CSSE 220 Day 17

Abstract Data Types
Some Low-Level Implementations

CSSE 220 Day 17

- In Angel: **Lessons > Project Forms > Paint Evaluation > Paint**
  - Please finish paint partner review survey (on ANGEL) asap.
  - Have fun evaluating each other’s Paint programs
    - Friday, 5 pm if possible
    - I’ve looked at all of them and deeply at 4.

- Be working on Hardy's Taxi.
- Find a partner for Markov (different than your Paint partner)
  - Survey in class tomorrow

- Questions?
- Today: BinaryInteger exercise, more data structures.

ADT for non-negative integers

- How to represent? Let’s look at 2 choices:
  - Unary strings, e.g., 7 = “111111”
    - ZERO:
    - succ:
    - pred:
  - Binary strings, e.g., 6 = “011”
    - ZERO:
    - succ (addOne):
      - Let’s write some tests and develop an algorithm
    - plus:
      - Tests?

For the next 35 minutes

- Work on the BinaryInteger exercise (linked from the Schedule page)
- Work with a partner
- If you finish early, work on Hardy's Taxi

Data and Abstract Data Types (Recap)

- What is data? (bits!)
- What is a Data Type
  - An interpretation of the bits
    - basically a set of operations

- Abstract Data Type example: non-negative integer
  - ZERO, succ, pred, isZero (derived methods plus, mult).
  - 1st representation: unary strings
    - ZERO is "", succ(zero) is "1", succ(succ(zero)) is "11"
    - We wrote succ( ) and pred( )
  - 2nd rep: **binary strings** (least-significant bit first)
    - ZERO is "0", succ(zero) is "1", succ(succ(zero)) is "01"
    - We wrote succ( )

Data Structures

- Most of the time when we talk about a **data structure**, we mean an ADT for storing several items (usually all of the items have the same type).
- When studying a new data structure, consider three aspects:
  - Specification (interface for the operations)
  - Implementation (sometimes several alternate implementations)
  - Application (how can it be used?)
- **Mostly, these can be considered independently.**
  - If we understand the interface and trust the person who says she implemented it, we can feel free to apply it without having to understand the details of the implementation.
- **220 emphasizes specification and application.**
- **230 emphasizes specification and implementation.**

Interlude

- The dedication from *Data Structures and the Java Collections Framework* by William Collins (first edition):
    - To Karen, my wife of 35 years, for giving me 20 of the happiest years of my life.
- Go figure!

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;
```

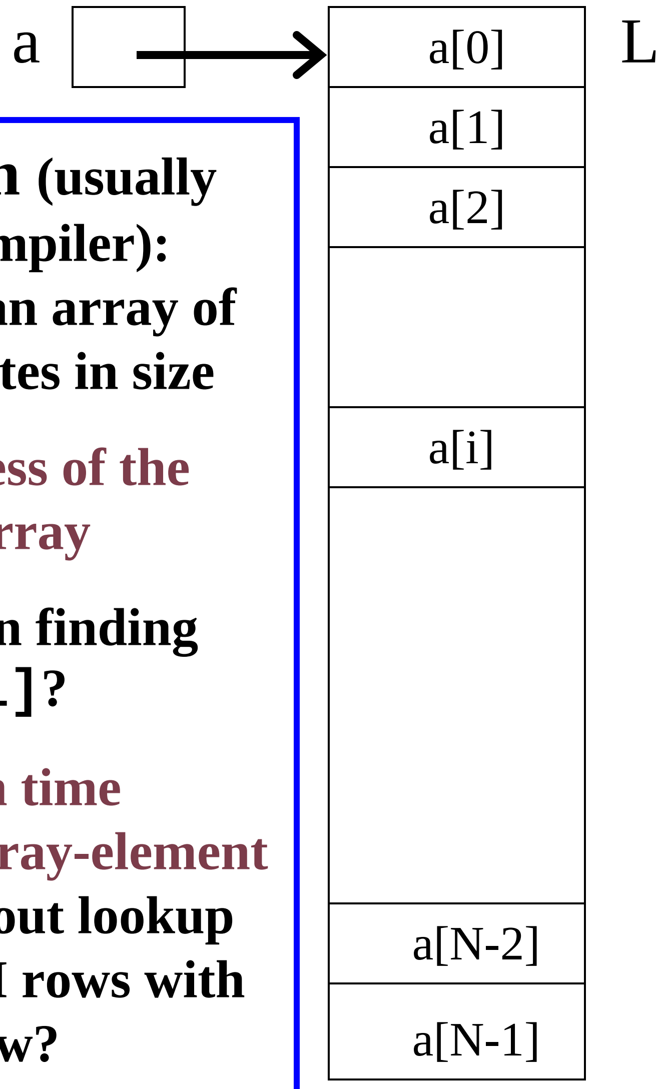
Implementation (usually handled by the compiler): Suppose we have an array of N items, each b bytes in size

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)?



Some basic data structures

What is "special" about each data type?

What is each used for?

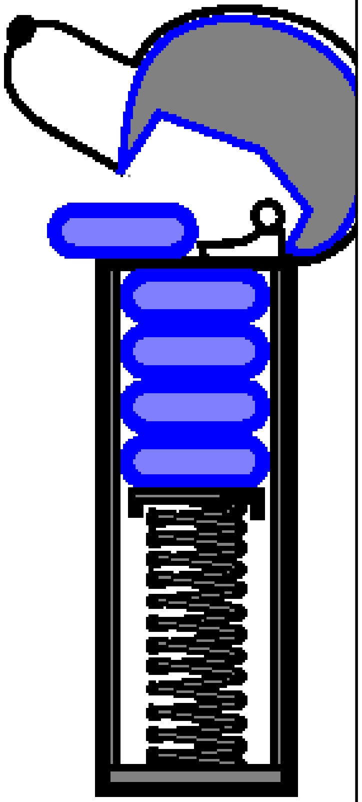
What can you say about time required for
- adding an element?
- removing an element?
- finding an element?

- Array (1D, 2D, …)
- Stack

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

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?

- Array (1D, 2D, …)
- Stack
- Queue

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
- You’ll implement a fixed-length queue next week

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?

- 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.