Data Structures & Algorithms @ Anthrocon 2026

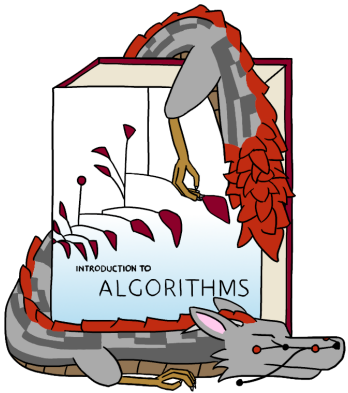
Critters, Cryptids, nCurses

Bunsen Bitti, Unsigned Long

July 3, 2026

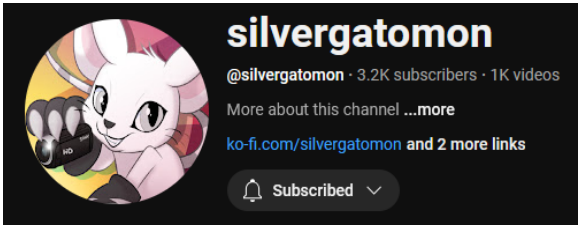
Overview

- This is a comedic/informative look at overlaps between furry and computer science
  - This is not a rigorous lecture
  - There will be puns ^w^
- Don't worry if our jokes don't make sense!
  - We've been coding for most of our lives
  - If you're confused, that's on us
  - We're out of touch with reality
- If you are curious about anything in these slides, feel free to chat with us!



How I Found the Fandom (Unsigned Long)

- Found furries through con videos



How I Found the Fandom (Unsigned Long)

- Found furries through con videos
- Found some eastern dragons with the word “Long” in their name

Tien Long

How I Found the Fandom (Unsigned Long)

- Found furries through con videos
- Found some eastern dragons with the word “Long” in their name
- Thought about “long” for too long

How I Found the Fandom (Unsigned Long)

- Found furries through con videos
- Found some eastern dragons with the word “Long” in their name
- Thought about “long” for too long
- Realized it’s in my code

`short`
`short int`
`signed short`
`signed short int`
	Short signed integer type. Capable of containing at least the [−32 767, +32 767] range.[3][a]

`unsigned short` `unsigned short int`	*Short* unsigned integer type. Contains at least the [0, 65 535] range.[3]

`int`
`signed`
`signed int`
	Basic signed integer type. Capable of containing at least the [−32 767, +32 767] range.[3][a]

`unsigned` `unsigned int`	Basic unsigned integer type. Contains at least the [0, 65 535] range.[3]

`long`
`long int`
`signed long`
`signed long int`
	Long signed integer type. Capable of containing at least the [−2 147 483 647, +2 147 483 647] range.[3][a]

`unsigned long` `unsigned long int`	*Long* unsigned integer type. Capable of containing at least the [0, 4 294 967 295] range.[3]

How I Found the Fandom (Unsigned Long)

- Found furries through con videos
- Found some eastern dragons with the word “Long” in their name
- Thought about “long” for too long
- Realized it’s in my code
- Pun was too good—I made a fursona

How I Found the Fandom (Bunsen)

- FNAF made me a bunny-obsessed furry back in 6th grade

artist: thepipefox (FurAffinity)

How I Found the Fandom (Bunsen)

- FNAF made me a bunny-obsessed furry back in 6th grade
- Started in algorithms, moved to hardware and performance engineering

artist: @Alextheyellowthing (Telegram)

How I Found the Fandom (Bunsen)

- FNAF made me a bunny-obsessed furry back in 6th grade
- Started in algorithms, moved to hardware and performance engineering
- Made an FPGA-controlled protogen for a final project because why not

ProtoFacer: FPGA-Driven Interactive Protogen Head

[*]Note: This is a partially redacted version (to respect Bunsen's privacy) of the original report submitted for MIT's 6.2050 Digital Systems Laboratory course.

1st Bunsen Bitti
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA, USA

***Abstract*—We present a design for a Protogen Head controlled by FPGA fabric. Using an inward-facing OV5640 camera inside the helmet, the Protogen Head reconstructs the cosplayer's face based on real-time video input. The FPGA processes 25 frames-per-second video input from the camera, (limited primarily by electromagnetic interference concerns).**

Experimental testing shows that the ProtoFacer is able to capture blinks and mimic mouth movement during speech, all while using under 400mW of power (excluding external power to the LED boards) and 1.5 MiB of block RAM. This makes it ideal for use in limited memory/power environments.

***Index Terms*—Digital systems, Field programmable gate arrays, Image processing.**

I. Introduction

Protogen heads are a type of mascot head which display a face using multiple LED boards behind a tinted visor. These allow for a greater range of expressions for the cosplayer to choose from since these displays can flexibly display many different facial expressions without mechanical components. Ideally, the displayed face should be representative of the cosplayer's face, allowing for common facial movements such as blinking and smiling to be expressed. However, most models construct faces from a fixed set of pre-determined expressions stored on the controller's memory.

The goal of the ProtoFacer project is to overcome this limitation by leveraging the parallelism of the FPGA fabric to implement a face reconstructor from real-time video input. This presents a number of design challenges:

- The system needs to fit within the limited space of a protogen head (which is no larger than a 6-inch box) and leave enough space for the camera to see the cosplayer's face.
- The system should use a small amount of power; it would be difficult for cosplayers to carry around large power
- Image Processing Pipeline (Section III-B)
- Expression Recognizer (Section IV-A)
- Face Image Buffers (Section V-A)
- Face Reconstructor (Section V-B)
- HUB75 Driver (Section VI-B)
- Power Modulator (Section VI-C)
- HUB75 LED Boards

We evaluate the performance of the ProtoFacer based on its resource usage in Section VII. We then end the report with a discussion of the system's limitations and future work in Section VIII.

II. Physical Construction

A. Head Design

Fig. 1. Exterior of the Protogen Head (with the tinted visor removed)

How I Found the Fandom (Bunsen)

- FNAF made me a bunny-obsessed furry back in 6th grade
- Started in algorithms, moved to hardware and performance engineering
- Made an FPGA-controlled protogen for a final project because why not
- Built another protogen because I didn't like my old one

photographer: @JustDragony (Telegram)

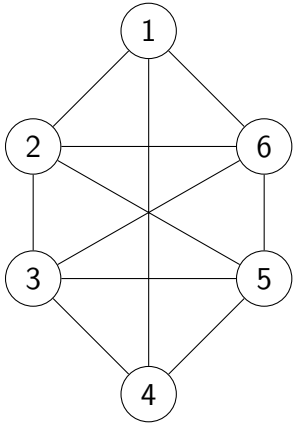
Fursuits are Planar Graphs

Bunsen's V1 head pattern

Unsigned Long's tail pattern

Fursuits are Planar Graphs

- Graph:
  - Vertices labeled $1, 2, \ldots, n$
  - Edges go between two vertices
  - At most $n \cdot (n-1)/2 = \underline{O(n^2)}$ edges[a]

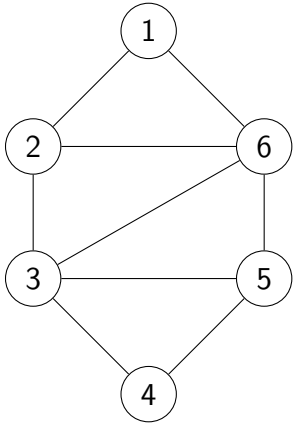


[a] Assuming simple graphs

Fursuits are Planar Graphs

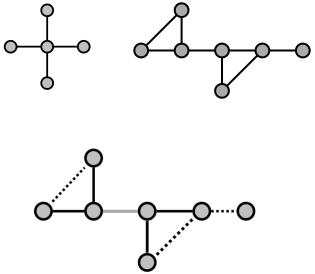
- Graph:
  - Vertices labeled $1, 2, \ldots, n$
  - Edges go between two vertices
  - At most $n \cdot (n-1)/2 = \underline{O(n^2)}$ edges[a]
- Planar graphs: graphs that can be drawn without crossing edges

[a] Assuming simple graphs



Planar Graphs

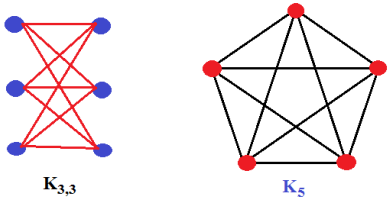
- Deleting vertices or contracting/deleting edges can't turn a planar graph into a non-planar graph.



diagrams from Wikimedia

Planar Graphs

- Deleting vertices or contracting/deleting edges can't turn a planar graph into a non-planar graph.
- **Wagner's Theorem**:
  A graph is planar if and only if you cannot form a K_5 or $K_{3,3}$ using deletions and contractions.
- K_5 are $K_{3,3}$ the **forbidden graph minors** of the family of planar graphs.



diagrams from Wikimedia

Forbidden Minors

- H is a **graph minor** of G if H can be obtained from G by deleting edges, isolated vertices **or** contracting edges.
- A family $\mathcal{F}$ of graphs is **closed under taking minors** if every minor of a graph in $\mathcal{F}$ also belongs to $\mathcal{F}$.
- **Robertson–Seymour theorem**:
  Every family of graphs that is closed under taking minors can be defined by a finite set of forbidden minors.

Planar Graph Sparsity

- Simple graphs have $O(n^2)$ edges
- Planar graphs have $\leq 3n - 6 = \underline{O(n)}$ edges![a]
- Fursuits are planar graphs
- Seam count $= O(\text{Fabric patch count})$

[a] Assuming faces have degree ≥ 3

Planar Graph Sparsity

- Simple graphs have $O(n^2)$ edges
- Planar graphs have $\leq 3n - 6 = \underline{O(n)}$ edges![a]
- Fursuits are planar graphs
- Seam count = O(Fabric patch count)
- Each patch you add only introduces a constant number of new seams you have to sew!

Top: Bunsen's V1 pattern with 40 pieces total (not all shown)

Bottom: Bunsen's V2 pattern with 19 pieces total

[a] Assuming faces have degree ≥ 3

Planar Graph Sparsity

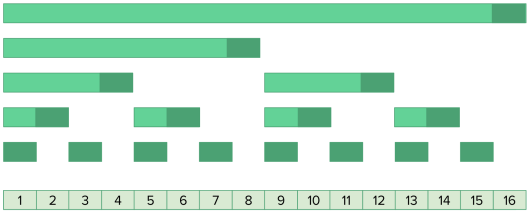
- Simple graphs have $O(n^2)$ edges
- Planar graphs have $\leq 3n - 6 = \underline{O(n)}$ edges![a]
- Fursuits are planar graphs
- Seam count $= O$(Fabric patch count)
- Each patch you add only introduces a constant number of new seams you have to sew!
- Therefore, it is totally reasonable to have 64 patches for your fursuit /j

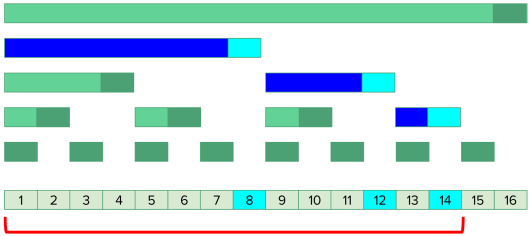
A part of Long's pattern with ~16 pieces

[a] Assuming faces have degree ≥ 3

Fenwick Tree[1]

- Binary-Indexed Tree
- Fast prefix queries in $O(\log n)$!





[1] Boris Ryabko (1989).

Splay?

splay 1 of 3 **verb**

splā

splayed; splaying; splays

Synonyms of *splay* ›

transitive verb

1 : to cause to spread outward

2 : to make oblique : **BEVEL**

intransitive verb

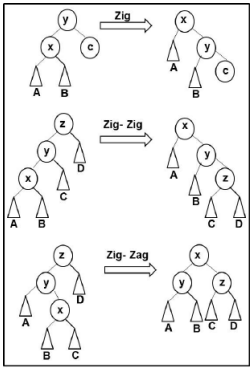
1 : to extend apart or outward especially in an awkward manner

2 : **SLOPE, SLANT**

https://en.wikifur.com/wiki/Furpile

Splay Tree[2]

- After every operation, splay the searched node to the root of the tree using double-rotations.
- Suspected to be optimal for any sequence of BST operations up to a constant factor (Dynamic Optimality Conjecture)

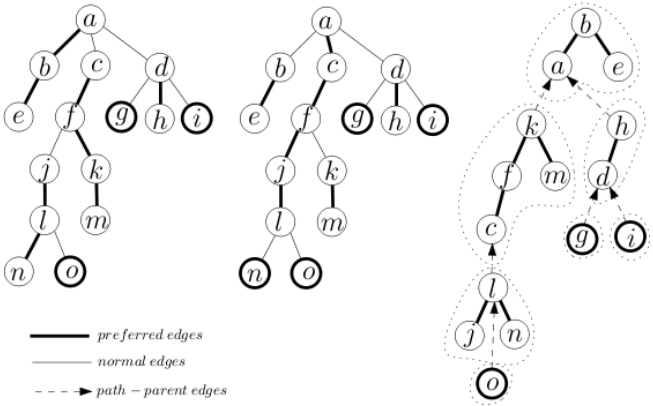


Trabelsi, Zouheir & Zeidan, Safaa & Masud, Mehedy & Ghoudi, Kilani. (2015). Statistical Dynamic Splay Tree Filters.

[2]Sleator, Daniel D.; Tarjan, Robert E. (1985).

Link-Cut Tree[3]

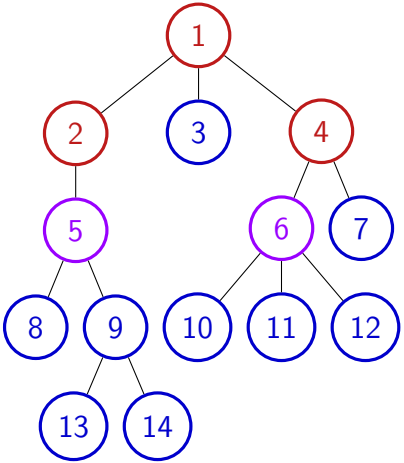
- Maintains a set of rooted trees
- Amortized $O(\log n)$ link, cut and find-root at any node
- Improves Dinic's Algorithm (for max-flow) from $O(V^2E)$ to $O(VE \log V)$.



[3]Sleator, D. D.; Tarjan, R. E. (1983).

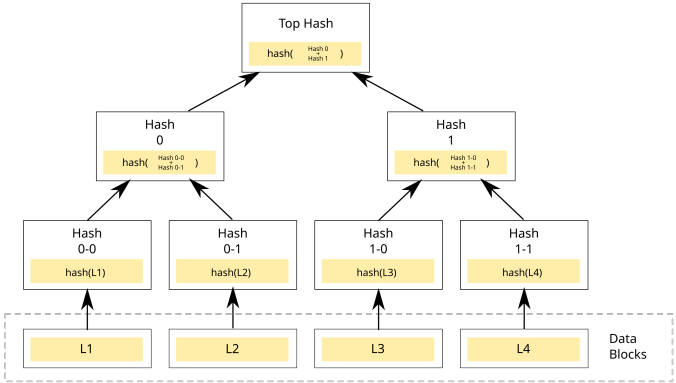
Macro-Micro Tree Decomposition

- Node is micro if it has less than $O(log(n))$ descendants (else macro)
- Node is macro leaf if it is macro and all its children are micro
- Subtree is microtree if its parent is a macro leaf
- There are at most $O(n/\log(n))$ macro leaves
  - Macro leaves have $O(\log(n))$ decendants, and do not share decendants
- There are $O(n^{1/c})$ distinct microtree shapes
  - Brute force on all microtree shapes



Merkle Tree[4]

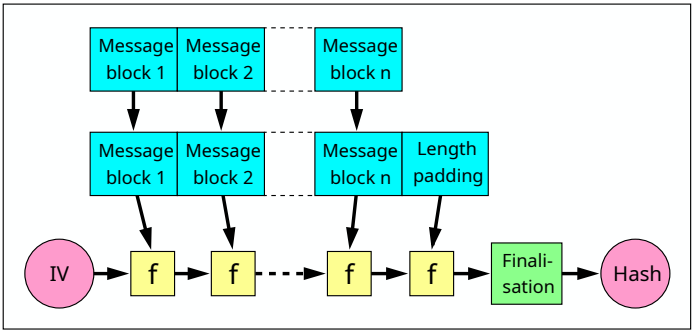
- Tree of hashes, widely used for data verification
  - Cryptographic schemes
  - P2P file sharing
  - Distrbuted filesystems
- Each internal node hashes the concatenated hashes of their children.
- Only recomputes $O(\log n)$ hashes when data changes!



[4] Merkle, R. C. (1979)

Message Digests

- Also called “hash functions”
- Takes any size input, makes fixed-size output
- Ideally, different inputs make different outputs
- Hard to find two inputs with same output for cryptographic hash functions
- e.g. MD5, SHA, BLAKE

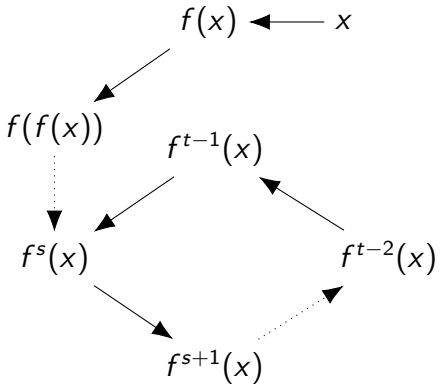


Tortoise and Hare: The Cycle Finding Problem

- Take some function $f : [n] \to [n]$, and some starting value x.
  - Notation: $[n]$ is the numbers $\{1, \ldots, n\}$, and f is a function that takes a number from $[n]$, and outputs a number in $[n]$

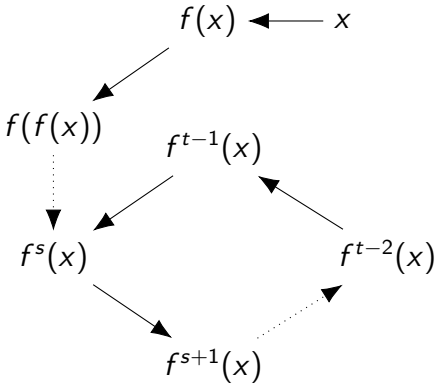
Tortoise and Hare: The Cycle Finding Problem

- Take some function $f : [n] \to [n]$, and some starting value x.
  - Notation: $[n]$ is the numbers $\{1, \ldots, n\}$, and f is a function that takes a number from $[n]$, and outputs a number in $[n]$
- By pidgeonhole principle, the sequence $x, f(x), f(f(x)), \ldots, f^i(x), \ldots$ will repeat



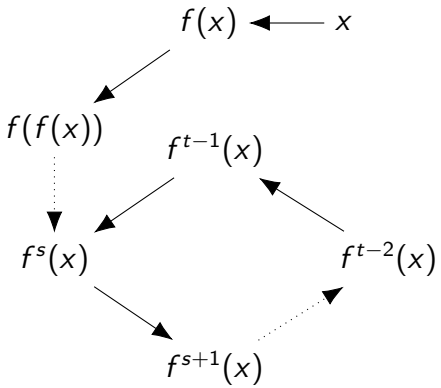
Tortoise and Hare: The Cycle Finding Problem

- Take some function $f : [n] \to [n]$, and some starting value x.
  - Notation: $[n]$ is the numbers $\{1, \ldots, n\}$, and f is a function that takes a number from $[n]$, and outputs a number in $[n]$
- By pidgeonhole principle, the sequence $x, f(x), f(f(x)), \ldots, f^i(x), \ldots$ will repeat
- Find s, t such that $f^s(x) = f^{s+t}(x)$



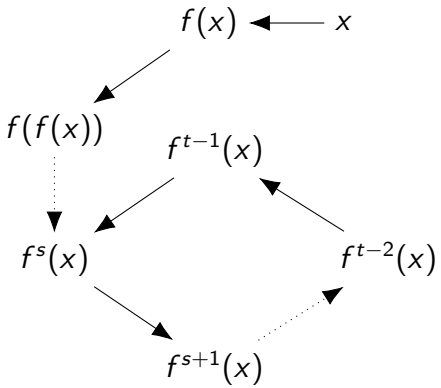
Tortoise and Hare: Floyd's Algorithm

- Algorithm description
  - Tortoise and hare start with $x_t := x$ and $x_h := x$.
  - When tortoise computes $x_t := f(x_t)$, hare computes $x_h := f(f(x_h))$.



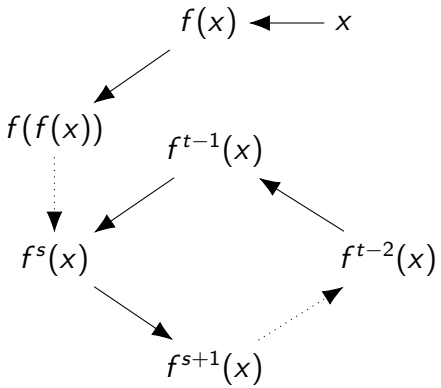
Tortoise and Hare: Floyd's Algorithm

- Algorithm description
  - Tortoise and hare start with $x_t := x$ and $x_h := x$.
  - When tortoise computes $x_t := f(x_t)$, hare computes $x_h := f(f(x_h))$.
- Algorithm analysis
  - Eventually both tortoise and hare will enter the cycle
  - When both are in the cycle, the hare will catch up to the tortoise



Tortoise and Hare: Floyd's Algorithm

- Algorithm description
  - Tortoise and hare start with $x_t := x$ and $x_h := x$.
  - When tortoise computes $x_t := f(x_t)$, hare computes $x_h := f(f(x_h))$.
- Algorithm analysis
  - Eventually both tortoise and hare will enter the cycle
  - When both are in the cycle, the hare will catch up to the tortoise
- Used in Pollard's rho algorithm



Tortoise and Hare: Pollard's Kangaroo

How Long Does it Take to Catch a Wild Kangaroo?

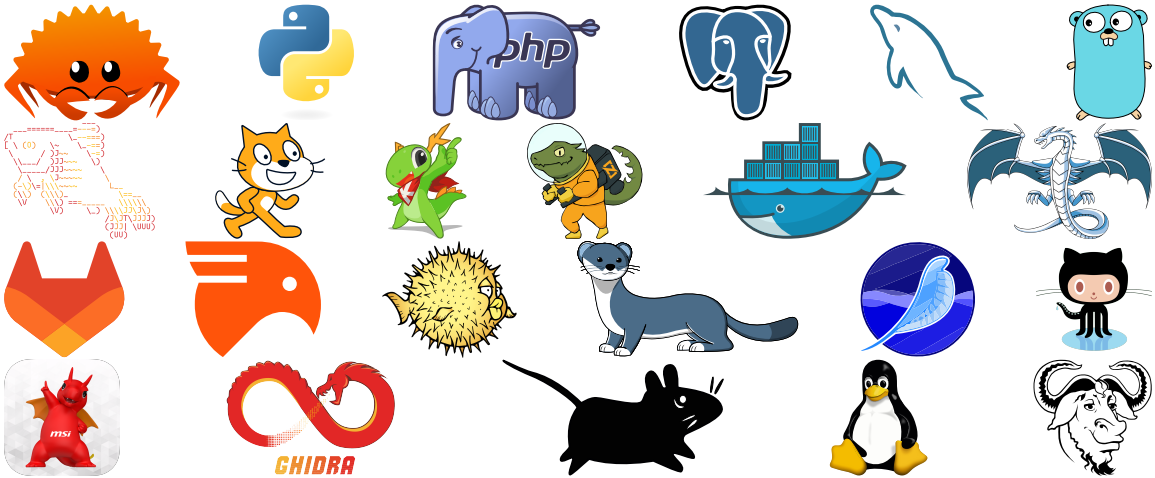
Ravi Montenegro * Prasad Tetali †

November 7, 2010

Abstract

We develop probabilistic tools for upper and lower bounding the expected time until two independent random walks on $\mathbb{Z}$ intersect each other. This leads to the first sharp analysis of a non-trivial Birthday attack, proving that Pollard's Kangaroo method solves the discrete logarithm problem $g^x = h$ on a cyclic group in expected time $(2 + o(1))\sqrt{b-a}$ for an average $x \in [a, b]$. Our methods also resolve a conjecture of Pollard's, by showing that the same bound holds when step sizes are generalized from powers of 2 to powers of any fixed n.

Animal Computing Mascots



Animal Computing Mascots: Trans Rights!

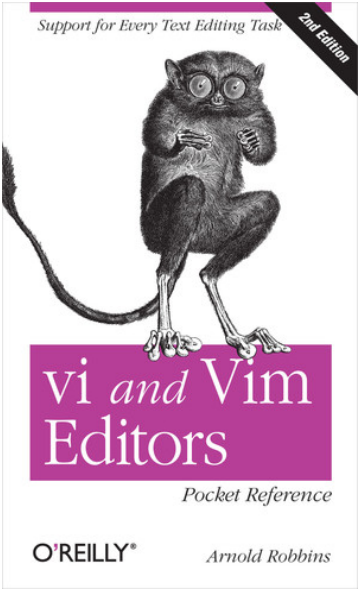
Animal Computing Mascots: Unofficial Mascot of C++

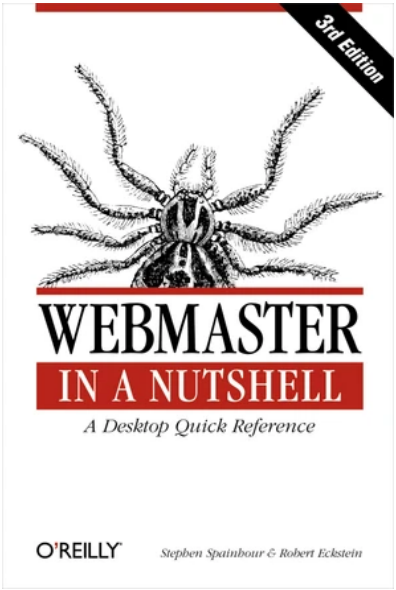
Animal Computing Mascots: Powershell...

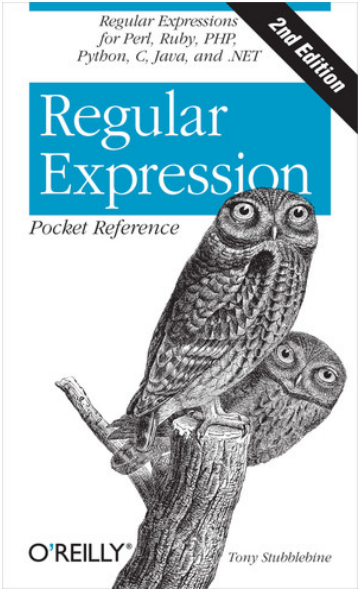
Animal Computing Mascots: WSL

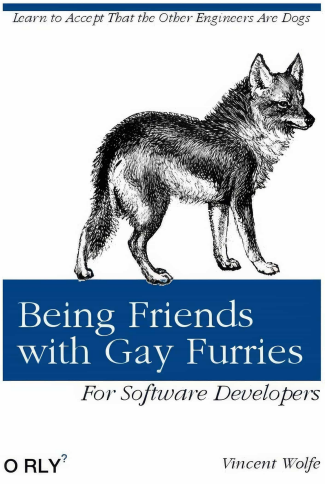
From: Richard Stallman
Subject: WSL
Date: Mon, 23 Jan 2023 22:50:01 -0500

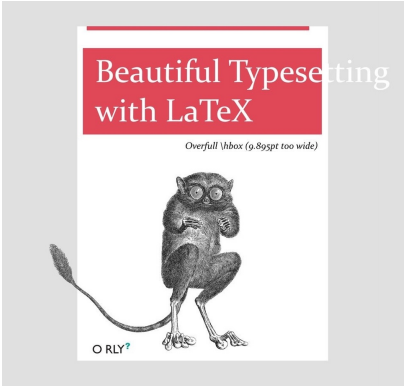
```
[[[ To any NSA and FBI agents reading my email: please consider    ]]]
[[[ whether defending the US Constitution against all enemies,     ]]]
[[[ foreign or domestic, requires you to follow Snowden's example. ]]]

How about pronouncing (and writing) "WSL" as "weasel"?

--
Dr Richard Stallman (https://stallman.org)
Chief GNUisance of the GNU Project (https://gnu.org)
Founder, Free Software Foundation (https://fsf.org)
Internet Hall-of-Famer (https://internethalloffame.org)
```

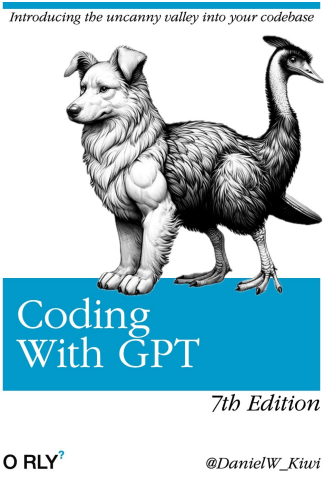

Support for Every Text Editing Task
2nd Edition
vi and Vim Editors
Pocket Reference
O'REILLY
Arnold Robbins


3rd Edition
WEBMASTER
IN A NUTSHELL
A Desktop Quick Reference
O'REILLY
Stephen Spainhour & Robert Eckstein


Regular Expressions for Perl, Ruby, PHP, Python, C, Java, and .NET
2nd Edition
Regular Expression
Pocket Reference
O'REILLY
Tony Stubblebine

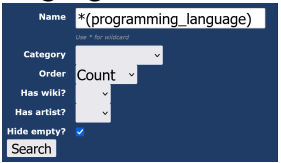

Learn to Accept That the Other Engineers Are Dogs
Being Friends with Gay Furries
For Software Developers
O RLY?
Vincent Wolfe


Beautiful Typesetting with LaTeX
Overfull \hbox (9.895pt too wide)
O RLY?


Introducing the uncanny valley into your codebase
Coding With GPT
7th Edition
O RLY?
@DanielW_Kiwi

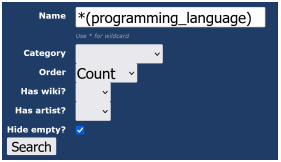
E621 Programming Language Tags

- How many posts exist for different programming languages?



E621 Programming Language Tags

- How many posts exist for different programming languages?

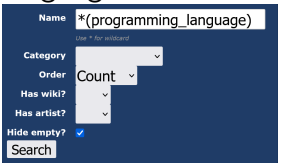


January 2026

Count	Name	
21	? python_(programming_language)	edit | history
9	? c_(programming_language)	edit | history
7	? zig_(programming_language)	edit | history
4	? c++_(programming_language)	edit | history
4	? rust_(programming_language)	edit | history
3	? java_(programming_language)	edit | history
1	? c_sharp_(programming_language)	edit | history
1	? go_(programming_language)	edit | history
1	? r_(programming_language)	edit | history
1	? php_(programming_language)	edit | history
1	? fortran_(programming_language)	edit | history

E621 Programming Language Tags

- How many posts exist for different programming languages?



June 2026

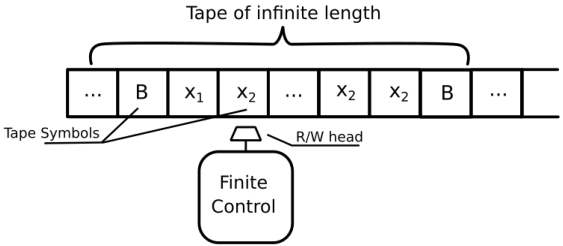
Count	Name	
21	? python_(programming_language)	edit | history
10	? c_(programming_language)	edit | history
9	? zig_(programming_language)	edit | history
5	? java_(programming_language)	edit | history
4	? c++_(programming_language)	edit | history
4	? rust_(programming_language)	edit | history
2	? fortran_(programming_language)	edit | history
1	? css_(programming_language)	edit | history
1	? go_(programming_language)	edit | history
1	? r_(programming_language)	edit | history
1	? php_(programming_language)	edit | history
1	? c_sharp_(programming_language)	edit | history

C (+1), Java (+2), Fortran (+1), CSS (+1, first one!)
Zig (+2 explicit)

Hoppers!

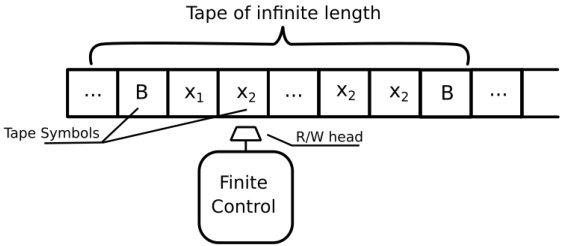
A Brief Introduction to Busy Beavers

- Turing Machine (TM): A finite state machine reading and writing to an infinite tape of symbols. Basically simulates modern computers.



A Brief Introduction to Busy Beavers

- Turing Machine (TM): A finite state machine reading and writing to an infinite tape of symbols. Basically simulates modern computers.
- TM halts if it reaches a halting state.**Halting problem** is well known to be undecidable.



A Brief Introduction to Busy Beavers

- What if we only look at TMs that **will halt**? How many steps does it take for all halting TMs to finish their computation?

A Brief Introduction to Busy Beavers

- What if we only look at TMs that **will halt**? How many steps does it take for all halting TMs to finish their computation?
- For a halting TM $\mathcal{M}$ with n states that can write $\{0, 1\}$, $\mathcal{M}$ is an n-state Busy Beaver if it runs for the maximum number of steps across all halting n-state TMs.

A Brief Introduction to Busy Beavers

- What if we only look at TMs that **will halt**? How many steps does it take for all halting TMs to finish their computation?
- For a halting TM $\mathcal{M}$ with n states that can write $\{0, 1\}$, $\mathcal{M}$ is an n-state Busy Beaver if it runs for the maximum number of steps across all halting n-state TMs.
- $S(n) =$ that maximum number of steps

A Brief Introduction to Busy Beavers

- How do we compute $S(n)$?

[5] Radó, Tibor (May 1962). "On non-computable functions" (PDF). Bell System Technical Journal. 41 (3): 877–884. doi:10.1002/j.1538-7305.1962.tb00480.x

A Brief Introduction to Busy Beavers

- How do we compute $S(n)$?
- You don't. $S(n)$ is an non-computable function[5].

[5] Radó, Tibor (May 1962). "On non-computable functions" (PDF). Bell System Technical Journal. 41 (3): 877–884. doi:10.1002/j.1538-7305.1962.tb00480.x

A Brief Introduction to Busy Beavers

- How do we compute $S(n)$?
- You don't. $S(n)$ is an non-computable function[5].
- But we can still try to figure out what $S(n)$ is!

[5] Radó, Tibor (May 1962). "On non-computable functions" (PDF). Bell System Technical Journal. 41 (3): 877–884. doi:10.1002/j.1538-7305.1962.tb00480.x

A Brief Introduction to Busy Beavers

- How do we compute $S(n)$?
- You don't. $S(n)$ is an non-computable function[5].
- But we can still try to figure out what $S(n)$ is!
- What do you think comes next in the following sequence?

n	2	3	4	5	6
$S(n)$	6	21	107	47176870	???

[5] Radó, Tibor (May 1962). "On non-computable functions" (PDF). Bell System Technical Journal. 41 (3): 877–884. doi:10.1002/j.1538-7305.1962.tb00480.x

The Difficulty of Computing $S(n)$

- In order to show that $S(n) \leq X$, one must show that every TM with n states either
  - Halts in at most X steps, or
  - Does not halt.

The Difficulty of Computing $S(n)$

- In order to show that $S(n) \leq X$, one must show that every TM with n states either
  - Halts in at most X steps, or
  - Does not halt.
- What if we construct TMs that halt if and only if some other problem is true?
- If an n-state TM $\mathcal{M}$ halts if and only if some conjecture P is true, then proving $S(n) = X$ means that $\mathcal{M}$ either halts in X steps or never halts.

Cryptids

- Coined by Shawn Ligocki in 2023.
  A **cryptid** is a TM that halts if and only if some hard conjecture is true/false.
- Cryptids exist for many hard mathematical conjectures
  - Collatz Conjecture
  - Goldbach Conjecture
  - Riemann Hypothesis
  - Consistency of PA and ZF set theory

Made by Lauren (2025), from Busy Beaver Challenge Wiki

3SUM-Hardness

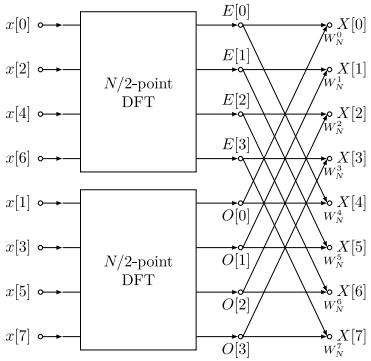
- 3SUM: given n numbers, can you find 3 that sum to 0?
- Solvable in $O(n^2)$ by hashing
- Assumed to not have a (significantly) faster solution
- Some problems are at least as hard to solve as 3SUM, and are called 3SUM-hard
  - Given lines in a plane, do any 3 intersect at a point?
  - Given triangles in a plane, does their union have a hole?
- Proof says, for any 3SUM instance, do some fast algorithm to transform into a 3SUM-hard instance whose solution matches the orignal problem

3SUM-Hardness

- 3SUM: given n numbers, can you find 3 that sum to 0?
- Solvable in $O(n^2)$ by hashing
- Assumed to not have a (significantly) faster solution
- Some problems are at least as hard to solve as 3SUM, and are called 3SUM-hard
  - Given lines in a plane, do any 3 intersect at a point?
  - Given triangles in a plane, does their union have a hole?
- Proof says, for any 3SUM instance, do some fast algorithm to transform into a 3SUM-hard instance whose solution matches the orignal problem
- Note: If numbers are integers in $[-m, m]$, solvable in $O(n + m \log m)$ via FFT

Fast Furrier Transform

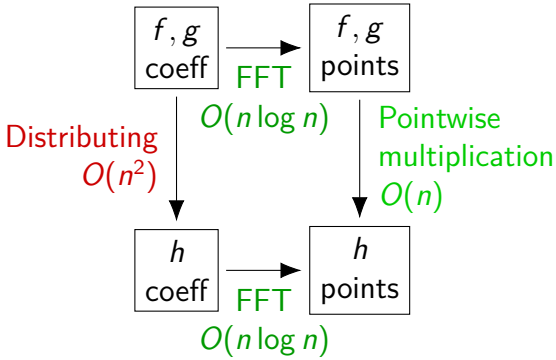
- A Fourier Transform (FT) decomposes a signal into its fundamental frequencies
- Typically computed with complex numbers
- Also has discrete version (DFT)
- Fast Fourier Transform (FFT) computes DFT in $O(n \log n)$
- Leads to pretty "butterfly diagrams"
- $\text{FT}(\text{Convolution}(f, g)) = \text{PointMult}(\text{FT}(f), \text{FT}(g))$



An FFT butterfly diagram

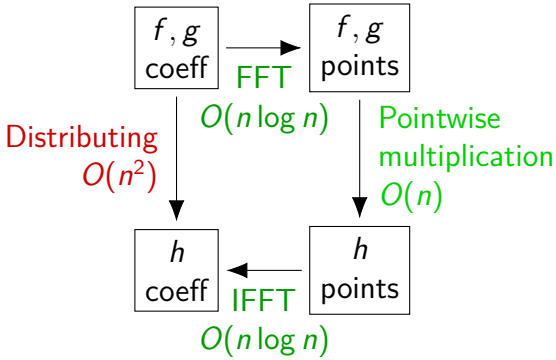
Fast Furrier Transform: Inverse FFT?

- Multiplying polynomials of degree n
  - Distributing would take $O(n^2)$
  - FFT takes $O(n \log n)$
  - Pointwise multiplication takes $O(n)$



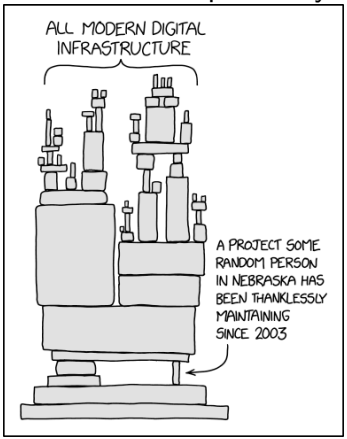
Fast Furrier Transform: Inverse FFT?

- Multiplying polynomials of degree n
  - Distributing would take $O(n^2)$
  - FFT takes $O(n \log n)$
  - Pointwise multiplication takes $O(n)$
- FFT is (approximately) its own inverse
  - No proof here: not a math panel
  - IFFT takes $O(n \log n)$
  - The Anti-Furry Transform is just a Furry Transform in disguise :3



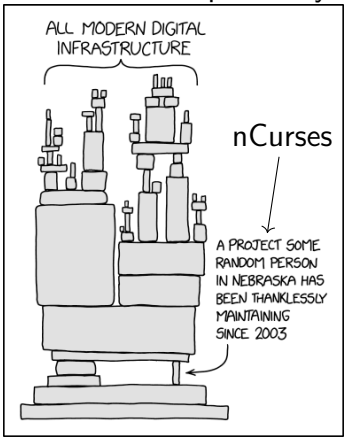
nCurses

xkcd 2347: Dependency



nCurses

xkcd 2347: Dependency




```
apt-cache rdepends --installed --recurse
libncurses6 | less
```

Data Structures & Algorithms @ Anthrocon 2026

Critters, Cryptids, nCurses

Bunsen Bitti, Unsigned Long

July 3, 2026