Announcements

• Last Day of class!
• Assign 9 by Monday, none accepted after that
• Assign 8, late by Friday, Dec 8!
• APT 8 due tonight, late by Sunday
• Form for taking Final exam another time
  – accommodations?

More Announcements

• Regrade for Exam 2 – submit by today
• Last Consulting Hours tonight
• Prof. Rodger extra office hours this week
  – Today 4:45-5:45pm, Friday 2:30-4:30pm
• Review Session Tues, Dec 12
  – LSRC B101, 4pm-5:30pm
• Concern form on forms page

Today:
  – Sorting, Wrapping up, Beyond CompSci 101
  – The Final exam

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>5%</td>
</tr>
<tr>
<td>Reading Quizzes</td>
<td>5%</td>
</tr>
<tr>
<td>Lecture Group work</td>
<td>5%</td>
</tr>
<tr>
<td>Apts</td>
<td>12%</td>
</tr>
<tr>
<td>Programming Assignments</td>
<td>12%</td>
</tr>
<tr>
<td>APT Quizzes</td>
<td>6%</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
  - 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%
  - Lab 11 covers two new topics!

Final Exam

- Sec 01– Thurs, Dec 14, 9am, LSRC B101
- Sec 02 – Sat, Dec 16, 2pm, LSRC B101
- 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

Time for Duke Course Eval and Seven Steps

1. Please fill out Duke Course Eval on DukeHub now
   1. Only 17% have filled it in as of last night
2. Anonymous feedback on the Seven Steps Announcement on Sakai and I emailed you
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

<table>
<thead>
<tr>
<th>Sorted, won’t move</th>
<th>final position</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>???</td>
</tr>
</tbody>
</table>

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 9 5 6 - end of 3rd pass, done

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

<table>
<thead>
<tr>
<th>Sorted, won’t move</th>
<th>final position</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>???</td>
</tr>
</tbody>
</table>
Bubble Sort – red area sorted

9 5 4 1 3 6 - compare, swap
5 9 4 1 3 6 - compare, swap
5 4 9 1 3 6 - compare, swap
5 4 1 9 3 6 - compare, swap
5 4 1 3 9 6 - compare, swap
5 4 1 3 6 9 - end of 1st pass
5 4 1 3 6 9

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

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

| Sorted relative to each other | ??? |
Insertion Sort

- Sort the list of numbers using InsertionSort.
- The body of the loop is one pass.
- Show the elements after each pass.
- [6, 4, 9, 7, 1, 3]

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
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

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:]).
        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

Timingsorts.py, what sort to call?

- Simple to understand, hard to do fast and at-scale
  - Scaling is what makes computer science …
    - Efficient algorithms don't matter on lists of 100 or 1000
  - Named algorithms in 201 and other courses
    - bubble sort, selection sort, mergesort, quicksort, …
    - See next slide and TimingSorts.py
- Basics of algorithm analysis: theory and practice
  - We can look at empirical results, would also like to be able to look at code and analyze mathematically! How does algorithm scale?
New sorting algorithms happen …

• timsort is standard on…
  - Python as of version 2.3, Android, Java 7
  - According to http://en.wikipedia.org/wiki/Timsort
    • Adaptive, stable, natural mergesort with supernatural performance

• Mergesort? Fast and Stable
  - What does this mean?
  - Which is most important?
  - Nothing is faster, what does that mean?
  - Quicksort is faster, what does that mean?

TimingSorts.py

<table>
<thead>
<tr>
<th>size</th>
<th>create</th>
<th>bubble</th>
<th>select</th>
<th>timsort</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>0.026</td>
<td>0.127</td>
<td>0.081</td>
<td>0.002</td>
</tr>
<tr>
<td>2000</td>
<td>0.045</td>
<td>0.537</td>
<td>0.273</td>
<td>0.001</td>
</tr>
<tr>
<td>3000</td>
<td>0.058</td>
<td>1.126</td>
<td>0.646</td>
<td>0.002</td>
</tr>
<tr>
<td>4000</td>
<td>0.082</td>
<td>2.174</td>
<td>1.208</td>
<td>0.003</td>
</tr>
<tr>
<td>5000</td>
<td>0.101</td>
<td>3.521</td>
<td>1.862</td>
<td>0.003</td>
</tr>
<tr>
<td>6000</td>
<td>0.118</td>
<td>4.617</td>
<td>3.005</td>
<td>0.004</td>
</tr>
<tr>
<td>7000</td>
<td>0.168</td>
<td>7.504</td>
<td>4.237</td>
<td>0.005</td>
</tr>
<tr>
<td>8000</td>
<td>0.156</td>
<td>9.074</td>
<td>6.152</td>
<td>0.007</td>
</tr>
<tr>
<td>9000</td>
<td>0.184</td>
<td>11.611</td>
<td>8.089</td>
<td>0.007</td>
</tr>
<tr>
<td>10000</td>
<td>0.212</td>
<td>14.502</td>
<td>9.384</td>
<td>0.008</td>
</tr>
</tbody>
</table>

TimingSorts.py Questions

bit.ly/101f17-1207-3

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

• 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](https://biology.duke.edu/people/all-faculty)
  
- **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]*)@((\w+[.\w]+)`

- **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?
Traveling Band questions
bit.ly/101f17-1207-4

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-existent (comparatively)
  - Easy to make mistakes when beginning
  - No GUIs, complicated compilation model
  - What about other languages?
Find all unique/different words in a file, in sorted order

Unique Words in 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()
<?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>";
}
?>