

# Welcome to CSCI 235

# Today's Plan

Welcome

Logistics (Rules of the game)

What is CSCI 235?

Why Software Engineering?



# Acknowledgments

This course was designed with input from many great resources other than the required textbook

Many thanks for materials and inspiration to

**Tiziana Ligorio**

Simon Ayzman

Susan Epstein

Keith Schwarz

Ioannis Stamos

Stewart Weiss

# Logistics

## (The rules of the game)

Your first assignment.  
**MUST READ!!!**

Course Webpage

Syllabus, Programming Guidelines

Programming Projects

Gradescope

Linux Accounts

Communication and Help

# Course Webpage

[maryash.github.io/235/index.html](http://maryash.github.io/235/index.html)

Visit regularly for:

Tentative Schedule and Announcements

Lecture Notes

Programming Projects

Study Questions

# Programming Projects

## 6 Programming Projects

They build upon each other - **don't fall behind!!!**

All projects submitted on **Gradescope**

If you haven't done so already, login to **Gradescope ASAP**

**MUST USE YOUR CORRECT EMPL ID**

**Projects due by 11pm on the due date**

**MUST READ:** [Programming Guidelines](#) on course webpage

# Gradescope

To be used for submission of ALL programming projects

**Check your email and follow invitation instructions**

# Gradescope

**If you haven't received an invitation email:**

Course Entry Code: **M4W2J8**

**If you DON'T already have an account:**

1. Go to [www.gradescope.com](http://www.gradescope.com)
2. Sign Up as a Student
3. Enter Course Entry Code
4. Enter your information

**If you DO already have an account:**

1. Go to [www.gradescope.com](http://www.gradescope.com) and log in
2. At the bottom right click on Enroll in Course
3. Enter Course Entry Code



Gadescopes are not for  
Debugging!!!

# Linux Accounts

## **MUST HAVE!!!**

- Login to new account as soon as you receive email (look for it in your Hunter mail in the next few days)
- Reclaim your existing account by **September 16** otherwise it will be deleted (type **touch fall.2019** at the terminal after logging in)

## **Assignment 1 (BASELINE) :**

using your Linux account, upload `hello_world.cpp` to a machine in the Linux lab, then remotely compile using `g++` and run your program

## **Follow instructions in:**

- **Programming Guidelines document on course web page**  
**and**

-<http://compsci.hunter.cuny.edu/~csdir/>



# Plagiarism

- Submitting to Gradescope does not mean no one will look at your code
- **We check for plagiarism** and **report ALL cases to the office of student affairs**
- Sadly, last semester we reported several cases

Communication and Help

# Let us hear from you!

If you find a typo or mistake let me know!!!

Blackboard forum

If you don't understand something ask!!!

In class, on Blackboard forum or UTA

If you have concerns on something other than course content, talk to me:

Email, office hours or by appointment

# Questions and Forum Etiquette

Different prior exposure to the material

**All questions are good questions!!!**

Friendly and collegial environment - we are here to help

# Be Proactive if you want Help!

- If everyone show up for help on the project due date it will not be possible to give help to all!
- Please be proactive, start early and plan ahead!

# Introducing Lecture Activity

6% of final grade

Attendance and must show an adequate attempt

Work out a new problem 



# What is CSCI 235?

Programming => Software Analysis and Design  
Expected professional and responsible conduct

Think like a Computer Scientist:

Design and maintain complex programs

Software Engineering, Abstraction, OOP

Design and represent data and its management

Abstract Data Types

Implement data representation and operations

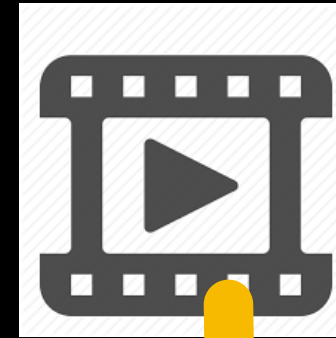
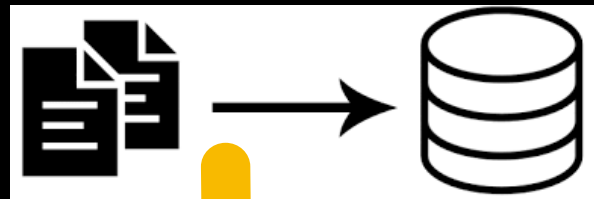
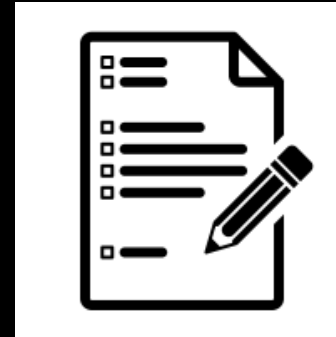
Data Structures

Algorithms

Understand Algorithm Efficiency



“It’s all just bits and bytes...”



STRUCTURE



101010101010101010  
010101010101010101  
10011001100110011001

# Increasing software size

Society keeps digitizing more aspects of life

Software keeps getting bigger

Size and interaction of software systems ever increasing

Exciting!!!

Daunting for software engineers



# Typical software size / lines of code

~1 - 10	Hello world
~100	Most STL queue implementations
~1,000	Typical Computer Science curriculum term project
~10,000	Intensive team project
~100,000	Most Linux command line utilities
~1,000,000	Linux g++ compiler
~10,000,000	Mozilla Firefox
~50,000,000	Microsoft's Windows
~2,000,000,000	Google (search, maps, docs, gmail, ...)

# Problems with large software

Every bit counts!

A single incorrect bit may result in:

- negative instead of positive int
- pointer past the end of an array
- unsorted rather than sorted vector
- ...

**Program performs unexpectedly**

# Problems with large software

Every bit counts!

A single incorrect bit may result in:

- negative instead of positive int
- pointer past the end of an array
- unsorted rather than sorted vector
- ...

**Program performs unexpectedly**



# Problems with large software

Two lines of code **interact** if they manipulate same data

```
int x = 5;    // if I change the x to my_var  
cout << x;    // I must change it here too
```

Assume  $n$  lines of code

Any line may interact with any number of other lines

$n(n-1)/2 = (n^2-n)/2$  possible interactions

With 10 lines of code there may be  
45 interactions

Unlikely but it gives  
you an idea of how  
bad it can get




# Problems with large software

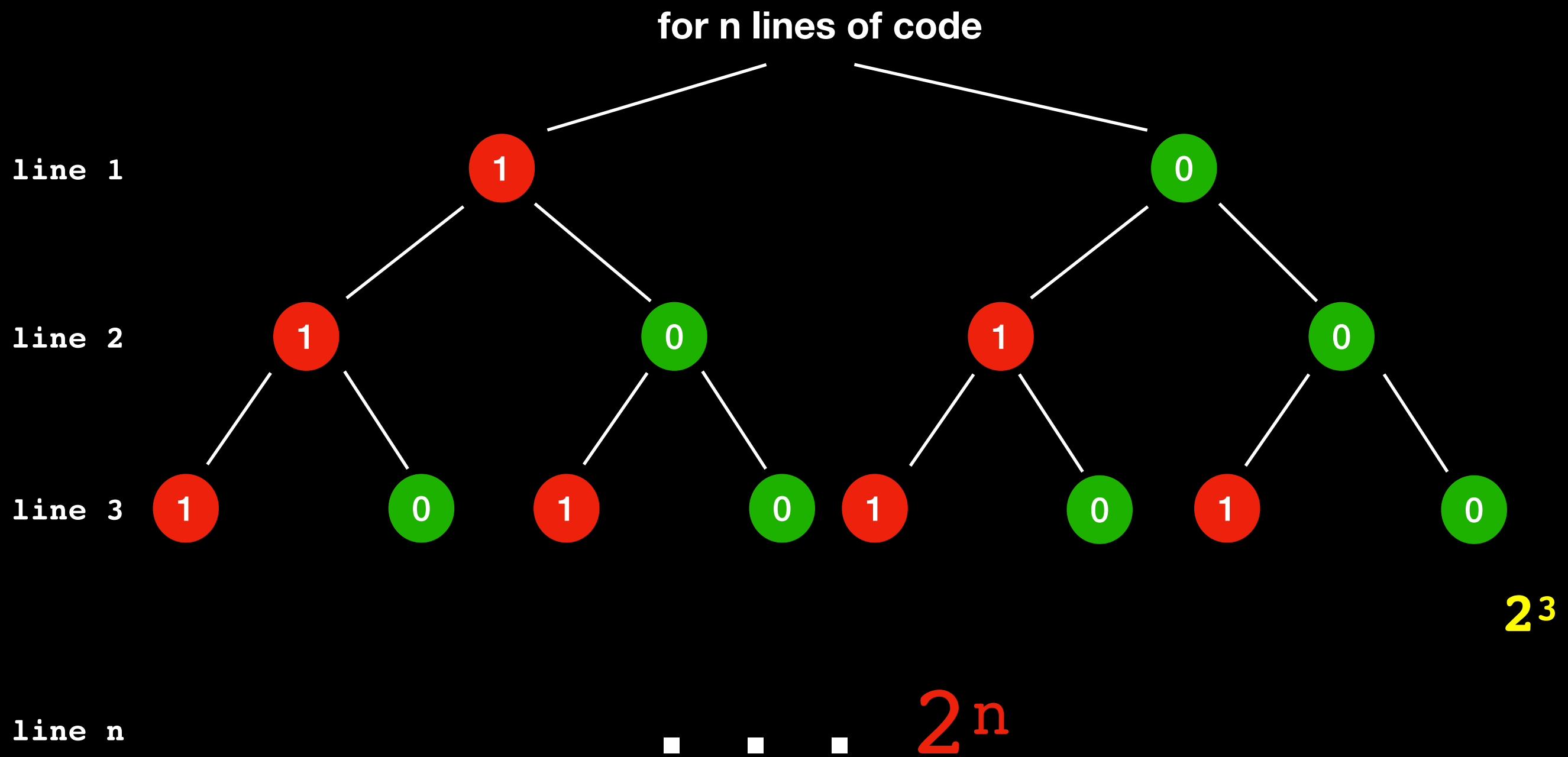
Assume line 1 shares same data with lines 6 and 23.  
So in actuality the three lines all together form an  
**interaction**

If we think of subsets of lines of code interacting  
(sharing the same data) . . .

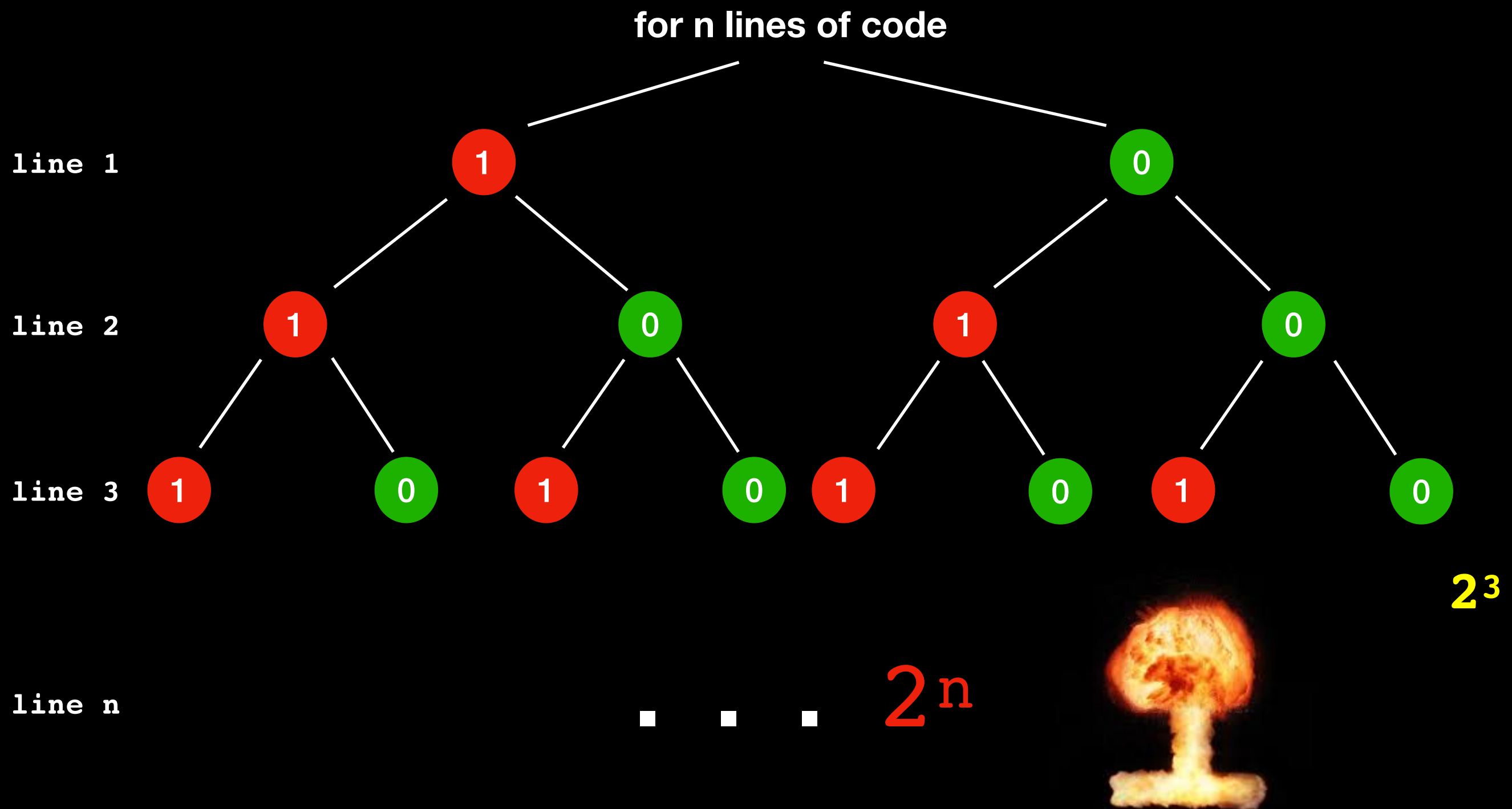
How many possible subsets?



Again unlikely all possible  
subsets will interact but it  
gives you an idea of why  
you'd want to control it



Every path down the tree is an interaction among one possible subset of lines of code



Every path down the tree is an interaction among one possible subset of lines of code

# Lecture Activity

Draw a **very small** square on the leftmost bottom corner

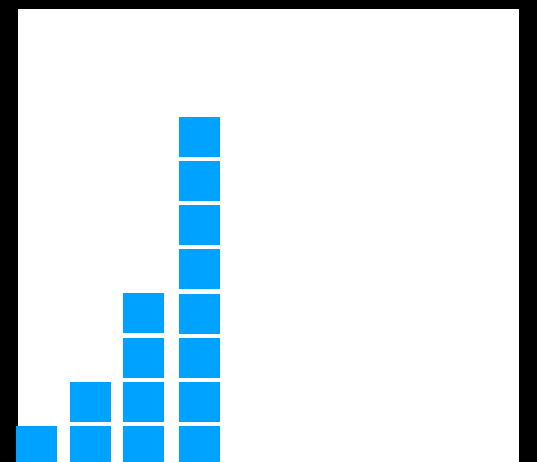
Next to it, double it (2 squares one on top of other)

Next to it, double it (4 squares one on top of other)

Next to it double it , ...

Keep going...

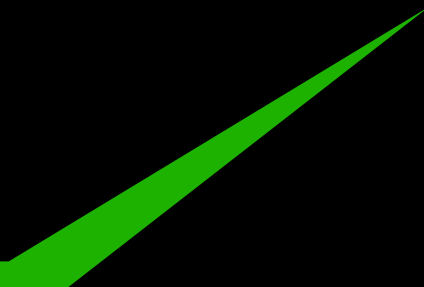
**How quickly do you run off the top of the page?**



# Problems with large software

How folding paper can get you to the moon:

<https://www.youtube.com/watch?v=AmFMJJC45f1Q>



Watch this home if we  
don't have time at the  
end of lecture

# Problems with large software

How do you go about modifying code with many **interactions**?

Larger software has greater likelihood of error

More difficult to debug and modify

# Control software interaction!!!

Software with **many interactions is bad!!!!**

Write small units of code

Minimize Interactions/Coupling!!!

Enforce strict rules on how code interacts

How? We will consider:

- Principles of Software Engineering
- Object Oriented Programming

# What is Software Engineering?

"The application of a **systematic**, **disciplined**, **quantifiable** approach to the development, operation, and maintenance of software"

*IEEE Standard Glossary of Software Engineering Terminology*



# Big Ideas of Software Engineering

Modularity

Modifiability/Extensibility

Ease of Use

Fail-Safe Programming

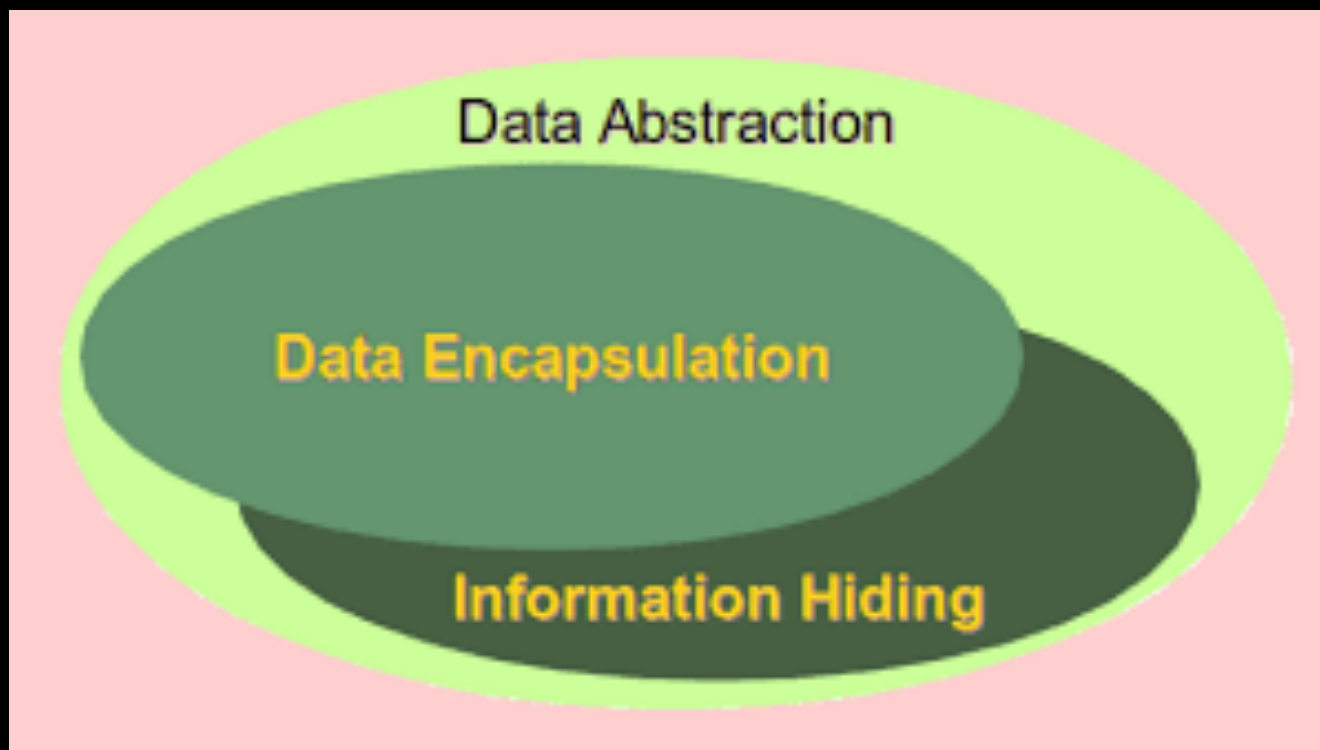
Debugging

Testing

**We will come back to these throughout the course**

**APPENDIX B**

# Object Oriented Programming



# Next Time

## Abstraction and OOP