COMP 3760

Design and Analysis of Algorithms

BCIT – Computer Systems Technology

Today's plan

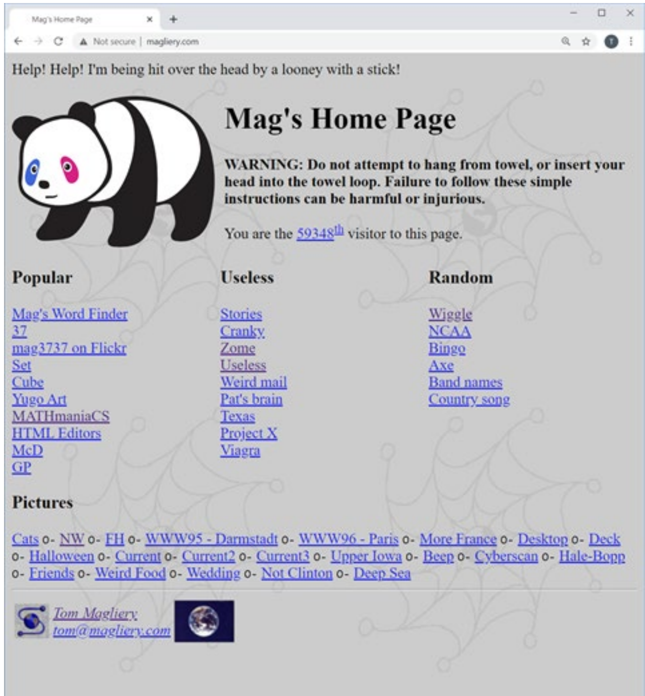
- Course intro slides (this)
- Reminder of Math You Should Know
  - *Offline review*
  - See slides on Learning Hub
  - Bring questions to Discord or Lab
- Week 1 Lecture
  - Slides on Learning Hub, lecture here & now

About me

- Tom Magliery
- Email: tmagliery@bcit.ca
- Office hours:
  - This means “Pledge to be online and reachable now”
  - Days/times TBD
  - Note: I’m online lots of other times
- Other communication:
  - Discord https://disco...

My history

- Undergrad: Math and Computer Science (Kansas)
- Grad: Computer Science (Illinois)
- Some things I’ve helped create:
  - NCSA Mosaic, HTML, XML, XMetaL
- My home page →
  has existed since 1994
- Joined BCIT Computing in 2019



Help! Help! I'm being hit over the head by a looney with a stick!

Mag's Home Page

WARNING: Do not attempt to hang from towel, or insert your head into the towel loop. Failure to follow these simple instructions can be harmful or injurious.

You are the 59348th visitor to this page.

Popular

Mag's Word Finder
37
mag3737 on Flickr
Set
Cube
Yugo Art
MATHmaniaCS
HTML Editors
McD
GP

Useless

Stories
Cranky
Zome
Useless
Weird mail
Pat's brain
Texas
Project X
Viagra

Random

Wiggle
NCAA
Bingo
Axe
Band names
Country song

Pictures

Cats o- NW o- FH o- WWW95 - Darmstadt o- WWW96 - Paris o- More France o- Desktop o- Deck o- Halloween o- Current o- Current2 o- Current3 o- Upper Iowa o- Beep o- Cyberscan o- Hale-Bopp o- Friends o- Weird Food o- Wedding o- Not Clinton o- Deep Sea

Tom Magliery
tom@magliery.com

Course learning objectives

- Understand the basic framework of algorithm analysis.
- Analyze pseudo-code using asymptotic notations.
- Compare the order of growth of different algorithms.
- Understand the differences between nonrecursive and recursive algorithms.
- Describe some common algorithm design strategies: Divide and Conquer, Transform and Conquer, Greedy Technique, Dynamic Programming, etc.
- Recognize different types of computing problems and how to solve them.
- Apply algorithm design techniques to solve some practical problems.
- Specify algorithms in pseudocode.
- Implement solutions by using appropriate data structures.
- Deduce the complexity of a program by running different experiments.
- Argue the correctness of the algorithms.
- Find lower bounds for some simple problems.

Textbook

- Introduction to The Design and Analysis of Algorithms, 3rd Ed.
  - Author: Anany Levitin

- Read the sections that relate to the material we cover in class

Course sessions

- Lectures:
  - Attendance required
  - 1st week online
    - Prediction: more, too

- Labs:
  - Attendance required
  - Quizzes
  - Supplemental activities
  - Question period
  - Time to work on lab assignments

Course grade components

Course grade components

- Quizzes – 15%
- Labs – 25%
- Participation – 5%
- Midterm – 25%
- Final – 30%

Quizzes – 15%

- Held during lab (starting next week!)
  - Covers most recent lecture material
  - 15-20 minutes, 5-8 pts
  - Usually online (LH), occasionally paper (😲)
- Good practice for exams
- Aiming for 9 quizzes
  - Lowest quiz score is dropped
- Each quiz is small, but they do add up

Lab assignments – 25%

- Implementing the algorithms we’re studying
- Reinforcing lecture/course concepts
- ~9 assignments (all will count)
- Timing (usually):
  - Available: Probably Fridays
  - Due: Saturday midnight

- Other requirements
  - Language: Java
  - Environment: Whatever you have
  - Coding: More details with first coding assignment

Midterm (25%) and Final (30%)

- Details when the time comes

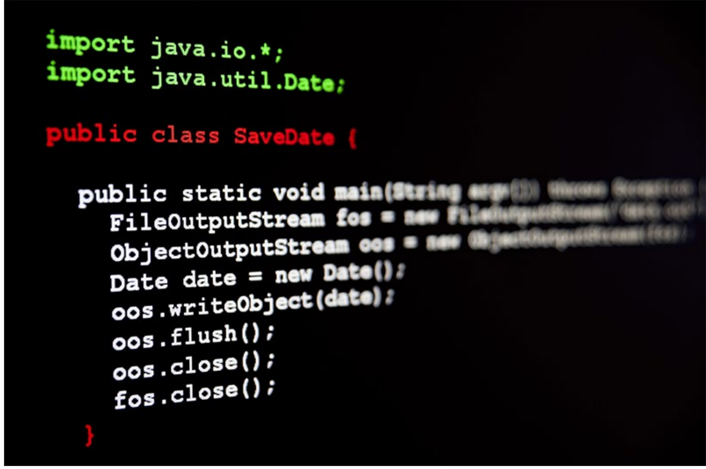
- This course does NOT have a “must score 50% on the midterm/final” requirement

Participation – 5%

- Things that can affect this:
  - Attendance
  - Raising questions/comments in class
  - Participating in lab discussions/activities
  - Generally engaging beyond default levels

Other points of interest

This is not a programming course!



```
import java.io.*;
import java.util.Date;

public class SaveDate {

  public static void main(String args[]) throws Exception {
    FileOutputStream fos = new FileOutputStream("date.ser"
    ObjectOutputStream oos = new ObjectOutputStream(fos);
    Date date = new Date();
    oos.writeObject(date);
    oos.flush();
    oos.close();
    fos.close();
  }
```

- Is this paradoxical? Ironic? Just plain false?
  - Maybe paradoxical – we certainly will write code
- But: We are mostly studying the *abstract idea* of algorithms
- You may actually need to suspend some of your coding skilz at times

- But we do need *some* kind of language to represent, communicate, and discuss algorithms ...

Pseudocode

- We will use pseudocode a LOT
  - Lectures, textbook, quizzes, labs, exams
- Pseudocode expresses basic programming statements
  - Variables, assignments, expressions, conditional statements, loops, subroutines
- My own style is ... ill-defined
- Postel’s Law: Be liberal in what you accept, and conservative in what you send
- When in doubt, ask

Tips for success

- Practice!
- Keep up with the material
- Interrupt me ANY time in class (lecture or lab)
    - If I say something that confuses you, don’t assume that I know what I’m talking about
    - I make mistkaes*!

* I truly swear I originally made that typo by accident. But now I have left it there on purpose for ~~two~~ ~~three~~ ~~four~~ *hrair* terms in a row.

Virtual donuts 🍩

- I give out virtual donuts
  - Could be: planned, spontaneous, pre-announced, secret, easter eggs, unlocked achievements, ...
- Spontaneous ones may occur
  - Surprise me, impress me, amuse me, ...
- Maybe (in person) some virtual donuts will turn into *real* donuts someday