# CS2020
# Data Structures and Algorithms

Welcome!

# Introductions

# Introductions

Seth Gilbert

- Asst. Professor in the CS Department
- Office: COM2-3-23
- Office hours: TBA (and by appointment)
- Research interests: distributed algorithms

# Introductions

Alan, Cheng Ho-lun

- Senior Lecturer in
  The CS Department
- Office: AS6-05-03
- Office hours: TBA
  (and by appointment)
- Ext. 68732
- Research interests: Graphics, Geometry, Games

Tutors

# Introductions

Tutors: Khor Shi-Jie

"I have tutored 2 other tutors before."

# Introductions

## Tutors: Davin Choo

Tips on doing well in CS2020:

1. NEVER skip classes.
2. You GONNA do your work.
3. You better GIVE us your best effort.
4. It's up to YOU to clarify your doubts.
5. Never, ever, ever give UP!

# Introductions

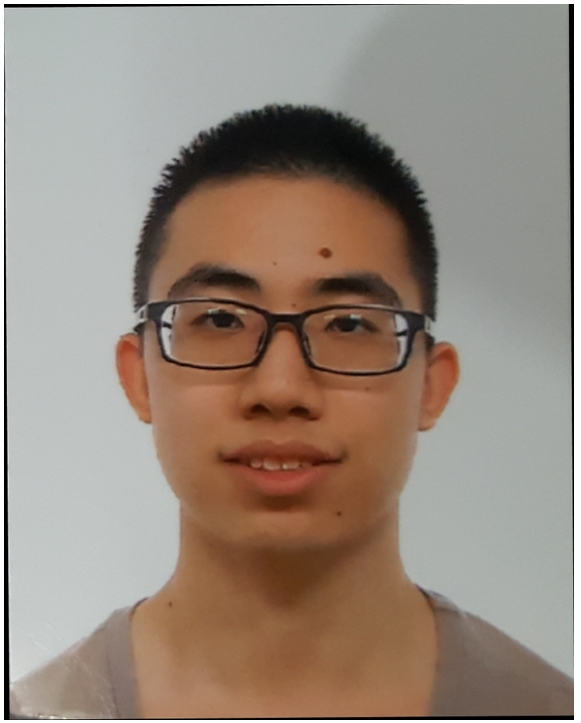
Tutors: Choyuk Chow

Why are trees in real life always upside down?

# Introductions

Tutors: Curtis Tan Wei Jie

Knowing that the sight of your tutor, means having cleared another week of CS2020, fills you with DETERMINATION.



# Introductions

## Tutors: Jonathan Gunawan

Welcome to CS3233. In this class we will learn …

Wait, this is not CS3233? Oh shoot, I applied for a wrong module.


Anyway, good luck and have fun in this module, whatever module this is.

# Introductions

Tutors: Kai Yuan Thng

"Life would be so much easier if we only had the source code."

# Introductions

Tutors: Leonard Hio

"FYI: I'm actually the one on the right."


2013

What now?

Algorithms

Object-oriented programming

Java

# Algorithms

What is an *algorithm*?

- Set of instructions for solving a problem

  "First, wash the tomatoes."

  "Second, peel and cut the carrots."

  "Third, mix the olive oil and vinegar."

  "Finally, combine everything in a bowl."

- *Finite* sequence of steps
- Unambiguous
- English, Chinese, pseudocode, Java, etc.

# Algorithms

History

- Named for al-Khwārizmī (780-850)
  - Persian mathematician
- Many ancient algorithms

# Algorithms

History

- Named for al-Khwārizmī (780-850)
  - Persian mathematician

- Many ancient algorithms
  - Multiplication: Rhind Papyrus
    - Babylon and Egypt: ~1800BC
  - Euclidean Algorithm: Elements
    - Greece: ~300BC
  - Sieve of Eratosthenes
    - Greece: ~200BC

"If you need your software to run twice as fast, hire better programmers.

But if you need your software to run more than twice as fast, use a better **algorithm.**"

-- *Software Lead at Microsoft*

# Software: desirable features

# Software

Desirable features?

- Speed / Performance
- Correctness / lack of bugs
- Memory usage
- Easy to maintain / easy to read
- Modular
- Completed on schedule
- Elegant
- Portable

# Algorithms

Goals of this course:

- How to organize and manipulate data?
  - Efficiency
    - Time: *How long does it take?*
    - Space: *How much memory? How much disk?*
    - Others: Energy, parallelism, etc.
  - Scalability
    - Inputs are *large* : e.g., the internet.
    - Bigger problems consume more resources.
- Solve real (fun!) problems

# Algorithms

**How to solve a problem**:

- Identify the problem
- Abstract: discard irrelevant details
- Find good algorithms
- Implement (in Java)
- Evaluate

  How fast? How does it scale?

# Semester Overview

- Topic 1: Linked data structures
  - Arrays
  - Searching
  - Sorting
  - Lists, Stacks, Queues
  - Divide-and-Conquer

- Example problems: document distance, peak finding

# Semester Overview

- Topic 2: Trees
  - Binary Search Trees
  - Balanced Trees
  - Priority Queues
  - Heaps

- Example problems: simple scheduling

# Semester Overview

- Topic 3: Hash Tables
  - Dictionaries
  - Hash functions
  - Chaining
  - Amortized Analysis

- Example problems: DNA similarity

# Semester Overview

- Topic 4: Graphs
  - Searching in a graph
  - Spanning trees
  - Shortest paths
  - Directed graphs

- Example problems: game playing, map searching

# Semester Overview

- Topic 5: Advanced Topics
  - Dynamic programming
  - Optimization
  - Numerical methods
  - Computational geometry
  - Concurrent algorithms

# Semester Overview

For each algorithm:

1. What problem does it solve?
2. Why does it work?
3. How do you implement it?
4. What is its asymptotic performance?
5. What is its real world performance?
6. What are the trade-offs?

# Java

# Language Does Not Matter

Algorithms are more important:

Fact: C can be 20x as fast as Python!

| Algorithm | Language | Time | 10,000 elements |
|---|---|---|---|
| **Fast** (MergeSort) | **Slow** (Python) | 2n log(n) $\mu$s | 0.266s |
| **Slow** (InsertionSort) | **Fast** (C) | $0.01n^2$ $\mu$s | 1s |

**(Source: MIT 6.006, 2008)**

# Computer languages

- Hardware: Assembly language
- Procedural (imperative) languages:
  - Fortran, COBOL, BASIC, Pascal , C
- Functional languages:
  - IPL, Lisp, Scheme, Haskell
- Declarative languages:
  - SQL, Lex/Yacc
- Scripting languages:
  - Javascript, Perl
- Object-oriented languages:
  - Simula 67, Smalltalk 80, C++, Java, C#

Modern languages?
Javascript, Python, Ruby

# Objected-oriented programming

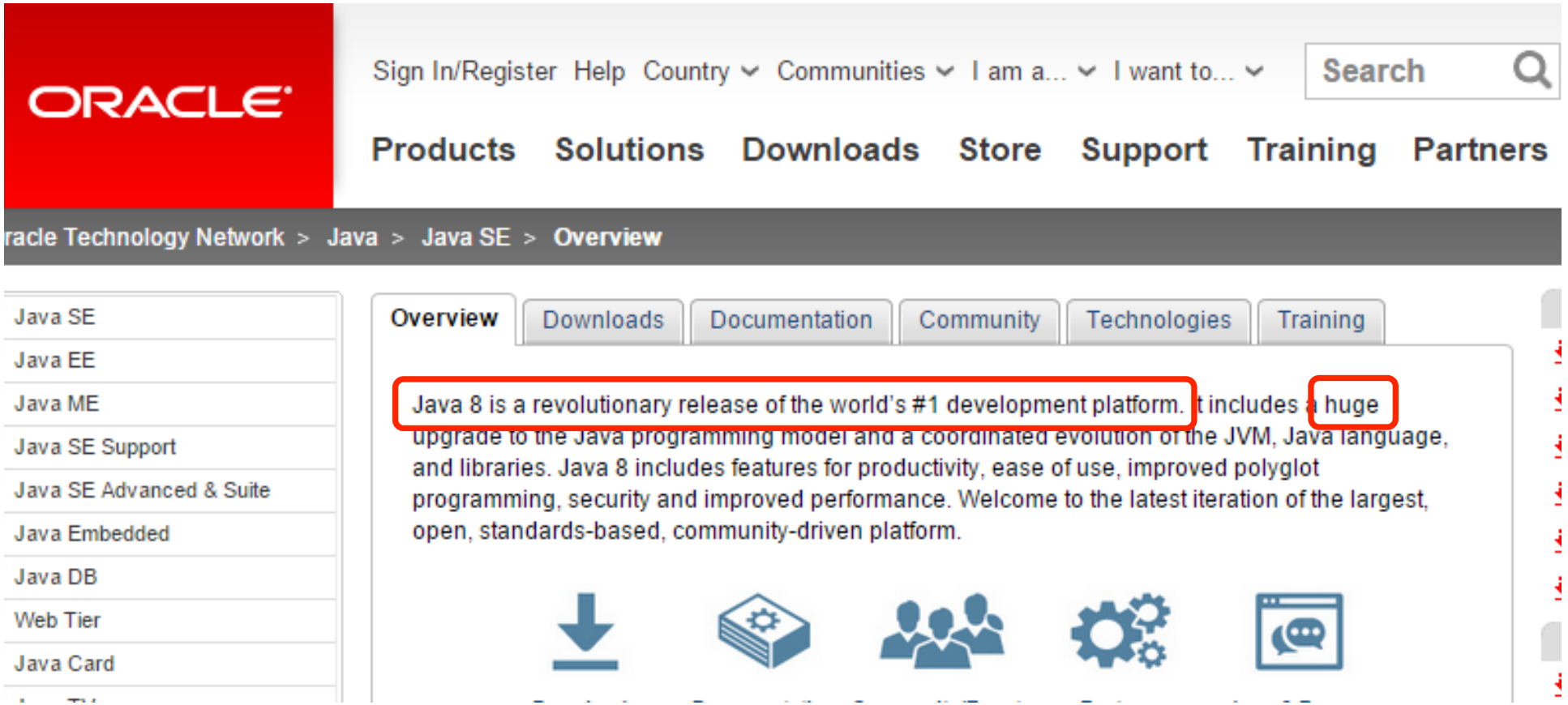
# Why Java?

Remember:
Language does not matter!

- Good aspects:
  - Common in industry / real-world / web
  - Object-oriented in a deep way
  - Modularity / abstraction via OOP
  - Avoids memory leak issues of C/C++

- Less good aspects:
  - Performance??  (compare to: C++)
  - Elegance??  (compare to: Scheme)

# Java 8!


ORACLE
Sign In/Register Help Country Communities I am a... I want to...
Search
Products Solutions Downloads Store Support Training Partners
racle Technology Network > Java > Java SE > Overview
Java SE
Java EE
Java ME
Java SE Support
Java SE Advanced & Suite
Java Embedded
Java DB
Web Tier
Java Card
Overview
Downloads
Documentation
Community
Technologies
Training
Java 8 is a revolutionary release of the world's #1 development platform.
includes a huge
upgrade to the Java programming model and a coordinated evolution of the JVM, Java language,
and libraries. Java 8 includes features for productivity, ease of use, improved polyglot
programming, security and improved performance. Welcome to the latest iteration of the largest,
open, standards-based, community-driven platform.

# Java 8

Features:

- Lambda expressions
- Default methods
- Repeating annotations
- Type annotations
- Method parameter reflection
- Stream collection class
- Etc.

# Java 8

For now:

- Do not use special Java 8 features.

Why?

- Focus on classic object-oriented programming.
- Focus on algorithms and data structures, not advanced language features.
- Maybe at the end of the semester…

# How to learn Java?

Knowledge

Experience

Talent

# Problem Sets

Practice, practice, practice…

# Problem Sets

If the tutor does not understand your solution, then it will not be graded.

Style and comments matter.

The tutor may ask you to explain your code/ algorithm better.

The tutor is not a compiler.

# Problem Sets

Late submissions:

- 24 hours: 20% penalty
- 2 weeks: 40% penalty
- Last day of class: 60% penalty

Hand in problem sets on time!

Even if late, do them anyways!

# Coursemology

coursemology

Gamified Online Education Platform:

Making your class a world of games in a universe of fun.


Name

Email

Password

Password confirmation

Sign up

### Engaging

Coursemology allows educators to add gamification elements, such as experience points, levels, achievements, to their classroom exercises and assignments.

The gamification elements of Coursemology motivate students to do assignments and trainings.

### General

It's built for all subjects. The gamification system of Coursemology doesn't make assumption on the course's subject.

Through Coursemology, any teacher who teaches any subject can turn his course excercies into a online game.

### Simple

It's built for all teachers. You don't need to have any programming knowledge to master the platform.

Coursemology is easy and intuitive to use for both teachers and students.

# Coursemology

Problem sets:

- Access problem sets
- Submit solutions
  - Upload solution as .java files.
  - Ignore code box.
- Interact with tutors
- Get grades

How to join Coursemology?

- You will receive an invitation e-mail.

# (Computer) Chess



## Rybka (Computer) – Shredder (Computer) 1–0

**C67** WCCC Pamplona Openclass (6) 13.05.2009

1.e4 e5 2.♘f3 ♘c6 3.♗b5 ♘f6 4.0-0 ♘xe4 5.♕e2 ♘g5 6.♘xg5 ♕xg5 7.d4 ♕e7 8.dxe5 ♘d4 9.♕d3 ♕xe5 10.♘c3 ♗c5 11.♕d1 ♘e6 12.♖e1 ♕d4 13.♕f3 0-0 14.♖e4 ♕d6 15.♖h4 ♕e5 16.♗d2 f5 17.♖e1 ♕f6 18.♕h3 ♕g6 19.♘d5 c6 20.♖xe6 ♕xe6 21.♘f4 ♕xa2 22.♖xh7 cxb5 23.g3 ♖f6 24.♗c3 ♔f7 25.♕h4 ♕a1+ 26.♔g2 ♕a6 27.♗xf6 ♕xf6 28.♕h5+

**1–0**

Download PGN

## Computer chess reels from 'biggest sporting scandal since Ben Johnson'

# Czech mate, Mr Cheat

IT'S a story that has sent pawns and rooks spinning off chess boards across the world.

Rybka, the best chess-playing computer on the planet, is a cheat.

And its developer, Vasik Rajlich – one half of a couple dubbed the Posh and Becks of the game – has been shunned as a plagiarist, banned from competing, stripped of his titles and ordered to hand back his trophies and prize money.

Rajlich, himself an international master of the game, was found guilty by his peers of basically copying earlier chess programs when creating Rybka.

A 34-member panel found the 40-year-old Czech from Ohio, but now living in Hungary, plagiarised two other programs, Crafty and Fruit. Their report states: 'Not a single panel member believed him innocent. Vasik Rajlich's claims of complete originality are contrary to the facts.'

Not since IBM's Deep Blue computer defeated grand master Garry Kasparov in 1997 – and was subsequently accused of cheating – has the world of computer chess been in such uproar. Rybka won the International Computer Games Association world championship from 2008 to 2010.

By **Tariq Tahir**

Peter Doggers, from online site Chess Vibes, said: 'The impact in the computer chess world must be comparable to arguably the most famous example of doping in athletics – the positive drug testing of Canadian sprinter Ben Johnson.'

For his part, Rajlich has not commented, save an email sent to the association in which he disputed Rybka included code written by others.

He never made grand master as a player so turned to programming. 'I figured there were about 2,000 people in the world stronger than me in chess,' he once said, 'but not one chess player that was stronger than me in programming.'

By 2005, Rybka – Czech for 'little fish' – was ready. The chief tester is his wife, Iveta, herself an international master. David Levy, president of the ICGA, said: 'We are convinced the evidence against Rajlich is both overwhelming in its volume and beyond reasonable question in its nature.'

Posh and checks: Vasik Rajlich and his chief tester, wife Iveta

# Problem Sets

## Collaboration Policy

- Working together is strongly encouraged!
- You must write/code your problems sets alone.
- You must list on your submission the name of everyone you worked with, and all sources used.
- Cheating / plagiarism will be dealt with harshly.

# Administrative Details

Weekly schedule:

- Two lectures: Wed/Fri 10am-12pm
- One problem session (1hr): Friday
- One discussion group (2hr): Tues/Thurs

Lecture slides:

- Posted after lecture (see later)
- Screencast (sometimes)

# Discussion Groups

# Administrative Details

Seven Discussion Groups (tutorials):

- Tuesday:
  - 10-12pm
  - 12-2pm
  - 2-4pm
  - 4-6pm
- Thursday
  - 10-12pm
  - 4-6pm

Register on CORS…

DGs start Week 3.

# Administrative Details

Problem Sessions (recitations):

- Register via CORS
- Three slots:
  - 1pm
  - 2pm
  - 3pm

- Start this week!
- Choose any of the 3 slots, temporarily!

# Administrative Details

## Problem Sessions (recitations):

### Typical distribution:

- 1pm : 35 students
- 2pm : 25 students
- 3pm : 15 students

### Idea: choose a later timeslot!

# Where to ask questions?

# Discussion Forums

Facebook:

- Announcements
- Interesting related information
- Short questions

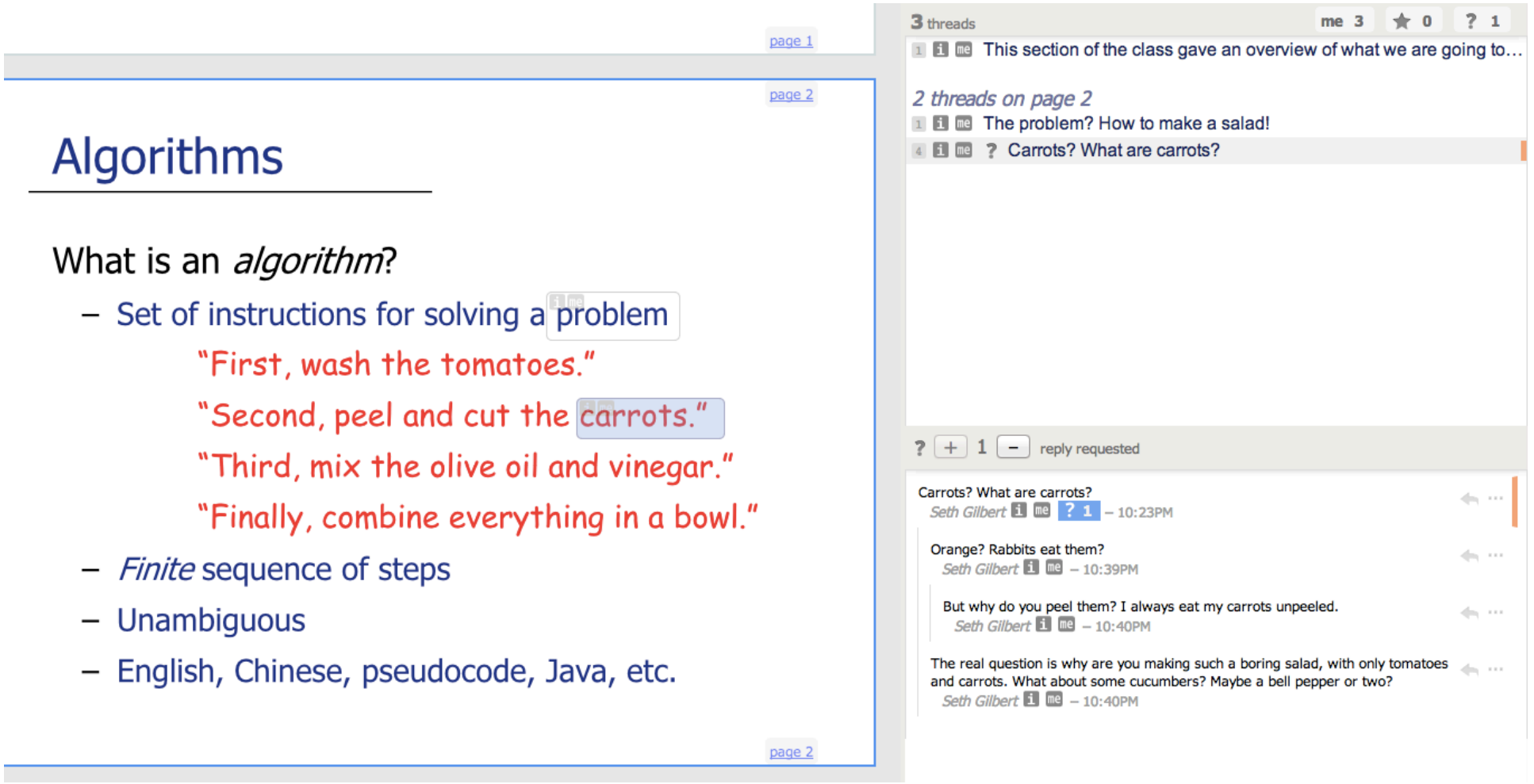
https://www.facebook.com/groups/cs2020.2016/

# Discussion Forums

Nota Bena (NB):

- Annotated lecture notes
- Ongoing conversation
  - Ask questions about lecture
  - Ask questions about the problem set
  - Ask questions about code
- I will e-mail you link to join

http://nb.mit.edu/


page 1
page 2
Algorithms
What is an *algorithm*?
– Set of instructions for solving a problem
"First, wash the tomatoes."
"Second, peel and cut the carrots."
"Third, mix the olive oil and vinegar."
"Finally, combine everything in a bowl."
– *Finite* sequence of steps
– Unambiguous
– English, Chinese, pseudocode, Java, etc.
page 2
3 threads
me 3
0
? 1
This section of the class gave an overview of what we are going to…
2 threads on page 2
The problem? How to make a salad!
? Carrots? What are carrots?
? + 1 – reply requested
Carrots? What are carrots?
Seth Gilbert ? 1 – 10:23PM
Orange? Rabbits eat them?
Seth Gilbert – 10:39PM
But why do you peel them? I always eat my carrots unpeeled.
Seth Gilbert – 10:40PM
The real question is why are you making such a boring salad, with only tomatoes and carrots. What about some cucumbers? Maybe a bell pepper or two?
Seth Gilbert – 10:40PM

Assignment 1:

Today, go and add one comment on Nota Bene to today's lecture. Either:

1. Ask a question about something on some other slide.

2. Annotate **this slide** by adding your favorite joke. Here.

# Administrative Details

Announcements:

- E-mail: your official NUS e-mail address
- Facebook group (cs2020 2016)
- Coursemology announcements

Please check at least one of these regularly!

# Administrative Details

Quizzes:

- Quiz 1 – Feb. 12
- Coding Quiz – Mar. 8/10
- Quiz 2 – Frid. Apr. 1

(Final) Exam:

- April 25 (evening)
- Exam will be returned (via e-mail) by the end of semester

# Other grading components

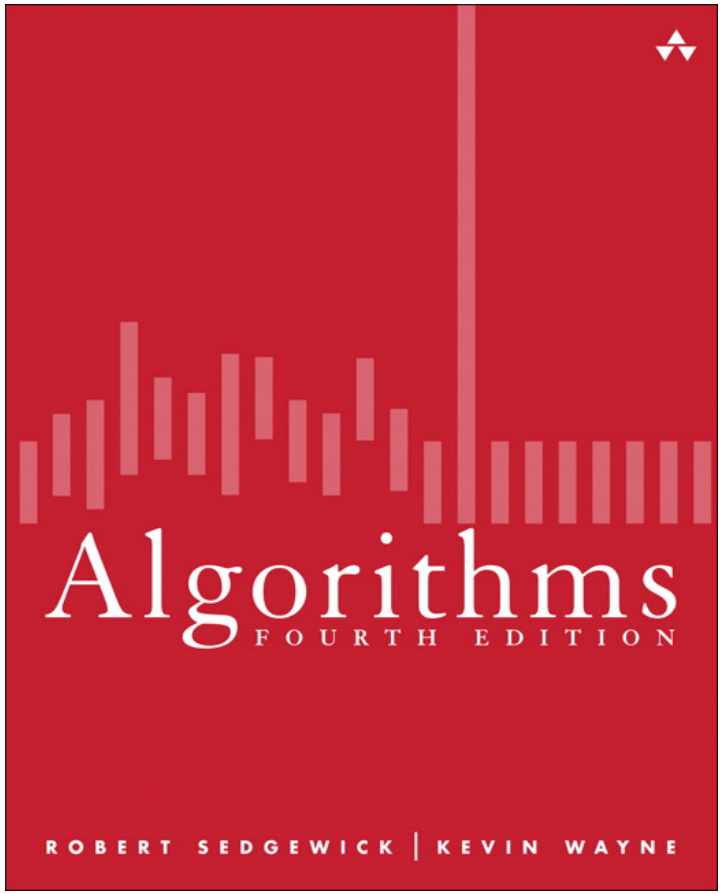
Participation: 10%

- Discussion groups
- Problem sessions
- Forums / Nota Bene / Facebook

# Administrative Details

Textbook: Algorithms

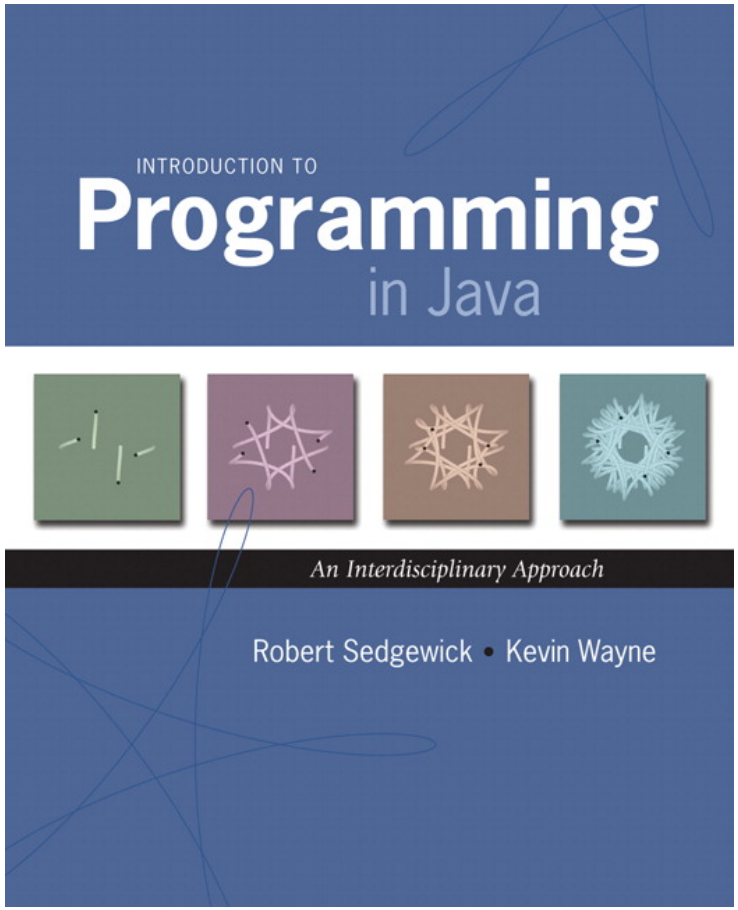
- Robert Sedgewick and Kevin Wayne



# Administrative Details

Textbook: Introduction to Programming in Java

- Robert Sedgewick and Kevin Wayne

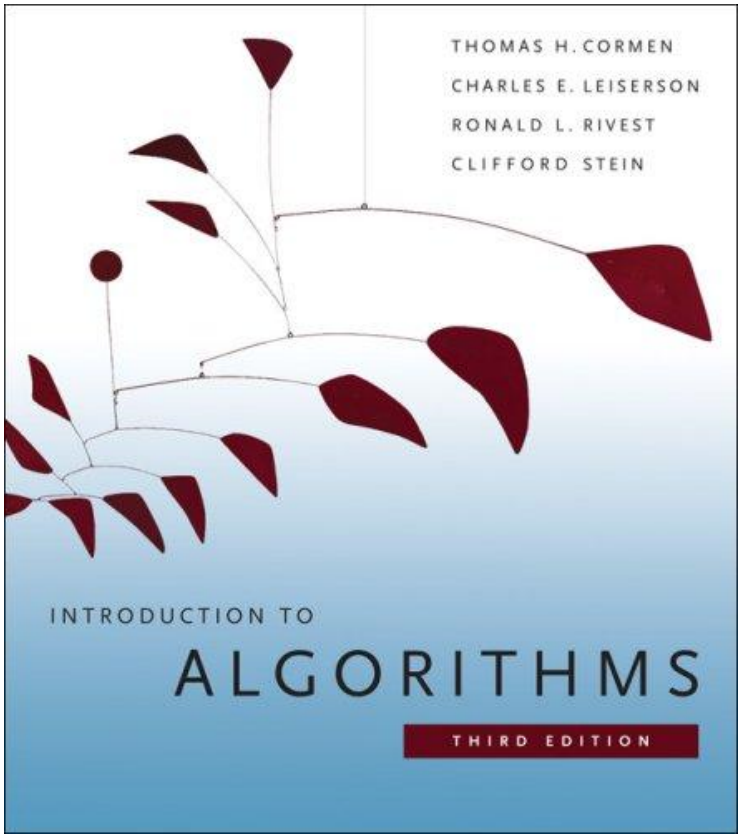



- Optional

# Administrative Details

Textbook: Introduction to Algorithms

- Cormen, Leiserson, Rivest, Stein




- Optional (but recommended)

# Should I take CS2020?

# Administrative Details

Deadlines:

- Add module: end of Week 1 (Jan. 15)
- Drop module: end of Week 2 (Jan. 22)
- Drop module with “W”: end of recess week

Note: cannot add CS1020 after Jan. 15!

# Administrative Details

If you are not currently registered for CS2020:

- Sign up using this form:
  http://goo.gl/forms/l818MBSp11
  (link is on Facebook page)

- Come talk to me after class today.

# Clickers...

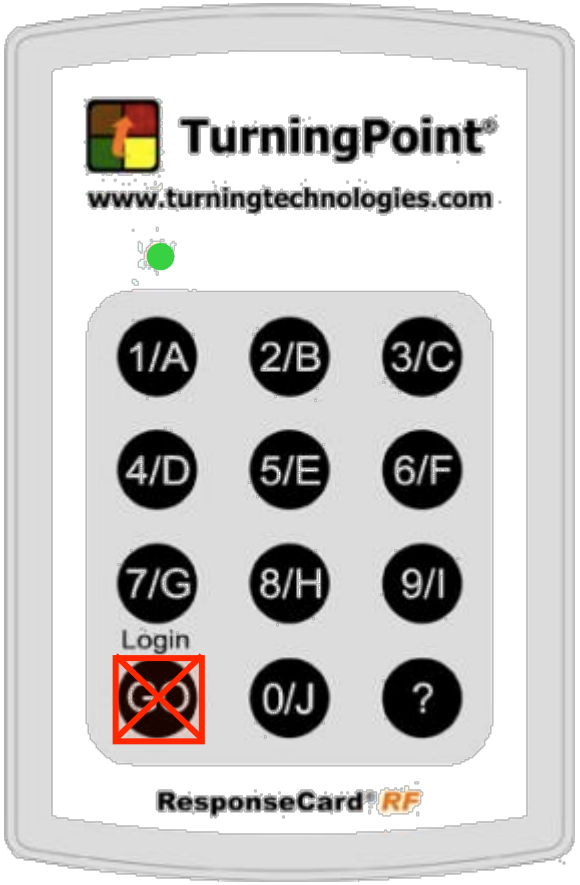
Who is your favorite author?

a) Shakespeare

b) J.K. Rowling

c) Confucius

d) Homer Simpson



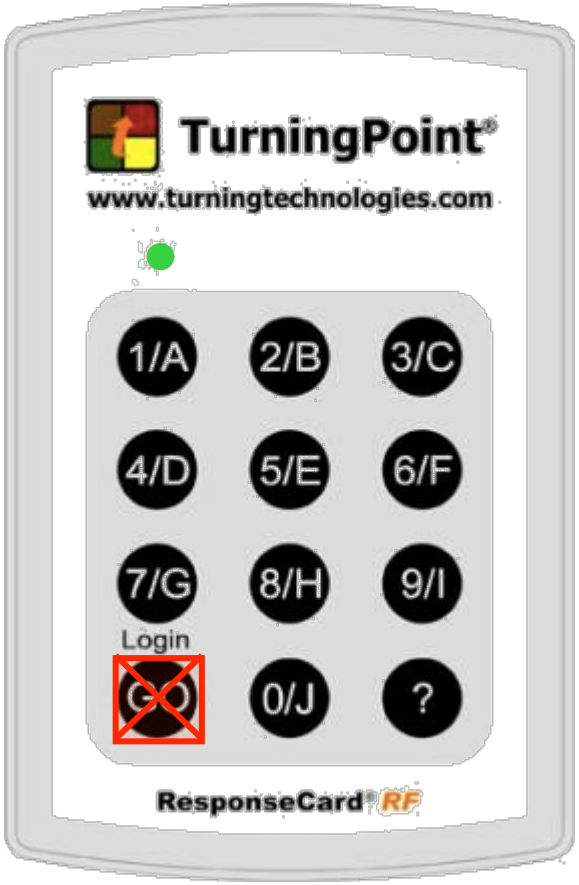
# Clickers...

Today:

Sign up for a clicker.

If your clicker goes missing:

SGD 105 / clicker.



# Break time

Questions?

Sign up for a clicker…

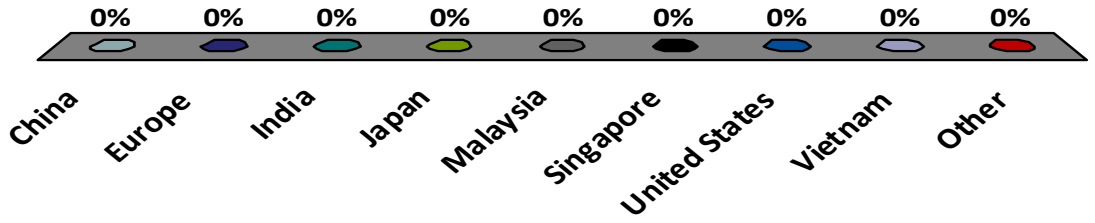
# Today

- Problem: Document Distance
  - How similar are two documents?

- How do we answer this question?
  - Algorithm idea
  - Java implementation
  - Performance measurement

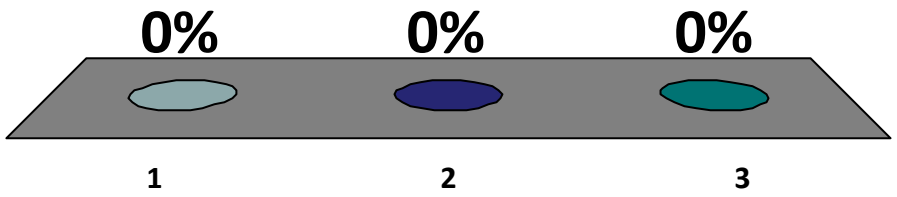
# Where are you from?

1. China
2. Europe
3. India
4. Japan
5. Malaysia
6. Singapore
7. United States
8. Vietnam
9. Other



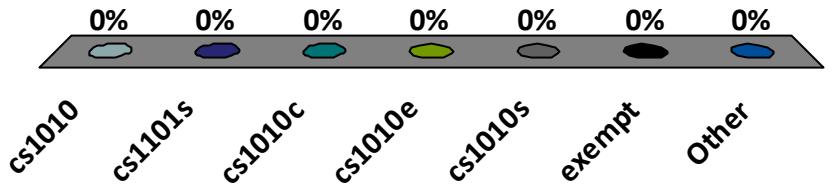
Are you:

1. Male
2. Female
3. Unspecified



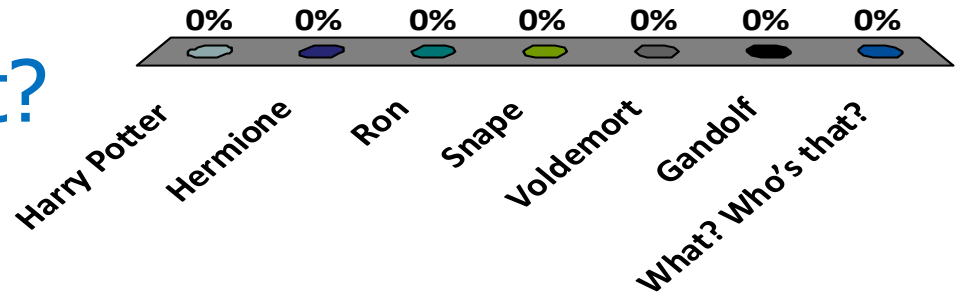
# Which class did you take last semester?

A. cs1010
B. cs1101s
C. cs1010c
D. cs1010e
E. cs1010s
F. exempt
G. Other

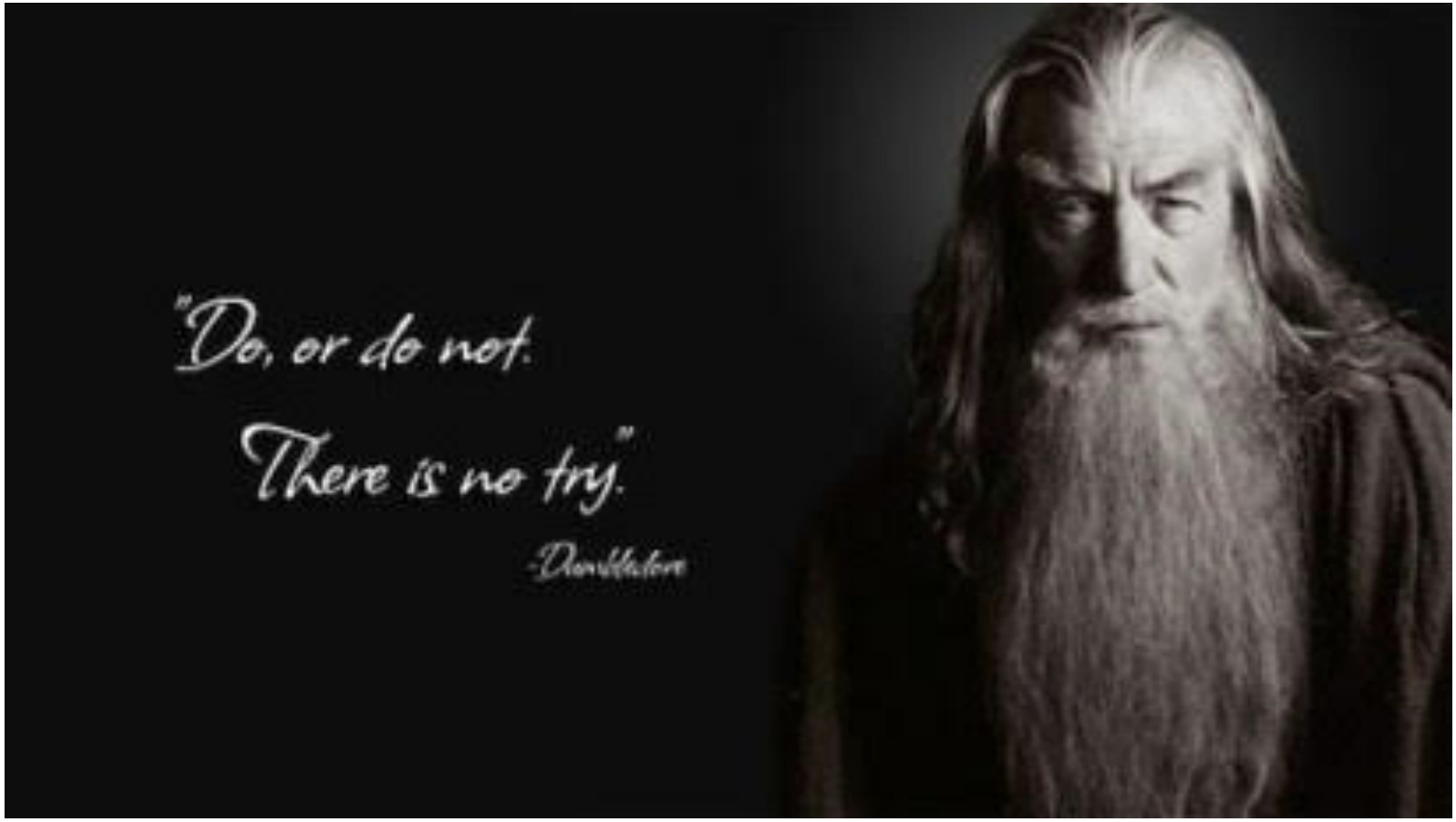


# Who is your favorite Harry Potter character?

1. Harry Potter
2. Hermione
3. Ron
4. Snape
5. Voldemort
6. Gandolf
7. What? Who's that?



**Who is your favorite Harry Potter character?**

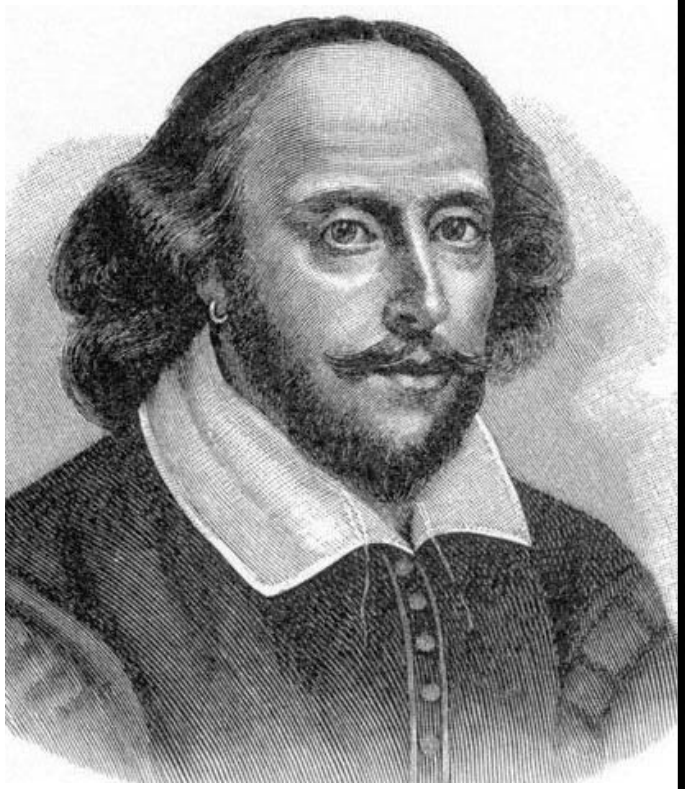

"Do, or do not.
There is no try."
-Dumbledore

# Today

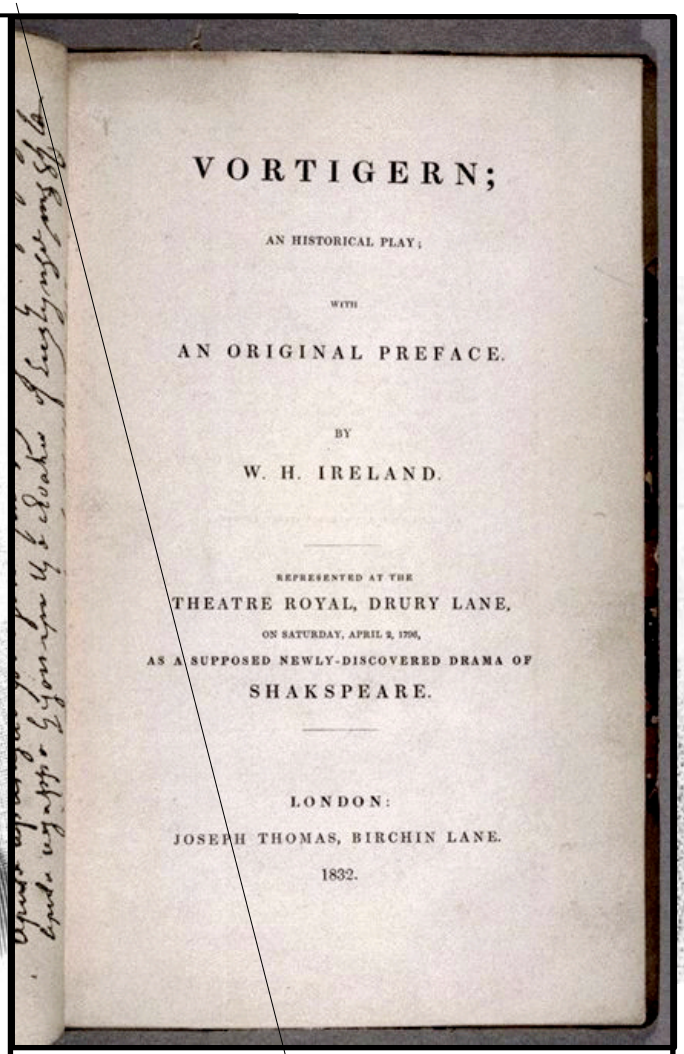
- Problem: Document Distance
  - How similar are two documents?

- Solution:
  - Algorithm idea
  - Java implementation
  - Performance measurement

# Who wrote this?



**William Shakespeare??**



VORTIGERN;

AN HISTORICAL PLAY;

WITH

AN ORIGINAL PREFACE.

BY

W. H. IRELAND.

REPRESENTED AT THE

THEATRE ROYAL, DRURY LANE,

ON SATURDAY, APRIL 2, 1796,

AS A SUPPOSED NEWLY-DISCOVERED DRAMA OF

SHAKSPEARE.

LONDON:

JOSEPH THOMAS, BIRCHIN LANE.

1832.

***mystery* play "found" in 1796**

**William Henry Ireland??**

# Document distance

- How similar are two documents?
  - Are two documents written by the same author?
  - Detect forgeries
  - Find plagiarism / cheating
  - Was Homer one author or many?

- What does “similar” mean?

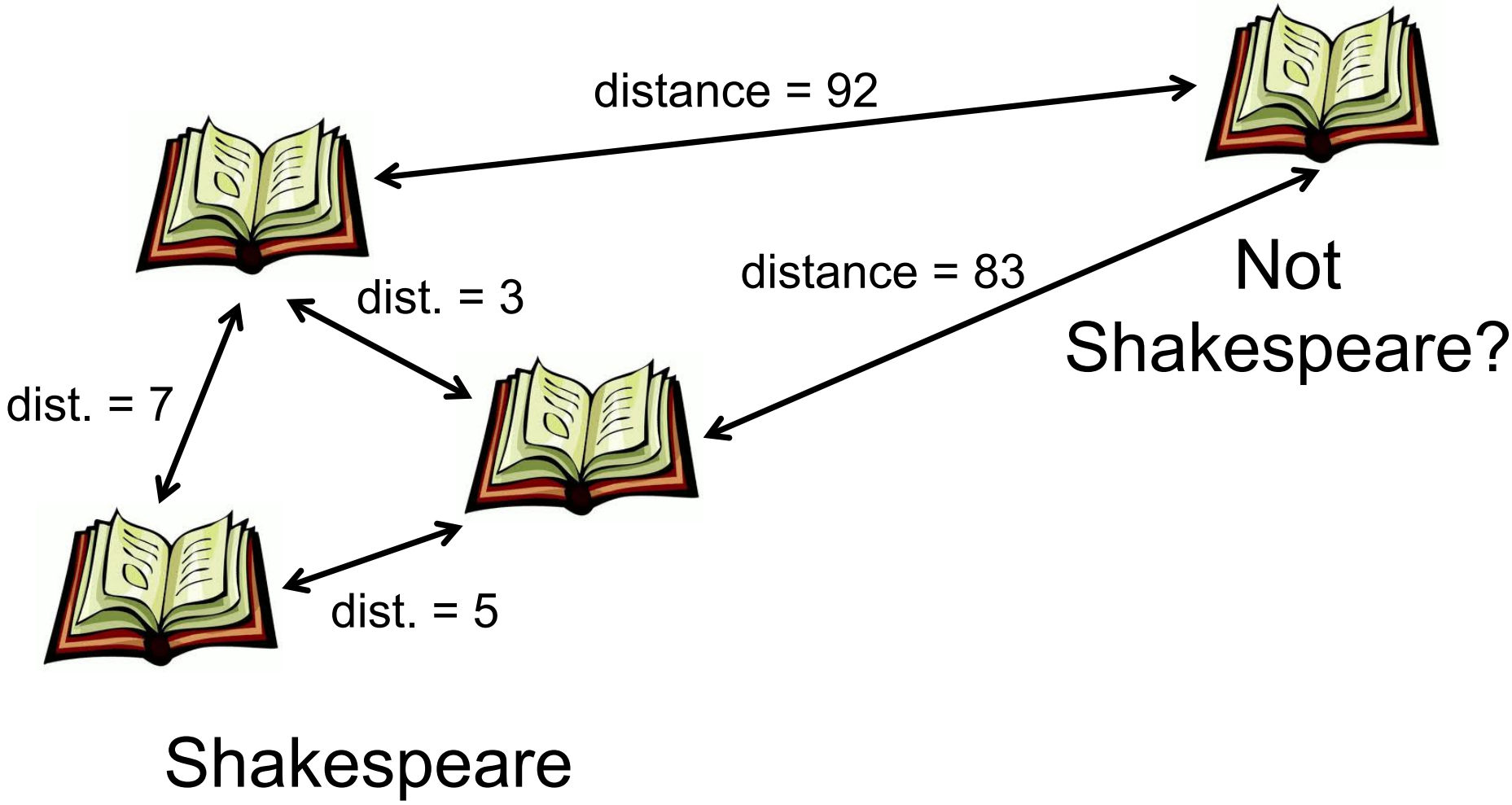
# Metrics of similarity

- Binary: (e.g., detect plagiarism)
  - Exactly same words in same order

- Scalar:
  - Number of words in the same order
  - Number of shared *uncommon* words
  - Same # of words per sentence
  - Same ratio of adjectives / nouns
  - Written on similar paper / using similar ink

# Document distance

How similar are two documents?

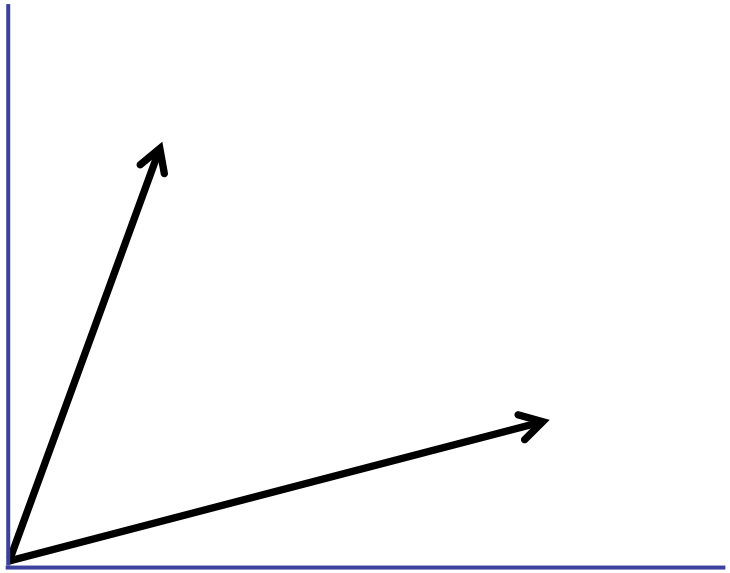


# Vector Space Model

Strategy:

- View each document as a high-dimensional vector.

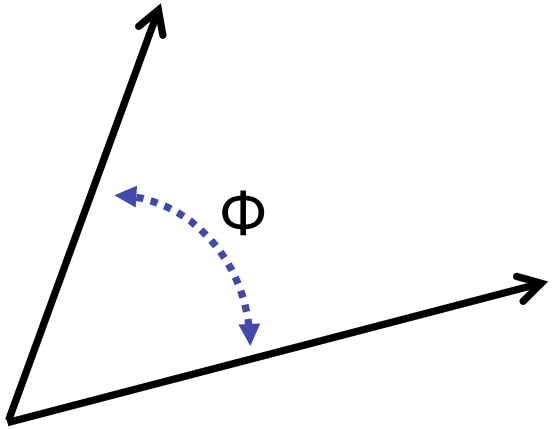
[Salton, Wang, Yang '75]



# Vector Space Model

Strategy:

- View each document as a high-dimensional vector.
- The *metric of similarity* is the angle between the two vectors.




- Identical: $\Phi = 0$
- No words in common: $\Phi = \pi/2$

[Salton, Wang, Yang '75]

# Vector Space Model

Document as vector:

Example 1:

"to be or not to be" = [2,1,1,2]

| be | not | or | to |
|---|---|---|---|
| 2 | 1 | 1 | 2 |

# Vector Space Model

Example 1:

"to be or not to be" = [0,2,0,1,0,1,2]

Example 2:

"be not afraid of greatness" = [1,1,1,1,1,0,0]

| afraid | be | greatness | not | of | or | to |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 0 | 0 |

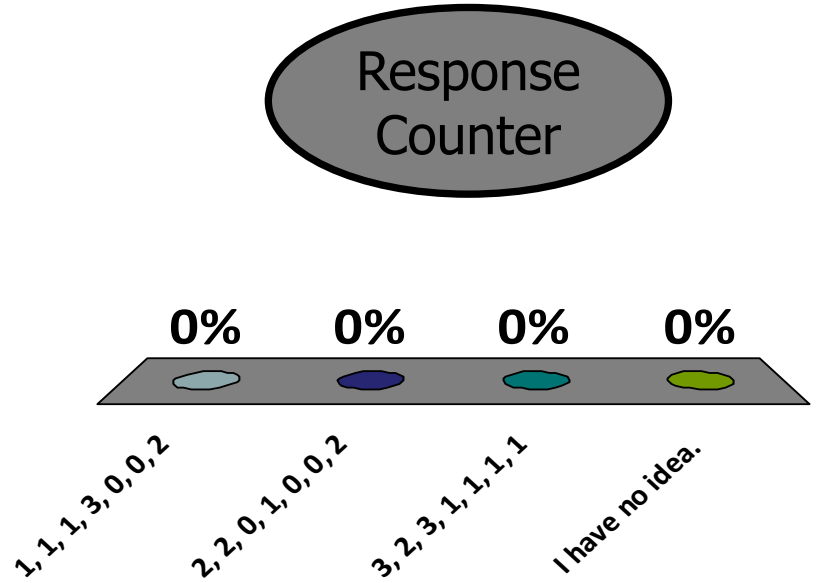
# Example 3: “to be afraid, to be not afraid”

a. 1, 1, 1, 3, 0, 0, 2

✓ b. 2, 2, 0, 1, 0, 0, 2

c. 3, 2, 3, 1, 1, 1, 1

d. I have no idea.




| afraid | be | greatness | not | of | or | to |
|---|---|---|---|---|---|---|
| ? | ? | ? | ? | ? | ? | ? |

# Vector Space Model

Dot Product:

$$v = [v_1, v_2, v_3, v_4]$$
$$w = [w_1, w_2, w_3, w_4]$$

$$v \cdot w = v_1w_1 + v_2w_2 + v_3w_3 + v_4w_4$$

# Vector Space Model

Dot Product:

$$v = [v_1, v_2, \ldots, v_n]$$
$$w = [w_1, w_2, \ldots, w_n]$$

$$v \cdot w = \sum v_i w_i$$

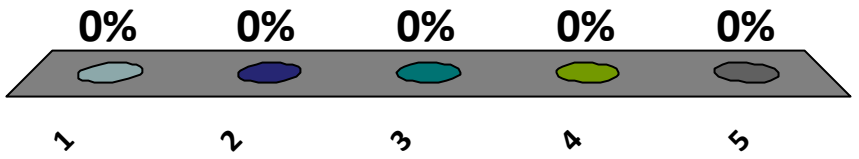
# Dot product question:

v = [0, 2, 0, 1]
w = [1, 1, 1, 1]

(v·w) =

a. 1
b. 2
✓ c. 3
d. 4
e. 5

Response Counter



# Vector Space Model

Norm of a vector (L2 norm):

$$|v| = \text{SQRT}(v \bullet v)$$

Example: distance between two points

$$\left|\left(x_1, y_1\right) - \left(x_2, y_2\right)\right| = \sqrt{\left(x_2 - x_1\right)^2 + \left(y_2 - y_1\right)^2}$$

# Vector Space Model

Norm of a vector (L2 norm):

$$|v| = \sqrt{v \cdot v}$$

$$|v| = \sqrt{\sum_{i=1}^{n} v_i \cdot v_i}$$

Example: NORM(3, 0, 4, 0) =

SQRT(3*3 + 0*0 + 4*4 + 0*0) = 5

# Vector Space Model

Law of cosines:

$$\Theta(v, w) = \cos^{-1}\left(\frac{v \cdot w}{\|v\| \cdot \|w\|}\right)$$

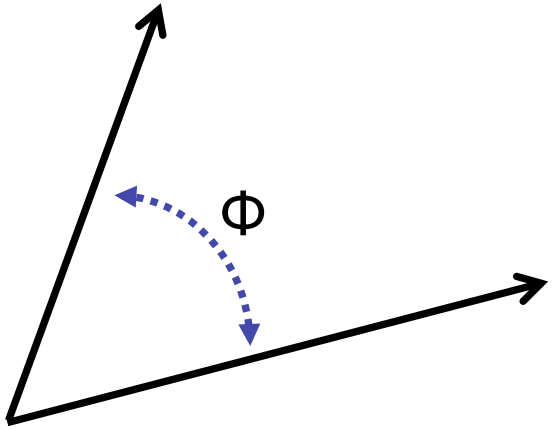
Notes:

- $\Phi$ is an angle between (0, pi)
- If (v=w), then $\Phi$=0.
- If (v • w) = 0, then $\Phi$=pi.

# Vector Space Model

Strategy:

- View each document as a high-dimensional vector.
- The *metric of similarity* is the angle between the two vectors.




- Identical: $\Phi = 0$
- No words in common: $\Phi = \pi/2$

[Salton, Wang, Yang '75]

# Compare Two Documents

Given: documents A and B

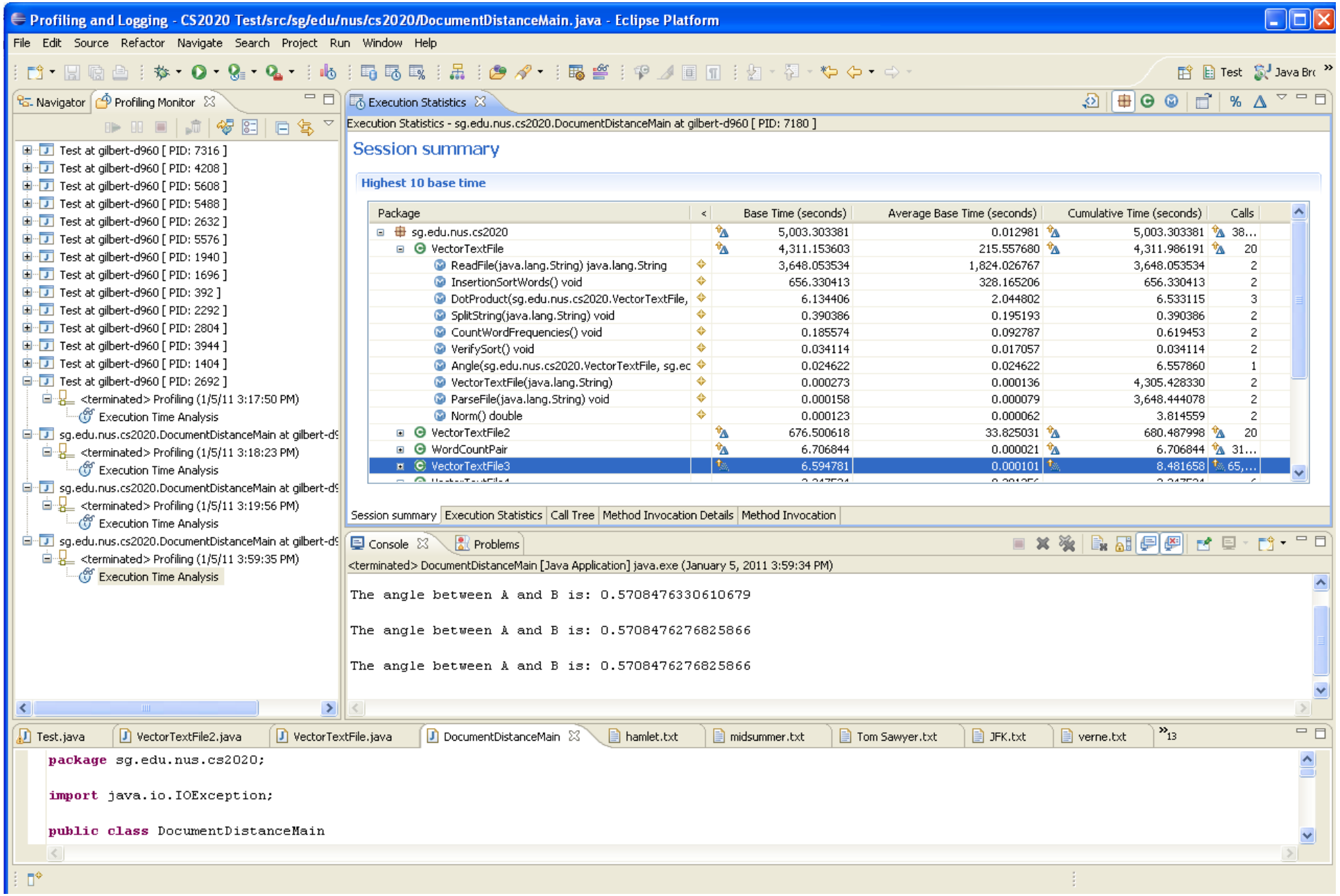
1. Create vectors $v_A$ and $v_B$
2. Calculate norm: $|v_A|$
3. Calculate norm: $|v_B|$
4. Calculate dot product: $(v_A \cdot v_B)$
5. Calculate angle $\Phi(v_A, v_B)$

# Performance Profiling

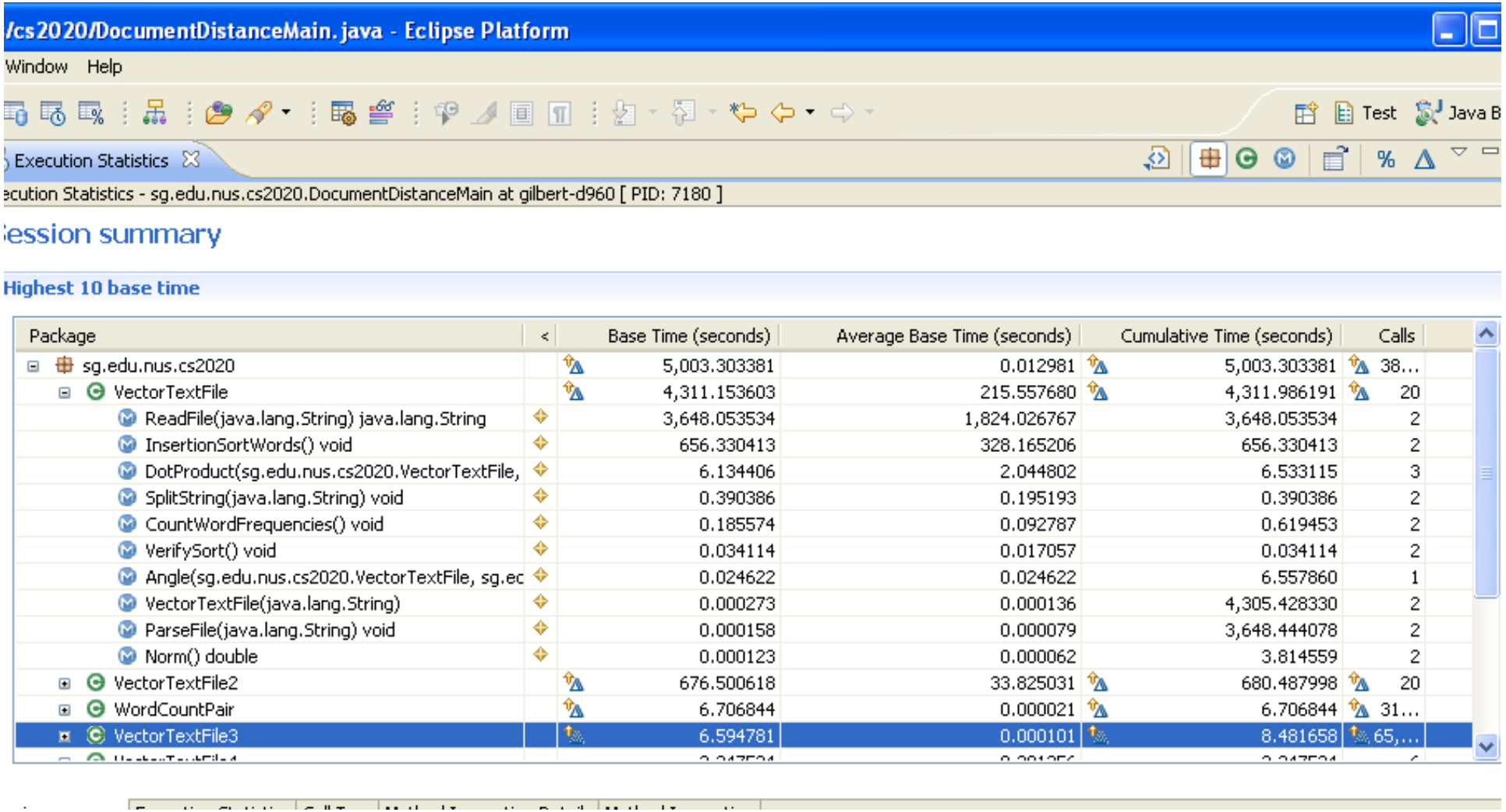
(*Dracula* vs. *Lewis & Clark*)

| Step | Function | Running Time |
|---|---|---|
| Create vectors: | Read each file | 1,824.00s |
| | Parse each file | 0.20s |
| | Sort words in each file | 328.00s |
| | Count word frequencies | 0.31s |
| Dot product: | | 6.12s |
| Norm: | | 3.81s |
| Angle: | | 6.56s |
| **Total:** | | **72minutes ≈ 4,311.00s** |

# Eclipse-TPTP



# Eclipse-TPTP




| Package | < | Base Time (seconds) | Average Base Time (seconds) | Cumulative Time (seconds) | Calls |
|---|---|---|---|---|---|
| sg.edu.nus.cs2020 | | 5,003.303381 | 0.012981 | 5,003.303381 | 38... |
| VectorTextFile | | 4,311.153603 | 215.557680 | 4,311.986191 | 20 |
| ReadFile(java.lang.String) java.lang.String | | 3,648.053534 | 1,824.026767 | 3,648.053534 | 2 |
| InsertionSortWords() void | | 656.330413 | 328.165206 | 656.330413 | 2 |
| DotProduct(sg.edu.nus.cs2020.VectorTextFile, | | 6.134406 | 2.044802 | 6.533115 | 3 |
| SplitString(java.lang.String) void | | 0.390386 | 0.195193 | 0.390386 | 2 |
| CountWordFrequencies() void | | 0.185574 | 0.092787 | 0.619453 | 2 |
| VerifySort() void | | 0.034114 | 0.017057 | 0.034114 | 2 |
| Angle(sg.edu.nus.cs2020.VectorTextFile, sg.ec | | 0.024622 | 0.024622 | 6.557860 | 1 |
| VectorTextFile(java.lang.String) | | 0.000273 | 0.000136 | 4,305.428330 | 2 |
| ParseFile(java.lang.String) void | | 0.000158 | 0.000079 | 3,648.444078 | 2 |
| Norm() double | | 0.000123 | 0.000062 | 3.814559 | 2 |
| VectorTextFile2 | | 676.500618 | 33.825031 | 680.487998 | 20 |
| WordCountPair | | 6.706844 | 0.000021 | 6.706844 | 31... |
| VectorTextFile3 | | 6.594781 | 0.000101 | 8.481658 | 65,... |

# Performance Profiling

(*Dracula* vs. *Lewis & Clark*)

| Step | Function | Running Time |
|---|---|---|
| Create vectors: | Read each file | 1,824.00s |
| | Parse each file | 0.20s |
| | Sort words in each file | 328.00s |
| | Count word frequencies | 0.31s |
| Dot product: | | 6.12s |
| Norm: | | 3.81s |
| Angle: | | 6.56s |
| **Total:** | | **72minutes ≈ 4,311.00s** |

# Performance Profiling

(*Dracula* vs. *Lewis & Clark*)

| Version | Change | Running Time |
|---|---|---|
| Version 1 | | 4,311.00s |
| Version 2 | Better file handling | 676.50s |
| Version 3 | Faster sorting | 6.59s |
| Version 4 | No sorting! | 2.35s |

- Version 4 will be released later in the semester…

# Performance Profiling

(*Dracula* vs. *Lewis & Clark*)

| Step | Function | Running Time |
|---|---|---|
| Create vectors: | Read each file | 1,824.00s |
| | Parse each file | 0.20s |
| | Sort words in each file | 328.00s |
| | Count word frequencies | 0.31s |
| Dot product: | | 6.12s |
| Norm: | | 3.81s |
| Angle: | | 6.56s |
| **Total:** | | **72minutes ≈ 4,311.00s** |

# ReadFile (excerpt)

```
// Open the file as a stream and find its size
inputStream = new FileInputStream(fileName);
iSize = inputStream.available();

// Read in the file, one character at a time, normalizing as we go.
for (int i=0; i<iSize; i++)
{
    // Read a character
    char c = (char)inputStream.read();

    // Ensure that the character is lower-case
    c = Character.toLowerCase(c);

    // Check if the character is a letter
    if (Character.isLetter(c))
    {
        strTextFile = strTextFile + c;
    }
    // Check if the character is a space or an end-of-line marker
    else if (((c == ' ') || (c == '\n')) && (!strTextFile.endsWith(" ")))
    {
                strTextFile = strTextFile + ' ';
    }
}
```

# String Problem!

What happens when:

- strTextFile = strTextFile + c

1. Creates new temporary string.
2. Copies strTextFile to the new string.
3. Adds the new character *c*.
4. Reassigns strTextFile to point to the new string.

# String Problem!

What happens when:

- strTextFile = strTextFile + c

1. Creates new temporary string.
2. **Copies strTextFile to the new string**.
3. Adds the new character *c*.
4. Reassigns strTextFile to point to the new string.

Copying a string of k characters takes time k!

# How long does it take here to read a file containing n characters?

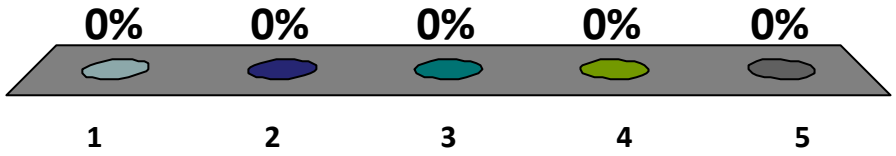
1. O(n)
2. O(n log n)
3. ✓ $O(n^2)$
4. $O(2^n)$
5. Big-O notation?

```
// Open the file as a stream and find its size
inputStream = new FileInputStream(fileName);
iSize = inputStream.available();

// Read in the file, one character at a time, normalizing as
for (int i=0; i<iSize; i++)
{
    // Read a character
    char c = (char)inputStream.read();

    // Ensure that the character is lower-case
    c = Character.toLowerCase(c);

    // Check if the character is a letter
    if (Character.isLetter(c))
    {
        strTextFile = strTextFile + c;
    }
    // Check if the character is a space or an end-of-line ma
    else if (((c == ' ') || (c == '\n')) && (!strTextFile.end
    {
                strTextFile = strTextFile + ' ';
    }
}
```




Response Counter

# String Problem!

How long to read in a file of $n$ characters?.

$$1 + 2 + 3 + 4 + \ldots + n = n(n+1)/2 = \Theta(n^2)$$

Very, very, very slow!

# Fix the string problem!

```
// Open the file as a stream and find its size
inputStream = new FileInputStream(fileName);
iSize = inputStream.available();

// Initialize the char buffer to be arrays of the appropriate size.
charBuffer = new char[iSize];

// Read in the file, one character at a time, normalizing as we go.
for (int i=0; i<iSize; i++)
{
    // Read a character
    char c = (char)inputStream.read();

    // Ensure that the character is lower-case
    c = Character.toLowerCase(c);

    // Check if the character is a letter
    if (Character.isLetter(c))
    {
        charBuffer[iCharCount] = c;
        iCharCount++;
    }
    // Check if the character is a space or an end-of-line marker
    else if (((c == ' ') || (c == '\n')) && (!strTextFile.endsWith(" ")))
    {
        charBuffer[iCharCount] = ' ';
        iCharCount++;
    }
}
```

# Performance Profiling, V2

(*Dracula* vs. *Lewis & Clark*)

| Step | Function | Running Time |
|---|---|---|
| Create vectors: | Read each file | 1.09s |
| | Parse each file | 3.68s |
| | Sort words in each file | 332.13s |
| | Count word frequencies | 0.30s |
| Dot product: | | 6.06s |
| Norm: | | 3.80s |
| Angle: | | 6.06s |
| **Total:** | | **11minutes ≈ 680.49s** |

# Goals for the Semester

Algorithms:

- Design of efficient algorithms
- Analysis of algorithms

Implementation:

- Solve real problems
- Analyze and profile performance
- Improve performance via better algorithms

# Document Distance

(*Dracula* vs. *Lewis & Clark*)

| Version | Change | Running Time |
|---|---|---|
| Version 1 | | 4,311.00s |
| Version 2 | Better file handling | 676.50s |
| Version 3 | Faster sorting | 6.59s |
| Version 4 | No sorting! | 2.35s |

# For next time...

Friday lecture:

- Java introduction
- Object-oriented programming

Friday problem session:

- Example: Elevators!

Discussion Groups:

- None this week. Sign up in CORS.

Problem Set 1:

- Released. Due next week.

# Administrative Details

Registration:

1. If you are not currently registered (via CORS), talk to me.
2. Register for "tutorial" session on CORS.
3. Register for "recitation" on CORS.
4. Join Facebook group

Check your e-mail:

- Invitation to Coursemology
- Invitation to NB

# Puzzle of the Week

Imagine three dice:

- A has six sides: 2, 2, 4, 4, 9, 9
- B has six sides: 1, 1, 6, 6, 8, 8
- C has six sides: 3, 3, 5, 5, 7, 7

# Puzzle of the Week

Imagine three dice:

- A has six sides: 2, 2, 4, 4, 9, 9
- B has six sides: 1, 1, 6, 6, 8, 8
- C has six sides: 3, 3, 5, 5, 7, 7


Game:

- Alice chooses a die.
- Bob chooses a die.
- Alice and Bob both roll.
- The higher value wins.

# Puzzle of the Week

Imagine three dice:

- A has six sides: 2, 2, 4, 4, 9, 9
- B has six sides: 1, 1, 6, 6, 8, 8
- C has six sides: 3, 3, 5, 5, 7, 7


Game:

- Alice chooses a die.
- Bob chooses a die.
- Alice and Bob both roll.
- The higher value wins.

Questions:

- Which die should Alice choose?
- Which die should Bob choose?
- Who is more likely to win?

# Puzzle of the Week

Go discuss in Nota Bene!

Imagine three dice:

- A has six sides: 2, 2, 4, 4, 9, 9
- B has six sides: 1, 1, 6, 6, 8, 8
- C has six sides: 3, 3, 5, 5, 7, 7


Game:

- Alice chooses a die.
- Bob chooses a die.
- Alice and Bob both roll.
- The higher value wins.

Questions:

- Which die should Alice choose?
- Which die should Bob choose?
- Who is more likely to win?