Announcements

- Last Day of class!
- Assign 9 by Friday, none accepted after that
- APT 9 due by Thursday, no Late APTs after that
- Form for taking Final exam another time
  - accommodations?
  - Three exams in a 24 hour period?
  - Room to take final with the other section
  - Fill out by Friday for consideration!!!
Calculate Your Grade

• From “About” tab on course web page

<table>
<thead>
<tr>
<th>Labs</th>
<th>5%</th>
</tr>
</thead>
<tbody>
<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%

Final Exam

• Sec 01– Sat., May 6, 9am, LSRC B101
• Sec 02 – Tues, May 2, 7pm, 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
Fill out Duke Course Eval

- Please fill out Duke Course Eval on DukeHub now
  - Only 7% 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

| Sorted, won’t move final position | ??? |

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 - find smallest, swap

Selection Sort (cont.)

1 3 4 9 5 6 - end of 3rd pass
1 3 4 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
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>???</th>
<th>Sorted, won't move final position</th>
</tr>
</thead>
</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
Bubble Sort
bit.ly/101s17-0425-1
• Sort the list of numbers using BubbleSort.
• The body of the loop is one pass.
• Show the elements after each pass.
• [6, 4, 9, 7, 1, 3]

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

| Sorted relative to | each other | ??? |

Insertion Sort – red area sorted

9 5 1 4 3 6 - insert 5
5 9 1 4 3 6 - 1st pass, now insert 1
1 5 9 4 3 6 - 2nd pass, now insert 4
1 4 5 9 3 6 - 3rd pass, now insert 3
1 3 4 5 9 6 - 4th pass, now insert 6
Insertion Sort – red area sorted

1 3 4 5 6 9 - 5th pass

Insertion Sort
bit.ly/101s17-0425-2

• 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
**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

---

**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)
```
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:
  - `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 Sanford/PubPol faculty

- Pattern:
  \[ r'\left(\w+\[.\w\]*\right)@\left(\w+\[.\w+\]*\right)' \]
- URL
  \[ https://sanford.duke.edu/people/ \]
- 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?

Answer questions
bit.ly/101s17-0425-4

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>
<tr>
<td>Number of Cities</td>
<td>All paths – N!</td>
<td>Time to solve - $10^9$ Instructions per second</td>
</tr>
<tr>
<td>------------------</td>
<td>----------------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>10</td>
<td>3 million</td>
<td>&lt; sec</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 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>

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?

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()
End with A CS Story
bit.ly/101s17-0425-5