Chapter 9
Abstract Data Types and Algorithms

Chapter Goals

• Define an abstract data type and discuss its role in algorithm development
• Distinguish between a data type and a data structure
• Distinguish between an array-based implementation and a linked implementation
• Distinguish between an array and a list

• Distinguish between an unsorted list and a sorted list
• Distinguish between a selection sort and a bubble sort
• Describe the Quicksort algorithm
• Apply the selection sort, the bubble sort, and the Quicksort to a list of items by hand
• Apply the binary search algorithm

• Distinguish between the behavior of a stack and a queue
• Draw the binary search tree that is built from inserting a series of items
• Demonstrate your understanding of the algorithms in this chapter by hand simulating them with a sequence of items
Abstract Data Types

Abstract data type
A data type whose properties (data and operations) are specified independently of any particular implementation

Remember what the most powerful tool there is for managing complexity?

Three Views of Data

Application (user) level
View of the data within a particular problem

View sees data objects in terms of properties and behaviors

Three Views of Data

Logical (abstract) level
Abstract view of the data and the set of operations to manipulate them

View sees data objects as groups of objects with similar properties and behaviors

Three Views of Data

Implementation level
A specific representation of the structure that hold the data items and the coding of the operations in a programming language

View sees the properties represented as specific data fields and behaviors represented as methods implemented in code
Three Views of Data

Describe a word processor from the three views

Composite data type
A data type in which a name is given to a collection of data values

Data structures
The implementation of a composite data fields in an abstract data type

Containers
Objects whose role is to hold and manipulate other objects

Logical Implementations

Two logical implementations of containers

Array-based implementation
Objects in the container are kept in an array

Linked-based implementation
Objects in the container are not kept physically together, but each item tells you where to go to get the next one in the structure

Think of the container as a list of items
Here are the logical operations that can be applied to lists

Add item Put an item into the list
Remove item Remove an item from the list
Get next item Get (look) at the next item
more items Are there more items?
Unsorted and Sorted Containers

**Unsorted container**
The items in the container are not ordered in any way

**Sorted container**
The items in the container are ordered by the value of some field within the items
Array-Based Implementations

How do we implement the operations?

- **Add item**: given an index, shift following items down and store item at index
- **Remove item**: given an index, shift following items up one
- **Get next item**: increment value of index and return value at that position
- **more items**: value of index < length - 1

Linked Implementation

**Linked implementation**

An implementation based on the concept of a node

**Node**

A holder for two pieces of information
- the item that the user wants in the list (item)
- a pointer to the next node in the list (next)
Linked Implementation

How do we implement the operations?

Add item  given current, insert a new node with item in the info part between current and next(current)

Remove item  given current, remove next(current)

Get next item  set current to next(current)

more items  current does not contain null

Figure 9.6  A sorted linked list

Figure 9.7  Store a node with info of 67 after current

Figure 9.8  Remove node next(current)
Lists

List operations
- Create itself (Initialize)
- Insert an item
- Delete an item
- Print itself
- Know the number of items it contains

Generic data type (or class)
A data type or class in which the operations are specified but the type or class of the objects being manipulated is not

Unsorted Lists

Create (initialize)
- Set length to 0

Insert (item)
- Find where the item belongs
- Put the item there
- Increment length

Remove (item)
- Find the item
- Remove the item
- Decrement length

Print
- While (more items)
  - Get next item
  - Print Item

Insert (item)
- Find where the item belongs
- Put the item there
- Increment length

Know Length
- return length

Sorted Lists

From the application view, how do the sorted an unsorted list differ?

The decomposition of which algorithm steps must be different?
Unfinished Algorithm Steps

Find where the items belongs (unsorted):
  Item belongs at the length position

Find where the items belongs (sorted):
  Set tempItem to the first item
  While (item.compareTo(tempItem) > 0)
    Set tempItem to next item
  Item belongs at tempItem

Find the item:
  Set tempItem to first item
  While (item.compareTo(tempItem) not equal 0)
    Set tempItem to next item

Sorting

Arranging items in a collection so that there is an ordering on one (or more) of the fields in the items

Sort Key
The field (or fields) on which the ordering is based

Sorting algorithms
Algorithms that order the items in the collection based on the sort key

Why is sorting important?

Selection Sort

Given a list of names, put them in alphabetical order
  – Find the name that comes first in the alphabet, and write it on a second sheet of paper
  – Cross out the name off the original list
  – Continue this cycle until all the names on the original list have been crossed out and written onto the second list, at which point the second list contains the same items but in sorted order

Selection Sort

A slight adjustment to this manual approach does away with the need to duplicate space
  – As you cross a name off the original list, a free space opens up
  – Instead of writing the value found on a second list, exchange it with the value currently in the position where the crossed-off item should go
**Selection Sort**

Selection Sort

![Figure 9.9 Example of a selection sort (sorted elements are shaded)]

Figure 9.9 Example of a selection sort (sorted elements are shaded)

**Bubble Sort**

**Bubble Sort uses the same strategy:**

Find the next item
Put it into its proper place
But uses a different scheme for finding the next item
Starting with the last list element, compare successive pairs of elements, swapping whenever the bottom element of the pair is smaller than the one above it

**Algorithms**

Can you write the algorithms for the selection sort and the bubble sort?

Can you think of a way to make the bubble sort more efficient?
Quicksort

Figure 9.12 Ordering a list using the Quicksort algorithm

It is easier to sort a smaller number of items: Sort A...F, G...L, M...R, and S...Z and A...Z is sorted

Quicksort

If (there is more than one item in list[first]..list[last])
Select splitVal
Split the list so that
list[first]..list[splitPoint-1] <= splitVal
list[splitPoint] = splitVal
list[splitPoint+1]..list[last] > splitVal
Quicksort the left half
Quicksort the right half

Quicksort

splitVal = 9

[9 20 6 10 14 8 60 11]
[first] [last]

smaller values

[9 6 6 10 14 20 60 11]
[first] [last]

larger values

[6 6 9 10 14 20 60 11]
[first] [splitPoint] [last]

Split

Set left to first + 1
Set right to last
Do
Increment left until list[left] > splitVal OR left > right
Decrement right until list[right] < splitVal OR left > right
If (left < right)
Swap list[left] and list[right]
While (left <= right)
Set splitPoint to right
Swap list[first] and last[right]
QuickSort

Figure 9.13 Splitting algorithm

Binary Search

Sequential search
Search begins at the beginning of the list and continues until the item is found or the entire list has been searched

Binary search (list must be sorted)
Search begins at the middle and finds the item or eliminates half of the unexamined items; process is repeated on the half where the item might be

Say that again...

Boolean Binary Search (first, last)
If (first > last) return false
Else
  Set middle to (first + last)/2
  Set result to item.compareTo(list[middle])
  If (result is equal to 0) return true
  Else
    If (result < 0) Binary Search (first, middle - 1)
    Else Binary Search (middle + 1, last)

Figure 9.14 Trace of the binary search
Binary Search

<table>
<thead>
<tr>
<th>Length</th>
<th>Sequential Search</th>
<th>Binary Search</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Base 10</td>
<td>Base 2</td>
</tr>
<tr>
<td>10</td>
<td>5.5</td>
<td>2.3</td>
</tr>
<tr>
<td>100</td>
<td>50.5</td>
<td>5.8</td>
</tr>
<tr>
<td>1000</td>
<td>500.5</td>
<td>9.2</td>
</tr>
<tr>
<td>10000</td>
<td>5000.5</td>
<td>12.0</td>
</tr>
</tbody>
</table>

Table 9.1 Average Number of Comparisons

Is a binary search always better?

Stacks

Stack
An abstract data type in which accesses are made at only one end
- LIFO, which stands for Last In First Out
- The insert is called Push and the delete is called Pop

Name three everyday structures that are stacks

Queues

Queue
An abstract data type in which items are entered at one end and removed from the other end
- FIFO, for First In First Out
- No standard queue terminology
  - Enqueue, Enque, Enq, Enter, and Insert are used for the insertion operation
  - Dequeue, Deque, Deq, Delete, and Remove are used for the deletion operation.

Name three everyday structures that are queues

Stacks and Queues

Figure 9.15 Stack and queue visualized as linked structures
Trees

Structure such as lists, stacks, and queues are linear in nature; only one relationship is being modeled.

More complex relationships require more complex structures.

*Can you name three more complex relationships?*

Binary tree

A linked container with a unique starting node called the root, in which each node is capable of having two child nodes, and in which a unique path (series of nodes) exists from the root to every other node.

A picture is worth a thousand words…

Trees

Root node

Node with two children

Node with right child

Leaf node

What is the unique path to the node containing 5? 9? 7? …

Binary Search Trees

Binary search tree (BST)

A binary tree (shape property) that has the (semantic) property that a value in any node is greater than the value in any node in its left subtree and less than the value in any node in its right subtree.
Each node is the root of a subtree made up of its left and right children.

Prove that this tree is a BST.

---

**Boolean IsThere(current, item)**

If (current is null)
  return false
Else
  Set result to item.compareTo(info(current))
  If (result is equal to 0)
    return true
  Else
    If (result < 0)
      IsThere(item, left(current))
    Else
      IsThere(item, right(current))

---

Trace the nodes passed as you search for 18, 8, 5, 4, 9, and 15.

What is special about where you are when you find null?
### Binary Search Tree

**Insert** (current, item)

- If (tree is null)
  - Put item in tree
- Else
  - If (item.compareTo(info(current)) < 0)
    - Insert (item, left(current))
  - Else
    - Insert (item, right(current))

**Print** (tree)

- If (tree is not null)
  - Print (left(tree))
  - Write info(tree)
  - Print (right(tree))

Is that all there is to it? Yes!
Remember we said that recursive algorithms could be very powerful.

### Graphs

**Graph**

A data structure that consists of a set of nodes (called vertices) and a set of edges that relate the nodes to each other.

**Undirected graph**

A graph in which the edges have no direction.

**Directed graph (Digraph)**

A graph in which each edge is directed from one vertex to another (or the same) vertex.

![Figure 9.21: Examples of graphs](image)
Graphs

Figure 9.21 Examples of graphs

(b) Vertices: Cities
Edges: Direct Flights

Graphs

Figure 9.21 Examples of graphs

(c) Vertices: Courses
Edges: Prerequisites

Ethical Issues

Computer Hoaxes and Scams

Have you ever received a letter from Nigeria?

Have you ever received email asking you follow a link to "your bank"?

Have you ever given your credit card number to someone who contacted you?

Who am I?

My wife Jill and I are holding the medal I received when I was knighted. What university did I retire from and where am I working now?
Do you know?

Is Software at Sea a spa for programmers?
What is Extreme Programming?
What is the rationale behind paired programming?
How does graph theory relate to terrorist detection?