CSSE 220 Day 26

Sorting Wrap-up
Data Structures Intro

Checkout *BinaryInteger* project from SVN

Questions

Course Goals for Sorting: You should…

- Be able to describe basic sorting algorithms:
  - Selection sort
  - Insertion sort
  - Merge sort
- Know the run-time efficiency of each
- Know the best and worst case inputs for each

Recap: Selection Sort

- Basic idea:
  - Think of the list as having a sorted part (at the beginning) and an unsorted part (the rest)
  - Find the smallest number in the unsorted part
  - Exchange it with the element at the beginning of the unsorted part (making the sorted part bigger and the unsorted part smaller)

Repeat until unsorted part is empty

Q1a

Recap: Insertion Sort

- Basic idea:
  - Think of the list as having a sorted part (at the beginning) and an unsorted part (the rest)

  - Get the first number in the unsorted part
  - Insert it into the correct location in the sorted part, moving larger values up in the array to make room

Repeat until unsorted part is empty

Q1b

Merge Sort

- Basic recursive idea:
  - If list is length 0 or 1, then it's already sorted
  - Otherwise:
    - Divide list into two halves
    - Recursively sort the two halves
    - **Merge** the sorted halves back together

Analyzing Merge Sort

- Use a recurrence relation again:
  - Let T(n) denote the worst-case number of **array access** to sort an array of length n
  - Assume n is a power of 2 again, $n = 2^m$, for some m

- Or use tree-based sketch…

Q2, 3, 1b

Data Structures and Abstract Data Types (ADT)

- Understanding the engineering trade-offs when storing data

Data Types

- What is "data"
- What do we mean by "data type"?
- An _____ of the ____
- An interpretation is basically a set of _____.
- The interpretation may be provided
  - by the hardware, as for int and double types
  - by software, as for the java.math.BigInteger type.
  - by software with much assistance from the hardware, as for the java.lang.Array type.

Abstract Data Type (ADT)

- A mathematical model of a data type. Specifies:
  - The type of data stored
  - the operations supported
  - the types and return values of these operations
  - Specifies **what** each operation does, but not **how** it is implemented.

Abstract Data Type example

- **Non-negative integer ADT.**
  A special value: *zero*:
- Basic operations include *succ pred isZero* .
  Derived operations include *plus* .
  - **Sample rules:**
    isZero(succ(n)) ➔ false
    plus(n, zero) ➔ n
    plus(n, succ(m)) ➔ succ(plus(n, m))

Standard implementation: Binary numbers.
But there are many other possibilities.

Rules are independent of implementation.

Abstract Data Type example

- **Non-negative integer ADT.**
  A special value: *zero*:
- Basic operations include *succ pred isZero* .
  Derived operations include *plus* .
  - **Sample rules**:
    isZero(succ(n)) ➔ false
    plus(n, zero) ➔ n
    plus(n, succ(m)) ➔ succ(plus(n, m))

Sample Implementation: Unary strings.
4 is represented by `"xxxx"`, 2 by `"xx"` 0 by `""`

Sample Implementation: Reversed binary strings.
4 is represented by `"001"`, 11 by `"1101"`
zero is represented by `"0"` or `""`
(the latter to make recursion easier)

Integer Representation Exercise

- **`addOne()`** together
  `plus()` for HW (challenging!)

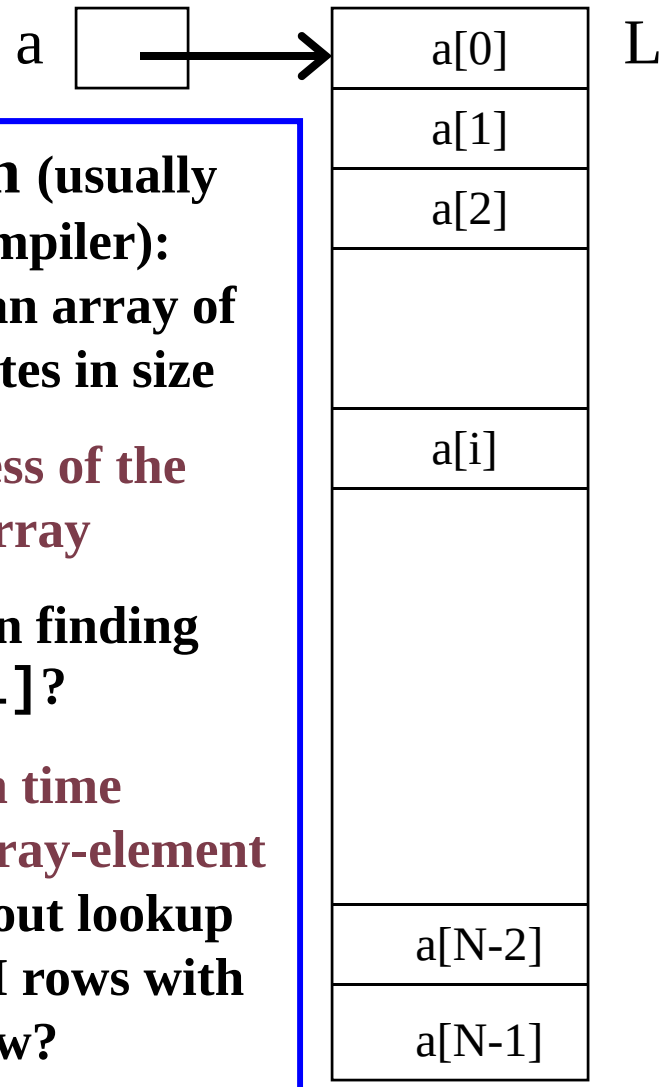
Data Structures

- Efficient ways to store data based on how we'll use it
- The main theme for the last 1/6 of the course
- So far we've seen `ArrayLists`
  - Fast addition **to end of list**
  - Fast access to any existing position
  - Slow inserts into and deletes from the middle of the list

The most common collection data structure is …

- An array.
- Size must be declared when the array is constructed
- We can look up or store items by index

```
a[i+1] = a[i] + 2;
```



Implementation (usually handled by the compiler): Suppose we have an array of N items, each b bytes in size

Let L be the address of the beginning of the array

What is involved in finding the address of `a[i]`?

What is the Big-oh time required for an array-element lookup? What about lookup in a 2D array of M rows with N items in each row?

What about lookup in a 3D array (M x N x P)?

Q12-16

Some basic data structures

- Array (1D, 2D, …)
- Stack

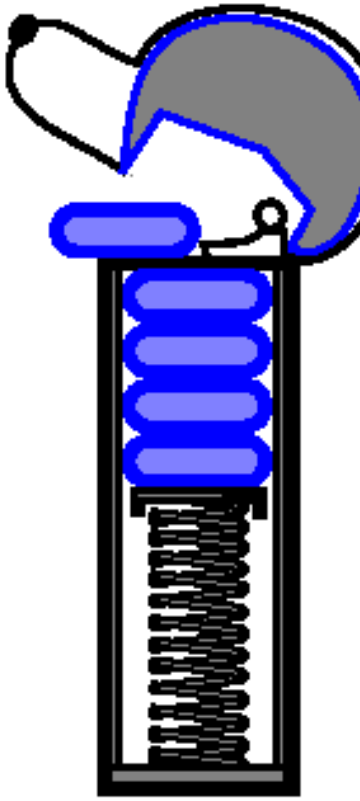
What is "special" about each data type?

What is each used for?

What can you say about time required for
- adding an element?
- removing an element?
- finding an element?

You should be able to answer all of these by the end of this course.

Stack



- Last-in-first-out (LIFO)
- Only top element is accessible
- Operations: push, pop, top, topAndPop
  - All constant-time.
- Easy to implement as a (growable) array with the last filled position in the array being the top of the stack.
- Applications:
  - Match parentheses and braces in an expression
  - Keep track of pending function calls with their arguments and local variables.
  - Depth-first search of a tree or graph.

Some basic data structures

- Array (1D, 2D, …)
- Stack
- Queue

What is "special" about each data type?

What is each used for?

What can you say about time required for
- adding an element?
- removing an element?
- finding an element?

You should be able to answer all of these by the end of this course.

Queue

- First-in-first-out (FIFO)
- Only oldest element in the queue is accessible
- Operations: enqueue, dequeue
  - All constant-time.
- Can mplement as a (growable) "circular" array
  - http://maven.smith.edu/~streinu/Teaching/Courses/112/Applets/Queue/myApplet.html
- Applications:
  - Simulations of real-world situations
  - Managing jobs for a printer
  - Managing processes in an operating system
  - Breadth-first search of a graph

Some basic data structures

What is "special" about each data type?

What is each used for?

What can you say about time required for
- adding an element?
- removing an element?
- finding an element?

A quick preview of the rest of the list

- Array (1D, 2D, …)
- Stack
- Queue
- List
  - ArrayList
  - LinkedList
- Set
- MultiSet
- Map (a.k.a. table, dictionary)
  - HashMap
  - TreeMap
- PriorityQueue
- Tree
- Graph
- Network

You should be able to answer all of these by the end of this course.

Work on MineSweeper