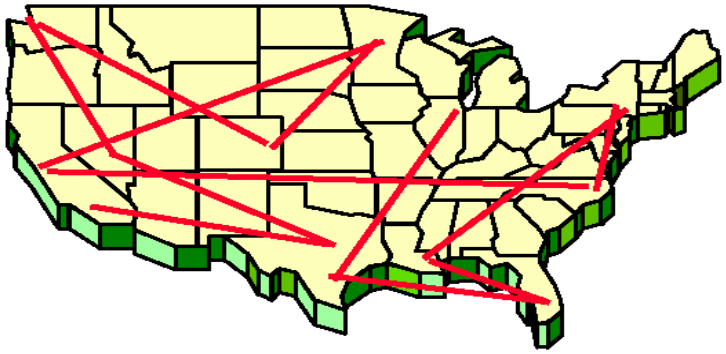
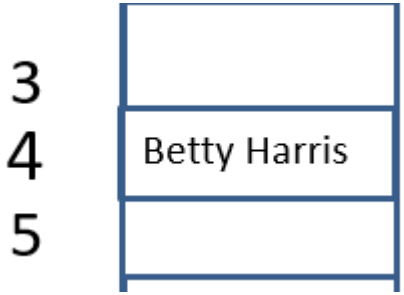
# Compsci 101
# Recursion and Beyond
# Last Lecture



Susan Rodger

April 25, 2023



# `Y` is for …

- **YAML and YACC**
  - Yet Another …
- **Y2K:** https://www.youtube.com/watch?v=rblt2EtFfC4
  - How many bits are enough bits?
- **YouTube**
  - Connected to computing …

# Z is for …

- **Zero**
  - There are two, or 10 bits in the universe
- **Zip**
  - Compressed file archive format
- **Zork**
  - Text-based adventure game
  - https://www.youtube.com/watch?v=TNN4VPlRBJ8

# Raja Kushalnagar

- **Professor Gallaudet University**
- **PhD CS and MS of Laws (LLM) in Intellectual Property and Information Law from Univ Houston**
- **Juris Doctor (JD) Texas Southern University**
- **As a Deaf professor, he advocates in bringing consumers, industry, and policymakers together on accessibility issues, advocating for a deaf/hard of hearing perspective, as well as developing prototype technologies for improving the accessibility of such systems.**

# Announcements

- **APT-7 due today**
  - Today is last day for Office Hours
  - One grace day, NO LATE DAYS!
  - MUST TURN in BY tomorrow
- **Assign 7 Create due, tomorrow**
  - Extended Grace period is through Sunday, April 30
  - No turnin's after that.
  - You turn in on SAKAI!
- **Last time for UTA office hours is tonight!**

- **Exam 3 …. Regrade requests thru Thursday night**
  - Everyone gets 2 extra points on Exam 3!!!!

# PFTD

- **Final Exam**
- **How do dictionaries work so fast?**
- **Finishing up**
  - What more is there in CS?

# Final Exam – Thurs, May 4 9am - Noon

- **Exam is in our regular classroom, Griffith Theatre**
- **Set three alarms!!!!!!**

# Final Exam

- **Study like you studied for Exam 3**
  - Use Exam 3 handout
- **We only have a little material since then**
  - Recommender
    - – this is all about stuff we did before
  - Modules
    - Exceptions
  - Recursion – reading only, no writing
- **Not on the exam**
  - Images, turtles, exceptions, writing classes (lab 10)

# Calculate Your Grade

- From "About" tab on course web page

| | |
|---|---|
| Labs | 10% |
| Sakai Quizzes | 5% |
| Class Participation (WOTOs) | 5% |
| Apts | 10% |
| Programming Assignments | 10% |
| APT Quizzes | 10% |
| Three Exams(12% each) and Final(14%) | 50% |

- Final exam is required
- Use your final exam grade to replace your lowest test grade

# More on Grades

- Class Participation-WOTOs – **ignore** the first two weeks (drop/add period), plus drop 10 **points**
- Sakai Quizzes **299** points– will drop **39** points
  - Your points/260, can't get more than 100
- Lab – drop **15** points (each lab is 5 pts)
  - Except Lab 10 that was required
- That is all we drop

# Time for CompSci 101 Course Eval

1. **Please fill out Duke Course Eval and give me feedback on the course!**

# How do Dictionaries work so fast?

- **How are they implemented?**

# Review: Problem – which word occurs the most frequently in a file

- **Need to count how many times each word occurs**
  - slowcount – for each word in a set, calls count
  - fastcount – counting dictionary

# slowcount function
# Short Code and Long Time

- **See module WordFrequencies.py**
  - Find # times each word in a list of words occurs
  - We have tuple/pair: word and word-frequency

```
37  def slowcount(words):
38      pairs = [(w,words.count(w)) for w in set(words)]
39      return sorted(pairs)
```

- **Think: How many times is `words.count(w)` called?**
  - Why is **`set(words)`** used in list comprehension?

# Using fastcount

- **Update count if we've seen word before**
  - Otherwise it's the first time, occurs once

```
28  def fastcount(words):
29      d = {}
30      for w in words:
31          if w in d:
32              d[w] += 1
33          else:
34              d[w] = 1
35      return sorted(d.items())
```

# Let’s run them and compare them!

- **Run with Melville and observe time**
  - slowcount about 0.76 seconds
  - fastcount about 0.00 seconds

- **Run with Hawthorne and observe time**
  - slowcount about 14.6 seconds
  - fastcount about 0.03 seconds

# Here is the idea  behind how dictionaries work

# Simple Example

## Want a mapping of Soc Sec Num to Names

- **Duke's CS Student Union wants to be able to quickly find out info about its members. Also add, delete and update members. Doesn't need members sorted.**

  **267-89-5431 John Smith**

  **703-25-6141 Ademola Olayinka**

  **319-86-2115 Betty Harris**

  **476-82-5120 Rose Black**

- **Dictionary d – SSN to names**
  - d['267-89-5431'] = 'John Smith'
  - How does it find 'John Smith' so fast?

# Dictionary d(SSN) = (SSN, name)

- **We actually would map the SSN to the tuple of (SSN, name).**
- **That is a lot to display on a slide, so we will just show SSN to name**
- **But remember name is really a tuple of (SSN,name)**

# Simple Example

## Let's look under the hood.
## How are dictionaries implemented?

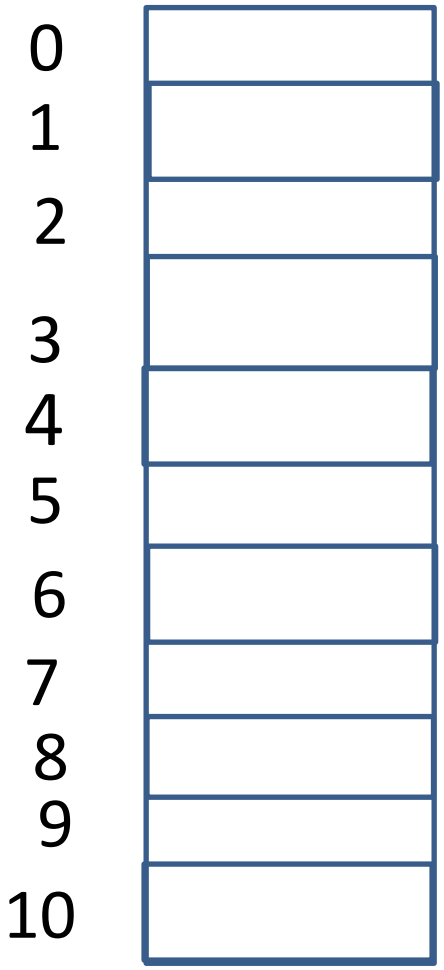
- **Dictionaries implemented with a list, in a clever way**
- **How do we put something into the list fast?**
- **How do we find it in the list quickly?**
  - **d['267-89-5431'] = 'John Smith'**
- **List size is 11 – from 0 to 10**
- **d['267-89-5431'] calculates index location in list of where to put this tuple (SSN,name)**
- **Use a function to calculate where to store 'John Smith**
  - H(ssn) = (last 2 digits of ssn) mod 11
  - Called a Hash function

# Have a list of size 11 from 0 to 10

- **Insert these into the list**
- **Insert as (key, value) tuple**

**(267-89-5431, John Smith)**

**(in example, only showing name)**

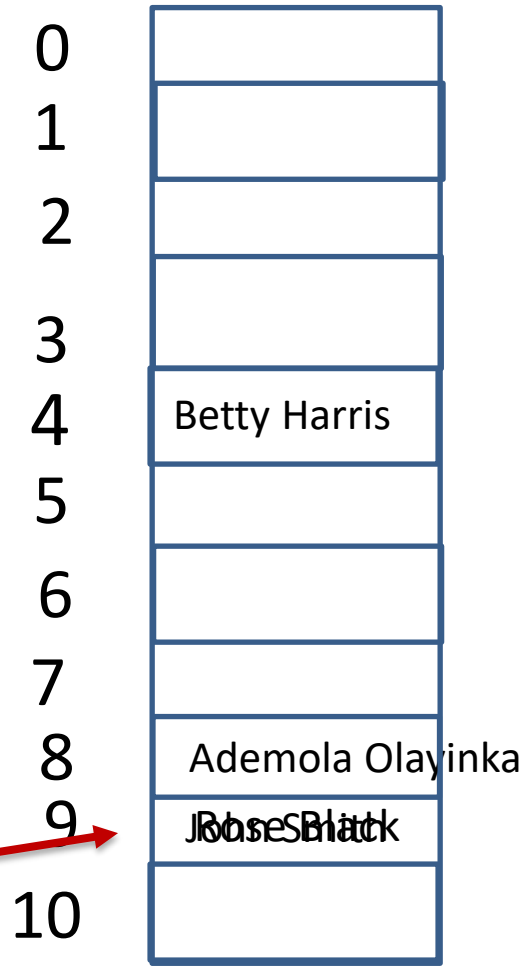


# Have a list of size 11 from 0 to 10

- **Insert these into the list**
- **Insert as (key, value) tuple**

**(267-89-5431, John Smith)**
**(in example, only showing name)**

**H(267-89-5431) = 31 %11 = 9**
**John Smith**
**H(703-25-6141) = 41%11 = 8**
**Ademola Olayinka**
**H(319-86-2115 ) = 15 %11 = 4**
**Betty Harris**
**H(476-82-5120) = 20%11 = 9**
**Rose Black**




Collision!

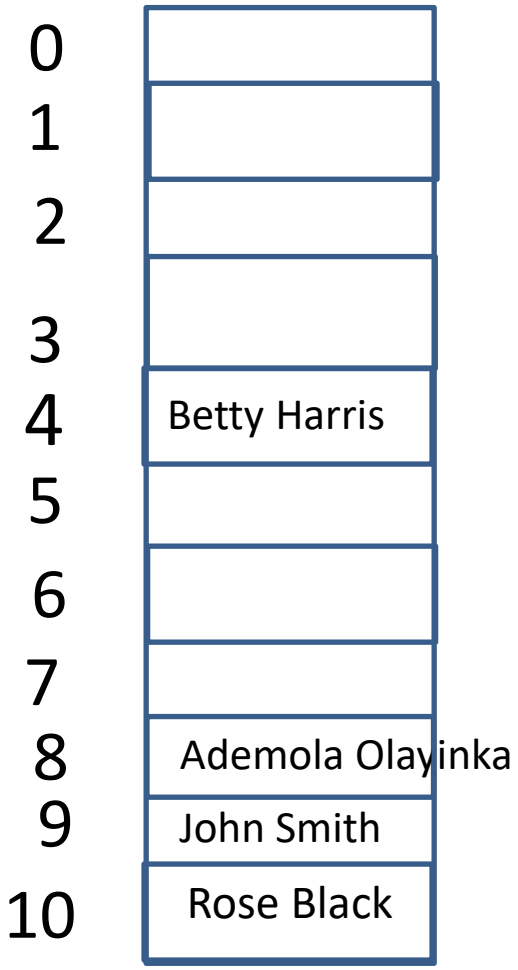
# Have a list of size 11 from 0 to 10

- **Insert these into the list**
- **Insert as (key, value) tuple**

**(267-89-5431, John Smith)**
**(in example, only showing name)**

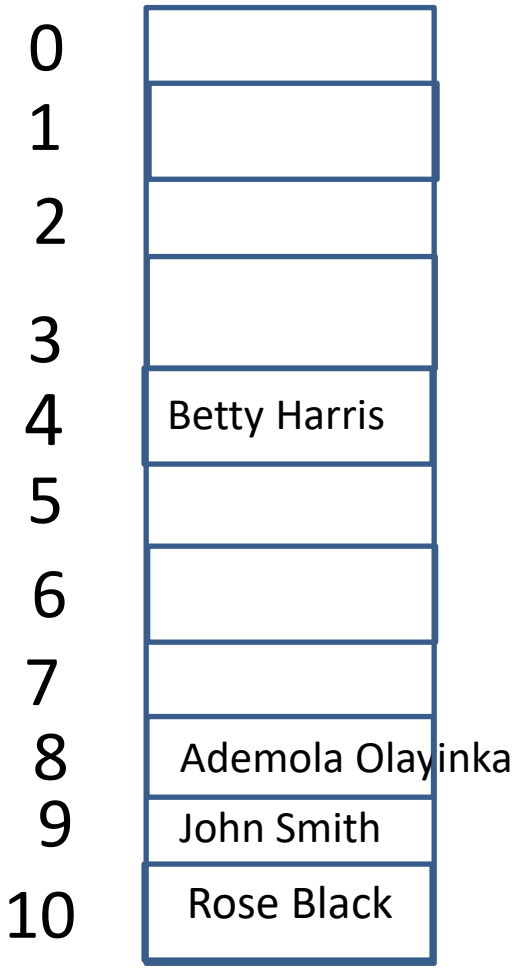
**H(267-89-5431) = 31 %11 = 9**
**John Smith**
**H(703-25-6141) = 41%11 = 8**
**Ademola Olayinka**
**H(319-86-2115 )= 15 %11 = 4**
**Betty Harris**
**H(476-82-5120) = 20%11 = 9**
**Rose Black**

Must resolve collisions



# When does this work well?

- **When there are few collisions**
- **You have to deal with collisions**
- **Use a list large enough to spread out your data**

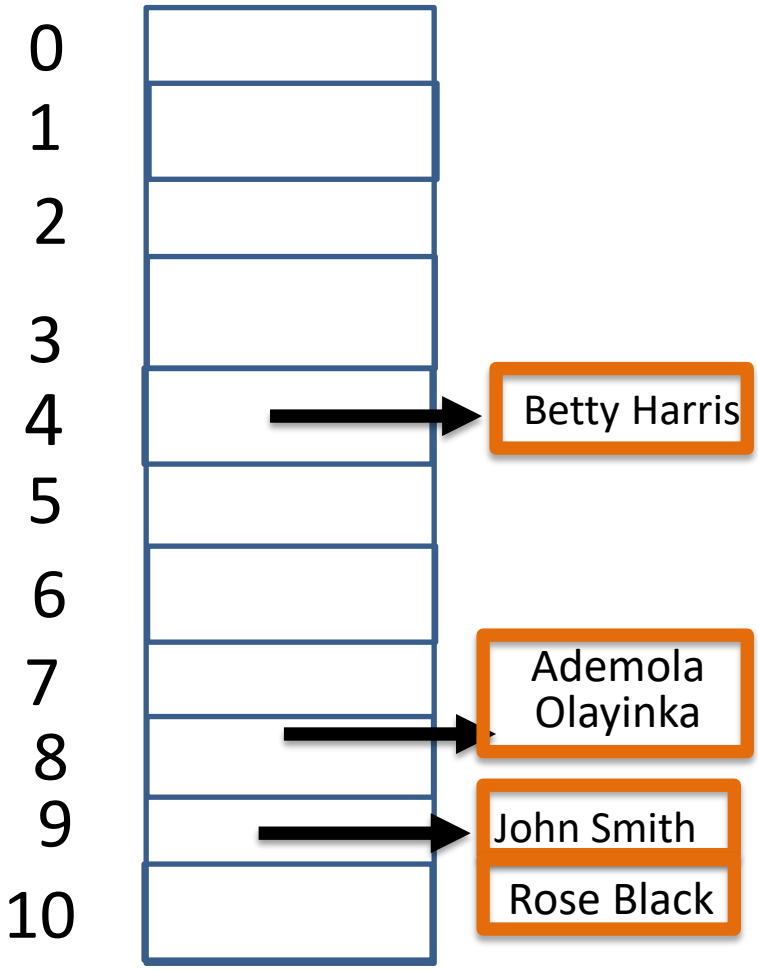


# Another way: Use a list of lists

- **Insert these into the list**
- **Insert as (key, value) tuple**

**(267-89-5431, John Smith)**
**(in example, only showing name)**

**H(267-89-5431) = 31 %11 = 9**
**John Smith**
**H(703-25-6141) = 41%11 = 8**
**Ademola Olayinka**
**H(319-86-2115 )= 15 %11 = 4**
**Betty Harris**
**H(476-82-5120) = 20%11 = 9**
**Rose Black**

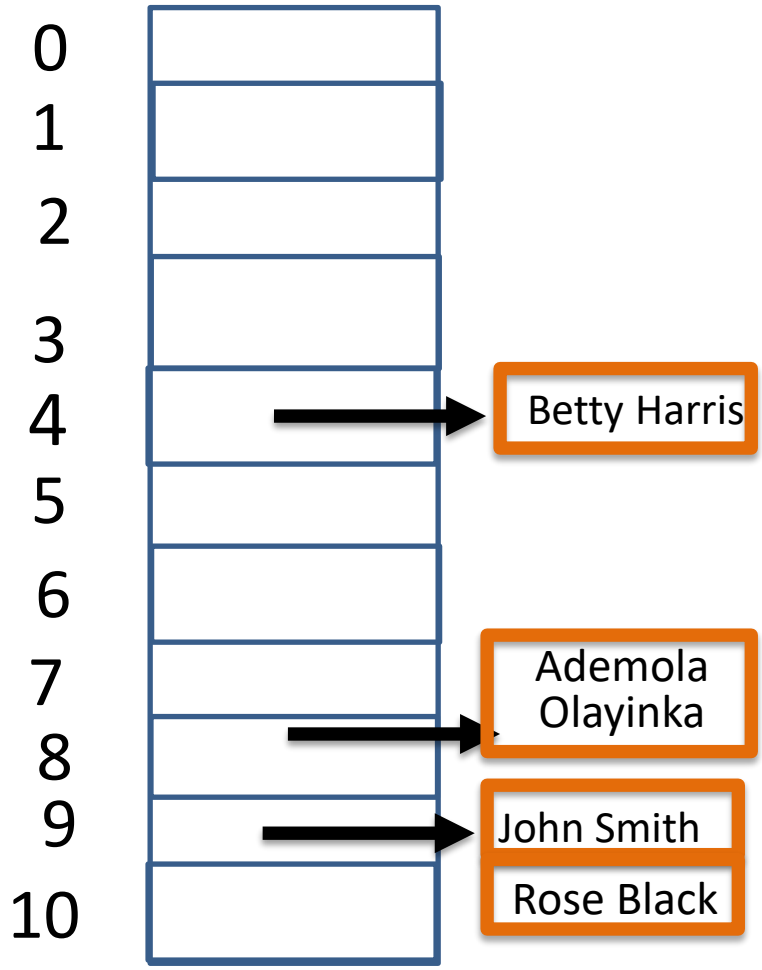


# Another way: Use a list of lists

- **Insert these into the list**
- **Insert as (key, value) tuple**

**(267-89-5431, John Smith)**
**(in example, only showing name)**

**H(267-89-5431) = 31 %11 = 9**
**John Smith**
**H(703-25-6141) = 41%11 = 8**
**Ademola Olayinka**
**H(319-86-2115 )= 15 %11 = 4**
**Betty Harris**
**H(476-82-5120) = 20%11 = 9**
**Rose Black**




Collisions added to list, 2 in list 9

# WOTO-3 How Dictionaries Work
# http://bit.ly/101s23-0425-1

# A peek into CompSci 201

- **Sorting  list of numbers**
- **Quicksort algorithm**
  - Uses recursion

# Quicksort - Idea

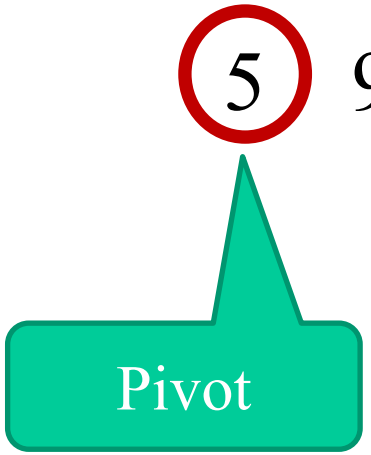
- Pivot – select and adjust the list
  - Select one of the elements
  - Put it where it belongs in sorted order
  - Put elements less than it, to its left
  - Put elements greater than it, to its right
- Recursively sort the elements to its left
- Recursively sort the elements to its right
- Done!

# Quicksort - Idea

- Pivot – select and adjust list
- Recursively sort the elements to its left
- Recursively sort the elements to its right

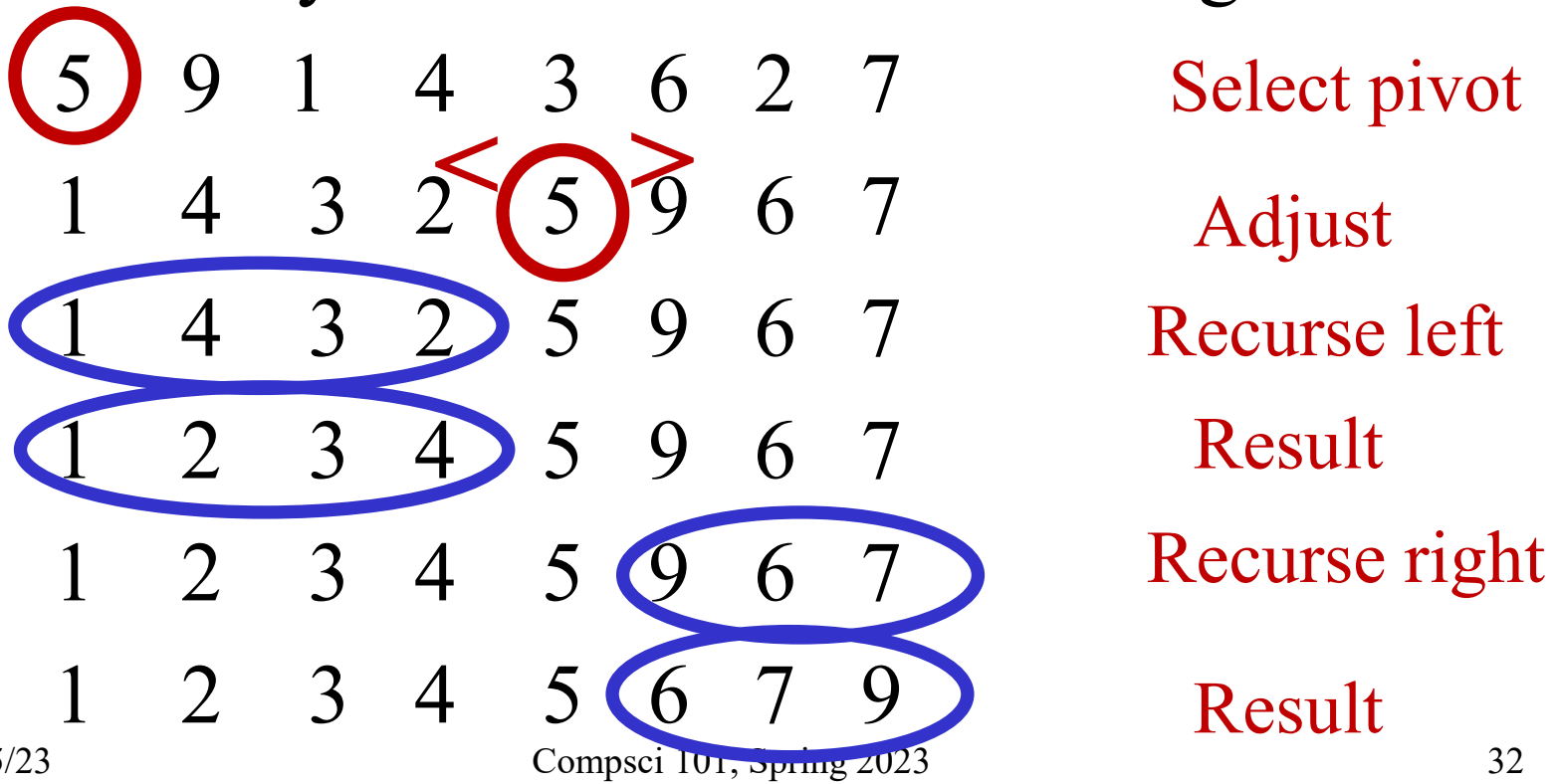
# Quicksort - Idea

- Pivot – select and adjust list
- Recursively sort the elements to its left
- Recursively sort the elements to its right



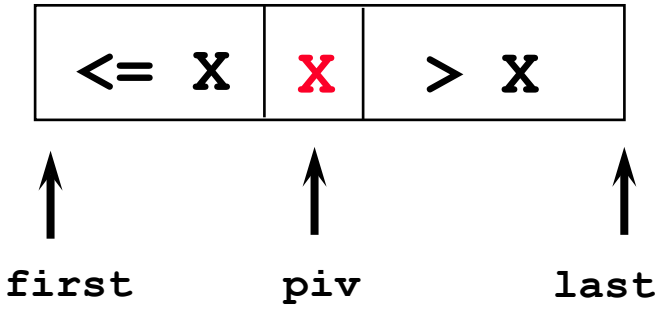
# Quicksort - Idea

- Pivot – select and adjust list
- Recursively sort the elements to its left
- Recursively sort the elements to its right



# Quicksort: fast in practice

```
def doQuick(list, first, last) {
    if (first >= last) return

    piv = pivot(list,first,last)
    doQuick(list,first,piv-1)
    doQuick(list,piv+1,last)
}
```



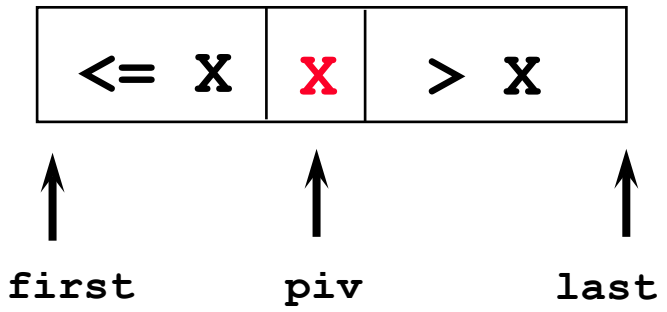
# Quicksort: fast in practice

```
def doQuick(list, first, last) {
    if (first >= last) return

    piv = pivot(list,first,last)
    doQuick(list,first,piv-1)
    doQuick(list,piv+1,last)
}
```

piv, first and last are indexes

Not showing pivot function

Elegant!

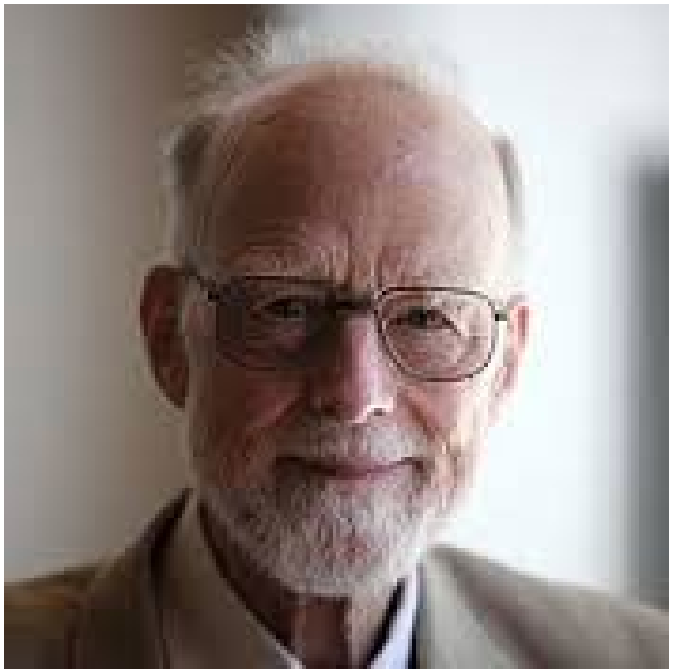


# Sir Anthony (Tony) Hoare

Invented Quicksort in 1962
- he didn't get recursion

Turing Award winner
- programming language design
- Algol 60

"There are two ways of constructing a software design. One way is to make it so simple that there are obviously no deficiencies. And the other way is to make it so complicated that there are no obvious deficiencies."



"Inside every large program is a small program struggling to get out."

# 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, …

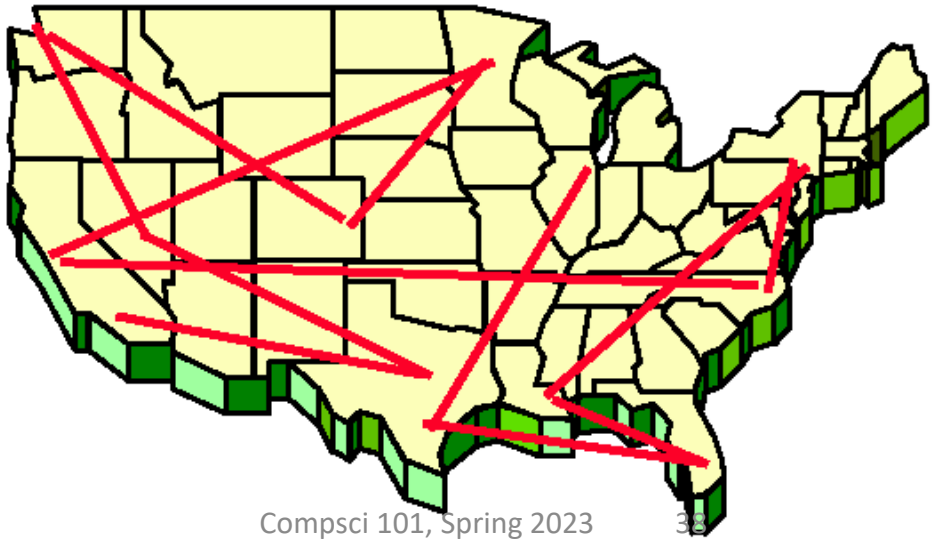
# Computing - solve all problems?

- **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
    - Add up the distance in this order
  - Dallas, Atlanta, Raleigh, Reno, Chicago
    - Add up the distance in this order
  - Etc.
- Would you agree to code this up?

# Traveling Band questions
# bit.ly/101s23-0425-2

# 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 long?

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

# How long?

| Number of Cities | All paths – N! | Time to solve - $10^9$ Instructions per second |
|---|---|---|
| 10 | 3 million | < sec |
| 15 | $10^{12}$ | 16 min |
| 18 | $10^{15}$ | |
| 20 | $10^{18}$ | |
| 25 | $10^{25}$ | |

# How long?

| Number of Cities | All paths – N! | Time to solve - $10^9$ Instructions per second |
|---|---|---|
| 10 | 3 million | < sec |
| 15 | $10^{12}$ | 16 min |
| 18 | $10^{15}$ | 11 days |
| 20 | $10^{18}$ | |
| 25 | $10^{25}$ | |

# How long?

| Number of Cities | All paths – N! | Time to solve - $10^9$ Instructions per second |
|---|---|---|
| 10 | 3 million | < sec |
| 15 | $10^{12}$ | 16 min |
| 18 | $10^{15}$ | 11 days |
| 20 | $10^{18}$ | 31 years |
| 25 | $10^{25}$ | |

# How long?

| Number of Cities | All paths – N! | Time to solve - $10^9$ Instructions per second |
|---|---|---|
| 10 | 3 million | < sec |
| 15 | $10^{12}$ | 16 min |
| 18 | $10^{15}$ | 11 days |
| 20 | $10^{18}$ | 31 years |
| 25 | $10^{25}$ | $10^8$ years |

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

# 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

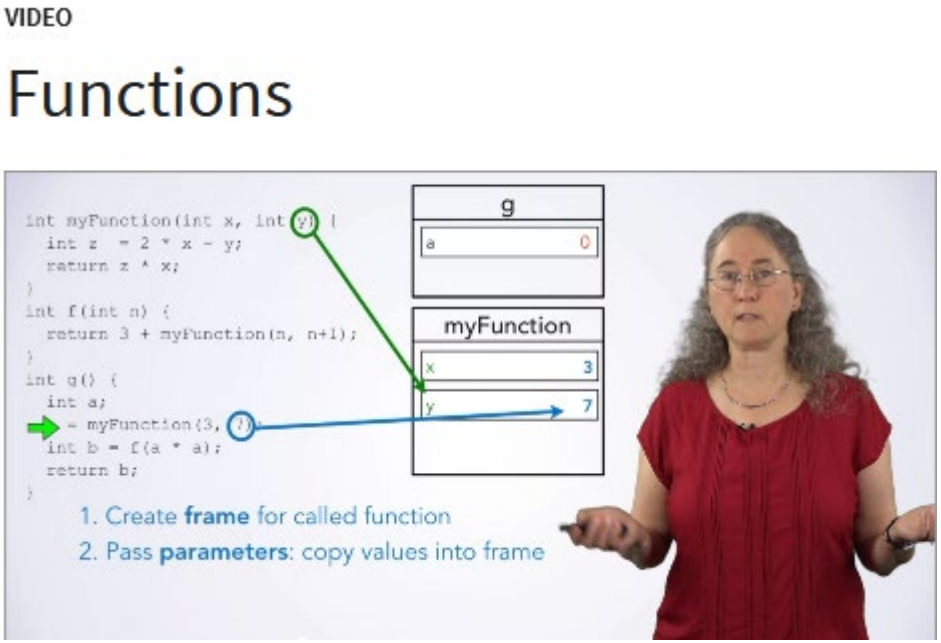
```
<?php

$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>";
}

?>
```

# What is next?

- **CompSci 201**
  - Java, efficiency, other ways to organize data
- **CompSci 230 – can take concurrently with 201**
  - Discrete Mathematics
    - Course substitutions if you take a lot of math/stats
- **CompSci 260 Computational Biology**
- **CompSci 216 Everything Data**
- **CompSci 240 Race, Gender, Class and Computing**

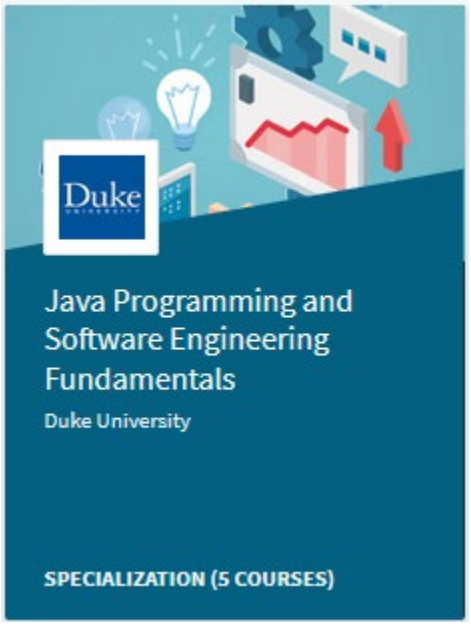
# What to do over Winter Break? Take a Duke Coursera course Free

- **Course on Java**



# Duke Coursera course on Java

- **Coursera for Duke**
  - https://online.duke.edu/coursera-for-duke/
- **Java Programming/Soft Eng Fund**
  - 5 courses
    - HTML/CSS/JavaScript
    - 4 courses on Java
- **Course is FREE for Duke students**
  - Go through link above



# End with A CS Story
# bit.ly/101s23-0425-3