Announcements

• Last Day of class!
• Assign 8 – With LATE Penalty thru Fri, Dec 9
• Assign 9 by Monday, none accepted after that
• APT 10 due by Monday, no Late APTs

• 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 tomorrow for consideration!!!

More Announcements

• Regrade for Exam 2 – submit by Friday, Dec 9
• Review Session – Tuesday 4pm (LSRC B101)
• Last Consulting Hours tonight
• Prof. Rodger will have office hours
  – Today 3-5pm, Tomorrow 2-5pm, more…
• Concern form – last minute concerns

Final Exam

• Sec 01– Monday, Dec 19, 2pm, **LSRC B101**
• Sec 02 – Thurs, Dec 15, 7pm, **Bio Sci 111**
• 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>Percentage</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 30 points
  • Lots of problems with Sakai this semester
  • Check your grades to make sure they copied over – fill out duke oit help form if they are wrong
• Lab – drop 6 points (each lab is 4 pts)
  • 44 pts total– 38 pts is 100%

Fill out Duke Course Eval

• Please fill out Duke Course Eval on DukeHub now
  – Only 15% have filled it in as of last night
• If you already have , then go to Sakai and fill out feedback on UTAs
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

Selection Sort (cont.)

1 3 4 9 5 6 - end of 3rd pass

1 3 9 5 6 - find smallest, swap

1 3 4 5 9 6 - end of 4th pass

1 3 4 5 9 6 - find smallest, swap

1 3 4 5 6 9 - end of 5th pass, done

Selection Sort – red area sorted

9 5 4 1 3 6 - find smallest, swap

1 5 4 9 3 6 - end of 1st pass

1 5 4 9 3 6 - find smallest, swap

1 3 4 9 5 6 - end of 2nd pass

1 3 4 9 5 6 - find smallest, swap

1 3 4 5 9 6 - end of 4th pass

1 3 4 5 6 9 - end of 5th pass, done

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
Bubble Sort

- Sort the list of numbers using BubbleSort.
- The body of the loop is one pass.
- Show the elements after each pass.
- [9, 5, 4, 1, 3, 6]

Two more passes would guarantee sorted.
Or Check if sorted and skip last two passes.
Code for Bubblesort

```python
def bubblesort(data):
    for j in range(len(data)-1, 0, -1):
        print(data)
        for k in range(0, j):
            if data[k] > data[k+1]:
                data[k], data[k+1] = data[k+1], data[k]
    return data
```

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

Insertion Sort – red area sorted

- Sort the list of numbers using InsertionSort.
- The body of the loop is one pass.
- Show the elements after each pass.
- [9, 5, 1, 4, 3, 6]
Insertion Sort – red area sorted

9 3 4 5 6 9 - 5th pass

Merge Sort

• Idea: Divide and Conquer
• Divide list into two halves
• Sort both halves (smaller problem)
• Merge the two sorted halves

9 5 1 4 3 6 2 7

Merge Sort

• Idea: Divide and Conquer
• Divide list into two halves
• Sort both halves (smaller problem)
• Merge the two sorted halves

9 5 1 4 3 6 2 7
9 5 1 4 3 6 2 7
divide list into 2 halves
1 4 5 9 2 3 6 7 recursively sort each half
Merge Sort

- Idea: Divide and Conquer
- Divide list into two halves
- Sort both halves (smaller problem)
- Merge the two sorted halves

9 5 1 4 3 6 2 7
9 5 1 4 3 6 2 7  divide list into 2 halves
1 4 5 9 2 3 6 7  recursively sort each half
1 2 3 4 5 6 7 9  merge the two sorted list

What does recursively sort mean?

Merge Sort

- Use the same Merge Sort algorithm
  - Divide list into two halves
  - Sort both halves (smaller problem)
  - Merge the two sorted halves

9 5 1 4
9 5 1 4  divide list into 2 halves
5 9 1 4  recursively sort each half
1 4 5 9  merge the two sorted list

MergeSort idea for code

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

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

bit.ly/101f16-1208-3
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?
Scraping Biology faculty

- Pattern:
  - `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 Sanford/PubPol faculty

- **Pattern:**
  - `\w+\.[\w\s]*@\w+\.[\w\s]+`'
- **URL**
  - [https://sanford.duke.edu/people/](https://sanford.duke.edu/people/)
- **Matches (call 26 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, …

Computing - solve all problems?

- 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!

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?

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>

Answer questions

bit.ly/101f16-1208-4
<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>&lt; sec</td>
</tr>
<tr>
<td>15</td>
<td>10^{12}</td>
<td>16 min</td>
</tr>
<tr>
<td>18</td>
<td>10^{15}</td>
<td>11 days</td>
</tr>
<tr>
<td>20</td>
<td>10^{18}</td>
<td>31 years</td>
</tr>
<tr>
<td>25</td>
<td>10^{25}</td>
<td>10^8 years</td>
</tr>
</tbody>
</table>

**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>&lt; sec</td>
</tr>
<tr>
<td>15</td>
<td>10^{12}</td>
<td>16 min</td>
</tr>
<tr>
<td>18</td>
<td>10^{15}</td>
<td>11 days</td>
</tr>
<tr>
<td>20</td>
<td>10^{18}</td>
<td>31 years</td>
</tr>
<tr>
<td>25</td>
<td>10^{25}</td>
<td>10^8 years</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?

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()
```

Find all unique/different words in a file, in sorted order
Unique words in Java

```java
import java.util.*;
import java.io.*;
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>";
}
?>
```

End with A CS Story

bit.ly/101fl6-1208-5