Margaret Hamilton

standing next to listings of the actual Apollo Guidance Computer (AGC) source code

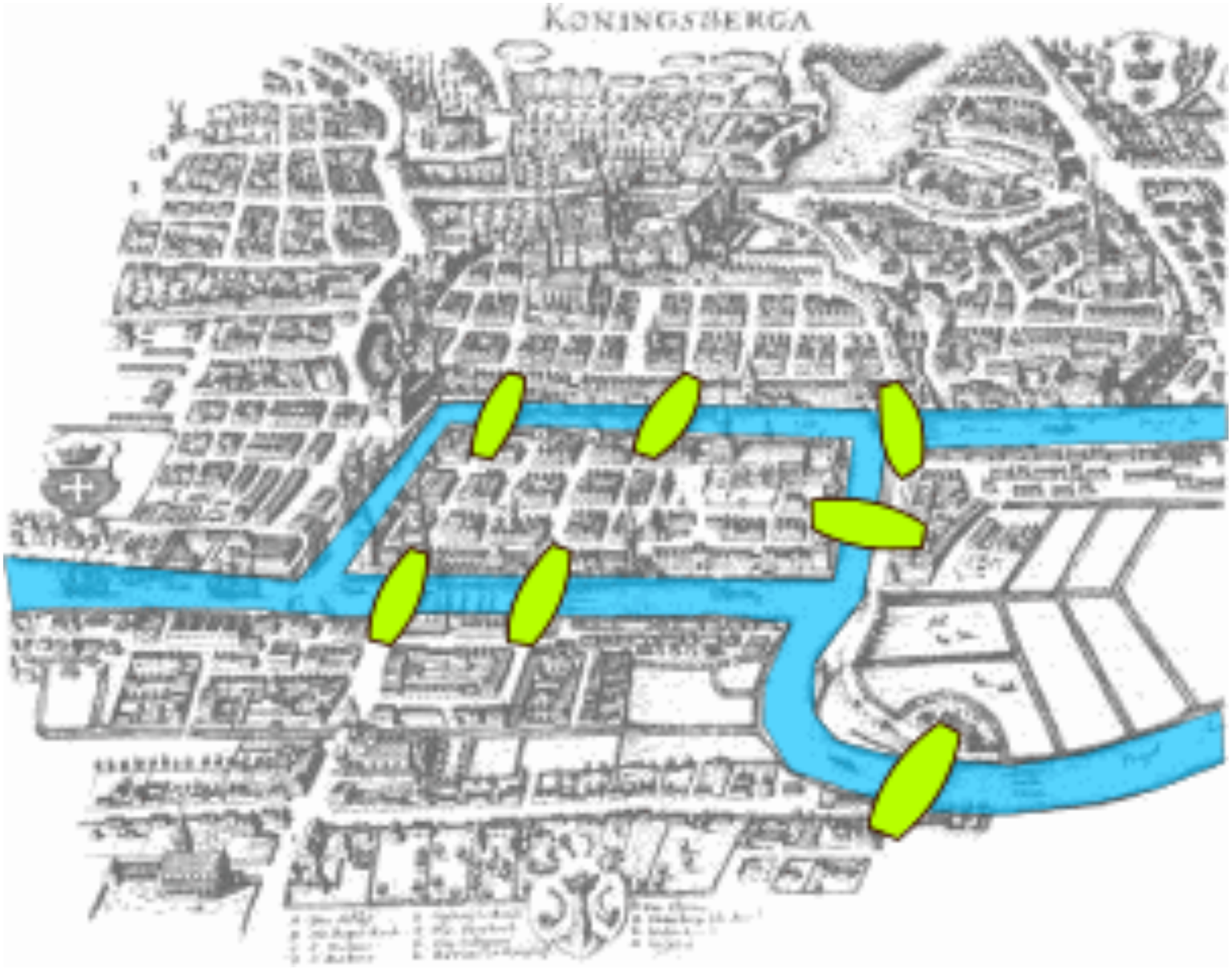
Prepared by Julian García, based on material by
David Albrecht and María García de la Banda

Lecture 1
Introduction

FIT 1008
Introduction to Computer Science

Computer Science is **NOT** just about **Programming**

- Hardware
- Computer Systems Organisation
- Software and Data
- Theory of Computation
- Mathematics of Computing
- Analysis of computing methods
- Information Systems
- Computing Applications
- Computing Culture


KONINGSBERGA

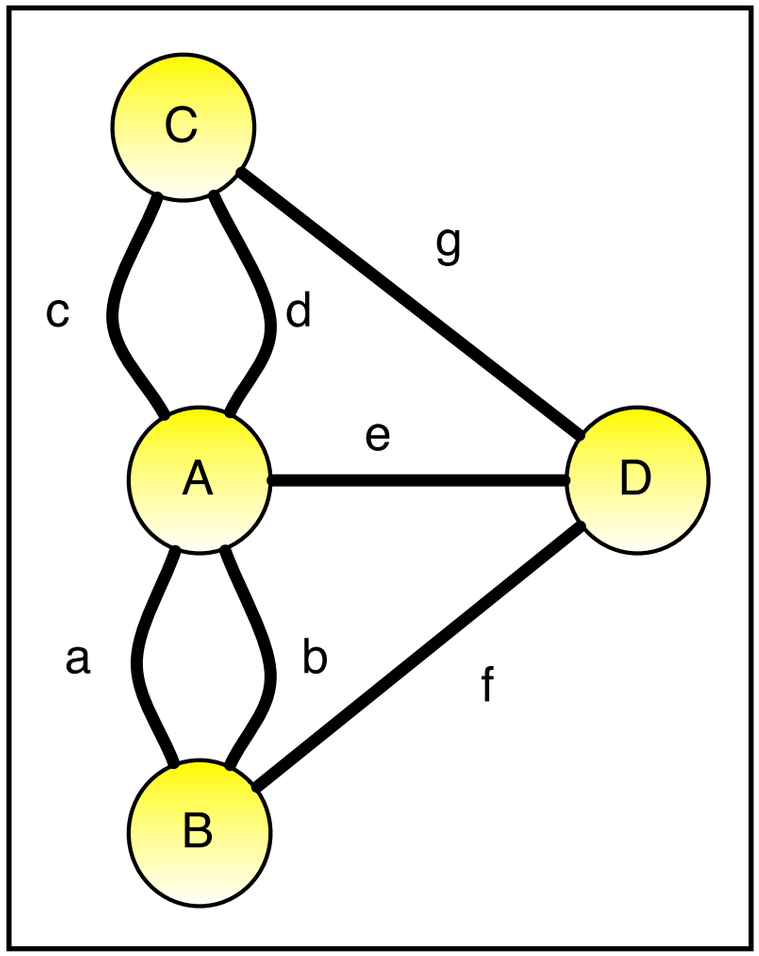
We put the vertices in a list…

$$V = [A, B, C, D]$$
$$\quad\ 0\ \ 1\ \ 2\ \ 3$$

Eulerian(V[0, n-1])

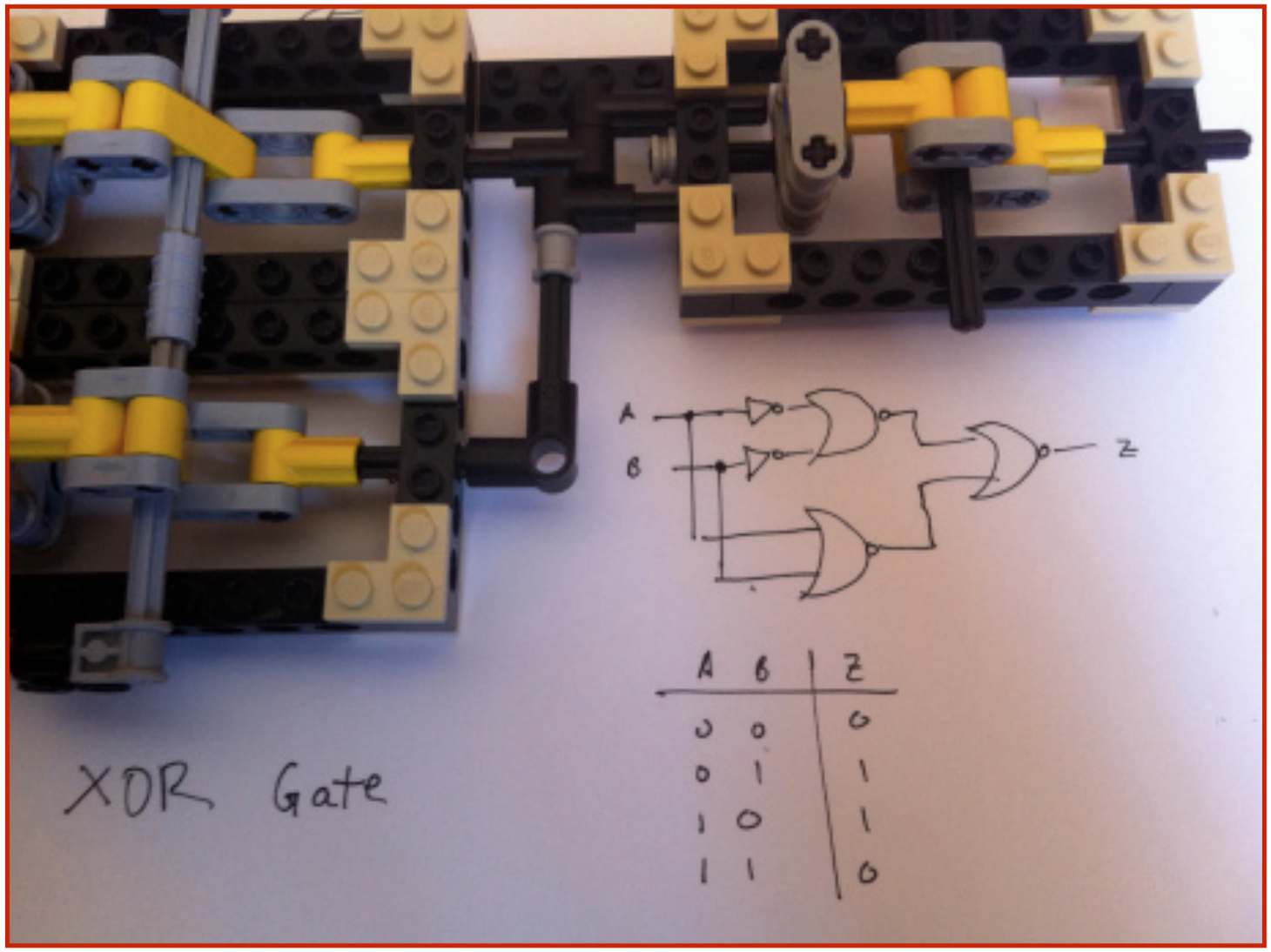
Input: A list of *n* vertices.

Output: **True** if the graph is Eulerian, **False** otherwise

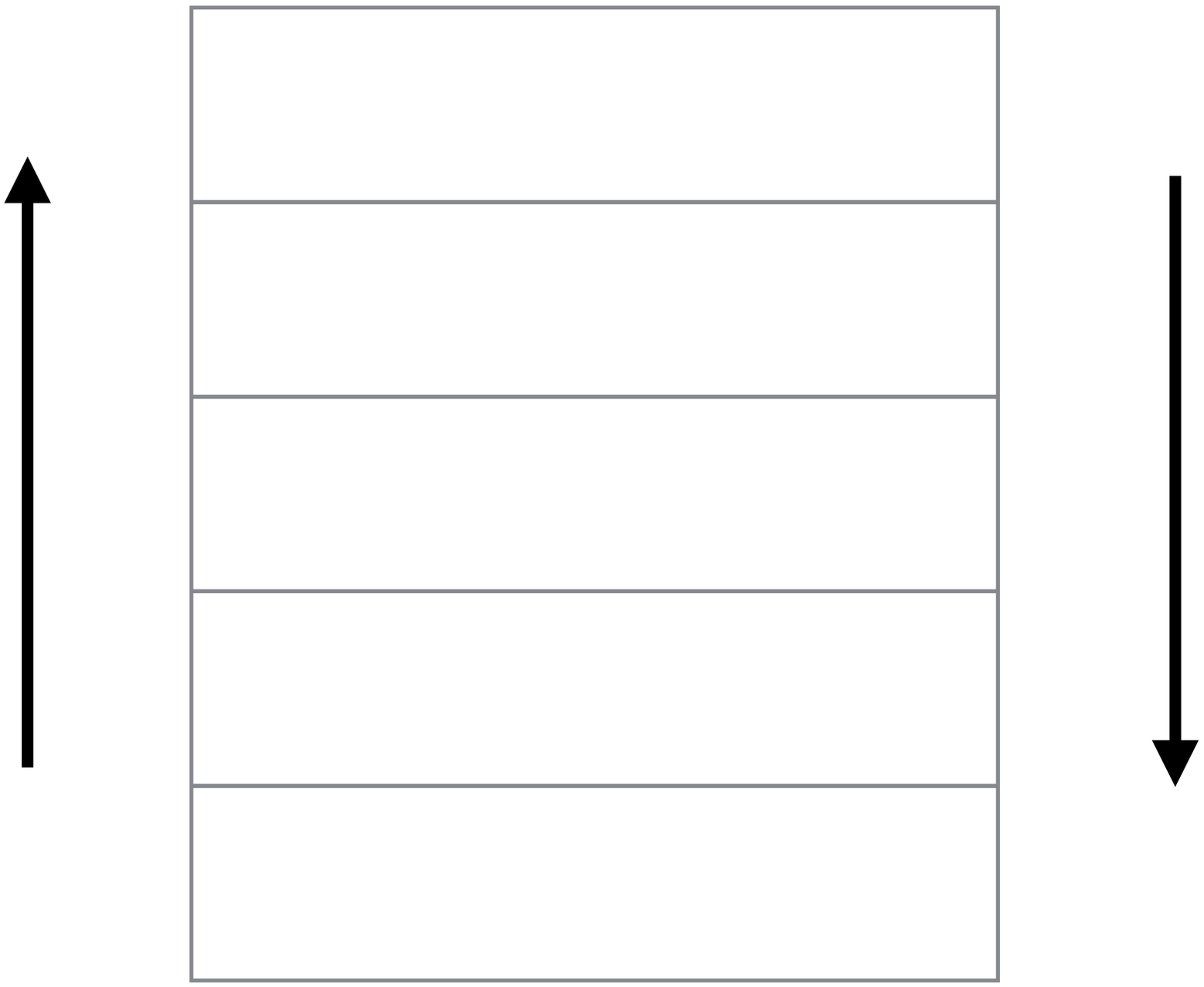



```
v←0
while(v < n)
{
    if( degree(L[v]) is odd)
    {
        return FALSE
    }
    v←v+1
}
return TRUE
```

"all computation done by large combinations of **on-and-off** switches, wired together in meaningful ways"



https://keshavsaharia.files.wordpress.com/2011/05/img_0270.jpg?w=460&h=343



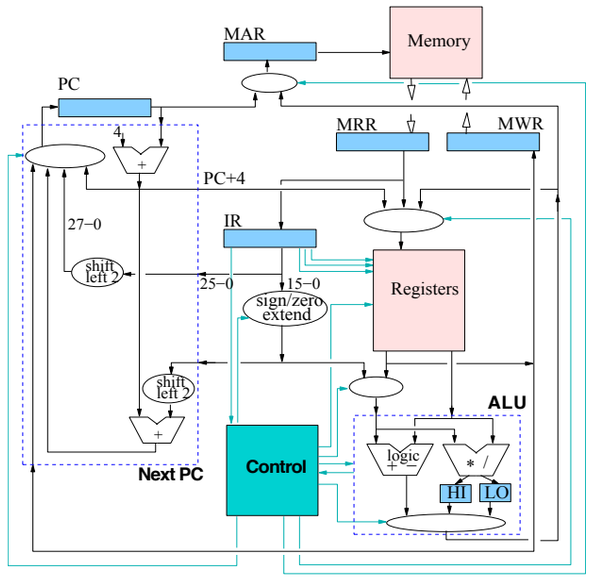
You

???

```
f = 1
n = int(input("Enter int:"))
while n > 0:
    f = f*n
    n = n-1
print(f)
```

Human-readable code

Machine language



CPU

How Do Computers Work? [closed]

59

104

This is almost embarrassing ask...I have a degree in Computer Science (and a second one in progress). I've worked as a full-time .NET Developer for nearly five years. I generally seem competent at what I do.

But I Don't Know How Computers Work!

Please, bare with me for a second. A quick Google of 'How a Computer Works' will yield lots and lots of results, but I struggled to find one that really answered what I'm looking for. I realize this is a huge, huge question, so really, if you can just give me some keywords or some direction.

I know there are components....the power supply, the motherboard, ram, CPU, etc...and I get the 'general idea' of what they do. But I really don't understand how you go from a line of code like `Console.Readline()` in .NET (or Java or C++) and have it actually *do* stuff.

Sure, I'm vaguely aware of MSIL (in the case of .NET), and that some magic happens with the JIT compiler and it turns into native code (I think). I'm told Java is similar, and C++ cuts out the middle step.

I've done some mainframe assembly, it was a few years back now. I remember there were some instructions and some CPU registers, and I wrote code... and then some magic happened... and my program would work (or crash). From what I understand, an Emulator would simulate what happens when you call an instruction and it would update the CPU registers; but what makes those instructions work the way they do?

Does this turn into an Electronics question and not a 'Computer' question? I'm guessing there isn't any practical reason for me to understand this, but I feel like I should be able to.

(Yes, this is what happens when you spend a day with a small child. It takes them about 10 minutes and five iterations of asking 'Why?' for you to realize how much you don't know)

hardware assembly

share

asked Jun 5 '11 at 4:35

Rob P.

584 ● 2 ● 6 ● 10

http://programmers.stackexchange.com/questions/81624/how-do-computers-work

High level programming language

Assembly Language Program

Machine language

```
def find_duplicates(a_list):
    n = len(a_list)
    k = 0
    while k < n:
        j = k + 1
        while j < n:
            if a_list[k] == a_list[j]:
                print(a_list[k])
            j += 1
        k += 1
```

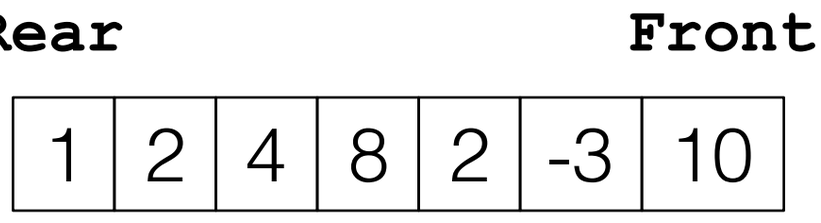
```
main:   # 1 * 4 = 4 bytes local.

        addi $fp, $sp, 0

        addi $sp, $sp, -4

        sw   $0, -4($fp) # n = 0

        addi $v0, $0, 5

        syscall

        sw $v0, -4($fp)  # n
```

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```

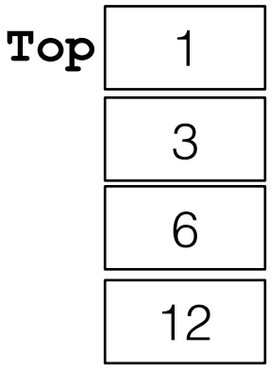
Programs = Algorithms + Data Structures

(Programming is still important)

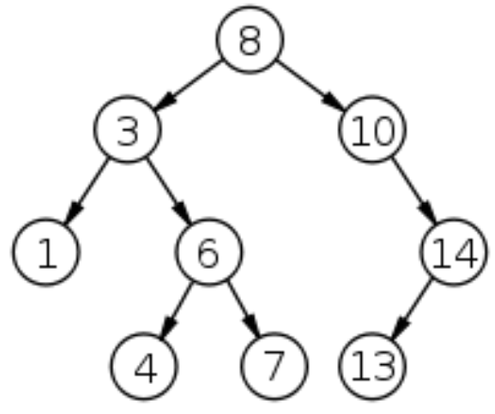
Data Structures



Queues

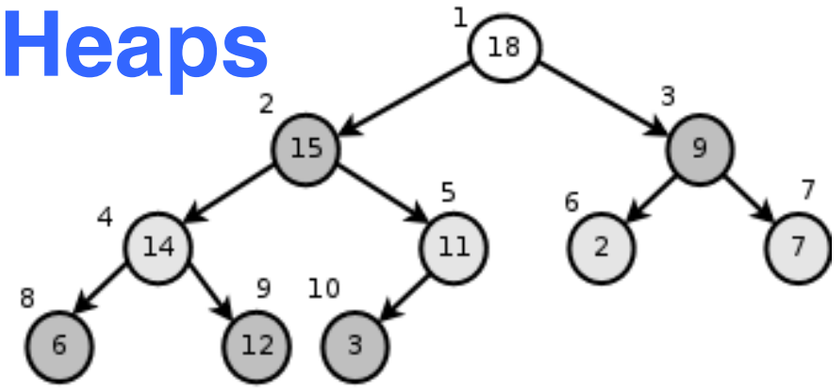


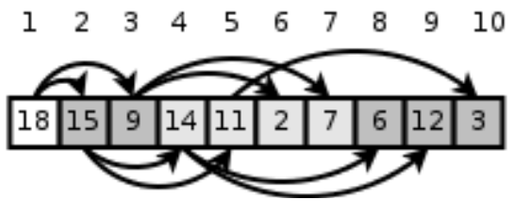
Stacks



Binary Search Trees

Heaps

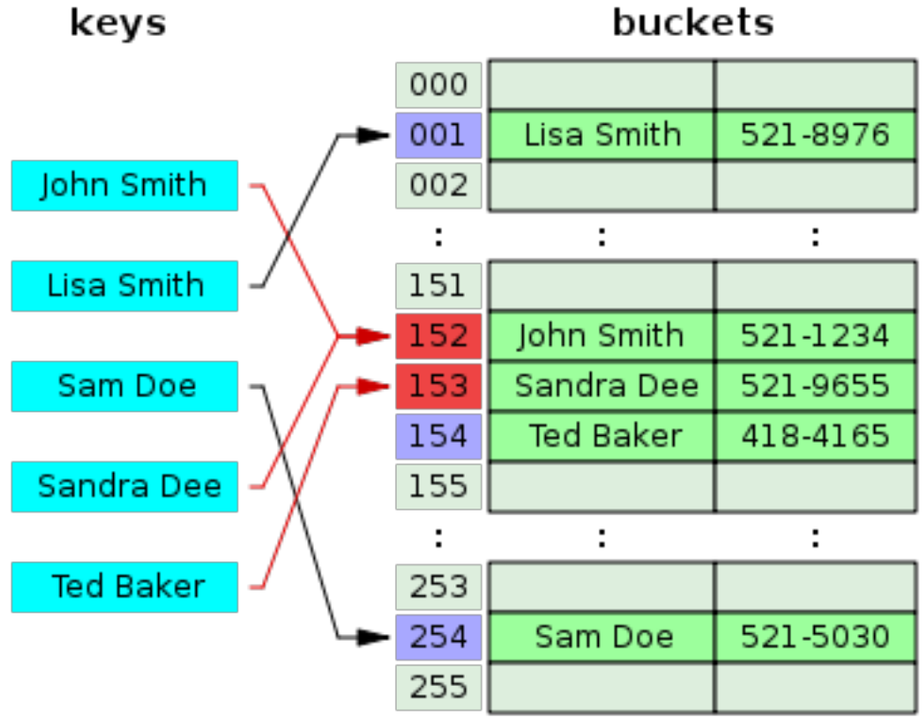


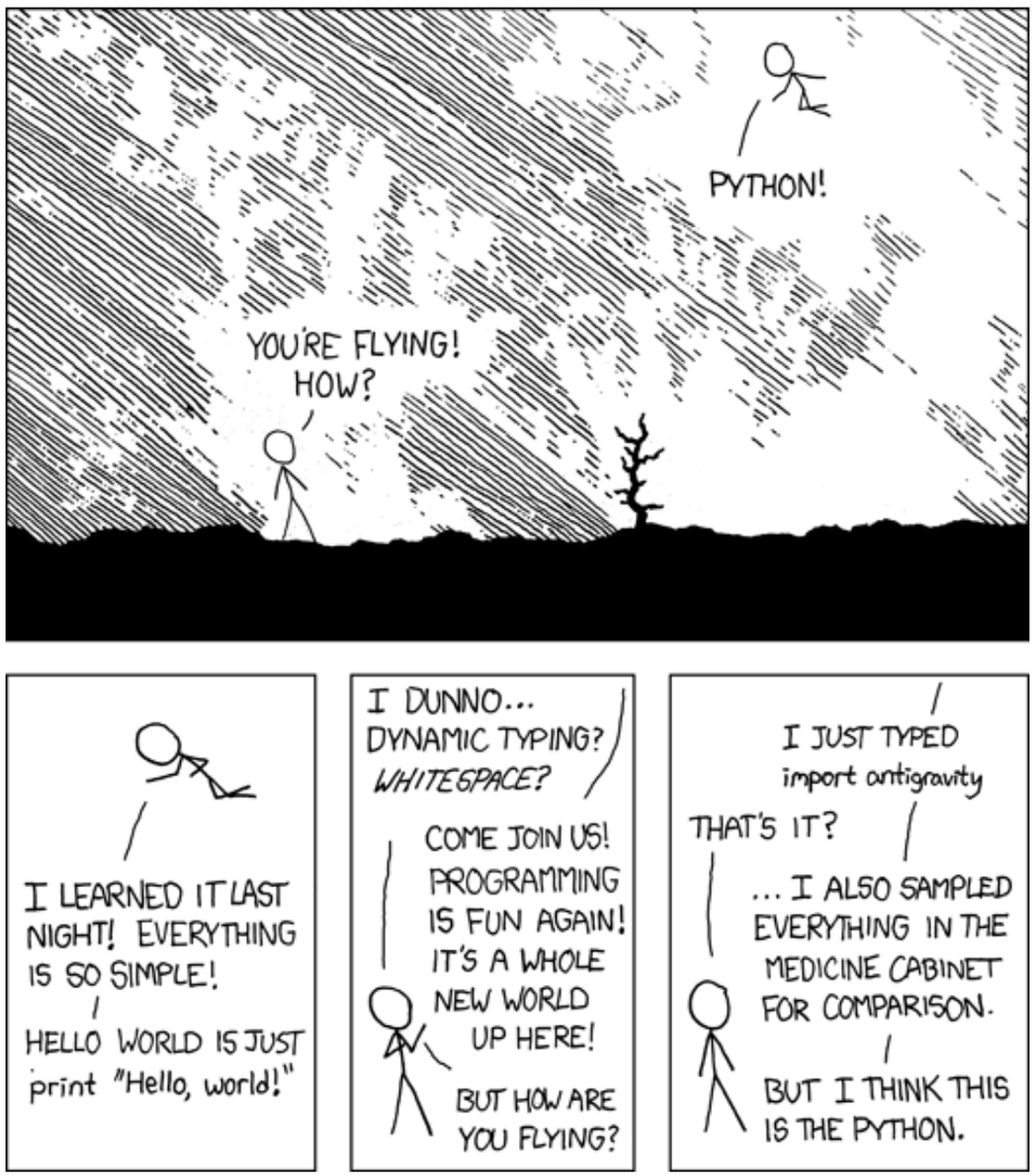


0. stop
1. pots
2. tops

Lists

Hash Tables





- We will use Python to **implement the algorithms.**
- We will use Python to **implement the data structures.**
- However this is **NOT a Python Course**. You do not need to know all the Python details.
- Disclaimer: I am **NOT** a Python expert.

Why Python

- **General** purpose
- Very popular with many libraries
- Very **easy** to program
- Has **depth** if you need it
  - Multi-paradigm: OO, functional, and imperative
  - Associated programming concepts: objects, higher order, etc.

```
def swap(the_list, i, j):
    the_list[i], the_list[j] = the_list[j], the_list[i]


def selection_sort(the_list):
    n = len(the_list)
    for k in range(n):
        min_position = find_minimum(the_list, k)
        swap(the_list, k, min_position)
```

```
def find_minimum(the_list, starting_index):
    min_position = starting_index
    n = len(the_list)
    for i in range(starting_index, n):
        if the_list[i] < the_list[min_position]:
            min_position = i
    return min_position
```

Objectives of the unit

Develop the following **skills**:

- Implement and modify **data types**.
- Compare and **evaluate** different **implementations of data types**.
- Design and implement **algorithms**.
- Calculate **complexity** of simple algorithms.
- Manually translate **high level code into assembly**.

Tentative Timetable

Week	Lecture		
1	1	Intro and House Rules, A simple python program	Simple Python & Algorithmics Workshop
	2	MIPS Architechture	
	3	MIPS Simple programs	
2	4	Decisions in MIPS	MIPS/MARS Workshop
	5	Arrays in MIPS	
	6	MIPS Memory	
3	7	Functions MIPS (Part 1 - Calling)	MIPS Prac - Checkpoint
	8	Functions MIPS (Part 2 - Returning)	
	9	(Extra MIPS lecture + MIPS recursion)	
4	10	Complexity: Searching, Sorting	MIPS Prac - Assessment
	11	Sorting and Complexity II (Invariants)	
	12	Recursion	
5	13	Recursive Sorts and Complexity	Complexity Workshop - Experimental
	14	Assertions, Exceptions, Testing	
	15	Abstract Data Types	
6	16	Classes / Objects	Classes & Objects Workshop / Testing
	17	Variables and Scoping in Python	
	18	List Array	
7	19	Sorted List (Binary Search & Overloading)	No Pracs
	20	Stack Array	
		Mid Semester Test	
		BREAK	

Tentative Timetable

8	21	Queue Array	Containers - checkpoint
		(Anzac Day - Check prelecture video instead)	
	22	Linked Structures & Linked Stacks	
9	23	Linked Queues	Containers - Assessment
	24	Linked Lists	
	25	Iterators	
10	26	Recursion vs Iteration	Iterators / Generators / Linked Structures
	27	Hash Tables	
	28	Collision Resolution	
11	29	Collision Resolution II	Hashtables - checkpoint
	30	Binary Tree Traversal	
	31	Binary Search Trees	
12	32	Priority Queues	Hashtables - Assessment
	33	Heaps	
	34	Heaps II / Epilogue	

Timetable Synopsis

Lectures

- **Monday** (3 PM, R5)
- **Tuesday** (9 AM, H5)
- **Thursday** (1 PM, E3)

Pracs: Once a week.

- **3 Interview Pracs:**
  - Each assessed Prac runs over two weeks.
  - First week: aim to reach the **checkpoint** (**hurdle**) and get feedback.
  - Second week: Whole prac will be marked.
- **5 Workshops:** All other weeks (except week 7)
  - Prepare before the workshop.
  - **Code review** at the end of the workshop. Working in pairs.
    Must happen at the end of the prac, must be present.

Tutes: Once a week (1 hour)

Assessment

- Assessed practical sessions (20%)
- Code review reports (5%) - during workshops
- Student participation — via **quizzes** (5%)
- Mid Semester Test (10%) — Week 7 during a lecture.
- Exam (3 hours) (60%)

Special consideration

http://www.monash.edu.au/exams/special-consideration.html

Pracs

Week 0: Revision

Week 1: Simple programs and Assembly

Learning Objective:

The first week is dedicated to:

- Revise simple Python Programs
- MIPS Architecture
- MIPS Simple programs

Please make sure you have read EVERYTHING included in Week 0. You will need it.

Documents for Tute 1

Tute 1 77.8KB PDF document

Documents for Prac 1

FIT1008 PracGuide

Workshop Week 1

Prac Submission

Workshop Week 1

Lecture Notes Week 1

Lecture 1: Introduction 3.4MB PDF document

Lecture 2: MIPS Architecture 12.9MB PDF document

Lecture 3: Assembly Programming 603.4KB PDF document

Resources

MIPS Pamphlet (epub available)

MIPS Reference Sheet 100.2KB PDF document

MIPS_AppendixA

Lecture MIPS code

Submitting Pracs

- At the end of **each prac** you must submit your solutions (for both assessed & non-assessed pracs).

- You must compress your source files and associated documentation in one **zip file**.

- You **must** name your zip file with the following format:

<STUDENTID>_PRAC<N>.zip

- eg. 123456789_Prac1.zip.

- **START EARLY**

Cheating, Collusion, Plagiarism

- **Cheating:** Seeking to obtain an unfair advantage in an examination or in other written or practical work required to be submitted or completed for assessment.

- **Collusion:** Unauthorised collaboration on assessable work with another person or persons.

- **Plagiarism:** To take and use another person's ideas and or manner of expressing them and to pass them off as one's own by failing to give appropriate acknowledgement. This includes material from any source, staff, students or the Internet – published and un-published works.

http://infotech.monash.edu.au/resources/student/assignments/policies.html

Moss

A System for Detecting Software Plagiarism

What is Moss?

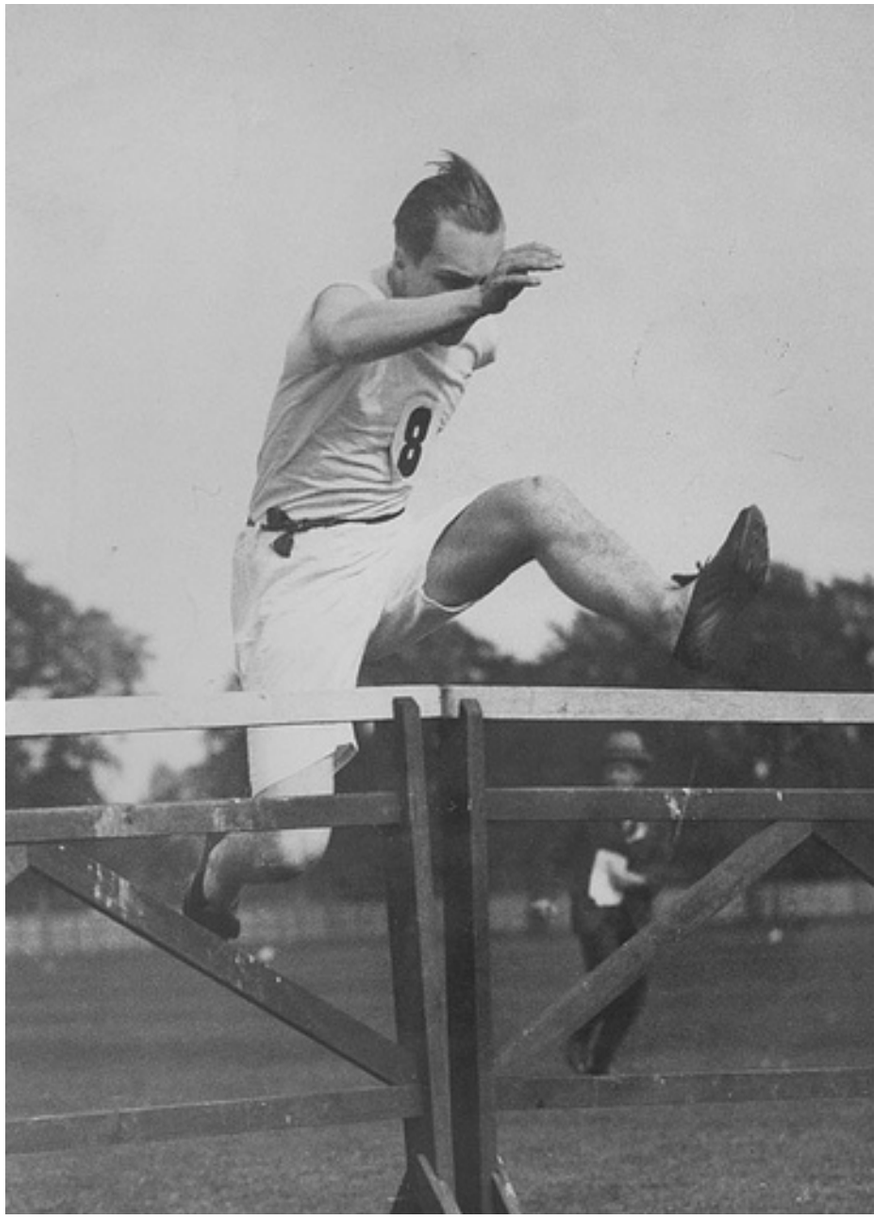
Moss (for a Measure Of Software Similarity) is an automatic system for determining the similarity of programs. To date, the main application of Moss has been in detecting plagiarism in programming classes. Since its development in 1994, Moss has been very effective in this role. The algorithm behind moss is a significant improvement over other cheating detection algorithms (at least, over those known to us).

https://theory.stanford.edu/~aiken/moss/

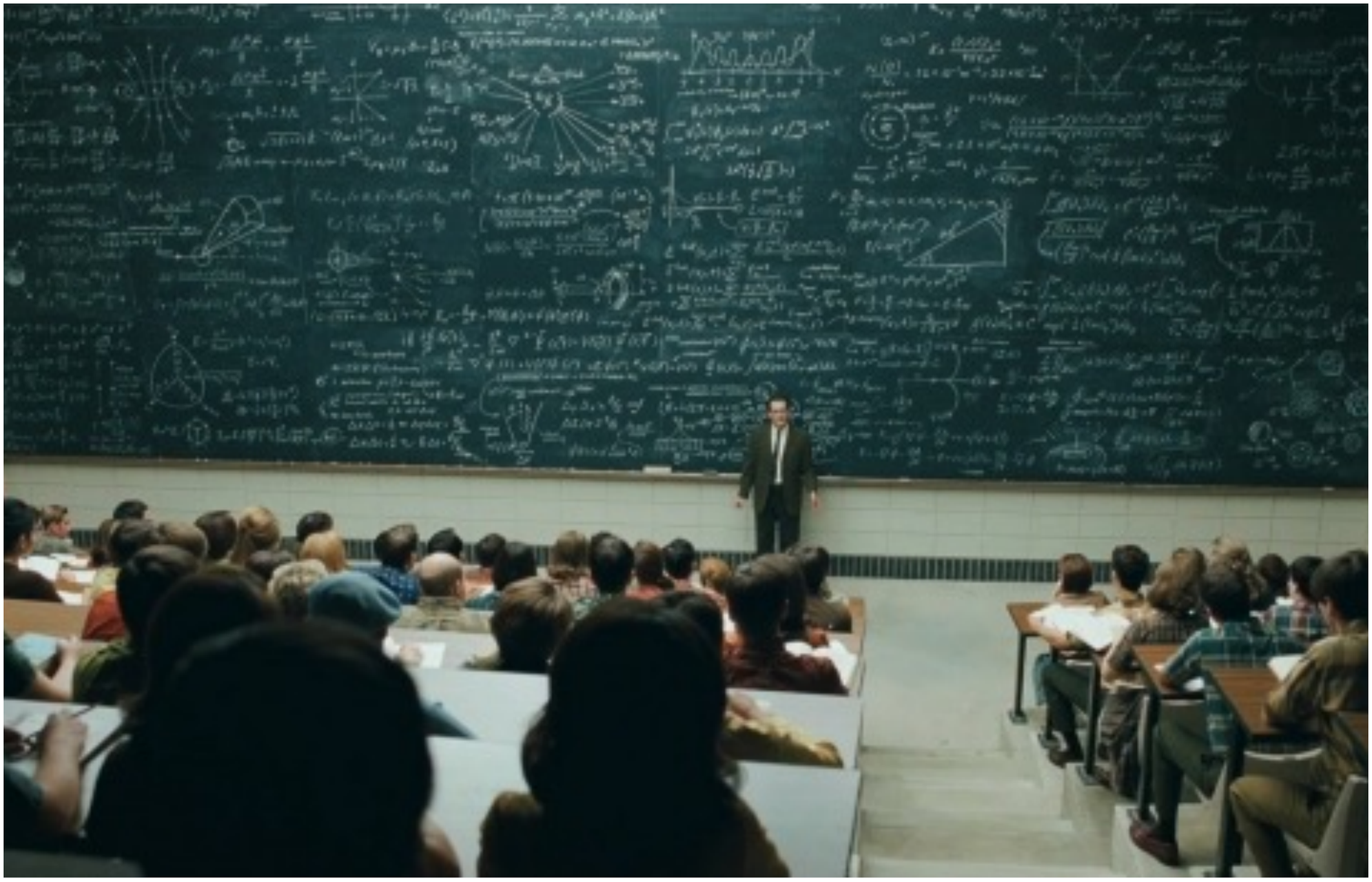
Hurdles

- To pass this unit a student must obtain:
  - ➡ **At least 40% of the total within semester assessment.**
  - ➡ **At least 40% of the exam marks.**
  - ➡ **An overall unit mark of 50% or more.**

- If a student does not pass these hurdles then a mark of no greater than **49N** will be recorded for the unit.



Not how we do lectures.



http://www.pedagogygeek.com/category/uncategorized/

http://gregteselle.com

Some Issues

- Difficult for students to **remain attentive**.
- Not enough opportunity to **critically think through the concepts**.
- We want **learning**, **not memorising** material.
- Difficult for lecturers to **gauge students' understanding.**

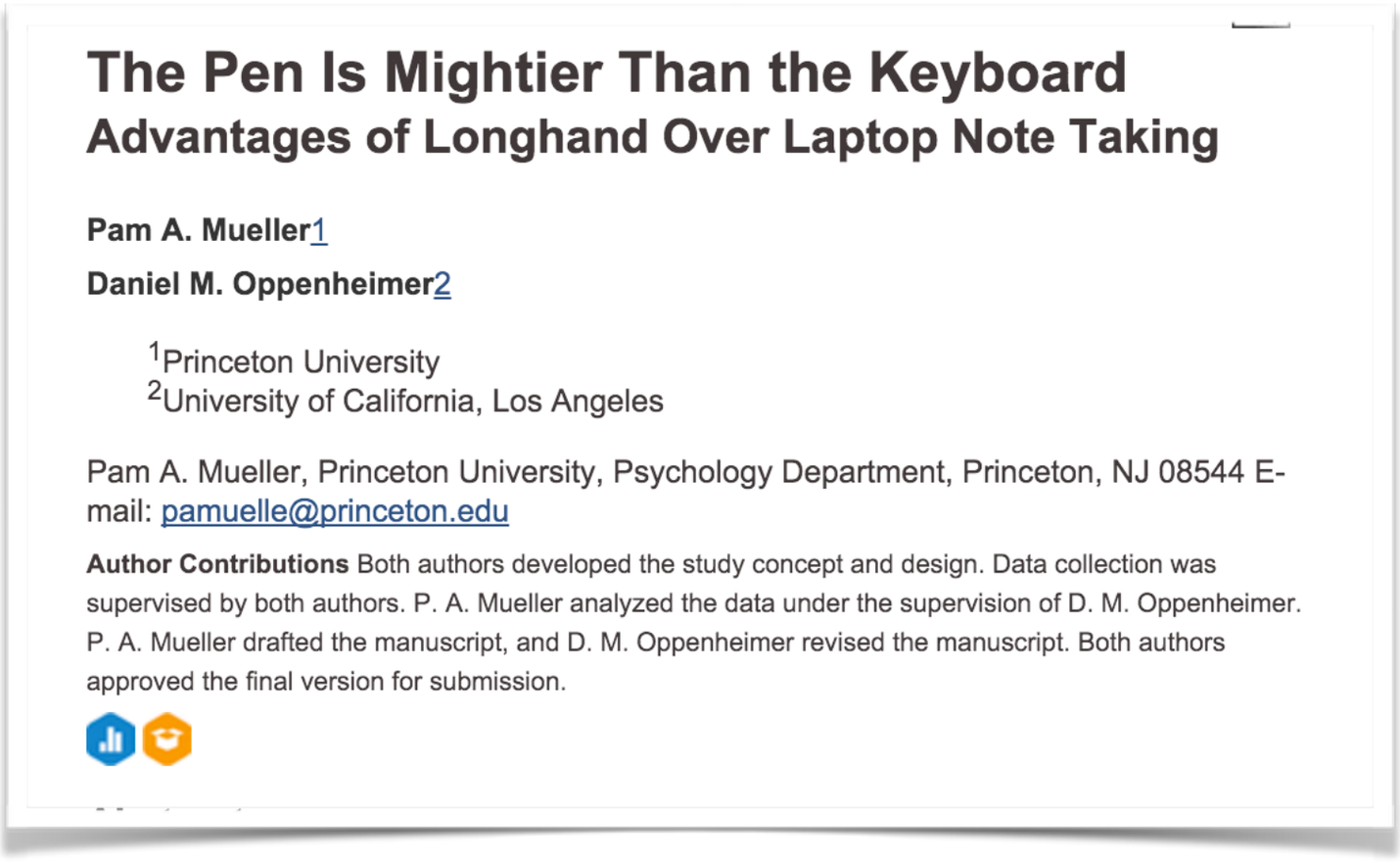
How we do lectures.

Peer Instruction

The Approach

- Lectures are **interspersed with questions**.
- You are given some time to think about the question
- You then give your responses.
- We look at the responses.
- You then **spend some time discussing your answers** with the students around you.
- You respond again.
- We then look at the responses and **discuss** them as a class.

If possible take notes…



The Pen Is Mightier Than the Keyboard
Advantages of Longhand Over Laptop Note Taking

Pam A. Mueller[1]
Daniel M. Oppenheimer[2]

[1]Princeton University
[2]University of California, Los Angeles

Pam A. Mueller, Princeton University, Psychology Department, Princeton, NJ 08544 E-mail: pamuelle@princeton.edu

Author Contributions Both authors developed the study concept and design. Data collection was supervised by both authors. P. A. Mueller analyzed the data under the supervision of D. M. Oppenheimer. P. A. Mueller drafted the manuscript, and D. M. Oppenheimer revised the manuscript. Both authors approved the final version for submission.

(slides are usually not self-standing **on purpose**)

Recommended Reading

- **MIPS Assembly Language Programming.** Britton.

- **Problem Solving with algorithms and data structures using Python.** Miller & Ranum.
  [available online under CC license:
  https://interactivepython.org/runestone/static/pythonds/index.html]

- **Data Structures & Algorithms in Python.** Goodrich, Tamassia & Goldwasser.

Help is Available

- Lecturer
- Tutors
- Help Room Consultations (every week TBA)
- Moodle
- Co-ordinators
- Administration Officers

Consultations

Julian García

Thursday: 2 PM to 3 PM

Office 230, 25 Exhibition Walk, Clayton

Do you have any form of condition (medical, disability other) that impacts on your ability to study?

Disability Support Services provides a range of services for registered students including:

- Notetakers and Auslan interpreters
- Readings in alternative formats
- Adaptive equipment and software
- Alternative arrangements for exams

For further information and details about how to register:

Email: disabilitysupportservices@monash.edu
Phone: 03 9905 5704
Web: monash.edu/disability

ALSO:

http://www.monash.edu/health/counselling

Study Hacks by Cal Newport

On Preparation:

Drizzle Test Preparation Over Many Days
How early should you start studying? This post lays out the basic philosophy preached in Straight-A. Put simply: start early; work in little batches.

Use Focused-Question Clusters to Study for Knowledge Based Tests
How should you study for classes that require you to know a large number of facts and concepts? I overlooked these classes in Straight-A (as many of you subsequently brought to my attention.) In this post I rectify this oversight. It was originally written for multiple choice tests, but the advice is relevant for any exam requiring a large amount of memorized information.

Pseudo-Work Doesn't Equal Work and Studying is a Technical Skill
How are some high-scoring students able to escape the stress of the grind lifestyle? These two early posts, from a series titled "The Straight-A Gospels," lay out the core philosophical ideas behind the mysterious, yeti-like low-stress 'A.' I recommend a quick review before diving too deep into exam period chaos.

http://calnewport.com/blog/2008/04/28/monday-master-class-the-study-hacks-guide-to-exams/

Studying is a skill

Lecturer

Dr Julian Garcia

- **Room 230**, 25 Exhibition Walk, Clayton
- Tel: 9905 3654
- **Email**: Julian.Garcia@monash.edu
- **Consultation** Times:
  - Thursday, 2-3PM

Looking forward!