CS2040S
Data Structures and Algorithms

Welcome!

(Go try it.)

I do not monitor Zoom chat.

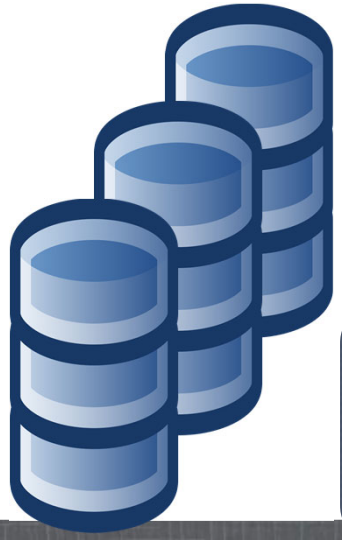
Algorithms Everywhere

Web browser:
Parsing
Substring manipulation (Week 7)
XML trees (Week 5)

Internet

Internet routing:
TCP (congestion control)
IP routing
BGP (Bellman-Ford, Week 9)
Content caching (Week 7)
DNS

Google:
PageRank
(Week 10)
String matching
(Week 7)



Web servers:
Load balancing (PS 2)
Scheduling (Week 8)
Memory allocation

Database:
B-trees (Week 5)
Search (Week 2)
Sorting (Week 3)

Data Structures

Problem Solving

Algorithms

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 (and precise)
- Designed to accomplish some task

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

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

WHAT ARE YOUR GOALS?

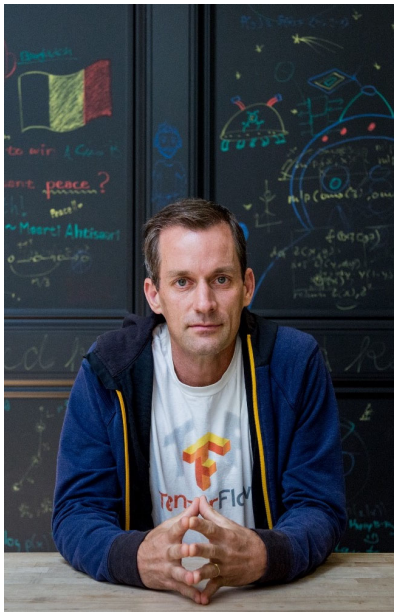
Did you answer on archipelago?

Yes or No on Zoom.

ARCHIPELAGO

is open

BUILD USEFUL SYSTEMS FOR PEOPLE



Jeff Dean
(Google)

Ruchi Sanghvi
(Facebook)

BUILD FAST SYSTEMS

"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*

"Bad programmers worry about the code. Good programmers worry about ***data structures* and *their relationships*.**"

- Linus Torvalds*

*Creator of git and the linux kernel (in Linux OS, Chrome OS, Android)

START A TECH COMPANY

THINK ABOUT BEAUTIFUL ALGORITHMS

Donald Knuth

``People who analyze algorithms have double happiness. First of all they experience the sheer beauty of elegant mathematical patterns that surround elegant computational procedures. Then they receive a practical payoff when their theories make it possible to get other jobs done more quickly and more economically."

CHANGE THE WORLD

COMPUTER SCIENTISTS HAVE CHANGED THE WORLD…

. . . BY SOLVING PROBLEMS

CS2040S is about **solving problems**.

It prepares you for what lies ahead…

Search

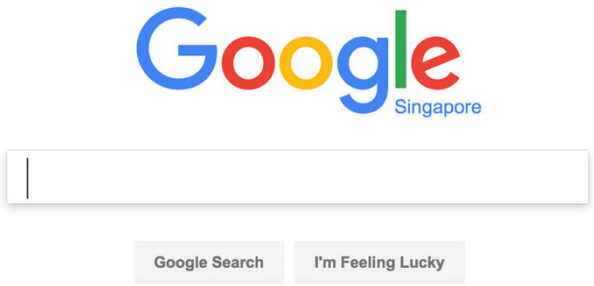
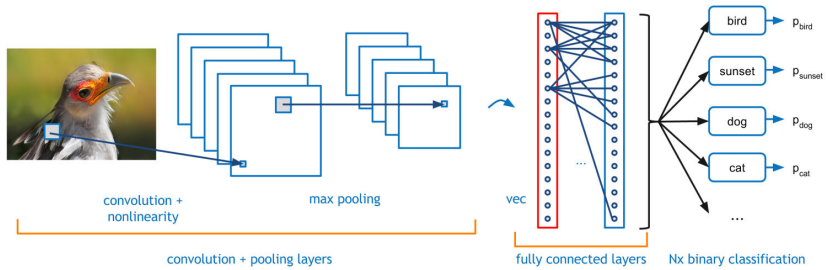


Image Recognition



WHAT ARE YOUR GOALS?

WHAT ABOUT THE CODE?

WHAT ABOUT THE CODE?

CS2040S is about **solving problems** in Java.

Why Java?

Language does not really matter!

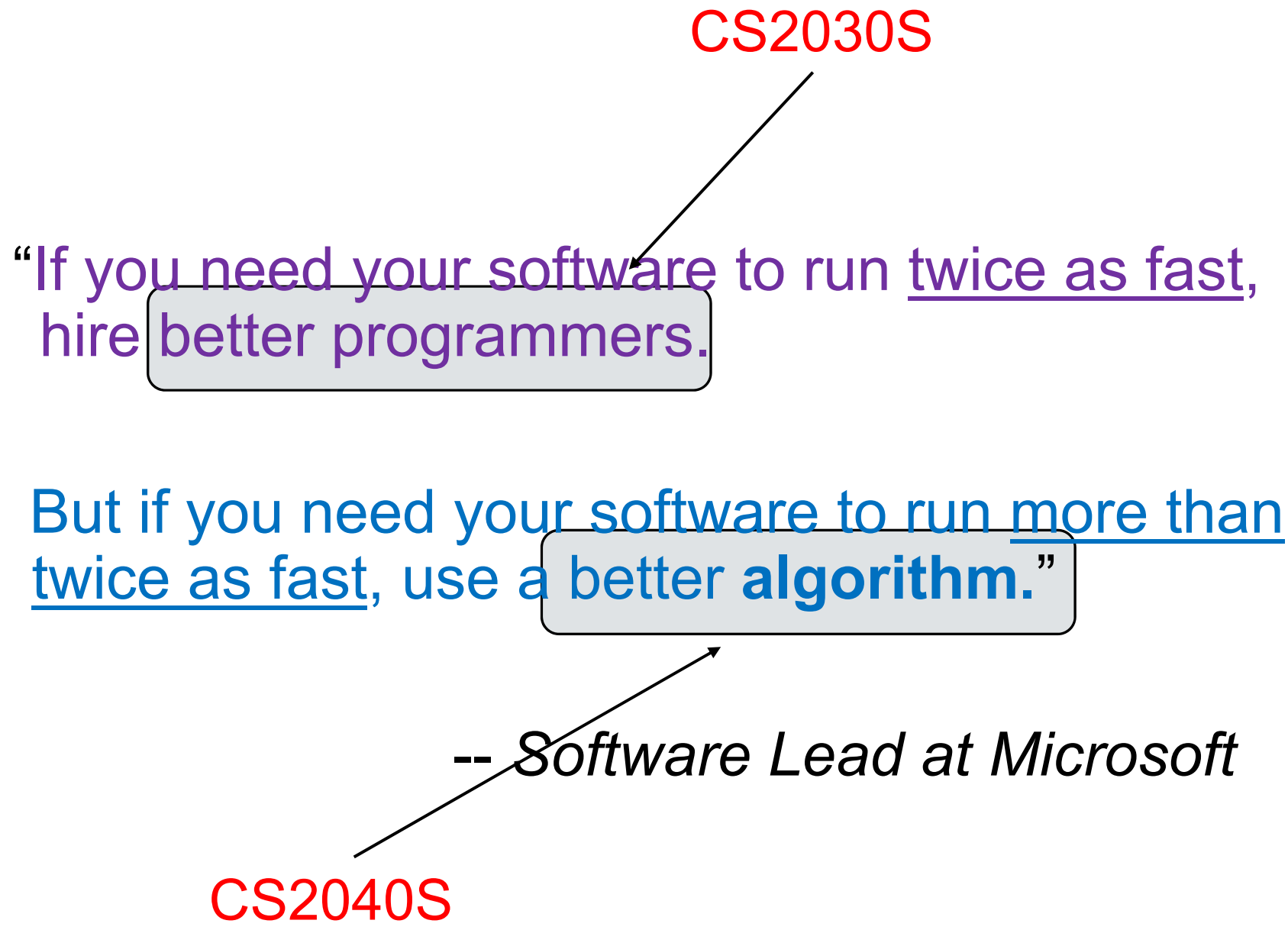
- Good aspects:
  - Common in industry / real-world / web
  - Object-oriented in a deep way
  - Modularity / abstraction via OOP
  - Avoids memory leak issues of C/C++

- Less good aspects:
  - Performance (compare to: C++)
  - More mental overhead (compare to: Python)

WHAT ABOUT THE CODE?

CS2040S is about **solving problems** in Java.

CS2040S is not about learning Java.


CS2030S
"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*
CS2040S

Assumption:

- You are taking CS2030S this semester.

OR

- You already took CS2030S a previous semester.

OR

- You already are an expert Java programmer.

OR

- You plan to spend some extra time learning Java.

WHAT ABOUT THE CODE?

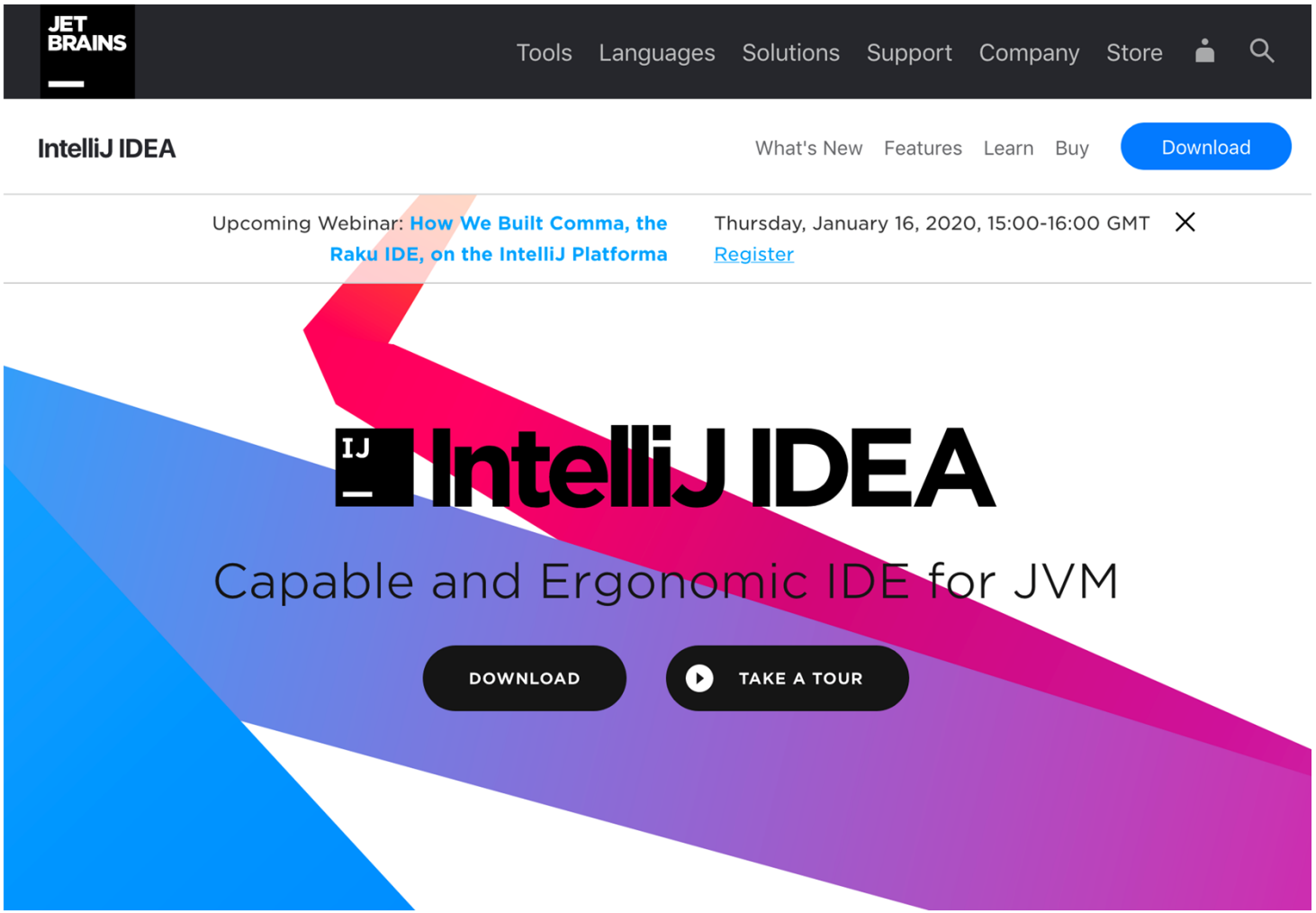
CS2040S is about **solving problems** in Java.

CS2040S is not about learning Java.

We want you to be great programmers too!
Staff are here to help.

Always program using the best development environment you can!

CS2040S Recommended IDE: **IntelliJ**



ARCHIPELAGO

is open

Introductions

Teaching Team

Seth Gilbert

Ben Leong

Teaching Team

Prof. Seth

Prof. Ben

What should I call you?

- If I mispronounce your name, please correct me.
- If the name you use is different than the one I have, please tell me.
- If I am addressing you incorrectly, let me know.

Teaching Team

Seth Gilbert

Interests:
Algorithms
Distributed Algorithms
Parallel Algorithms
Optimization

Ben Leong

Interests:
Distributed Systems
Networking
Mobile Networks
SDN

Teaching Team

Seth Gilbert

Talk to me about:
Curriculum questions
Algorithms questions
CS Puzzles

Ben Leong

Talk to me about:
Logistics issues
Tutorial scheduling
Coursemology
How to succeed in life

Teaching Staff

Tutors:

~40 tutors

NUS undergrads

Former CS2040 students

Teaching Assistant:

Jin Zhe

PhD student in CS

Administrative Details

WEEKLY SCHEDULE

2x Lectures (online)

- Tuesday: 4pm-6pm
- Thursday: 4pm-5pm

1x Recitation

- 13+ slots (Monday/Tuesdays)
- **Starts in Week 3**

1x Tutorial

- 14+ slots (Thursdays/Fridays), ~40 options
- **Starts in Week 3**

Administrative Details

Lectures:

- Two lectures:

  Tuesday. 4pm-6pm

  Thurs. 4pm-5pm

- Lecturer: me!

Live Zoom lecture:

- Participation expected
- Recorded
- Slides posted after lecture

Administrative Details

Recitations:

- One recitation: Monday/Tuesday (1 hour)

  13 different sessions
- Size: ~40 students
- Goal: algorithm design and problem solving

Instructors:

- Prof. Ben + Jin Zhe

Tutorial + Recitation Participation:
~10% of your grade

Administrative Details

Recitations:

- One recitation: Monday/Tuesday (1 hour)

  13 different sessions
- Size: ~40 students
- Goal: algorithm design and problem solving

Register:

- Survey on Coursemology by ~~Thurs.~~ Weds. 11:59pm
- NO ModReg.
- Sessions start Week 3.

Administrative Details

Tutorials:

- One tutorial: Thursday/Friday (2 hours)

  40 different sessions

- Size: ~15 students

Tutorial + Recitation Participation:
~10% of your grade

Your tutor's job:

- Help you learn the material.
- Provide advice on how best to learn.
- Help you catch up if you fall behind.
- Challenge you if you are ahead.
- Show you neat exciting aspects of CS2040S.

Administrative Details

Tutorials:

- One tutorial: Thursday/Friday (2 hours)

  40 different sessions

- Size: ~15 students

How to sign up?

- Survey on Coursemology by ~~Thurs.~~ Weds. 11:59pm.
- Please be flexible: fill in as many sessions as you can.
- Please be patient.
- Sessions start on Week 3.

WEEKLY SCHEDULE

2x Lectures (online)

- Tuesday: 4pm-6pm
- Thursday: 4pm-5pm

1x Recitation

- 13+ slots (Monday/Tuesdays)
- **Starts in Week 3**

1x Tutorial

- 14+ slots (Thursdays/Fridays), ~40 options
- **Starts in Week 3**

Administrative Details

Midterm:

- March 9
- If you cannot make it, notify us *at least* two weeks in advance. *(Exception: illness.)*

Final Exam:

- Sat. April 24 (13:00) --- *check for updates*.

DROP DATES

Check ModReg: http://www.nus.edu.sg/ModReg/

My current impression:

Drop with:

- **No record:** until **24/01/2021 11:59pm.**
- **'W'** grade: **25/01/2021 00:00** through **28/02/2021 11:59pm.**
- **'F'** grade: **01/03/2021 00:00** onwards.

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

Built here at NUS by Prof. Ben and his team!

Platform for course management:

- Announcements
- Lecture slides
- Problem sets
- Feedback
- Discussion forum
- Etc.

Please check Coursemology, and read your (Coursemology) e-mail.

Replacement for Luminus.

Coursemology

How do you join?

- If you are registered for the class, you have already received an invitation.
- If you are not registered for the class, you will receive an invitation when you do register.
- If you are auditing the class, e-mail us.

If something doesn't work…. ask on forum!

Luminus

Reasons to check Luminus?

- None.

Least interesting part of CS2040S?

- Grades

  AND

- Talking about grades

  AND

- Optimizing for grades

  AND

- Thinking about grades

Solution: Everything gives EXP

- Participation

  AND

- Assignments

  AND

- Surveys

  AND

- Optional exercises

EXP ➔ Levels

- final level = grade for classwork
- Maximum grade: 35
- Maximum level: > 35
- Bonus / optional work may lead to levels higher than 35.
- Goal: everyone reaches level 35 (or close)

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.

How to be good at ~~anything~~ CS2040S?

Knowledge

Experience

Talent

Practice, practice, practice…

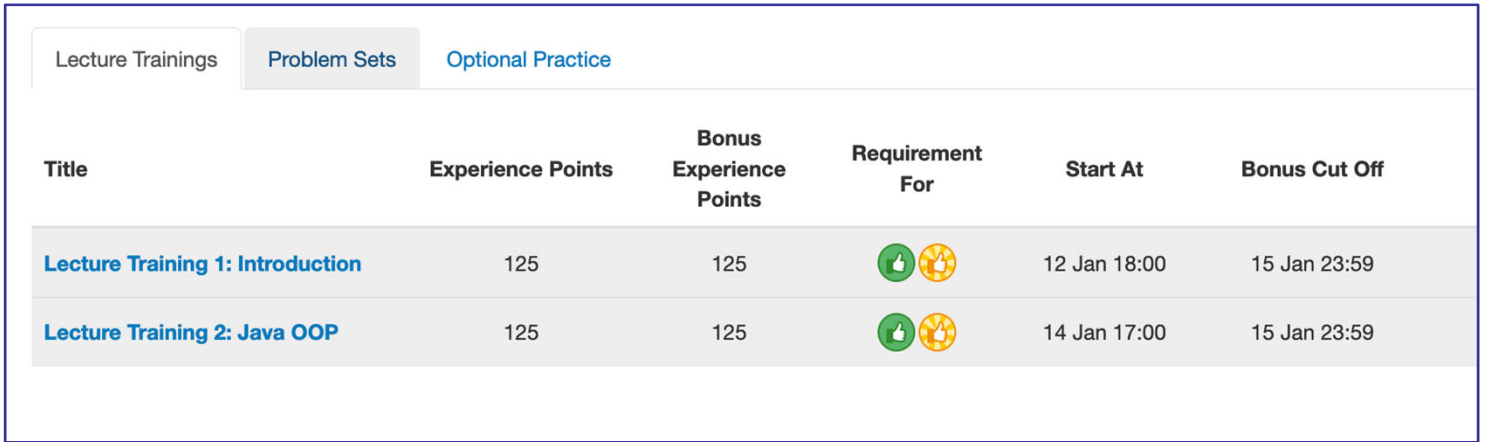
Lecture Training

Quick review of topics from class

< 15 minutes

Tuesday/Thursday: both due Friday 11:59pm

(50% reduced EXP after deadline)



Lecture Trainings | Problem Sets | Optional Practice

Title	Experience Points	Bonus Experience Points	Requirement For	Start At	Bonus Cut Off
Lecture Training 1: Introduction	125	125		12 Jan 18:00	15 Jan 23:59
Lecture Training 2: Java OOP	125	125		14 Jan 17:00	15 Jan 23:59

Problem Sets

Practice the techniques from class.

Find out what you do and do not understand.

Practice regularly, every week.

Problem Sets

Practice the techniques from class.

Find out what you do and do not understand.

Practice regularly, every week.

Problem set 1 is available on Coursemology….
as soon as you complete the Lecture Training.

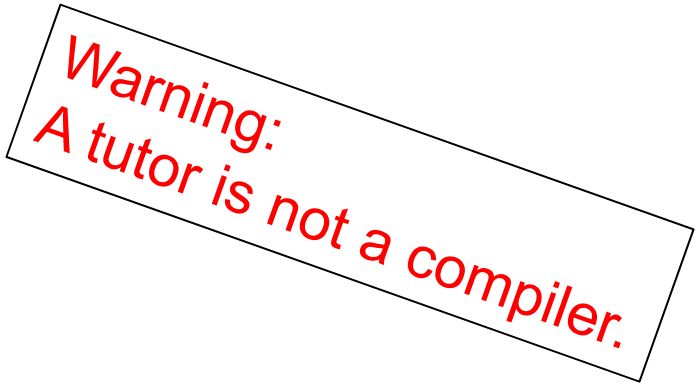
Problem Sets

Some problems may be hard.

Some problems may involve figuring something out that we haven't done in class!

Most problems will involve writing some Java code.

Your tutor is there to help!



Problem Sets

Submit problem sets on Coursemology.

Late submissions:

- 1 week: 25% penalty
- Last day of class: 50% penalty

Hand in problem sets on time!

Even if late, do them anyways!

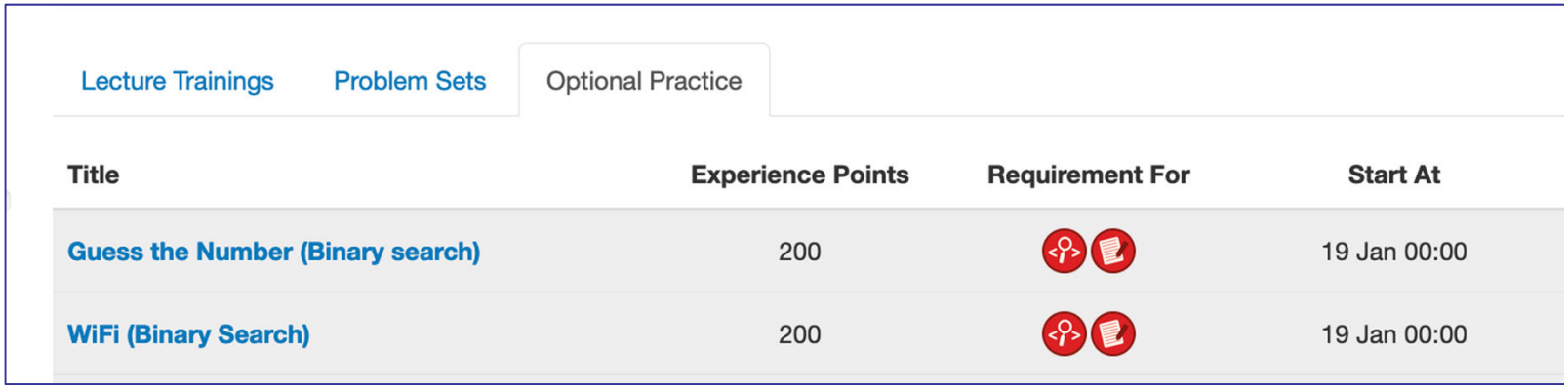
Practice Problems

Ensure that you really know how to use the algorithms & data structures.

Easier than problem sets

Good practice for coding skills (interviews, etc.)

Simple problem solving



Recitations

Problem solving

(Apply ideas from lecture.)

Problem solving techniques

(How to go about solving hard problems.)

Missing details

(Fill in missing pieces from lecture.)

Tutorials

Type 1: Review Problems

(Builds directly on ideas from lectures.)

Type 2: Group Challenge Problems

(Work together to solve a hard problem.)

Who here plays chess?

Rybka

Won four consecutive World Chess Championships (2007-2011).

Today's best chess engines:

- AlphaZero
- Stockfish
- Komodo
- …

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

Czech mate, Mr Cheat

IT'S a story that has seen 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 shafted 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 36-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 his 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

By **Tariq Tahir**

the International Computer Games Association world championship from 2007 to 2010.

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 e-mail 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, Iweta, 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 Iweta

Disqualified!

Titles revoked!

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

Czech mate, Mr Cheat

By Tariq Tahir

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

Arguments against:

- Relied on code from Crafty and Fruit.
- Did not meet the ”originality” rules for the tournament.

Arguments for:

- It was much, much better than Crafty, Fruit, or any existing chess player!

ARCHIPELAGO

is open

Problem Sets

Collaboration Policy

- Working together is strongly encouraged!
- You must write/code your problems sets alone.
- Cheating / plagiarism will be dealt with harshly.

WHAT ARE YOUR GOALS?

- Learn something.
- Become a better person.
- Position yourself to succeed after graduation.
- *Get a good grade on a problem set in CS2040S??*

WHY DID YOU CHEAT?

Friday, July 1, 2011 METRO

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

Czech mate, Mr Cheat

By **Tariq Tahir**

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 shamed 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 2007 to 2010.

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 e-mail 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, Iweta, 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 Iweta

WHY DID YOU CHEAT?

Friday, July 1, 2011 METRO

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

Czech mate, Mr Cheat

By Tariq Tahir

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

- I thought I was going to fail the class!

If you put in the time, you will not fail the class.

If you cheat, you will learn less, and likely do worse in the end.

WHY DID YOU CHEAT?

Friday, July 1, 2011 METRO

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

Czech mate, Mr Cheat

By Tariq Tahir

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 shamed 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 2007 to 2010.

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 e-mail 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, Iweta, 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 Iweta

- I'm not smart enough…
- Everyone else is smarter than I am…

You are smart enough; but you need to keep practicing, keep working at it.

WHY DID YOU CHEAT?

Friday, July 1, 2011 METRO

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

Czech mate, Mr Cheat

By Tariq Tahir

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

- I thought I was going to fail the class! *You won't (unless you cheat).*
- Everyone else is cheating! *No, they aren't.*

ARCHIPELAGO

is open

WHY DID YOU CHEAT?

Friday, July 1, 2011 METRO

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

Czech mate, Mr Cheat

By Tariq Tahir

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 2005 to 2010.

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 e-mail 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, Iweta, 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 Iweta

- I thought I was going to fail the class! *You won't (unless you cheat).*
- Everyone else is cheating! *No, they aren't.*
- It wasn't really cheating... *The rules are the rules.*

WHAT ARE YOUR GOALS?

- Learn something.
- Become a better person.
- Position yourself to succeed after graduation.

Where can I get help?

Your tutor!

Discussion Forums

Coursemology forums:

- Lectures
- Problem Sets
- Java
- Recitations & Tutorials
- and more….

Ask questions on Coursemology

(Don’t e-mail unless it is private.)

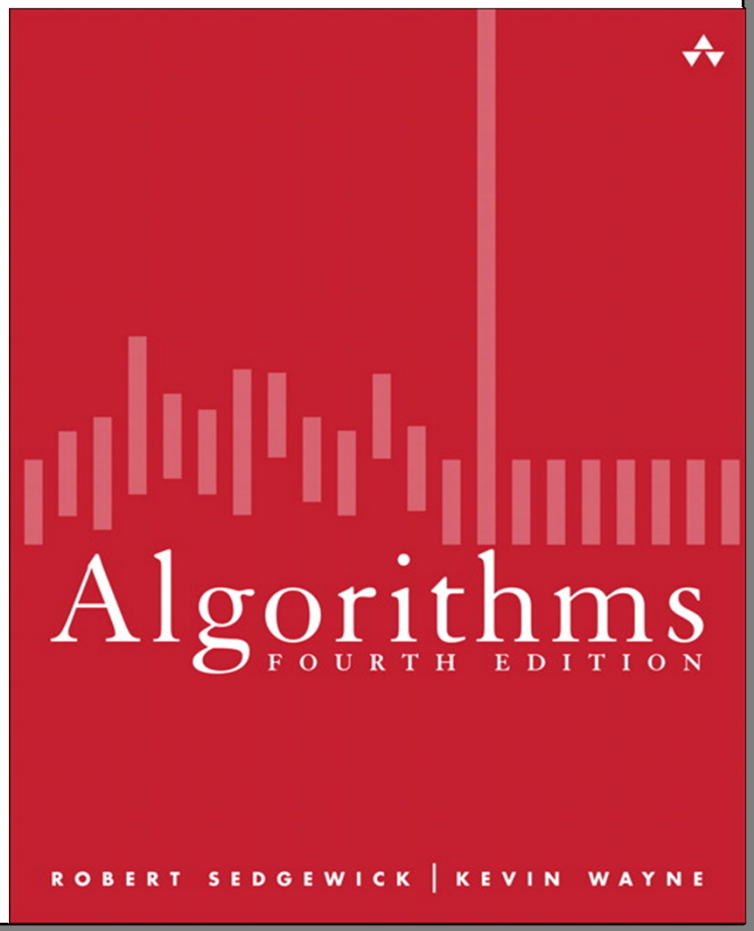
Other questions

Options:

- Chat with me after lecture.
- Talk after recitation (with instructor).
- Schedule an appointment with me.

Textbook: Algorithms

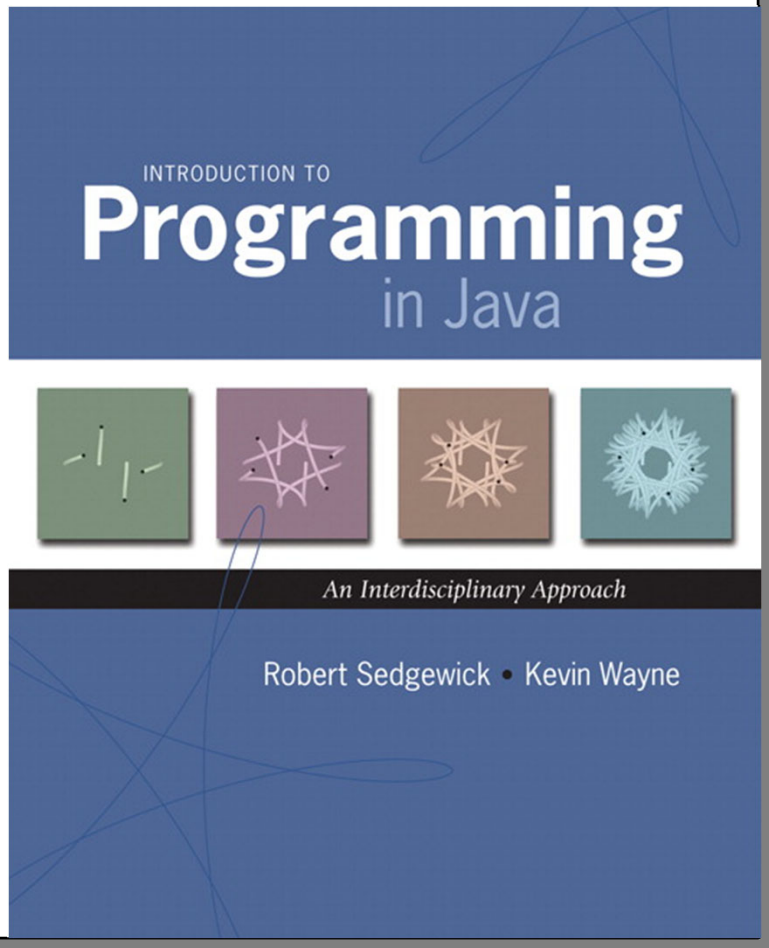
Robert Sedgewick and Kevin Wayne



Optional:

Introduction to Programming in Java

Robert Sedgewick and Kevin Wayne



VISUALGO.NET

Stress?

If stress is getting high:

- Chat with University Counselling Services.
- They have lots of good advice on stress management, organization, how to achieve your goals!

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

ARCHIPELAGO
is open

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:

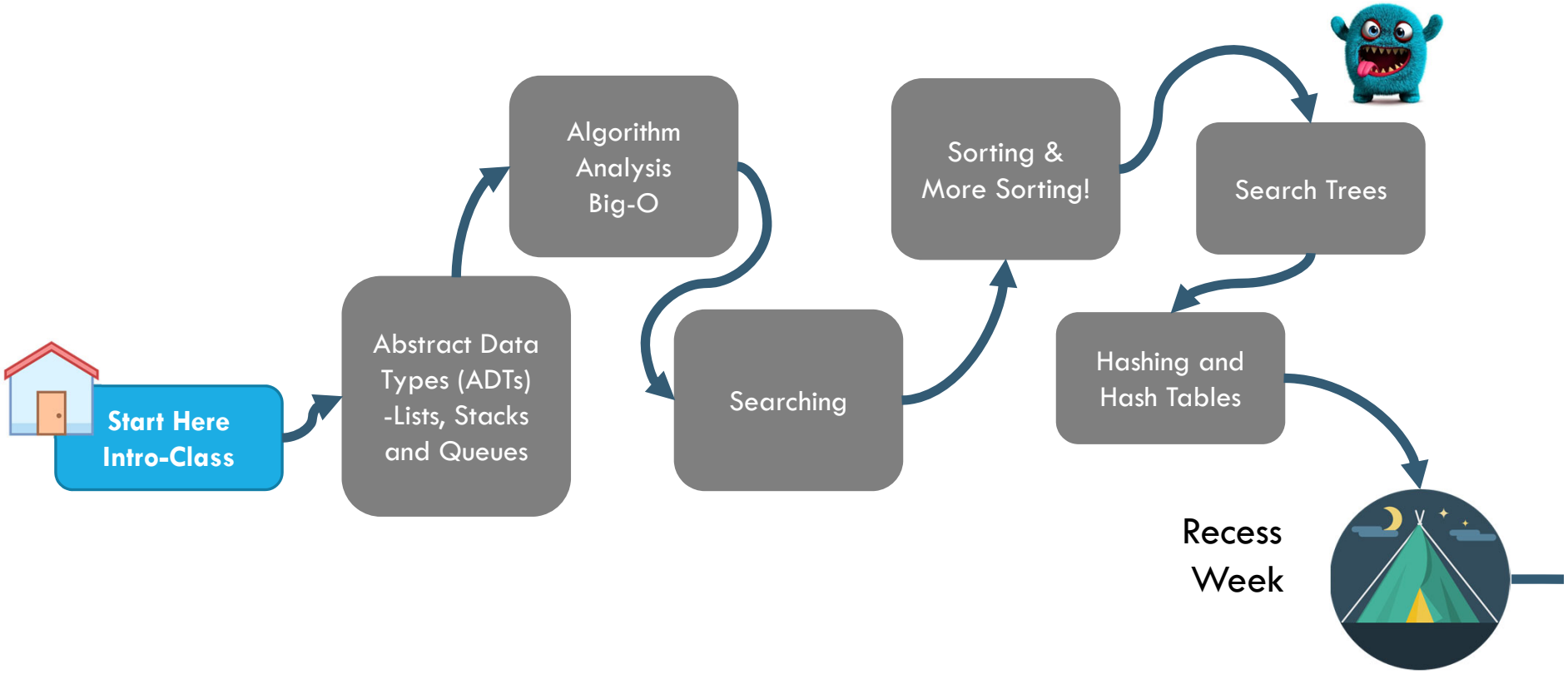
- Alice chooses a die.
- Bob chooses a die.
- Alice and Bob both roll.
- The higher value wins.

Questions:

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

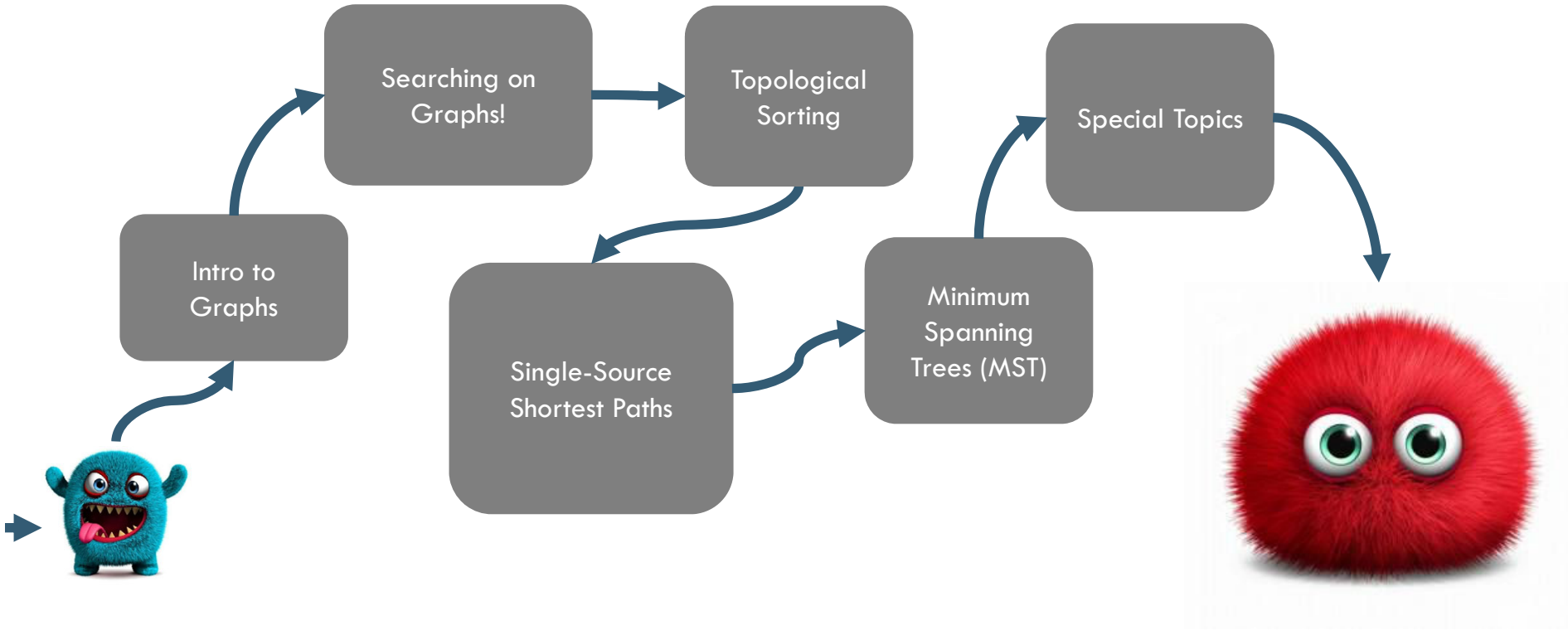
What is CS2040S about?

COURSE STRUCTURE



PART I: ORGANIZING YOUR DATA

COURSE STRUCTURE



PART II: MODELLING AND SOLVING PROBLEMS

Desirable features of your algorithm?

Desirable features of your algorithm:

Desirable features of your algorithm:

How do you choose the right algorithm for the right problem?

How do you design new algorithms for new problems?

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

FRAMEWORK: ALGORITHM & DATA STRUCTURE

What problem does it solve?

How does it work?

How to implement it?

What is its asymptotic performance?

What is its real world performance?

What are the trade-offs?

GOALS

By the end of this course, you should be able to:

- **Apply algorithmic thinking and techniques** for solving computational problems.
- **Describe the structure and operation** of different **data structures and algorithms** under the standard computational model.
- **Assess the suitability** of different data-structures and algorithms for a specific computational problem.
- **Adapt existing data-structures and algorithms** to solve specific computational problems.

A little algorithmic thinking…

Searching

Finding a big item.

Finding many big items.

Searching an array.

Searching faster?

Similarity Test

Comparing two texts

An algorithm

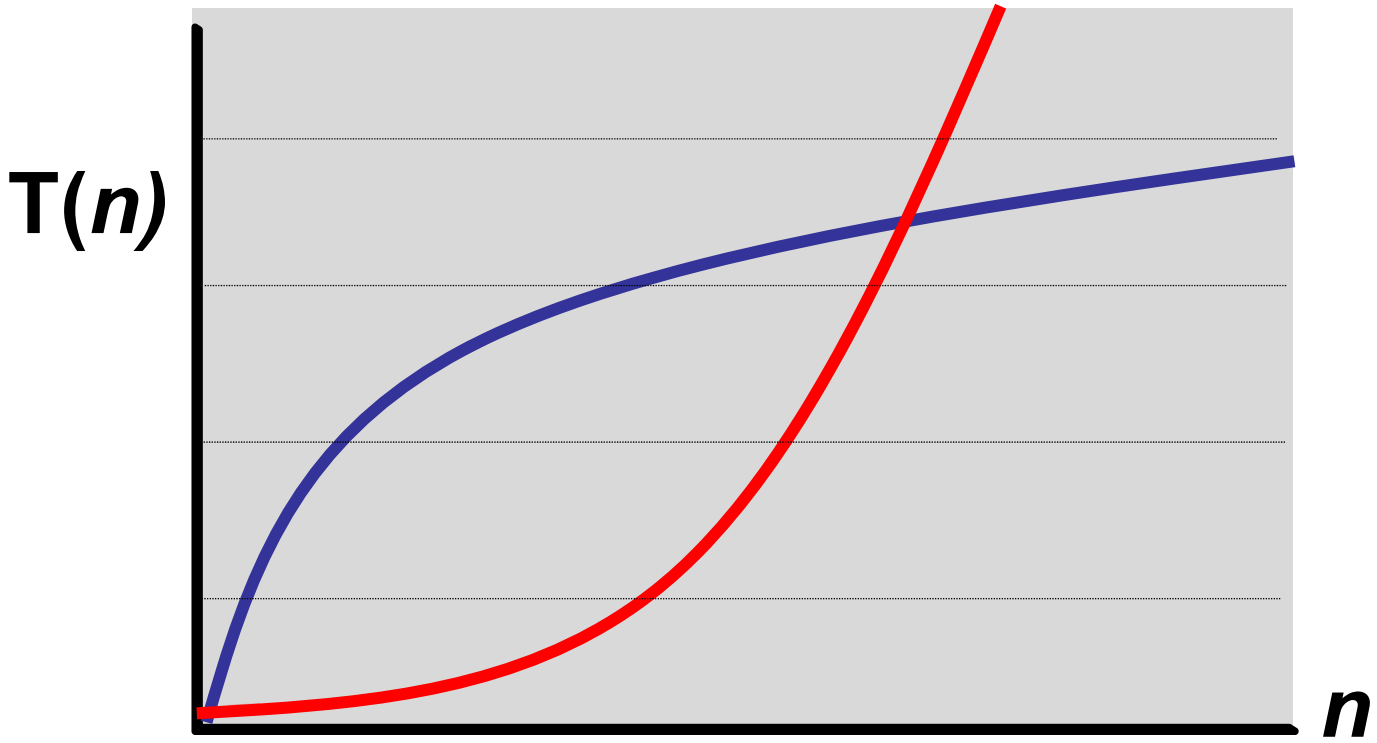
Performance profiling

Fixing a few mistakes

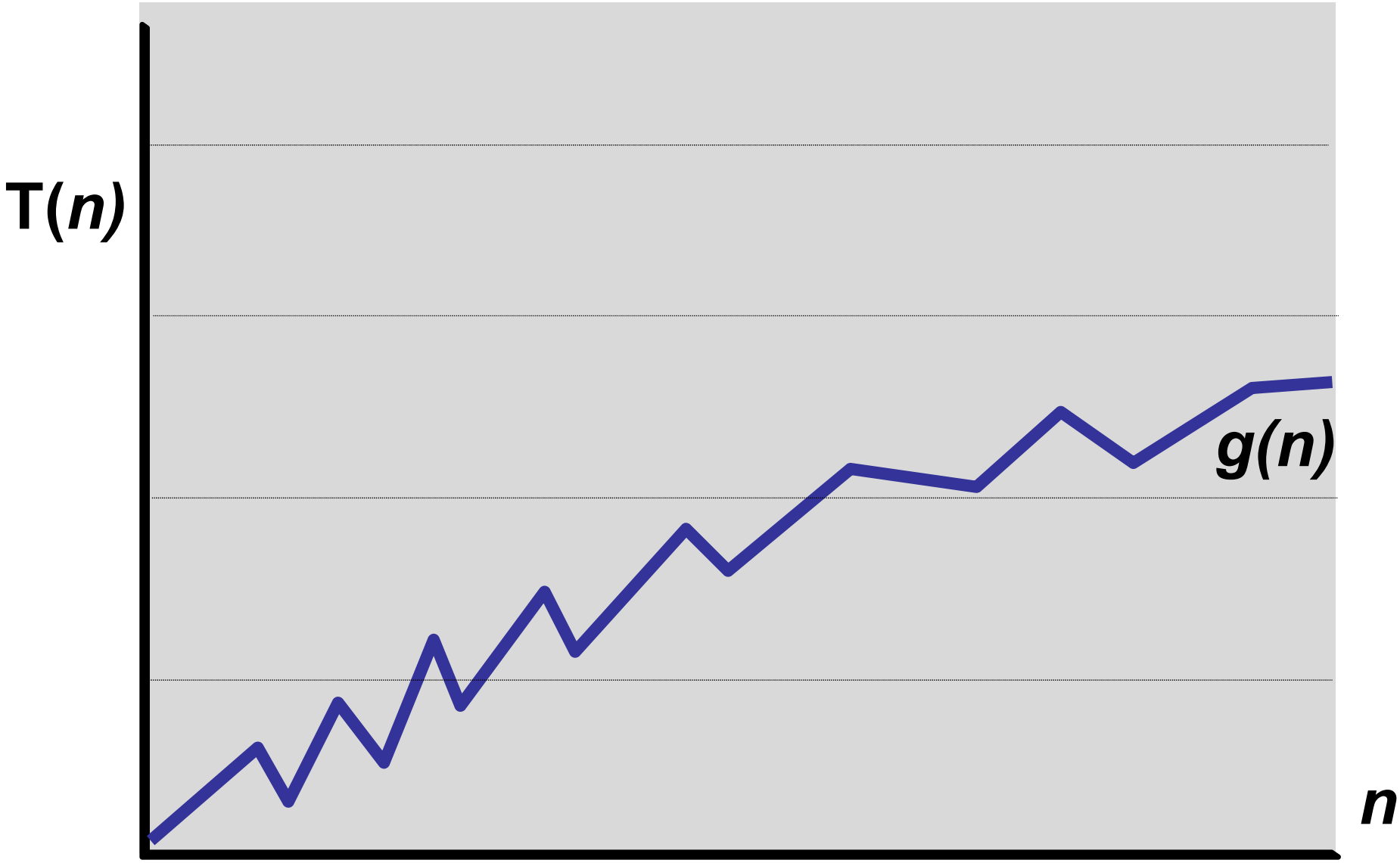
Big-O Notation

How does an algorithm scale?

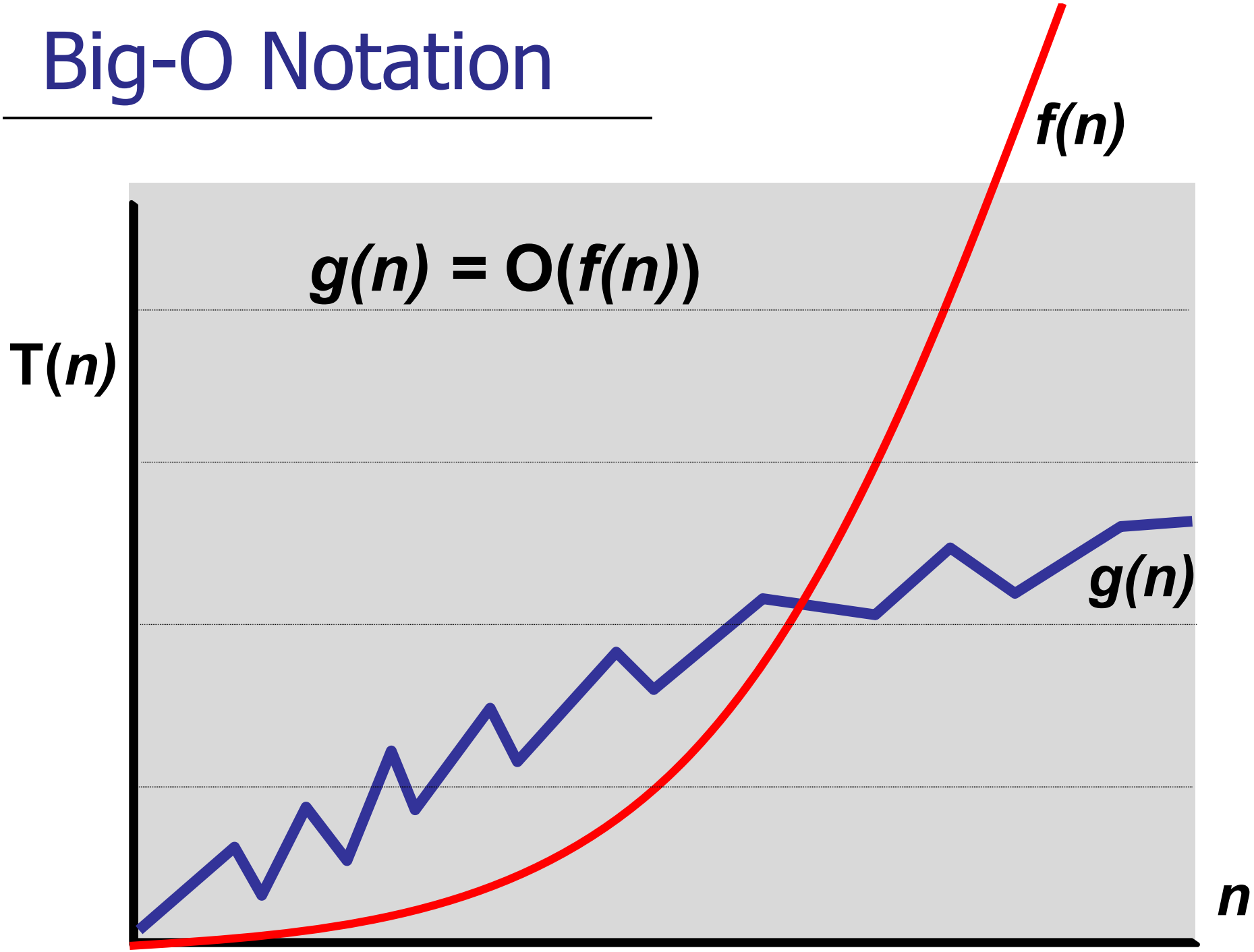
- For large inputs, what is the running time?
- T(n) = running time on inputs of size n



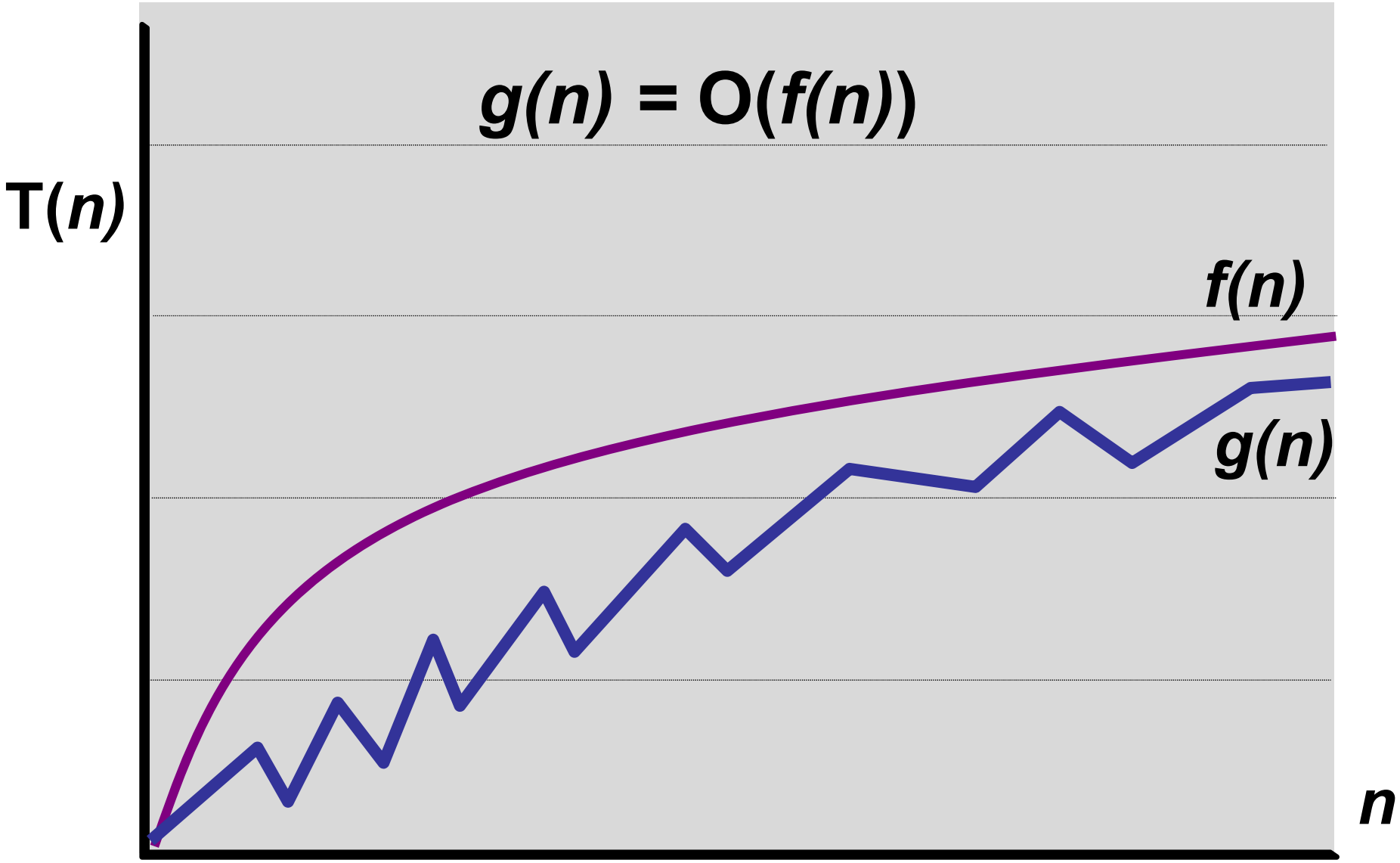
Big-O Notation



Big-O Notation



Big-O Notation



Example

T(n)	f(n)	big-O
$T(n) = 1000n$	$f(n) = n$	$T(n) = O(n)$
$T(n) = 1000n$	$f(n) = n^2$	$T(n) = O(n^2)$
$T(n) = n^2$	$f(n) = n$	$T(n) \neq O(n)$
$T(n) = 13n^2 + n$	$f(n) = n^2$	$T(n) = O(n^2)$

Do asymptotics matter?

Algorithms are more important than language:

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

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

(Source: MIT 6.006)

Simple Problem: Searching

Given:

a collection of n items

Find:

the largest item in the collection

Algorithm 1.

1. Sort the collection.
2. Return the largest item.

Simple Problem: Searching

Given:

a collection of n items

Find:

the largest item in the collection

Algorithm 1.

1. Sort the collection.
2. Return the largest item.

Algorithm 2.

1. Iterate through the collection.
2. Return the largest item.

Simple Problem: Searching

Given:

a collection of n items

Find:

the largest item in the collection

Algorithm 1.

1. Sort the collection.
2. Return the largest item.

Algorithm 2.

1. Iterate through the collection.
2. Return the largest item.

Simple Problem: Searching

Given:

a collection of n items

Find:

the largest item in the collection

Algorithm 1.

1. Sort the collection.
2. Return the largest item.

$O(n \log n)$

Algorithm 2.

1. Iterate through the collection.
2. Return the largest item.

Simple Problem: Searching

Given:

a collection of n items

Find:

the largest item in the collection

Algorithm 1.

1. Sort the collection.
2. Return the largest item.

$O(n \log n)$

Algorithm 2.

1. Iterate through the collection.
2. Return the largest item.

$O(n)$

Simple Problem: Searching

Given:

a collection of n items

Find:

the largest k items

in the collection

Algorithm 1.

1. Sort the collection.
2. Return the largest k items.

Algorithm 2.

1. Iterate through the collection. Store the largest k items seen.
2. Return the largest k items.

Simple Problem: Searching

Given:

a collection of n items

Find:

the largest k items
in the collection

Algorithm 1.

1. Sort the collection.
2. Return the largest k items.

Algorithm 2.

1. Iterate through the collection. Store the largest k items seen.
2. Return the largest k items.

Simple Problem: Searching

Extra memory: 1

Time: $O(n \log n + k)$

Algorithm 1.
1. Sort the collection.
2. Return the largest k items.

Extra memory: k

Time: ??

Algorithm 2.
1. Iterate through the collection. Store the largest k items seen.
2. Return the largest k items.

Simple Problem: Searching

Extra memory: 1

Time: $O(n \log n + k)$

Algorithm 1.
1. Sort the collection.
2. Return the largest k items.

Extra memory: k

Time: ??

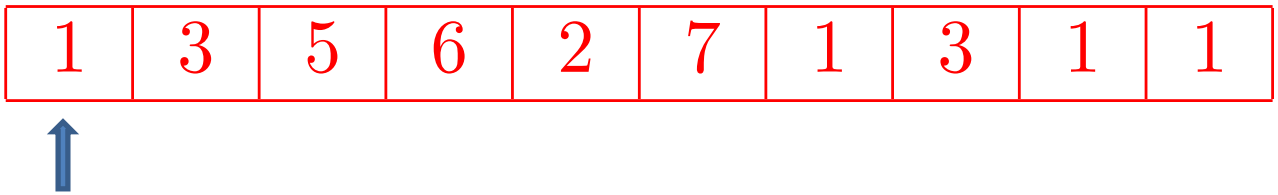
Store k largest items in a sorted array. Insert new big items into the array.

Algorithm 2.
1. Iterate through the collection. Store the largest k items seen.
2. Return the largest k items.

Simple Problem: Searching

Example: k = 3

Collection:



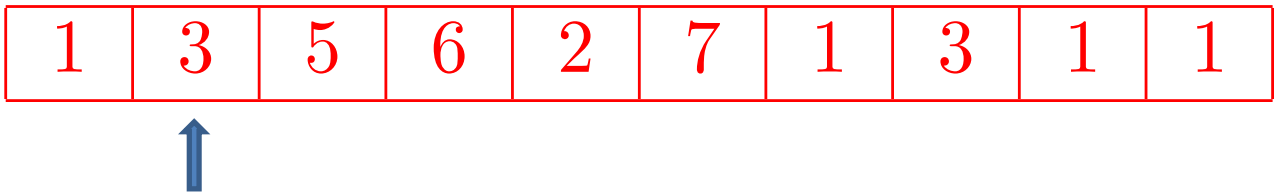
Big items:

Cost of processing first item: 1

Simple Problem: Searching

Example: k = 3

Collection:



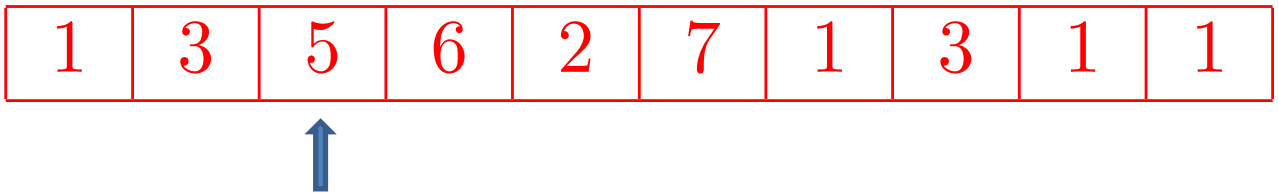
Big items:

Cost of processing second item: 1

Simple Problem: Searching

Example: k = 3

Collection:



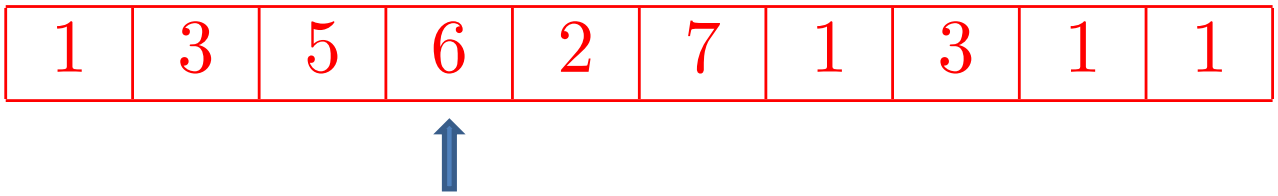
Big items:

Cost of processing third item: 1

Simple Problem: Searching

Example: k = 3

Collection:

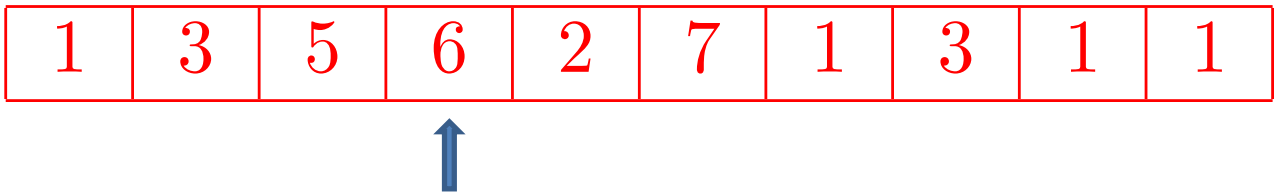


Big items:

Simple Problem: Searching

Example: k = 3

Collection:

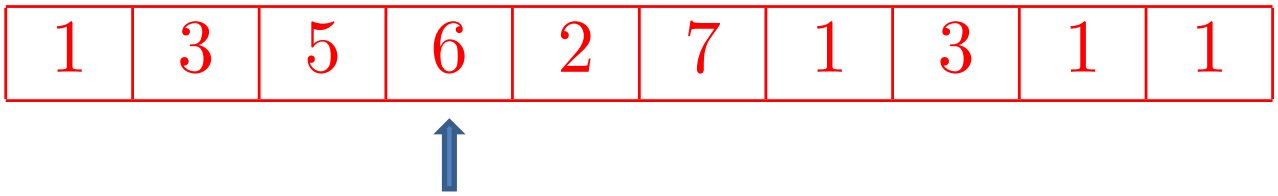


Big items:

Simple Problem: Searching

Example: k = 3

Collection:

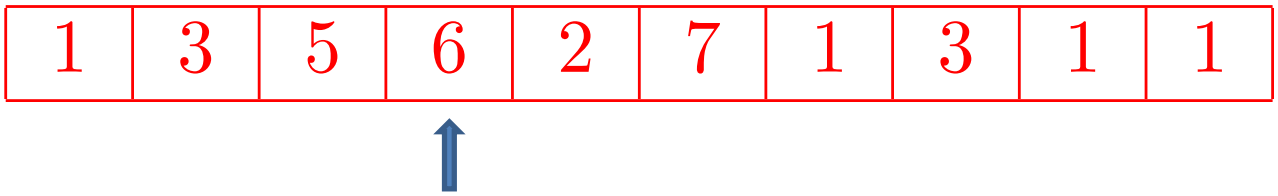


Big items:

Simple Problem: Searching

Example: k = 3

Collection:

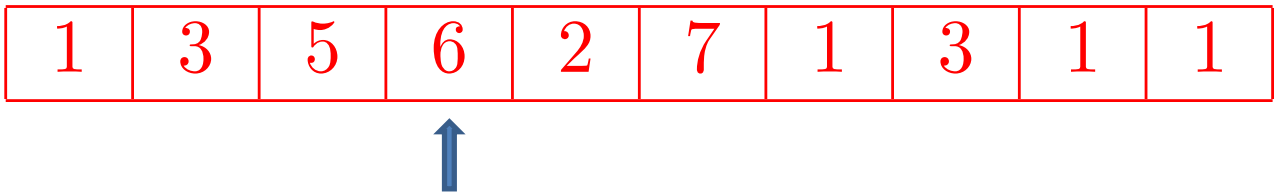


Big items:

Simple Problem: Searching

Example: k = 3

Collection:

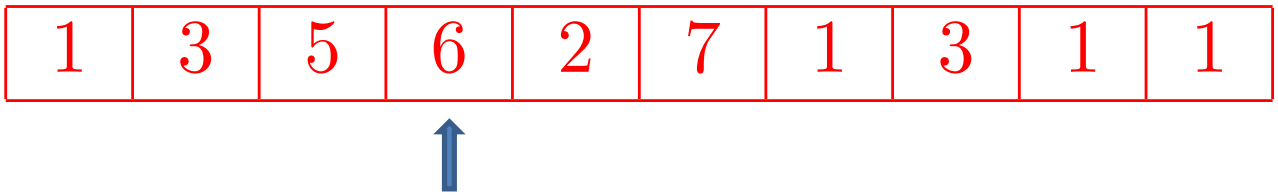


Big items:

Simple Problem: Searching

Example: k = 3

Collection:



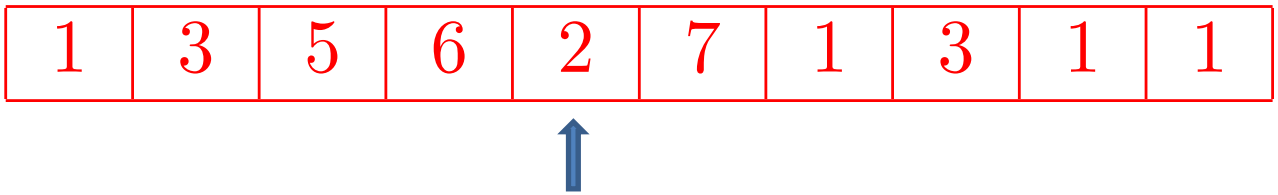
Big items:

Cost of processing fourth item: k

Simple Problem: Searching

Example: k = 3

Collection:



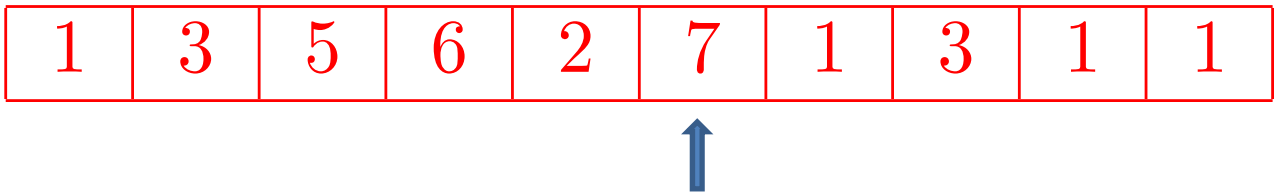
Big items:

Cost of processing fifth item: 0

Simple Problem: Searching

Example: k = 3

Collection:



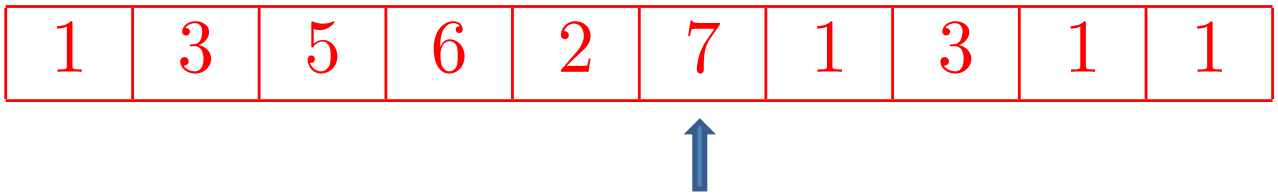
Big items:

Cost of processing: k

Simple Problem: Searching

Example: k = 3

Collection:



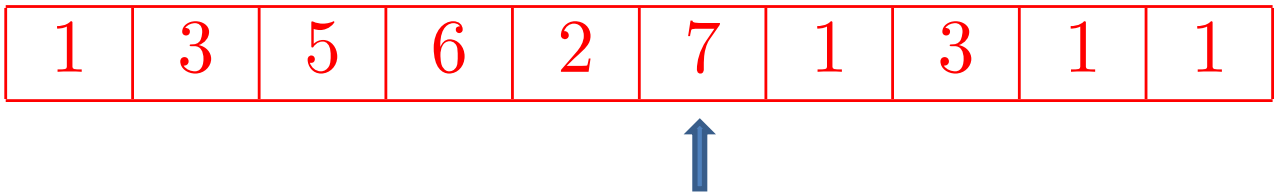
Big items:

Cost of processing: k

Simple Problem: Searching

Example: k = 3

Collection:



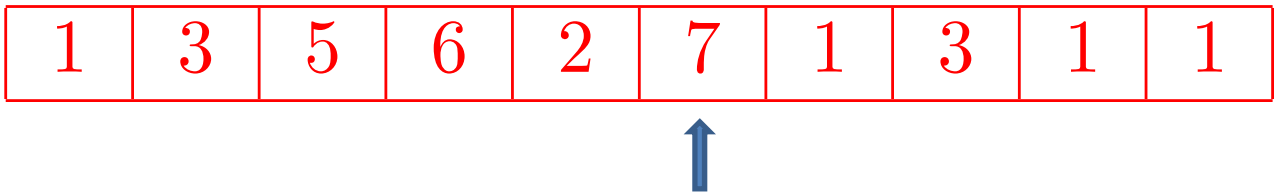
Big items:

What is the worst-case time to process the entire collection?

Simple Problem: Searching

Example: k = 3

Collection:



Big items:

Worst case cost to process collection: $O(nk)$

Simple Problem: Searching

Extra memory: 1

Time: $O(n \log n + k)$

Algorithm 1.
1. Sort the collection.
2. Return the largest k items.

Extra memory: k

Time: $O(nk)$

Algorithm 2.
1. Iterate through the collection. Store the largest k items seen.
2. Return the largest k items.

Simple Problem: Searching

Extra memory: 1

Time: $O(n \log n + k)$

Algorithm 1.
1. Sort the collection.
2. Return the largest k items.

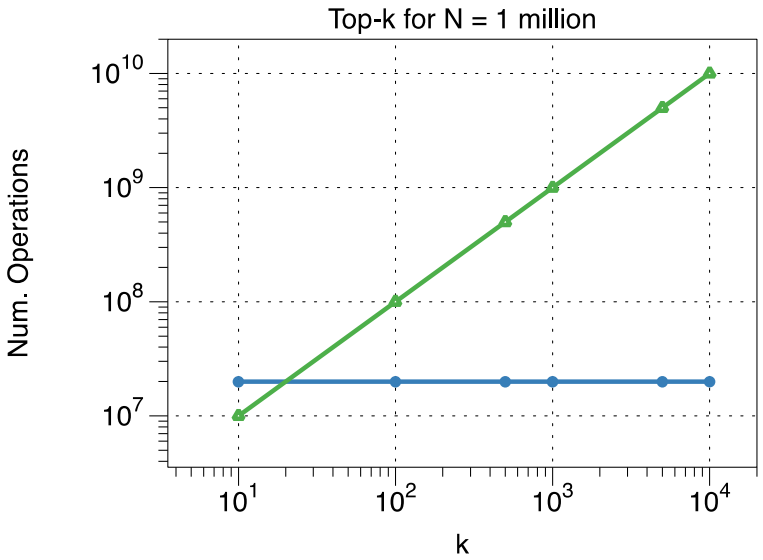
Extra memory: k

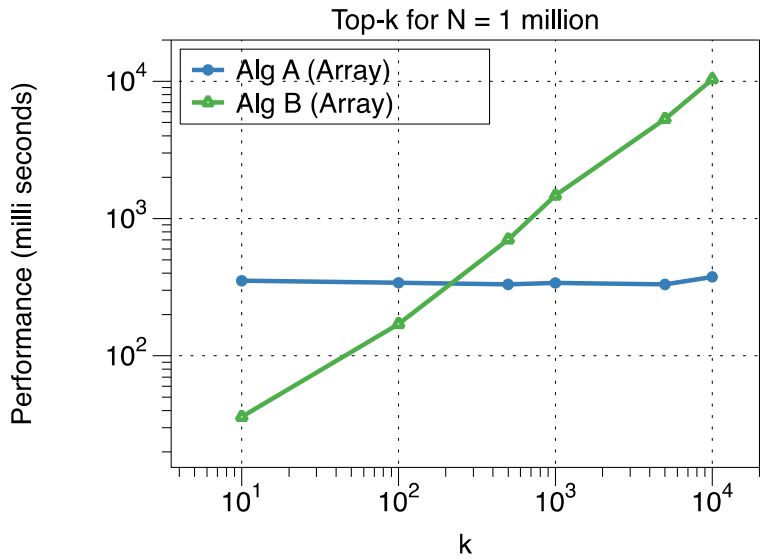
Time: $O(nk)$

Algorithm 2.
1. Iterate through the collection. Store the largest k items seen.
2. Return the largest k items.

Better solution in Week 5.

Theoretical vs Real-World Performance





(Hat tip: Harold Soh)

More Searching

Given:

a sorted array of n items

Find:

find smallest item at least as big as value x?

Algorithm 1.
Linear search

More Searching

Given:

a sorted array of n items

Find:

find smallest item at least as big as value x?

Algorithm 1.
Linear search

Algorithm 2.
Binary search

Week 2:
Check midpoint and recurse into one of the two halves.

More Searching

Given:

a sorted array of n items

Find:

find smallest item at least as big as value x?

Algorithm 1.
Linear search

$O(n)$

Algorithm 2.
Binary search

$O(\log n)$

More Searching

Are we done?

Is binary search always the best searching algorithm on real machines?

Yes No

Algorithm 1.
Linear search

$O(n)$

Algorithm 2.
Binary search

$O(\log n)$

More Searching

Are we done?

Information theory says that this problem requires at least $\Omega(\log n)$ ops.

Algorithm 1.
Linear search

$O(n)$

Algorithm 2.
Binary search

$O(\log n)$

Predicting Performance

Example: 100 TB of data

1) Store data sorted in an array

⇒ Scan all the data: O(n)

⇒ (Binary) search: O(log n)

2) Store data in a linked list

⇒ Scan all the data: O(n)

⇒ Search: O(n)

3) Store data in a balanced tree

⇒ Scan all the data: O(n)

⇒ Search: O(log n)

Predicting Performance

Example: 100 TB of data

1) Store data sorted in an array

⇒ Scan all the data: O(n)

⇒ (Binary) search: O(log n)

2) Store data in a linked list

⇒ Scan all the data: O(n)

⇒ Search: O(n)

3) Store data in a balanced tree

⇒ Scan all the data: O(n)

⇒ Search: O(log n)

Which is fastest?

Predicting Performance

Example: 100 TB of data

1) Store data sorted in an array

⇒ Scan all the data: O(n)

⇒ (Binary) search: O(log n)

2) Store data in a linked list

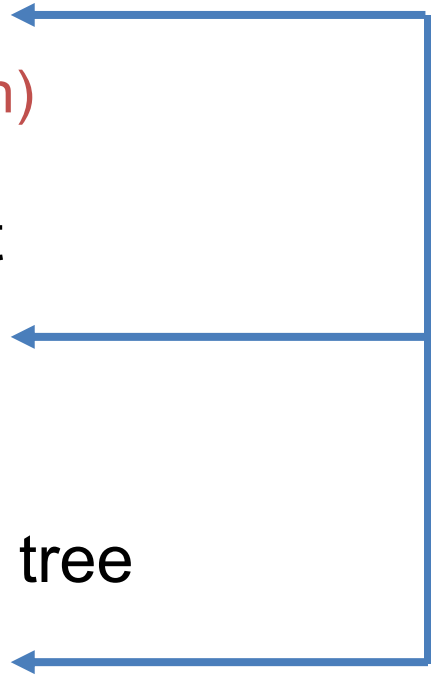
⇒ Scan all the data: O(n)

⇒ Search: O(n)

3) Store data in a balanced tree

⇒ Scan all the data: O(n)

⇒ Search: O(log n)



Same theoretical performance!

Predicting Performance

Example: 100 TB of data

1) Store data sorted in an array

⇒ Scan all the data: O(n)

⇒ (Binary) search: O(log n)

This is MUCH faster in practice by an order of magnitude.

2) Store data in a linked list

⇒ Scan all the data: O(n)

⇒ Search: O(n)

3) Store data in a red-black tree

⇒ Scan all the data: O(n)

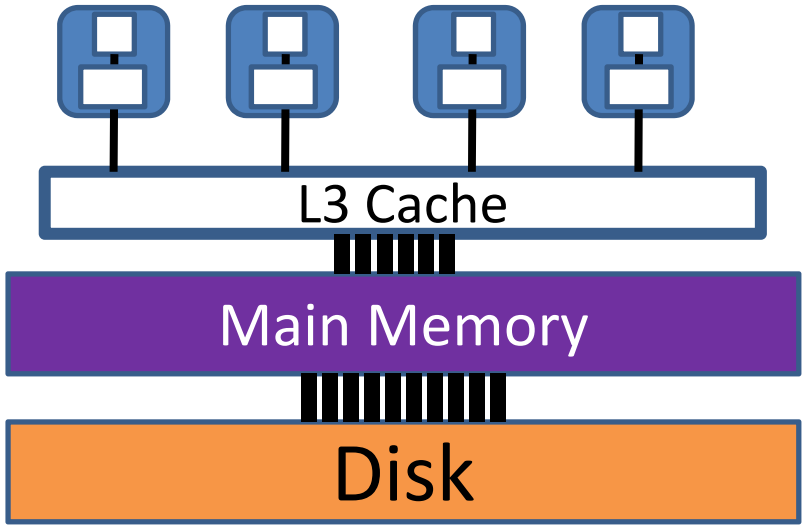
⇒ Search: O(log n)

Haswell Architecture (2-18 cores)

Memory Type	size	line size	clock cycles
L1 cache	64 KB	64 B	~4
L2 cache	256 KB	64 B	~10
L3 cache	2-40 MB	64 B	40-74
L4 (optional)	128 MB		
Main Memory	< 128 GB	16 KB	~200-350
SSD Disk	BIG	Variable (e.g., 16KB)	~20,000
Disk	BIGGER	Variable (e.g., 16KB)	~20,000,000

Notes:

- Several other ”caches” e.g., TLB, micro-op cache, instruction cache, etc.
- L1/L2 caches are per core.
- L3/L4 cache are shared per socket



More Searching

Faster to keep your data stored in a B-tree than in a sorted array!

Array: $O(\log(n/B))$

B-tree: $O(\log_B n)$

B is a parameter of the cache.

Algorithm 1.
Linear search

Algorithm 2.
Binary search

Algorithm 3.
Store data in B-tree

A little algorithmic thinking…

Searching

Finding a big item.

Finding many big items.

Searching an array.

Searching faster?

Similarity Test

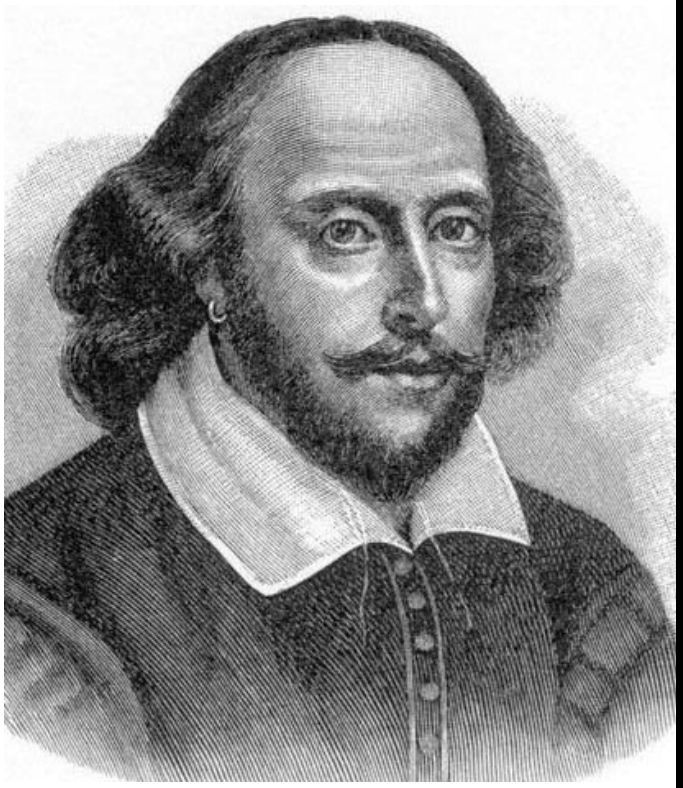
Comparing two texts

An algorithm

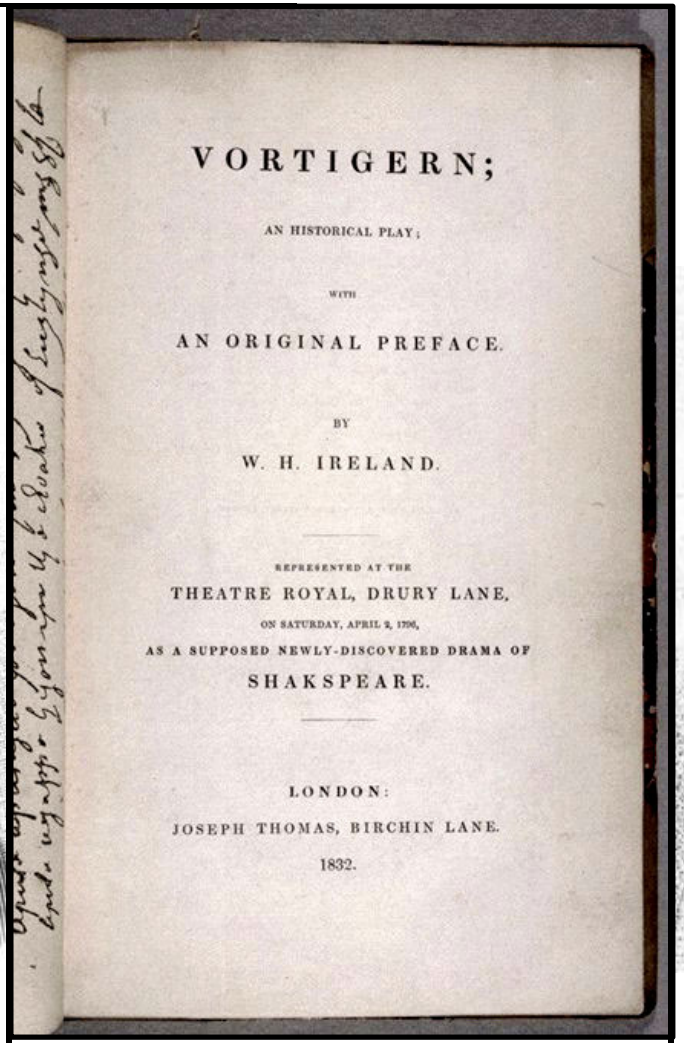
Performance profiling

Fixing a few mistakes

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??

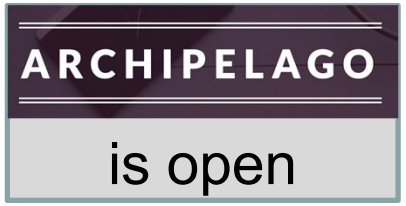
Document distance

- 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

What are good ways to quantify similarity?



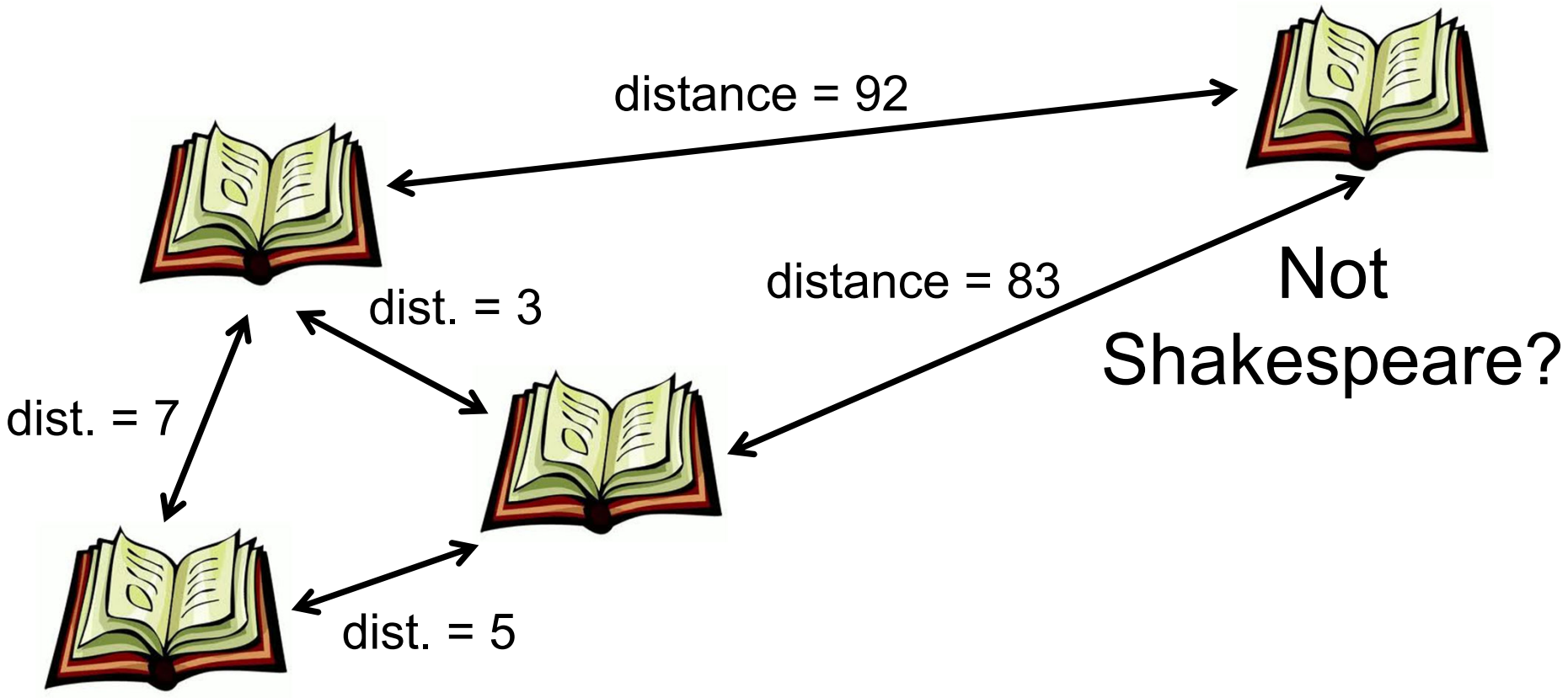
Metrics of similarity

- 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?

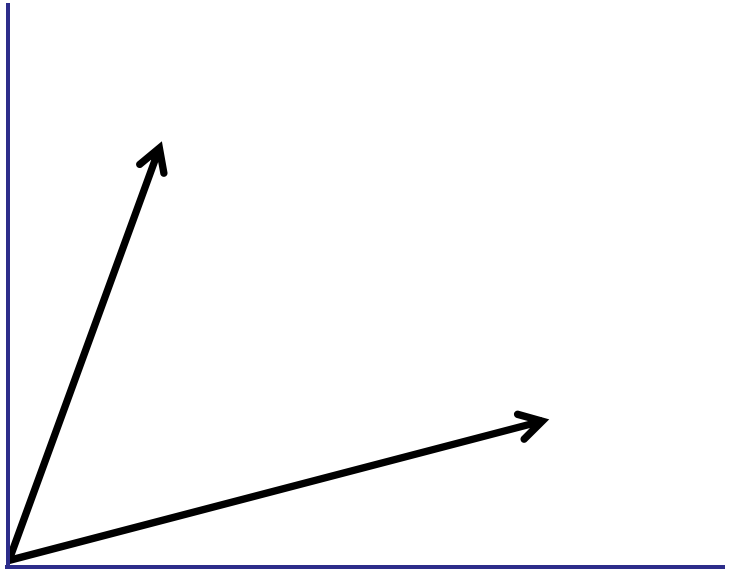


Vector Space Model

Strategy:

- View each document as a high-dimensional vector.

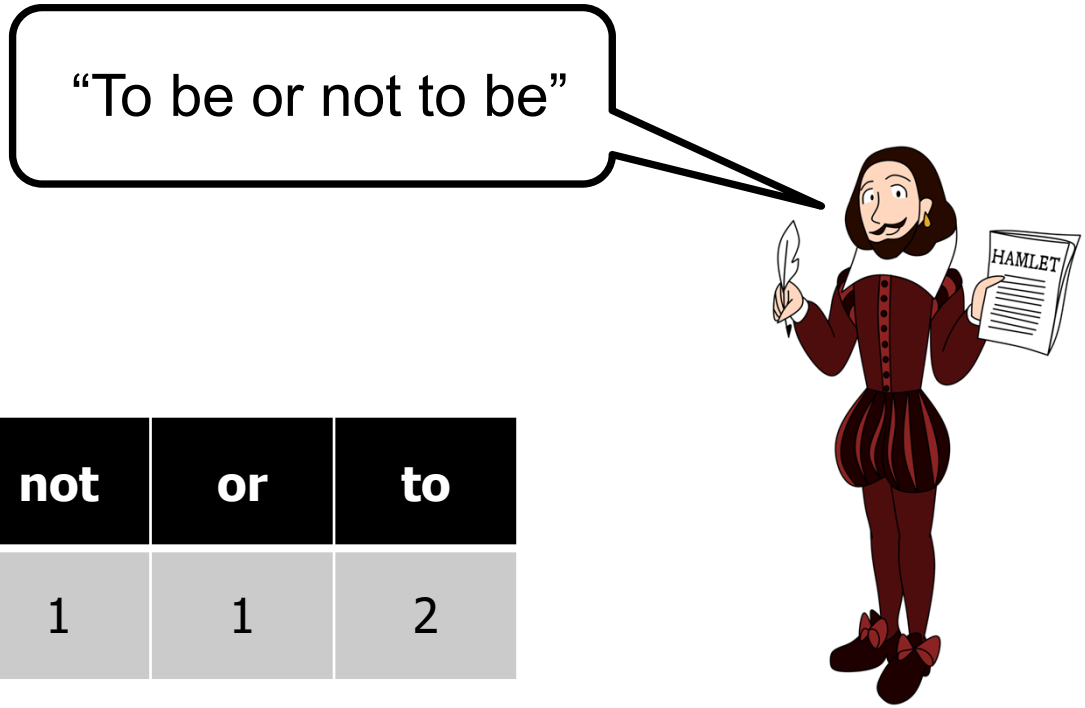
[Salton, Wang, Yang '75]



Vector Space Model

Strategy:

Each dimension is # times each word appears.



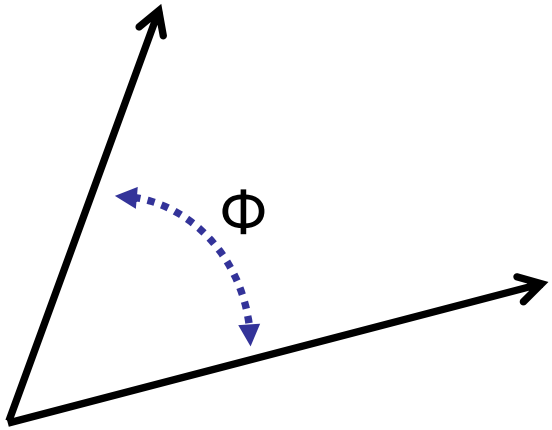
4d-vector:

be	not	or	to
2	1	1	2

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

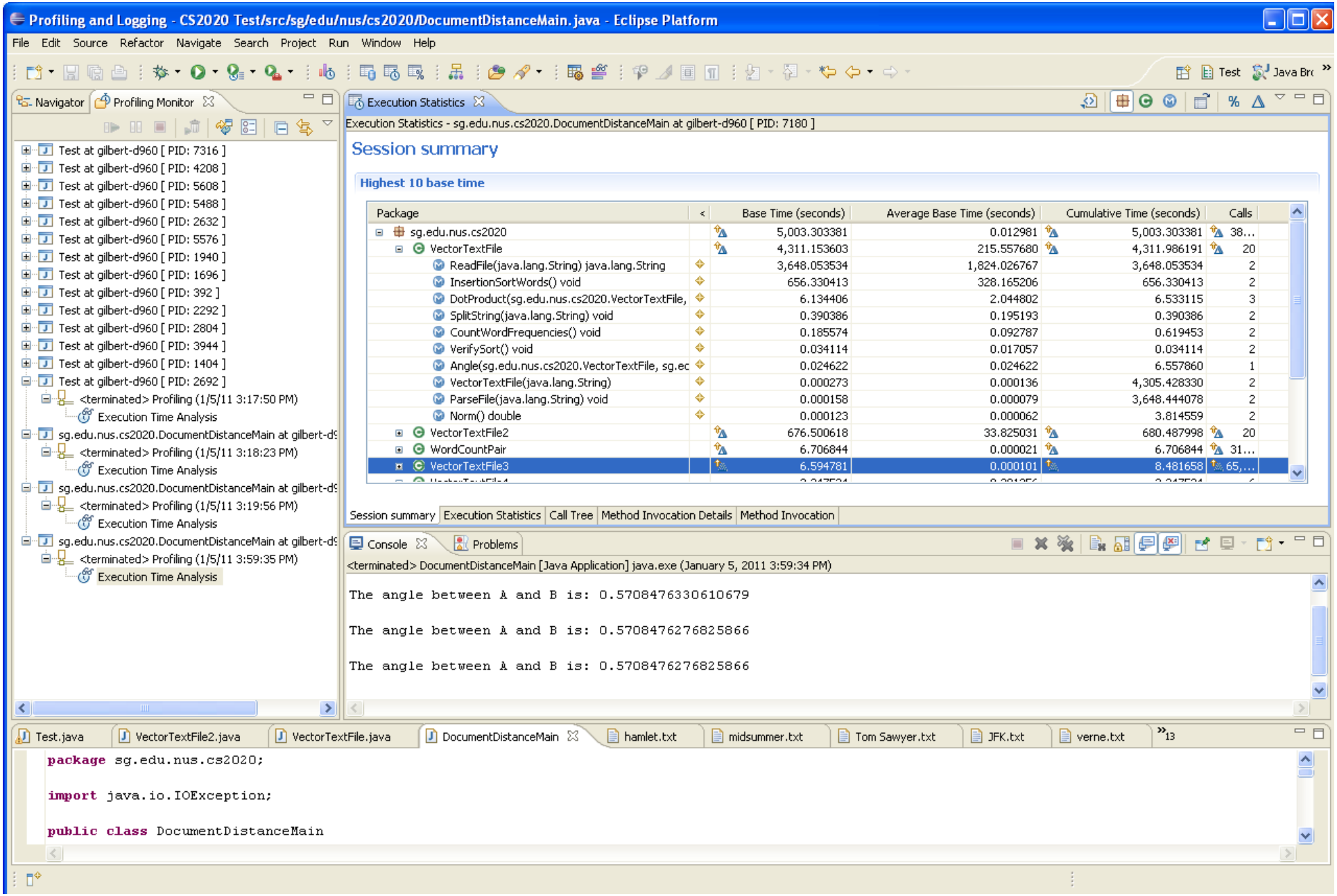
1. Create vectors v_A and v_B
   - ➔ count number of times each word appears
2. Calculate vector norms
3. Calculate dot product: $(v_A \cdot v_B)$
4. Calculate angle $\Phi(v_A, v_B)$

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**

Profiler


Profiling and Logging - CS2020 Test/src/sg/edu/nus/cs2020/DocumentDistanceMain.java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
Navigator
Profiling Monitor
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-d
<terminated> Profiling (1/5/11 3:18:23 PM)
Execution Time Analysis
sg.edu.nus.cs2020.DocumentDistanceMain at gilbert-d
<terminated> Profiling (1/5/11 3:19:56 PM)
Execution Time Analysis
sg.edu.nus.cs2020.DocumentDistanceMain at gilbert-d
<terminated> Profiling (1/5/11 3:59:35 PM)
Execution Time Analysis
Execution Statistics
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,...
Session summary | Execution Statistics | Call Tree | Method Invocation Details | Method Invocation
Console
Problems
<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
Test.java
VectorTextFile2.java
VectorTextFile.java
DocumentDistanceMain
hamlet.txt
midsummer.txt
Tom Sawyer.txt
JFK.txt
verne.txt
package sg.edu.nus.cs2020;
import java.io.IOException;
public class DocumentDistanceMain

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**

Problem 1: Why does it take SO long to read the file?

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:

```
textFile = textFile + c
```

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

String Problem!

What happens when:

textFile = textFile + c

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

Copying a string of k characters takes time k!

String Problem!

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

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**

Problem 2: Can we sort faster?

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

Use $O(n \log n)$ sorting algorithm (MergeSort) instead of $O(n^2)$ sorting algorithm (InsertionSort).

Problem 3: Do we need to sort at all?

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

Use O(n) hashing approach

instead of O(n log n) sorting algorithm (MergeSort).

Goals for the Semester

Speed up your code by a factor of 2040!

Algorithms:

- Design of efficient algorithms
- Analysis of algorithms

Implementation:

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

For next time…

Thursday lecture:

- Java introduction, OOP
- Stacks, Queues, Lists

Lecture 1 Training: Released. Due next week.

Problem Set 1: Released. Due Friday.

Important: Deadline ~~Thursday~~ Wednesday 11:59pm.

- Fill out recitation survey (Coursemology)
- Fill out tutorial survey (Coursemology)