Data Structures & Algorithms @ AC2025

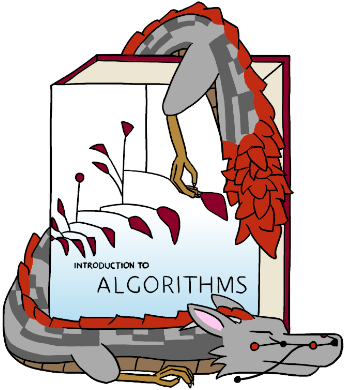
Deep C Adventures

Bunsen Bitti, Unsigned Long

July 3, 2025

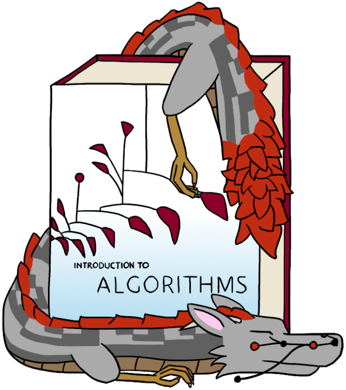
Overview

- This is a comedic/informative look at overlaps between furry and computer science
  - This is not a rigorous lecture
  - There will be puns `^w^`



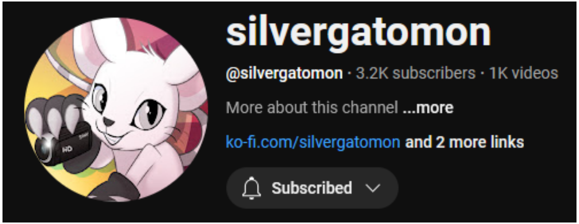
Overview

- 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



How I Found the Fandom

- Found furries through con videos



How I Found the Fandom

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

Tien Long

How I Found the Fandom

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

lóng

龙

How I Found the Fandom

- 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]

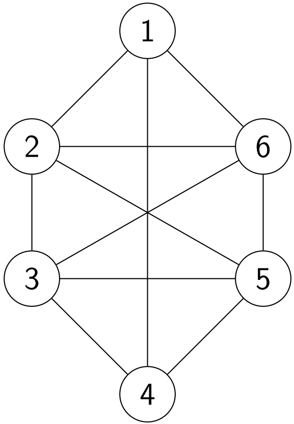
`unsigned long` `unsigned long int`	*Long* unsigned integer type. Capable of containing at least the [0, 4 294 967 295] range.[3]

How I Found the Fandom

- 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

Fursuits are Planar Graphs

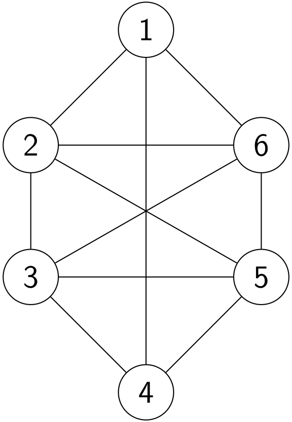
- Graph:
  - Vertices labeled $1, 2, \ldots, n$
  - Edges go between two vertices
  - At most $n \cdot (n-1)/2$ edges[a]



[a] Assuming simple graphs

Fursuits are Planar Graphs

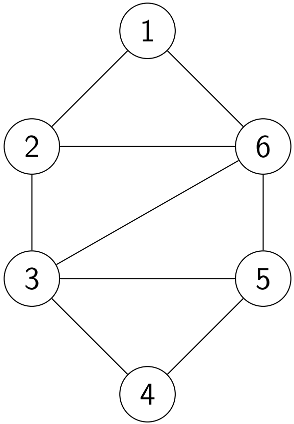
- Graph:
  - Vertices labeled $1, 2, \ldots, n$
  - Edges go between two vertices
  - At most $n \cdot (n-1)/2 = \underline{O(n^2)}$ edges[a]



[a] Assuming simple graphs

Fursuits are Planar Graphs

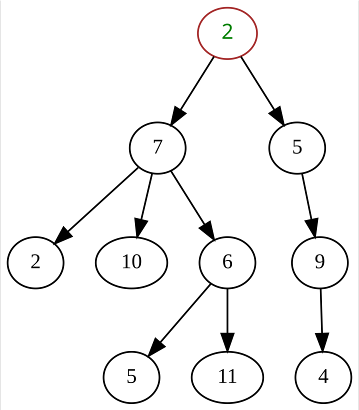
- Graph:
  - Vertices labeled $1, 2, \ldots, n$
  - Edges go between two vertices
  - At most $n \cdot (n-1)/2 = \underline{O(n^2)}$ edges[a]
- Planar graphs: graphs that can be drawn without crossing edges
  - At most $3n - 6 = \underline{O(n)}$ edges![b]
  - Fursuits are planar graphs
  - Seam count $= O(\text{Fabric patch count})$



[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.



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

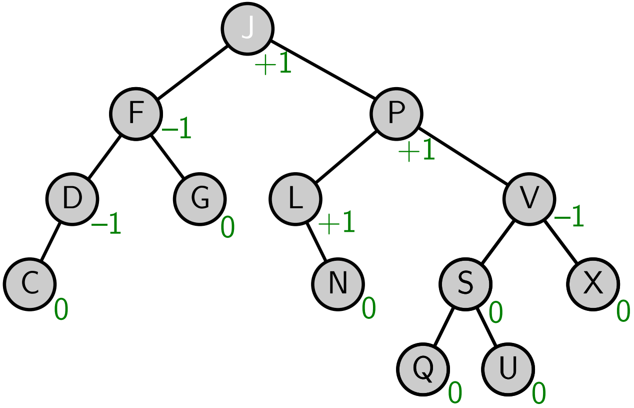
~~Subsections~~ Subtrees

- **ARS**TZ : Balanced Binary Search Trees (BST)
- **B**C**M**UXY : Other Binary Trees
- I**K**QV : Spatial Data Structures
- **E**FGJ**L**P : Niche data structures
- **DHN**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)

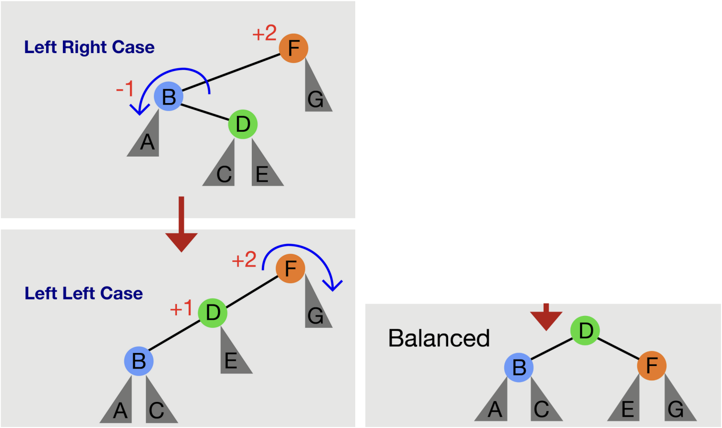
AVL Trees (Rotation)

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

/r/furry_irl/comments/1dumfg3/

AVL Trees (Rotation)

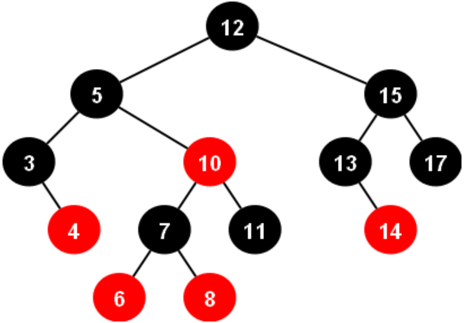
- Imbalances are fixed by rotations, fundamental local operations for many balanced BSTs



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**

ˈsplā

splayed; splaying; splays

Synonyms of *splay* >

transitive verb

1 : to cause to spread outward

2 : to make oblique : **BEVEL**

intransitive verb

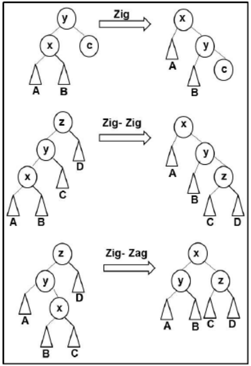
1 : to extend apart or outward especially in an awkward manner

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.

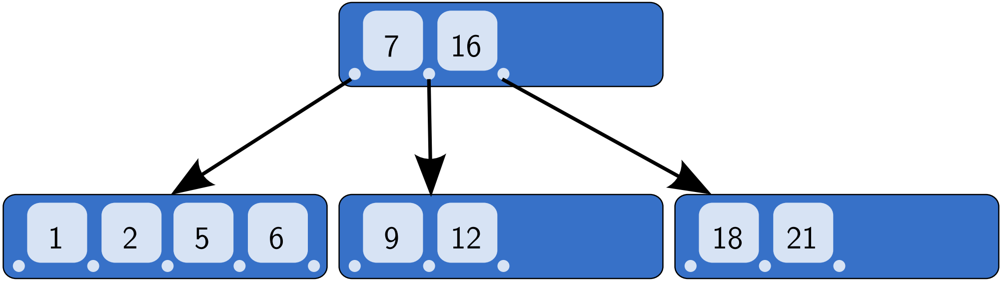
[3]Sleator, Daniel D.; Tarjan, Robert E. (1985).

B Tree[4]

- Generalising binary search trees to > 2 children
- Internal nodes hold both data and pointers to children nodes in sorted order
- Forms the basis of many filesystem structures
  - Apple HFS+, Microsoft NTFS, Linux ext4

[4] Bayer, R.; McCreight, E. (July 1970).

B Tree (Diagram)

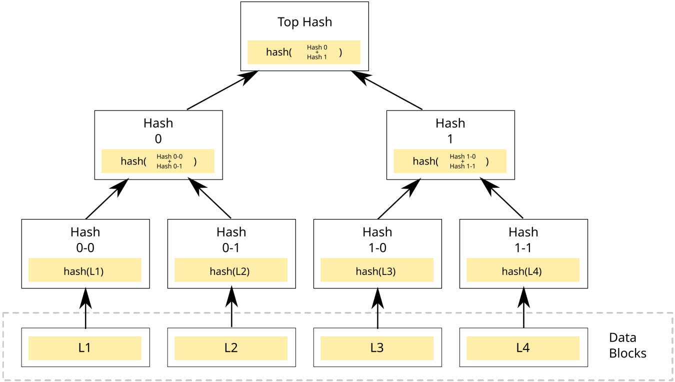


What Rudy [Bayer] likes to say is, the more you think about what the B in B-Tree means, the better you understand B-Trees!

- Edward M. McCreight

Merkle Tree[5]

- 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!

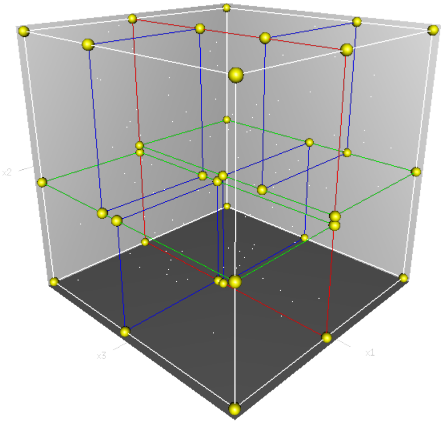


[5] Merkle, R. C. (1979)

k-d tree[6]

- Each node is a k-dimensional point, each internal node splits one of the dimensions at the point.
- $O(\log n)$ insert, delete and search for a point, fast[a] range search and nearest neighbour search.

[a] subject to curse of dimensionality



[6] Bentley, J. L. (1975).

Exponential Trees[7]

- Has niche uses in hash tables
- Time complexity is a *bit cursed*

Abstract

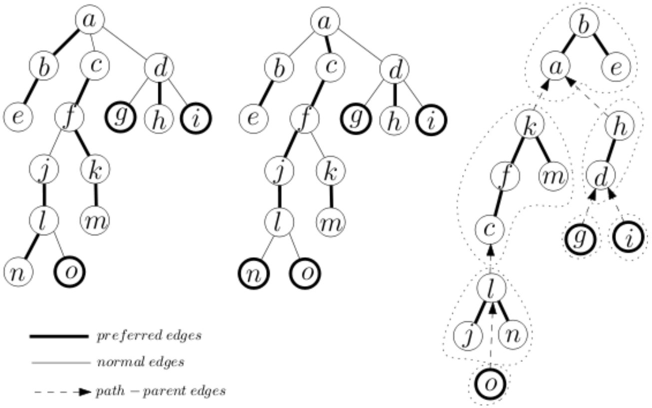
We present a significant improvement on linear space deterministic sorting and searching. On a unit-cost RAM with word size w, an ordered set of n w-bit keys (viewed as binary strings or integers) can be maintained in

$$O\left(\min\left\{\begin{array}{l}\sqrt{\log n}\\ \frac{\log n}{\log w} + \log\log n\\ \log w \log\log n\end{array}\right\}\right)$$

[7] Andersson, Arne (October 1996)

Link-Cut Tree[8]

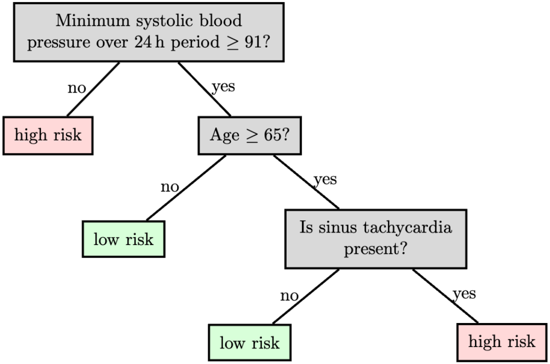
- Maintains a set of rooted trees
- Amortized $O(\log n)$ link, cut and find-root at any node
- Improves Dinic's Algorithm (for max-flow) from $O(V^2E)$ to $O(VE \log V)$.



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

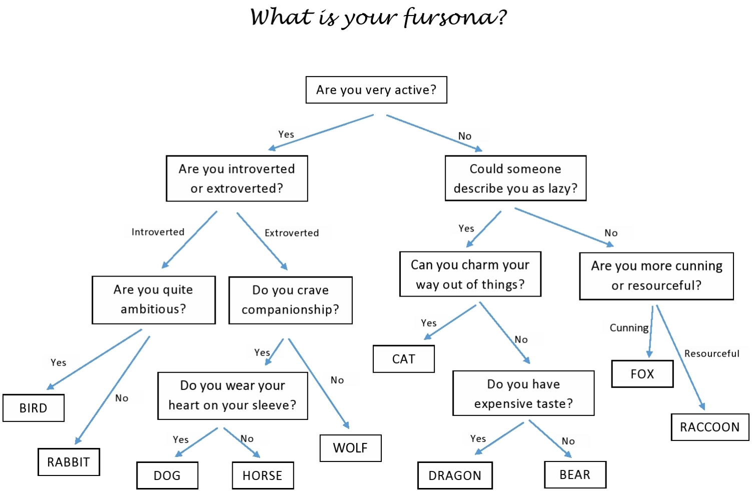
Decision Trees

- 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



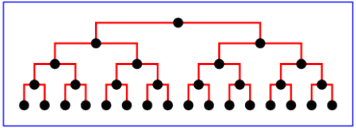
MIT 6.390 Intro ML Course Notes and Breiman, Friedman, Olshen, Stone (1984)

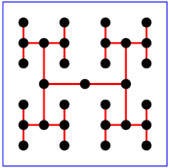
(Fury) Decision Trees (`r/furry/comments/4gxm01/`)



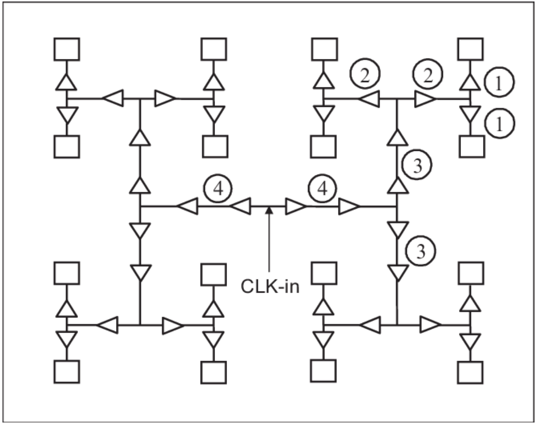
H Trees

- Binary tree fractal!
- Used in chip design in order to distribute wide reaching nets





https://www.tamurajones.net/FractalGenealogy.xhtml



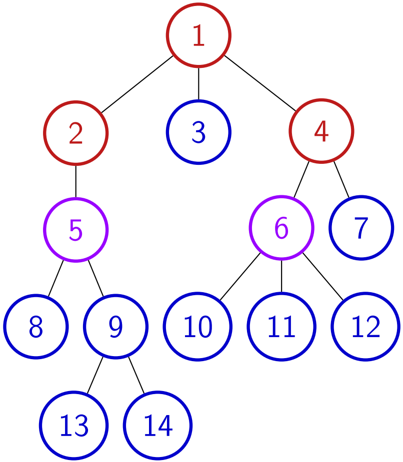
Tawfik, Sherif & Kursun, Volkan. (2008).

Missing N-tree

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

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 $O(n^{1/c})$



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." $\rightarrow$

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

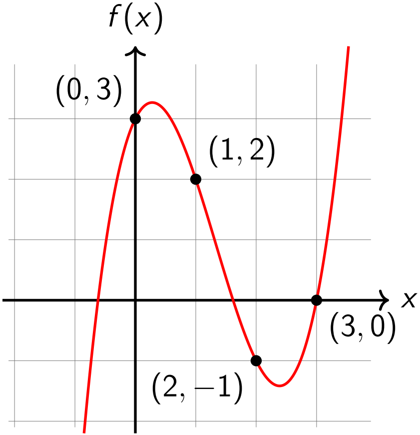
Fast Furrier Transform: Polynomials

coefficient term degree

$$f(x) = 3 + 2x \boxed{\boxed{-4}x^2} + x^{\boxed{3}}$$

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

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



Fast Furrier Transform: Multiplication

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

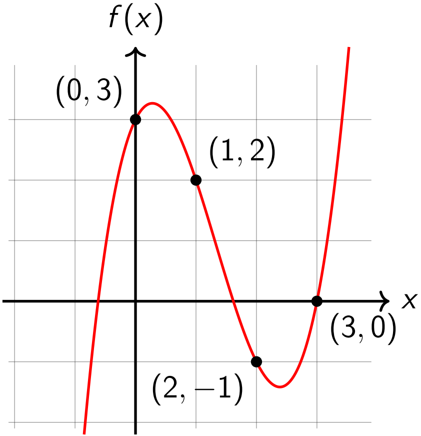
$$g(x) = 2 \quad - x \quad + x^2$$

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

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

$$f(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + \cdots + a_{n-2}x^{n-2} + a_{n-1}x^{n-1}$$

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

Fast Furrier Transform: Evaluation at ± 1

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

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

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

Fast Furrier Transform: Evaluation at ± 1

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

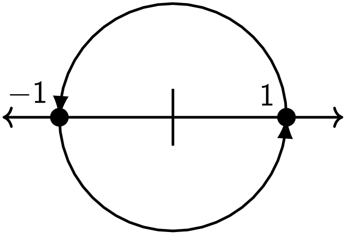
$$\begin{aligned} f(1) &= a_0 + a_1 + a_2 + a_3 + \cdots + a_{n-2} + a_{n-1} \\ f(-1) &= a_0 - a_1 + a_2 - a_3 + \cdots + a_{n-2} - a_{n-1} \end{aligned}$$

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

Number of operations is halved!

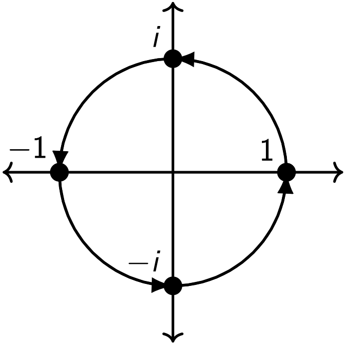
Fast Furrier Transform: Roots of Unity

- 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

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

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

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

$$f(x) = a_0 + a_1x + a_2x^2 + \cdots + a_{n-1}x^{n-1}$$

$$\begin{aligned} f_e(x) &= a_0 \ + a_2x^2 + a_4x^4 + \cdots + a_{n-2}x^{n-2} \\ f_o(x) &= a_1x + a_3x^3 + a_5x^5 + \cdots + a_{n-1}x^{n-1} \\ f_o(x) &= x(a_1 \ + a_3x^2 + a_5x^4 + \cdots + a_{n-1}x^{n-2}) \end{aligned}$$

Fast Furrier Transform: Odd/Even Trick

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

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

Fast Furrier Transform: Odd/Even Trick

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

$$f_e(x) = a_0 \quad + a_2 x^2 + a_4 x^4 + \cdots + a_{n-2} x^{n-2}$$

$$f_o(x) = a_1 x + a_3 x^3 + a_5 x^5 + \cdots + a_{n-1} x^{n-1}$$

$$f_o(x) = x(a_1 \quad + a_3 x^2 + a_5 x^4 + \cdots + a_{n-1} x^{n-2})$$

$$g_e(y) = a_0 \quad + a_2 y \quad + a_4 y^2 + \cdots + a_{n-2} y^{n/2-1}$$

$$g_o(y) = a_1 \quad + a_3 y \quad + a_5 y^2 + \cdots + a_{n-1} y^{n/2-1}$$

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

Fast Furrier Transform: Divide and Conquer

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

- 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

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

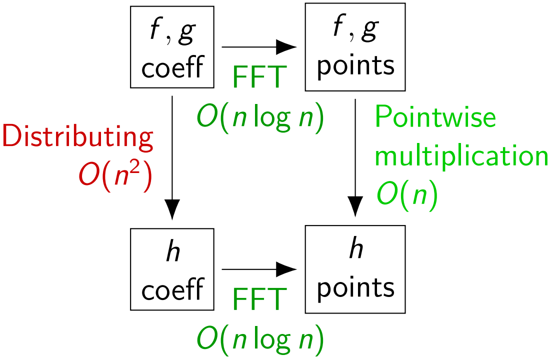
- We need to evaluate g_e and g_o at $y = x^2 =$

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

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

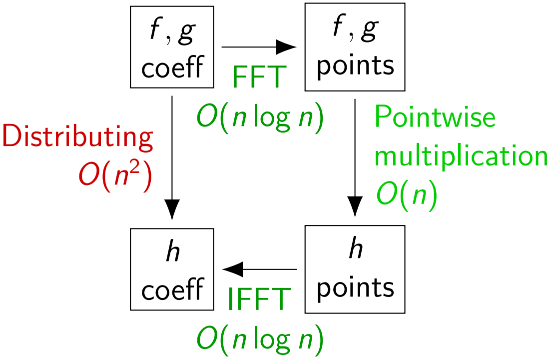
Fast Furrier Transform: Inverse FFT?

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

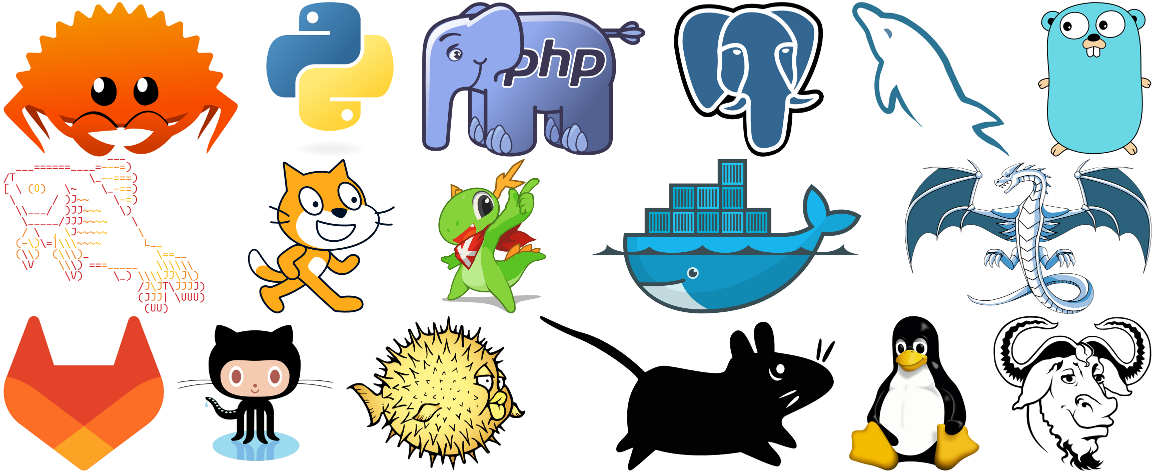


Fast Furrier Transform: Inverse FFT?

- 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



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)
```

Gray Code

Gray Himakar

Gray Code

- Normal counting can have many digits change when going to the next value
  - Physical switches changing might not be simultaneous

Binary	
000	
001	
010	
011	
100	
101	
110	
111	

Gray Code

- Normal counting can have many digits change when going to the next value
  - Physical switches changing might not be simultaneous
- Binary reflected gray code (BRGC): to add a bit, reflect existing sequence, give one half a 0, and the other half a 1

Binary	BRGC
000	0
001	1
010	
011	
100	
101	
110	
111	

Gray Code

- Normal counting can have many digits change when going to the next value
  - Physical switches changing might not be simultaneous
- Binary reflected gray code (BRGC): to add a bit, reflect existing sequence, give one half a 0, and the other half a 1

Binary	BRGC
000	0
001	1
010	1
011	0
100	
101	
110	
111	

Gray Code

- Normal counting can have many digits change when going to the next value
  - Physical switches changing might not be simultaneous
- Binary reflected gray code (BRGC): to add a bit, reflect existing sequence, give one half a 0, and the other half a 1

Binary	BRGC
000	00
001	01
010	11
011	10
100	
101	
110	
111	

Gray Code

- Normal counting can have many digits change when going to the next value
  - Physical switches changing might not be simultaneous
- Binary reflected gray code (BRGC): to add a bit, reflect existing sequence, give one half a 0, and the other half a 1

Binary	BRGC
000	00
001	01
010	11
011	10
100	10
101	11
110	01
111	00

Gray Code

- Normal counting can have many digits change when going to the next value
  - Physical switches changing might not be simultaneous
- Binary reflected gray code (BRGC): to add a bit, reflect existing sequence, give one half a 0, and the other half a 1

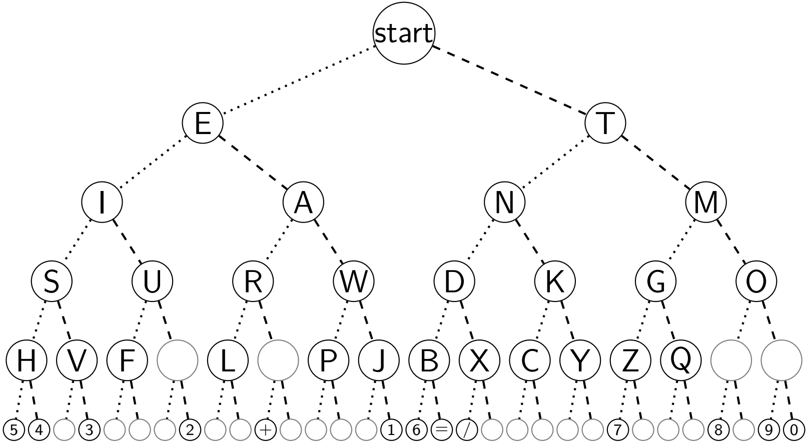
Binary	BRGC
000	000
001	001
010	011
011	010
100	110
101	111
110	101
111	100

Gray Code

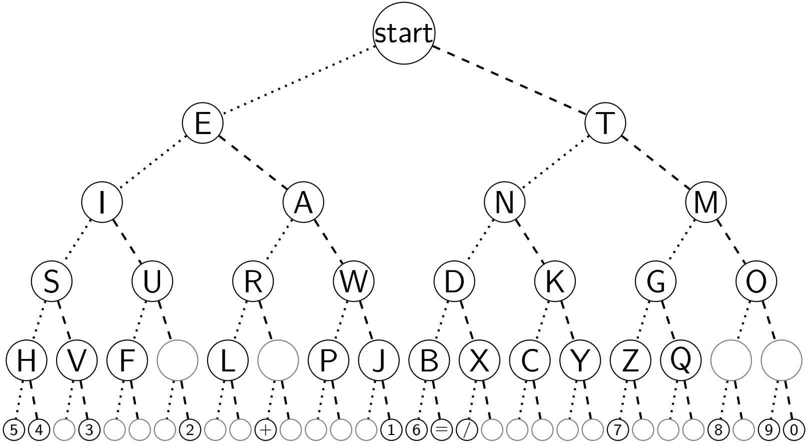
- Normal counting can have many digits change when going to the next value
  - Physical switches changing might not be simultaneous
- Binary reflected gray code (BRGC): to add a bit, reflect existing sequence, give one half a 0, and the other half a 1
- There are cool bit hacks for computing the next BRGC value and converting between binary and BRGC
- There are also variants that balance the number of transitions per bit

Binary	BRGC
000	000
001	001
010	011
011	010
100	110
101	111
110	101
111	100

Morse Code

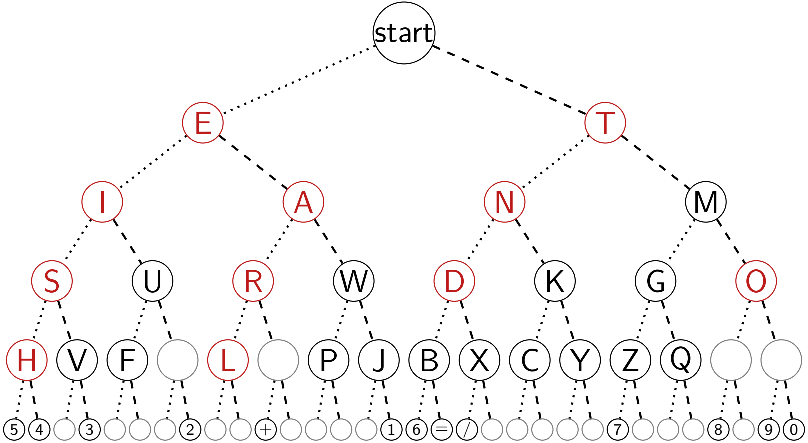


Morse Code



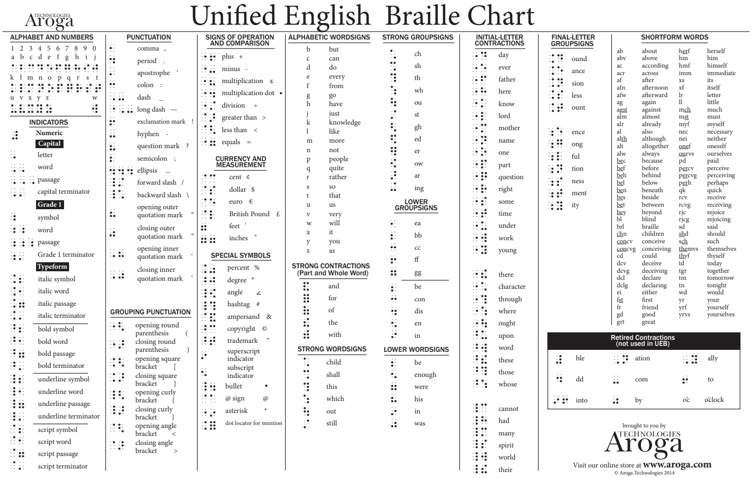
Letter	Frequency
E	12.7%
T	9.1%
A	8.2%
O	7.5%
I	7.0%
N	6.7%
S	6.3%
H	6.1%
R	6.0%
D	4.3%
L	4.0%

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



Aroga TECHNOLOGIES
Unified English Braille Chart
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
less
ount
ence
ong
ful
tion
ness
ment
ity
SHORTFORM WORDS
ab about
abv above
ac according
acr across
af after
afn afternoon
afw afterward
ag again
agst against
alm almost
alr already
al also
alth although
alt altogether
alw always
bec because
bef before
beh behind
bel below
ben beneath
bes beside
bet between
bey beyond
bl blind
brl braille
chn children
concv conceive
concvg conceiving
cd could
dcv deceive
dcvg deceiving
dcl declare
dclg declaring
ei either
fst first
fr friend
gd good
grt great
herf herself
hm him
hmf himself
imm immediate
xs its
xf itself
lr letter
ll little
mch much
mst must
myf myself
nec necessary
nei neither
onef oneself
ourvs ourselves
pd paid
percv perceive
percvg perceiving
perh perhaps
qk quick
rcv receive
rcvg receiving
rjc rejoice
rjcg rejoicing
sd said
shd should
sch such
themvs themselves
thyf thyself
td today
tgr together
tm tomorrow
tn tonight
wd would
yr your
yrf yourself
yrvs yourselves
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

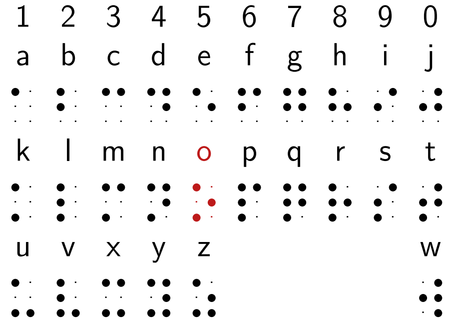
1	2	3	4	5	6	7	8	9	0
a	b	c	d	e	f	g	h	i	j
⠁	⠃	⠉	⠙	⠑	⠋	⠛	⠓	⠊	⠚
k	l	m	n	o	p	q	r	s	t
⠅	⠇	⠍	⠝	⠕	⠏	⠟	⠗	⠎	⠞
u	v	x	y	z					w
⠥	⠧	⠭	⠽	⠵					⠺

STRONG GROUPSIGNS

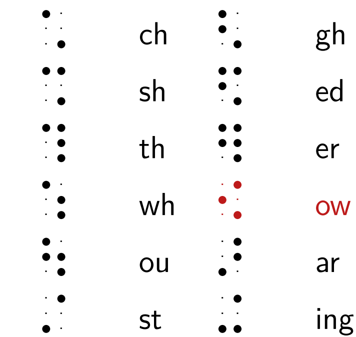
⠡	ch	⠣	gh
⠩	sh	⠫	ed
⠹	th	⠻	er
⠱	wh	⠪	ow
⠳	ou	⠜	ar
⠌	st	⠬	ing

Braille

ALPHABET AND NUMBERS



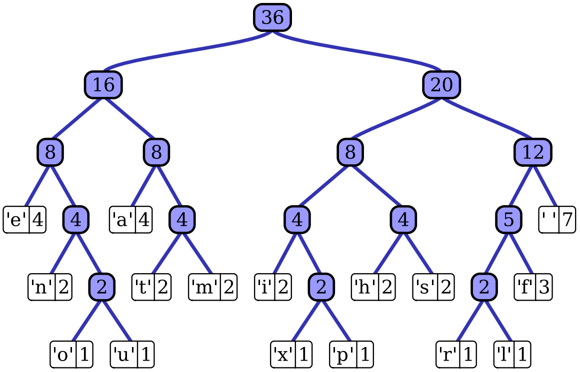
STRONG GROUPSIGNS



“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

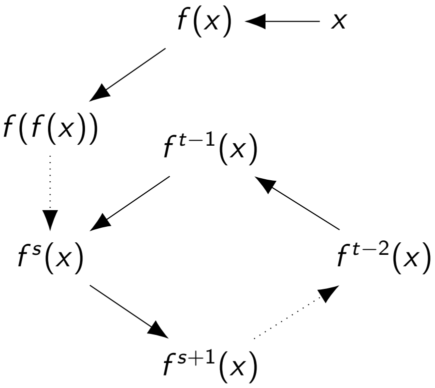


Tortoise and Hare: The Cycle Finding Problem

- 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 $[n]$

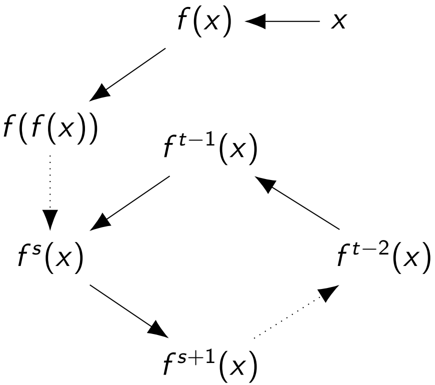
Tortoise and Hare: The Cycle Finding Problem

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



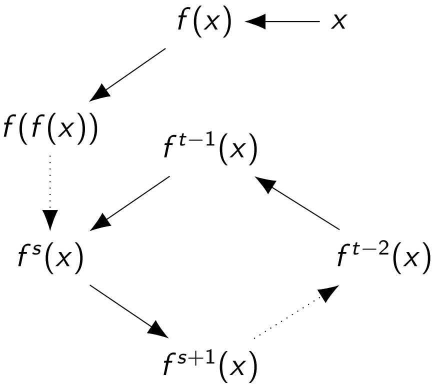
Tortoise and Hare: The Cycle Finding Problem

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



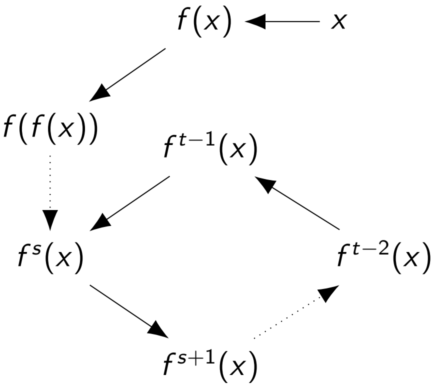
Tortoise and Hare: Floyd's Algorithm

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



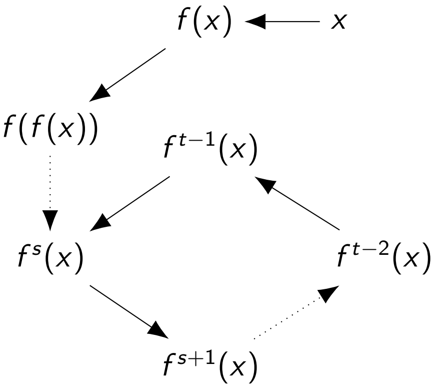
Tortoise and Hare: Floyd's Algorithm

- 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
  - Pollard's kangaroo exists???



Primal Duel

- Optimization problems
  - Find minimum/maximum of objective function
  - Variables must satisfy constraints

Primal Duel

- Optimization problems
  - Find minimum/maximum of objective function
  - Variables must satisfy constraints
- Initial formulation called the “primal”

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"

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

Data Structures & Algorithms @ AC2025

Deep C Adventures

Bunsen Bitti, Unsigned Long

July 3, 2025