Linked List

Data Structures Made Easy

DUBLIN CITY UNIVERSITY

Contents

1. ***Definition***

A linked list is an ordered set of nodes, where each node contains a link to the next node. It is also possible for the node to contain a linked to the previous node.

There are two types of linked lists:

- **Singly Linked List**: A linked list that contains a pointer in each node, which points to the next node in the list. To locate a desired node, the list must be traversed from the beginning. Traversing the list can only be accomplished in one direction. By discovering a node with a pointer to the next node, thus locates the originally desired node.

- **Doubly Linked List:** A linked list that contains two pointers in each node. One pointer points to the next node, whilst the other pointer points to the last node in the list. Traversing the list can be accomplished in two directions.

2. ***Implementation***

A linked list works as follows:

1. Specify the type of data that is desired to be stored.

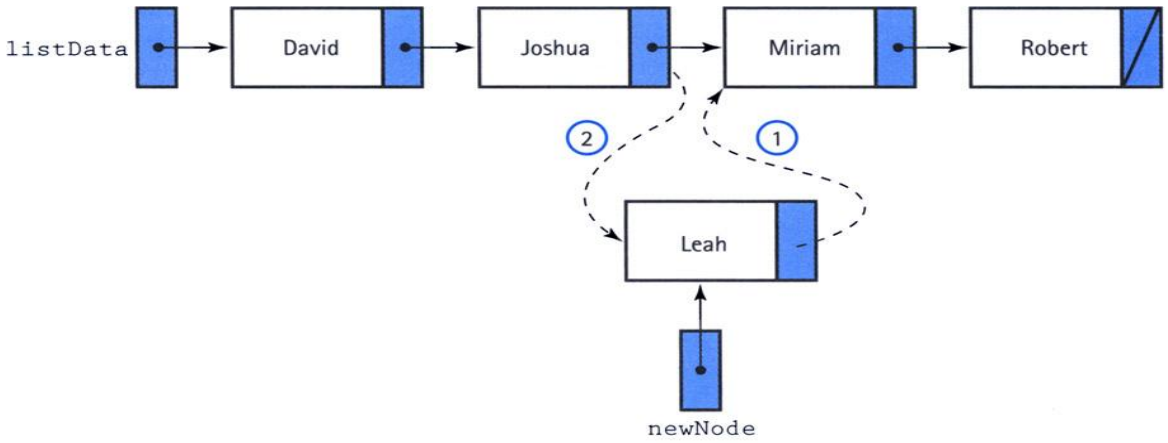
2. Insert the element into the list at a specified location.

3. Retrieve the specified element from the list.

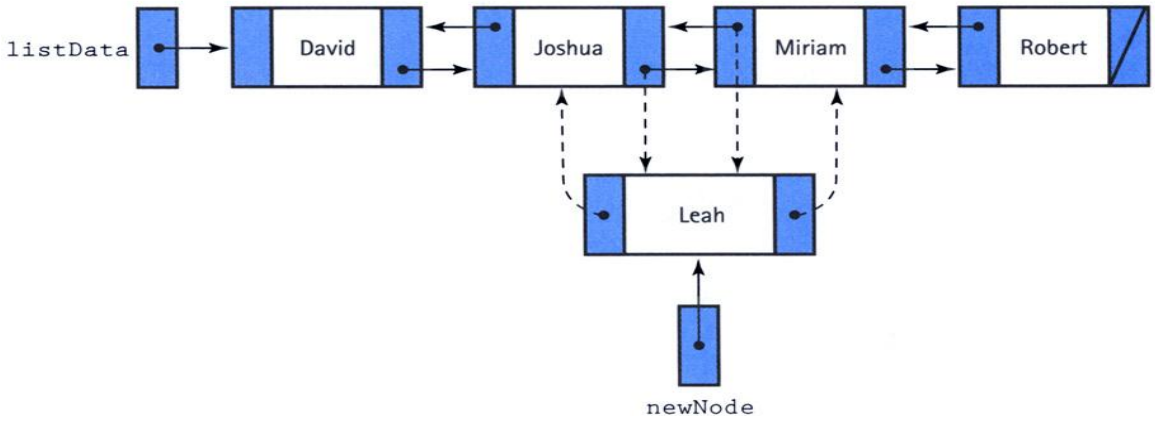
4. Nodes after the index location of the newly inserted/removed element are moved backward / forward. The head and/or the tail are pointed to the correct nodes.

3. ***Example***

(a) Inserting into a singly linked list (Insert Leah)



(b) Inserting into a doubly linked list



4. ***Functions***

The implementation of a linked list involves the following functions

- **LinkedList<Object>()**
  This constructor creates an empty linked list that holds the object type, which is specified upon the creation of the list.

- **add(Object obj)**
  This function attaches the element to the end of the list.

- **add(int index, Object obj)**
  This function inserts the element at a certain position in the list, specified by the index. If the index is out of range, the function throws an exception.

- **size()**
  This function returns the amount of elements in the list.

- **indexOf(Object obj)**
  This function returns the index number of the element in the list, which is specified by the element.

- **get(int index)**
  This function returns the element in the list, which is specified by the index.

- **addFirst(Object obj)**
  This function inserts a specified element at the beginning of the list.

- **addLast(Object obj)**
  This function inserts a specified element at the end of the list.

- **remove(int index)**
  This function removes the element at the specified position in the list. If the index is out of range, the function throws an exception.

- **removeLast()**
  This function removes the element at the end of the list.

5. ***Singly Linked List Pseudocode***

```
class Node < T >{

        T item
        Node null

        constructor Node
}

function size (){

        return number of items in list
}

function get ( index ){

        if index is out of bounds
                throw exception

        search list until index is found

        return Node item
}

function set ( index, T ){

        if index is out of bounds
                throw exception

        search list until index is found

        temporary = Node item
        Node item = T

        return temporary
}

function add ( T ){

        T = tail item
        add ( number of items in list, T )
}
```

```
function add ( index, T ){

        if index is out of bounds
                throw exception

        if index == 0
                if tail == null
                        tail = head

        else
                Node = head

                search list until index is found

                if tail == Node
                        tail = next

        increment number of items in list
}

function addFirst ( T ){

        T = head item
        add ( number of items in list, T )
}

function indexOf ( T ){

        if tail == null
                return -1
        else
                while Node item != T
                        increment index

                return index
}

function removeLast (){

        if no elements in list
                throw exception

        T = tail item
        Node = head

        while not at last index of list
                increment index
```

```
        tail = Node next
        decrement number of items in list

        return T
}
```

6. ***Doubly Linked List Pseudocode***

```
class Node < T >{

        T item
        Node null

        constructor Node
}

function size (){

        return number of items in list
}

function get ( index ){

        if index is out of bounds
                throw exception

        search list until index is found

        return Node item
}

function set ( index, T ){

        if index is out of bounds
                throw exception

        search list until index is found

        temporary = Node item
        Node item = T

        return temporary
}
```

```
function add ( index, T ){

        if index is out of bounds
                throw exception

        if head == null
                if tail == null
                        tail = head
                else
                        head next predecessor = head
        else
                search list until index is found

                if tail == Node
                        tail = Node next
                else
                        Node next predecessor = Node next

        increment number of items in list
}

function removeLast (){

        if no elements in list
                throw exception

        T = tail item
        tail = tail predecessor

        if tail != null
                tail next = null
        else
                head = null

        decrement number of items in list
        return T
}
```

6. ***Complexity***

The time complexities of functions for a linked list with n nodes are the following:

- 0(1)
  - size
  - addFirst
  - removeFirst
  - add (doubly linked list)
  - remove (doubly linked list)
  - addLast (doubly linked list)
  - removeLast (doubly linked list)

- O(N)
  - get
  - remove
  - index
  - add (singly linked list)
  - remove (singly linked list)
  - addLast (singly linked list)
  - removeLast (singly linked list)

7. ***Advantages of Linked Lists***

The advantage of a singly linked list is that there is only one pointer present. This saves space in the storing and retrieval of elements.

The advantage of a doubly linked list is that it has the ability to traverse in both forward and backward directions. This bi-directional feature increases the performance of storing and retrieval abilities.

8. ***Disadvantages of Linked Lists***

The disadvantage of a singly linked list is that it can only traverse in one direction. The absence of this feature leads to the doubly linked list being the optimal choice for storing and retrieval needs.

The disadvantage of a doubly linked list is the memory consumed with the second pointer. The bi-directional features require additional code and time, in order to update more pointers that are needed.

9. ***References***

davin.50webs 2003. *An improved doubly linked list system* [Online]. Available from: http://davin.50webs.com/research/2003/aidlls.html

Wiki Answers 2009. *Definitions of a single and double linked list?* [Online]. Available from: http://wiki.answers.com/Q/Definitions_of_single_and_double_linked_list

Young Inc 2010. *Doubly Linked List* [Online]. Available from: http://younginc.site11.com/source/5895/fos0052.html

Rose India 2007. *Link List Example in Java* [Online]. Available from: http://www.roseindia.net/java/beginners/linked-list-demo.shtml