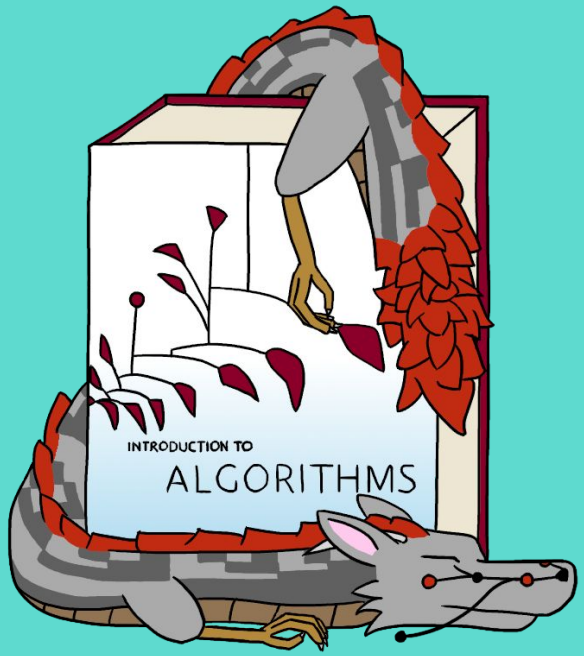
Data Structures and Algorithms

By Unsigned Long

@InverseHackermann

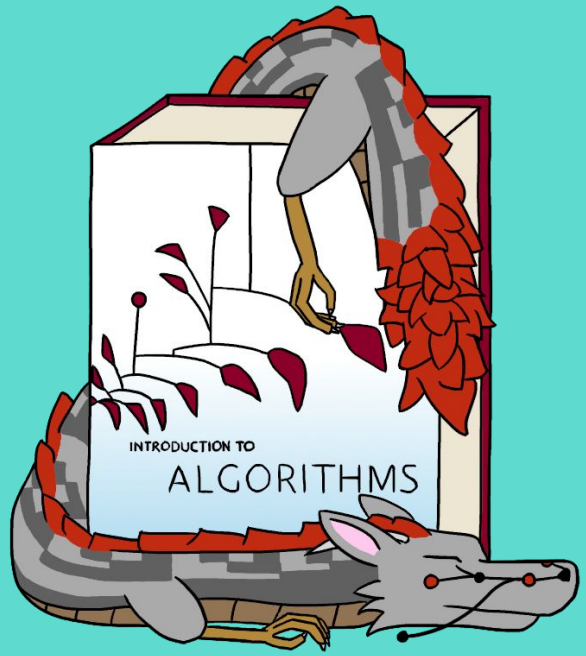
Overview

- This is a comedic/informative look at overlaps between furry and computer science
    - There will be puns ^w^



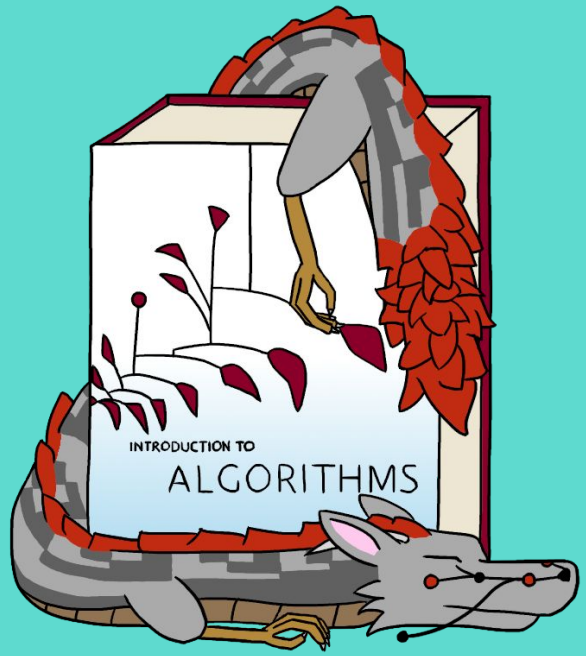
Overview

- This is a comedic/informative look at overlaps between furry and computer science
  - There will be puns ^w^
- This is **not** a rigorous lecture
  - We're trying to have fun here :3



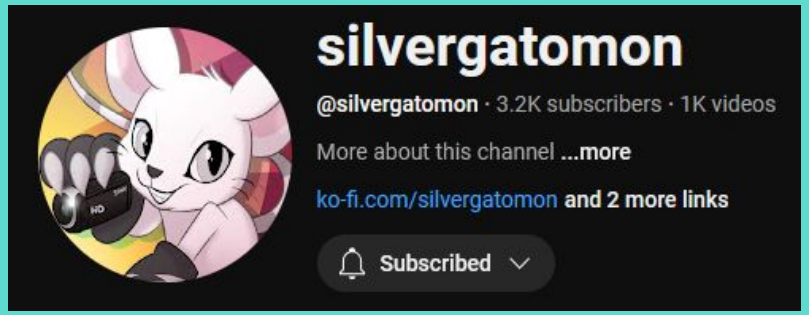
Overview

- This is a comedic/informative look at overlaps between furry and computer science
  - There will be puns ^w^
- This is **not** a rigorous lecture
  - We're trying to have fun here :3
- Ask questions!
  - I've been coding for most of my life
  - If you don't understand things, **that's on me**
  - I'm out of touch with reality



About Me

- Found furries through con videos



About Me

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

https://www.furaffinity.net/user/tienlong/

About Me

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

lóng

About Me

- 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]

About Me

- 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

Measuring Information

- Bit: stores 0 or 1
- Byte: 8 bits
- Nibble: 4 bits = half a byte
    - Not commonly used

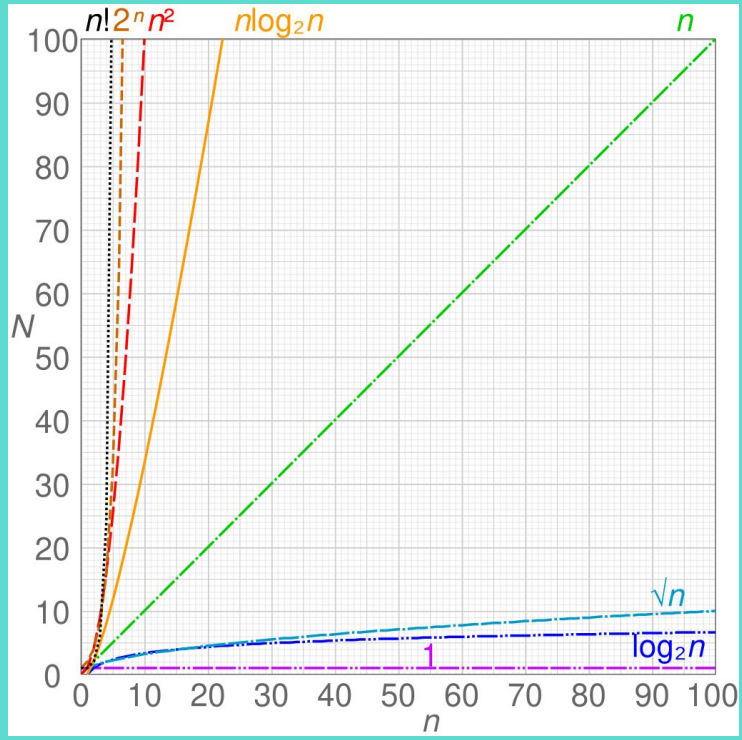
https://www.corgibeansshop.com/

Order Notation

Oops I forgot to go over this in previous iterations of this panel

Order Notation

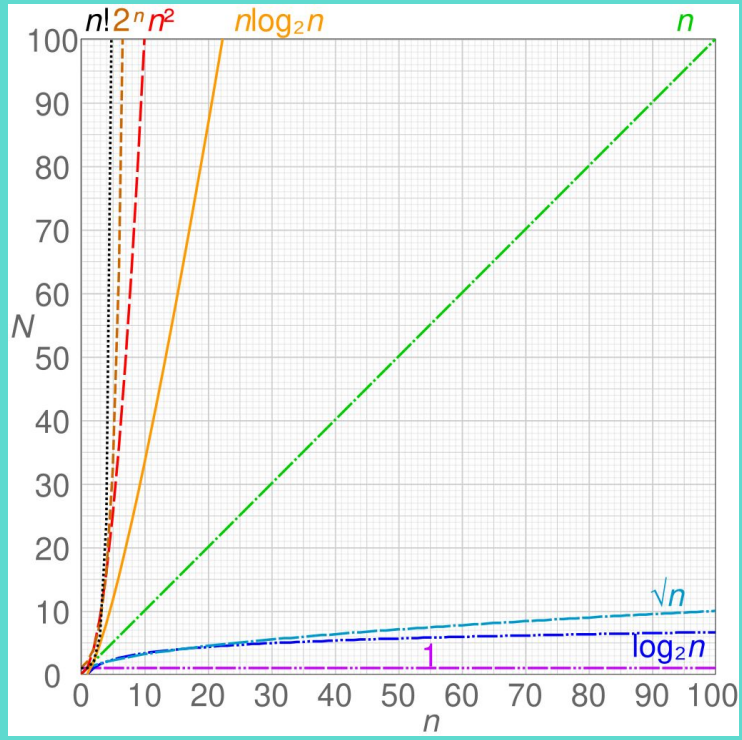
- Formally ignore constant factors
  - $n,\ 5n+3,\ 0.1n-9$ are all $O(n)$



https://en.wikipedia.org/wiki/Big_O_notation

Order Notation

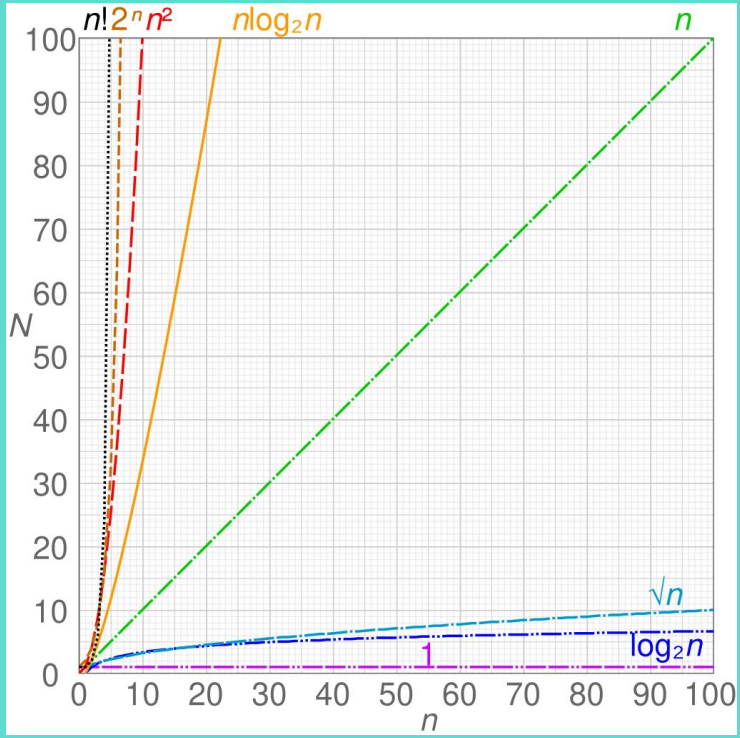
- Formally ignore constant factors
  - $n,\ 5n+3,\ 0.1n-9$ are all $O(n)$
- Only care about long-term growth
  - Eventually, $100n < n^2$



https://en.wikipedia.org/wiki/Big_O_notation

Order Notation

- Formally ignore constant factors
  - $n,\ 5n+3,\ 0.1n-9$ are all $O(n)$
- Only care about long-term growth
  - Eventually, $100n < n^2$
- Some people ignore bigger factors
  - $n^3 \log^4(n) = \widetilde{O}\left(n^3\right) = O\left(n^c\right)$
  - The more you ignore, the more theoretical your analysis is



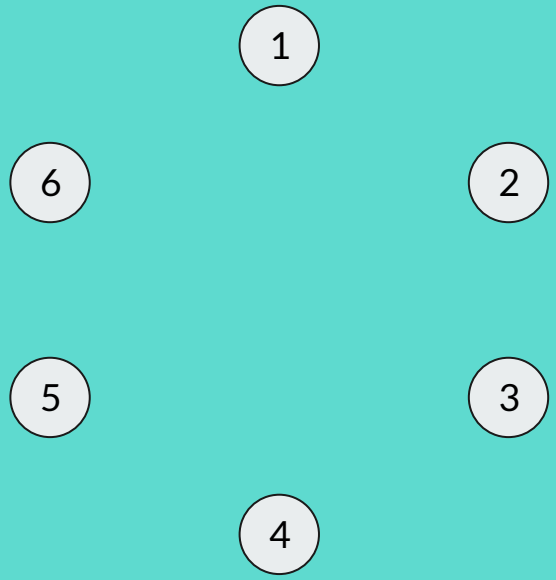
https://en.wikipedia.org/wiki/Big_O_notation

Planar Graphs

Thank you Euler :3

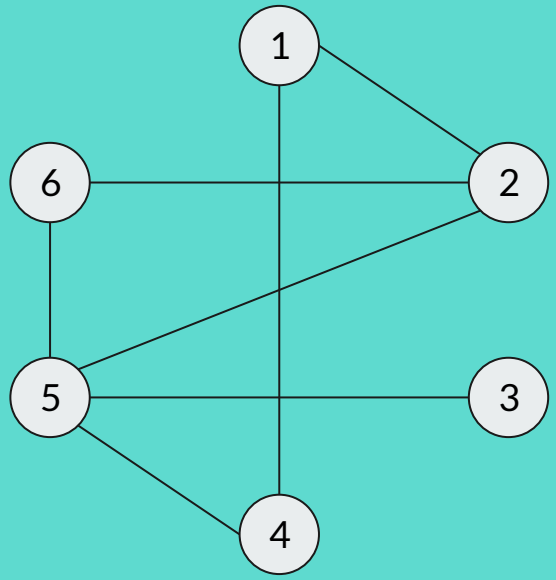
Planar Graph Complexity

- Vertices V, often labeled 1, 2, …, n



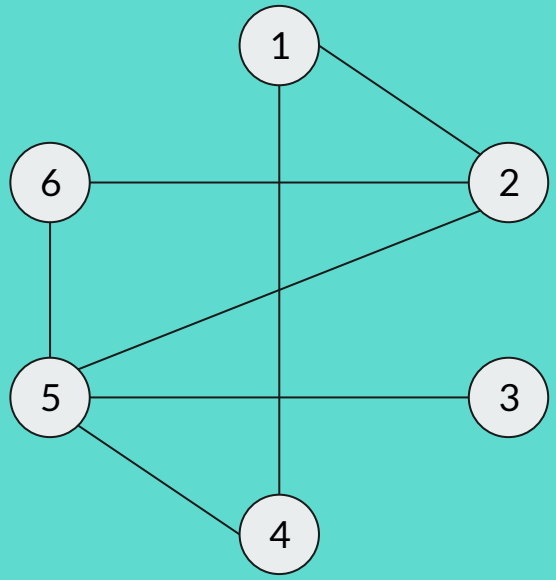
Planar Graph Complexity

- Vertices V, often labeled 1, 2, …, n
- Edges are between two vertices
  - Worst case m = O(n^2) edges



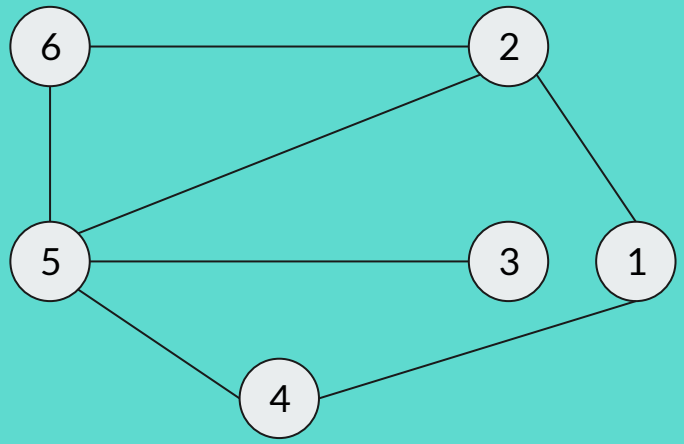
Planar Graph Complexity

- Vertices V, often labeled 1, 2, …, n
- Edges are between two vertices
  - Worst case m = O(n^2) edges
- Planar graphs can be drawn without crossing edges



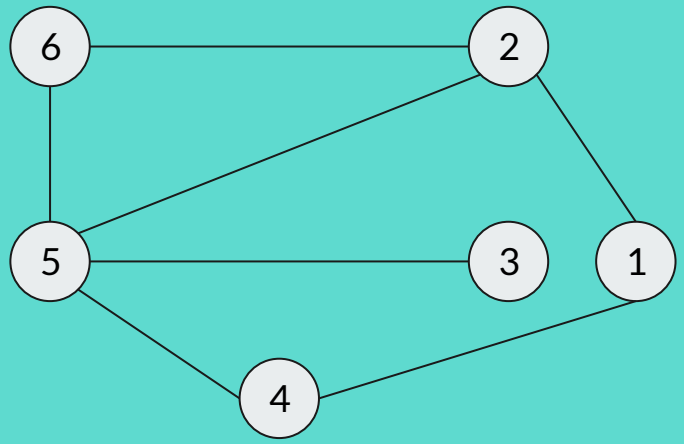
Planar Graph Complexity

- Vertices V, often labeled 1, 2, …, n
- Edges are between two vertices
  - Worst case m = O(n^2) edges
- Planar graphs can be drawn without crossing edges



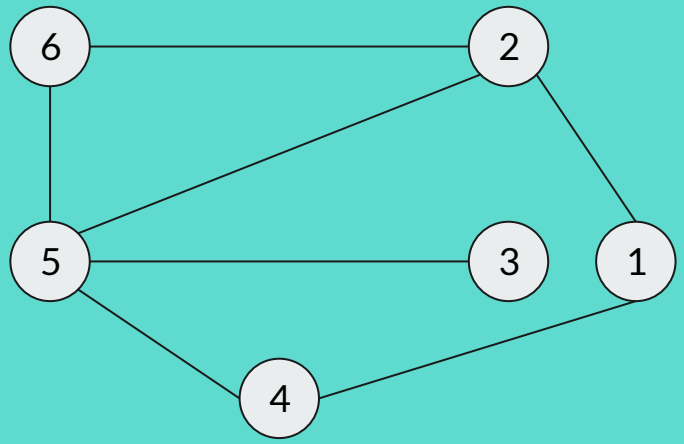
Planar Graph Complexity

- Vertices V, often labeled 1, 2, …, n
- Edges are between two vertices
  - Worst case m = O(n^2) edges
- Planar graphs can be drawn without crossing edges
- Planar graphs have m=O(n)



Planar Graph Complexity

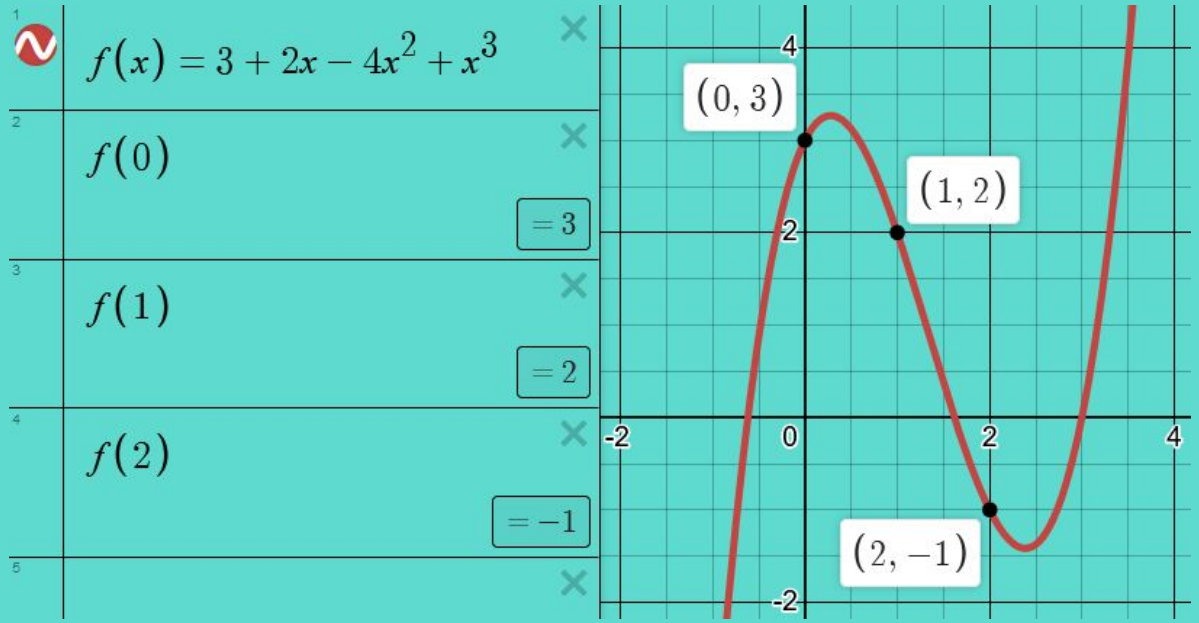
- Vertices V, often labeled 1, 2, …, n
- Edges are between two vertices
  - Worst case m = O(n^2) edges
- Planar graphs can be drawn without crossing edges
- Planar graphs have m=O(n)
- Fursuits are planar graphs
  - Vertices are fabric patches
  - Edges are seams
  - Number of seams is O(n)



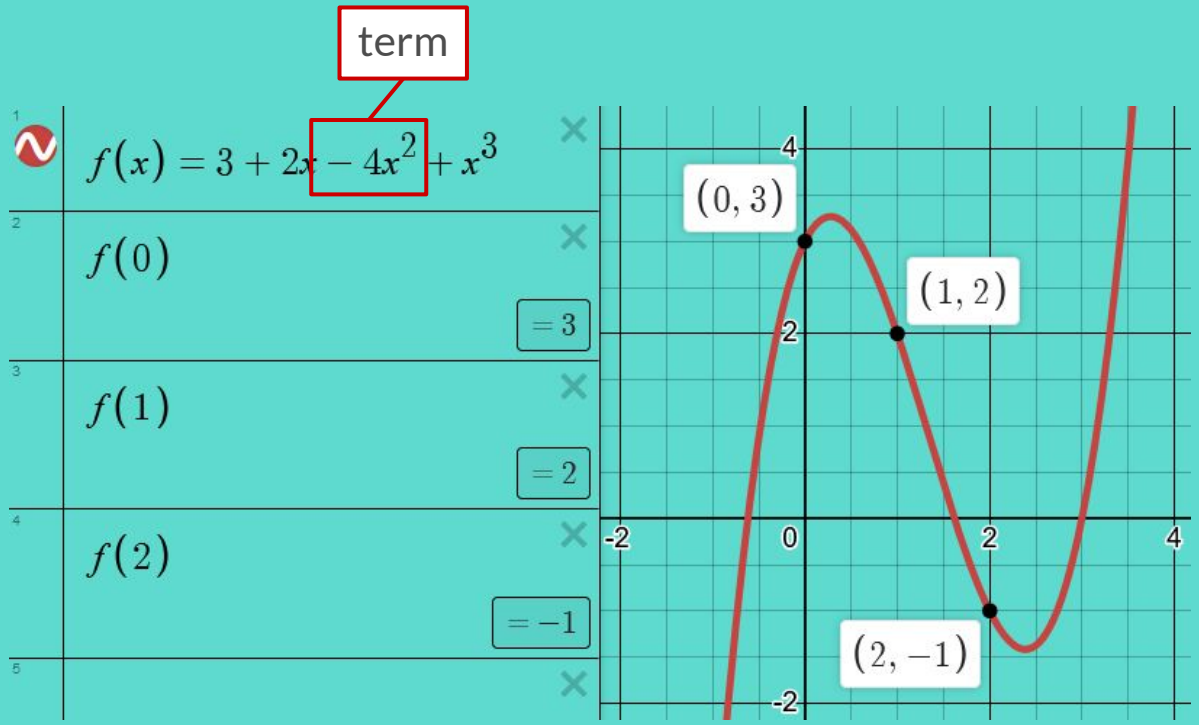
Fast Fourier Transform (FFT)

I have butterflies in my stomach >w<

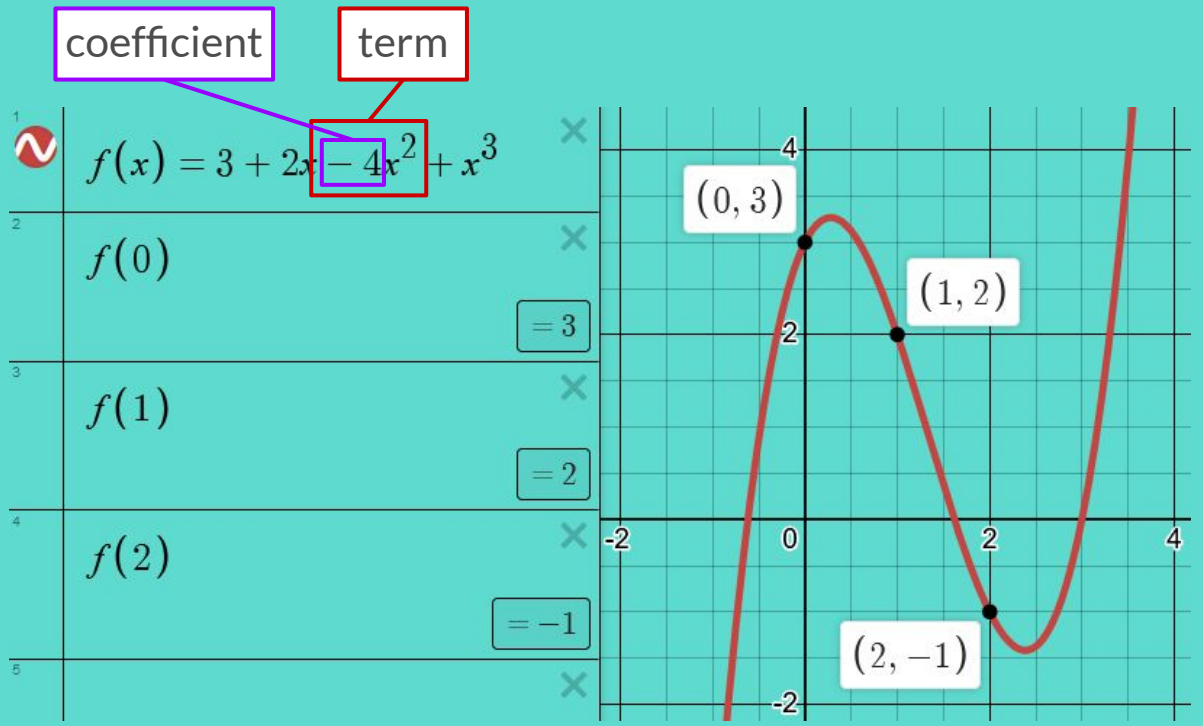
Polynomials



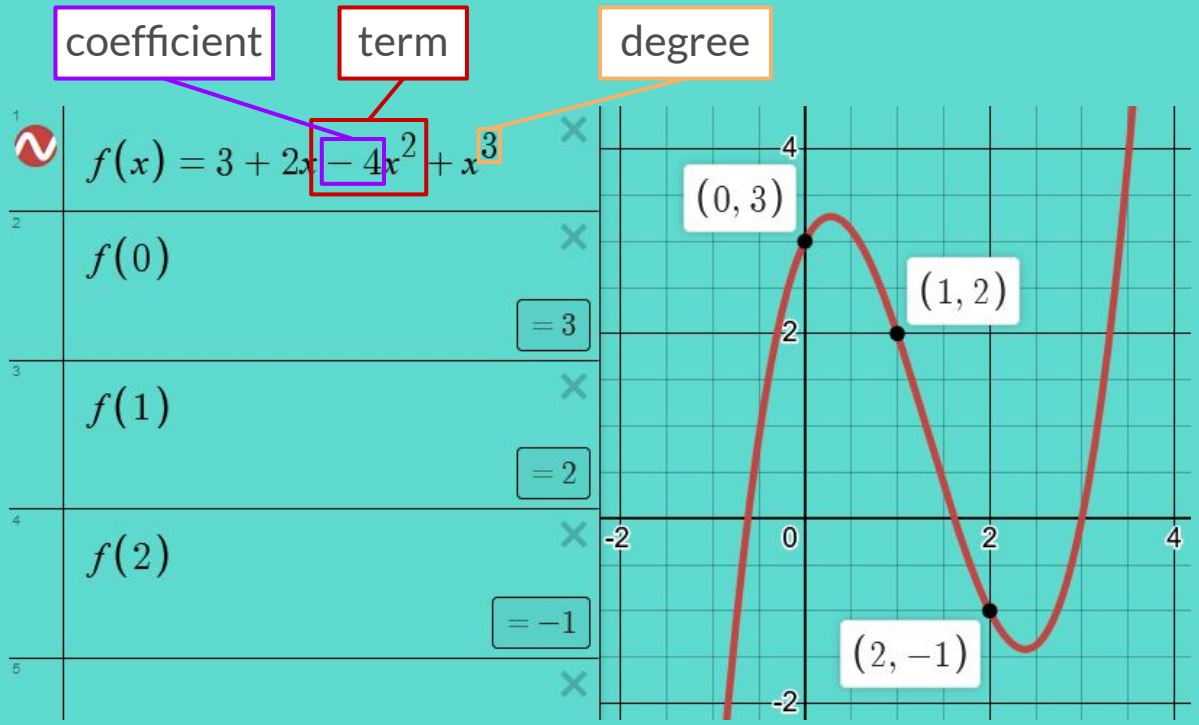
Polynomials



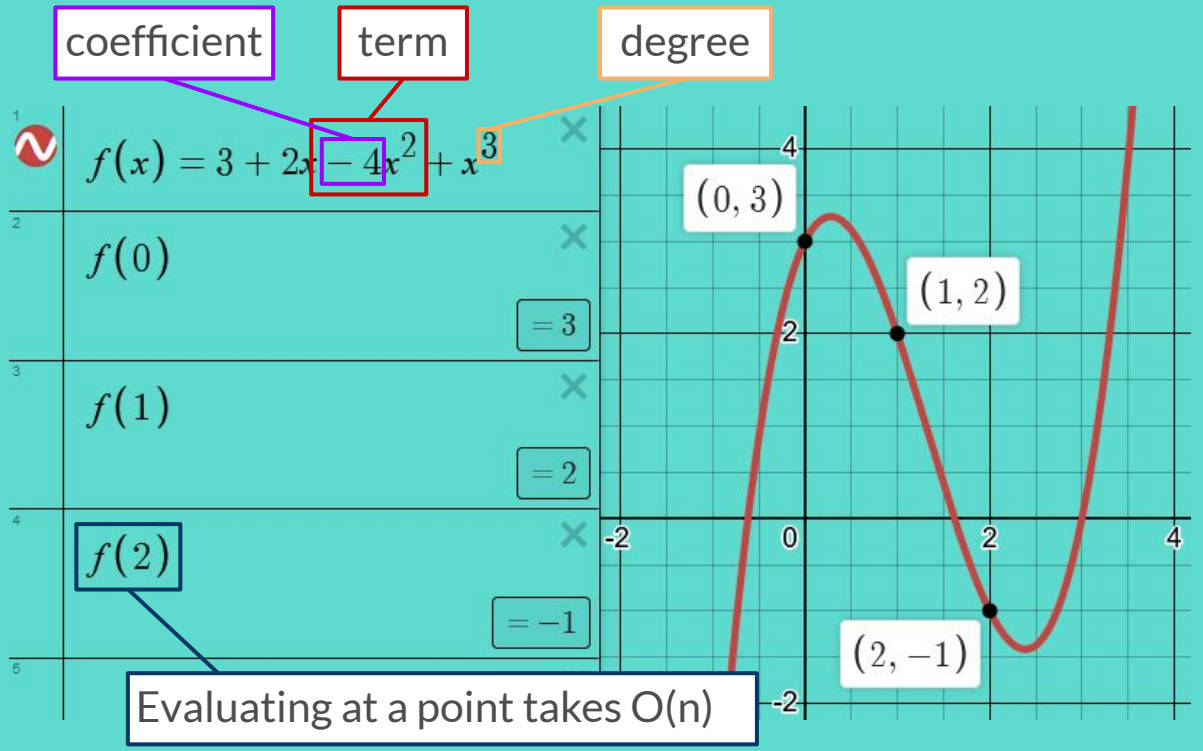
Polynomials



Polynomials

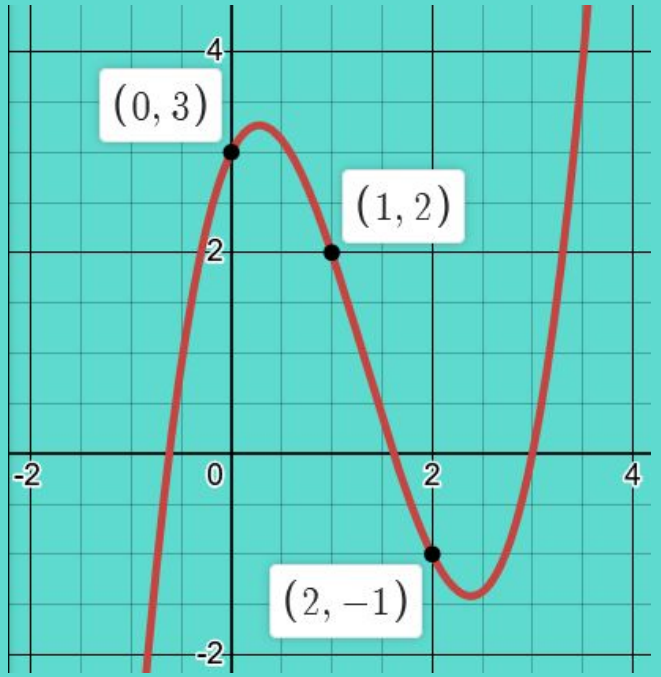


Polynomials



Polynomial Interpolation/Evaluation

- For n points, there is a unique polynomial with degree less than n that passes through all the points
- Evaluating the polynomial to find these points typically takes O(n^2), but if we choose points cleverly, we can do better



Evaluating at x=-1 and x=1

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

Evaluating at x=-1 and x=1

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

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

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

Evaluating at x=-1 and x=1

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

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

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

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

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

Evaluating at x=-1 and x=1

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

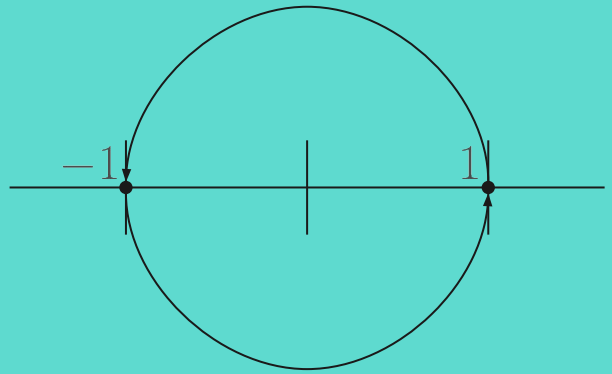
$$f(1) = a_0 + a_1 + \cdots + a_{n-2} + a_{n-1}$$
$$f(-1) = a_0 - a_1 + \cdots + a_{n-2} - a_{n-1}$$

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

Reduces number of operations by a half!

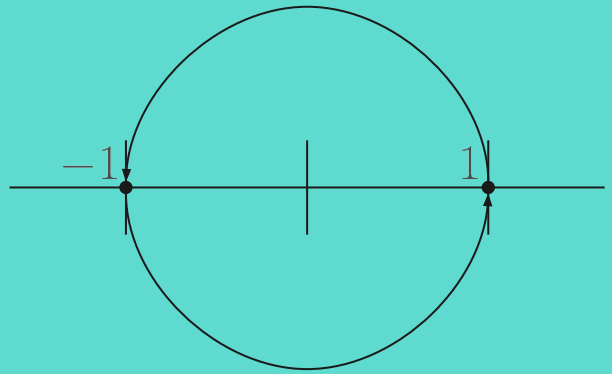
More x values to evaluate at

- The even/odd trick works because repeatedly multiplying by -1 cycles between -1 and 1



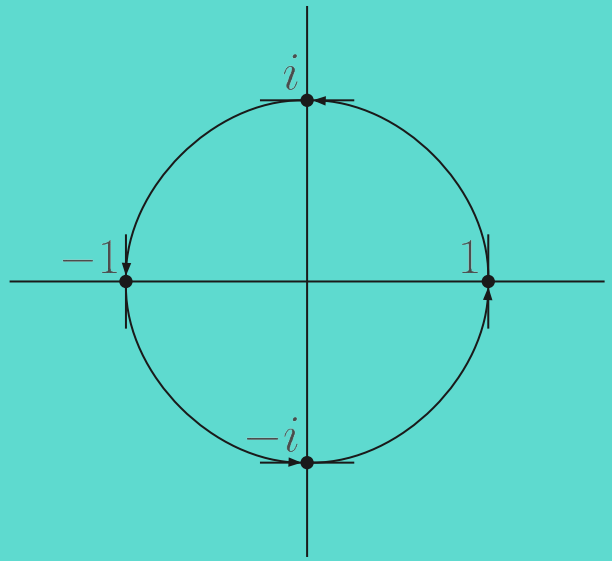
More x values to evaluate at

- 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



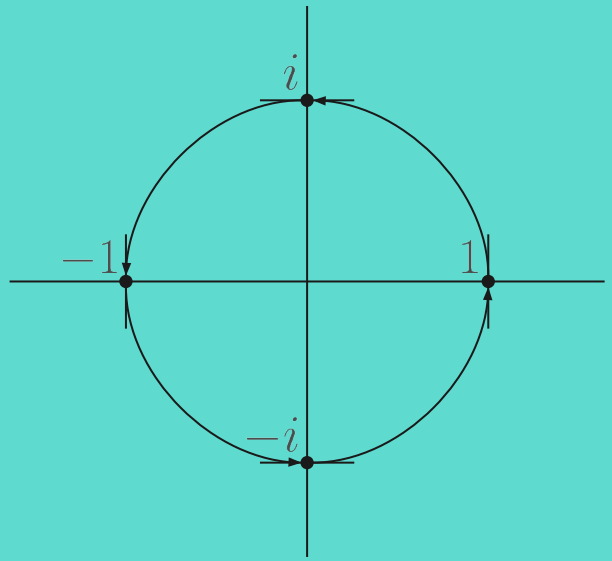
More x values to evaluate at

- 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 $r^n = 1$



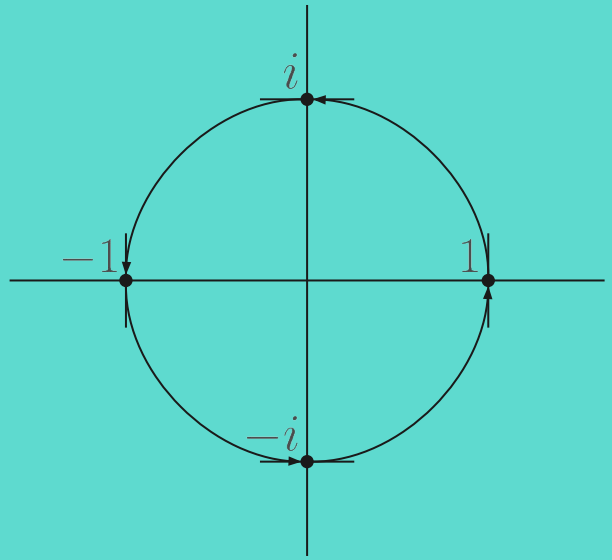
More x values to evaluate at

- 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**



More x values to evaluate at

- 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 $1, r, r^2, \ldots, r^{n-1}$



Applying even/odd trick

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

Applying even/odd trick

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

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

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

Applying even/odd trick

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

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

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

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

Applying even/odd trick

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

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

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

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

Applying even/odd trick

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

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

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

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

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

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

Applying even/odd trick

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

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

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

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

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

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

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

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 at

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

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 at

$$(1)^2,\ (r)^2,\ (r^2)^2, \ldots,\ (r^{n/2-1})^2,\ (r^{n/2})^2,\ (r^{n/2+1})^2, \ldots,\ (r^{n-1})^2$$
$$1,\quad r^2,\quad r^4, \ldots,\quad r^{n-2},\ r^n = 1,\quad r^2, \ldots,\quad r^{n-2}$$

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 at

$$(1)^2,\ (r)^2,\ (r^2)^2, \ldots,\ (r^{n/2-1})^2,\ (r^{n/2})^2,\ (r^{n/2+1})^2, \ldots,\ (r^{n-1})^2$$
$$1,\ r^2,\ r^4, \ldots,\ r^{n-2},\ r^n = 1,\ r^2, \ldots,\ r^{n-2}$$

- To evaluate f at n roots of unity, we evaluate g_e and g_o at $n/2$ roots of unity

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 at

$$(1)^2,\ (r)^2,\ (r^2)^2, \ldots,\ (r^{n/2-1})^2,\ (r^{n/2})^2,\ (r^{n/2+1})^2, \ldots,\ (r^{n-1})^2$$
$$1,\quad r^2,\quad r^4, \ldots,\quad r^{n-2},\ r^n = 1,\quad r^2, \ldots,\quad r^{n-2}$$

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

Inverse Fast Fourier Transform?

- Turns out FFT is its own inverse

Inverse Fast Fourier Transform?

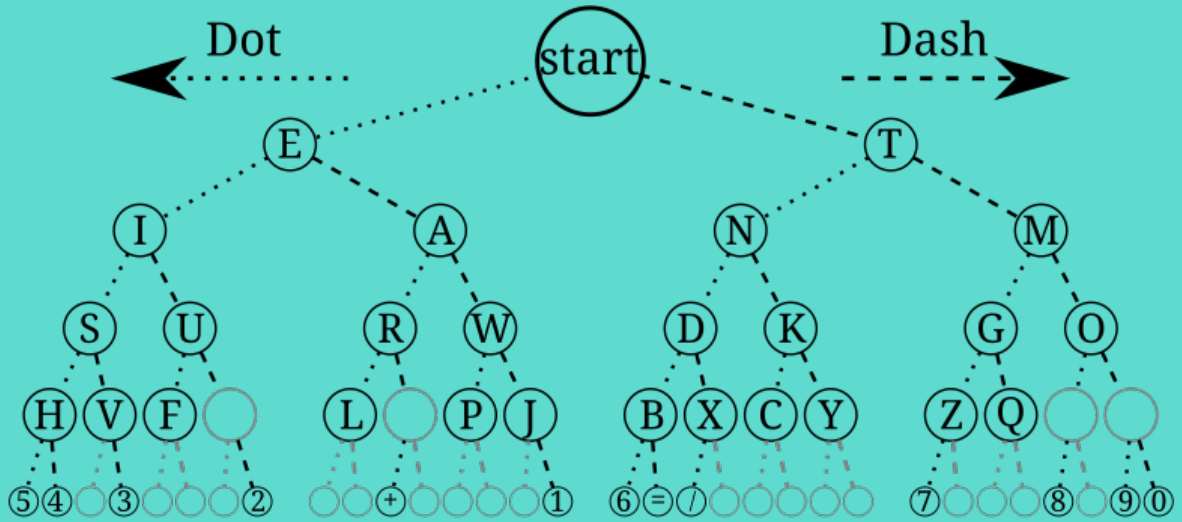
- Turns out FFT is its own inverse
- Multiplying polynomials of degree n
  - Distributing would take $O(n^2)$
  - FFT takes $O(n \log n)$
  - Pointwise multiplication takes $O(n)$
  - (Inverse) FFT takes $O(n \log n)$

Inverse Fast Fourier Transform?

- Turns out FFT is its own inverse
- Multiplying polynomials of degree n
  - Distributing would take $O(n^2)$
  - FFT takes $O(n \log n)$
  - Pointwise multiplication takes $O(n)$
  - (Inverse) FFT takes $O(n \log n)$
- I think FFT being its own inverse is related to why anti-furries eventually become furries

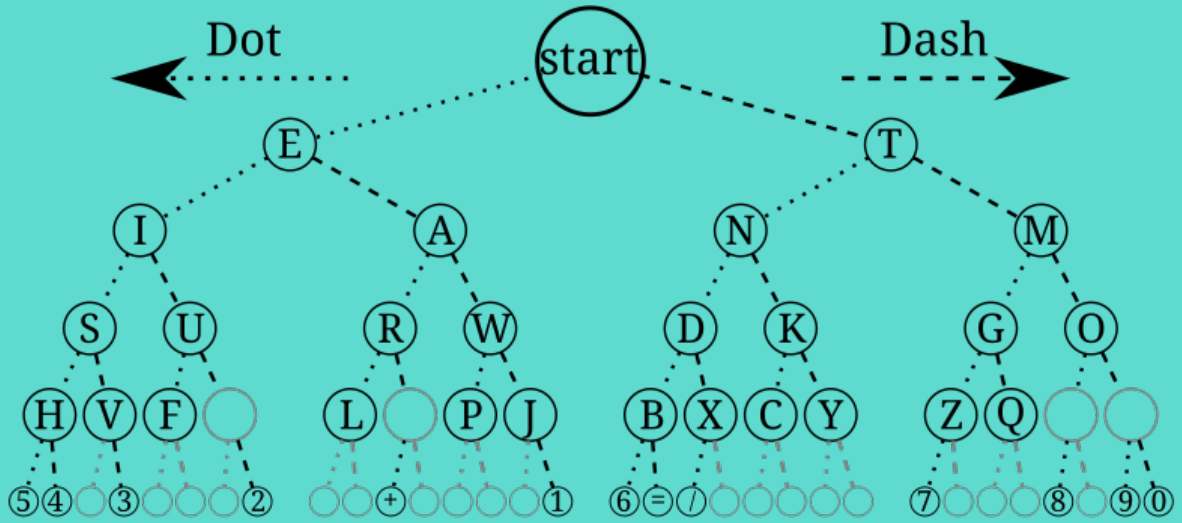
Morse Code, Braille, and Connections to Coding Theory

Morse Code



https://commons.wikimedia.org/wiki/File:Morse-code-tree.svg

Morse Code

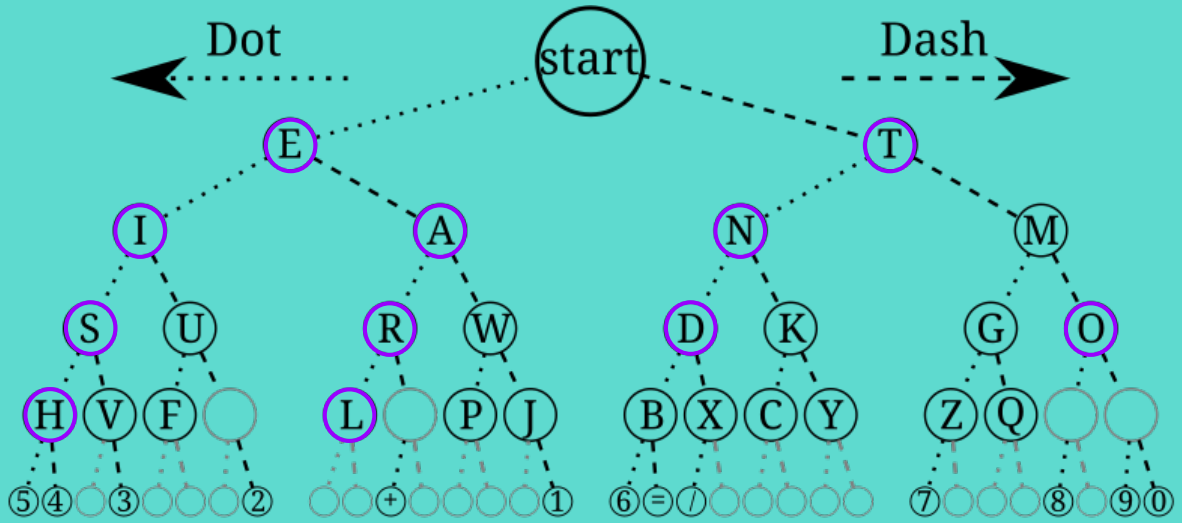


Letter	Relative frequency in the English language[1] (Texts)
E	12.7%
T	9.1%
A	8.2%
O	7.5%
I	7.0%
N	6.7%
S	6.3%
H	6.1%
R	6.0%
D	4.3%
L	4.0%

https://commons.wikimedia.org/wiki/File:Morse-code-tree.svg

https://en.wikipedia.org/wiki/Letter_frequency

Morse Code



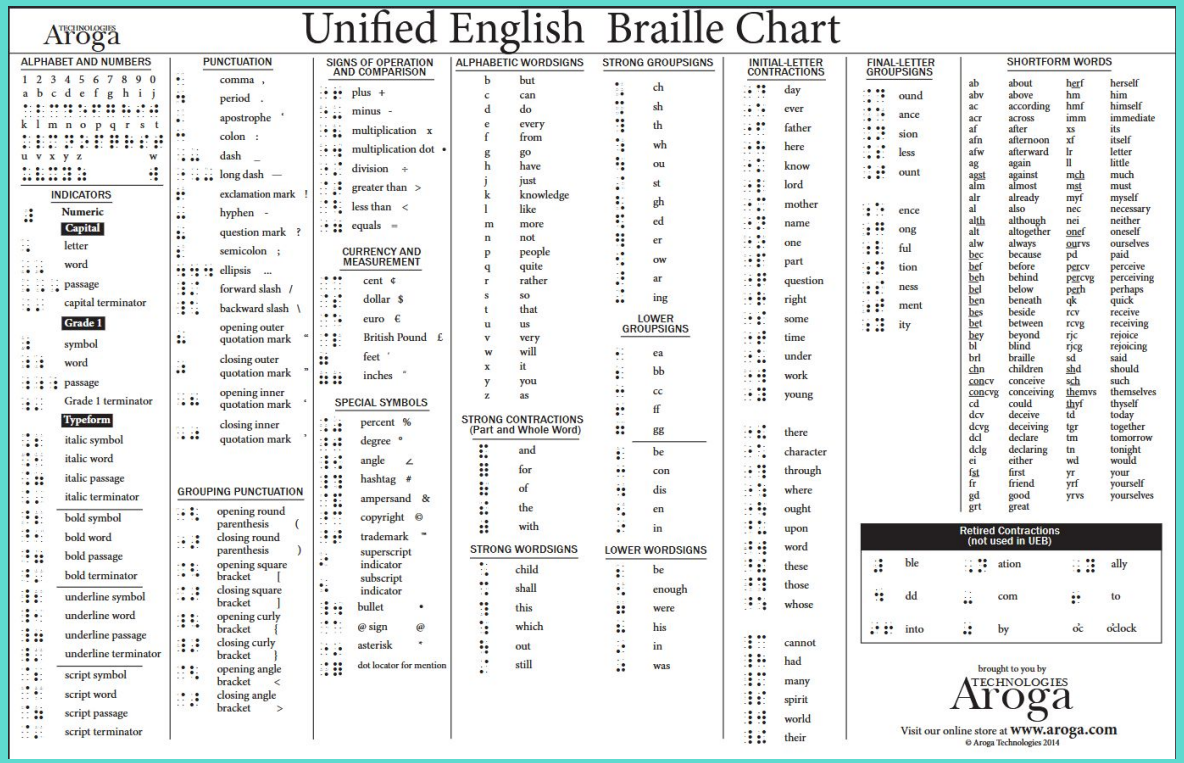
Letter	Relative frequency in the English language[1] (Texts)
E	12.7%
T	9.1%
A	8.2%
O	7.5%
I	7.0%
N	6.7%
S	6.3%
H	6.1%
R	6.0%
D	4.3%
L	4.0%

More frequently used characters are given shorter encodings

https://commons.wikimedia.org/wiki/File:Morse-code-tree.svg

https://en.wikipedia.org/wiki/Letter_frequency

Braille



Unified English Braille Chart

TECHNOLOGIES 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
less
ount

ence
ong
ful
tion
ness
ment
ity

SHORTFORM WORDS

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

Retired Contractions (not used in UEB)

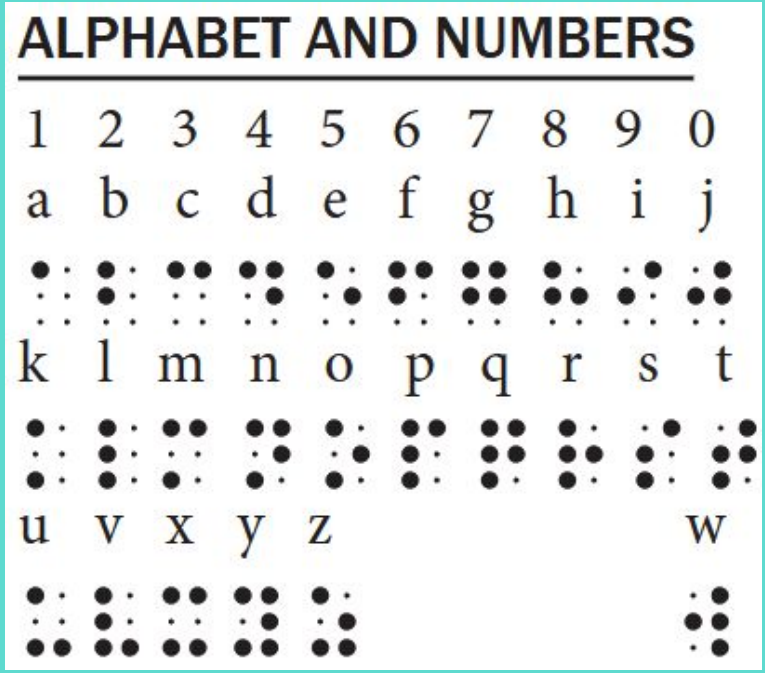
ble	ation	ally	
dd	com	to	
into	by	oc	oclock

brought to you by TECHNOLOGIES Aroga

Visit our online store at www.aroga.com

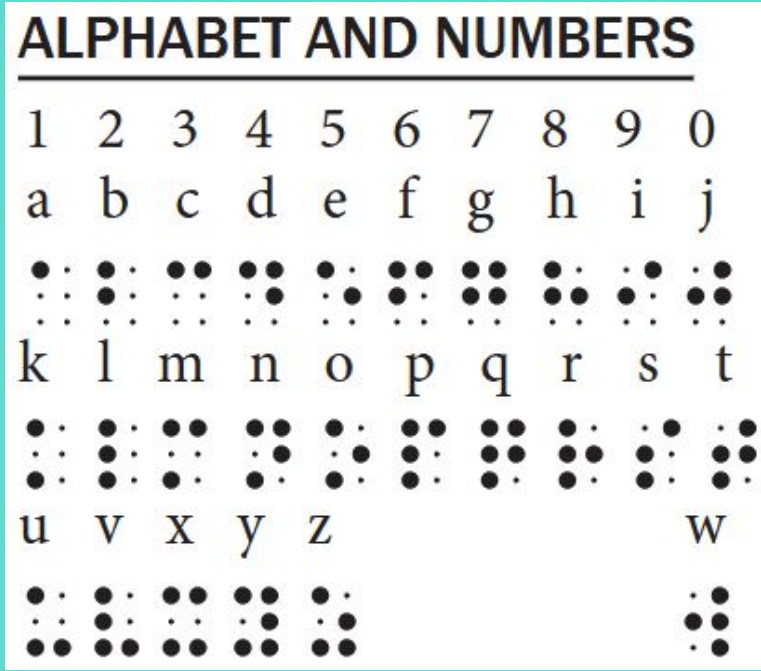
https://www.teachingvisuallyimpaired.com/uploads/1/4/1/2/14122361/ueb_braille_chart.pdf

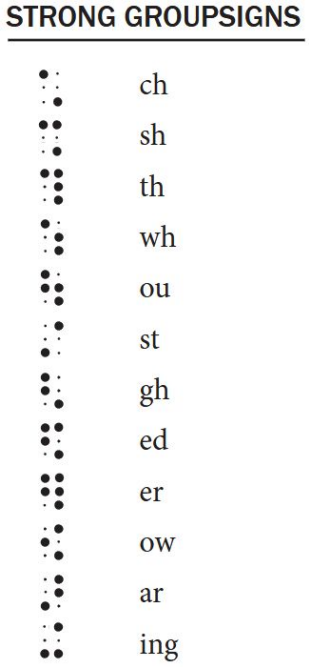
Braille



https://www.teachingvisuallyimpaired.com/uploads/1/4/1/2/14122361/ueb_braille_chart.pdf

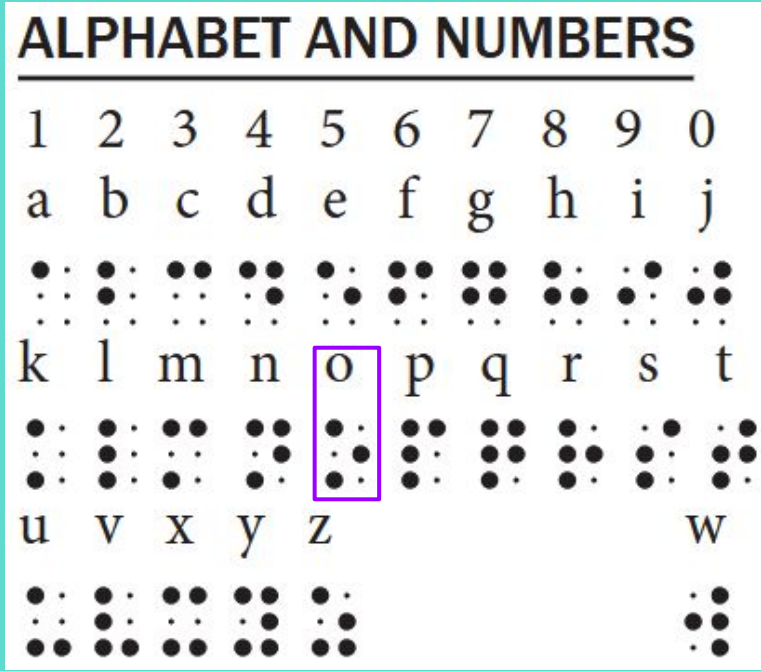
Braille

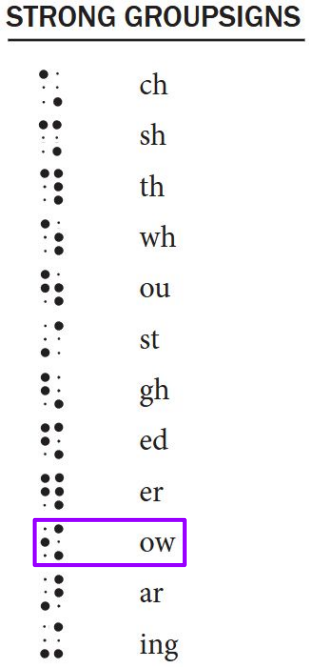




https://www.teachingvisuallyimpaired.com/uploads/1/4/1/2/14122361/ueb_braille_chart.pdf

Braille



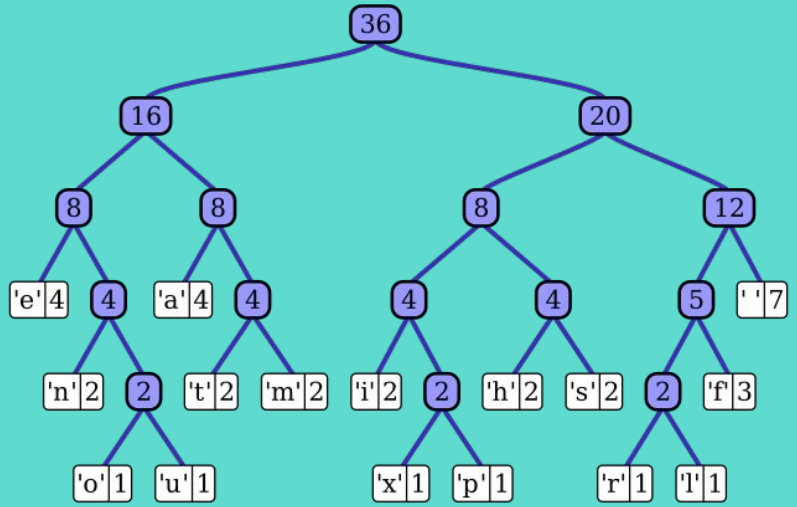


“owo” is ⠪⠺⠕

https://www.teachingvisuallyimpaired.com/uploads/1/4/1/2/14122361/ueb_braille_chart.pdf

Huffman Coding

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



Macro-Micro Tree Decomposition

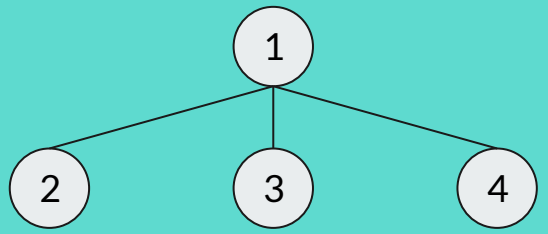
Four russians removing logs in a tree

Macro-Micro Tree Decomposition

- Trees
  - Begin with root node
  - Nodes can have child nodes
  - Repeat
  - Nodes without children are **leaves**

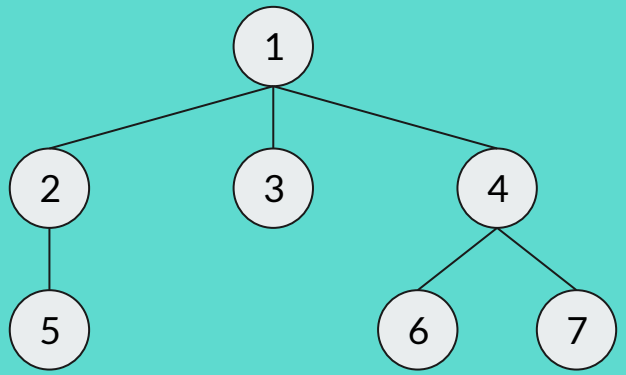
Macro-Micro Tree Decomposition

- Trees
  - Begin with root node
  - Nodes can have child nodes
  - Repeat
  - Nodes without children are **leaves**



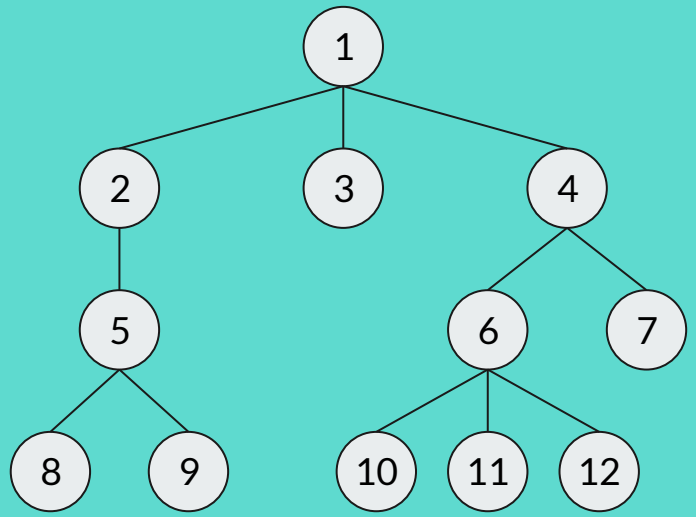
Macro-Micro Tree Decomposition

- Trees
  - Begin with root node
  - Nodes can have child nodes
  - Repeat
  - Nodes without children are **leaves**



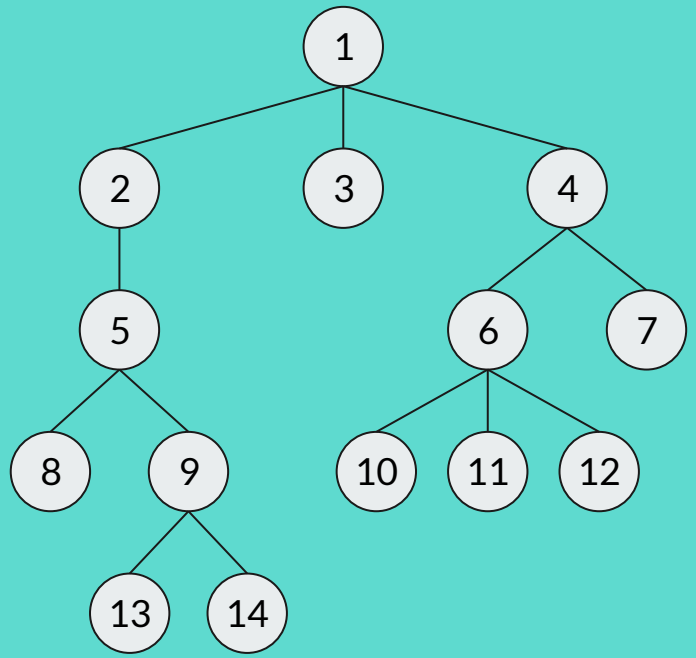
Macro-Micro Tree Decomposition

- Trees
  - Begin with root node
  - Nodes can have child nodes
  - Repeat
  - Nodes without children are **leaves**



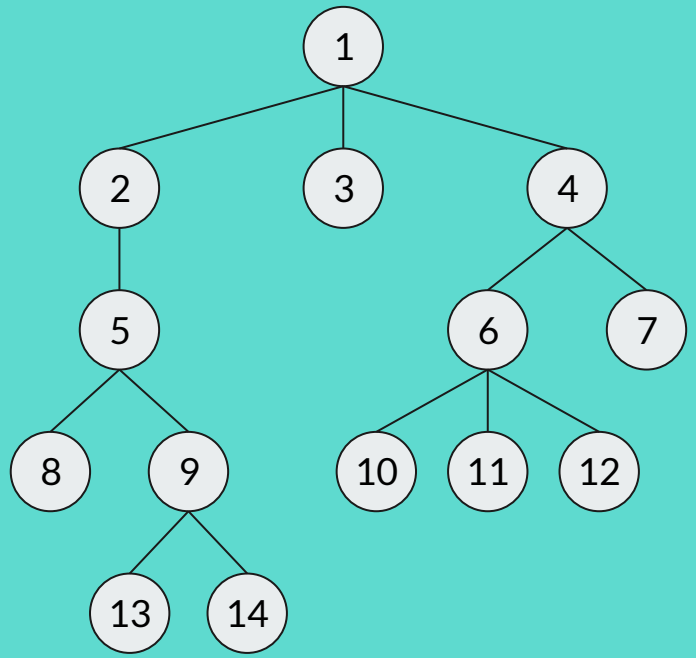
Macro-Micro Tree Decomposition

- Trees
  - Begin with root node
  - Nodes can have child nodes
  - Repeat
  - Nodes without children are **leaves**



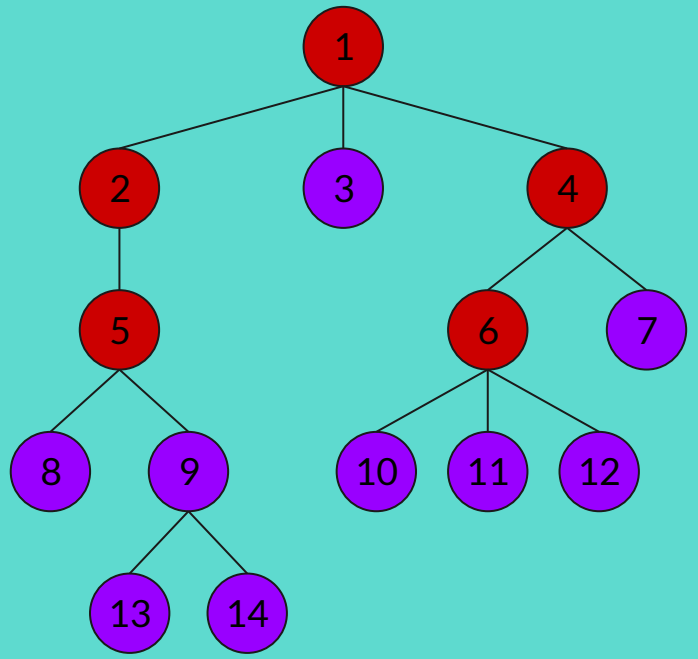
Macro-Micro Tree Decomposition

- Trees
  - Begin with root node
  - Nodes can have child nodes
  - Repeat
  - Nodes without children are **leaves**
- When limited to n nodes, there are exponentially many distinct trees.



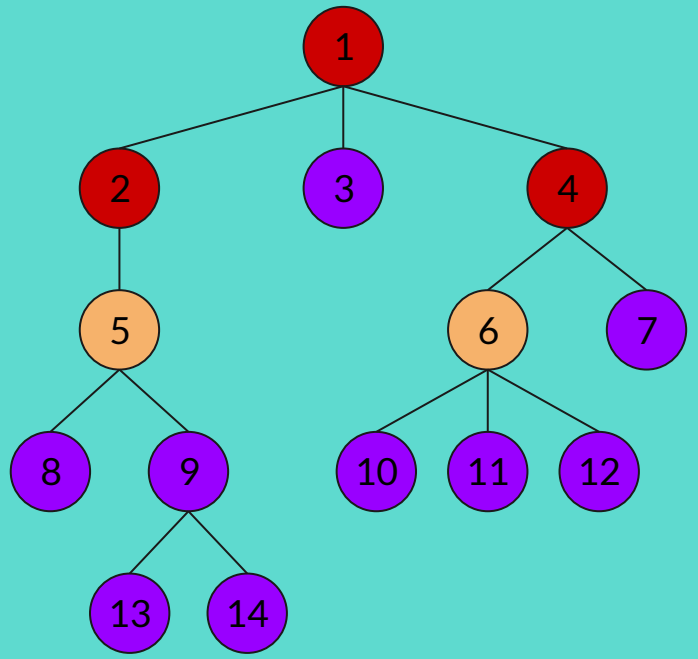
Macro-Micro Tree Decomposition

- Trees
  - Begin with root node
  - Nodes can have child nodes
  - Repeat
  - Nodes without children are **leaves**
- When limited to n nodes, there are exponentially many distinct trees.
- Node is **micro** if has less than O(log(n)) descendants (else **macro**)



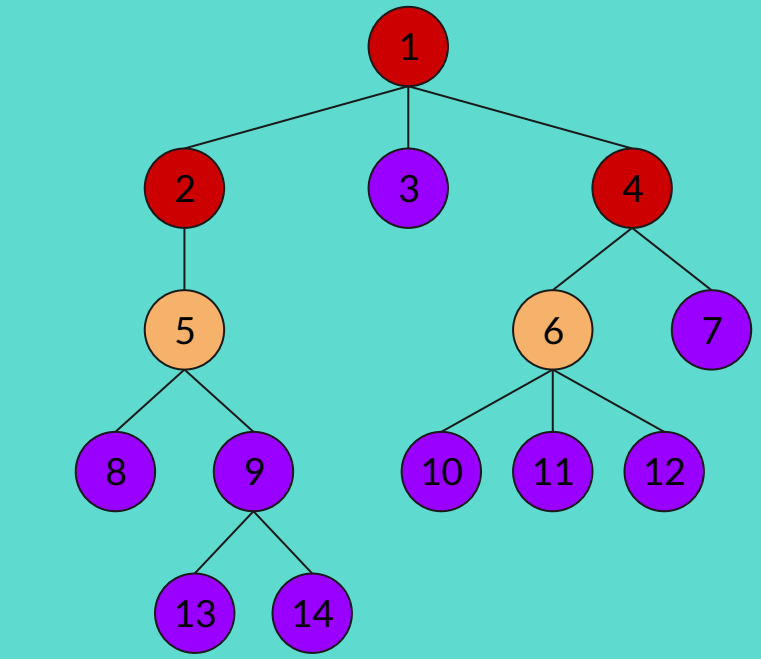
Macro-Micro Tree Decomposition

- Trees
  - Begin with root node
  - Nodes can have child nodes
  - Repeat
  - Nodes without children are **leaves**
- When limited to n nodes, there are exponentially many distinct trees.
- Node is **micro** if has less than O(log(n)) descendants (else **macro**)
- Node is **macro leaf** if it is macro and all its children are micro



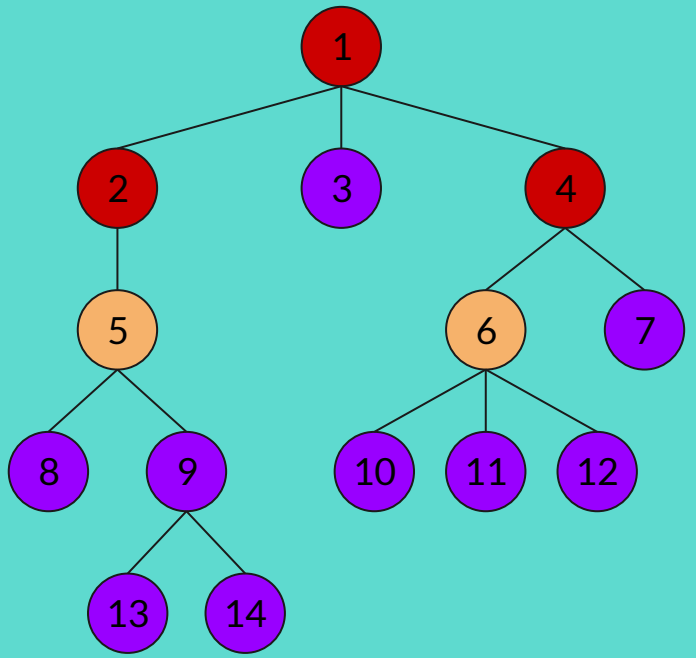
Macro-Micro Tree Decomposition

- Trees
  - Begin with root node
  - Nodes can have child nodes
  - Repeat
  - Nodes without children are **leaves**
- When limited to n nodes, there are exponentially many distinct trees.
- Node is **micro** if has less than O(log(n)) descendants (else **macro**)
- Node is **macro leaf** if it is macro and all its children are micro



Macro-Micro Tree Decomposition

- Trees
  - Begin with root node
  - Nodes can have child nodes
  - Repeat
  - Nodes without children are **leaves**
- When limited to n nodes, there are exponentially many distinct trees.
- Node is **micro** if has less than O(log(n)) descendants (else **macro**)
- Node is **macro leaf** if it is macro and all its children are micro



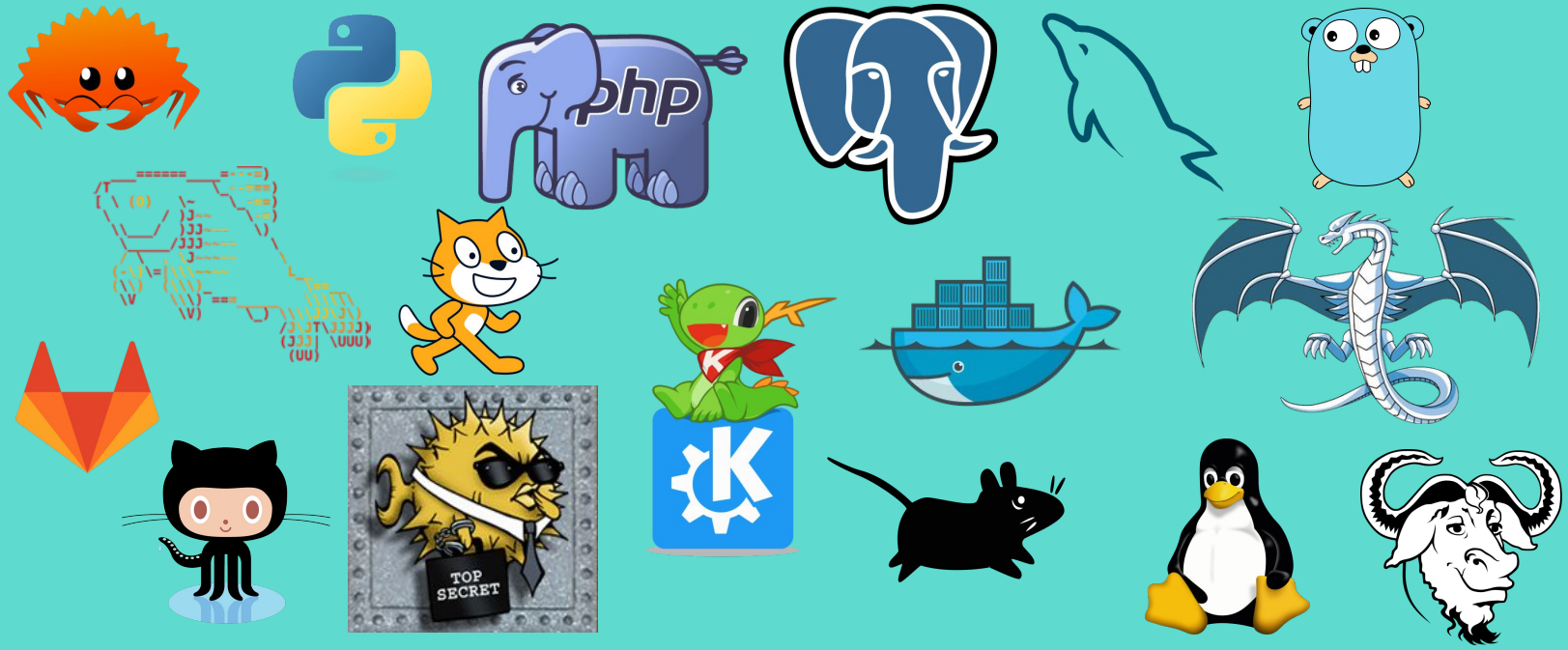
At most O(n/log(n)) macro leaves

Only O(n^(1/c)) distinct microtrees

Animal Computing Mascots

The true meaning of ACM

Animal Computing Mascots



Trans Rights!

https://xenia-linux-site.glitch.me/

Powershell

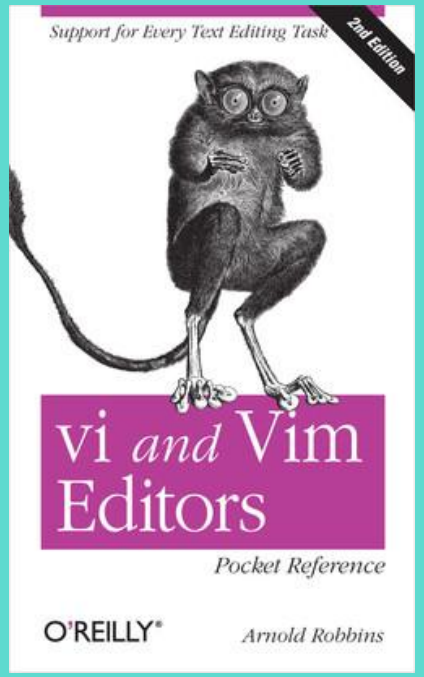
WSL

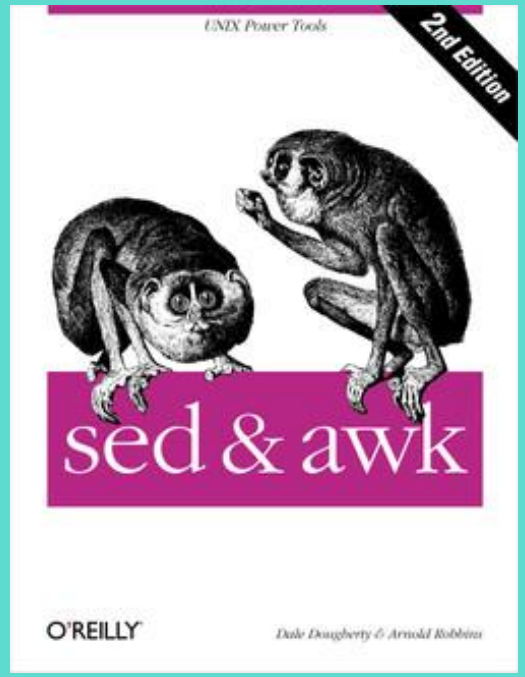
From: Richard Stallman
Subject: WSL
Date: Mon, 23 Jan 2023 22:50:01 -0500

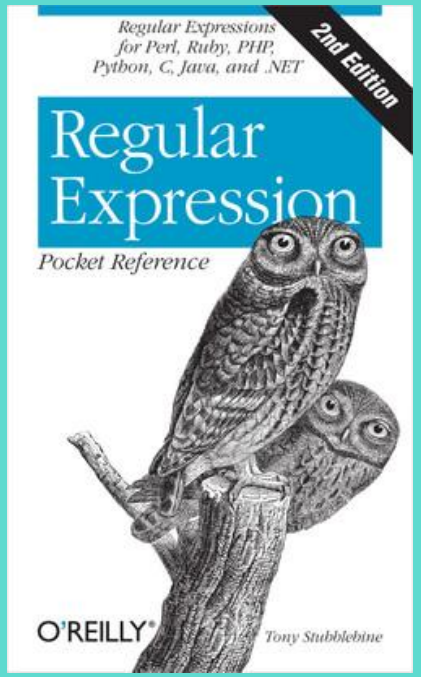
```
[[[ To any NSA and FBI agents reading my email: please consider    ]]]
[[[ whether defending the US Constitution against all enemies,     ]]]
[[[ foreign or domestic, requires you to follow Snowden's example. ]]]

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)
```

https://github.com/codetent/weasel

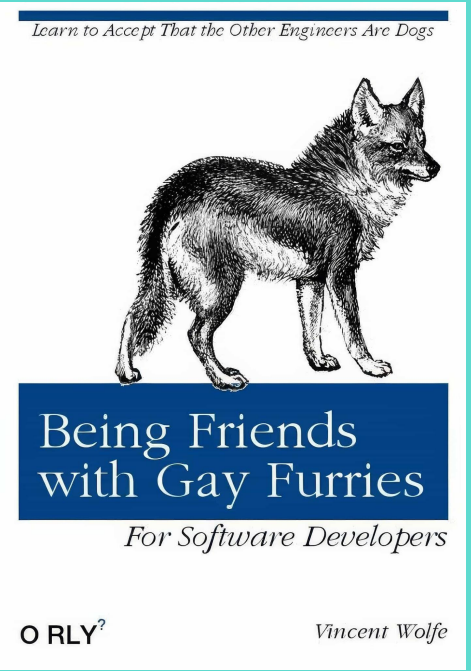
O'Reilly Book Covers

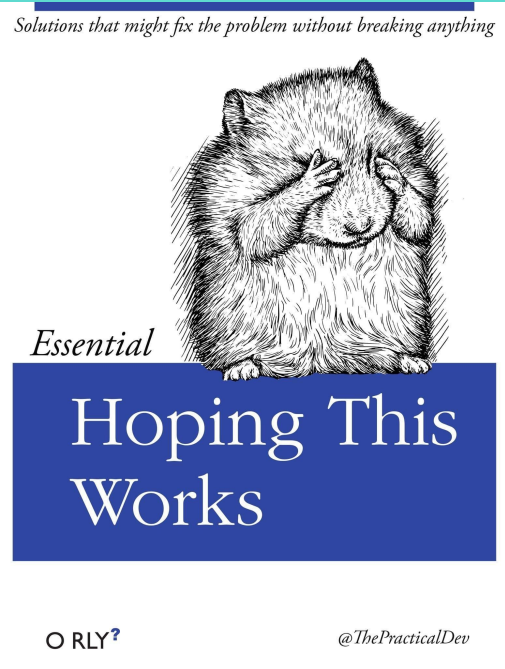






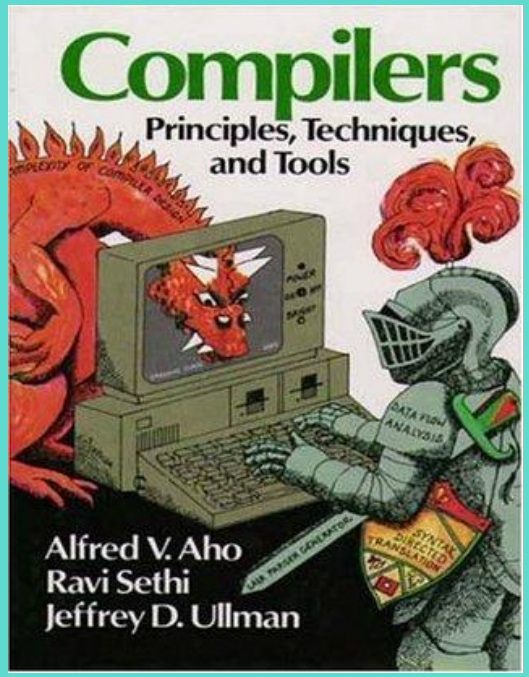
O'RLY Book Covers

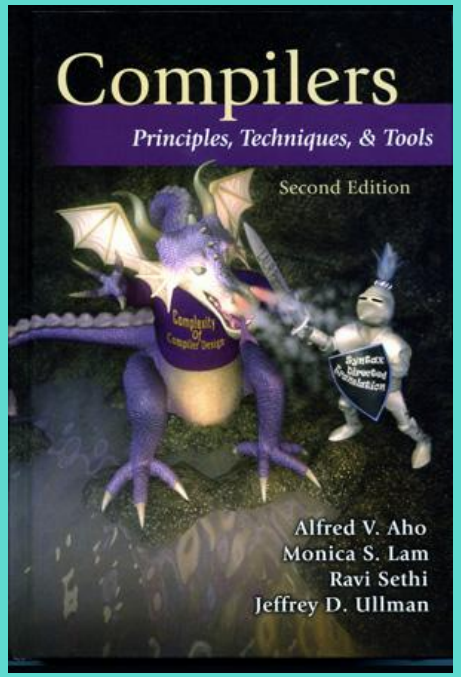




https://orlybooks.com/

Dragon Books





Primal Dual Methods

Lagrange Multipliers

- Suppose we're minimizing $f(x)$ subject to $g(x) = 0$

Lagrange Multipliers

- Suppose we're minimizing $f(x)$ subject to $g(x) = 0$
- Constraint is annoying

Lagrange Multipliers

- Suppose we're minimizing $f(x)$ subject to $g(x) = 0$
- Constraint is annoying
- Introduce multiplier "λ", get $\lambda g(x)$

Lagrange Multipliers

- Suppose we're minimizing $f(x)$ subject to $g(x) = 0$
- Constraint is annoying
- Introduce multiplier "λ", get $\lambda g(x)$
- What happens if we try to maximize λg(x)?
  - When g(x)=0, maximum is 0
  - Otherwise, maximum is infinite

Lagrange Multipliers

- Suppose we're minimizing $f(x)$ subject to $g(x) = 0$
- Constraint is annoying
- Introduce multiplier "λ", get $\lambda g(x)$
- What happens if we try to maximize λg(x)?
  - When g(x)=0, maximum is 0
  - Otherwise, maximum is infinite

$$\min_{x|g(x)=0} f(x) = \min_{x}(\max_{\lambda}(f(x) + \lambda g(x)))$$

$$\mathcal{L}(x, \lambda) = f(x) + \lambda g(x))$$

Lagrange Multipliers

- Suppose we're minimizing $f(x)$ subject to $g(x) = 0$
- Constraint is annoying
- Introduce multiplier "λ", get $\lambda g(x)$
- What happens if we try to maximize λg(x)?
  - When g(x)=0, maximum is 0
  - Otherwise, maximum is infinite
- There's more cool math here, but we'll skip it for time

$$\min_{x|g(x)=0} f(x) = \min_{x}(\max_{\lambda}(f(x) + \lambda g(x)))$$

$$\mathcal{L}(x, \lambda) = f(x) + \lambda g(x))$$

Duality in Linear Programming

$$\min_{Ax \leq b} c^\intercal x$$

Duality in Linear Programming

$$\min_{Ax \leq b} c^\intercal x$$

$$\min_{0 \leq b - Ax} c^\intercal x$$

Duality in Linear Programming

$$\min_{Ax \leq b} c^\intercal x$$

$$\min_{0 \leq b - Ax} c^\intercal x$$

$$\min_{x} \max_{\lambda \leq 0} (c^\intercal x + \lambda^\intercal (b - Ax))$$

Duality in Linear Programming

$$\min_{Ax \leq b} c^\intercal x$$

$$\min_{0 \leq b - Ax} c^\intercal x$$

$$\min_{x} \max_{\lambda \leq 0} (c^\intercal x + \lambda^\intercal (b - Ax))$$

$$\min_{x} \max_{\lambda \leq 0} (c^\intercal x + \lambda^\intercal b - \lambda^\intercal Ax)$$

Duality in Linear Programming

$$\min_{Ax \leq b} c^\intercal x$$

$$\min_{0 \leq b - Ax} c^\intercal x$$

$$\min_{x} \max_{\lambda \leq 0} (c^\intercal x + \lambda^\intercal (b - Ax))$$

$$\min_{x} \max_{\lambda \leq 0} (c^\intercal x + \lambda^\intercal b - \lambda^\intercal Ax) \quad \max_{\lambda \leq 0} \min_{x} (b^\intercal \lambda + x^\intercal c - x^\intercal A^\intercal \lambda)$$

Duality in Linear Programming

$$\min_{Ax \leq b} c^\intercal x$$

$$\min_{0 \leq b - Ax} c^\intercal x$$

$$\min_{x} \max_{\lambda \leq 0} (c^\intercal x + \lambda^\intercal (b - Ax)) \quad \max_{\lambda \leq 0} \min_{x} (b^\intercal \lambda + x^\intercal (c - A^\intercal \lambda))$$

$$\min_{x} \max_{\lambda \leq 0} (c^\intercal x + \lambda^\intercal b - \lambda^\intercal Ax) \quad \max_{\lambda \leq 0} \min_{x} (b^\intercal \lambda + x^\intercal c - x^\intercal A^\intercal \lambda)$$

Duality in Linear Programming

$$\min_{Ax \leq b} c^\intercal x$$

$$\min_{0 \leq b - Ax} \quad c^\intercal x \qquad\qquad \max_{\lambda \leq 0, (c - A^\intercal \lambda) = 0} \quad b^\intercal \lambda$$

$$\min_x \max_{\lambda \leq 0} (c^\intercal x + \lambda^\intercal (b - Ax)) \qquad \max_{\lambda \leq 0} \min_x (b^\intercal \lambda + x^\intercal (c - A^\intercal \lambda))$$

$$\min_x \max_{\lambda \leq 0} (c^\intercal x + \lambda^\intercal b - \lambda^\intercal Ax) \qquad \max_{\lambda \leq 0} \min_x (b^\intercal \lambda + x^\intercal c - x^\intercal A^\intercal \lambda)$$

Duality in Linear Programming

$$\min_{Ax \leq b} c^\intercal x \qquad \max_{\lambda \leq 0, A^\intercal \lambda = c} b^\intercal \lambda$$

$$\min_{0 \leq b - Ax} c^\intercal x \qquad \max_{\lambda \leq 0, (c - A^\intercal \lambda) = 0} b^\intercal \lambda$$

$$\min_{x} \max_{\lambda \leq 0} (c^\intercal x + \lambda^\intercal (b - Ax)) \qquad \max_{\lambda \leq 0} \min_{x} (b^\intercal \lambda + x^\intercal (c - A^\intercal \lambda))$$

$$\min_{x} \max_{\lambda \leq 0} (c^\intercal x + \lambda^\intercal b - \lambda^\intercal Ax) \qquad \max_{\lambda \leq 0} \min_{x} (b^\intercal \lambda + x^\intercal c - x^\intercal A^\intercal \lambda)$$

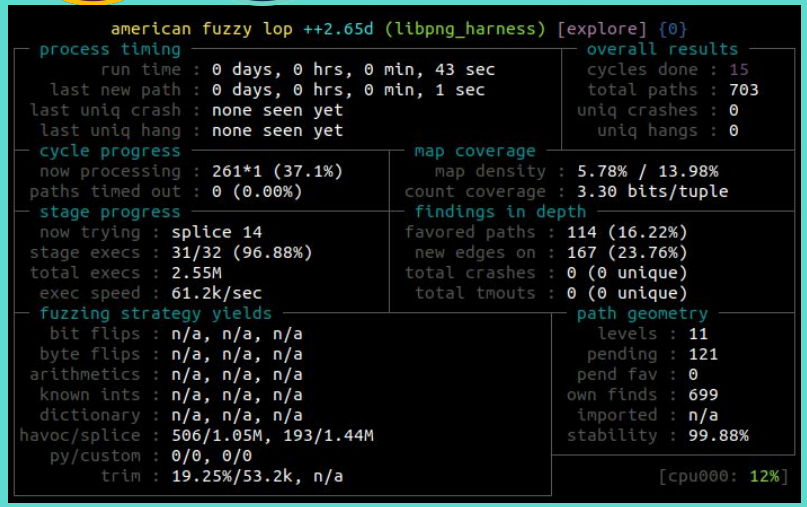
Fuzzing

bottom text

American Fuzzy Lop

American Fuzzy Lop

- Fuzzer: program that automatically searches for buggy input
- Initial idea: purely random input
- Add rules/strategies/heuristics to increase chance of buggy input



https://aflplus.plus/

Data Structures and Algorithms

By Unsigned Long

@InverseHackermann