Organizing Data

The Power of Structure...

Notes for CSC 100 - The Beauty and Joy of Computing
The University of North Carolina at Greensboro

Reminders

Lab this Friday: Lists! Remember Pre-lab reading.
Blown to Bits: Moving on to Chapter 3… start reading!
Homework 2: Due date still 2 weeks away - you should be working on it!

A Flood of Data

Consider the amount of data we deal with:

- Human genome: Just over 3 billion base pairs
  - Typing in 12pt on 8.5x11 paper fits 2880 characters
  - So the human genome would be over a million pages (printed two-sided; an 86 foot high stack of paper)

  - 1.23 billion monthly active users
  - Around 10 billion messages sent per day
  - On index cards, would be a stack 1260 miles high!
  - ... or end-to-end would stretch around the world 30 times

- Large Synoptic Survey Telescope
  - 16 terabytes (16,000,000,000,000 bytes) will be captured per day
  - No human being will ever see most of this data
“Big Data” is the “Big Thing”

What everyone is talking about...

Organizing Data

Until now in this class, we have talked about operations on one or two numbers at a time:

But we might think about operations on collections of data:
- Students in a class
- Customers of a store
- Star locations
- Messages on Facebook
- Roads in the United States

We need:
- Abstractions to think about
- Representations to implement
Abstract Data Types and Data Structures

An abstract data type is a type of data with associated operations, but whose representation is hidden.

Example:

**Type of data:** Collection of student names

**Associated operations:** Add new student, delete student from collection, check if student is in collection, iterate through all students in collection

I can use this collection without worrying about how these operations are accomplished - abstraction!

A data structure is a particular implementation of an abstract data type.

- Determines efficiency of operations
- Roughly: Function definitions for operations

Lists - The Abstract Data Type

A "list" is a very fundamental idea in programming and in life

**Type of data:** An ordered collection of items

**Associated Operations:**

- Add an item to the front
- Add an item to the end
- Add an item at a specific position
- Delete an item from the front
- Delete an item from the end
- Delete an item from a specific position
- Check the list contains a given value
- Get the first item in the list
- Get the last item in the list
- Get the item from a specific position
- Report how many items are in the list

Class Exercise: What sort of operations would you like to be able to perform on a list?
Lists - Programming Language View

All modern programming languages support lists!

Create and store a list

BYOB

Python: mylist = ['work', 'shop', 'eat', 'sleep']
Java: List<String> mylist = Arrays.asList("work", "shop", "eat", "sleep")
(Note: The Java is not really the same as the others…)

Add to the end of a list

BYOB

Python: mylist.append("wake up")
Java: mylist.add("eat")

Get the 3rd item in a list

BYOB

Python: mylist[2]
Java: mylist.get(2)

See also: http://docs.oracle.com/javase/6/docs/api/java/util/List.html

Important Points

We use lists all the time to organize things in our lives

They are just as useful for organizing data in a program

When you want to use a list, you don’t really want to worry about how the computer implements the basic list operations

What we didn’t talk about: There are many ways for a computer to actually store a list (many implementations)

● Some have efficient insertions and some don’t
● Some use less memory than others
● Need to be more comfortable with how things are stored in memory to say much more…

A Flavor of Something More Advanced

Dictionary ADT

Type of data: Collection of pairs of items

● Each pair is a unique identifier (a key) and associated data
● Examples of pairs:
  ○ Student ID number and GPA  (886517124 , 3.45)
  ○ Facebook IDs and profiles  (joe@example.com , "Joe Walsh, …")
  ○ Social Security Numbers and incomes  (491-24-6243 , $43,700)

Associated Operations:

● Get item from key
● Add new (key, data) pair
● Delete pair using key
● Iterate through all pairs
A Flavor of Something More Advanced
Dictionary Implementation 1 - Using a List!

Example: Storing (SSN, income) pairs

<table>
<thead>
<tr>
<th>SSN</th>
<th>Income</th>
</tr>
</thead>
<tbody>
<tr>
<td>353-39-5411</td>
<td>$79,500</td>
</tr>
<tr>
<td>805-19-6915</td>
<td>$51,000</td>
</tr>
<tr>
<td>669-88-7009</td>
<td>$67,500</td>
</tr>
<tr>
<td>452-73-0902</td>
<td>$46,200</td>
</tr>
<tr>
<td>345-63-3094</td>
<td>$60,300</td>
</tr>
<tr>
<td>539-10-3245</td>
<td>$47,800</td>
</tr>
<tr>
<td>408-79-6919</td>
<td>$29,300</td>
</tr>
</tbody>
</table>

Idea 1: Keep all pairs in a list

To add a new pair:
- Put it at the end

To find an item by key:
- Iterate through the list checking each key

To delete an item by key:
- Find it and then delete the pair from the list

With 7 items, looking for the last key (the "worst case") takes 7 iterations
With 300,000,000 items, the worst case takes 300,000,000 iterations

A Flavor of Something More Advanced
Dictionary Implementation 2 - A Binary Search Tree

Idea 2: Instead of one "path" through the pairs, make two "next choices" at each step of the iteration
- One choice for smaller keys
- One choice for larger keys

412-73-0902 : $46,200
353-39-5411 : $79,500
088-46-7609 : $67,300
345-63-3094 : $60,300
408-79-6919 : $29,300
539-10-3245 : $47,800
089-18-6919 : $31,000

Worse case for 7 keys is now 3 comparisons/iterations \(7 = 2^3 - 1\)
In 4 steps could handle 15 keys \((2^4 - 1)\)
In 5 steps could handle 31 keys \((2^5 - 1)\)
... In 29 steps could handle 536,870,911 keys \((2^{29} - 1)\) - enough for all 314 million U.S. citizens

Using a list would take over 18 million times longer!

Summary

Big take-aways:
1. **Abstract Data Types** allow you to focus on using your data without worrying about how it is organized.
2. **Data Structures** describe how data is organized, and can make a huge difference on how efficiently you can use it.

Other things to remember from this lecture:
- Lists are the most fundamental data structure - understand lists!
- Binary Search Trees can locate information fast - know the basic idea!

If you study more computer science:
- You'll learn about a variety of generally useful ways to think about data (ADTs)
- You'll learn about many advanced ways to organize data (data structures)
- You'll learn how to analyze, discuss, and compare efficiency of alternatives