Doubly Linked List

The **Node** class is similar to the one in the singly linked list but includes an additional **prev** pointer for the previous node in a doubly linked list.

DoublyLinkedList Class:

Constructor **DoublyLinkedList()**

- Initializes an empty doubly linked list with **head**, **tail**, and **length**.

Function **isEmpty()**

- Checks if the doubly linked list is empty.
- Time Complexity: O(1)
- Space Complexity: O(1)

Function **insertWhileEmpty(Node* newNode)**

- Inserts a new node when the linked list is empty.
- Time Complexity: O(1)
- Space Complexity: O(1)

Function **findIndex(int index)**

- Finds the node at a given index.
- Time Complexity: O(index)
- Space Complexity: O(1)

Function **deleteOnlyElement()**

- Deletes the only element from the linked list.
- Time Complexity: O(1)
- Space Complexity: O(1)

Function **findValue(int value)**

- Finds the index of a given value in the linked list.
- Time Complexity: O(n)
- Space Complexity: O(1)

Function **pushBack(int value)**

- Adds an element to the end of the doubly linked list.
- Time Complexity: O(1)

- Space Complexity: O(1)

Function **pushFront(int value)**

- Adds an element to the beginning of the doubly linked list.
- Time Complexity: O(1)
- Space Complexity: O(1)

Function **insertAfterIndex(int index, int value)**

- Inserts an element after a specified index.
- Time Complexity: O(index)
- Space Complexity: O(1)

Function **insertBeforeIndex(int index, int value)**

- Inserts an element before a specified index.
- Time Complexity: O(index)
- Space Complexity: O(1)

Function **popBack()**

- Removes the last element from the doubly linked list.
- Time Complexity: O(1)
- Space Complexity: O(1)

Function **popFront()**

- Removes the first element from the doubly linked list.
- Time Complexity: O(1)
- Space Complexity: O(1)

Function **deleteNode(int index)**

- Deletes the element at the specified index.
- Time Complexity: O(index)
- Space Complexity: O(1)

Function **display()**

- Displays the elements of the doubly linked list.
- Time Complexity: O(n)
- Space Complexity: O(1)

Function **reverse()**

- Reverses the doubly linked list.
- Time Complexity: O(n)
- Space Complexity: O(1)

Function **search(int value)**

- Searches for an element and returns its index.
- Time Complexity: O(n)
- Space Complexity: O(1)

Function **update(int index, int value)**

- Updates the value of a node at a given index.
- Time Complexity: O(index)
- Space Complexity: O(1)

Function **headPointer()**

- Returns the value of the head node.
- Time Complexity: O(1)
- Space Complexity: O(1)

Function **tailPointer()**

- Returns the value of the tail node.
- Time Complexity: O(1)
- Space Complexity: O(1)

Function **findLength()**

- Returns the length of the doubly linked list.
- Time Complexity: O(1)
- Space Complexity: O(1)

Destructor **~DoublyLinkedList()**

- Frees memory by deleting all nodes in the doubly linked list.
- Time Complexity: O(n)
- Space Complexity: O(1)

Main Function:

The **main** function demonstrates the usage of various doubly linked list operations, including insertion, deletion, updating, searching, and reversing.