Data Structures & Algorithms @ ANE 2026

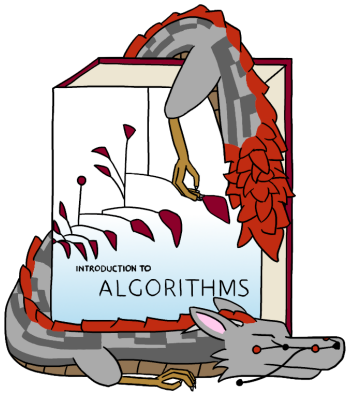
The Dominance Frontier

Bunsen Bitti, Unsigned Long

January 17, 2026

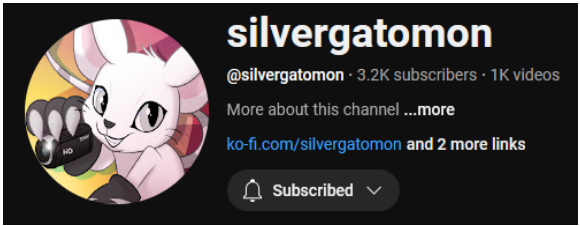
Overview

- 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 to do nothing with, so I made a fursona

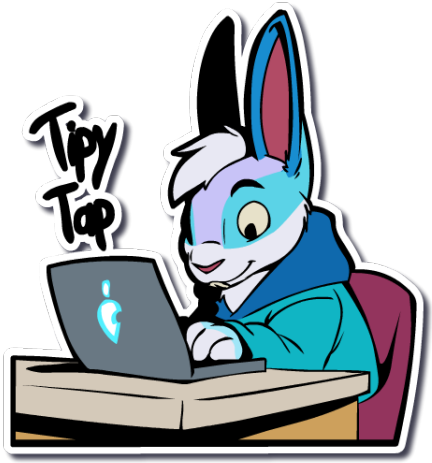
How I Found the Fandom (Bunsen)

- 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)

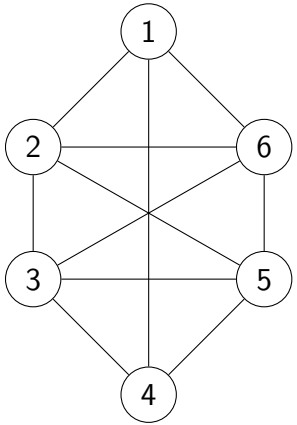
Fursuits are Planar Graphs

Bunsen's head pattern

Unsigned Long's tail pattern

Fursuits are Planar Graphs

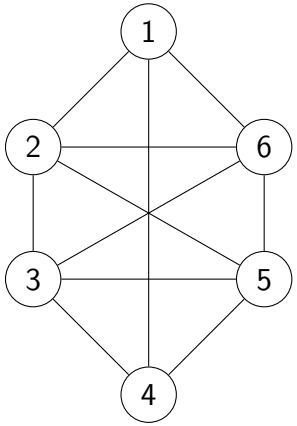
- Graph:
  - Vertices labeled $1, 2, \ldots, n$
  - Edges go between two vertices
  - At most $n \cdot (n-1)/2$ edges[a]



[a] Assuming simple graphs

Fursuits are Planar Graphs

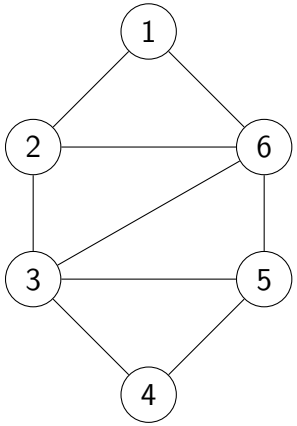
- Graph:
  - Vertices labeled $1, 2, \ldots, n$
  - Edges go between two vertices
  - At most $n \cdot (n-1)/2 = \underline{O(n^2)}$ edges[a]



[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
  - At most $3n - 6 = \underline{O(n)}$ edges![b]
  - Fursuits are planar graphs
  - Seam count $= O$(Fabric patch count)

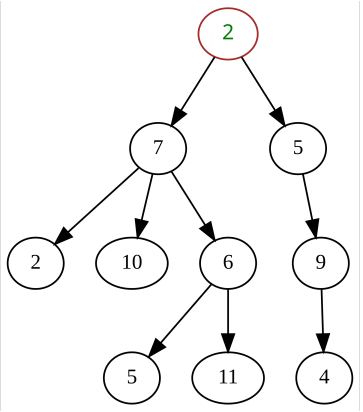


[a] Assuming simple graphs

[b] Assuming faces have degree ≥ 3

Trees!

- Trees are the building block of most data structures.
- Every node has one parent, except for the root which has no parent.
- Nodes with no children are leaves, nodes with children are internal nodes.



By Paddy3118 - Own work, CC BY-SA 4.0,

https://commons.wikimedia.org/w/index.php?curid=83223854

A-Z of Trees (non-exhaustive)

AVL Trees

B Trees

Cartesian Trees

Decision Tree

Exponential Trees

Fenwick Trees

Gomory-Hu Tree

H-Tree

Interval Tree

Judy Array

k-d Tree

Link-Cut Tree

Merkle Trees

N Tree

Order Statistics Tree

PQ Tree

Quadtree

Red Black Trees

Splay Trees

Tango Trees

Universal B Trees

Vantage Point Tree

Wallace Tree

X-fast Trie

Y-fast Trie

Zip Trees

~~Subsections~~ Subtrees

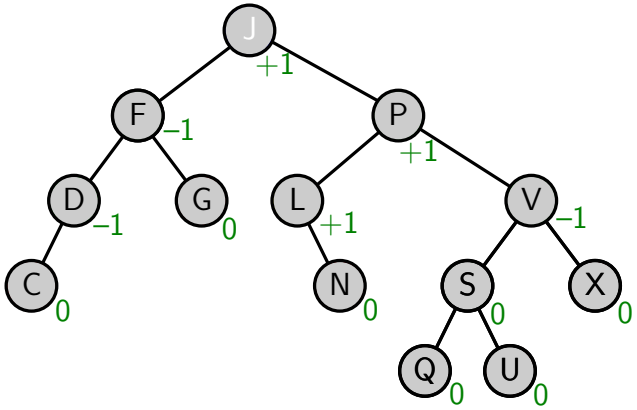
- ARSTZ : Balanced Binary Search Trees (BST)
- BCMUXY : Other Binary Trees
- IKQV : Spatial Data Structures
- EFGJLP : Niche data structures
- DHNOW : Non-data structures

~~Subsections~~ Subtrees

- **ARS**TZ : Balanced Binary Search Trees (BST)
- BCMUXY : Other Binary Trees
- IKQV : Spatial Data Structures
- EFGJ**L**P : Niche data structures
- **D**H**N**OW : Non-data structures

AVL Trees[1]

- First $O(\log n)$ self-balancing binary search tree!
- Ensures the height of its children do not differ by > 1



https://commons.wikimedia.org/w/index.php?curid=49182185

[1] Adelson-Velsky, Georgy; Landis, Evgenii (1962)

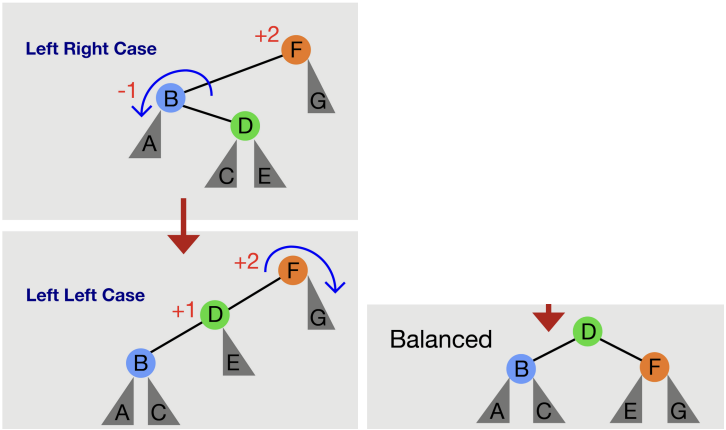
AVL Trees (Rotation)

- Imbalances are fixed by rotations, fundamental local operations for many balanced BSTs

/r/furry_irl/comments/1dumfg3/

AVL Trees (Rotation)

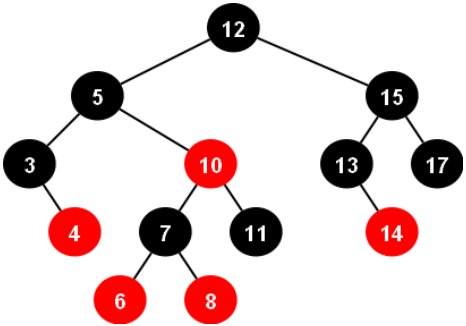
- Imbalances are fixed by rotations, fundamental local operations for many balanced BSTs



https://pages.cs.wisc.edu/~qingyi/

Red Black Tree[2]

- Balanced BST, red-black coloring chosen because it looked best on the laser printer they used.
- All root-to-leaf paths have the same number of black nodes
- No red node has a red parent



https://pages.cs.wisc.edu/~wyoungjun/

[2]Guibas, Leonidas J.; Sedgewick, Robert (1978).

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

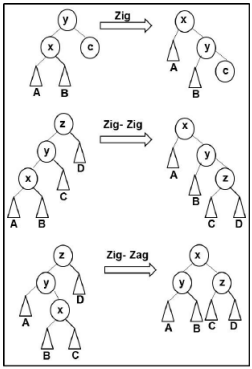
1 : to extend apart or outward especially in an awkward manner

2 : **SLOPE, SLANT**

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

Splay Tree[3]

- 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)

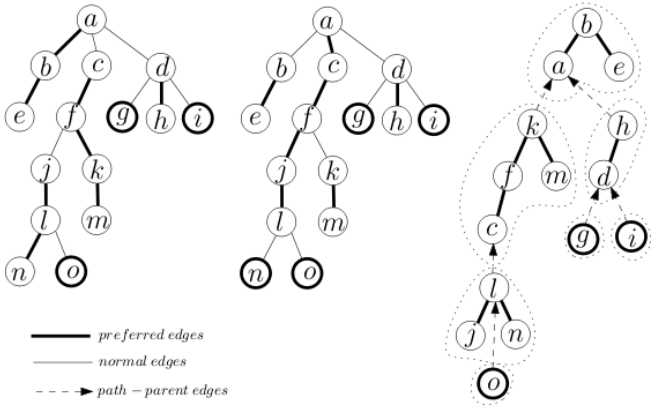


Trabelsi, Zouheir & Zeidan, Safaa & Masud, Mehedy & Ghoudi, Kilani. (2015). Statistical Dynamic Splay Tree Filters.

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

Link-Cut Tree[4]

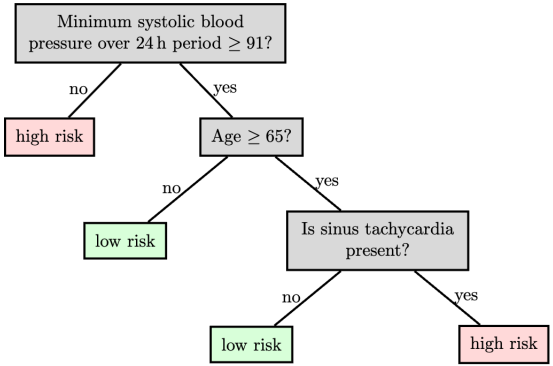
- Maintains a set of rooted trees
- Amortized $O(\log n)$ link, cut and find-root at any node
- Improves Dinic's Algorithm (for max-flow) from $O(V^2E)$ to $O(VE \log V)$.



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

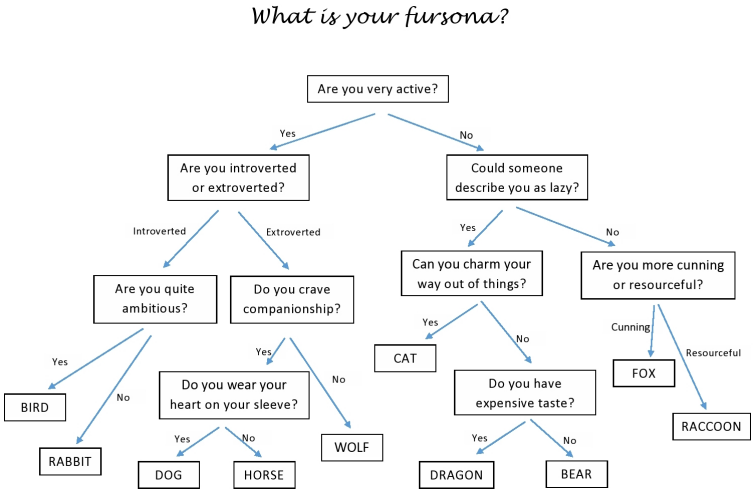
Decision Trees

- Each node represents a query, each leaf represents a category/decision.
- Commonly used in decision analysis and machine learning
- Fuzzy Decision Trees are also a thing



MIT 6.390 Intro ML Course Notes and Breiman, Friedman, Olshen, Stone (1984)

(Furry) Decision Trees (r/furry/comments/4gxm01/)

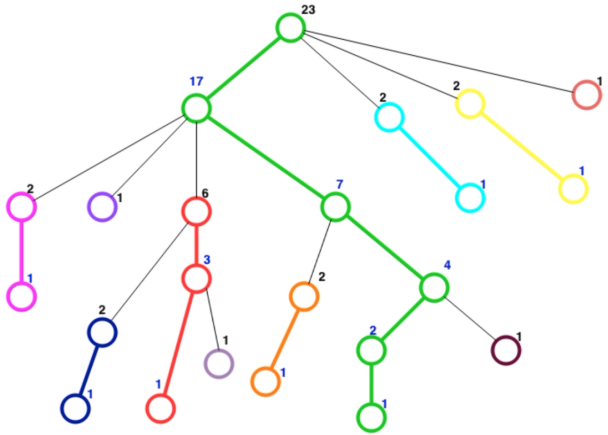


Missing N-tree

- Seriously I could not find any tree data structure that starts with N

Tree Decomposition

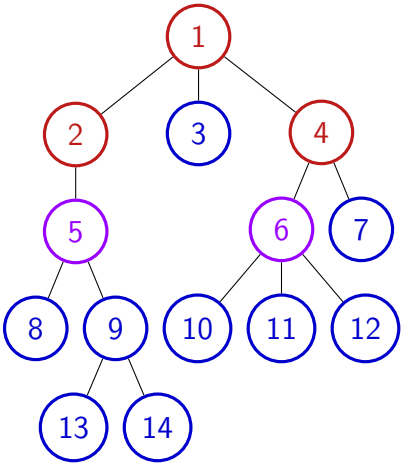
- Breaks down a rooted tree into smaller subparts to more efficiently solve subproblems
- Heavy-Light Decomposition
  - Separates edges into heavy ($\geq 1/2$ subtree size) and light ($< 1/2$ subtree size).
  - Every path from root to leaf contains $\leq log_2(n)$ light edges.



https://www.naukri.com/code360/library/heavy-light-decomposition-hld

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
  - Tree shape count is exponential in node count
  - Number of microtree nodes is logarithmic in n
  - log and exp cancel out to $O(n^{1/c})$



Instruction Set Architecture

- CPU Programs = lots of instructions that the CPU steps through to execute
- ISA is how the CPU interprets those instructions.
- Examples: x86, RISC-V etc.
- There are **a lot** of instructions.

x86 ISA

ADD — Add

Opcode	Instruction	Op/En	64-bit Mode	Compat/Leg Mode	Description
04 ib	ADD AL, imm8	I	Valid	Valid	Add imm8 to AL.
05 iw	ADD AX, imm16	I	Valid	Valid	Add imm16 to AX.
05 id	ADD EAX, imm32	I	Valid	Valid	Add imm32 to EAX.
REX.W + 05 id	ADD RAX, imm32	I	Valid	N.E.	Add imm32 sign-extended to 64-bits to RAX.
80 /0 ib	ADD r/m8, imm8	MI	Valid	Valid	Add imm8 to r/m8.
REX + 80 /0 ib	ADD r/m8*, imm8	MI	Valid	N.E.	Add sign-extended imm8 to r/m8.
81 /0 iw	ADD r/m16, imm16	MI	Valid	Valid	Add imm16 to r/m16.
81 /0 id	ADD r/m32, imm32	MI	Valid	Valid	Add imm32 to r/m32.
REX.W + 81 /0 id	ADD r/m64, imm32	MI	Valid	N.E.	Add imm32 sign-extended to 64-bits to r/m64.
83 /0 ib	ADD r/m16, imm8	MI	Valid	Valid	Add sign-extended imm8 to r/m16.
83 /0 ib	ADD r/m32, imm8	MI	Valid	Valid	Add sign-extended imm8 to r/m32.
REX.W + 83 /0 ib	ADD r/m64, imm8	MI	Valid	N.E.	Add sign-extended imm8 to r/m64.
00 /r	ADD r/m8, r8	MR	Valid	Valid	Add r8 to r/m8.
REX + 00 /r	ADD r/m8*, r8*	MR	Valid	N.E.	Add r8 to r/m8.
01 /r	ADD r/m16, r16	MR	Valid	Valid	Add r16 to r/m16.
01 /r	ADD r/m32, r32	MR	Valid	Valid	Add r32 to r/m32.
REX.W + 01 /r	ADD r/m64, r64	MR	Valid	N.E.	Add r64 to r/m64.
02 /r	ADD r8, r/m8	RM	Valid	Valid	Add r/m8 to r8.

x86 ISA

FADD/FADDP/FIADD — Add

Opcode	Instruction	64-Bit Mode	Compat/Leg Mode	Description
D8 /0	FADD m32fp	Valid	Valid	Add m32fp to ST(0) and store result in ST(0).
DC /0	FADD m64fp	Valid	Valid	Add m64fp to ST(0) and store result in ST(0).
D8 C0+i	FADD ST(0), ST(i)	Valid	Valid	Add ST(0) to ST(i) and store result in ST(0).
DC C0+i	FADD ST(i), ST(0)	Valid	Valid	Add ST(i) to ST(0) and store result in ST(i).
DE C0+i	FADDP ST(i), ST(0)	Valid	Valid	Add ST(0) to ST(i), store result in ST(i), and pop the register stack.
DE C1	FADDP	Valid	Valid	Add ST(0) to ST(1), store result in ST(1), and pop the register stack.
DA /0	FIADD m32int	Valid	Valid	Add m32int to ST(0) and store result in ST(0).
DE /0	FIADD m16int	Valid	Valid	Add m16int to ST(0) and store result in ST(0).

FMUL/FMULP/FIMUL — Multiply

Opcode	Instruction	64-Bit Mode	Compat/Leg Mode	Description
D8 /1	FMUL m32fp	Valid	Valid	Multiply ST(0) by m32fp and store result in ST(0).
DC /1	FMUL m64fp	Valid	Valid	Multiply ST(0) by m64fp and store result in ST(0).
D8 C8+i	FMUL ST(0), ST(i)	Valid	Valid	Multiply ST(0) by ST(i) and store result in ST(0).
DC C8+i	FMUL ST(i), ST(0)	Valid	Valid	Multiply ST(i) by ST(0) and store result in ST(i).
DE C8+i	FMULP ST(i), ST(0)	Valid	Valid	Multiply ST(i) by ST(0), store result in ST(i), and pop the register stack
DE C9	FMULP	Valid	Valid	Multiply ST(1) by ST(0), store result in ST(1), and pop the register stac
DA /1	FIMUL m32int	Valid	Valid	Multiply ST(0) by m32int and store result in ST(0).
DE /1	FIMUL m16int	Valid	Valid	Multiply ST(0) by m16int and store result in ST(0).

x86 ISA

VADDPH — Add Packed FP16 Values

Instruction En Bit Mode Flag Support Instruction En Bit Mode Flag Support 64/32 CPUID Feature Instruction En Bit Mode Flag CPUID Feature Instruction En Bit Mode Flag Op/ 64/32 CPUID Feature Instruction En Bit Mode Flag 64/32 CPUID Feature Instruction En Bit Mode Flag CPUID Feature Instruction En Bit Mode Flag Op/ 64/32 CPUID Feature		Support		Description
EVEX.128.NP.MAP5.W0 58 /r VADDPH xmm1{k1}{z}, xmm2, xmm3/m128/m16bcst	A	V/V	AVX512-FP16 AVX512VL	Add packed FP16 value from xmm3/m128/m16bcst to xmm2, and store result in xmm1 subject to writemask k1.
EVEX.256.NP.MAP5.W0 58 /r VADDPH ymm1{k1}{z}, ymm2, ymm3/m256/m16bcst	A	V/V	AVX512-FP16 AVX512VL	Add packed FP16 value from ymm3/m256/m16bcst to ymm2, and store result in ymm1 subject to writemask k1.
EVEX.512.NP.MAP5.W0 58 /r VADDPH zmm1{k1}{z}, zmm2, zmm3/m512/m16bcst {er}	A	V/V	AVX512-FP16	Add packed FP16 value from zmm3/m512/m16bcst to zmm2, and store result in zmm1 subject to writemask k1.

VMULPH — Multiply Packed FP16 Values

Instruction En bit Mode Flag Support Instruction En bit Mode Flag Support 64/32 CPUID Feature Instruction En bit Mode Flag CPUID Feature Instruction En bit Mode Flag Op/ 64/32 CPUID Feature Instruction En bit Mode Flag 64/32 CPUID Feature Instruction En bit Mode Flag CPUID Feature Instruction En bit Mode Flag Op/ 64/32 CPUID Feature		Support		Description
EVEX.128.NP.MAP5.W0 59 /r VMULPH xmm1{k1}{z}, xmm2, xmm3/m128/m16bcst	A	V/V	AVX512-FP16 AVX512VL	Multiply packed FP16 values from xmm3/m128/m16bcst to xmm2 and store the result in xmm1 subject to writemask k1.
EVEX.256.NP.MAP5.W0 59 /r VMULPH ymm1{k1}{z}, ymm2, ymm3/m256/m16bcst	A	V/V	AVX512-FP16 AVX512VL	Multiply packed FP16 values from ymm3/m256/m16bcst to ymm2 and store the result in ymm1 subject to writemask k1.
EVEX.512.NP.MAP5.W0 59 /r VMULPH zmm1{k1}{z}, zmm2, zmm3/m512/m16bcst {er}	A	V/V	AVX512-FP16	Multiply packed FP16 values in zmm3/m512/m16bcst with zmm2 and store the result in zmm1 subject to writemask k1.

x86 ISA

VFMADD132PH/VFNMADD132PH/VFMADD213PH/VFNMADD213PH/VFMADD231PH/VFNMADD231PH — Fused Multiply-Add of Packed FP16 Values

Instruction En Bit Mode Flag Support Instruction En Bit Mode Flag Support 64/32 CPUID Feature Instruction En Bit Mode Flag CPUID Feature Instruction En Bit Mode Flag Op/ 64/32 CPUID Feature Instruction En Bit Mode Flag 64/32 CPUID Feature Instruction En Bit Mode Flag CPUID Feature Instruction En Bit Mode Flag Op/ 64/32 CPUID Feature		Support		Description
EVEX.128.66.MAP6.W0 98 /r VFMADD132PH xmm1{k1}{z}, xmm2, xmm3/m128/m16bcst	A	V/V	AVX512-FP16 AVX512VL	Multiply packed FP16 values from xmm1 and xmm3/m128/m16bcst, add to xmm2, and store the result in xmm1.
EVEX.256.66.MAP6.W0 98 /r VFMADD132PH ymm1{k1}{z}, ymm2, ymm3/m256/m16bcst	A	V/V	AVX512-FP16 AVX512VL	Multiply packed FP16 values from ymm1 and ymm3/m256/m16bcst, add to ymm2, and store the result in ymm1.
EVEX.512.66.MAP6.W0 98 /r VFMADD132PH zmm1{k1}{z}, zmm2, zmm3/m512/m16bcst {er}	A	V/V	AVX512-FP16	Multiply packed FP16 values from zmm1 and zmm3/m512/m16bcst, add to zmm2, and store the result in zmm1.
EVEX.128.66.MAP6.W0 A8 /r VFMADD213PH xmm1{k1}{z}, xmm2, xmm3/m128/m16bcst	A	V/V	AVX512-FP16 AVX512VL	Multiply packed FP16 values from xmm1 and xmm2, add to xmm3/m128/ m16bcst, and store the result in xmm1.
EVEX.256.66.MAP6.W0 A8 /r VFMADD213PH ymm1{k1}{z}, ymm2, ymm3/m256/m16bcst	A	V/V	AVX512-FP16 AVX512VL	Multiply packed FP16 values from ymm1 and ymm2, add to ymm3/m256/ m16bcst, and store the result in ymm1.
EVEX.512.66.MAP6.W0 A8 /r VFMADD213PH zmm1{k1}{z}, zmm2, zmm3/m512/m16bcst {er}	A	V/V	AVX512-FP16	Multiply packed FP16 values from zmm1 and zmm2, add to zmm3/m512/ m16bcst, and store the result in zmm1.
EVEX.128.66.MAP6.W0 B8 /r VFMADD231PH xmm1{k1}{z}, xmm2, xmm3/m128/m16bcst	A	V/V	AVX512-FP16 AVX512VL	Multiply packed FP16 values from xmm2 and xmm3/m128/m16bcst, add to xmm1, and store the result in xmm1.
EVEX.256.66.MAP6.W0 B8 /r VFMADD231PH ymm1{k1}{z}, ymm2, ymm3/m256/m16bcst	A	V/V	AVX512-FP16 AVX512VL	Multiply packed FP16 values from ymm2 and ymm3/m256/m16bcst, add to ymm1, and store the result in ymm1.
EVEX.512.66.MAP6.W0 B8 /r VFMADD231PH zmm1{k1}{z}, zmm2, zmm3/m512/m16bcst {er}	A	V/V	AVX512-FP16	Multiply packed FP16 values from zmm2 and zmm3/m512/m16bcst, add to zmm1, and store the result in zmm1.
EVEX.128.66.MAP6.W0 9C /r VFNMADD132PH xmm1{k1}{z}, xmm2, xmm3/m128/m16bcst	A	V/V	AVX512-FP16 AVX512VL	Multiply packed FP16 values from xmm1 and xmm3/m128/m16bcst, and negate the value. Add this value to xmm2, and store the result in xmm1.
EVEX.256.66.MAP6.W0 9C /r VFNMADD132PH ymm1{k1}{z}, ymm2, ymm3/m256/m16bcst	A	V/V	AVX512-FP16 AVX512VL	Multiply packed FP16 values from ymm1 and ymm3/m256/m16bcst, and negate the value. Add this value to ymm2, and store the result in ymm1.
EVEX.512.66.MAP6.W0 9C /r VFNMADD132PH zmm1{k1}{z}, zmm2, zmm3/m512/m16bcst {er}	A	V/V	AVX512-FP16	Multiply packed FP16 values from zmm1 and zmm3/m512/m16bcst, and negate the value. Add this value to zmm2, and store the result in zmm1.
EVEX.128.66.MAP6.W0 AC /r VFNMADD213PH xmm1{k1}{z}, xmm2, xmm3/m128/m16bcst	A	V/V	AVX512-FP16 AVX512VL	Multiply packed FP16 values from xmm1 and xmm2, and negate the value. Add this value to xmm3/m128/m16bcst, and store the result in xmm1.
EVEX.256.66.MAP6.W0 AC /r VFNMADD213PH ymm1{k1}{z}, ymm2, ymm3/m256/m16bcst	A	V/V	AVX512-FP16 AVX512VL	Multiply packed FP16 values from ymm1 and ymm2, and negate the value. Add this value to ymm3/m256/m16bcst, and store the result in ymm1.
EVEX.512.66.MAP6.W0 AC /r VFNMADD213PH zmm1{k1}{z}, zmm2, zmm3/m512/m16bcst {er}	A	V/V	AVX512-FP16	Multiply packed FP16 values from zmm1 and zmm2, and negate the value. Add this value to zmm3/m512/m16bcst, and store the result in zmm1.
EVEX.128.66.MAP6.W0 BC /r VFNMADD231PH xmm1{k1}{z}, xmm2, xmm3/m128/m16bcst	A	V/V	AVX512-FP16 AVX512VL	Multiply packed FP16 values from xmm2 and xmm3/m128/m16bcst, and negate the value. Add this value to xmm1, and store the result in xmm1.
EVEX.256.66.MAP6.W0 BC /r VFNMADD231PH ymm1{k1}{z}, ymm2, ymm3/m256/m16bcst	A	V/V	AVX512-FP16 AVX512VL	Multiply packed FP16 values from ymm2 and ymm3/m256/m16bcst, and negate the value. Add this value to ymm1, and store the result in ymm1.
EVEX.512.66.MAP6.W0 BC /r VFNMADD231PH zmm1{k1}{z}, zmm2, zmm3/m512/m16bcst {er}	A	V/V	AVX512-FP16	Multiply packed FP16 values from zmm2 and zmm3/m512/m16bcst, and negate the value. Add this value to zmm1, and store the result in zmm1.

x86 ISA

VFMADDSUB132PH/VFMADDSUB213PH/VFMADDSUB231PH — Fused Multiply-AlternatingAdd/Subtract of Packed FP16 Values

Instruction En Bit Mode Flag Support Instruction En Bit Mode Flag Support 64/32 CPUID Feature Instruction En Bit Mode Flag CPUID Feature Instruction En Bit Mode Flag Op/ 64/32 CPUID Feature Instruction En Bit Mode Flag 64/32 CPUID Feature Instruction En Bit Mode Flag CPUID Feature Instruction En Bit Mode Flag Op/ 64/32 CPUID Feature		Support		Description
EVEX.128.66.MAP6.W0 96 /r VFMADDSUB132PH xmm1{k1}{z}, xmm2, xmm3/m128/m16bcst	A	V/V	AVX512-FP16 AVX512VL	Multiply packed FP16 values from xmm1 and xmm3/m128/ m16bcst, add/subtract elements in xmm2, and store the result in xmm1 subject to writemask k1.
EVEX.256.66.MAP6.W0 96 /r VFMADDSUB132PH ymm1{k1}{z}, ymm2, ymm3/m256/m16bcst	A	V/V	AVX512-FP16 AVX512VL	Multiply packed FP16 values from ymm1 and ymm3/m256/ m16bcst, add/subtract elements in ymm2, and store the result in ymm1 subject to writemask k1.
EVEX.512.66.MAP6.W0 96 /r VFMADDSUB132PH zmm1{k1}{z}, zmm2, zmm3/m512/m16bcst {er}	A	V/V	AVX512-FP16	Multiply packed FP16 values from zmm1 and zmm3/m512/ m16bcst, add/subtract elements in zmm2, and store the result in zmm1 subject to writemask k1.
EVEX.128.66.MAP6.W0 A6 /r VFMADDSUB213PH xmm1{k1}{z}, xmm2, xmm3/m128/m16bcst	A	V/V	AVX512-FP16 AVX512VL	Multiply packed FP16 values from xmm1 and xmm2, add/ subtract elements in xmm3/m128/m16bcst, and store the result in xmm1 subject to writemask k1.
EVEX.256.66.MAP6.W0 A6 /r VFMADDSUB213PH ymm1{k1}{z}, ymm2, ymm3/m256/m16bcst	A	V/V	AVX512-FP16 AVX512VL	Multiply packed FP16 values from ymm1 and ymm2, add/ subtract elements in ymm3/m256/m16bcst, and store the result in ymm1 subject to writemask k1.
EVEX.512.66.MAP6.W0 A6 /r VFMADDSUB213PH zmm1{k1}{z}, zmm2, zmm3/m512/m16bcst {er}	A	V/V	AVX512-FP16	Multiply packed FP16 values from zmm1 and zmm2, add/ subtract elements in zmm3/m512/m16bcst, and store the result in zmm1 subject to writemask k1.
EVEX.128.66.MAP6.W0 B6 /r VFMADDSUB231PH xmm1{k1}{z}, xmm2, xmm3/m128/m16bcst	A	V/V	AVX512-FP16 AVX512VL	Multiply packed FP16 values from xmm2 and xmm3/m128/ m16bcst, add/subtract elements in xmm1, and store the result in xmm1 subject to writemask k1.
EVEX.256.66.MAP6.W0 B6 /r VFMADDSUB231PH ymm1{k1}{z}, ymm2, ymm3/m256/m16bcst	A	V/V	AVX512-FP16 AVX512VL	Multiply packed FP16 values from ymm2 and ymm3/m256/ m16bcst, add/subtract elements in ymm1, and store the result in ymm1 subject to writemask k1.
EVEX.512.66.MAP6.W0 B6 /r VFMADDSUB231PH zmm1{k1}{z}, zmm2, zmm3/m512/m16bcst {er}	A	V/V	AVX512-FP16	Multiply packed FP16 values from zmm2 and zmm3/m512/ m16bcst, add/subtract elements in zmm1, and store the

x86 ISA

Notation	Odd Elements	Even Elements
132	dest = dest*src3+src2	dest = dest*src3-src2
231	dest = src2*src3+dest	dest = src2*src3-dest
213	dest = src2*dest+src3	dest = src2*dest-src3

KNOTD

KNOTW/KNOTB/KNOTQ/KNOTD—NOT Mask Register

Opcode/ Instruction	Op/En	64/32 bit Mode Support	CPUID Feature Flag	Description
VEX.L0.0F.W0 44 /r KNOTW k1, k2	RR	V/V	AVX512F	Bitwise NOT of 16 bits mask k2.
VEX.L0.66.0F.W0 44 /r KNOTB k1, k2	RR	V/V	AVX512DQ	Bitwise NOT of 8 bits mask k2.
VEX.L0.0F.W1 44 /r KNOTQ k1, k2	RR	V/V	AVX512BW	Bitwise NOT of 64 bits mask k2.
VEX.L0.66.0F.W1 44 /r KNOTD k1, k2	RR	V/V	AVX512BW	Bitwise NOT of 32 bits mask k2.

NeedsMoreGPUs • 4y ago

Two Star Pixel Defender

Of course it's AVX512. AVX512 is the most degenerate instruction set.

7 Share ···

Fast Furrier Transform

- Fourier Transform?
  - Typically computed with complex numbers
  - Transform signal between time and frequency domain

Fast Furrier Transform

- Fourier Transform?
  - Typically computed with complex numbers
  - Transform signal between time and frequency domain
- This is not a math panel!
  - See this 3Blue1Brown video:
    "But what is the Fourier Transform? A visual introduction." →

Fast Furrier Transform

- Fourier Transform?
  - Typically computed with complex numbers
  - Transform signal between time and frequency domain
- This is not a math panel!
  - See this 3Blue1Brown video:
    "But what is the Fourier Transform? A visual introduction." →

- This is an algorithms panel :3
  - We will use FFT to multiply polynomials in $O(n \log n)$ time

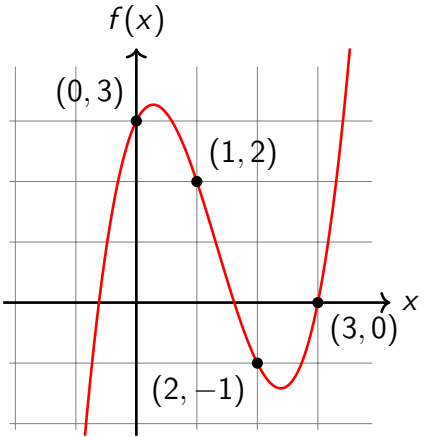
Fast Furrier Transform: Polynomials

coefficient | term | degree

$$f(x) = 3 + 2x - 4x^2 + x^3$$

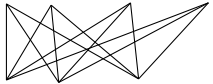
$$f(0) = 3$$
$$f(1) = 2$$
$$f(2) = -1$$
$$f(3) = 0$$

Evaluating a polynomial at a point takes $O(n)$ time



Fast Furrier Transform: Multiplication

$$f(x) = 3 + 2x - 4x^2 + x^3$$



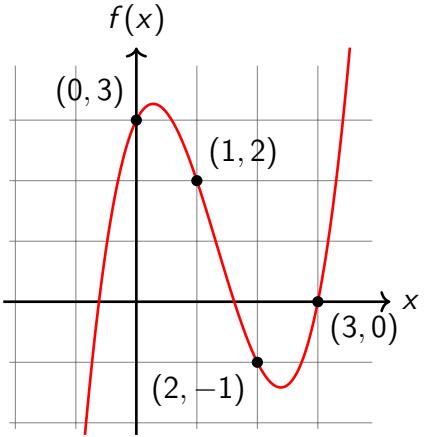
$$g(x) = 2 \quad - x \quad + x^2$$

$$h(x) = f(x) \cdot g(x) = 6 + x - 7x^2 + 8x^3 - 5x^4 + x^5$$

Multiplying two polynomials of degree n by distributing takes $O(n^2)$ time

Fast Furrier Transform: Polynomial Interpolation/Evaluation

- For a set of n points, there is a unique polynomial with degree less than n that passes through all the points
- Evaluating the polynomial at n different x values to find these points typically takes $O(n^2)$ time
- If we choose the x values cleverly, we can do better



Fast Furrier Transform: Evaluation at ± 1

$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots + a_{n-2} x^{n-2} + a_{n-1} x^{n-1}$$

Fast Furrier Transform: Evaluation at ± 1

$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots + a_{n-2} x^{n-2} + a_{n-1} x^{n-1}$$

$$f(1) = a_0 + a_1 + a_2 + a_3 + \cdots + a_{n-2} + a_{n-1}$$
$$f(-1) = a_0 - a_1 + a_2 - a_3 + \cdots + a_{n-2} - a_{n-1}$$

Fast Furrier Transform: Evaluation at ± 1

$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots + a_{n-2} x^{n-2} + a_{n-1} x^{n-1}$$

$$\begin{aligned} f(1) &= a_0 + a_1 + a_2 + a_3 + \cdots + a_{n-2} + a_{n-1} \\ f(-1) &= a_0 - a_1 + a_2 - a_3 + \cdots + a_{n-2} - a_{n-1} \end{aligned}$$

$$\begin{aligned} f(1) &= (a_0 + a_2 + \cdots + a_{n-2}) + (a_1 + a_3 + \cdots + a_{n-1}) \\ f(-1) &= (a_0 + a_2 + \cdots + a_{n-2}) - (a_1 + a_3 + \cdots + a_{n-1}) \end{aligned}$$

Fast Furrier Transform: Evaluation at ± 1

$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots + a_{n-2} x^{n-2} + a_{n-1} x^{n-1}$$

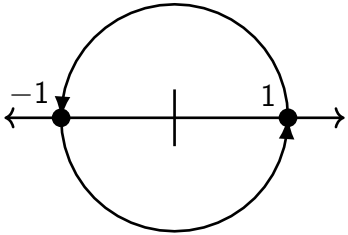
$$\begin{aligned} f(1) &= a_0 + a_1 + a_2 + a_3 + \cdots + a_{n-2} + a_{n-1} \\ f(-1) &= a_0 - a_1 + a_2 - a_3 + \cdots + a_{n-2} - a_{n-1} \end{aligned}$$

$$\begin{aligned} f(1) &= (a_0 + a_2 + \cdots + a_{n-2}) + (a_1 + a_3 + \cdots + a_{n-1}) \\ f(-1) &= (a_0 + a_2 + \cdots + a_{n-2}) - (a_1 + a_3 + \cdots + a_{n-1}) \end{aligned}$$

Number of operations is halved!

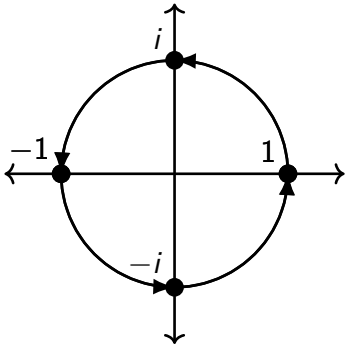
Fast Furrier Transform: Roots of Unity

- The even/odd trick works because repeatedly multiplying by -1 cycles between -1 and 1
- It'd be convenient if we had other values making this kind of cycle, and if these cycles were longer



Fast Furrier Transform: Roots of Unity

- The even/odd trick works because repeatedly multiplying by -1 cycles between -1 and 1
- It'd be convenient if we had other values making this kind of cycle, and if these cycles were longer
- Complex numbers give us values r where $r^n = 1$
- These are called roots of unity
- FFT evaluates f at $1, r, r^2, \ldots, r^{n-1}$



Fast Furrier Transform: Odd/Even Trick

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1}$$

Fast Furrier Transform: Odd/Even Trick

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1}$$

$$f_e(x) = a_0 \ + a_2 x^2 + a_4 x^4 + \cdots + a_{n-2} x^{n-2}$$
$$f_o(x) = a_1 x + a_3 x^3 + a_5 x^5 + \cdots + a_{n-1} x^{n-1}$$

Fast Furrier Transform: Odd/Even Trick

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1}$$

$$f_e(x) = a_0 \ + a_2 x^2 + a_4 x^4 + \cdots + a_{n-2} x^{n-2}$$
$$f_o(x) = a_1 x + a_3 x^3 + a_5 x^5 + \cdots + a_{n-1} x^{n-1}$$

Fast Furrier Transform: Odd/Even Trick

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1}$$

$$f_e(x) = a_0 \ + a_2 x^2 + a_4 x^4 + \cdots + a_{n-2} x^{n-2}$$
$$f_o(x) = a_1 x + a_3 x^3 + a_5 x^5 + \cdots + a_{n-1} x^{n-1}$$
$$f_o(x) = x(a_1 \ + a_3 x^2 + a_5 x^4 + \cdots + a_{n-1} x^{n-2})$$

Fast Furrier Transform: Odd/Even Trick

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1}$$

$$f_e(x) = a_0 \ + a_2 x^2 + a_4 x^4 + \cdots + a_{n-2} x^{n-2}$$
$$f_o(x) = a_1 x + a_3 x^3 + a_5 x^5 + \cdots + a_{n-1} x^{n-1}$$
$$f_o(x) = x(a_1 \ + a_3 x^2 + a_5 x^4 + \cdots + a_{n-1} x^{n-2})$$

Fast Furrier Transform: Odd/Even Trick

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1}$$

$$\begin{aligned} f_e(x) &= a_0 \ + a_2 x^2 + a_4 x^4 + \cdots + a_{n-2} x^{n-2} \\ f_o(x) &= a_1 x + a_3 x^3 + a_5 x^5 + \cdots + a_{n-1} x^{n-1} \\ f_o(x) &= x(a_1 \ + a_3 x^2 + a_5 x^4 + \cdots + a_{n-1} x^{n-2}) \end{aligned}$$

$$\begin{aligned} g_e(y) &= a_0 \ + a_2 y \ + a_4 y^2 + \cdots + a_{n-2} y^{n/2-1} \\ g_o(y) &= a_1 \ + a_3 y \ + a_5 y^2 + \cdots + a_{n-1} y^{n/2-1} \end{aligned}$$

$$f(x) = g_e(x^2) + x g_o(x^2)$$

Fast Furrier Transform: Divide and Conquer

$$f(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1} = g_e(x^2) + x g_o(x^2)$$
$$g_e(y) = a_0 + a_2 y + a_4 y^2 + \ldots + a_{n-2} y^{n/2-1}$$
$$g_o(y) = a_1 + a_3 y + a_5 y^2 + \ldots + a_{n-1} y^{n/2-1}$$

- We need to evaluate g_e and g_o at $y = x^2 =$

$$(1)^2, \quad (r)^2, \quad (r^2)^2, \ldots, \quad (r^{n/2-1})^2, \quad (r^{n/2})^2, \quad (r^{n/2+1})^2, \ldots, \quad (r^{n-1})^2$$

Fast Furrier Transform: Divide and Conquer

$$f(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1} = g_e(x^2) + x g_o(x^2)$$
$$g_e(y) = a_0 + a_2 y + a_4 y^2 + \ldots + a_{n-2} y^{n/2-1}$$
$$g_o(y) = a_1 + a_3 y + a_5 y^2 + \ldots + a_{n-1} y^{n/2-1}$$

- We need to evaluate g_e and g_o at $y = x^2 =$

$$\begin{array}{cccccccc} (1)^2, & (r)^2, & (r^2)^2, \ldots, & (r^{n/2-1})^2, & (r^{n/2})^2, & (r^{n/2+1})^2, \ldots, & (r^{n-1})^2 \\ 1, & r^2, & r^4, \ldots, & r^{n-2}, & r^n = 1, & r^2, \ldots, & r^{n-2} \end{array}$$

Fast Furrier Transform: Divide and Conquer

$$f(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1} = g_e(x^2) + x g_o(x^2)$$
$$g_e(y) = a_0 + a_2 y + a_4 y^2 + \ldots + a_{n-2} y^{n/2-1}$$
$$g_o(y) = a_1 + a_3 y + a_5 y^2 + \ldots + a_{n-1} y^{n/2-1}$$

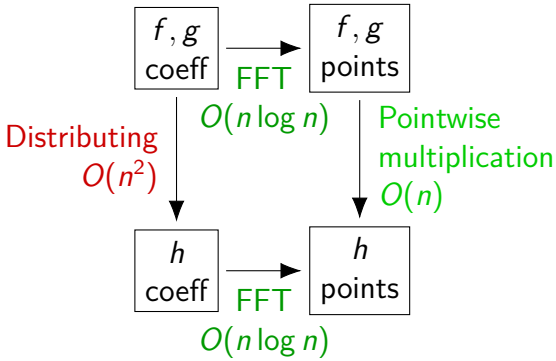
- We need to evaluate g_e and g_o at $y = x^2 =$

$$\begin{array}{cccccccc} (1)^2, & (r)^2, & (r^2)^2, \ldots, & (r^{n/2-1})^2, & (r^{n/2})^2, & (r^{n/2+1})^2, \ldots, & (r^{n-1})^2 \\ 1, & r^2, & r^4, \ldots, & r^{n-2}, & r^n = 1, & r^2, \ldots, & r^{n-2} \end{array}$$

- To evaluate f at n roots of unity, we evaluate g_e and g_o at $n/2$ roots of unity
- Recursion! $T(n) = 2T(n/2) + O(n) = O(n \log n)$

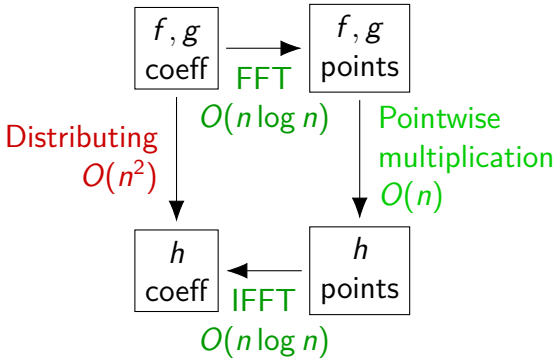
Fast Furrier Transform: Inverse FFT?

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



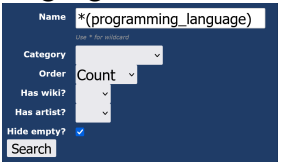
Fast Furrier Transform: Inverse FFT?

- 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



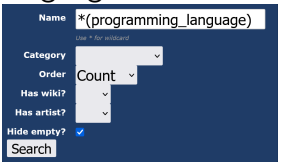
E621 Programming Language Tags

- How many posts exist for different programming languages?



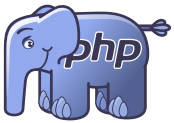
E621 Programming Language Tags

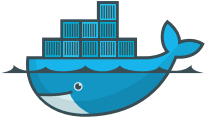
- How many posts exist for different programming languages?

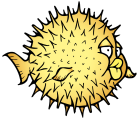


Count	Name	
21	? python_(programming_language)	edit | history
9	? c_(programming_language)	edit | history
7	? zig_(programming_language)	edit | history
4	? c++_(programming_language)	edit | history
4	? rust_(programming_language)	edit | history
3	? java_(programming_language)	edit | history
1	? c_sharp_(programming_language)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

Animal Computing Mascots







Animal Computing Mascots: Trans Rights!

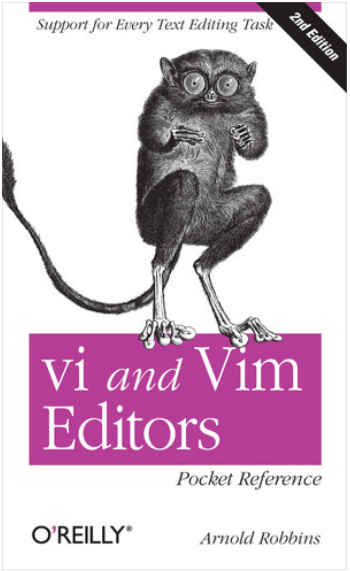
Animal Computing Mascots: Unofficial Mascot of C++

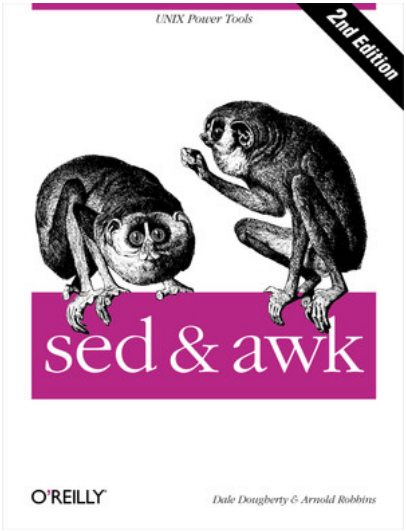
Animal Computing Mascots: Powershell...

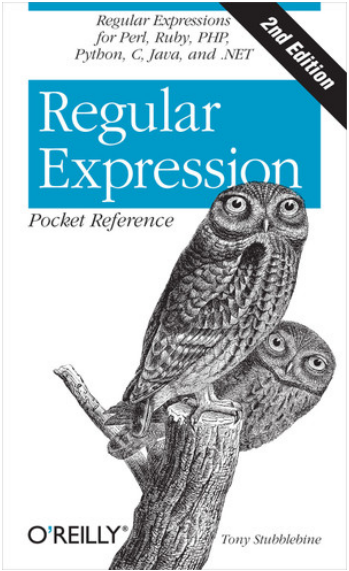
Animal Computing Mascots: WSL

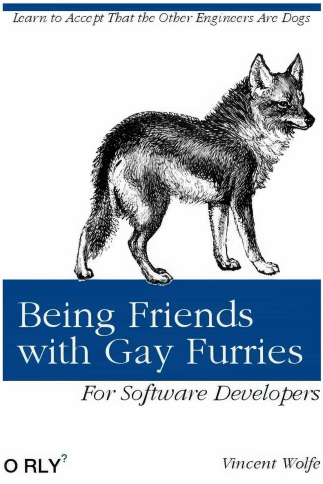
From: Richard Stallman
Subject: WSL
Date: Mon, 23 Jan 2023 22:50:01 -0500

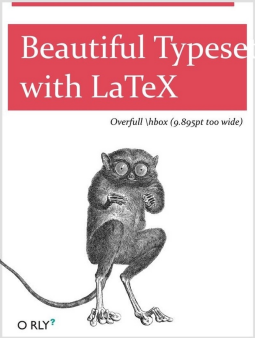
```
[[[ To any NSA and FBI agents reading my email: please consider    ]]]
[[[ whether defending the US Constitution against all enemies,     ]]]
[[[ foreign or domestic, requires you to follow Snowden's example. ]]]

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)
```

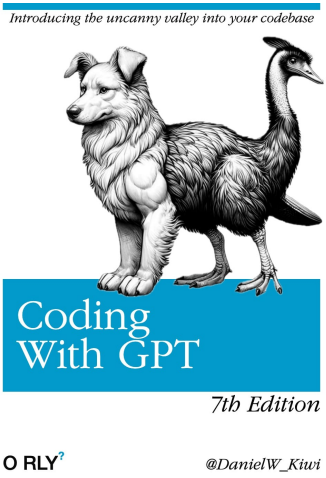

Support for Every Text Editing Task
2nd Edition
vi and Vim Editors
Pocket Reference
O'REILLY
Arnold Robbins


UNIX Power Tools
2nd Edition
sed & awk
O'REILLY
Dale Dougherty & Arnold Robbins


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

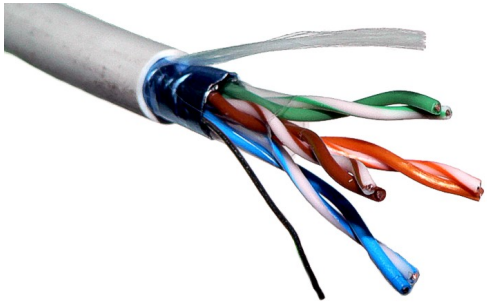

Beautiful Typesetting with LaTeX
Overfull \hbox (9.895pt too wide)
O RLY?


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

https://youtu.be/t9QugQkASdQ

Twisted Pair/Coaxial Cable[5]

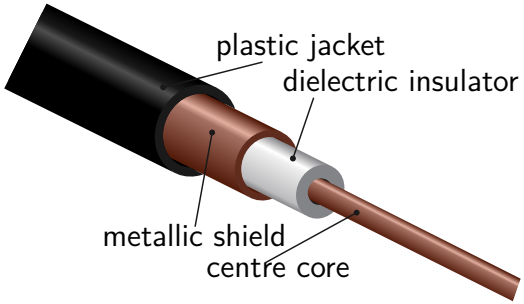
- Electric current makes magnetic fields
- Excessive magnetic fields causes noise and interference
- Idea: place reverse current close to forward current to have fields cancel
- Twisted pairs!



[5]Disclaimer: I am not an Electrical Engineer

Twisted Pair/Coaxial Cable[5]

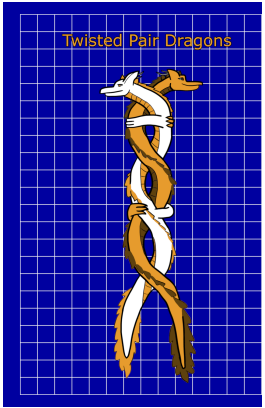
- Electric current makes magnetic fields
- Excessive magnetic fields causes noise and interference
- Idea: place reverse current close to forward current to have fields cancel
- Twisted pairs!
- To make the fields align and cancel even more, the two conductors can be made to share the same axis
- Coaxial cable!



[5]Disclaimer: I am not an Electrical Engineer

Twisted Pair/Coaxial Cable[5]

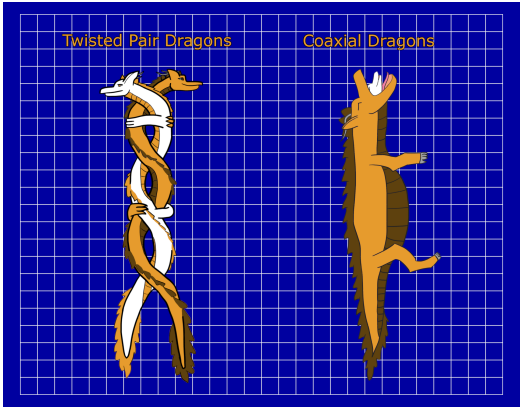
- Electric current makes magnetic fields
- Excessive magnetic fields causes noise and interference
- Idea: place reverse current close to forward current to have fields cancel
- Twisted pairs!
- To make the fields align and cancel even more, the two conductors can be made to share the same axis
- Coaxial cable!



[5]Disclaimer: I am not an Electrical Engineer

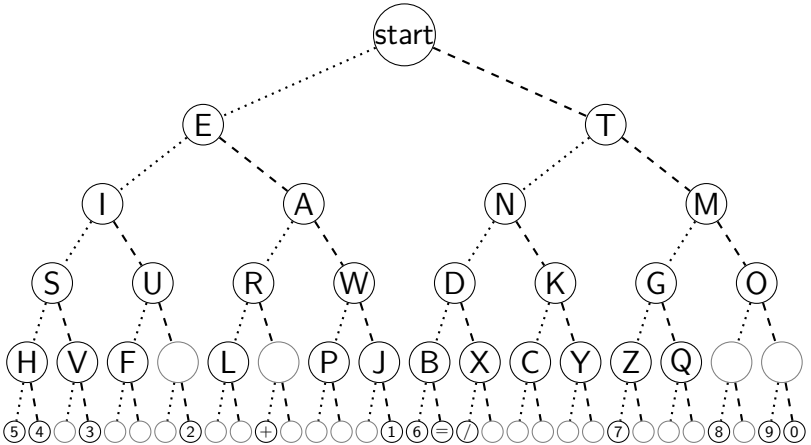
Twisted Pair/Coaxial Cable[5]

- Electric current makes magnetic fields
- Excessive magnetic fields causes noise and interference
- Idea: place reverse current close to forward current to have fields cancel
- Twisted pairs!
- To make the fields align and cancel even more, the two conductors can be made to share the same axis
- Coaxial cable!

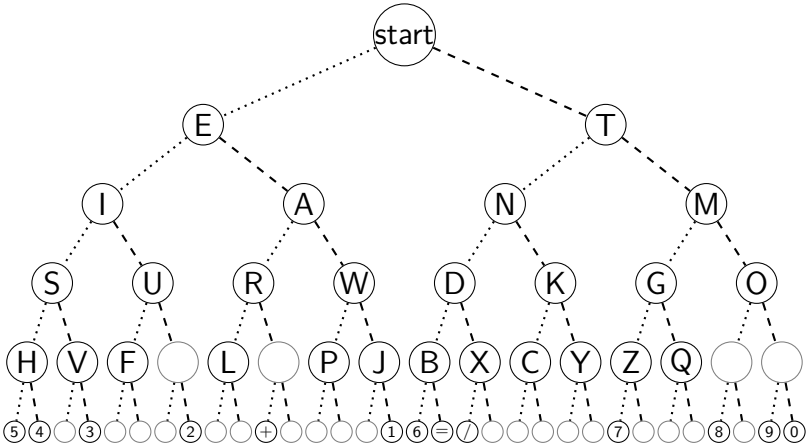


[5]Disclaimer: I am not an Electrical Engineer

Morse Code

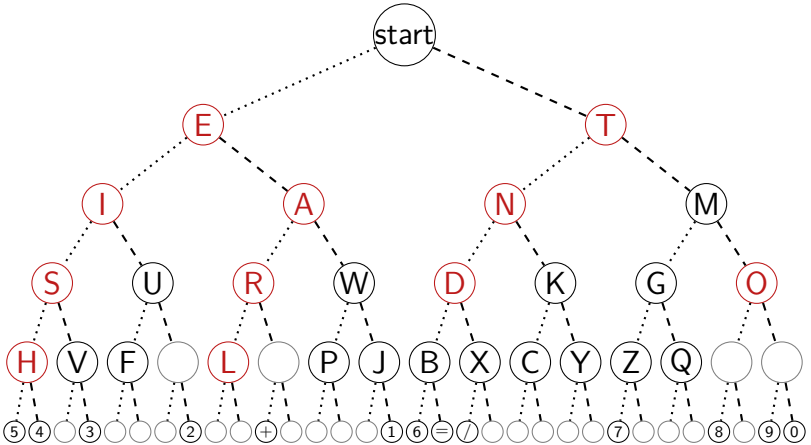


Morse Code



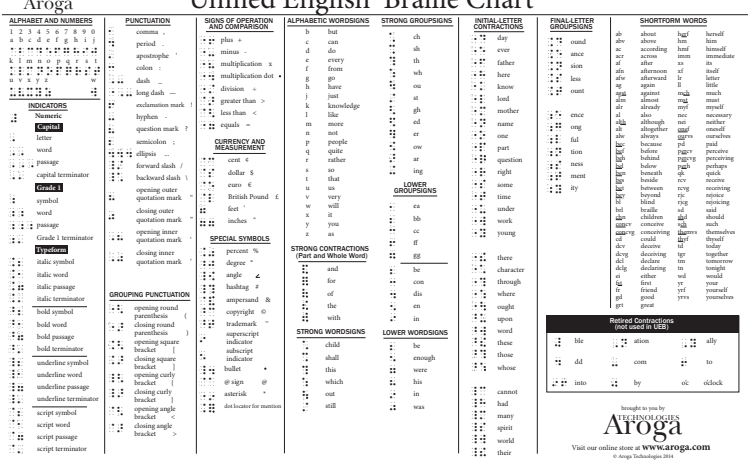
Letter	Frequency
E	12.7%
T	9.1%
A	8.2%
O	7.5%
I	7.0%
N	6.7%
S	6.3%
H	6.1%
R	6.0%
D	4.3%
L	4.0%

Morse Code



Letter	Frequency
E	12.7%
T	9.1%
A	8.2%
O	7.5%
I	7.0%
N	6.7%
S	6.3%
H	6.1%
R	6.0%
D	4.3%
L	4.0%

Braille


Unified English Braille Chart
Aroga
ALPHABET AND NUMBERS
1 2 3 4 5 6 7 8 9 0
a b c d e f g h i j
k l m n o p q r s t
u v x y z w
INDICATORS
Numeric
Capital
letter
word
passage
capital terminator
Grade 1
symbol
word
passage
Grade 1 terminator
Typeform
italic symbol
italic word
italic passage
italic terminator
bold symbol
bold word
bold passage
bold terminator
underline symbol
underline word
underline passage
underline terminator
script symbol
script word
script passage
script terminator
PUNCTUATION
comma ,
period .
apostrophe '
colon :
dash -
long dash —
exclamation mark !
hyphen -
question mark ?
semicolon ;
ellipsis …
forward slash /
backward slash \
opening outer quotation mark
closing outer quotation mark
opening inner quotation mark
closing inner quotation mark
GROUPING PUNCTUATION
opening round parenthesis (
closing round parenthesis)
opening square bracket [
closing square bracket]
opening curly bracket {
closing curly bracket }
opening angle bracket <
closing angle bracket >
SIGNS OF OPERATION AND COMPARISON
plus +
minus -
multiplication x
multiplication dot ·
division ÷
greater than >
less than <
equals =
CURRENCY AND MEASUREMENT
cent ¢
dollar $
euro €
British Pound £
feet '
inches "
SPECIAL SYMBOLS
percent %
degree °
angle ∠
hashtag #
ampersand &
copyright ©
trademark ™
superscript indicator
subscript indicator
bullet •
@ sign @
asterisk *
dot locator for mention
ALPHABETIC WORDSIGNS
b but
c can
d do
e every
f from
g go
h have
j just
k knowledge
l like
m more
n not
p people
q quite
r rather
s so
t that
u us
v very
w will
x it
y you
z as
STRONG CONTRACTIONS (Part and Whole Word)
and
for
of
the
with
STRONG WORDSIGNS
child
shall
this
which
out
still
STRONG GROUPSIGNS
ch
sh
th
wh
ou
st
gh
ed
er
ow
ar
ing
LOWER GROUPSIGNS
ea
bb
cc
ff
gg
be
con
dis
en
in
LOWER WORDSIGNS
be
enough
were
his
in
was
INITIAL-LETTER CONTRACTIONS
day
ever
father
here
know
lord
mother
name
one
part
question
right
some
time
under
work
young
there
character
through
where
ought
upon
word
these
those
whose
cannot
had
many
spirit
world
their
FINAL-LETTER GROUPSIGNS
ound
ance
sion
ount
ence
ong
ful
tion
ness
ment
ity
SHORTFORM WORDS
ab about ingf herself
abv above hm him
ac according hmf himself
acr across imm immediate
af after in its
afn afternoon xf itself
afw afterward lr latter
ag again ll little
agst against mch much
alm almost mst must
alr already myf myself
al also nec necessary
alth although ned neither
alt altogether nmf oneself
abv always ourvs ourselves
bec because pd paid
bef before perc perceive
beh behind perv perceiving
bel below perh perhaps
ben beneath qk quick
bes beside rcv receive
btw between rcvg receiving
byv beyond rjc rejoice
bl blind rjcg rejoicing
brl braille sd said
chd children shld should
cncv conceive sch such
cncvg conceiving themvs themselves
cd could thyf thyself
dcv deceive td today
dcvg deceiving tgr together
dcl declare tm tomorrow
dclg declaring tn tonight
et either wd would
fst first yr your
fr friend yrf yourself
gd good yrvs yourselves
grt great
Retired Contractions (not used in UEB)
ble
ation
ally
dd
com
to
into
by
o'c
o'clock
brought to you by
Aroga
TECHNOLOGIES
Visit our online store at www.aroga.com
© Aroga Technologies 2014

Braille

ALPHABET AND NUMBERS

1	2	3	4	5	6	7	8	9	0
a	b	c	d	e	f	g	h	i	j
k	l	m	n	o	p	q	r	s	t
u	v	x	y	z					w

Braille

ALPHABET AND NUMBERS

1	2	3	4	5	6	7	8	9	0
a	b	c	d	e	f	g	h	i	j
k	l	m	n	o	p	q	r	s	t
u	v	x	y	z					w

STRONG GROUPSIGNS

ch		gh	
sh		ed	
th		er	
wh		ow	
ou		ar	
st		ing	

Braille

ALPHABET AND NUMBERS

1	2	3	4	5	6	7	8	9	0
a	b	c	d	e	f	g	h	i	j
k	l	m	n	o	p	q	r	s	t
u	v	x	y	z					w

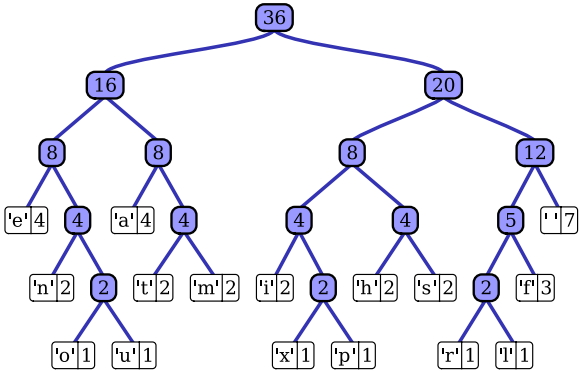
STRONG GROUPSIGNS

ch		gh	
sh		ed	
th		er	
wh		ow	
ou		ar	
st		ing	

"owo" is 

Huffman Coding

- More frequent characters get shorter encodings
- Used in .zip files and other compression algorithms
- Won't go over details due to time



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 $[-N, N]$, solvable in $O(n + N \log N)$ via FFT

VOL. 63, NO. 2, APRIL 1990

How Often Should You Beat Your Kids?

DON ZAGIER
University of Maryland
College Park, MD 20742

A result is proved which shows, roughly speaking, that one should beat one's kids every day except Sunday.

This note is a follow-up to the note "How to Beat Your Kids at Their Own Game," by K. Levasseur [1], in which the author proposes the following game to be played against one's two-year-old children: Starting with a deck consisting of n red cards and n black cards (in typical applications, $n = 26$), the cards are turned up one at a time, each player at each stage predicting the color of the card which is about to appear. The kid is supposed to guess "Red" or "Black" randomly with equal probability (this solves the problem of constructing a perfect random number generator), while you play what is obviously the optimal strategy—guessing randomly (or, if you prefer,

Multiplying this by the probability that $m(p) = m$ as computed above, we find finally

$$\begin{aligned}
\text{probability of winning} &\approx \frac{1}{2} + \sum_{m=0}^{\infty} \frac{m}{2n} e^{-m^2/4n} \left(\frac{1}{\sqrt{\pi}} \int_0^{m/2\sqrt{n}} e^{-u^2}\, du \right) \\
&\approx \frac{1}{2} + \int_0^{\infty} \frac{x}{2n} e^{-x^2/4n} \left(\frac{1}{\sqrt{\pi}} \int_0^{x/2\sqrt{n}} e^{-u^2}\, du \right) dx \\
&= \frac{1}{2} + \frac{1}{2\sqrt{\pi}} \int_0^{\infty} x e^{-x^2/4} \left(\int_0^{x/2} e^{-u^2}\, du \right) dx \\
&= \frac{1}{2} + \frac{1}{2\sqrt{\pi}} \int_0^{\infty} e^{-u^2} \left(\int_{2u}^{\infty} x e^{-x^2/4}\, dx \right) du \\
&= \frac{1}{2} + \frac{1}{2\sqrt{\pi}} \int_0^{\infty} e^{-u^2} \left(2e^{-u^2} \right) du \\
&= \frac{1}{2} + \frac{1}{2\sqrt{2}},
\end{aligned}$$

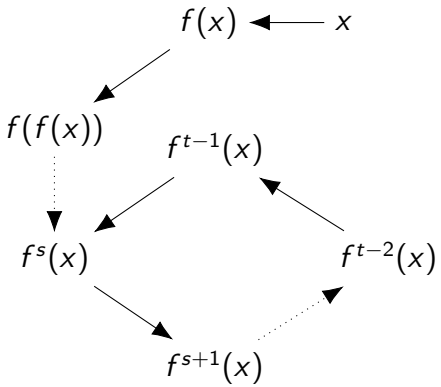
as claimed. This is very nearly 6/7, so the result of our paper can be conveniently implemented by beating one's kids on weekdays and Saturdays, but never on Sunday.

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 $[n]$

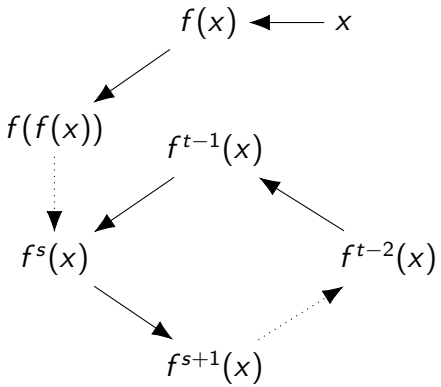
Tortoise and Hare: The Cycle Finding Problem

- 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



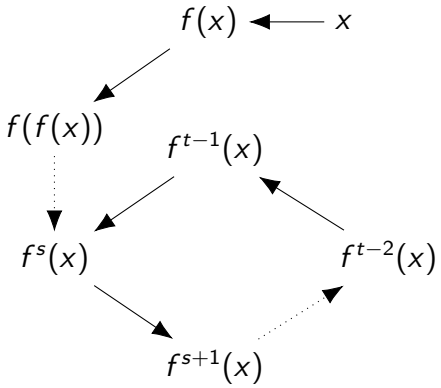
Tortoise and Hare: The Cycle Finding Problem

- 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 $f^s(x) = f^{s+t}(x)$



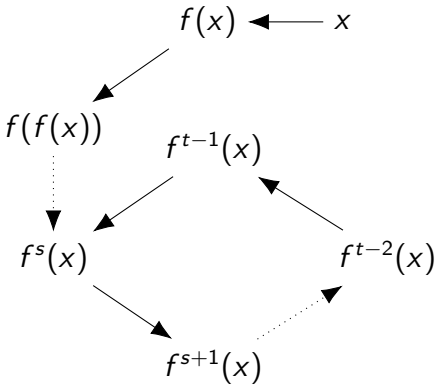
Tortoise and Hare: Floyd's Algorithm

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



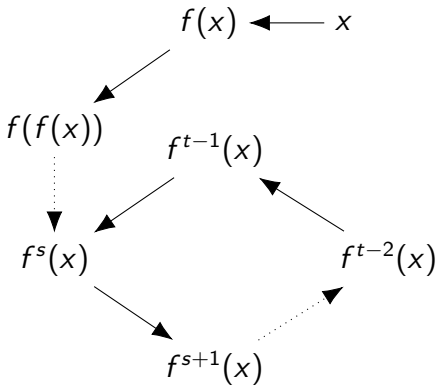
Tortoise and Hare: Floyd's Algorithm

- 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

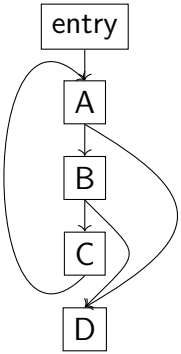
We develop probabilistic tools for upper and lower bounding the expected time until two independent random walks on $\mathbb{Z}$ intersect each other. This leads to the first sharp analysis of a non-trivial Birthday attack, proving that Pollard's Kangaroo method solves the discrete logarithm problem $g^x = h$ 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.

Primal Duel

- Optimization problems
  - Find minimum/maximum of objective function
  - Variables must satisfy constraints
- Initial formulation called the "primal"
- There is a transformation that moves primal constraints into objective function terms, and moves primal objective function terms into constraints
  - The resulting formulation is called the "dual"
- Some optimization algorithms switch between primal and dual formulations
  - Primal-dual methods

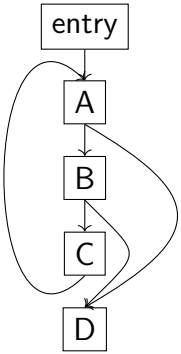
Dominators

- Analysis of control-flow graphs
- Node A dominates B if any path from the entry node to B must pass through A.
- Any node dominates itself; to exclude this case, use strict dominance.
- Strict dominance cannot have a cycle
  - Suppose A dominates B
  - Take a path to B with no cycles
  - Shorten this path so it ends at A
  - This is a path from the entry node to A that doesn't pass through B.



Dominators

- Analysis of control-flow graphs
- Node A dominates B if any path from the entry node to B must pass through A.
- Any node dominates itself; to exclude this case, use strict dominance.
- Strict dominance cannot have a cycle
  - Suppose A dominates B
  - Take a path to B with no cycles
  - Shorten this path so it ends at A
  - This is a path from the entry node to A that doesn't pass through B.
- The strict dominance graph can be top-sorted



Data Structures & Algorithms @ ANE 2026

The Dominance Frontier

Bunsen Bitti, Unsigned Long

January 17, 2026