CS101 - Data Abstraction
DS Basics - Module4

Aravind Mohan

Allegheny College

April 27, 2021

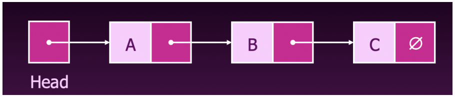
Doubly Linked List

- **Doubly Linked List:** A series of two-way connected data items called Nodes.
- A **Node** contains at least a piece of data item (of any type), a link (pointer) to the next node in the list and a link (pointer) to the previous node in the list.

Doubly Linked List

Five Properties

- **All nodes** should be linked to each other.
- **Head** pointer to the first node.
- **Tail** pointer to the last node.
- **Next** of last node points to null.
- **Previous** of first node points to null.



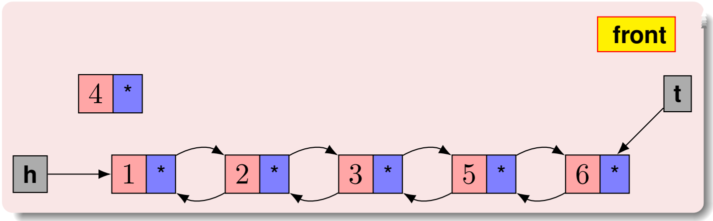
Pros and Cons of Doubly Linked List over Singly Linked List

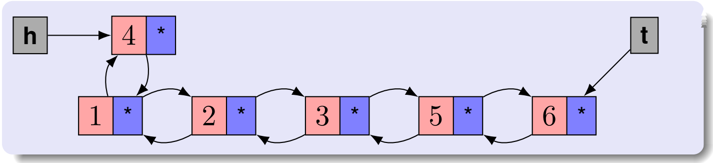
- **Traverse** both directions in Doubly Linked List. Efficient to process elements in both forward and reverse order compared to Singly Linked List.
- **Better** insert and delete performance compared to Singly Linked List. Similar idea from previous point!
- **Worst** space management. Takes up more space compared to Singly Linked List. Implementation has an additional pointer and this take up space.

Core Operations on Doubly Linked List

- **IsEmpty** determine whether or not the list is empty.
- **Insert** inserts a new node at the front, end, and/or a particular position.
- **Search** find a node with a given value.
- **Delete** delete a node with a given value.
- **Display** print all the nodes in the list.

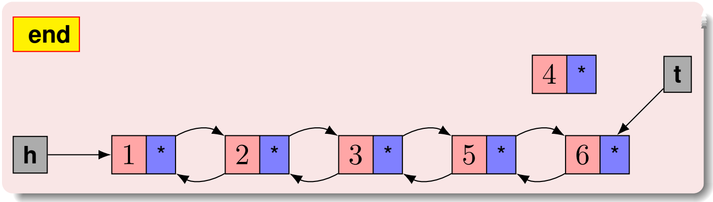
Doubly Linked List - Insert (Front)

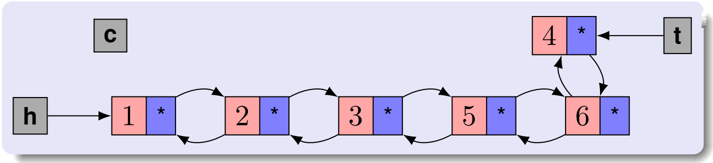




1. Create a new temp node with the value to be inserted.
2. Assign next of the temp node to the node pointed by the next of the head node N(0).
3. Assign previous of the node pointed by the next of the head node N(0) to the temp node.
4. Assign next of the head node N(0) to temp node.

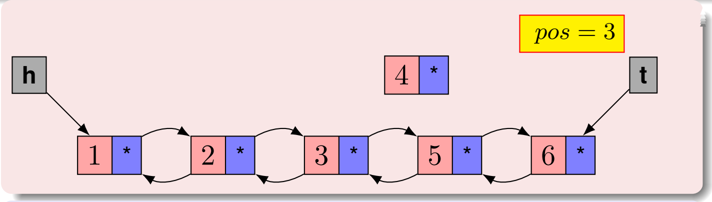
Doubly Linked List - Insert (End)

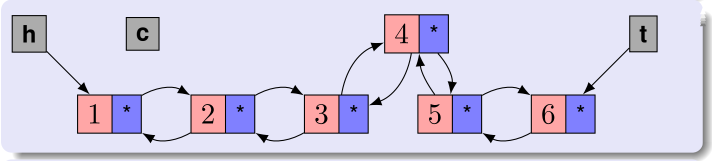




1. Create a new temp node with the value to be inserted.
2. Traverse through the list till last node using current pointer.
3. Assign next of the last node to the temp node.
4. Assign previous of the temp node to the last node.
5. Assign next of the tail node to the temp node.

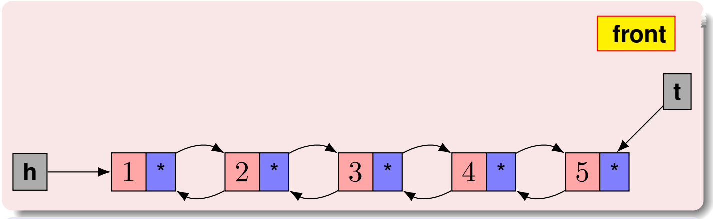
Doubly Linked List - Insert (Specific Position)

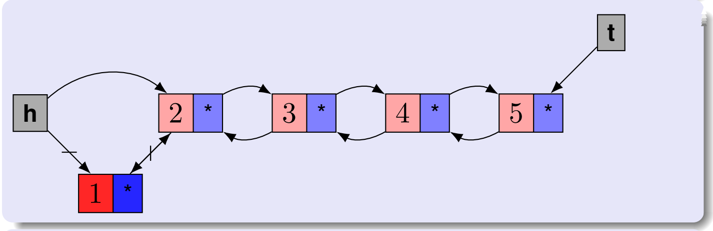




1. Create a new temp node with the value to be inserted.
2. Traverse through the list till $pos - 1$ using current pointer.
3. Assign next of the temp node to the node pointed by the next of N($pos - 1$)
4. Assign previous of the node pointed by the next of N($pos - 1$) to the temp node.
5. Assign next of N($pos - 1$) to the temp node.
6. Assign previous of the temp node to N($pos - 1$).

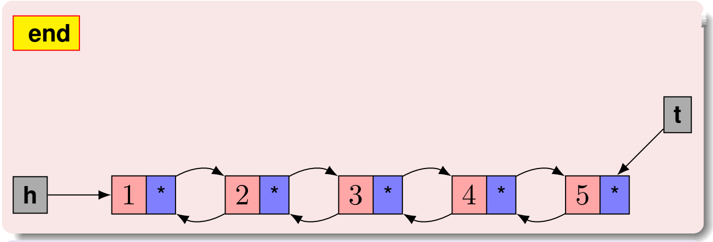
Doubly Linked List - Delete (Front)

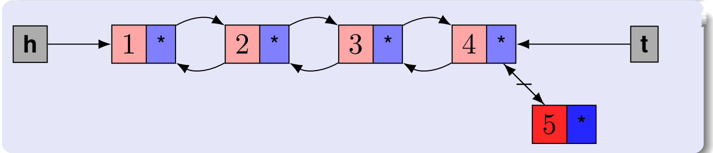




1. Assign next of the head node N(0) to node pointed by the next of the first node.
2. Assign previous of the node pointed by the next of the first node to None.
3. Free up the first node.

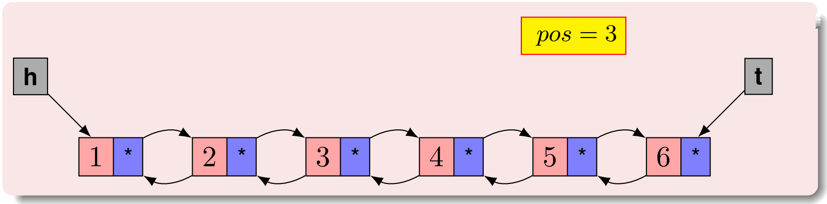
Doubly Linked List - Delete (End)

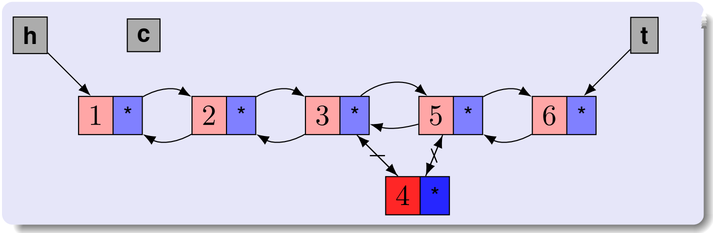




1. Traverse through the list till the previous of last node using current pointer.
2. Assign next of the previous of last node to null reference.
3. Assign previous of the last node to null reference.
4. Assign the tail node to the previous of last node.
5. Free up the last node.

Doubly Linked List - Delete (Specific Position)





1. Figure out the implementation steps on your own based on the diagram?

Implementation ...

- **PS** the dll folder in course repo.

Next ...

- **Stacks, Queues**

Reading Assignment

- **GT** Chapter 7 - 7.1, 7.3

Questions?

Please ask if there are any Questions!