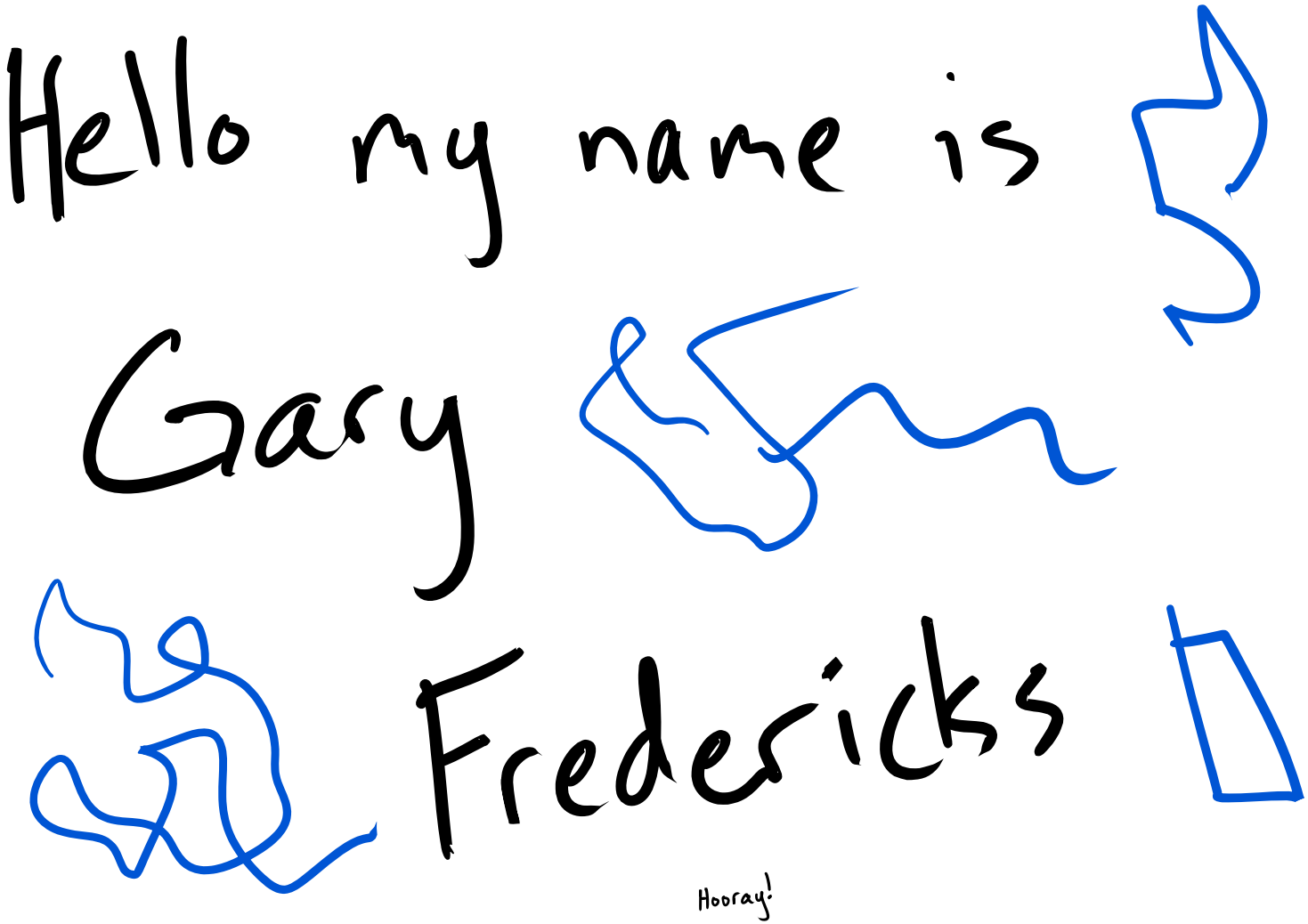
Computationology:

the good weird parts

May 10, 2019

8th Light University

About Me



About Me

What's All This Then?

All of the weird things I know about computational complexity and computability that I can say in under an hour

Disclaimer

It will not help you do your job.

There is insufficient time for

- thoroughness
- precision
- accuracy

The One Idea

"What can computers do?"

+

MATH!

=

surprising structure and variety

Roadmap

- Background
- Complexity
  - More Background
  - The Weird Parts
- Computability
  - More Background
  - The Weird Parts

Background

Can we just talk about recognizing strings instead?

"very easy"

"strings"

"characters"

please

Three Questions

What kind of things can computers do?

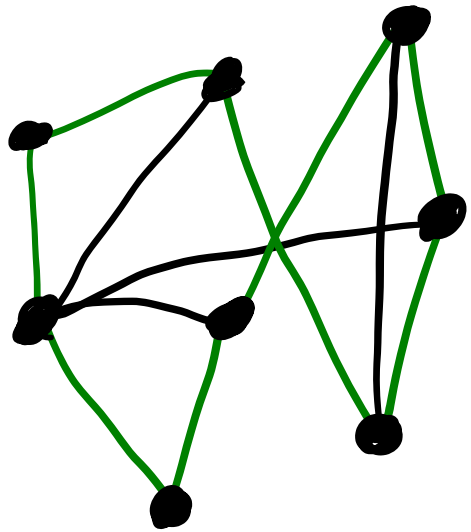
What kind of questions can computers answer?

What kind of strings can computers recognize?

Answering Questions

Do a Facebook!

Find a hamiltonian cycle in a graph



A Single Unit of Code

```
1  // creates a user in the database
2  void createUser(String username, boolean isAdmin){
3    // blah
4    // blah
5    // blah
6  }
```

A Single Pure Function

```
1  // returns a SQL string for creating
2  // a user in the database
3  String createUserStatement(String username,
4                             boolean isAdmin){
5    return "INSERT INTO USERS" +
6           "(username, isAdmin, etc)" +
7           " VALUES (" +
8    // blah
9    // blah
10   // blah
11 }
```

From answering questions to recognizing strings

What kind of things can computers do?

What kind of questions can computers answer?

What kind of strings can computers recognize?

A Single Pure Stringy Function

```
1  // returns a SQL string for creating
2  // a user in the database
3  String createUserStatement
4      (String usernameAndAdminFlag){
5    return "INSERT INTO USERS" +
6           "(username, isAdmin, etc)" +
7           " VALUES (" +
8    // blah
9    // blah
10   // blah
11 }
```

Requirements Gathering

Input	Output
""	"ERROR"
"_"	"ERROR"
...	...
...	...
"[\"\",false]"	"INSERT INTO USERS (username, isAdmin, etc) VALUES('',false,..."
...	...
...	...
"[\"gfredericks\",false]"	"INSERT INTO USERS (username, isAdmin, etc) VALUES('gfredericks',false,..."
...	...
...	...

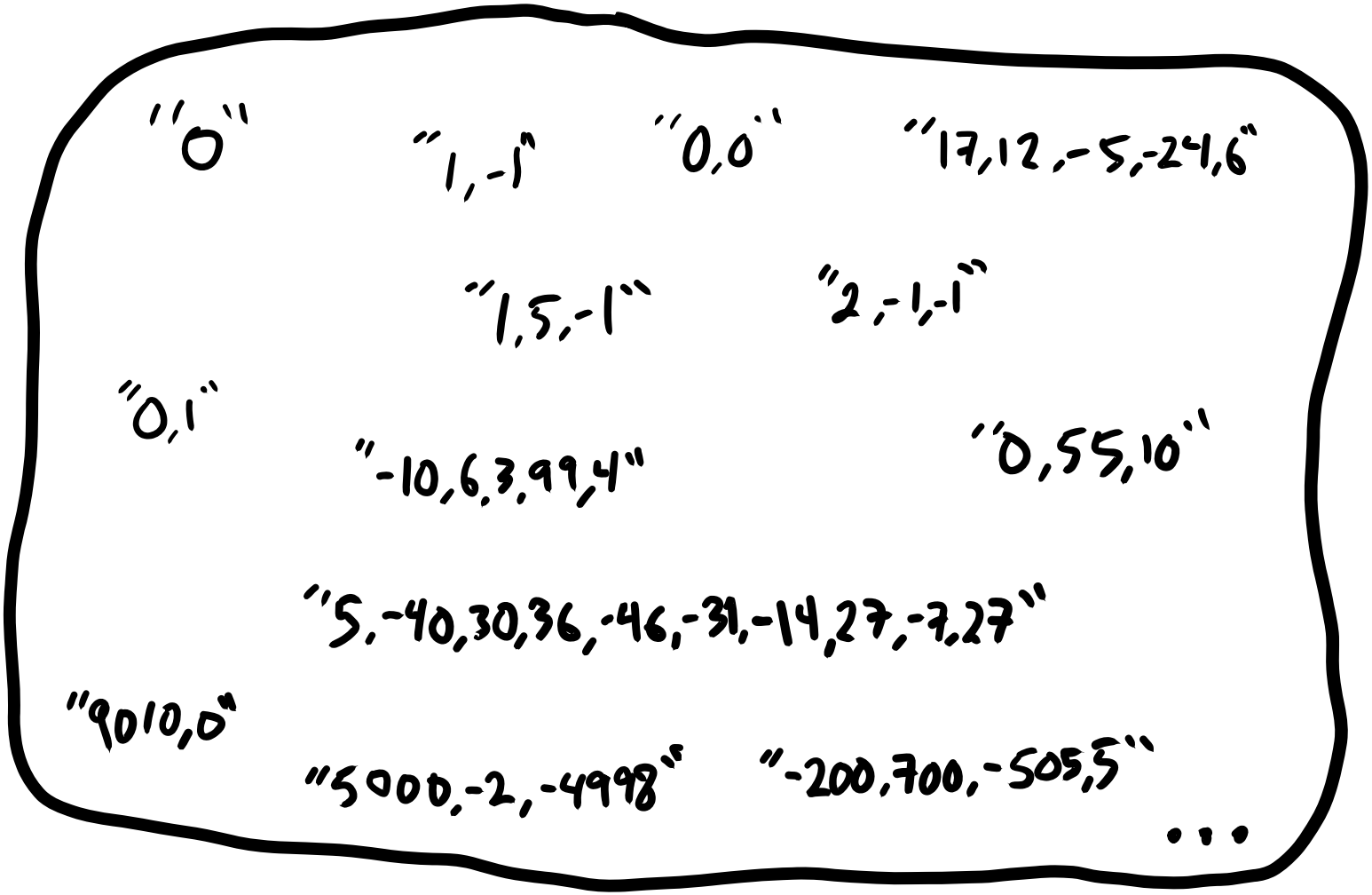
Predicates!

```
1  // returns true if the normal return
2  // value is lexicographically prior
3  // to the given retThreshold
4  boolean createUserStatement
5      (String usernameAndAdminFlagAndRetThreshold){
6    // blah
7    // blah
8    // blah
9  }
```

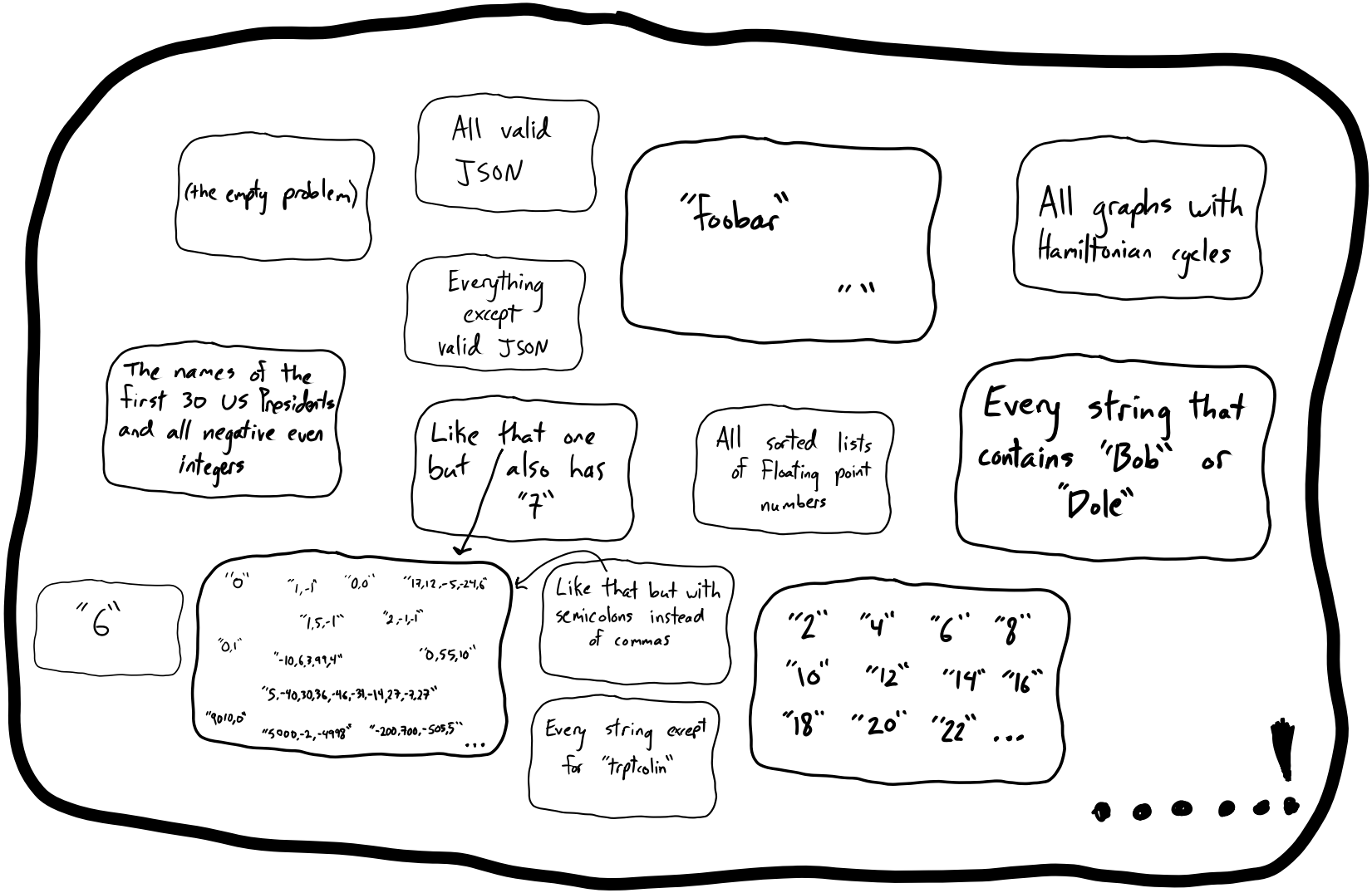
Predicate Requirements

- `"[\"\",false,\"\"]"`
- `"[\"\",false,\"a\"]"`
- `...`
- `...`
- `"[\"mdukakis\",true,\"INSERT and stuff\"]"`
- `...`
- `...`

The Subset Sum Problem



All Sets Of Strings



Classes of Problems

Background

What's a Computer?

It doesn't matter too much

- idealized variant/subset of `$FAVORITE_PROGRAMMING_LANGUAGE`
- turing machine
- circuit families
- lambda calculus
- SKI calculus

Background

The End of the Beginning

The End of the Beginning

- A problem is a (probably infinite) set of strings
- Answering "what can computers do" means determining what kinds of sets of strings there are
- We're focusing on basic, familiar models of computation

Complexity

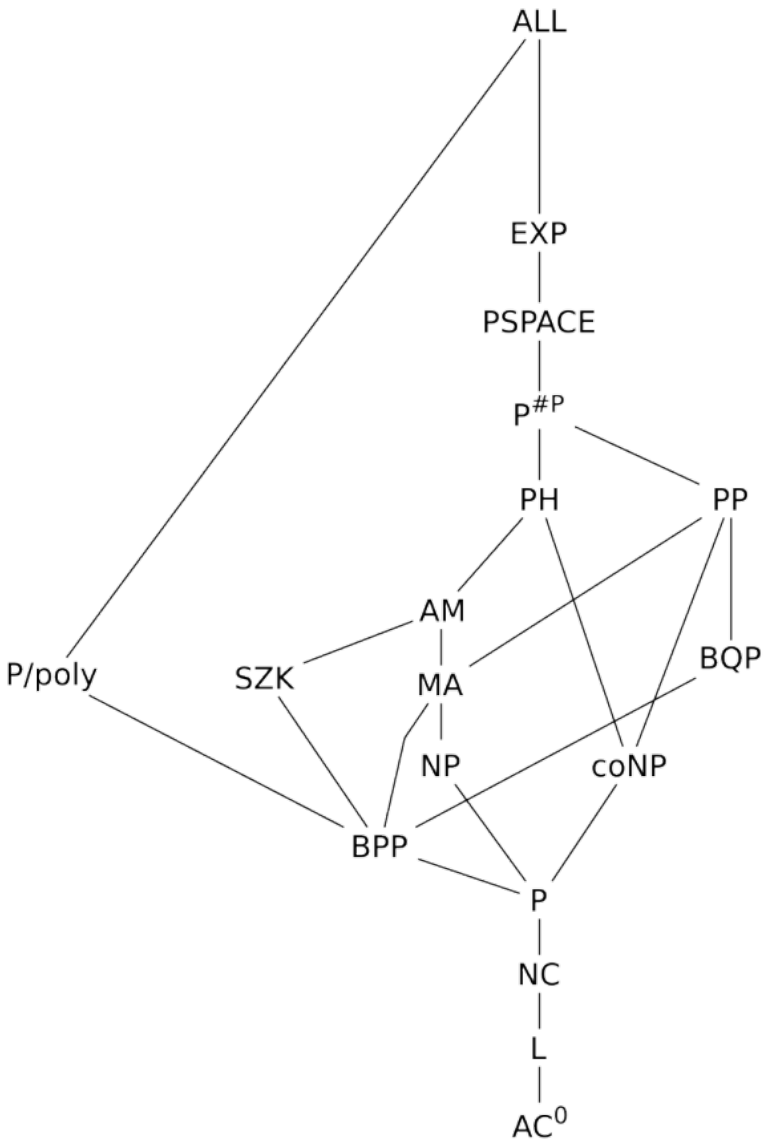
Complexity

More Background

Motivation

Resources	Features	Success Criteria
time	randomness	deterministic
storage space	quantum effects	nondeterministic
circuit size	oracles	probabilistic
circuit depth	useful "advice"	approximation ratio
	special gates	counting
	parallelism	

Complexity Classes



Resources are constrained as a function of input size

For some specific program:

Input Size	Max Running Time	Max Space Used
1	7	3
2	20	13
3	38	14
4	94	18
5	117	20
6	300	82
7	371	321
8	1349	440
9	1544	1463
10	8043	4791
...	...	...

Asymptotics

$$f(n) = 0.03n^2$$

grows faster than

$$f(n) = 856423712n$$

Asymptotics

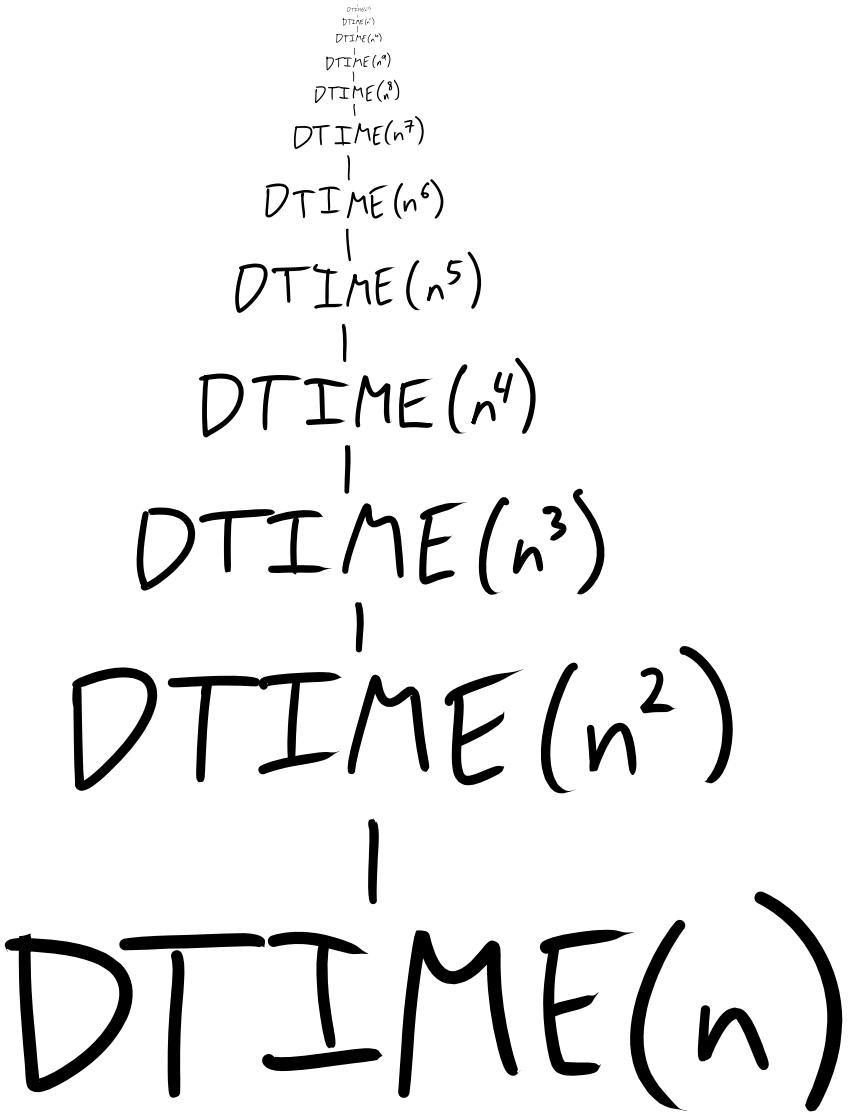
n	856423712n	$0.03n^2$
0	0	0
1	856423712	0
2	1712847424	0
3	2569271136	0
...	...	...
1000	856423712000	30000
1001	857280135712	30060
1002	858136559424	30120
1003	858992983136	30180
1004	859849406848	30240
...	...	...
28547457066	24448719148624348992	24448719148053399850
28547457067	24448719149480772704	24448719149766247274
...	...	...

Asymptotics

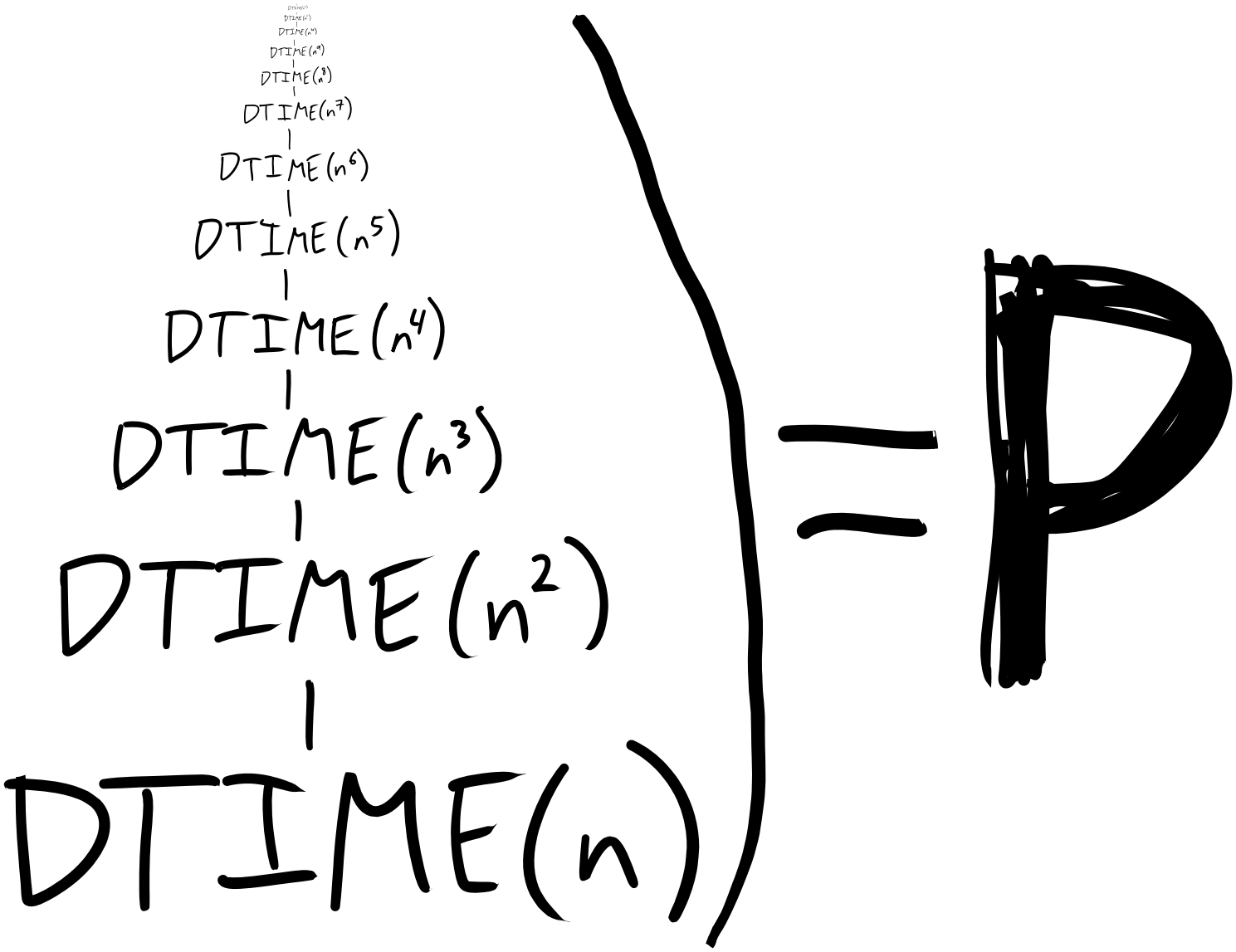
- $0.03n^2$ grows faster than $856423712n$
- an^3 grows faster than bn^2
- an^4 grows faster than bn^3
- an^5 grows faster than bn^4
- an^6 grows faster than bn^5
- ...
- 2^n grows faster than $an^{1005045580}$

Big Oh

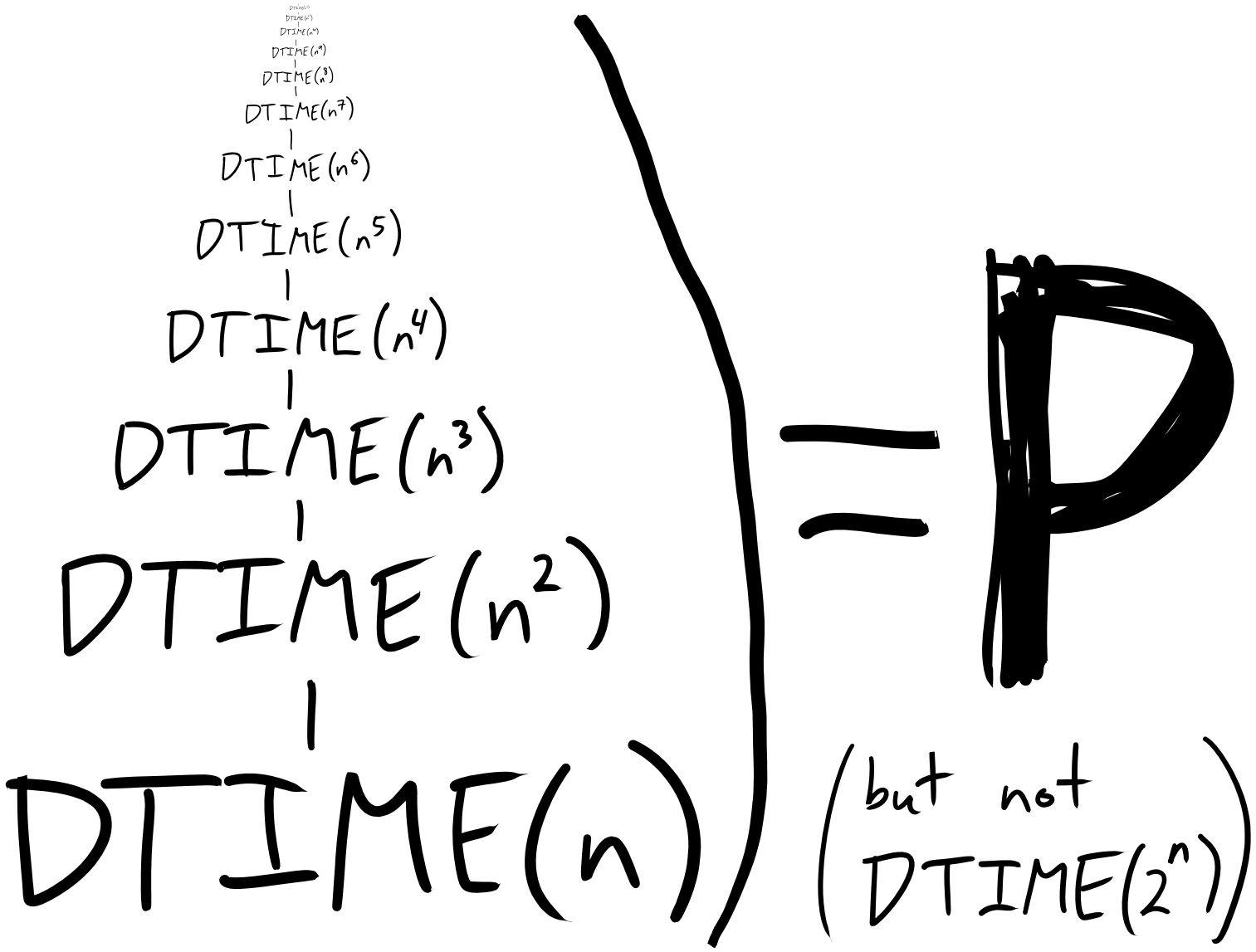
DTIME



P



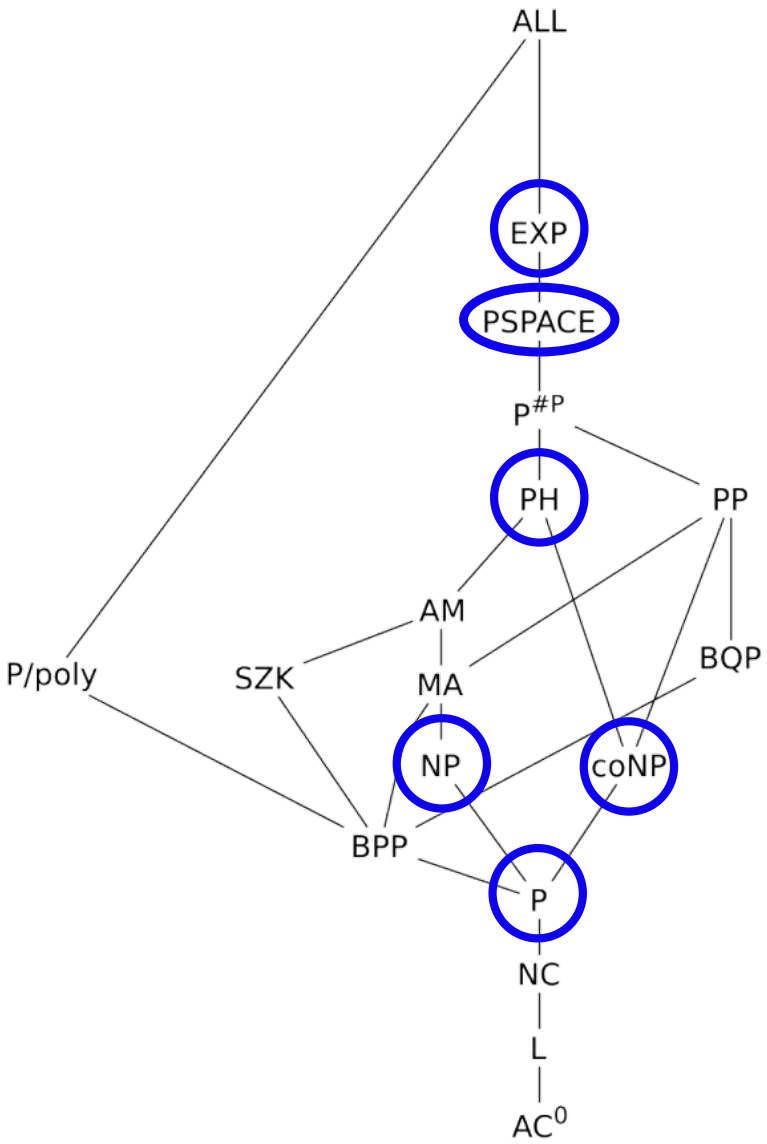
P


DTIME(n^7)
DTIME(n^6)
DTIME(n^5)
DTIME(n^4)
DTIME(n^3)
DTIME(n^2)
DTIME(n)
=P
(but not DTIME(2^n))

Why does polynomial-time mean "efficient"?

- it seems to roughly mean that in practice
  - realistic problems with efficient solutions have polynomial-time solutions
  - most realistic problems with no known efficient solutions have no known polynomial-time solutions
- it's a very convenient class to work with
  - details of computational model, encoding, largely don't matter
  - polynomial-time algorithms compose

Complexity Classes

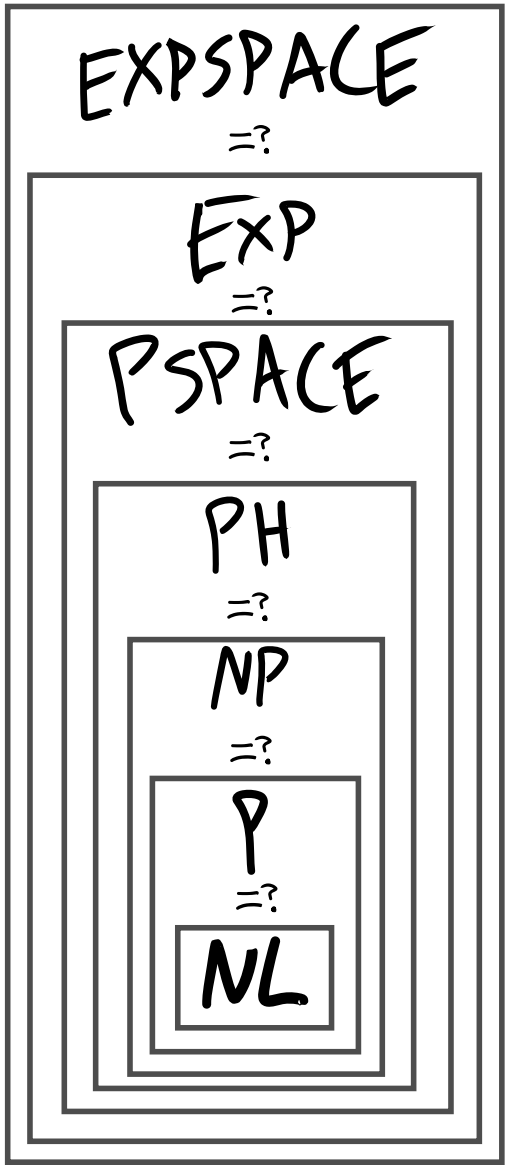


NP

NP Problems

- Boolean satisfiability problem
- Knapsack problem
- Hamiltonian path problem
- Travelling salesman problem
- Subgraph isomorphism problem
- Subset sum problem
- Clique problem
- Vertex cover problem
- Independent set problem
- Dominating set problem
- Graph coloring problem
- Integer factoring problem
- Graph isomorphism problem
- Discrete log problem
- ...

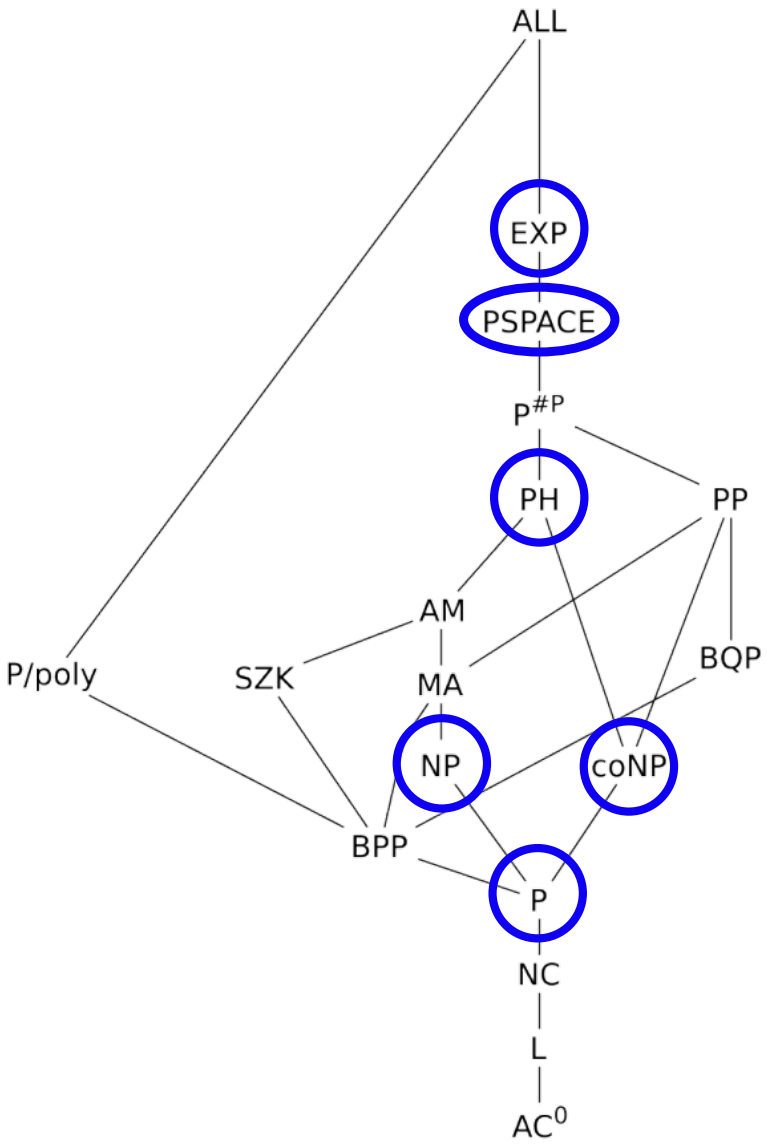
P vs NP vs . . .



Complexity

The Weird Parts

Complexity Classes



The Asymmetry of NP

Proof that `5,-40,30,36,-46,-31,-14,27,-7,27` has a sum-zero subset:

```
-40,-14,27,27
```

Proof that `22,-26,-32,-38,45,45,50,49,34,7` does not have a sum-zero subset:

???

coNP

E.g., the primality problem

- Proof that `1557088186273` is a prime
  - ???
- Proof that `1557088068331` is not a prime
  - `1557088068331 = 2741 * 568072991`

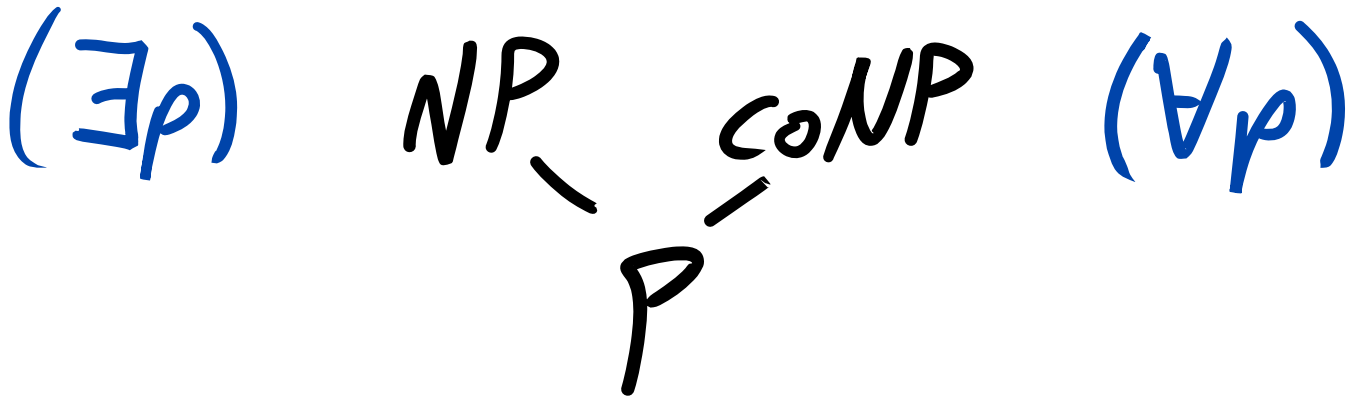
NP and coNP as quantified polynomial-time checks

P: p

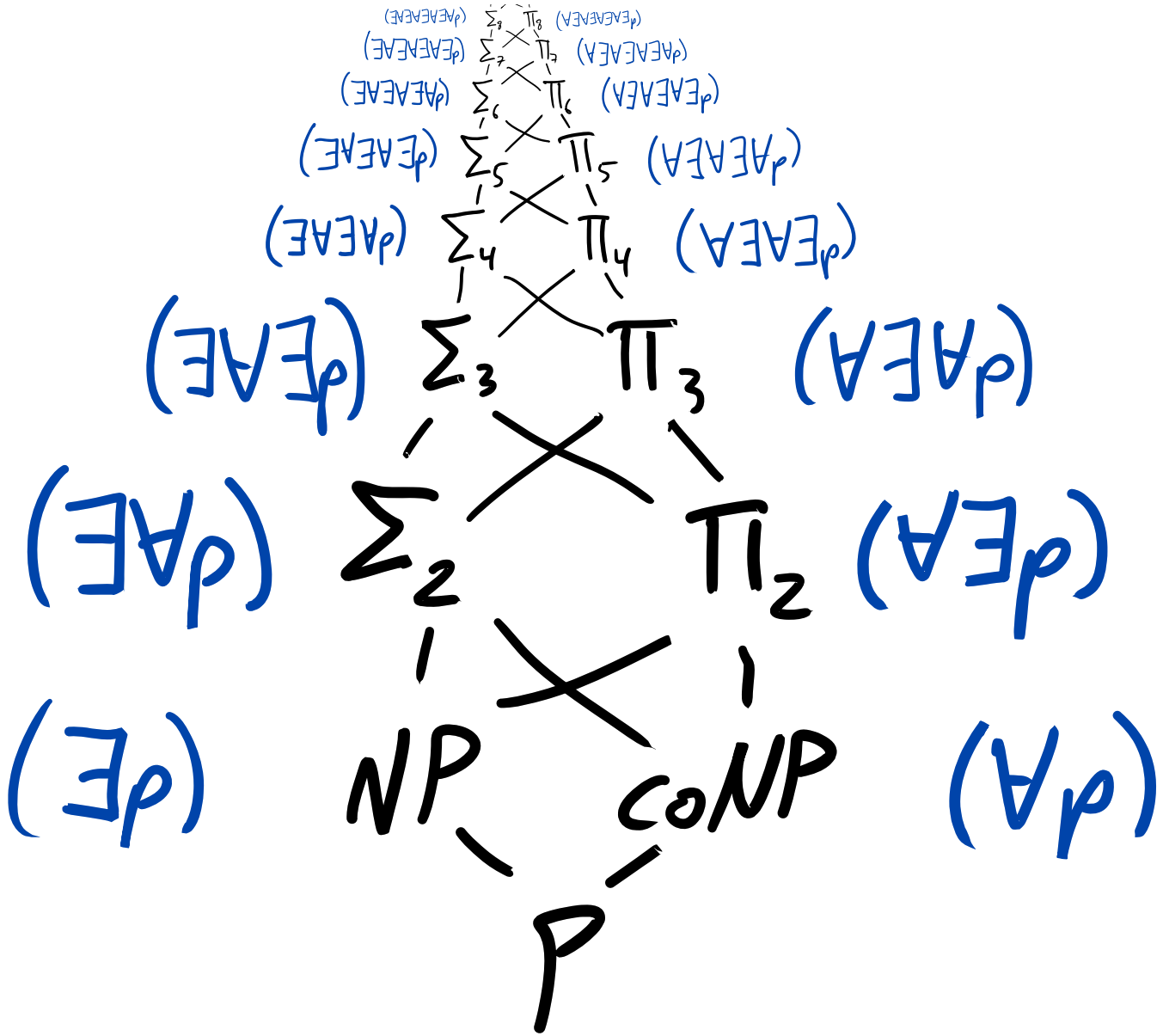
NP: $\exists p$

coNP: $\forall p$

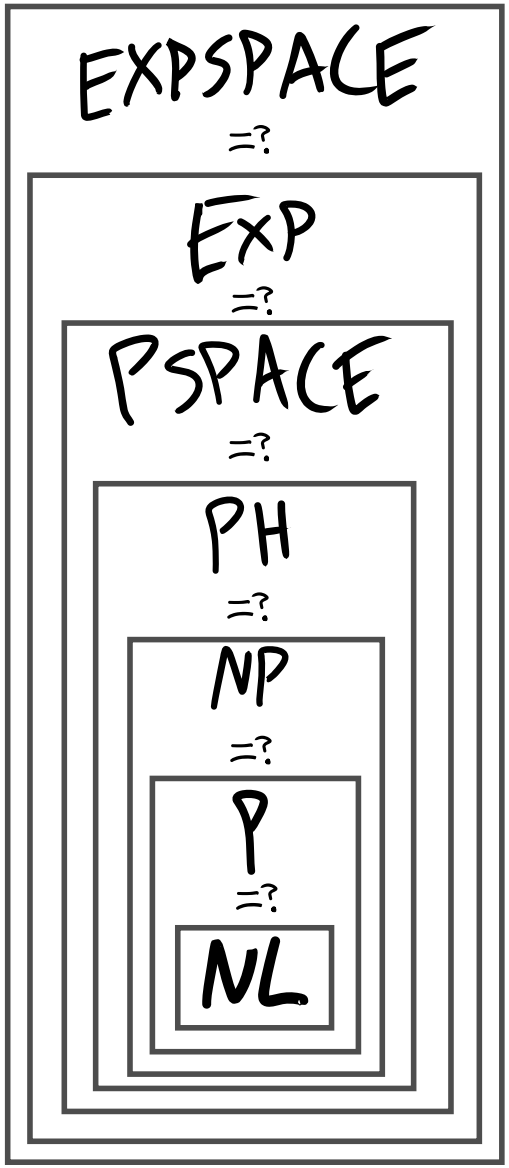
PH



PH



PH Distinctness



Suppose $\Sigma_3 = \Pi_3$

$$\exists\forall\exists p = \forall\exists\forall p$$

Suppose $\Sigma_3 = \Pi_3$

$$\exists\forall\exists p = \forall\exists\forall p$$

$$p\exists\forall\exists\forall$$

Suppose $\Sigma_3 = \Pi_3$

$$\exists\forall\exists p = \forall\exists\forall p$$

$$\forall\exists\forall\exists\forall p$$

$$\forall\exists\forall\exists\forall p$$

Suppose $\Sigma_3 = \Pi_3$

$$\exists\forall\exists p = \forall\exists\forall p$$

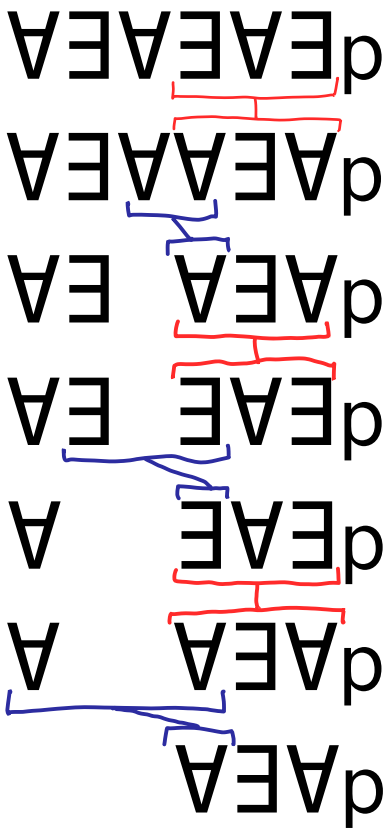
$$\forall\exists\forall\exists\forall\exists p$$

$$\forall\exists\forall\exists\forall p$$

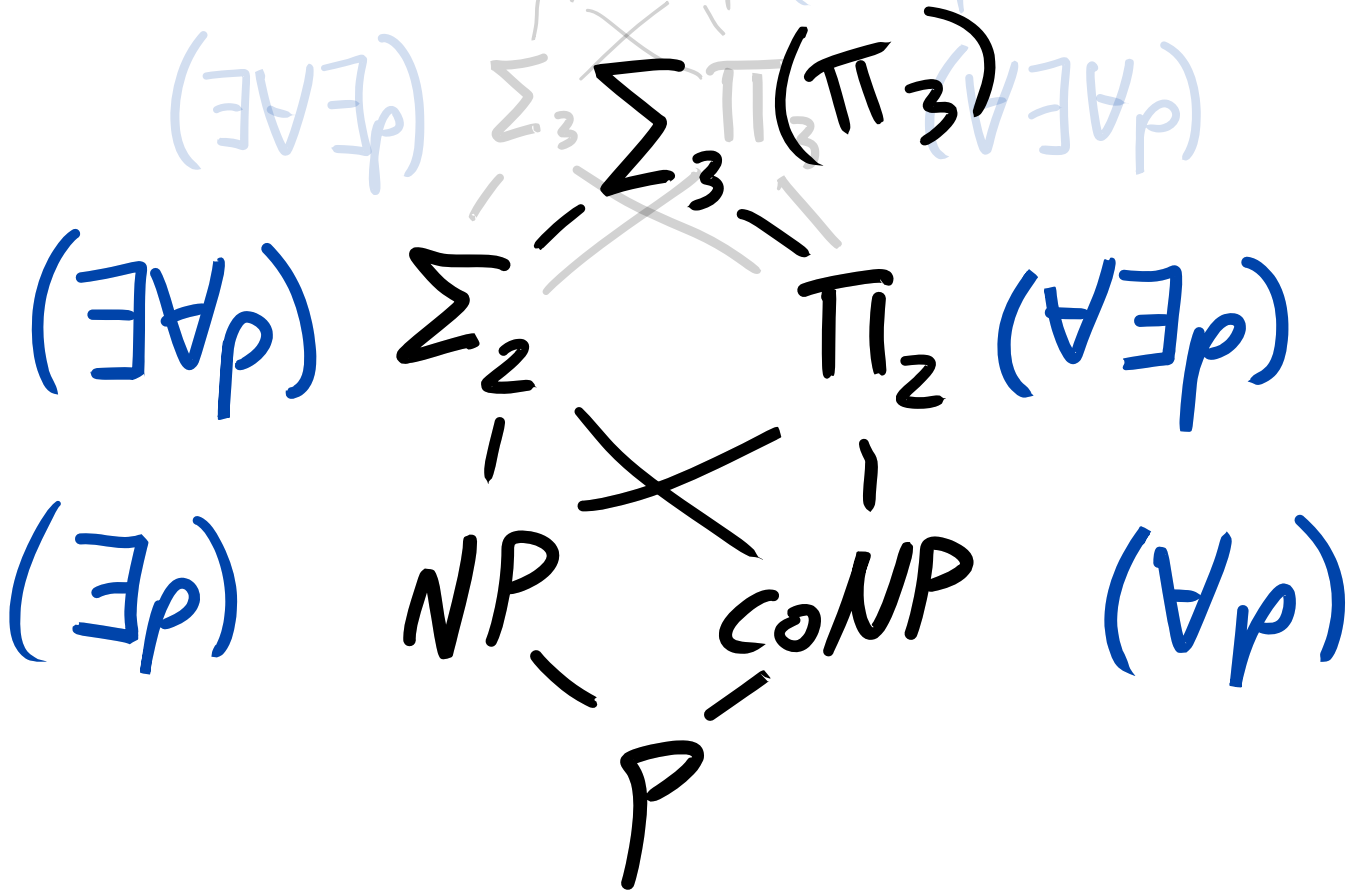
$$\forall\exists\forall\exists p$$

Suppose $\Sigma_3 = \Pi_3$

$$\exists\forall\exists p = \forall\exists\forall p$$



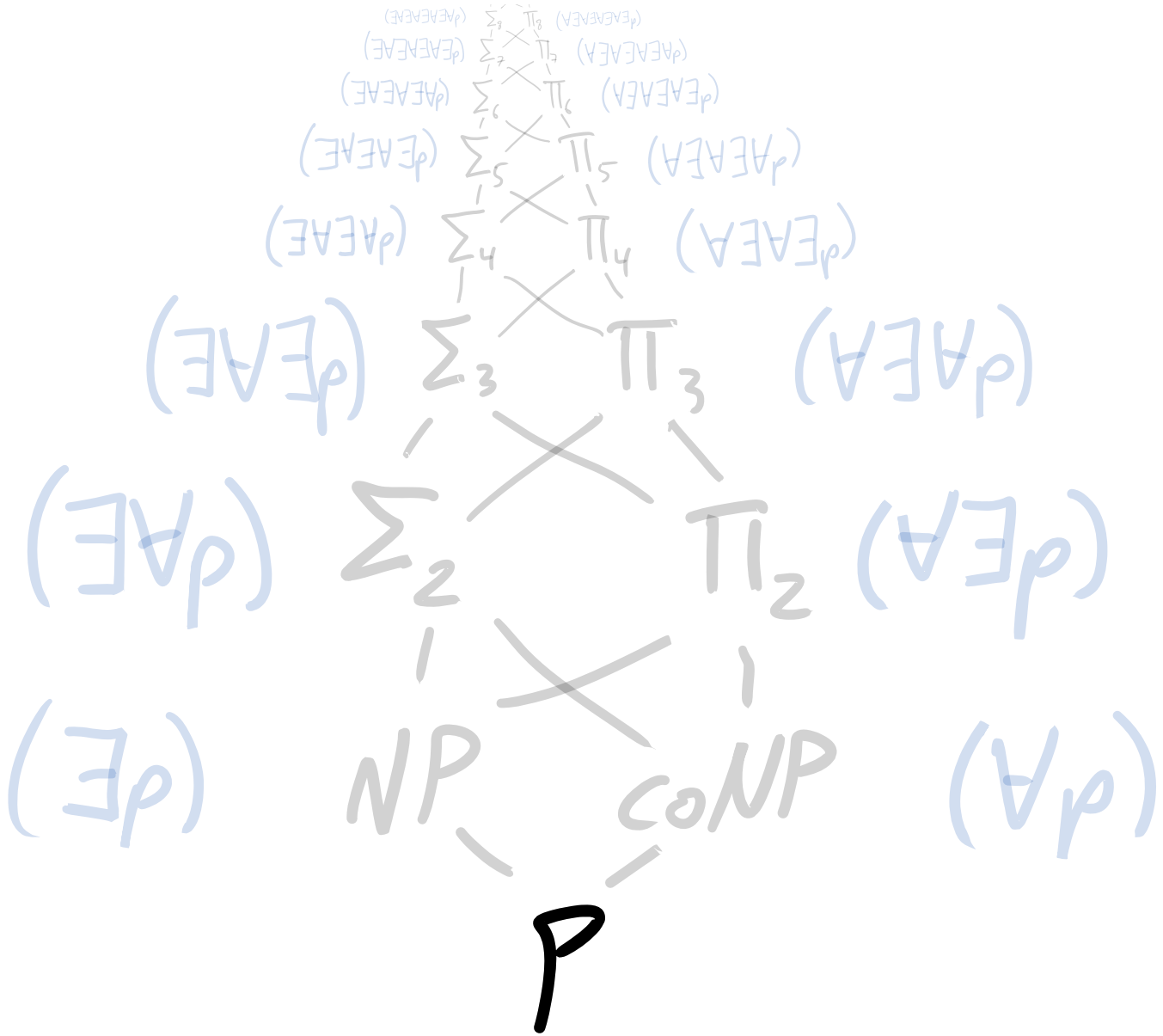
$\Sigma_3 = \Pi_3$ Collapse



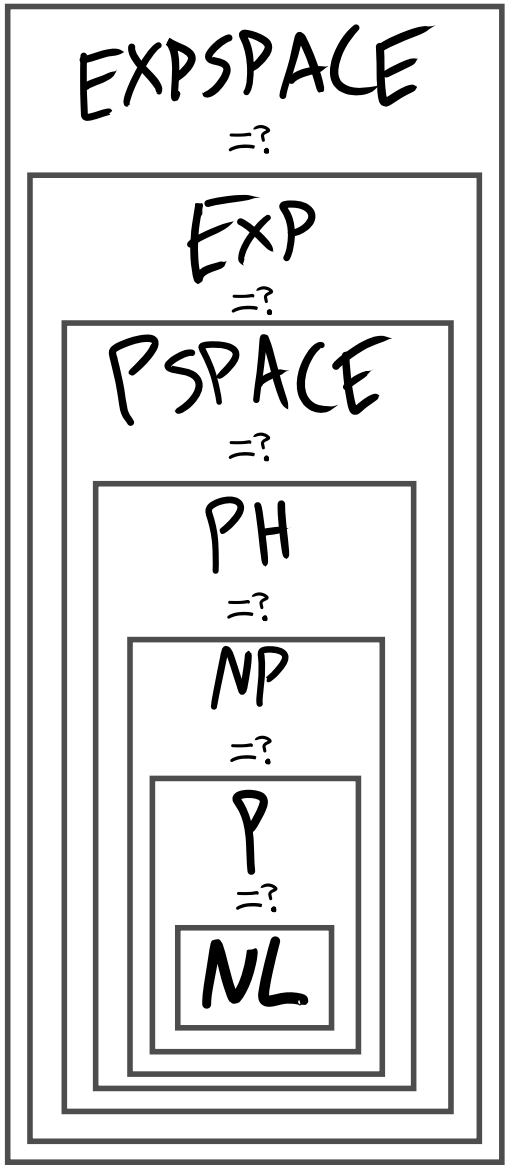
What If

What if P = NP?

P = NP Collapse



P = PSPACE?



Complexity

So what was all that

So what was all that

- Complexity studies how different constraints affect what problems can be solved, and how those constraints relate to each other
- P, NP, PH, PSPACE
- P = NP implies P = PH
- P =? PSPACE

Computability

Computability

More Background

Motivation

What problems can computers solve at all?

The Halting Problem

```
1  // returns true if the method described in the
2  // argument ever returns when called
3  boolean returns?(String zeroArgJavaMethod){
4    // eval the string and call the method???
5  }
```

returnsThwarter

```
1  void returnsThwarter(){
2
3    // use quine tricks to get source code
4    String myOwnSourceCode = "...";
5
6    if(returns?(myOwnSourceCode)){
7      // it says we return, so loop instead
8      while(true){}
9    } else {
10     // it says we don't return, so return
11     return;
12   }
13 }
```

Rice's Theorem and Reducibility

- does a function ever return?
- does a program ever print `"foobar"`?
- does a program ever throw an exception?
- does a program ever do `$ANY_SPECIFIC_THING`?

The Halting Problem Is Half Impossible

- you can solve the yes-half, but not the no-half
- called "computably enumerable"

Computability

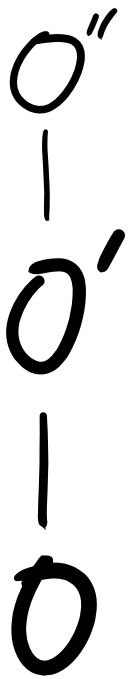
The Weird Parts

0 and 0′

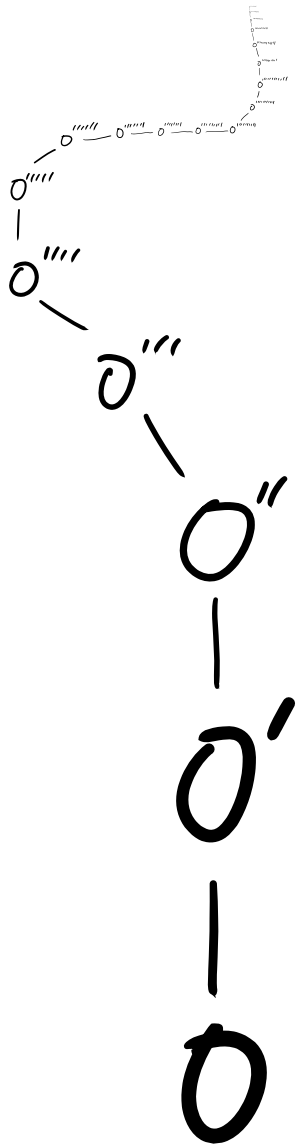
The Turing Jump

```
1  import HaltingChecker;
2
3  public class ProgramWithSuperPowers {
4    // ...
5  }
```

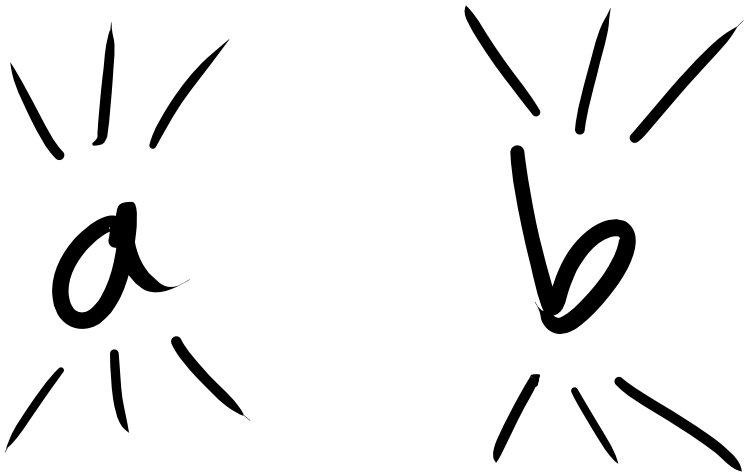
0″



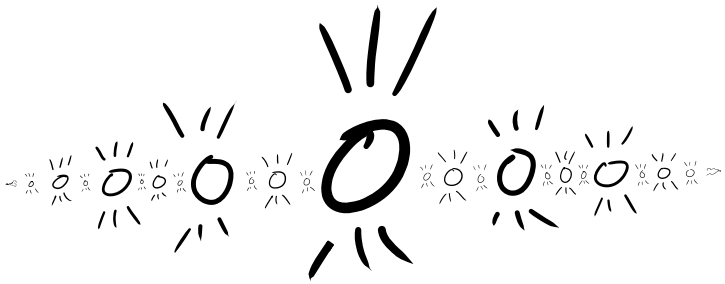
Increasingly Impossible

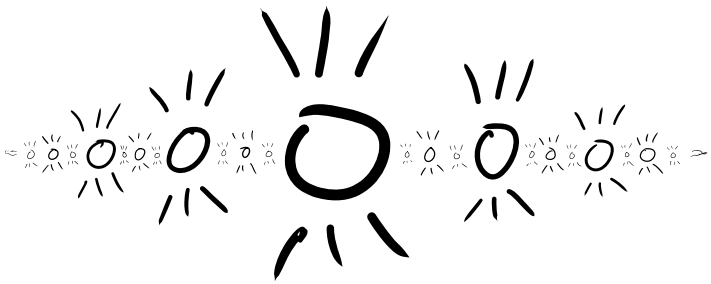


Every nonzero degree has incomparable degrees

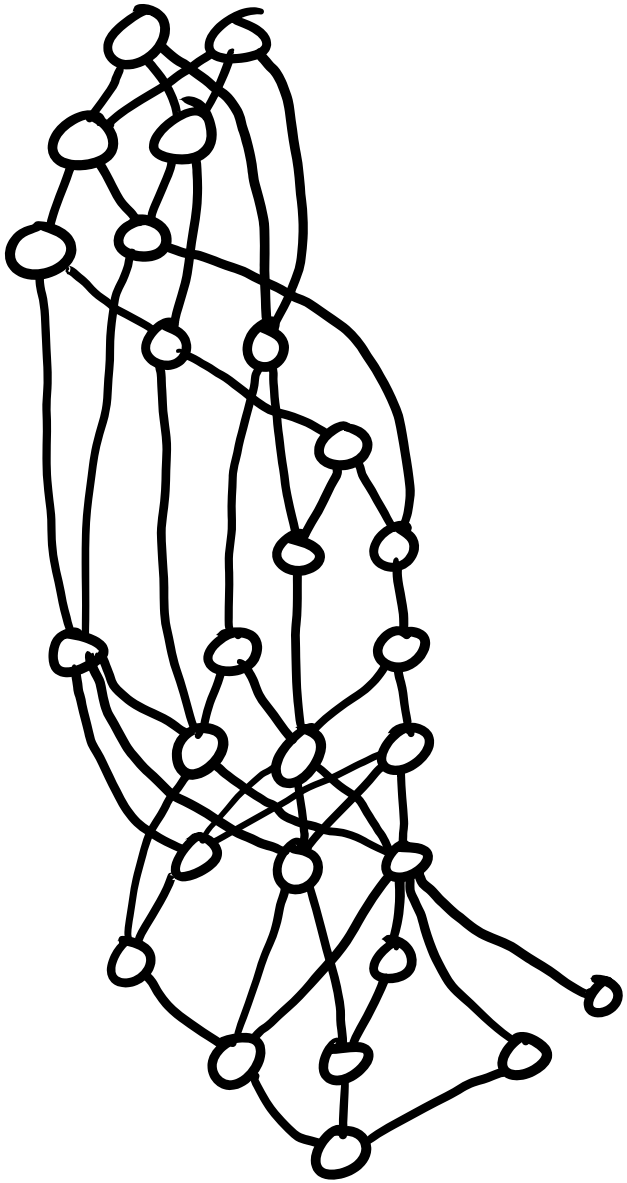


There's a set of uncountable pairwise incomparable degrees

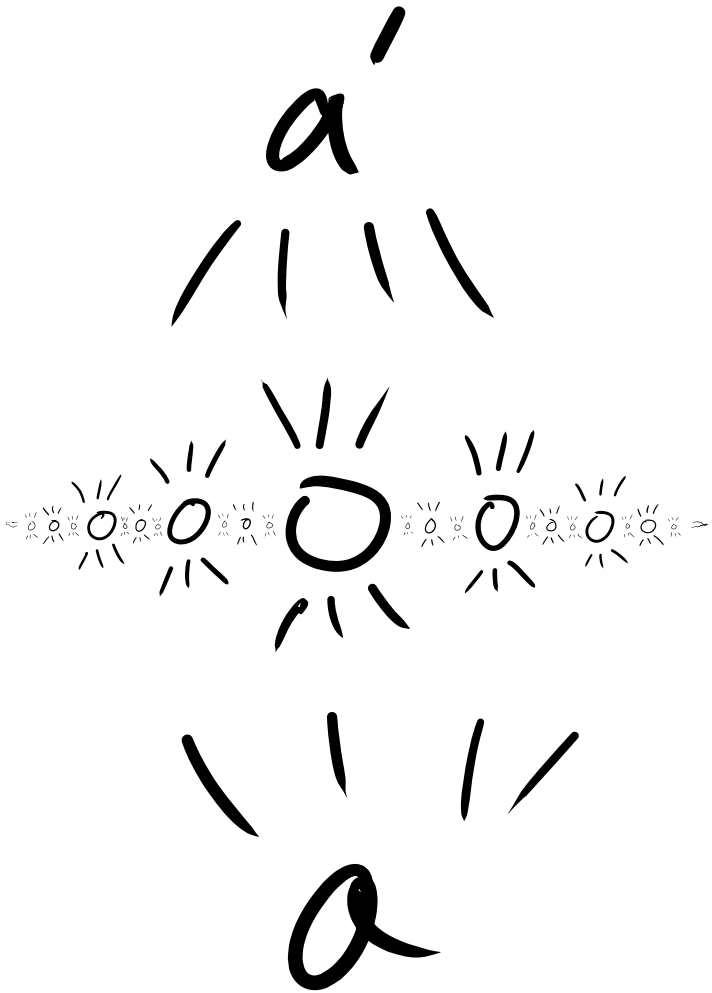




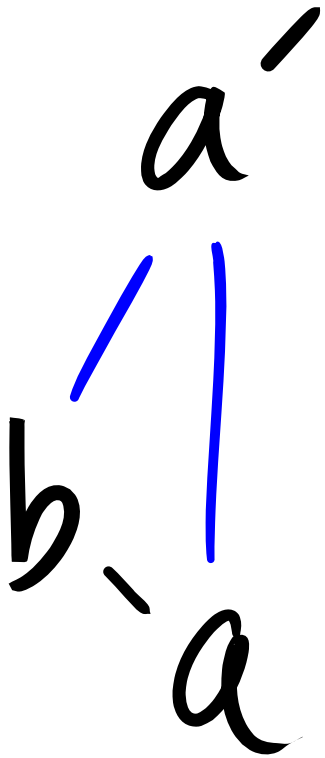
Every countable poset can be embedded



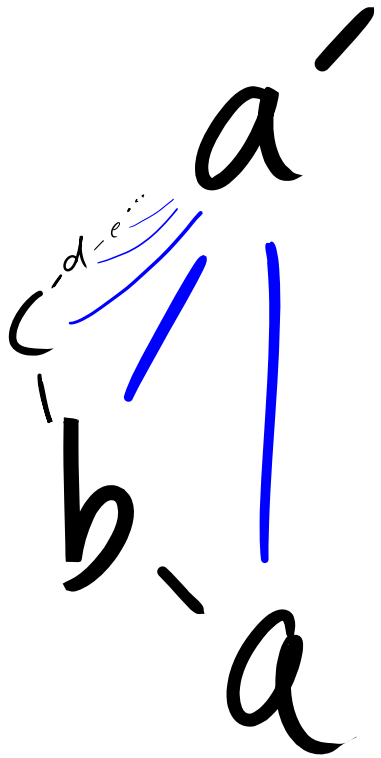
There's a countable sequence of pairwise incomparable degrees between a and a'



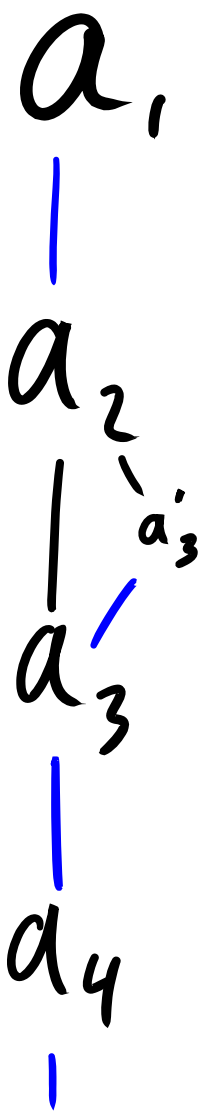
Any degree a has a b such that $a < b$ and $b' = a'$



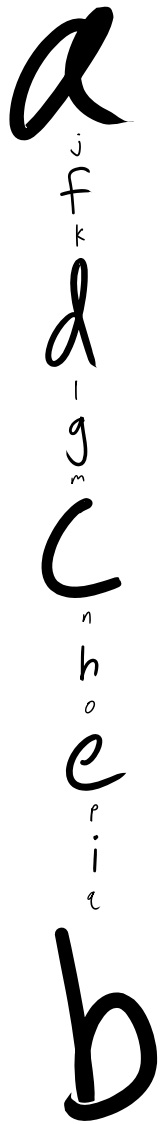
Any degree a has a b such that $a < b$ and $b' = a'$



There's a infinite descending sequence a_i w/ $a'_{i+1} \leq a_i$

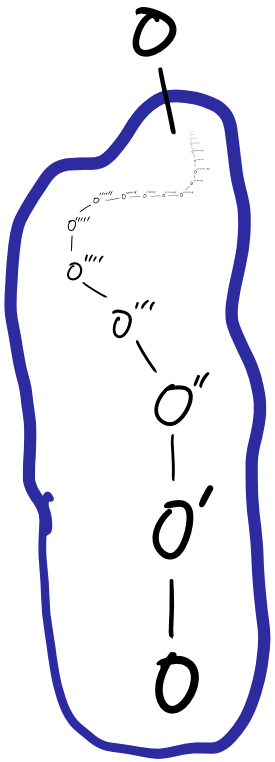


The computably enumerable degrees are dense

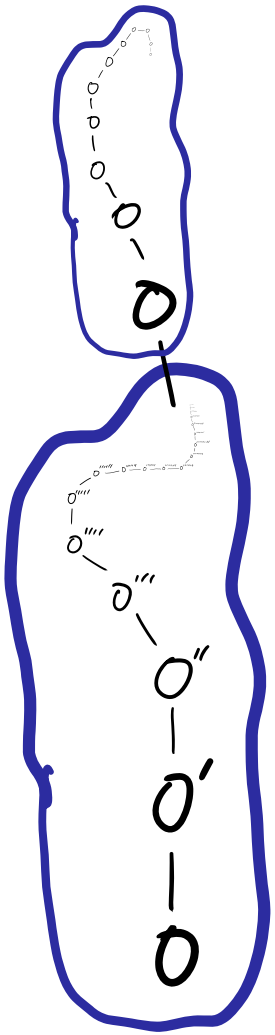


Even more degrees

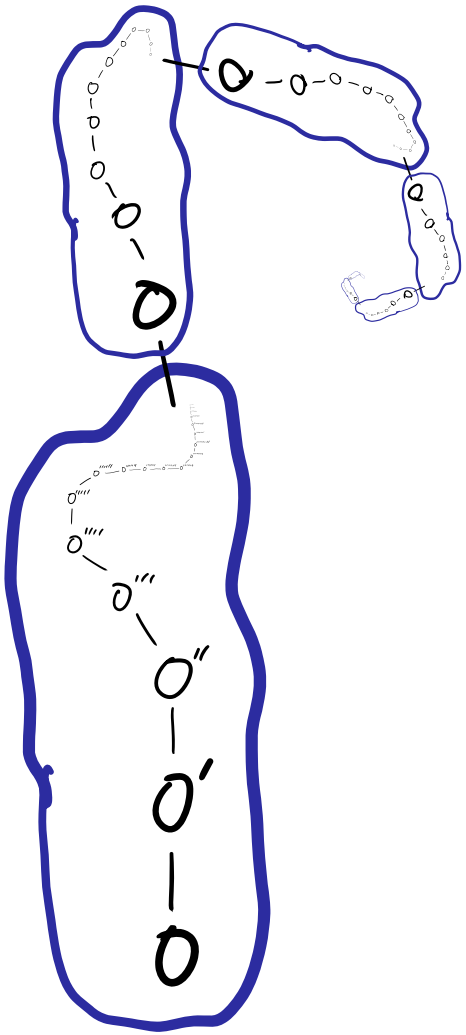
Even more degrees



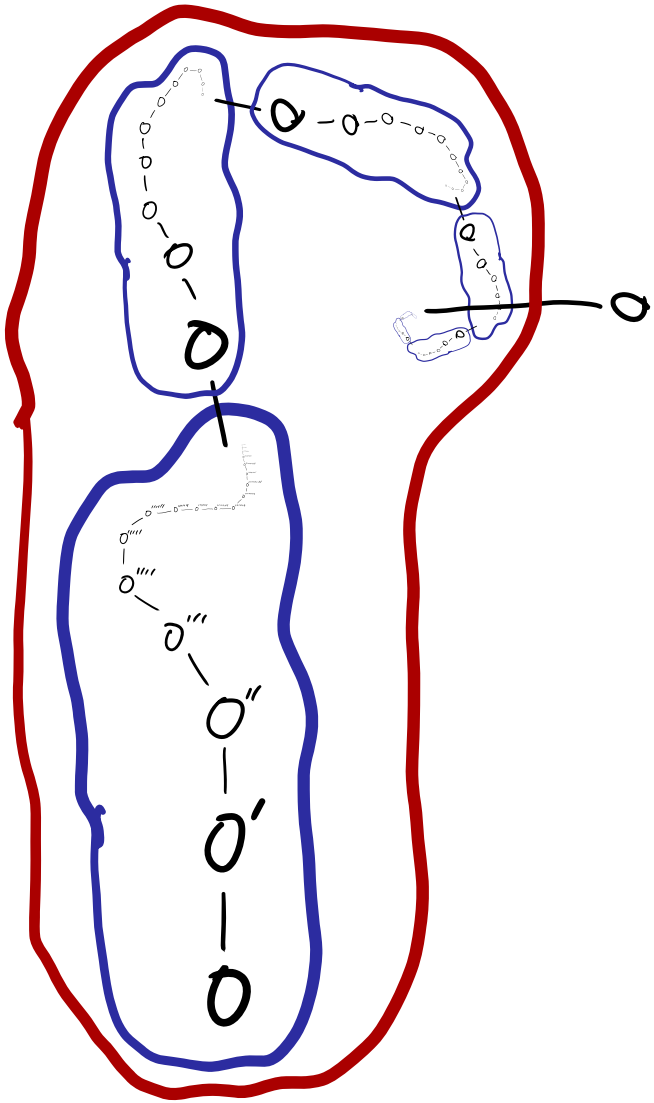
Even more degrees



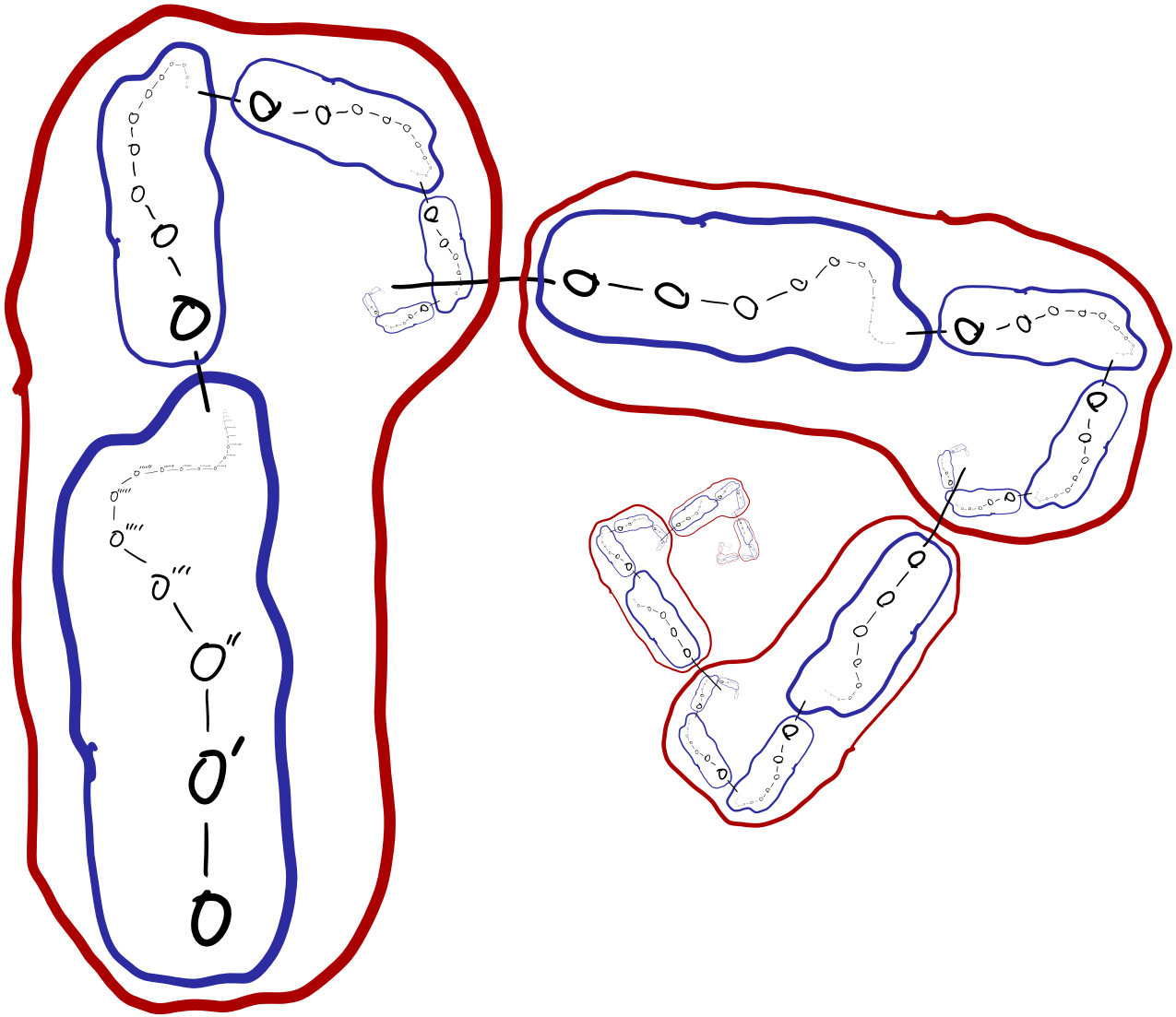
Even more degrees



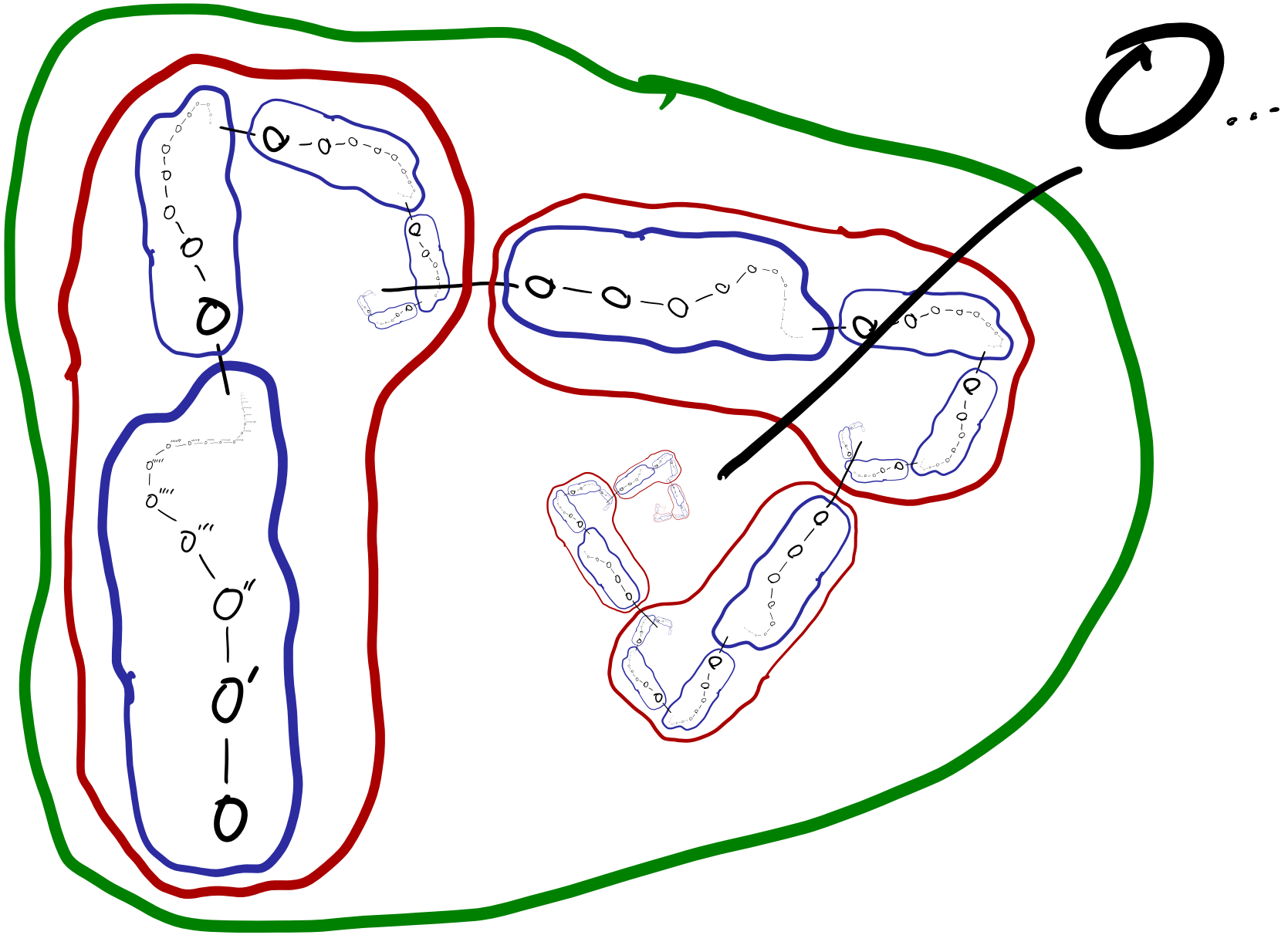
Even more degrees



Even more degrees



Even more degrees



Computability

So what was all that

So what was all that

- Unsolvable problems are crazypants
- There are a lot of them

Outro

Well I mean that's all I have, really

- Collapsing Hierarchies of Difficulties
- Recursively Infinite Impossibilities


THANKS
FOR
LISTENING