Intriguing, Rich, and Challenging Assignments in APCS

Owen Astrachan

http://www.cs.duke.edu/csed/talks
Biorhythms

- Physical, emotional, and intellectual cycles influence our lives
  - Period of intellectual cycle is 33 days
  - We “peak intellectually” 8.25 days after start of a cycle
  - On what day is your next peak?

- Needed to solve:
  - Basic C++
  - Date class
  - Thinking
Biorhythms: Intrigue and Challenge?

- **Multiple approaches yield a solution**
  - Possible to use loop?
  - Possible to use only arithmetic? Control statements?

- **Careful thinking about test cases helps**
  - If we were born today, does the method work?
  - If we were born yesterday?...
  - What other test cases can we use to check, is it a good idea to check with your own birthday?
  - Can we use web resources to check?

- **Generalize**
  - Emotional: 28 day cycle; physical: 23 day cycle
Sufficient Background

#include <iostream>
using namespace std;
#include "date.h"

int main()
{
    int month, day, year;
    cout << "enter month day year of birth ";
    cin >> month >> day >> year;
    Date today;
    Date birth (month, day, year);

    cout << "On " << today
         << " you’ve been alive for "
         << today - birth << " days" << endl;
    return 0;
}
Where are we going? An Overview

- **Biorhythm:** small example of ideal assignment
  - **Intriguing:** *excites curiosity or interest by puzzling, novel, or otherwise arresting qualities*
    - Can we find critical days?
    - Can we verify/test validity of biorhythm concept?
    - Can we do this without a computer?
  
  - **Rich:** *abounding in desirable elements or qualities*
    - Date objects are nearly *immutable* (change with `++`)
    - Usable in many contexts: from day two to day two hundred
  
  - **Challenging:** *requiring full use of one’s abilities, energy, or resources*
    - Doable with minimal programming experience, or lots!
    - Test algorithmic thinking and problem-solving
Student and Teacher Viewpoints

● **Intriguing**
  ▶ Exciting interest and curiosity of teacher is certain to have an affect on students
  ▶ Exciting interest and curiosity of student is certain to have an affect on teacher

● **Rich**
  ▶ Student view of desirable qualities doesn’t always match teacher view – but we know better (and we’re right)!

● **Challenging**
  ▶ Challenging ourselves helps to challenge our students
  ▶ Doing something hard is unbelievably rewarding
What Makes an Assignment a Pearl?

● **Bentley:** “origins lie beyond solid engineering, in the realm of insight and creativity”
  - Grow from real problems
  - Teach important programming techniques and design principles
  - Fun

● **Simple in a complex way, reusable from year to year**
  - Each function/method may be short, interactions complex
  - Seems simple when finished, impossible when starting
  - Change assignment slightly or re-use as is
  - Difficult to develop these assignments, steal when possible
Intriguing and Challenging Assignments

DNA Sequencing (from Rich Pattis)

- Strands (fragments) of DNA are matched to form new strands
  - From two strands one strand is formed, find matches, repeat until one strand left or no matches found
  - `acggtcac` and `gtcacatta` match yielding `acggtcacatta`
  - Threshold used to avoid bad sequencing, matched substring must be some minimum length
    - In example above threshold of six means failure to match
    - Containing matches have no threshold
      - `acggtcac + cggtc = acggtcac`
Cross-displinary, life-science and ethics

- Human Genome Project
- Many web resources

- Hard to test the program
  - Students have trouble
  - Easy to mess up
  - I like it

- [www-bio.llnl.gov/bbrp/genome/genome.html](http://www-bio.llnl.gov/bbrp/genome/genome.html)
- [www.er.doe.gov/production/ober/hug_top.html](http://www.er.doe.gov/production/ober/hug_top.html)
DNA sequencing: an Overview

- **Reconstruct DNA molecule from strands**
  - Input file of strands form a molecule with right threshold
    - B0
      - tgaaaatcctttctatatttaggcccX
    - B1
      - atgcaatggcattagggcggttaaX
    - A2
      - ggttaaX
    - C0
      - tgaaaatcctttctatatttaggcccatgcaatX

- **Read strands, try to form matches yielding a molecule**
  - Two strands potentially match at many indexes
  - For every pair of strands, form match if possible, continue
DNA Class and Program Structure

- **What are responsibilities of a strand?**
  - Knows how to read itself from a file
  - Knows how to match itself against another strand
  - Knows how to construct itself from two matched strands

- **What are responsibilities of molecule/clone?**
  - Knows how to read strands? Does it really?
  - Engineers matches, but who does the matching?

- **Communication, e.g., of threshold:**
  - what is prototype of `Strand::match`?
Simple but Complex

● **Prototype for Strand::read given below**

```cpp
bool Strand::read(istream& input)
// pre: strand is constructed but empty
// post: reads input and forms strand
//       returns true if successful, else
//       returns false, strand state undefined
```

● **Write code to read an entire file, storing strands read in a vector, e.g., Clone::read(istream&)**
  - Assume fields Clone::myCount and Clone::myStrands
  - What are initial values of these fields, where are they set?

● **How do we test whether our code works?**
Lessons from DNA assignment

- All methods fewer than 25 lines without debugging
  - Print/debugging really helps, students avoid it
  - I supply a working executable to help
- It can be tricky to write 25 lines of code, especially when one method call generates several other calls
  - `Clone::process` calls `Strand::match` which calls `private Strand::isMatchAt`...
Huffman Coding

● Data compression is intrinsically important, but also very topical
  ▶ What is MP3? What is the principle difference between a .wav file and a .mp3 file? What is Napster?
  ▶ Why don’t you want to compress your course notes or tax statements into a .jpg file, but a .gif file is ok?

● Huffman coding: optimal per-character compression
  ▶ Doesn’t yield optimal compression for most files: frequency of characters isn’t uniform
  ▶ Used as part of .zip compression, second stage
  ▶ Conceptual framework for studying other kinds of compression, but relatively simple to implement
ASCII coding: towards compression

- Text is coded with seven bits for 128 characters
  - Why does a char have 8 bits?
- View ASCII table as a tree
  - Values in leaves
  - Root-to-leaf path is code
  - How many levels?
- Given ‘a’, we find 1100001
  - See diagram for tree view
  - Look up in table, not tree
- Use tree to read one bit at a time
  - Decode 110000111000111100101
Unused should mean not stored

- If we want to write “go go gophers”, eight-bit characters waste space
  ▶ 103 111 32 103 111 32 103 111 112 104 101 114 115
  11001111011111000001100111110111110111111000011110011001111011110111111000110100011001011110011

- There are eight characters in “go go gophers” we can make do with a three-bit code
  ▶ 0 1 7 0 1 7 0 1 2 3 4 5 6 7
  00001111000001110000010100011100101110111

Intriguing and Challenging Assignments
Exploiting Redundancy

- If 'g' and 'o' occur more often than 's' then we should use fewer bits to encode 'g' than 's'
  - Frequent chars higher in tree
  - Chars in leaves
  - What's the code for 'e'? What does 1101 encode?

- How is the tree built?
  - How do we pick deep leaves?
  - Encode and decode?

00 01 100 00 01 100 00 01 1110 1101 101 1111 1100
Intriguing and Challenging Assignments

Building the Huffman tree: greedy

- Count occurrences of every character

- Find two minimally weighted nodes, join as children of new node

- Number of nodes decreases, repeat until root left
Trees to Tables and Back

● To encode we need a table:
  ▶ How is \textit{pro shop} encoded
  ▶ How do we get table from tree?

● To decode we need a tree
  ▶ Decode 010011111011100
  ▶ Get tree from table?

● Can we store a tree in a file?
  ▶ Directly
  ▶ Indirectly (how is tree made?)
From Tree to Table (again)

● What are base cases and logic for code:

```c
void treeToTable(Tree * t,
    apvector<apstring>& codes, 
    const apstring& path)
// pre: path is 010..1 encoding from
//   overall root to t
// post: stores root-to-leaf path in codes
//   for every leaf

Tree * root = makeTree(counts);
treeToTable(root, codes, "")

● Why not return the vector from treeToTable?
Huffman coding in a nutshell

1. **Build a coding tree**
   A. Count characters, build tree greedily
   B. Other alternatives, e.g., for English text?

2. **From tree, build table of codes for each character**
   A. Tree traversal keeping track of path to leaf: store path
   B. Related to 2000 AP exam question?

3. **Create compressed file by re-reading input file**
   A. First write data that allows tree to be reconstructed
      i. Can we write counts? How many bits?
      ii. Can we write the tree? How many bits?
   B. Re-read input file, look up coding in table and write it
   C. Need pseudo-eof to mark end of compressed file
Creating Tree from Counts

- Assume existence of priority queue class
  - Supports insert, getmin, deletemin
  - Templated like apstack, apqueue
    - Can we have an apstack of vectors? Of stream?
    - appriority objects must implement operator <

- Cannot insert pointers to tree nodes into appriority
  - Why? How can we use object oriented concepts to help?
    - Use Adapter or Wrapper class, see code

- Add all initial nodes to priority queue, then repeat greed algorithm. Should we add zero-count nodes?
Unhuff in a nutshell

- Tree is stored in a file: directly or indirectly
  - Read the tree e.g., in pre-order form
  - Read counts, then re-create the tree
  - Which method should we use first?

- Once we have the tree, read one bit at a time
  - Go left/right in the tree according to value of bit
  - At leaf, print character and reset to root
  - When do we stop?

- Vanilla version is fewer than 80 lines of code!
Procedural or Object-Oriented

● What are the classes in this program?
  ▶ How do we get encoding for a character?
    • Simple to use a vector of strings, but consider a map
    • Map of keys to values: insert, lookup

  ▶ How do we get a tree?
    • Read counts, create tree
    • Read information in file, create tree,

  ▶ How do we get counts?
    • Read file
    • Assume standard frequency
Universal Acclaim for Huffman

- Compresses reasonably well, especially for a program done in first year of study
  - 30% on binary files, 44% on text files
  - Why not as good as zip?

- Uses vectors, trees, maps, priority queues
  - Stresses data structures, e.g., could build priority queue

- Allows creativity in implementation
  - Robust uncompression and magic numbers
  - Storing information in compressed file
  - Built from scratch, only bit reading/writing provided
Shoulders of Giants

Newton: “If I have seen farther it is because I have stood on the shoulders of giants.”

Robert Burton: “a dwarf standing on the shoulders of a giant may see farther than the giant himself.”