Chapter 4: Linked List
Part II

Department of Computer Science and Engineering
Kathmandu University

Contents

- Implementation of Stack
- Implementation of Queue
- Circularly Linked List
- Doubly Linked List

Implementation of Stack using Linked List

- addToHead for push() operation
- removeFromHead for pop() operation
- HEAD->info for top()/peek() operation

Implementation of Queue using Linked List

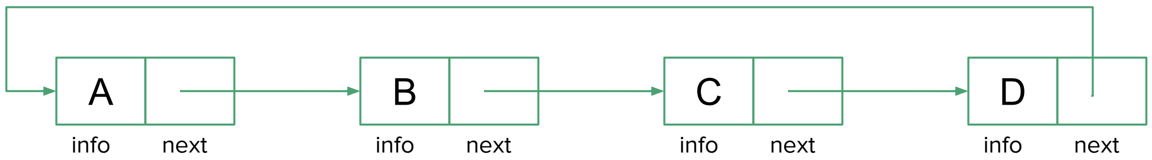
- addToTail for enqueue() operation
- removeFromHead for dequeue() operation
- HEAD->info for front() operation
- TAIL->info for rear() operation

Circularly Linked List

In a circularly linked list, the last node points to the first node.

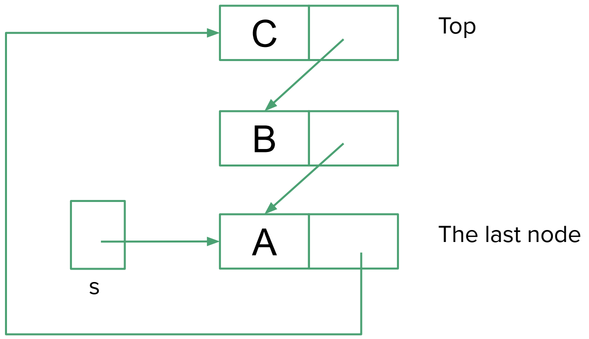
From any point in a circular list, it is possible to reach any other point in the list.

A circularly linked list does not have a natural first or last node. We must, therefore, establish first and last node by convention.



Stack as a Circularly Linked List

Let `s` be a pointer to the last node of a circularly linked list and let us adopt the convention that the first node is the top of the stack.



Stack as a Circularly Linked List: Algorithms

Algorithm: push(data)

Input: Stack(s)

Output:

Steps:

1. Create a new node, newNode
2. newNode->info = data
3. If the stack is empty
    a. s = newNode
4. else
    a. newNode->next = s->next
5. Endif
6. s->next = newNode

Stack as a Circularly Linked List: Algorithms

Algorithm: pop

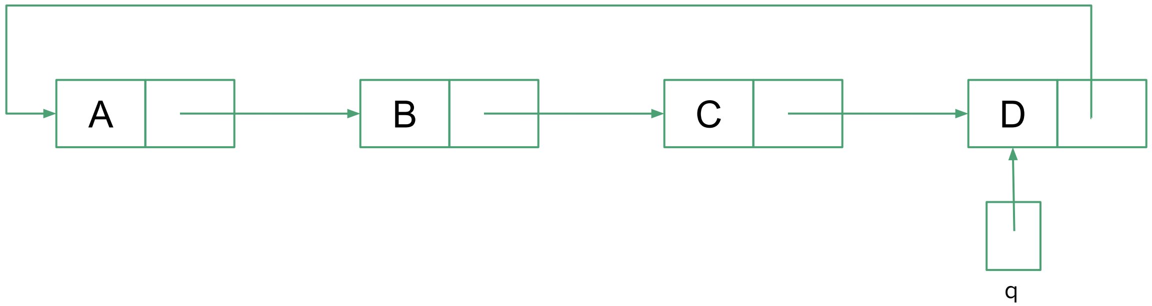
Steps:

1. If the stack is empty
    a. Print Stack underflow message
2. else
    a. top = s->next
    b. Data = top->info
    c. if (top == s) // Only one element in the stack
        i. s = NULL
    d. else
        i. s->next = top->next
    e. endif
    f. Remove top
    g. Return data
3. Endif

Queue as a Circularly Linked List

By using a circularly linked list, a queue may be specified by a single pointer to the list.

Let q be a pointer to the last inserted node of a circularly linked list.



Queue as a Circularly Linked List: Algorithms

Operations isEmpty and dequeue/remove are identical with those of stack.

Queue as a Circularly Linked List: Algorithms

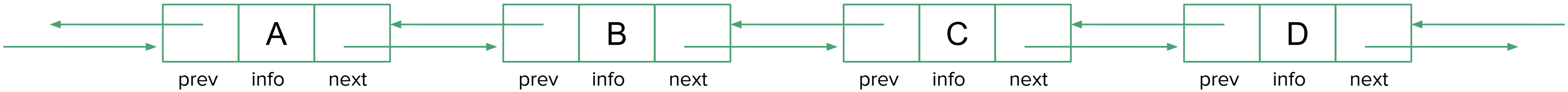
Algorithm: enqueue(data)

Steps:

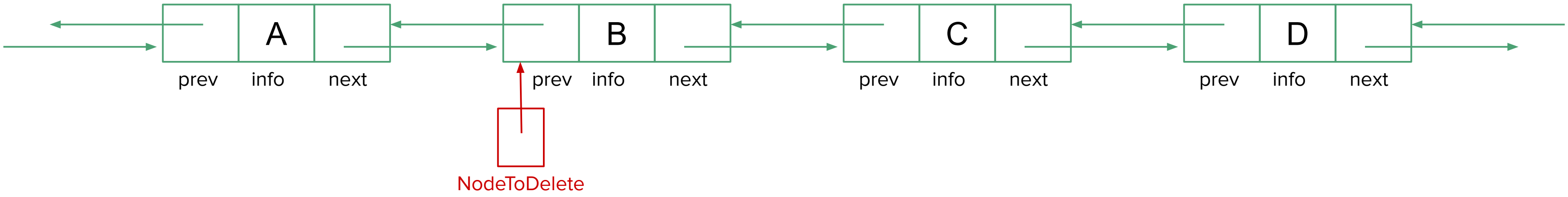
1. Create a new node, newNode
2. newNode -> info = data
3. If the queue is empty
   a. q = newNode
4. Else
   a. newNode->next = q->next
5. Endif
6. q->next = newNode
7. q = newNode

Doubly Linked List

In a doubly linked list, each node contains two pointers - one to its predecessor and another to its successor.

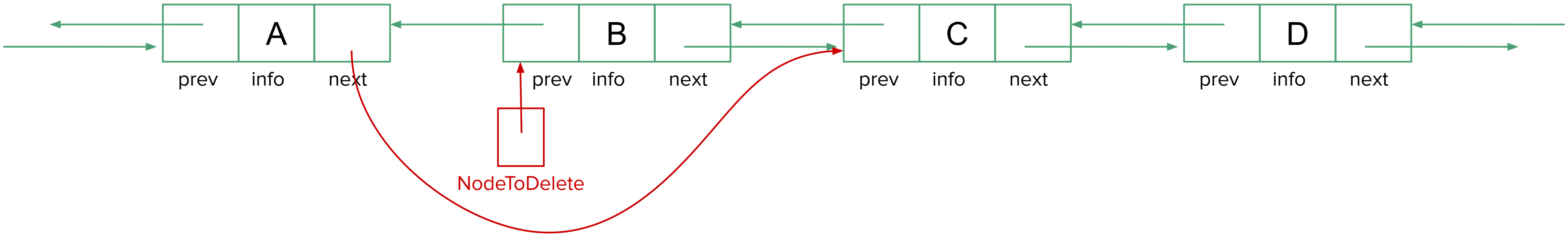


Doubly Linked List: Deletion



Doubly Linked List: Deletion

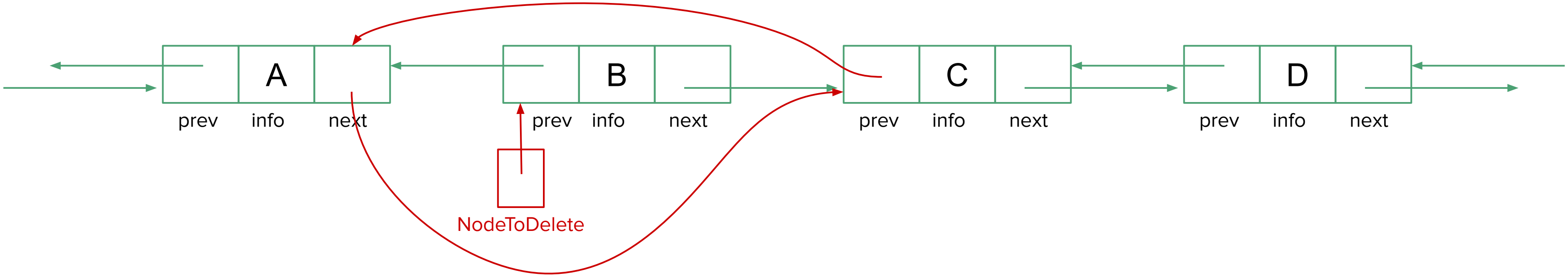
NodeToDelete->prev->next = NodeToDelete->next



Doubly Linked List: Deletion

NodeToDelete->prev->next = NodeToDelete->next

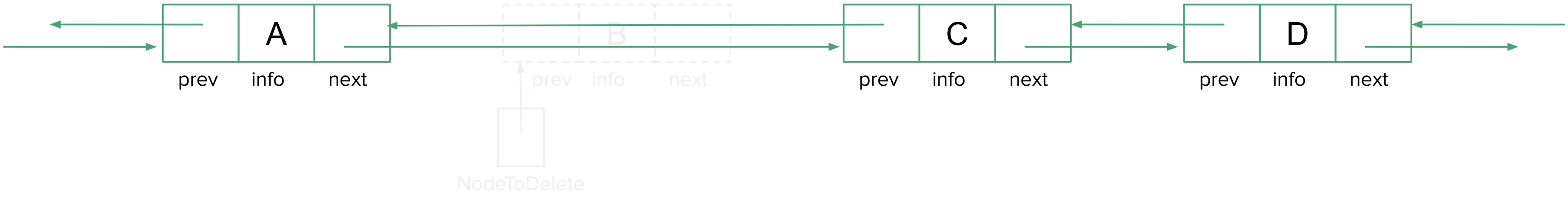
NodeToDelete->next->prev = NodeToDelete->prev



Doubly Linked List: Deletion

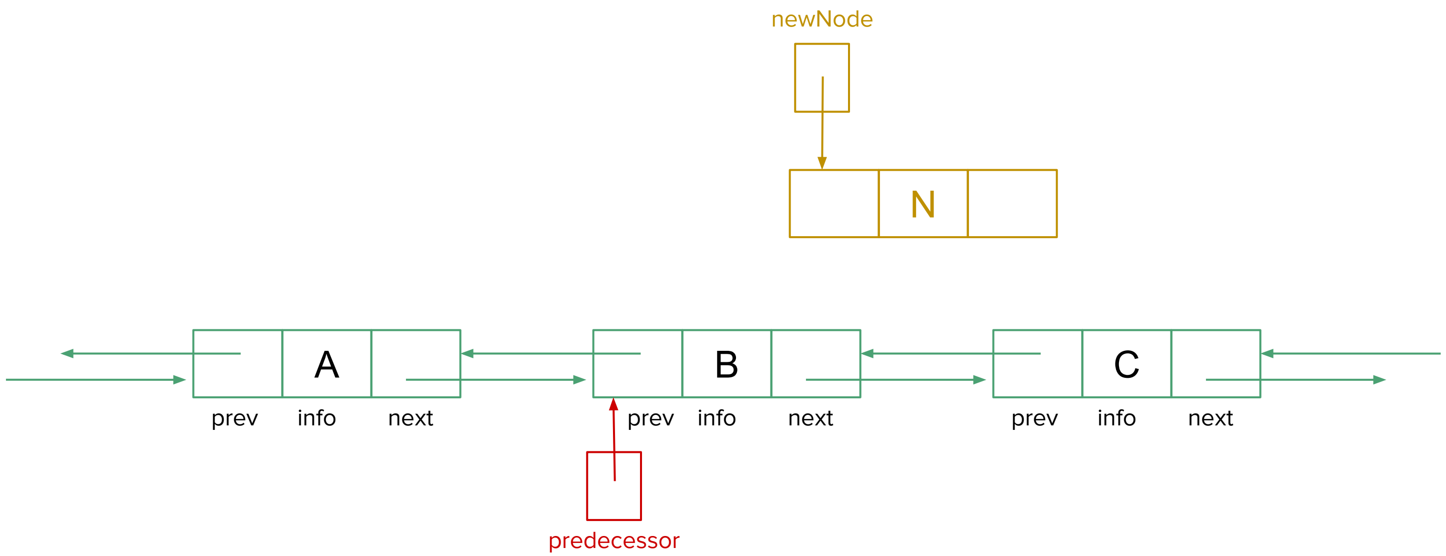
NodeToDelete->prev->next = NodeToDelete->next

NodeToDelete->next->prev = NodeToDelete->prev



Doubly Linked List: Insertion

Inserting a node to the right of a given node

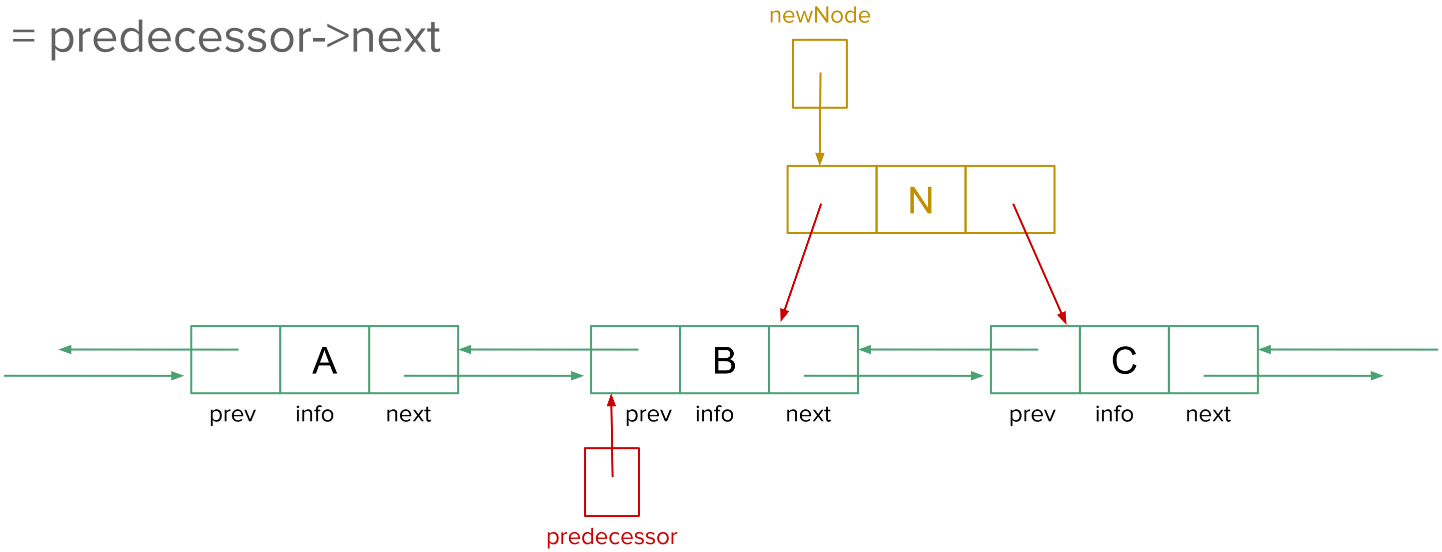


Doubly Linked List: Insertion

Inserting a node to the right of a given node

newNode->prev = predecessor

newNode->next = predecessor->next



Doubly Linked List: Insertion

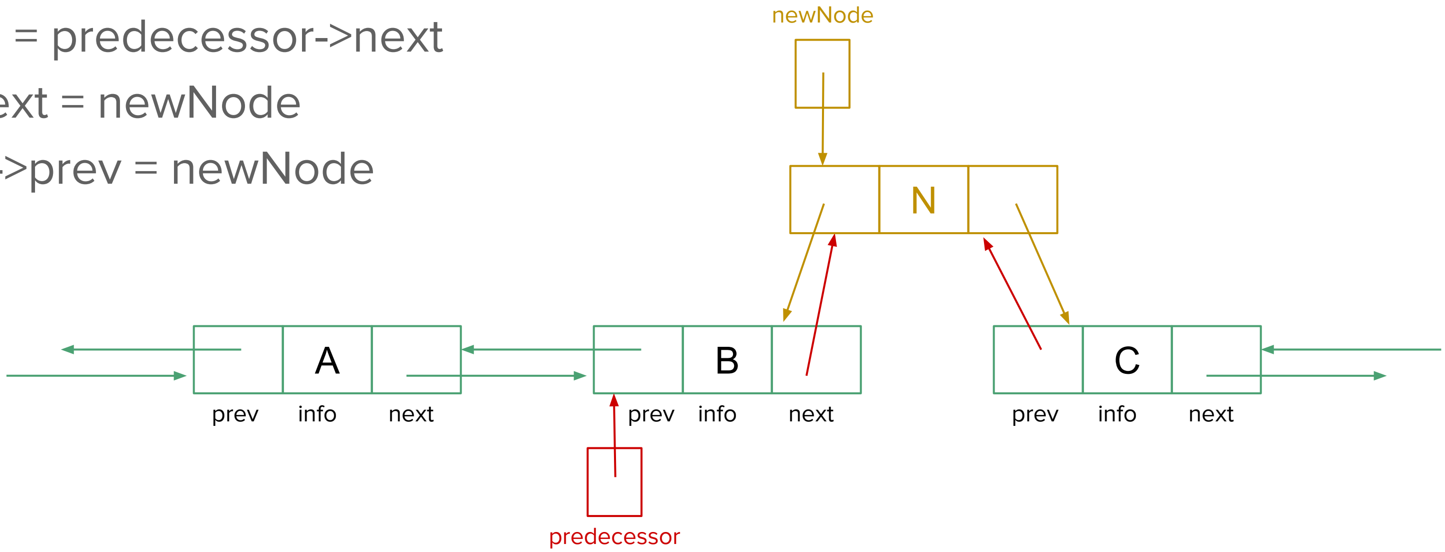
Inserting a node to the right of a given node

newNode->prev = predecessor

newNode->next = predecessor->next

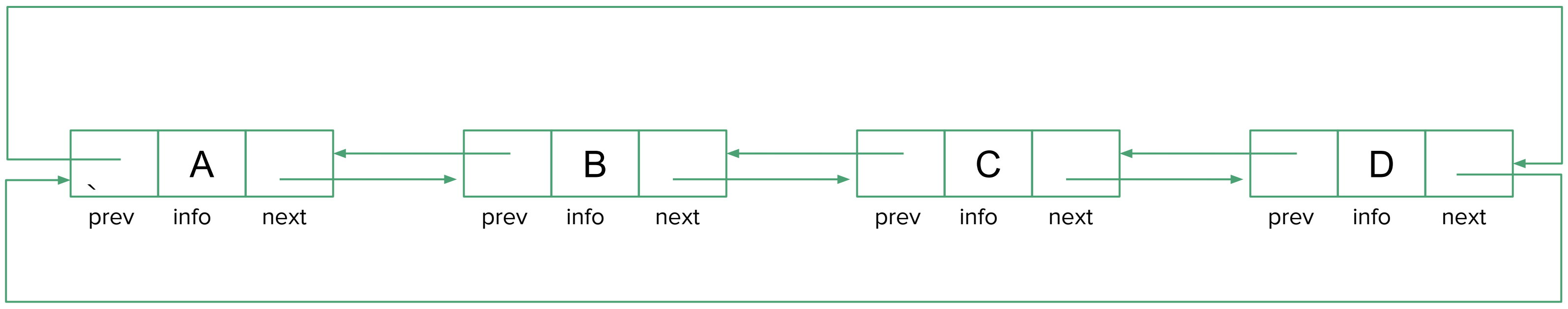
predecessor->next = newNode

newNode->next->prev = newNode



Circular Doubly Linked List

Previous link of the first node points to the last and the next link of the last node points to the first node.



Advantages of Linked Lists

- Dynamic data structure
- Efficient memory utilization
- Easier insertions and deletions
- Easy to carry out complex operations

Disadvantages of Linked Lists

- More memory is required
- Time consuming