CompSci 101
Introduction to Computer Science

April 26, 2016
Prof. Rodger

ACM Meeting

• Duke ACM is the student chapter of the professional organization for Computer Scientists
• Election and matching up with CS Buddies
• Tuesday, 6:15pm – LSRC D106

One cookie per person

Hangman
Snarky

Announcements

• Last Day of class!
• Assign 9 due by Thursday
• APT 11 (optional) due tonight
• APT Quiz 3 Sun-Tue
• Final Exam: Sec 01 Tues 7pm, Sec 02 Fri 7pm
• Form for taking Final exam another time
  – accommodations?
  – Three exams in a 24 hour period?
  – Room for some to take final with the other section
  – Fill out by tonight for consideration!!!
More Announcements

- Regrade for Exam 2 – get to Prof Rodger soon
- Be a UTA for CompSci 101
  - Rewarding and Learning Experience
  - www.cs.duke.edu/csed/uta

Today:
- Wrapping up
- Beyond CompSci 101
- The Final exam

Final Exam

- Sec 01 – Tues, May 3, 7pm
- Sec 02 – Fri, May 6, 7pm
- Closed Book, Closed Notes, Closed neighbor
- Python Reference Sheet
- Covers all topics through today
- Best way to study is practice writing code!
- See old tests (no old final exams)

Final Exam (cont)

- Test format
  - Multiple choice
  - Writing code – similar to exam 2
- Topics include:
  - if, loops, lists, sets, dictionaries, files, functions, sorting, etc
  - recursion, regular expressions – reading level only

Calculate Your Grade

- From “About” tab on course web page

<table>
<thead>
<tr>
<th>Component</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Labs</td>
<td>10%</td>
</tr>
<tr>
<td>Reading Quizzes</td>
<td>5%</td>
</tr>
<tr>
<td>Class/Group work</td>
<td>5%</td>
</tr>
<tr>
<td>Apts</td>
<td>10%</td>
</tr>
<tr>
<td>Programming Assignments</td>
<td>10%</td>
</tr>
<tr>
<td>APT Quizzes</td>
<td>5%</td>
</tr>
<tr>
<td>Two Midterm Exams</td>
<td>30%</td>
</tr>
<tr>
<td>final exam</td>
<td>25%</td>
</tr>
</tbody>
</table>
More on Grades

- Lecture – ignore the first two weeks (drop/add period) plus drop 4 points
- Reading Quizzes – will drop 20 points
- Lab – drop 6 points (each lab is 4 pts)

Review - Selection Sort

- Sort a list of numbers.
- Idea:
  - Repeat til sorted
    - Find the smallest element in part of list not sorted
    - Put it where it belongs in sorted order.
      - Swap it with the element where it should be
- Sort example

Review - Bubble Sort

- Sort a list of numbers.
- Idea:
  - Repeat til sorted
    - Compare all adjacent pairs, one at a time. If out of order then swap them
- Sort example

Review - Insertion Sort

- Sort a list of numbers.
- Idea:
  - Sort by repeated inserting another element
    - Leftmost element is sorted part of list
    - Insert another element in that sublist keeping it sorted
    - Insert another element in that sublist keeping it sorted
    - Etc.
- Sort example
Review MergeSort idea for code

def mergesort(data):
    n = len(data)
    if n == 1:
        return data
    else:
        d1 = mergesort(data[:n//2])
        d2 = mergesort(data[n//2:]
    merge(d1, d2)

bit.ly/101sp16-0426-1

Question 1
Which sort is this?

4 10 5 3 8 2
4 10 5 3 8 2
4 5 10 3 8 2
3 4 5 10 8 2
3 4 5 8 10 2
2 3 4 5 8 10

Question 2
Which sort is this?

4 10 5 3 8 2
4 2 5 3 8 10
4 2 5 3 8 10
4 2 3 5 8 10
3 2 4 5 8 10
2 3 4 5 8 10

When done, please fill out course eval in Sakai!

Course Evaluation in Sakai

- Only 15% have filled it out as of 4/25
- Take some time to fill it out now

Wrap up Sorting

- Some Ways to Compare sorts.
  - How many total swaps?
  - Is one faster for certain types of input?
  - Does the input matter

- Different ways to sort?
  - Over 50 sorting algorithms

- Does President Obama know his sorts?

- Sorting animations
  http://www.sorting-algorithms.com/
More on Sorting in CompSci 201

- Learn about this and other sorts in CompSci 201, also how to analyze them to determine which one works best.
- Python: Timsort
  - combines mergesort and insertion sort
- Shellsort
  - uses insertion sort on parts of the list repeatedly - those parts getting larger each time

Scraping email address from websites

- Suppose we want to send email to all Duke Faculty to let them know …
  - Visit Departmental website, people, faculty
  - View (HTML) Source
  - Develop regex to access email – if possible!
- RegexScraper.py
  - Python makes this simple
  - Ethical hacking?

Duke Biology Dept Website

View page source of html
Scraping Biology faculty

- Pattern: 
  \-r'mailto:(\w+\.[\w\]*)@(\w+\.[\w\+\]*)' 
- URL 
  \- https://biology.duke.edu/people/all-faculty/a 
- Matches (call 26 times with different URL)

  \- ('emily.bernhardt', 'duke.edu') ('emily.bernhardt', 'duke.edu') ('bhandawat', 'gmail.com') ('bhandawat', 'gmail.com') ('jboynton66', 'gmail.com') ('jboynton66', 'gmail.com')

Scraping math.duke.edu faculty

old website

- Pattern: 
  \-r'math/faculty/([^/\s]+)"\>(.+?)\<' 
- URL - (MATH OLD SITE) 
  \- http://fds.duke.edu/db/aas/math/faculty/ 
- Matches:

  \- ('motta', 'Francis C. Motta') ('jmmza', 'James Murphy') ('ryser', 'Marc D. Ryser') ('sv113', 'Stefano Vigogna') ('haizhao', 'Haizhao Yang')

New Math Website - Scrape like biology
Scrapping Sanford/PubPol faculty

- Pattern:
  - `r'(^w+.^w*)@(^w+.^w*)'`

- URL
  - https://sanford.duke.edu/people/

- Matches (call 16 times with different URL)
  - ('schanzer', 'duke.edu')
  - ('steveschewel', 'gmail.com')
  - ('michael.schoenfeld', 'duke.edu')
  - ('schroeder', 'law.duke.edu')

What is Computing? Informatics?

- What is computer science, what is its potential?
  - What can we do with computers in our lives?
  - What can we do with computing for society?
  - Will networks transform thinking/knowing/doing?
  - Society affecting and affected by computing?
  - Changes in science: biology, physics, chemistry, ...
  - Changes in humanity: access, revolution (?), ...

- Privileges and opportunities available if you know code
  - Writing and reading code, understanding algorithms
  - Majestic, magical, mathematical, mysterious, ...

What can be programmed?

- What class of problems can be solved?
  - Linux, Cloud, Mac, Windows10, Android,…
  - Alan Turing contributions
    - Halting problem, Church-Turing thesis

- What class of problems can be solved efficiently?
  - Problems with no practical solution
  - What does practical mean?
Schedule students, minimize conflicts

- Given student requests, available teachers
  - write a program that schedules classes
  - Minimize conflicts

- Add a GUI too
  - Web interface
  - ...
  - ...

I can't write this program because I'm too dumb

Still another scenario, is this better?

One scenario

I can't write this program but neither can all these famous people

Summary of Problem Categories

- Some problems can be solved 'efficiently'
  - Run large versions fast on modern computers
  - What is 'efficient'? It depends

- Some cannot be solved by computer.
  - Provable! We can't wait for smarter algorithms

- Some problems have no efficient solution
  - Provably exponential $2^n$ so for "small" $n$ ...

- Some have no known efficient solution, but
  - If one does they all do!
Entscheidungsproblem
bit.ly/101sp16-0426-2

- What can we program?
  - What kind of computer?
- What can't we program?
  - Can't we try harder?
- Can we write a program that will determine if any program $P$ will halt when run on input $S$?

Problem: Traveling Band

- Band wants you to schedule their concerts.
- They don’t like to travel. Minimize the time they are on the bus!
- Given $N$ cities, what is the best schedule (shortest distance) to visit all $N$ cities once?

How do you calculate the best path?

- Try all paths
  - Atlanta, Raleigh, Dallas, Reno, Chicago
  - Dallas, Atlanta, Raleigh, Reno, Chicago
  - Etc.
- Would you agree to code this up?

Answer questions
bit.ly/101sp16-0426-2
How long?

<table>
<thead>
<tr>
<th>Number of Cities</th>
<th>All paths – N!</th>
<th>Time to solve - $10^9$ Instructions per second</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>3 million</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>$10^{12}$</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>$10^{15}$</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>$10^{18}$</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>$10^{25}$</td>
<td></td>
</tr>
</tbody>
</table>

How is Python like all other programming languages, how is it different?

A Rose by any other name…C or Java?

- Why do we use [Python | Java] in courses?
  - [is | is not] Object oriented
  - Large collection of libraries
  - Safe for advanced programming and beginners
  - Harder to shoot ourselves in the foot
- Why don't we use C++ (or C)?
  - Standard libraries weak or non-existant (comparatively)
  - Easy to make mistakes when beginning
  - No GUIs, complicated compilation model
  - What about other languages?

Why do we learn other languages?

- Perl, Python, PHP, Ruby, C, C++, Java, Scheme, Haskell,
  - Can we do something different in one language?
    - In theory: no; in practice: yes
  - What languages do you know? All of them.
  - In what languages are you fluent? None of them

- In later courses why do we use C or C++?
  - Closer to the machine, understand abstractions at many levels
Find all unique/different words in a file, in sorted order

**Unique Words in Python**

```python
def main():
    f = open('/data/melville.txt', 'r')
    words = f.read().strip().split()
    allWords = set(words)
    for word in sorted(allWords):
        print word

if __name__ == '__main__':
    main()
```

**Unique Words in Java**

```java
import java.util.*;
import java.io.*;
import java.util.Scanner;
import java.util.TreeSet;
public class Unique {
    public static void main(String[] args)
        throws IOException{
        Scanner scan =
                new Scanner(new File("/data/melville.txt"));
        TreeSet<String> set = new TreeSet<String>();
        while (scan.hasNext()){
            String str = scan.next();
            set.add(str);
        }
        for(String s : set){
            System.out.println(s);
        }
    }
}
```

**Unique words in Java**

```java
import java.util.*;
import java.io.*;
import java.util.Scanner;
import java.util.TreeSet;
public class Unique {
    public static void main(String[] args)
        throws IOException{
        Scanner scan =
                new Scanner(new File("/data/melville.txt"));
        TreeSet<String> set = new TreeSet<String>();
        while (scan.hasNext()){
            String str = scan.next();
            set.add(str);
        }
        for(String s : set){
            System.out.println(s);
        }
    }
}
```

**Unique Words in C++**

```cpp
#include <iostream>
#include <fstream>
#include <set>
using namespace std;

int main(){
    ifstream input("/data/melville.txt");
    set<string> unique;
    string word;
    while (input >> word){
        unique.insert(word);
    }
    set<string>::iterator it = unique.begin();
    for(; it != unique.end(); it++){
        cout << *it << endl;
    }
    return 0;
}
```
Unique words in PHP

```php
<?php
$wholething = file_get_contents("file:///data/melville.txt");
$wholething = trim($wholething);
$array = preg_split("/\s+/", $wholething);
$uni = array_unique($array);
sort($uni);
foreach ($uni as $word){
    echo $word."<br>";
}
?>
```

Kernighan and Ritchie

- First C book, 1978
- First ‘hello world’
- Ritchie: Unix too!
  - Turing award 1983
- Kernighan: tools
  - Strunk and White

- Everyone knows that debugging is twice as hard as writing a program in the first place. So if you are as clever as you can be when you write it, how will you ever debug it?

Brian Kernighan

How do we read a file in C?

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int strcompare(const void * a, const void * b){
    char ** stra = (char **) a;
    char ** strb = (char **) b;
    return strcmp(*stra, *strb);
}

int main(){
    FILE * file = fopen("/data/melville.txt","r");
    char buf[1024];
    char ** words = (char **) malloc(5000*sizeof(char **));
    int count = 0;
    int k;
    while (fscanf(file,"%s",buf) != EOF){
        int found = 0;  // look for word just read
        for(k=0; k < count; k++){
            if (strcmp(buf,words[k]) == 0){
                found = 1;
                break;
            }
        }
        if (!found){     // not found, add to list
            words[count] = (char *) malloc(strlen(buf)+1);
            strcpy(words[count],buf);
            count++;
        }
    }
    return 0;
}
```

Storing words read when reading in C

- Complexity of reading/storing? Allocation of memory?
Sorting, Printing, Freeing in C

```c
qsort(words, count, sizeof(char *), strcompare);
for(k=0; k < count; k++) {
    printf("%s\n", words[k]);
}
for(k=0; k < count; k++) {
    free(words[k]);
}
free(words);
```

• Sorting, printing, and freeing
  - Ugh!