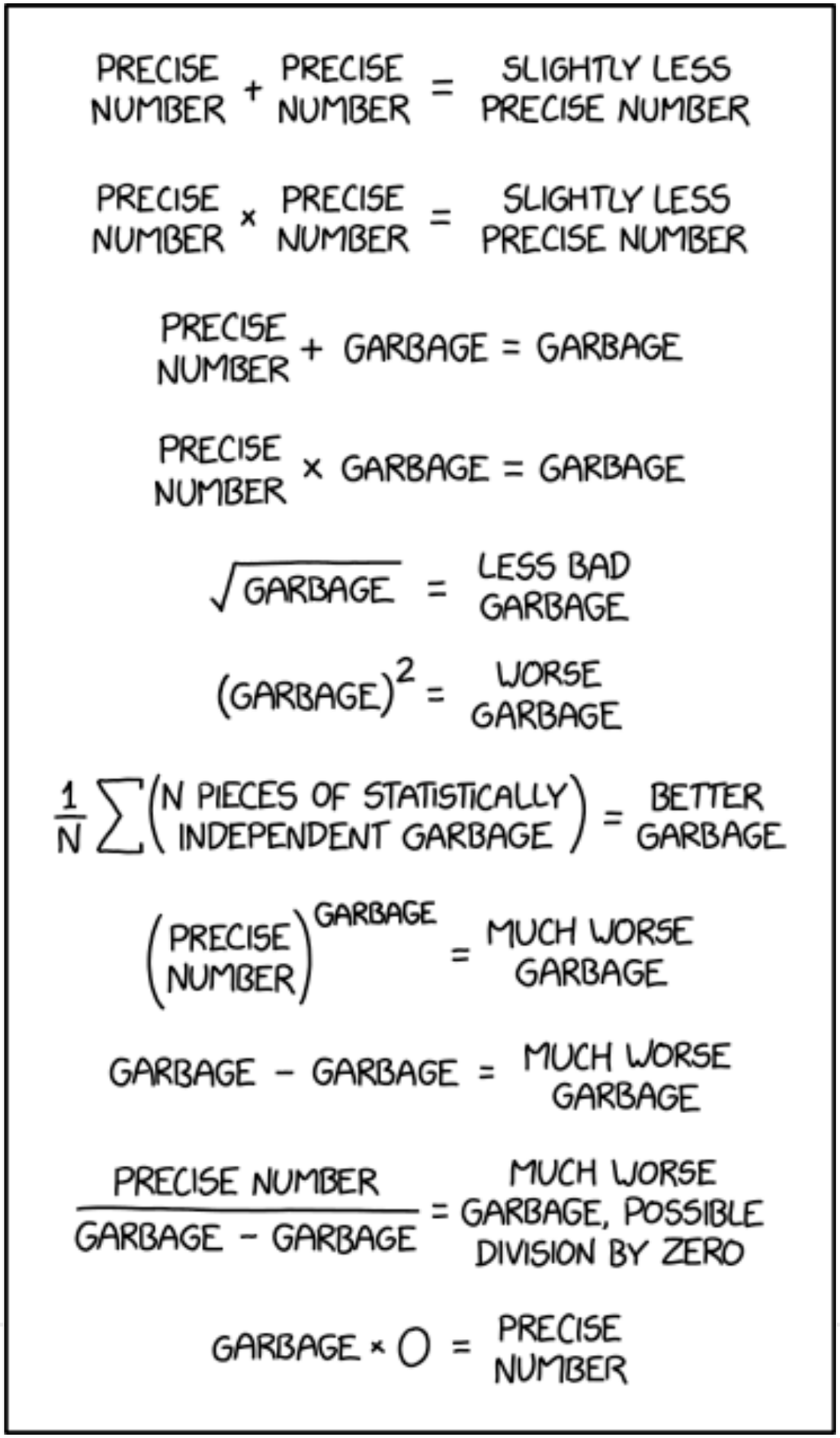
CS 649 Big Data: Tools and Methods
Spring Semester, 2022
Doc 30 End Remarks
May 3, 2022

End Remarks

https://xkcd.com/2295/



http://highscalability.com

how to scale software - primarily web sites & backends

Hacker News

https://news.ycombinator.com

Martin Fowler Bliki

A website on building software effectively

https://martinfowler.com

Author

Works at ThoughtWorks

Software Architecture Guide

https://martinfowler.com/architecture/

What is architecture?

Why does architecture matter?

Application Architecture

Application Boundary

Microservices Guide

Serverless Architectures

Micro Frontends

GUI Architectures

Presentation Domain Data Layering

ThoughtWorks Technology Radar

Techniques

Tools

Platforms

Languages & Frameworks

Adopt

Trial
- Worth pursing
- Try on projects that can handle risk

Assess
- Worth exploring
- How will it affect your enterprise

Hold
- Proceed with caution

Techniques

Data Mesh

decentralized organizational and technical approach in sharing, accessing and managing data for analytics and ML

CUPID

properties to achieve “joyful” code:

Code should be

composable,

follow the Unix philosophy

be predictable,

idiomatic and

domain based.

Techniques

The streaming data warehouse

SQL-based streaming applications across an enterprise might constitute a streaming data warehouse

ksqlDB

Platforms

Apache Iceberg

Open table format for very large analytic data sets

Google Cloud Dataflow

Cloud-based data-processing service for both batch and real-time data-streaming applications.

Temporal

Platform for developing long-running workflows, particularly for microservice architectures.

Tools

AKHQ

GUI for Apache Kafka

Metaflow

Python library and back-end service that helps data scientists and engineers build and manage production-ready data processing, ML training and inference workflows.

What every computer science major should know

Dr. Matt Might
University of Utah

http://matt.might.net/articles/what-cs-majors-should-know/

What should every student know to get a good job?

What should every student know to maintain lifelong employment?

What should every student know to enter graduate school?

What should every student know to benefit society?

Portfolio verse Resume

A resume says nothing of a programmer's ability

Portfolio
- Personal blog
- Projects
- Github
- Open source projects

Technical Communication

Lone wolves in computer science are an endangered species

In smaller companies, whether or not a programmer can communicate her ideas to management may make the difference between the company's success and failure

Writing for Computer Science by Zobel.

Even a Geek Can Speak by Asher.

Unix Philosophy

linguistic abstraction and composition

Should be able to

Navigate and manipulate the filesystem;
Compose processes with pipes;
Comfortably edit a file with emacs and vim;
Create, modify and execute a Makefile for a software project;
Write simple shell scripts.

Unix Philosophy

Sample tasks

Find the five folders in a given directory consuming the most space

Report duplicate MP3s (by file contents, not file name) on a computer.

Take a list of names whose first and last names have been lower-cased, and properly recapitalize them.

Find all words in English that have x as their second letter, and n as their second-to-last.

Directly route your microphone input over the network to another computer's speaker.

Replace all spaces in a filename with underscore for a given directory.

Report the last ten errant accesses to the web server coming from a specific IP address.

Systems administration

Every modern computer scientist should be able to:

Install and administer a Linux distribution.

Configure and compile the Linux kernel.

Troubleshoot a connection with dig, ping and traceroute.

Compile and configure a web server like apache.

Compile and configure a DNS daemon like bind.

Maintain a web site with a text editor.

Cut and crimp a network cable.

Programming languages

Programming languages rise and fall with the solar cycle.

A programmer's career should not.

The best way to learn how to learn programming languages is to learn multiple programming languages and programming paradigms.

To truly understand programming languages, one must implement one.

Programming languages

Racket

C

JavaScript

Squeak

Java

Standard ML

Prolog

Scala

Haskell

C++

Assembly

Racket

Aggressively simple syntax

For a small fraction of students, this syntax is an impediment.

To be blunt, if these students have a fundamental mental barrier to accepting an alien syntactic regime even temporarily, they lack the mental dexterity to survive a career in computer science.

Racket's powerful macro system and facilities for higher-order programming thoroughly erase the line between data and code.

If taught correctly, Lisp liberates

How to Design Programs

https://htdp.org

Squeak

Squeak is a modern dialect of Smalltalk, purest of object-oriented languages

It imparts the essence of "object-oriented."

Introductions to Squeak

http://wiki.squeak.org/squeak/377

Architecture

There is no substitute for a solid understanding of computer architecture

- transistors
- gates
- adders
- muxes
- flip flops
- ALUs
- control units
- caches
- RAM
- GPU

Operating systems

Any sufficiently large program eventually becomes an operating system

To get a better understanding of the kernel, students could:

Print "hello world" during the boot process;

Design their own scheduler;

Modify the page-handling policy; and

Create their own filesystem.

Networking

Computer scientists should have a firm understanding of the network stack and routing protocols within a network

Every computer scientist should implement the following:

- an HTTP client and daemon;
- a DNS resolver and server; and
- a command-line SMTP mailer.

No student should ever pass an intro networking class without sniffing their instructor's Google query off wireshark.

Security

Computer scientists must be aware of the means by which a program can be compromised

At a minimum, every computer scientist needs to understand:

- social engineering
- buffer overflows
- integer overflow
- code injection vulnerabilities
- race conditions
- privilege confusion

Metasploit: The Penetration Tester's Guide

Security Engineering: A Guide to Building Dependable Distributed Systems

Software testing

Software testing must be distributed throughout the entire curriculum

He uses test cases turned in by students against all other students

Students don't seem to care much about developing defensive test cases, but they unleash hell when it comes to sandbagging their classmates

Visualization

The modern world is a sea of data

The Visual Display of Quantitative Information by Tufte

Graphics and simulation

There is no discipline more dominated by "clever" than graphics.

The field is driven toward, even defined by, the "good enough."

As such, there is no better way to teach clever programming or a solid appreciation of optimizing effort than graphics and simulation.

Over half of the coding hacks I've learned came from my study of graphics.

Topics I left out

Databases
Artificial intelligence
Machine learning
Robotics
Software engineering
Parallelism
User experience design

Disarmingly Forthright MSCS Advice

Nick Black

http://nick-black.com/dankwiki/images/8/85/Msadvice.pdf

Read it

If you'll only take away two things

Read the damn man pages

Check your damn return values

You’re a CS MS student. Act it

Join the ACM and IEEE

Don’t embarrass yourself

- Passwords
- Backups

If you don’t have at least 100 semi-frequent, provocative/informative RSS feeds you’re checking a few times daily, you’re not learning enough

Programming

Vast majority of code you'll read is laughably broken

> if you aren't, at any given time, scandalized by code you wrote five or even three years ago, you're not learning anywhere near enough

Seek out, study, and bookmark good code

Learn to program axiomatically

> take each element of the system, language, and toolchain, and learn it throughout

Keep all your projects in source control systems like git or svn