Programming in C Quick Start!

Biostatistics 615/815
Lecture 2
Loose Ends ...

- Policy on Academic Conduct
  - Copy included with today's handouts

- Course Website
  - www.sph.umich.edu/csg/abecasis/class/
Describe 3 algorithms for tackling the connectivity problem

- Quick Find
- Quick Union
- Weighted Quick Union
### Pictorial Comparison

<table>
<thead>
<tr>
<th>Quick Find</th>
<th>Quick Union</th>
<th>Weighted</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Tree 1" /></td>
<td><img src="image2" alt="Tree 2" /></td>
<td><img src="image3" alt="Tree 3" /></td>
</tr>
<tr>
<td><img src="image4" alt="Tree 4" /></td>
<td><img src="image5" alt="Tree 5" /></td>
<td><img src="image6" alt="Tree 6" /></td>
</tr>
<tr>
<td><img src="image7" alt="Tree 7" /></td>
<td><img src="image8" alt="Tree 8" /></td>
<td><img src="image9" alt="Tree 9" /></td>
</tr>
<tr>
<td><img src="image10" alt="Tree 10" /></td>
<td><img src="image11" alt="Tree 11" /></td>
<td><img src="image12" alt="Tree 12" /></td>
</tr>
<tr>
<td><img src="image13" alt="Tree 13" /></td>
<td><img src="image14" alt="Tree 14" /></td>
<td><img src="image15" alt="Tree 15" /></td>
</tr>
<tr>
<td><img src="image16" alt="Tree 16" /></td>
<td><img src="image17" alt="Tree 17" /></td>
<td><img src="image18" alt="Tree 18" /></td>
</tr>
</tbody>
</table>
Quick Find in C

// Data Initialization
for (i = 0; i < N; i++)
    a[i] = i;

// Loop through connections
while (scanf(" %d %d", &p, &q) == 2)
{
    // Check that input is within bounds
    if (p < 0 || p >= N || q < 0 || q >= N) continue;

    // FIND operation
    if (a[p] == a[q]) continue;

    // UNION operation
    set = a[p];
    for (i = 0; i < N; i++)
        if (a[i] == set)
            a[i] = a[q];

    printf("%d %d is a new connection\n", p, q);
}
Quick Union in C

// To start, place each element in its own group
for (i = 0; i < N; i++)
    a[i] = i;

// Loop through connections
while (scanf(" %d %d", &p, &q) == 2)
{
    // Check that input is within bounds
    if (p < 0 || p >= N || q < 0 || q >= N) continue;

    // FIND operation
    for (i = a[p]; a[i] != i; i = a[i]) ;
    for (j = a[q]; a[j] != j; j = a[j]) ;
    if (i == j) continue;

    // UNION operation
    a[i] = j;

    printf("%d %d is a new connection\n", p, q);
}
Weighted Quick Union in C

// Initialize groupings and weights
for (i = 0; i < N; i++)
    weight[i] = 1, a[i] = i;

// Loop through connections
while (scanf(" %d %d", &p, &q) == 2)
{
    // Check that input is within bounds
    if (p < 0 || p >= N || q < 0 || q >= N) continue;

    // FIND operation
    for (i = a[p]; a[i] != i; i = a[i] ) ;
    for (j = a[q]; a[j] != j; j = a[j] ) ;
    if (i == j) continue;

    // UNION operation
    if (weight[i] < weight[j])
        { a[i] = j; weight[j] += weight[i]; }
    else
        { a[j] = i; weight[i] += weight[j]; }

    printf("%d %d is a new connection\n", p, q);
}
Today

- Basics of programming in C
  - General organization of C programs
  - How to compile and debug a C program
  - C function library
Very Short History of C

• C was developed by Dennis Ritchie at Bell Labs (1969 – 72)
  • Support the new UNIX operating system
  • Successor to B and BCPL

• Strongly typed language

• Dynamic memory allocation

• User defined data structures
The Modern C Language

- Portable language
  - C compilers are available for desktop computers, mainframes and mobile phones

- Very efficient

- C++ is the successor to C
  - Simplifies grouping of functions and related data
Anatomy of C Program

- A collection of functions
  - Receive a set of parameters
  - Declare local variables
  - Carry out processing
  - Return a value

- `main()` function
  - Called to start the program
C libraries

- Most programs are not build from scratch

- Rely on pre-existing collections of functions
  - e.g. the Standard C library

- Header (.h) files describe functions in these collections
  - e.g. accessed through `#include` statements
A C function definition

type function(argument_list)
{
    variable_declarations;

    statements;
}

- Each function has a type
- Each function argument has a type
- Each local variable has a type
A simple C program

#include <stdio.h>

int main()
{
    printf("Hello, I am a program\n");

    return 0;
}
Variables in C

- Must be declared before use

- Each variable has a specific type
  - integer
  - floating point
  - character

- Names are case-sensitive
Another C Program

#include <stdio.h>

int Multiply(int x, int y)
{
    int product = x * y;

    return product;
}

int main()
{
    int x = 2;

    printf("%d * %d = %d\n", x, x, Multiply(x, x));

    return 0;
}
Another C Function

```c
int search(int a[], int value, int start, int stop)
{
    // Variable declarations
    int i;

    // Search through each item
    for (i = start; i <= stop; i++)
        if (value == a[i])
            return i;

    // Search failed
    return -1;
}
```
Programming Constructs in C

- Function definitions and calls
- Compound statements
- Flow-control
  - if ... else ...
  - do ... while
  - while ...
  - for ...
  - case ...
Compound Statements

- C statements can be grouped with `{ }`
- Each compound statement starts with optional local variable declarations
- Individual statements separated by ‘;’
if ... else ...

```java
if (expression)
    statement1;
else
    statement2;
```

- When expression is true (or nonzero) statement1 is executed; otherwise statement2 is executed.
Example

```c
void Compare(int a, int b)
{
    if (a == b)
        printf("Values Match!\n");
    else
        printf("Values are different!\n");
}
```
## Some Standard C Libraries

<table>
<thead>
<tr>
<th>Header File</th>
<th>Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>ctype.h</td>
<td>Information about characters</td>
</tr>
<tr>
<td>float.h</td>
<td>Information about floating point</td>
</tr>
<tr>
<td>limits.h</td>
<td>Information about integers</td>
</tr>
<tr>
<td>math.h</td>
<td>Common mathematical functions</td>
</tr>
<tr>
<td>stdio.h</td>
<td>Basic input / output functions</td>
</tr>
<tr>
<td>stdlib.h</td>
<td>Kitchen Sink!</td>
</tr>
<tr>
<td>string.h</td>
<td>String manipulation functions</td>
</tr>
<tr>
<td>time.h</td>
<td>Time</td>
</tr>
</tbody>
</table>
#include <stdio.h>
#include <math.h>

int main()
{
    double x;
    printf("Table of Sines and Cosines\n");
    printf("================================\n\n";)
    printf("      x  Sin(x)  Cos(x)\n";)
    for (x = 0.0; x <= 1.00; x += 0.20)
        printf(" %7.4f %7.4f %7.4f\n", x, sin(x), cos(x));
}
### Example Output

Table of Sines and Cosines

<table>
<thead>
<tr>
<th>x</th>
<th>Sin(x)</th>
<th>Cos(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0000</td>
<td>0.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>0.2000</td>
<td>0.1987</td>
<td>0.9801</td>
</tr>
<tr>
<td>0.4000</td>
<td>0.3894</td>
<td>0.9211</td>
</tr>
<tr>
<td>0.6000</td>
<td>0.5646</td>
<td>0.8253</td>
</tr>
<tr>
<td>0.8000</td>
<td>0.7174</td>
<td>0.6967</td>
</tr>
<tr>
<td>1.0000</td>
<td>0.8415</td>
<td>0.5403</td>
</tr>
</tbody>
</table>
math.h, Mathematical Functions

- double exp(double x);
  - exponential of x
- double log(double x);
  - natural logarithm of x
- double log10(double x);
  - base-10 logarithm of x
- double pow(double x, double y);
  - x raised to power y

- double sin(double x);
- double cos(double x); ...
  - Standard trigonometric functions

- double sqrt(double x);
  - square root of x
- double ceil(double x);
  - smallest integer not less than x
- double floor(double x);
  - largest integer not greater than x
- double fabs(double x);
  - absolute value of x
Executing C Code

- C is a high level language
  - Relatively easy to understand

- Computer CPUs execute much more detailed, "lower-level" instructions

- A compiler performs the necessary translation…
Working in a UNIX Environment

- GCC
  • Compile code

- GDB
  • Debug and test code

- GPROF
  • Collect profiling information
GCC

- GCC is a free C compiler
  - GNU C Compiler

- Versions available for
  - Linux
  - Unix
  - Mac
  - Windows

- Developed by Free Software Foundation
Simple Usage

```
gcc my_program.c -o my_program
```

- Uses text file with C code as input
- Produces executable program as output
GDB

- A simple debugger

- Helps test and evaluate programs by:
  - Stopping program at specific points
  - Running program one line at a time
  - Displaying the values of specific variables
Simple Usage

- Compile program with debug information
  
  gcc -pg my_program.c -o my_program

- Load program into debugger
  
  gdb my_program

- Use debugger commands to control execution of program …
Essential GDB Commands

- **run**
  - Start execution of the program
- **next**
  - Execute a single line of code
- **step**
  - Execute a single line of code, stop at entry point of any called functions
- **continue**
  - Execute program until next *breakpoint*
More Essential GDB Commands

- `break main`
  - Stop program when `main` function is called
- `break n`
  - Stop program when line `n` is reached
- `print x`
  - Print contents of variable `x`
- `info locals`
  - Print all local variables
GPROF

- Works with GCC to collect information about a program's execution
  - How often is each function called?
  - How much time is spent in each function?

- Works in three steps
  - Compile code
  - Execute program
  - Tabulate profile information
Simple Usage

- Compile program with profile information
  
  ```
  gcc -pg my_program.c -o my_program
  ```

- Execute program
  
  ```
  ./my_program
  ```

- Summarize profile information
  
  ```
  gprof my_program
  ```
Working in a Windows Environment

- Good integrated toolsets exist

- Good options include:
  - Microsoft Visual Studio / Visual C++
    - Discounted version available through the University
  - Borland C++ Builder X
    - Free download for personal use, [www.borland.com](http://www.borland.com)
Microsoft Visual Studio

- "Student Tools for Visual Studio"
  - Included in Academic Edition
  - Simplifies environment slightly

- Each application you develop is a project
  - Can include a collection of source files
  - But only one for most of our examples

- Many types of applications possible …
  - Our examples will be “Console Applications”
File | New | Project ...

New Project

Project Types:
- Visual Basic Projects
- Visual C# Projects
- Visual J# Projects
- Visual C++ Projects
- Setup and Deployment Projects
- Other Projects
- Visual Studio Solutions

Templates:
- AP C++ Console...
- ASP.NET Web Service
- ATL Project
- ATL Server Project
- ATL Server Web Service
- C++ Console...

Create a simplified C/C++ Console Application.

Name: MyApplication
Location: C:\code\Visual Studio Projects

Add to Solution
Close Solution

Project will be created at C:\code\Visual Studio Projects\MyApplication.
“All-In-One”

- Edit and manage source files
- Compile code
- Run, debug and test
```c
#include <stdio.h>

int main(int argc, char* argv[])
{
    printf("I am a computer program.\n\n");
    return 0;
}
```
Important Commands

- **Build | Build MyApplication**
  - Compiles code, identifies problems with source

- **Debug | Start, Step Into, Step Over**
  - Manage execution of program

- **Debug | QuickWatch**
  - Examine contents of variables as program runs

- **Click to the left of particular source lines to set breakpoints**
Today

- Organization of C programs
- How to compile / run simple programs
  - Unix / Linux
  - Windows