CompSci 101
Introduction to Computer Science

Dec 7, 2017

Prof. Rodger

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

Labs	5%
Reading Quizzes	5%
Lecture Group work	5%
Apts	12%
Programming Assignments	12%
APT Quizzes	6%
Two Midterm Exams	30%
final exam	25%

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

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

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

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

???	*Sorted, won't move final position*

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

Bubble Sort – red area sorted

5 4 1 3 6 | 9 - compare, swap
4 5 1 3 6 | 9 - compare, swap
4 1 5 3 6 | 9 - compare, swap
4 1 3 5 6 | 9 - compare, no swap
4 1 3 5 6 | 9 - end of 2cd pass
4 1 3 5 | 6 9

Bubble Sort – red area sorted

4 1 3 5 | 6 9 - compare, swap
1 4 3 5 | 6 9 - compare, swap
1 3 4 5 | 6 9 - compare, no swap
1 3 4 5 | 6 9 - end of 3rd pass
1 3 4 | 5 6 9

Two more passes would guarantee sorted.

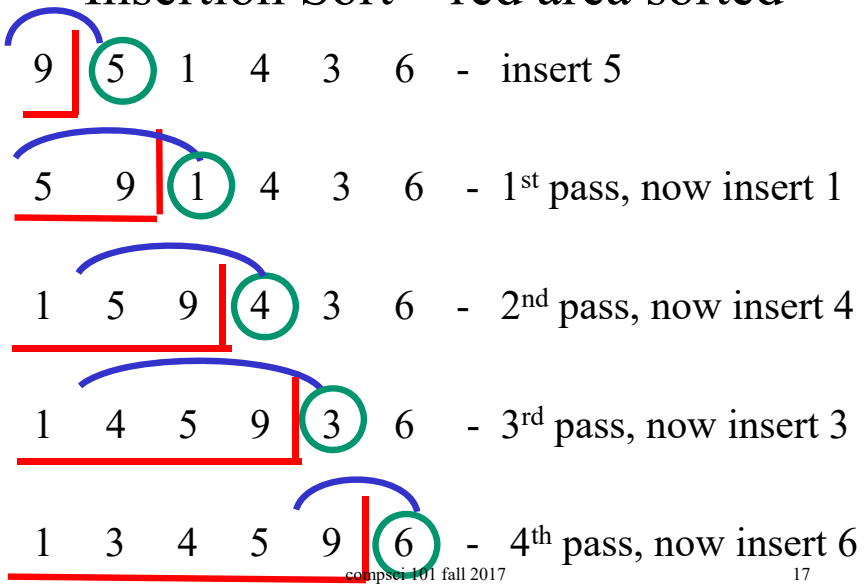
Or Check if sorted and skip last two passes

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 – red area sorted



Insertion Sort – red area sorted

Insertion Sort
bit.ly/101f17-1207-1

- 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)
```


bit.ly/101f17-1207-2

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 mathemetically! 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

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

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

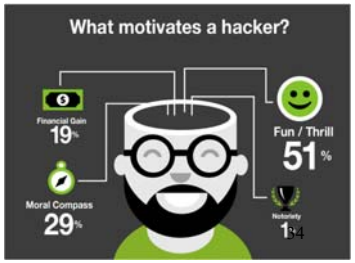
  http://www.sorting-algorithms.com/

More on Sorting in CompSci 201

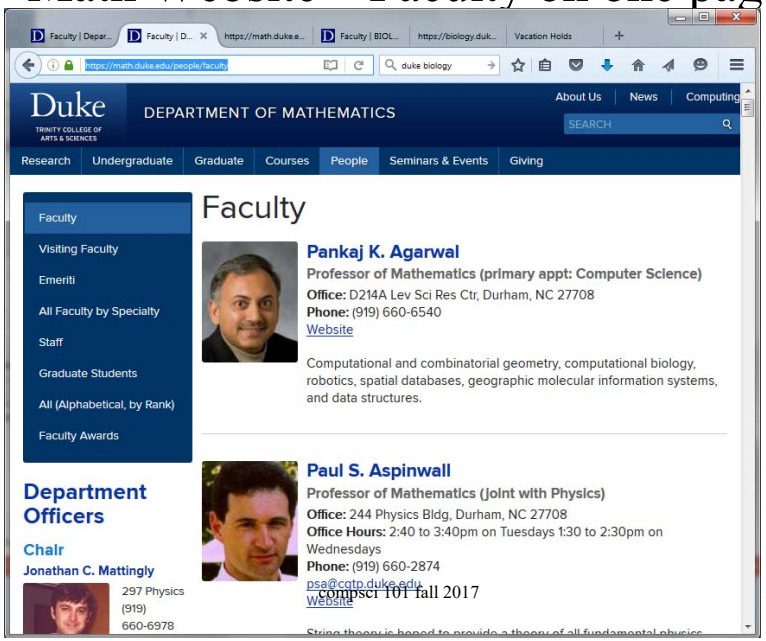
- Learn about this and other sorts in CompSci 201, also how to analyze them to determine which one works best.
- Python: Timsort
  - combines mergesort and insertion sort
- Shellsort
  - uses insertion sort on parts of the list repeatedly - those parts getting larger each time

Scraping email address from websites

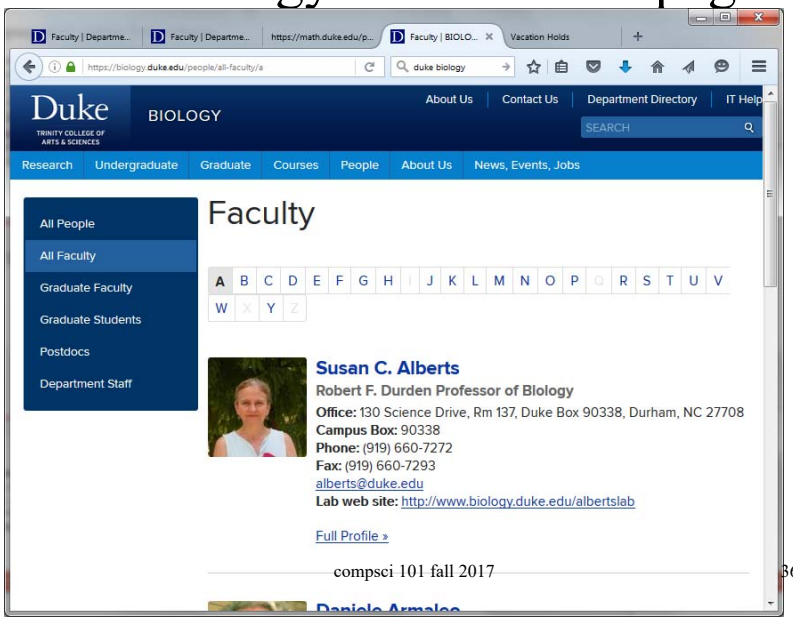
- Suppose we want to send email to all Duke Faculty to let them know …
  - Visit Departmental website, people, faculty
  - View (HTML) Source
  - Develop regex to access email – if possible!
- RegexScraper.py
  - Python makes this simple
  - Ethical hacking?



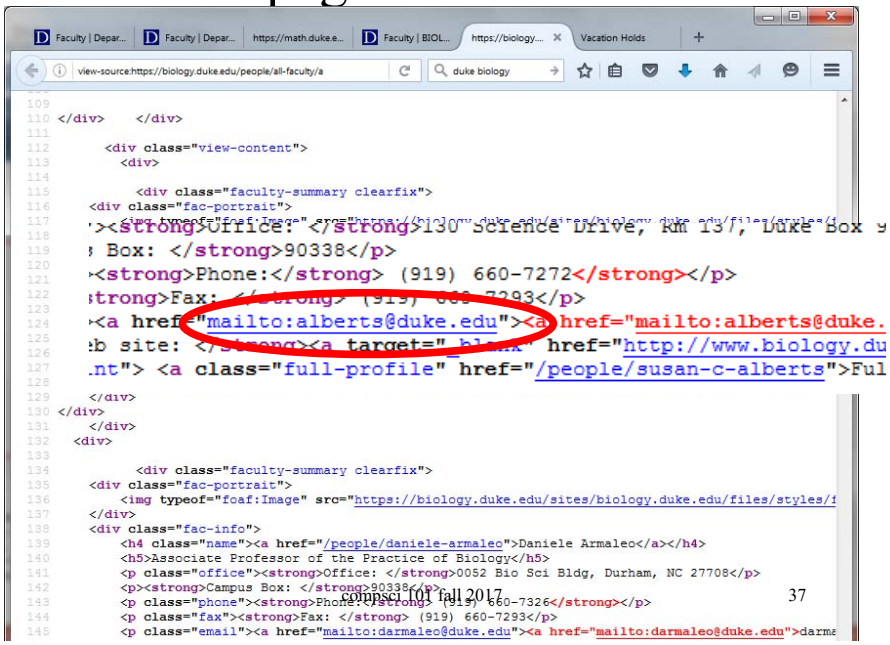
Math Website – Faculty on one page



Duke Biology Website A-Z pages



View page source of html

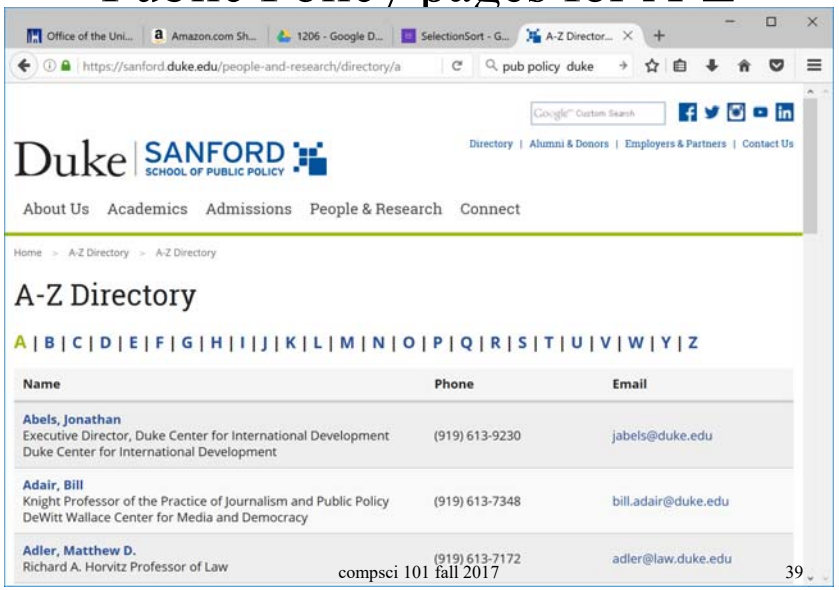


Scraping Biology faculty

- 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')

Public Policy pages for A-Z



Scraping Sanford/PubPol faculty

- Pattern:
  - `r'(\w+[.\w]*)@(\w+[.\w+]*)'`
- 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?

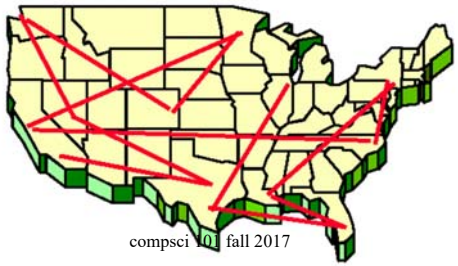
- What is computer science, what is its potential?
  - What can we do with computers in our lives?
  - What can we do with computing for society?
  - Will networks transform thinking/knowing/doing?
  - Society affecting and affected by computing?
  - Changes in science: biology, physics, chemistry, …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

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

How long?

Number of Cities	All paths – N!	Time to solve - 10^9 Instructions per second
10	3 million	
15	10^{12}	
18	10^{15}	
20	10^{18}	
25	10^{25}	

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?

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()
```


Unique words in 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++

```
#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

$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/101f17-1207-5