Doubly Linked Lists

Lecture 20
Section 17.5

Robb T. Koether

Hampden-Sydney College

Mon, Mar 6, 2017

Outline

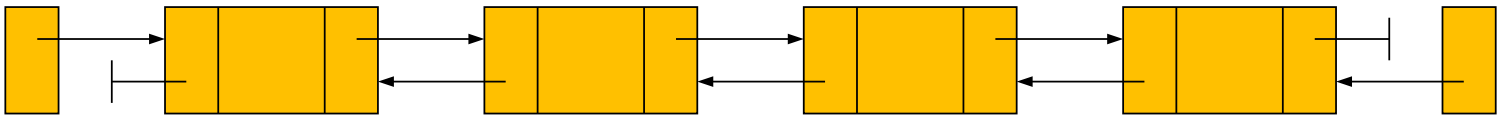
Doubly Linked Lists

- Adding a tail pointer to the `LinkedList` class greatly helped the `pushBack()` function, but it did not help the `popBack()` function.
- That was because we could not “back up” from the tail.
- To facilitate both `pushBack()` and `popBack()`, we could add “backward” pointers to every node.
- Then we could back up from any node to its predecessor.

Doubly Linked Lists

Definition (Doubly Linked List)

A doubly linked list is a linked list in which each node has two links: a forward link to its successor node and a backward link to its predecessor node.



Outline

Doubly Linked List Node Data Members

`DoublyLinkedListNode` Data Members

- `T m_value` - The value stored in the node.
- `DoublyLinkedListNode* m_prev` - A pointer to the previous node.
- `DoublyLinkedListNode* m_next` - A pointer to the next node.

- A doubly linked list must use doubly linked list nodes.

Doubly Linked List Data Members

`DoublyLinkedList` Data Members

- **`int`** `m_size` - Number of elements in the list.
- `DoublyLinkedListNode* m_head` - Pointer to the first node.
- `DoublyLinkedListNode* m_tail` - Pointer to the last node.

Chasing Pointers

- We can move both forwards and backwards in the list.
- When chasing pointers, it is not necessary to keep a pointer to the previous node.

Outline

Inserting and Deleting in a Doubly Linked List

- Apply the 12-step method to
  - Insert a node into a doubly linked list.
  - Delete a node from a doubly linked list.

Outline

Assignment

Assignment

- Read Section 17.5.