CS2020
Data Structures and Algorithms

Welcome!

Introductions

Seth Gilbert

- Assistant Professor in the CS Department

- Office: COM2-3-23
- Office hours: TBA

- Interests: distributed algorithms

Introductions

Steven Halim

- Lecturer in the CS Department

- Office: COM2-2-60
- Office hours: TBA

- Interests: competitive programming, visualization techniques, ICPC, IOI

Introductions

Tutors: James Yong Kim Leng

- Foodie (Live To Eat)
- Loves: engineering/reverse-engineering/hacking
- Very approachable although don't usually appear to be
- Game programmer
- Favorite IDE: Eclipse

Introductions

Tutors: Jon Hay

- Year 4 undergrad
- Double degree (!) in CS and applied math
- Game development
- Soft spot for: hamsters

Introductions

Tutors: Ngo Minh Duc

- Year 4 undergrad in computer science
- Took CS1101s
- Likes: algorithms and mathematics

Introductions

Tutors: Nguyen Hoanh Tien

- Year 3 undergrad in computer engineering
- Took CS1101s
- Likes: swimming and kite flying

Introductions

Tutors: Zhao Cong

- Competitive programmer since senior middle school
- Just finished internship at PayPal
- Hobbies: reading, blogging, F1, tennis, movies

Introductions

Tutors: Koh Zi Han

- Mysterious and secretive

Introductions

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

Algorithms

"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?

- Speed / Performance
- ???

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, power, heat, parallelism, etc.
  - Scalability
    - Inputs are *large* : e.g., the internet.
    - Bigger problems consume more resources.
- Solve real (fun!) problems

Algorithms

Goals of this course:

- Discover existing “toolbox” of algorithms and data structures that you can use to solve real world problems.
- Learn how to choose the right algorithm for the right problem.
- Learn how to design and analyze new algorithms and data structures when needed.

Algorithms

How to solve a problem:

- Identify the problem
  - Ex: what's the fastest way to get to NUS?
- Abstract irrelevant details
  - Ex: traffic+lights+merging+speed=time
- 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

- Example problems: Google map routes

Semester Overview

- Topic 5: Dynamic Programming
  - All-pairs shortest paths
  - Floyd-Warshall
  - Travelling Salesman Problem

Java

- Goal: Learn Java
  - Quick overview in lectures…
  - More details in recitations / discussion groups.
  - Learn on your own…
  - Solve problem sets.

Why Java?

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

Why Java?

- Good aspects:
  - Common in industry / real-world / web
  - Modularity / Abstraction via OOP
  - Avoids memory leak issues of C/C++

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

Language Does Not Matter

- Algorithms are more important:
  - Fact: C can be 20x as fast as Python!

    (Source: MIT 6.006, Spring 2008, Lecture 2)
  - Fast sorting in Python (merge-sort):
    - Time: $T(n) = 2n\log(n)$ µs
    - Total time for 10,000 elements: 0.266s
  - Slow sorting in C (insertion-sort):
    - Time: $T(n) = 0.01n^2$
    - Total time for 10,000 elements: 1s

Administrative Details

- Weekly schedule:
  - Two lectures: Tues/Fri 10am-12pm
  - One recitation: Friday
  - One discussion group: Thursday

- Weekly work:
  - Problem set
  - Discussion group problems
  - Occasional bonus problems

Administrative Details

Discussion Groups:

- Register via CORS (known as: labs)
- Fill out preference form here!

Tutorials:

- Register via CORS
- Three slots (in COM2-105):
  - 2pm,
  - 3pm
  - 4pm

Administrative Details

- Quizzes:
  - Quiz 1 - Feb. 11 (15%)
  - Practical Programming Quiz – Mar. 3 (10%)
  - Quiz 2 – Mar. 25 (15%)

- Final Exam:
  - Apr. 25 (40%)

- Remainder: problem sets and participation

Administrative Details

Course website:

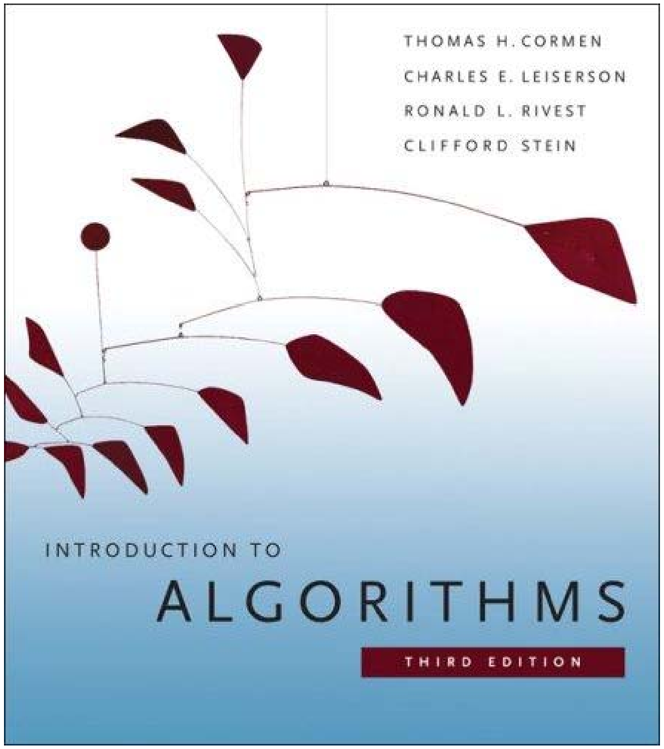
cs2020.ddns.nus.edu.sg

Go register!

- Same infrastructure as CS1101s
  - Experience points…
  - Levels…
  - Facebook connect…
- But no comic strip, no achievements, etc.

Administrative Details

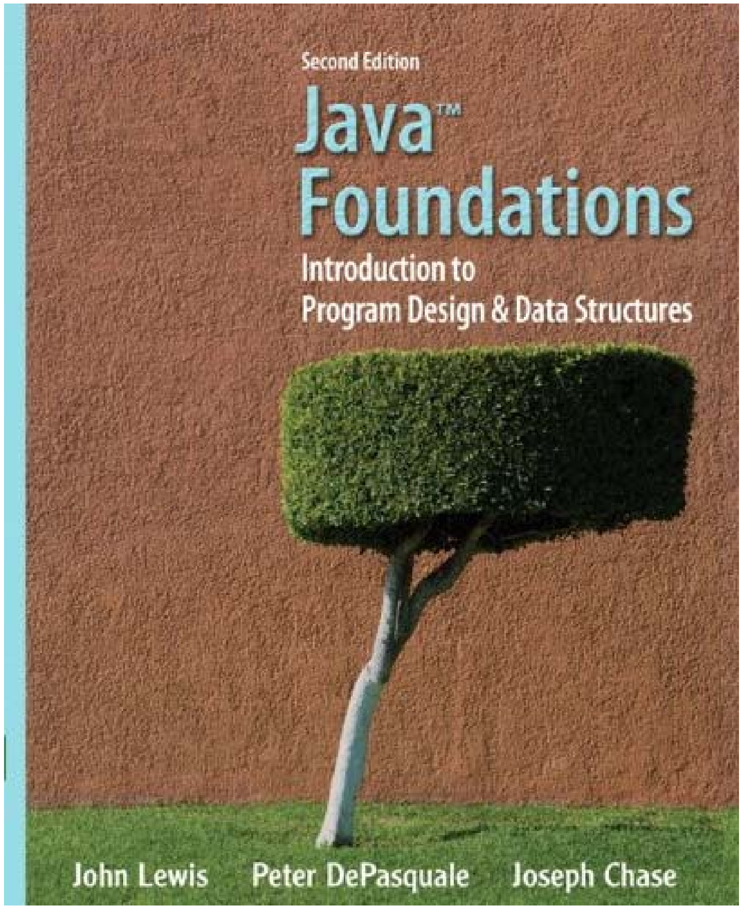
- Textbook: Introduction to Algorithms
  - Cormen, Leiserson, Rivest, Stein



  - Recommended…

Administrative Details

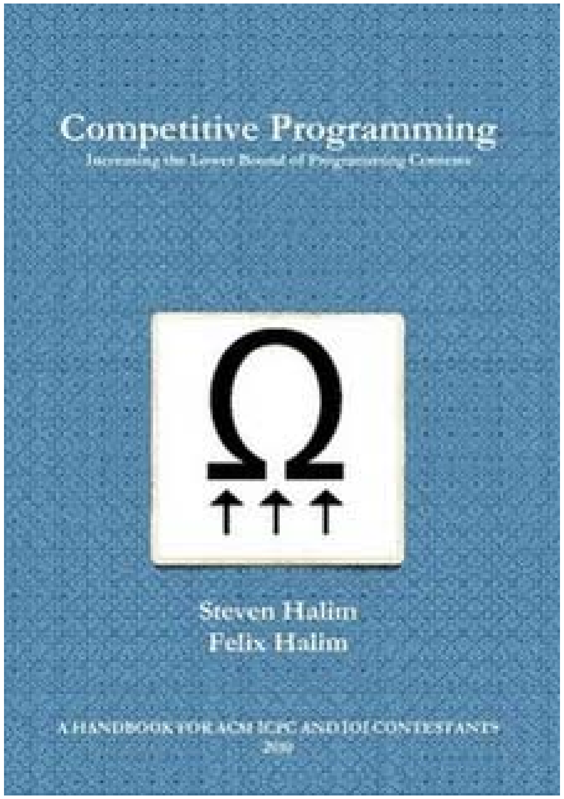
- Textbook: Java Foundations
  - Lewis, DePasquale, Chase



  - Optional…

Administrative Details

- Textbook: Competitive Programming
  - Halim


Competitive Programming
Increasing the Lower Bound of Programming Contests
Ω
Steven Halim
Felix Halim
A HANDBOOK FOR ACM ICPC AND IOI CONTESTANTS
2010

  - Optional…

Problem Sets

- Different types of questions:
  - Give an algorithm…
  - Show that an algorithm is correct…
  - Analyze the performance of an algorithm…
  - Implement an algorithm…
  - Measure the performance of a solution…

Problem Sets

"Give an algorithm…"

"Implement an algorithm…"

Five parts to each answer:

1. State the problem being solved.
2. Describe the solution in words. (If it is a Java algorithm, describe the algorithm being implemented.)
3. Give the algorithm/Java.
4. Explain why it works.
5. (Optional) Analyze its performance.

Problem Sets

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

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

The tutor is not a compiler.

Problem Sets

Collaboration Policy

- Working together is strongly encouraged!
- You must write-up 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.

Clickers...

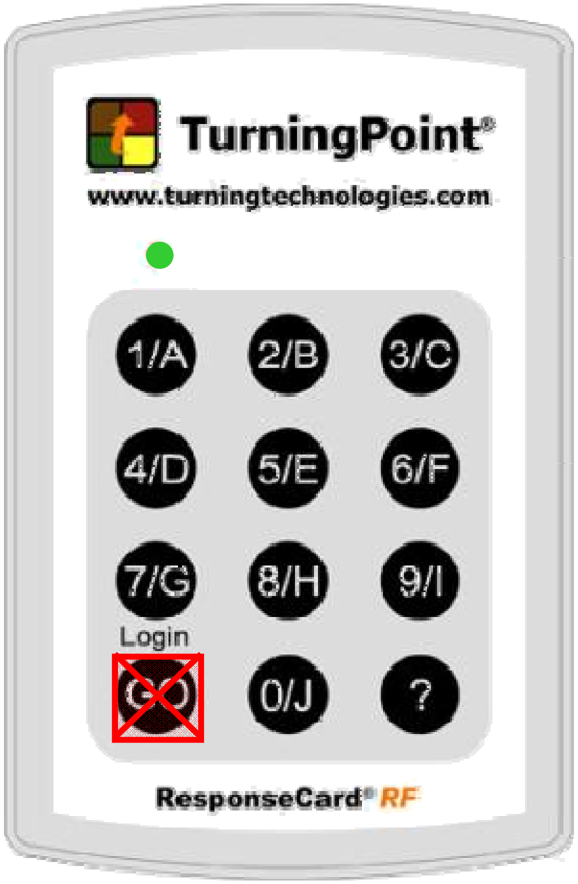
Who is your favorite author?

a) Shakespeare

b) J.K. Rowling

c) Confucius

d) Homer Simpson



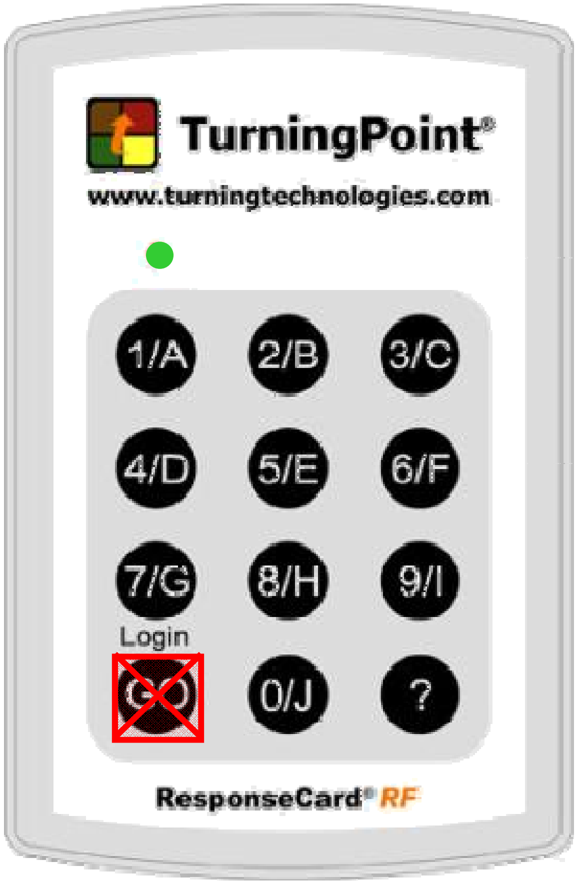
Clickers...

Today:

Sign up for a clicker.

If your clicker is missing:

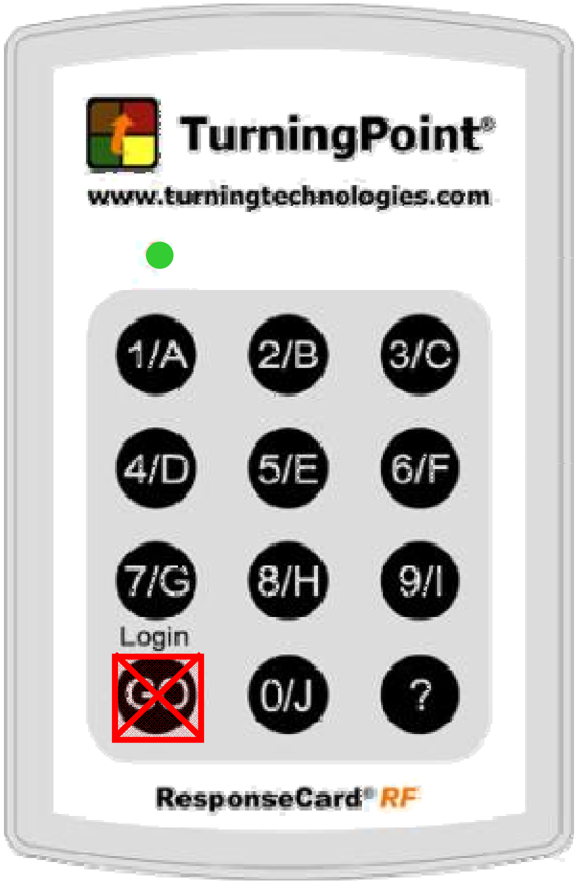
S$89 / clicker.



Clickers...

1. Simply choose your response from the keypad buttons.
2. The light will go **GREEN** to confirm your response has been received.
3. You can **change your answer** by simply keying in your new choice.

(The system will only count the last vote.)



NOTE:
Please DO NOT press the **GO** button, this will change the Radio Frequency of the Keypad.

Break time

Questions?

Sign up for a clicker…

Sign up for a “Discussion Group”…

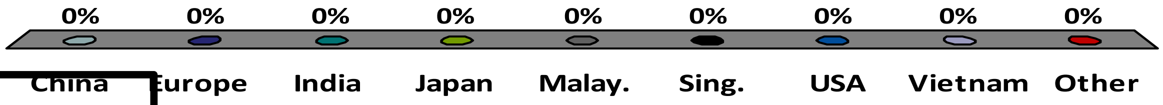
Today

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

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

Where are you from?

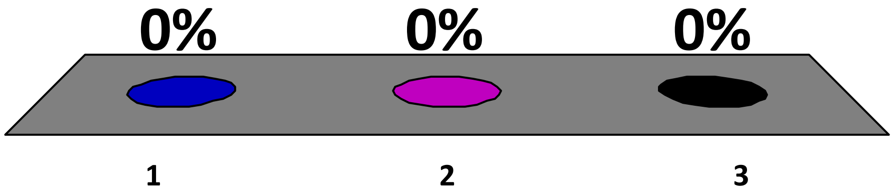
1. China
2. Europe
3. India
4. Japan
5. Malaysia
6. Singapore
7. United States
8. Vietnam
9. Other…



0 of 5

Are you:

1. Male
2. Female
3. Other



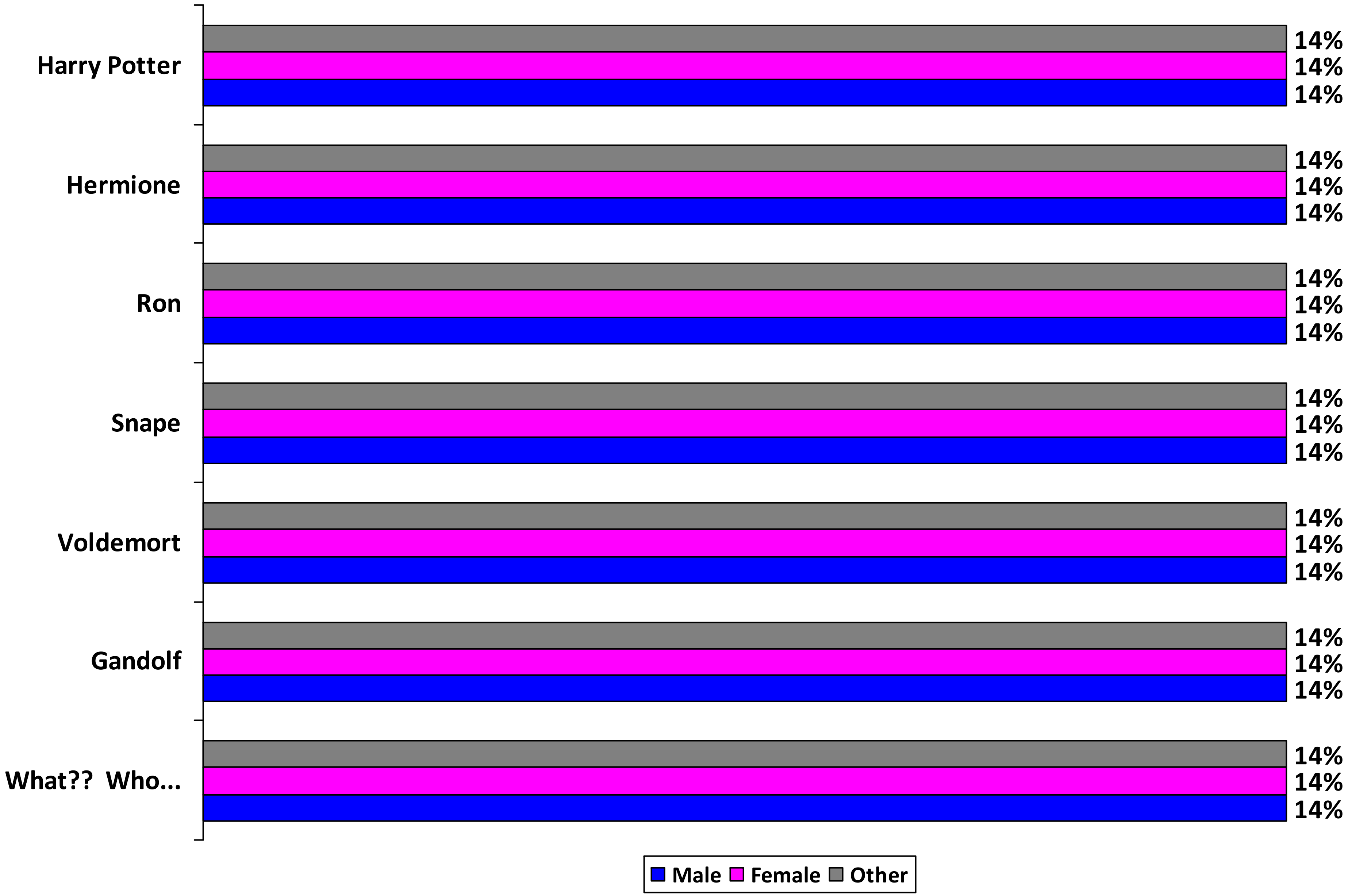
0 of 5

Who’s your favorite Harry Potter character?

0%	1. Harry Potter
0%	2. Hermione
0%	3. Ron
0%	4. Snape
0%	5. Voldemort
0%	6. Gandolf
0%	7. What?? Who’s that?

0 of 5

Who’s your favorite Harry Potter character?

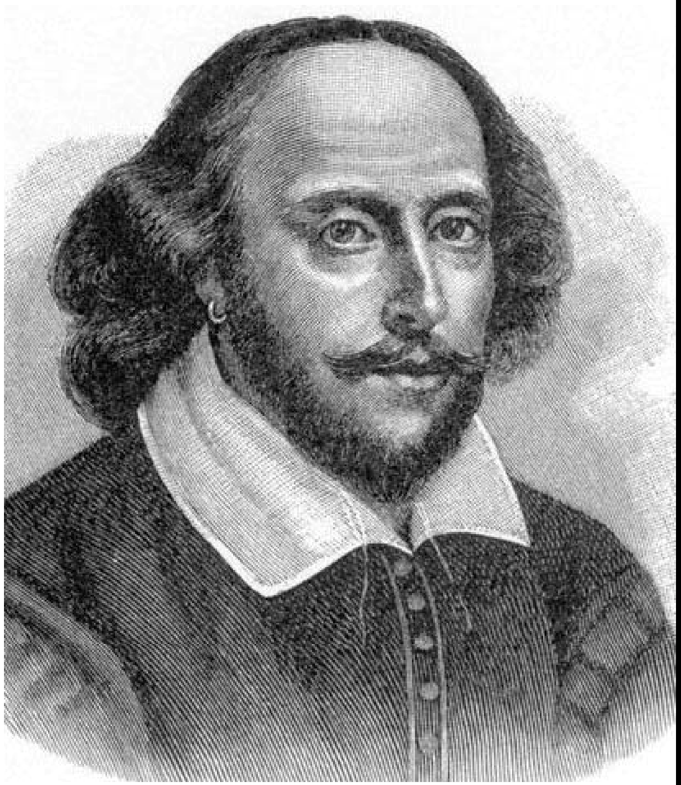


Today

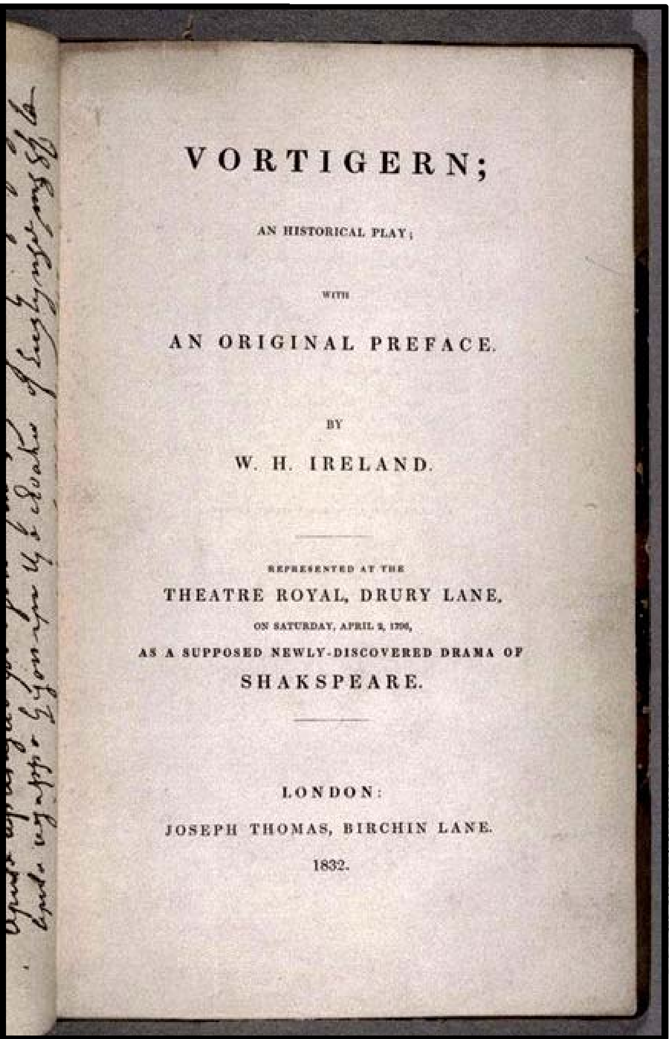
- Problem: Document Distance
  - How similar are two documents?

- 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

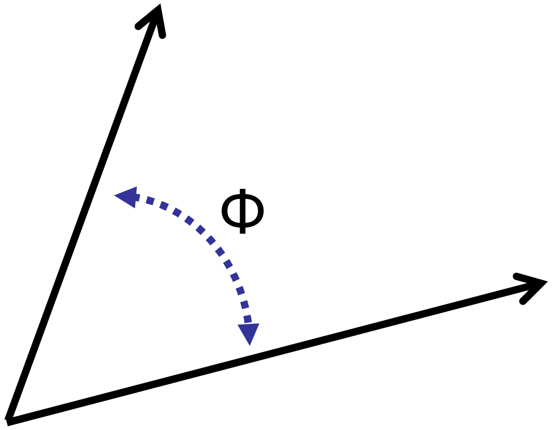
- Binary: (e.g., detect plagiarism)
  - Exactly same words in same order

- 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

Vector Space Model

Strategy:

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



- Identical: Φ = 0
- No words in common: Φ = π/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

Document as vector:

Example 1:

“to be or not to be” = [0,2,0,1,0,1,2]

afraid	be	greatness	not	of	or	to
0	2	0	1	0	1	2

Vector Space Model

Example 1:

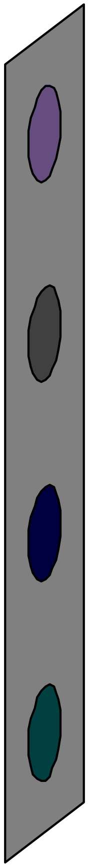
“to be or not to be” = [0,2,0,1,0,1,2]

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”



0 of 5

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

0% b. [✓ 2, 2, 0, 1, 0, 0, 2]

0% c. 3, 2, 3, 1, 1, 1, 1

0% d. I have no idea.

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

Vector Space Model

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

1. [1, 1, 1, 3, 0, 0, 2]
2. [2, 2, 0, 1, 0, 0, 2]
3. [3, 2, 3, 1, 1, 1, 1]
4. 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:

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

$(v \cdot w) =$

0% a. 1

0% b. 2

0% 🙂 3

0% d. 4

0% e. 5

0 of 5

Vector Space Model

Norm of a vector (L2 norm):

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

Example: distance between two points

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

Vector Space Model

Norm of a vector (L2 norm):

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

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

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:

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

Compare Two Documents

Given: documents A and B

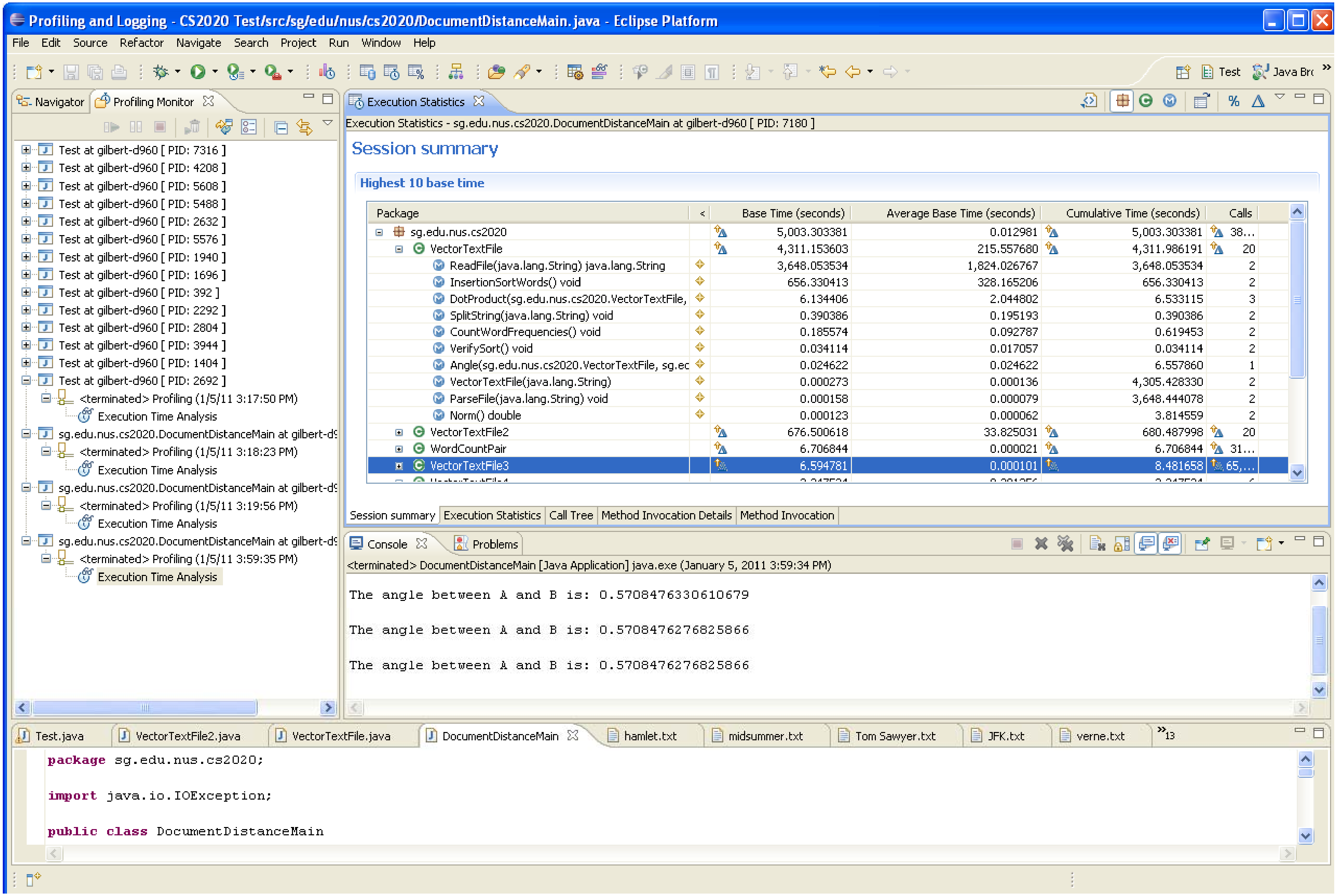
1. Create vectors v_A and v_B
2. Calculate norm: $|v_A|$
3. Calculate norm: $|v_B|$
4. Calculate dot product: $(v_A \cdot v_B)$
5. Calculate angle $\Phi(v_A, v_B)$

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



Test at gilbert-d960 [PID: 7316]
Test at gilbert-d960 [PID: 4208]
Test at gilbert-d960 [PID: 5608]
Test at gilbert-d960 [PID: 5488]
Test at gilbert-d960 [PID: 2632]
Test at gilbert-d960 [PID: 5576]
Test at gilbert-d960 [PID: 1940]
Test at gilbert-d960 [PID: 1696]
Test at gilbert-d960 [PID: 392]
Test at gilbert-d960 [PID: 2292]
Test at gilbert-d960 [PID: 2804]
Test at gilbert-d960 [PID: 3944]
Test at gilbert-d960 [PID: 1404]
Test at gilbert-d960 [PID: 2692]
<terminated> Profiling (1/5/11 3:17:50 PM)
Execution Time Analysis
sg.edu.nus.cs2020.DocumentDistanceMain at gilbert-d9
<terminated> Profiling (1/5/11 3:18:23 PM)
Execution Time Analysis
sg.edu.nus.cs2020.DocumentDistanceMain at gilbert-d9
<terminated> Profiling (1/5/11 3:19:56 PM)
Execution Time Analysis
sg.edu.nus.cs2020.DocumentDistanceMain at gilbert-d9
<terminated> Profiling (1/5/11 3:59:35 PM)
Execution Time Analysis

Execution Statistics - sg.edu.nus.cs2020.DocumentDistanceMain at gilbert-d960 [PID: 7180]

Session summary

Highest 10 base time

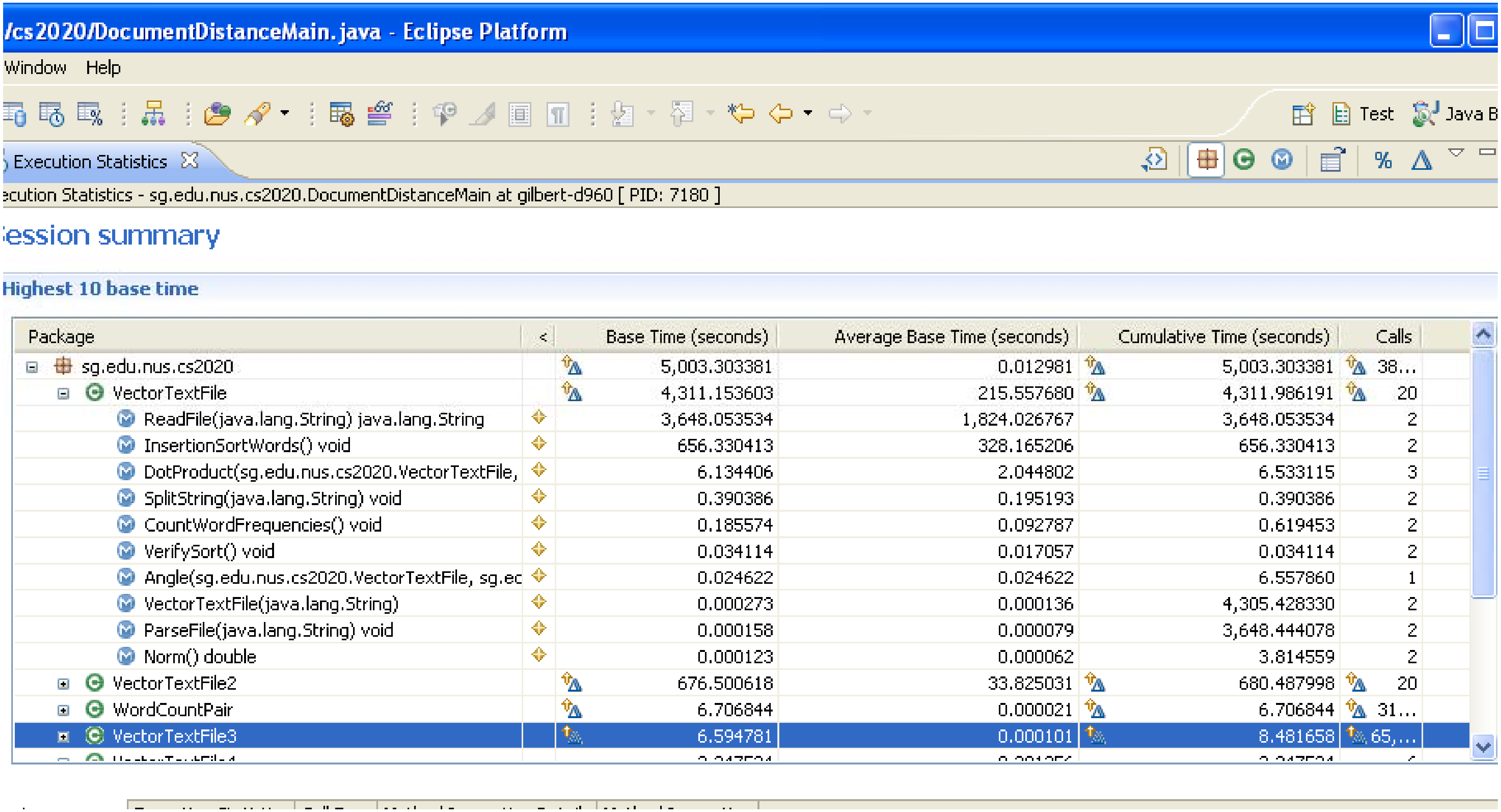
Package	<	Base Time (seconds)	Average Base Time (seconds)	Cumulative Time (seconds)	Calls
sg.edu.nus.cs2020		5,003.303381	0.012981	5,003.303381	38...
VectorTextFile		4,311.153603	215.557680	4,311.986191	20
ReadFile(java.lang.String) java.lang.String		3,648.053534	1,824.026767	3,648.053534	2
InsertionSortWords() 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,...

<terminated> DocumentDistanceMain [Java Application] java.exe (January 5, 2011 3:59:34 PM)

```
The angle between A and B is: 0.5708476330610679

The angle between A and B is: 0.5708476276825866

The angle between A and B is: 0.5708476276825866
```

```
package sg.edu.nus.cs2020;

import java.io.IOException;

public class DocumentDistanceMain
```

Eclipse-TPTP



/cs2020/DocumentDistanceMain.java - Eclipse Platform

ecution Statistics - sg.edu.nus.cs2020.DocumentDistanceMain at gilbert-d960 [PID: 7180]

ession summary

Highest 10 base time

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

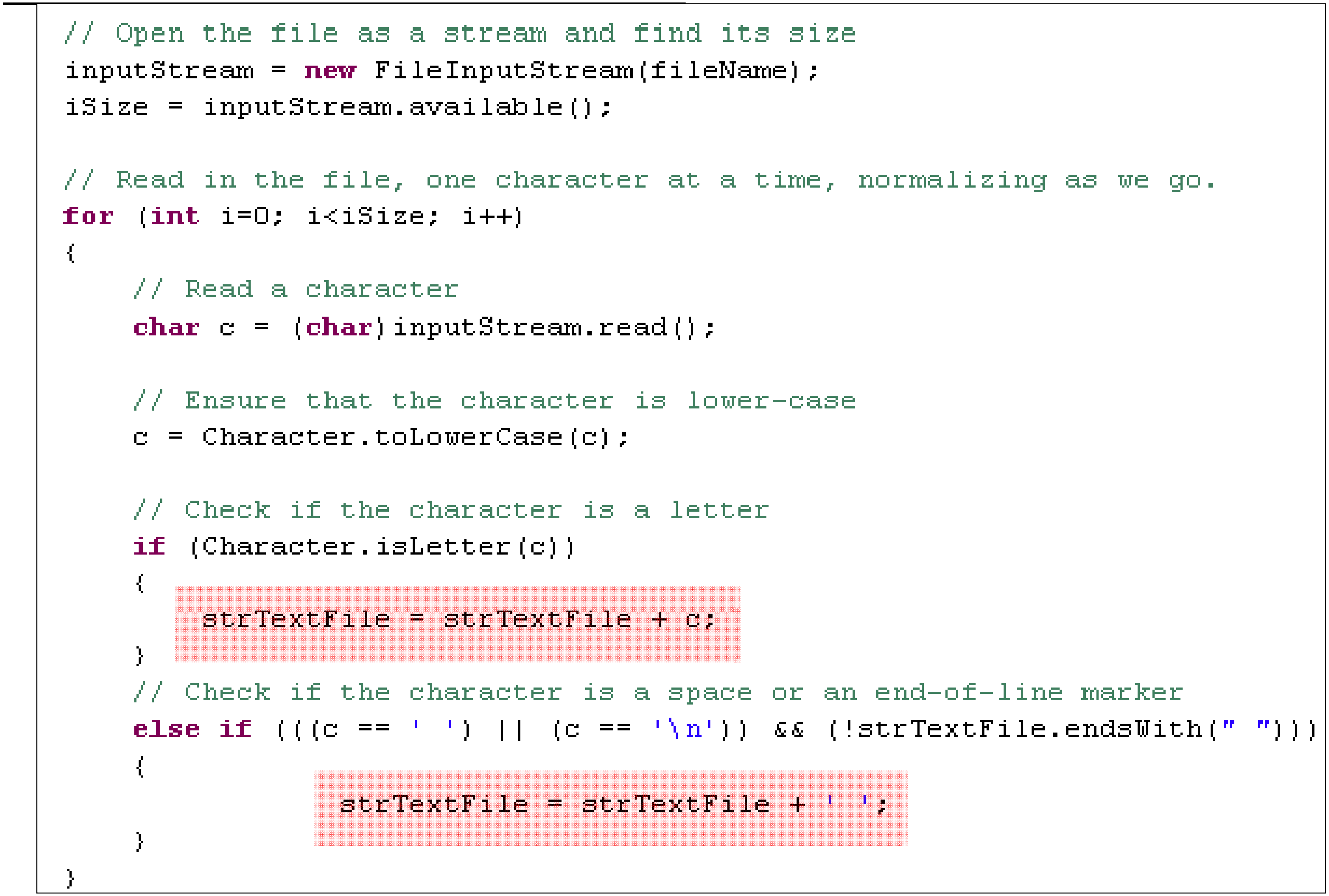
- Version 4 will be released later in the semester…

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:

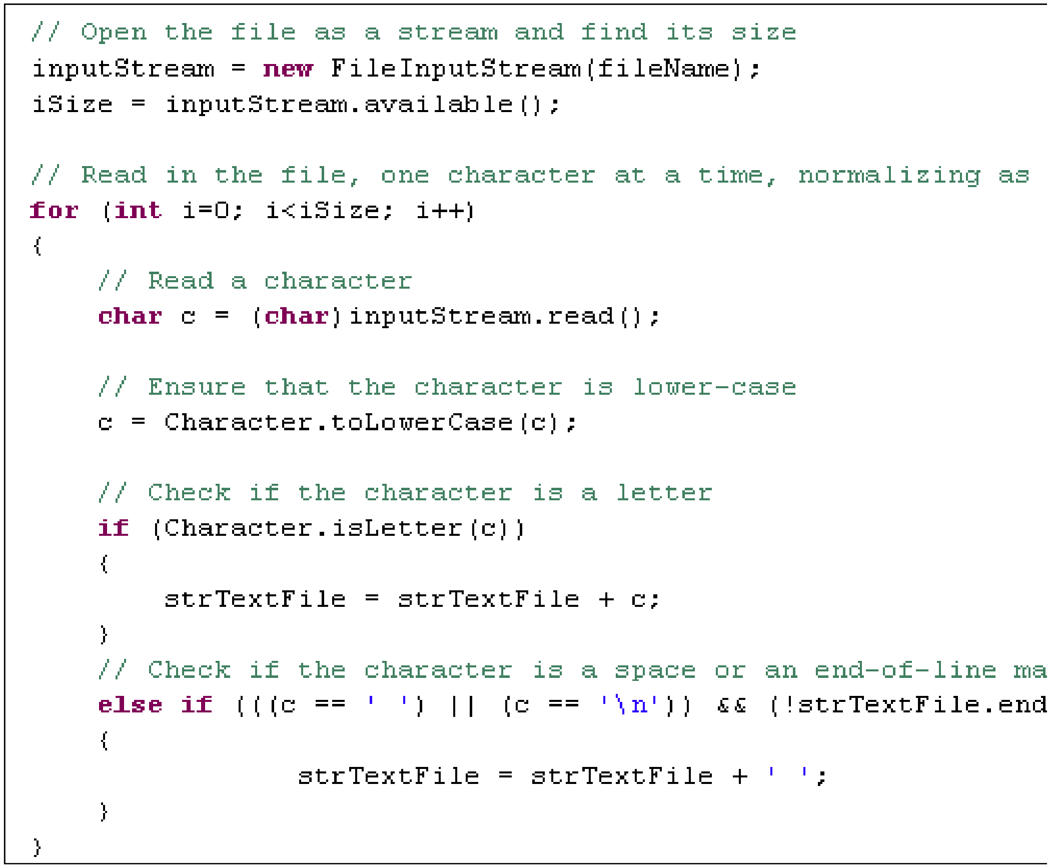
- strTextFile = strTextFile + c

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.

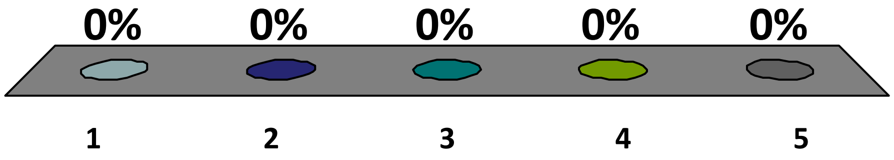
Copying a string of k characters takes time O(k)!

How long does it take 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 + ' ';
    }
}
```



0 of 5

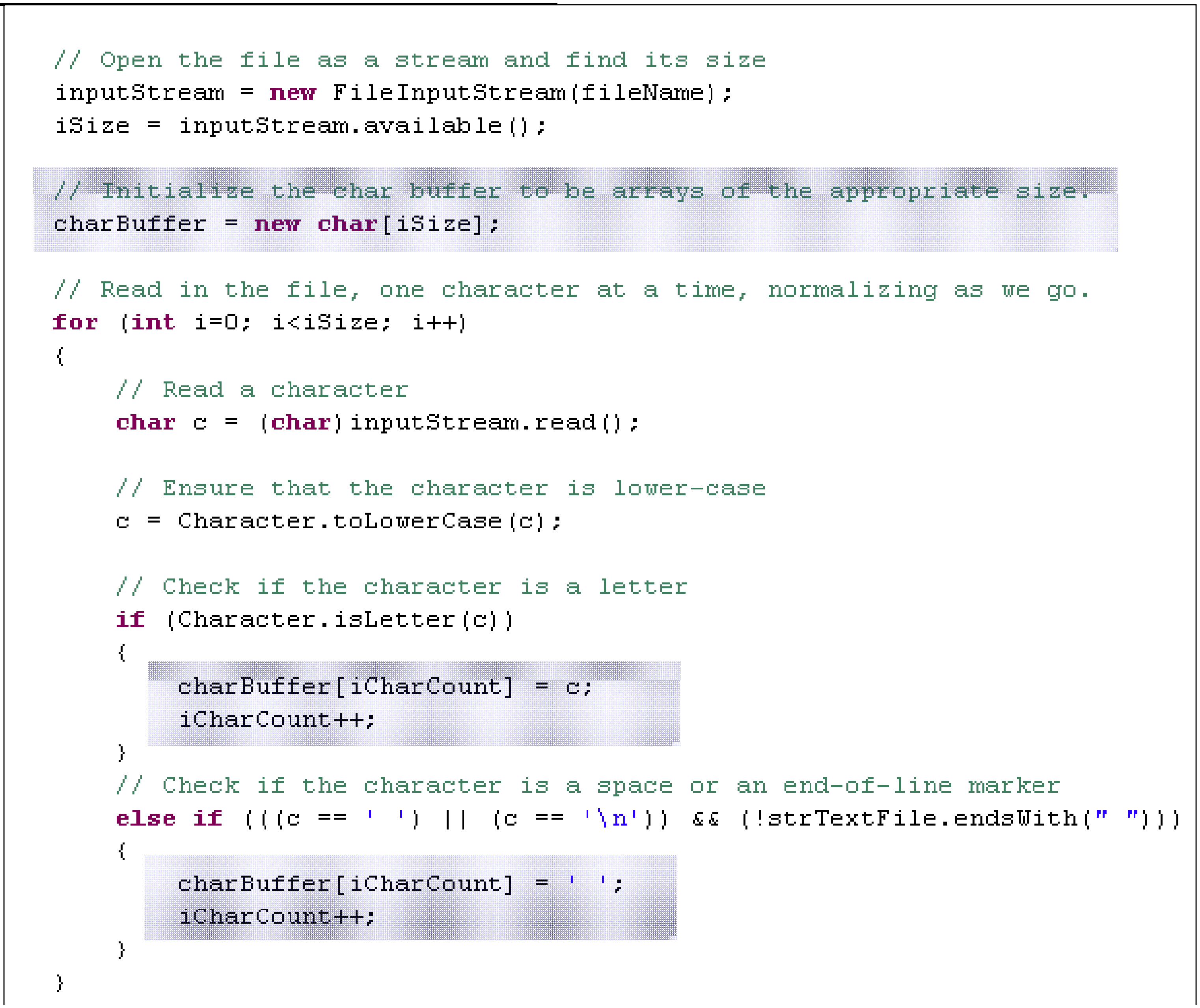
String Problem!

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

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

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:

- Object-oriented programming
- Doc. Distance V3: Sorting

Friday tutorial:

- Details of Document Distance implementation

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), send me an e-mail.
2. Go to cs2020.ddns.nus.edu.sg and register.
3. Register for “Tutorial” session on CORS.
4. Register for “Lab” (Discussion Group) on CORS.
5. Fill out Discussion Group Preference form.