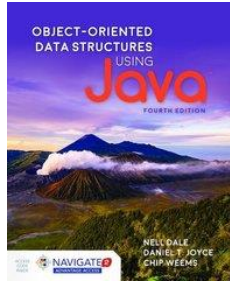


Chapter 6

The List ADT



Section 3

INTRODUCTION TO LINKED LISTS

Objectives

- Learn about linked lists
- Become aware of the basic properties of linked lists
- Note strengths and weaknesses of linked lists

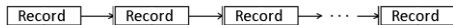
Recall

- For the array-based implementation:
 - First element is at location **0**
 - Successor of item at location i is at location $i + 1$
- What if we remove requirement that list elements be stored in consecutive locations?
 - We then need a "link" that connects each element to its successor (Linked List!!!)

4

Linked Lists

- A **linked list** is a data structure consisting of a collection of records which together represent a sequence

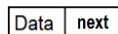


- Each record in a linked list is called a *node*

5

Linked List Nodes

- A linked list **node** consists of two fields:
 1. Data Field
 2. Link field, called *next*, that references the next node of the list

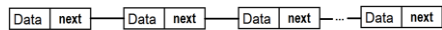


- The references to next nodes are called **successor links**

6

Linked List Nodes

- Nodes are building blocks for linked lists



- Example



7

Linked List Nodes

- The link field of a node in a linked list contains the address of the next node (except the last node)
- Hence, the link field is a reference variable
- The link field of the last node in a linked list is set to null



8

Usefulness of Linked Lists

- Linked lists are among the simplest and most common data structures
- A linked list is appropriate when the number of data elements to be represented in the data structure is unpredictable
- Linked lists are used to implement other data structures including stacks and queues

9

Advantages of Linked Lists

- Linked lists are dynamic
- So, the length of a list can increase or decrease as necessary
- The size of an array, however, cannot be altered

10

Advantages of Linked Lists

- Linked lists become full only when the system has insufficient memory to satisfy dynamic storage allocation requests
- Arrays can become full
 - If the space reserved for a dynamic array is exceeded, the array is reallocated dynamically and the old array is relocated and (possibly) copied into the new one, an expensive operation!

11

Advantages of Linked Lists

- Linked lists allow for efficient insertion or removal of elements from any position in the list without reallocation or reorganization of the entire structure
- Insertion or deletion of an item in an array at random locations will require moving half of the elements on average, and all the elements in the worst case

12

Advantages of Linked Lists

- Linked lists can be maintained in sorted order simply by inserting each new element at the proper point in the list
 - Existing list elements do not need to be moved
- Array elements must be moved to make room to insert a new item at the proper location in the array

13

Disadvantages of Linked Lists

- Arrays allow constant-time random access of items since an index is used
- Linked lists do not allow random access to the data, or any form of efficient indexing
 - Linked list nodes are normally not stored contiguously in memory
 - As a result, many basic operations on linked lists may require scanning most or all of the list elements
 - In addition, many sorting algorithms need direct access

14

Disadvantages of Linked Lists

- Arrays require only space for the data (and a very small amount of control data)
- Linked lists require extra storage for references
 - This requirement often makes linked lists impractical for lists of small data items such as characters or Boolean values, because the storage overhead for the links may exceed by a factor of two or more the size of the data

15

Accessing Nodes of Linked Lists

- To access the n^{th} node of a linked list, we must go through first node, and then second, and then third, etc.
- But how do we access the first node of a linked list?

16

The First Node of a Linked List

- The first node of a linked list is immediately accessible (without traversal) through a pointer (`firstNode`) which points to the first node of the list



17

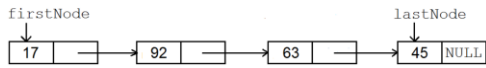
The Last Node of a Linked List

- For simplicity, we will also use a pointer (`lastNode`) to point to the last node of a linked list



18

Linked Lists Example



19

Linked List Traversal

```
currentNode = firstNode;

while( currentNode != null)
{
    // Move currentNode to next node
    currentNode = currentNode.link
}
```

20

Basic Linked Lists Operations

- Insert an item in the list
- Delete an item from the list
- Print the list

21

Printing a Linked List

```
currentNode = firstNode;

while( currentNode != null)
{
    // Display current data
    ..print( currentNode.data + " ");

    // Move currentNode to next node
    currentNode = currentNode.link
}
```





22

Inserting an Item in a Linked List

1. Use the **new** operator to create a new node using dynamic memory allocation
2. Set the data field of the new node to the value of item to be inserted
3. Initialize the link field of the new node to null
4. Insert the new node at the intended location in the linked list by manipulating the link fields

23

Item Insertion: Example

Statement	Effect
<code>newNode = new NodeType;</code>	
<code>newNode->data = 50;</code>	
<code>newNode->link = p->link;</code>	
<code>p->link = newNode;</code>	

24

Deleting an Item From a Linked List

1. Traverse the linked list to identify the node to be deleted and mark it using a pointer
2. Change the Link pointer of the previous node (relative to the node to be deleted) to point at the node that comes after the node to be deleted from the list
3. If so desired, save the value in the Data field of the node to be deleted
4. Use the **delete** operator to delete the node and deallocate memory

25

Item Deletion: Beware of Memory Leak!

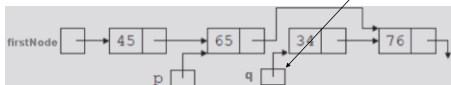
- Memory still occupied by node after deletion, yet memory is inaccessible



26

Item Deletion: Beware of Memory Leak!

- Always deallocate memory using a pointer to the node to be deleted!



27

Building a Linked List

- Linked list are built depending on the data read
- If the data is sorted, the linked list will be sorted
- But if the data is unsorted, we will get an unsorted list

28

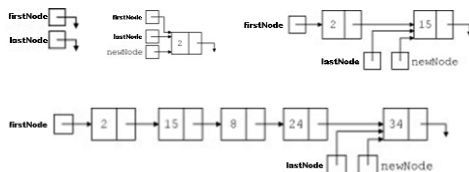
Building an Unsorted Linked List

- An unsorted linked list can be built in two ways:
 - **Forward:** A new node is always inserted at the end of the list
 - **Backward:** A new node is always inserted at the beginning of the list

29

Building a Linked List Forward

- Example: assume that we are building a linked list using data: 2 15 8 24 34



30

Building a Linked List Backward

- Example: assume that we are building a linked list using data: 2 15 8 24 34



31

