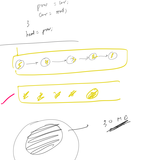
head, n

f, s = head;
f = f.next;
for(i = 1; i < n; i)
f = f.next;

while(f != NULL)
f = f.next;
s = s.next;

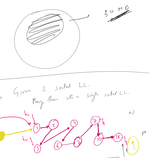
④ Given a LL. Find the middle of the LL.

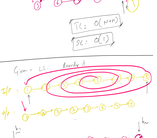




s = f
f = f.next
while(f != NULL && f.next != NULL) {
f = f.next.next;
s = s.next;
}

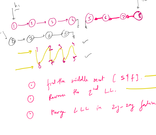
⑤ Given a LL.





curr = head
prev = NULL
while(curr != NULL {
tmp = curr.next;
curr.next = prev;
prev = curr;
curr = tmp;
}
head = prev;

TC: O(n)
SC: O(1)



⑥ Given 2 sorted LL.
Merge them into a single sorted LL.

⑦ Given a Binary Tree.

① find the middle node (5 + 1) → O(n)
② Reverse the 2nd LL. → O(n)
③ Merge 2 LL in zig-zag fashion → O(n)

TC: O(n)
SC: O(1)