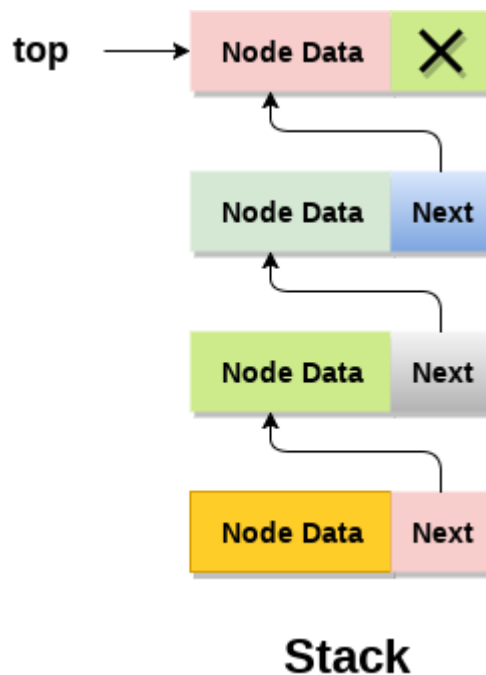


# Linked list implementation of stack

Instead of using array, we can also use linked list to implement stack. Linked list allocates the memory dynamically. However, time complexity in both the scenario is same for all the operations i.e. push, pop and peek.

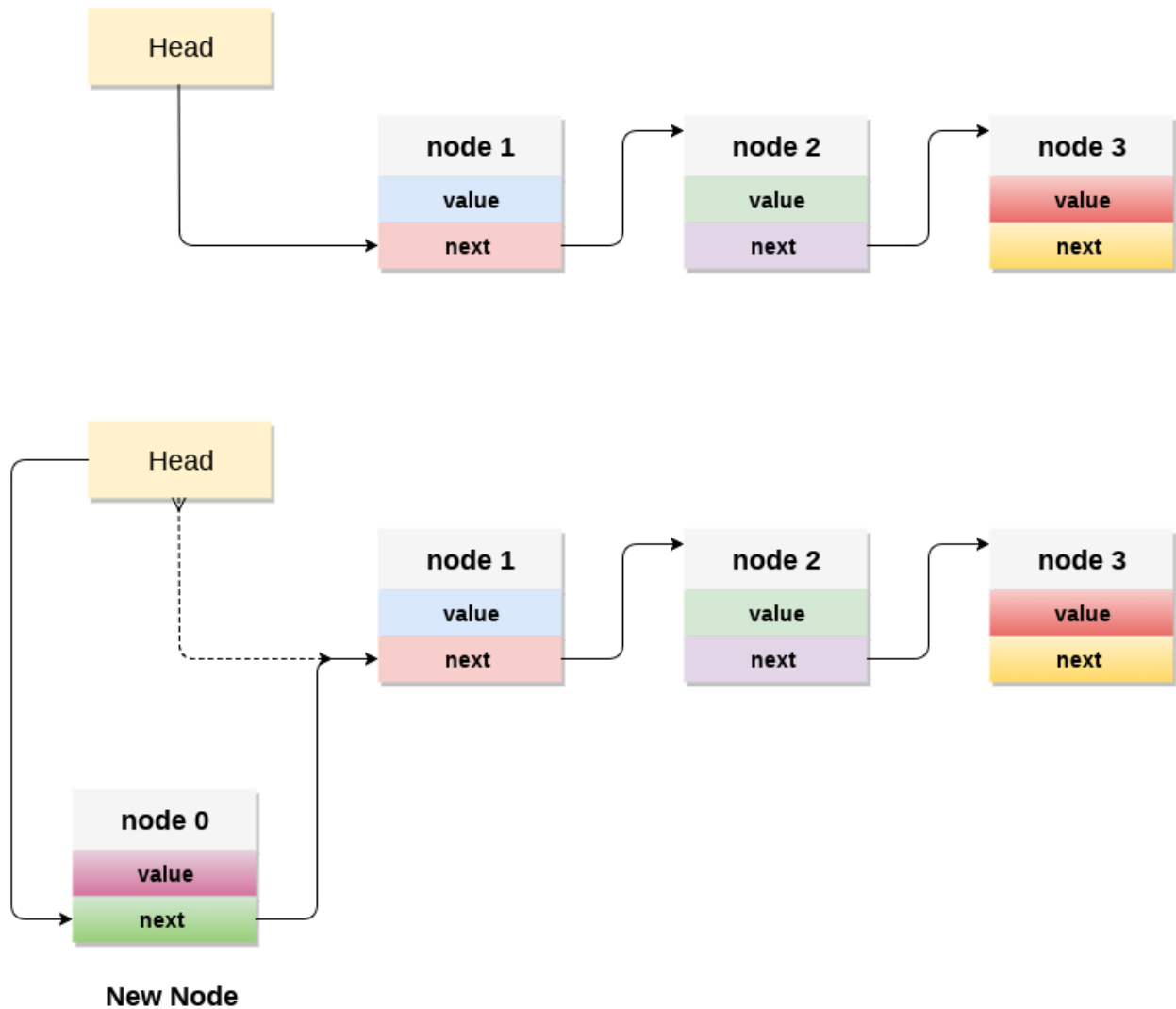
In linked list implementation of stack, the nodes are maintained non-contiguously in the memory. Each node contains a pointer to its immediate successor node in the stack. Stack is said to be overflown if the space left in the memory heap is not enough to create a node.



## Adding a node to the stack (Push operation)

Adding a node to the stack is referred to as **push** operation. Pushing an element to a stack in linked list implementation is different from that of an array implementation. In order to push an element onto the stack, the following steps are involved.

1. Create a node first and allocate memory to it.
2. If the list is empty then the item is to be pushed as the start node of the list. This includes assigning value to the data part of the node and assign null to the address part of the node.
3. If there are some nodes in the list already, then we have to add the new element in the beginning of the list (to not violate the property of the stack). For this purpose, assign the address of the starting element to the address field of the new node and make the new node, the starting node of the list.



### Deleting a node from the stack (POP operation)

Deleting a node from the top of stack is referred to as **pop** operation. Deleting a node from the linked list implementation of stack is different from that in the array implementation. In order to pop an element from the stack, we need to follow the following steps :

1. **Check for the underflow condition:** The underflow condition occurs when we try to pop from an already empty stack. The stack will be empty if the head pointer of the list points to null.
2. **Adjust the head pointer accordingly:** In stack, the elements are popped only from one end, therefore, the value stored in the head pointer must be deleted and the node must be freed. The next node of the head node now becomes the head node.

## Display the nodes (Traversing)

Displaying all the nodes of a stack needs traversing all the nodes of the linked list organized in the form of stack. For this purpose, we need to follow the following steps.

1. Copy the head pointer into a temporary pointer.
2. Move the temporary pointer through all the nodes of the list and print the value field attached to every node.