Tetris is Hard:
An Introduction to P vs NP

Based on "Tetris is Hard, Even to Approximate" in COCOON 2003 by

Erik D. Demaine (MIT)

Susan Hohenberger (JHU)

David Liben-Nowell (Carleton)

What's Your Problem?

A language is a set of strings:

$$\text{PRIMES} = \{2, 3, 5, 7, 11, 13, 17, 19, 23, \ldots\}$$

We are interested in the following kind of decision problem:

Given x, is $x \in \text{PRIMES}$?

How much harder does this question get as x gets larger?

The Tetris Problem

So what do we mean by "Tetris"?

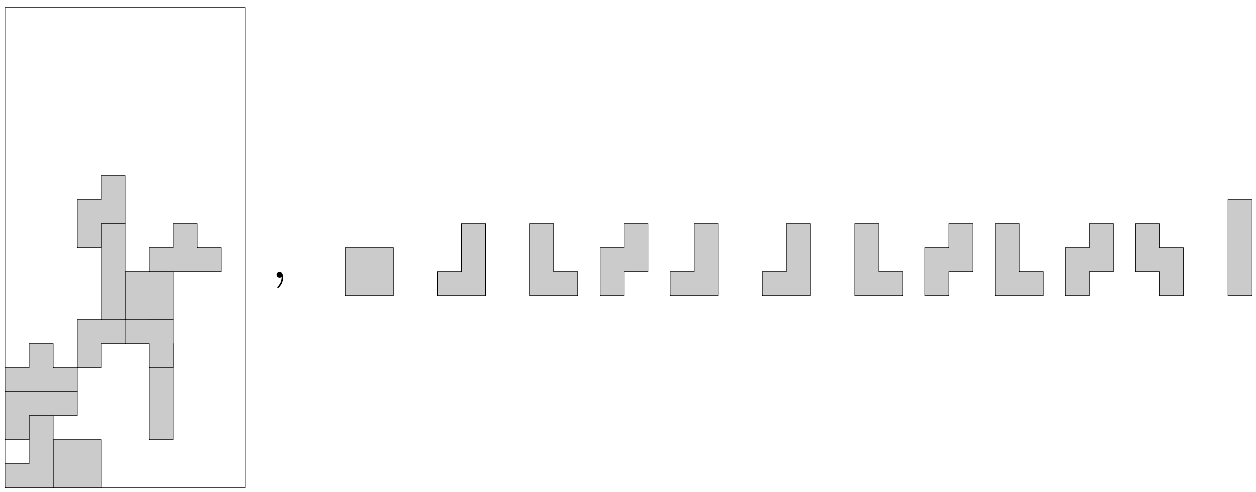
Offline Tetris: get to see entire piece sequence in advance.
"Advanced" level: the initial board is partially filled.

TETRIS $= \{\langle g, s \rangle :$
can play piece sequence s to clear entire gameboard $g\}$

$\langle g, s \rangle \in$ TETRIS: given a gameboard and a piece sequence, can we clear the entire board using the given sequence?

Instances of Decision Problems

Is 357667 $\in$ PRIMES?

Is $\langle$  $\rangle \in$ TETRIS?

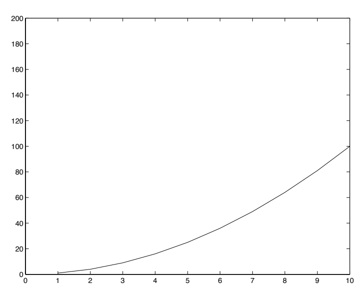
How much harder does this get as the gameboard gets bigger?

- Find an algorithm $\mathcal{A}$ that decides if $\langle g, s \rangle \in$ TETRIS.
- Measure resources consumed by $\mathcal{A}$.

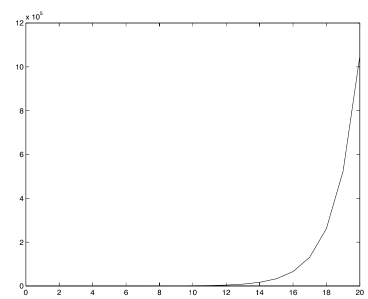
Measuring Resource Usage

How do we measure the resources consumed by $\mathcal{A}$?

- Be pessimistic.
  - Always consider the *worst* instance of a given size.
- Resources: time, space.
  - Run $\mathcal{A}$ on all instances of size n on some computer.
  - Graph n versus the amount of each resource consumed by $\mathcal{A}$ on the most consuming instance of size n.

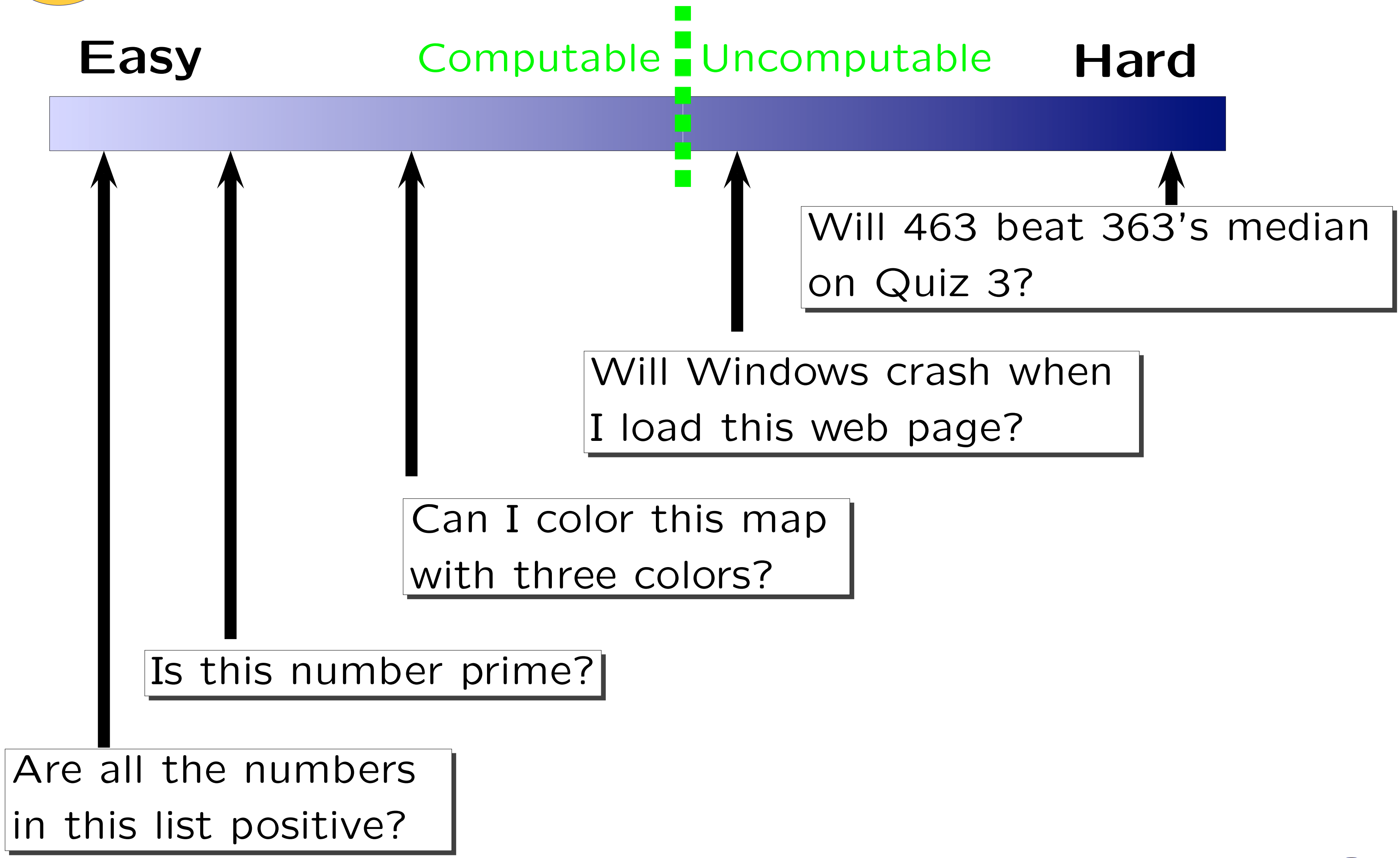


Polynomial



Exponential

A Computer Theorist's Worldview



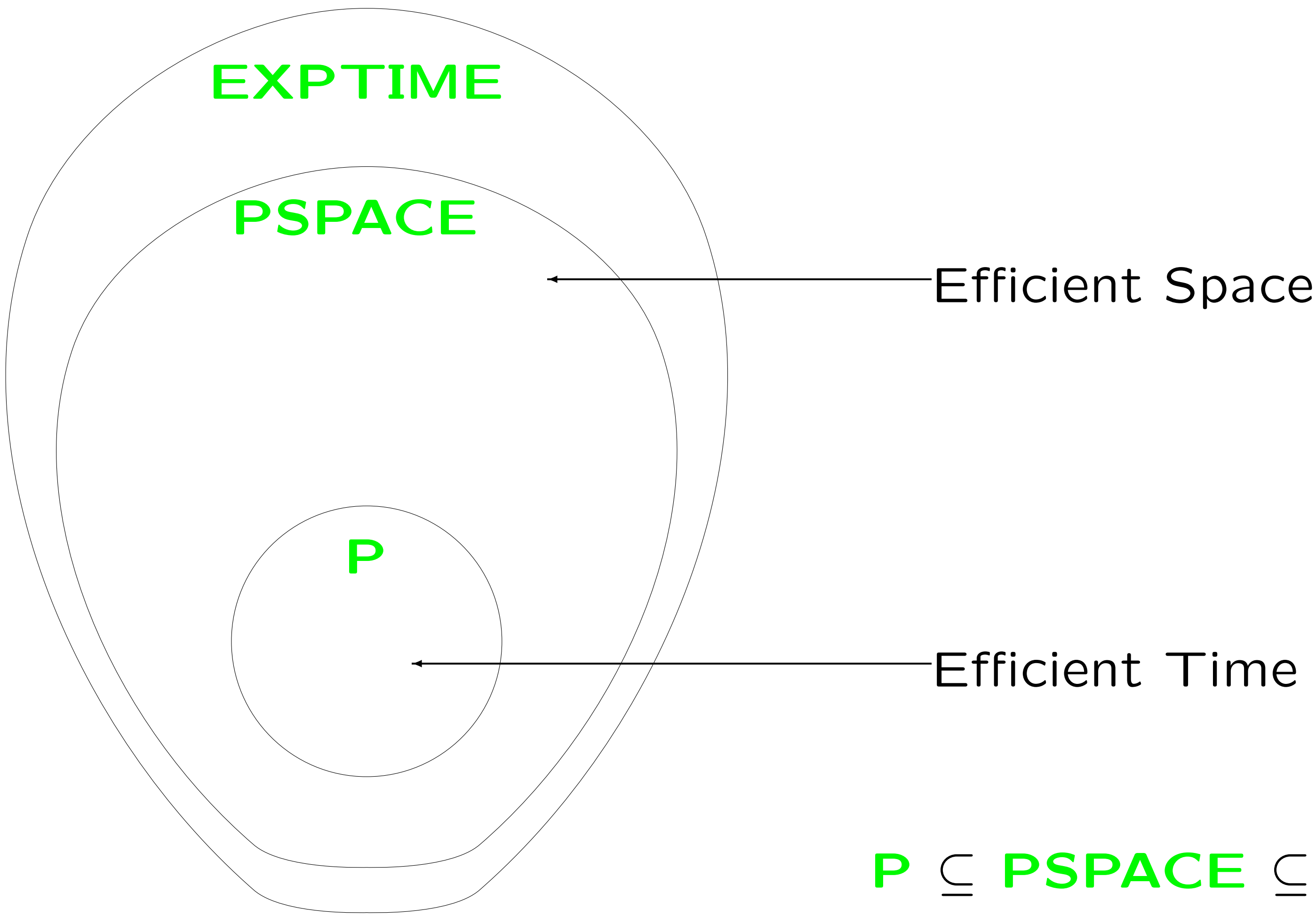
Complexity Classes

A complexity class is a set of languages.

A language L (PRIMES, TETRIS, ...) is a member of the following complexity classes if there is an algorithm $\mathcal{A}$ ...

P	... deciding all questions $x \in L$ in *polynomial time*. e.g., for every x, time taken by $\mathcal{A}(x)$ is $\leq 10\|x\|^3$. Games: JENGA
PSPACE	... deciding all questions $x \in L$ in *polynomial space*. Games: GO (in some countries), OTHELLO
EXPTIME	... deciding all questions $x \in L$ in *exponential time*. Games: CHESS, CHECKERS

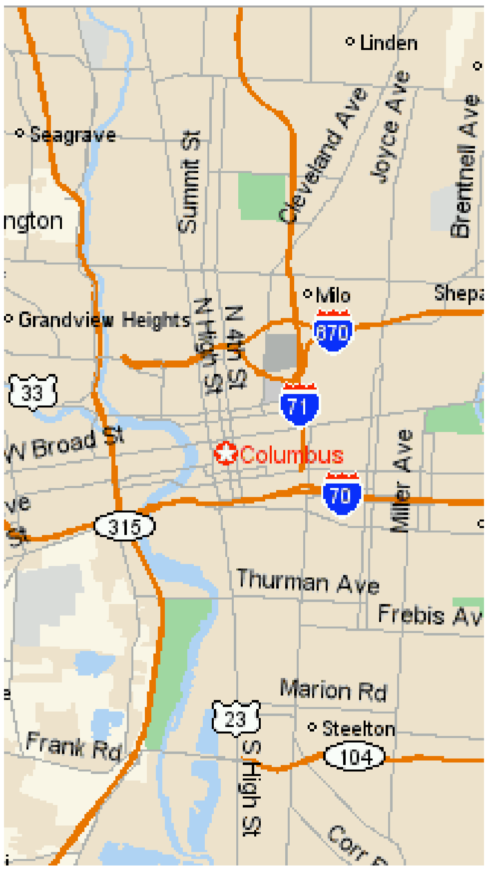
Complexity Hierarchy



$$\mathbf{P} \subseteq \mathbf{PSPACE} \subseteq \mathbf{EXPTIME}$$

$$\mathbf{P} \neq \mathbf{EXPTIME}$$

The Traveling Salesman Problem



You're selling fair trade coffee beans.

Start at Dean's Beans
World Headquarters in Dublin.

Visit (in any order):
Long's Bookstore, 1610 High St,
Cup of Joe, Easton Town Center,
446 Lane, Court House.

Drop off the leftovers in Dublin.

Can you do all deliveries in one hour?
What order should you use?

The "Naïve Solution" to TSP

The "naïve method" of solving TRAVELING-SALESMAN:

- List all possible orderings of the cities.
- Find the shortest ordering.
- Check if it requires under one hour.

So TRAVELING-SALESMAN is in **EXPTIME**.

16 cities: more orderings than the size of the US national debt.
63 cities: more orderings than atoms in the known universe.

Can we do better?

Can we solve TSP efficiently??

Nobody has been able to:

– give a faster solution to TRAVELING-SALESMAN.

– prove that TRAVELING-SALESMAN needs exponential time.

So TRAVELING-SALESMAN might be in **P**!

But notice:

For a particular ordering of the coffeeshops, it's easy to check
if the length of the trip is under one hour.
(Just add up the times.)

Guess and Check

Another way to measure the hardness of a language:

Suppose I claim that $x \in L$ (and give you an argument to try to convince you). How hard is it to check if my argument is right?

For example, for TETRIS: given $\langle g, s \rangle$

- I tell you all the moves m that I'll make in the game.
- You check if moves m clear the gameboard g using piece sequence s.

NP, continued

A language L is in **NP** if I can give you an argument A and then you can check that A is right in polynomial time.

NP: "Nondeterministic Polynomial Time"

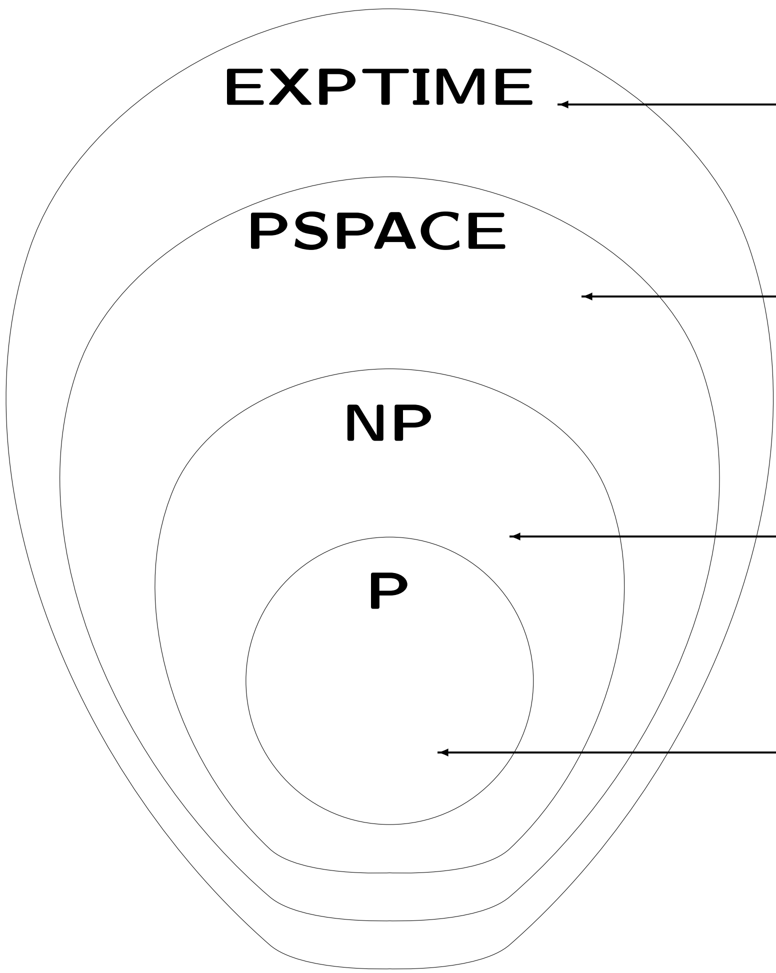
Nondeterminism: I give you an argument; you check it.

➡ For some games, like CHESS, the move sequence might be too long to check in polynomial time!

Summary: $L \in$ **P** can be decided quickly.

$L \in$ **NP** can be checked quickly.

Complexity Hierarchy



Chess, Checkers

Go, Othello, Sokoban,
Generalized Geography, Hex

Tetris, Mastermind,
Minesweeper, Sliding Blocks,
Traveling Salesman

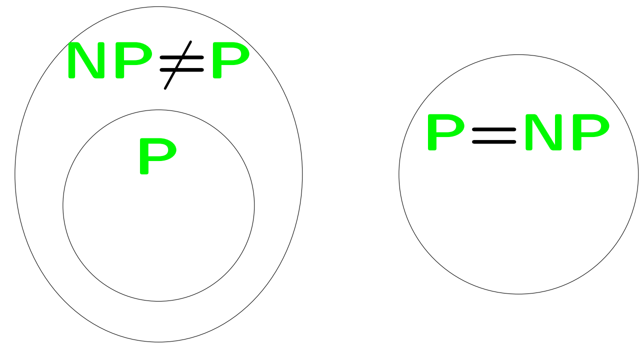
Jenga, I'm Thinking of a Number

$$\mathbf{P} \neq \mathbf{EXPTIME}$$

$$\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} \subseteq \mathbf{EXPTIME}$$

P vs. NP

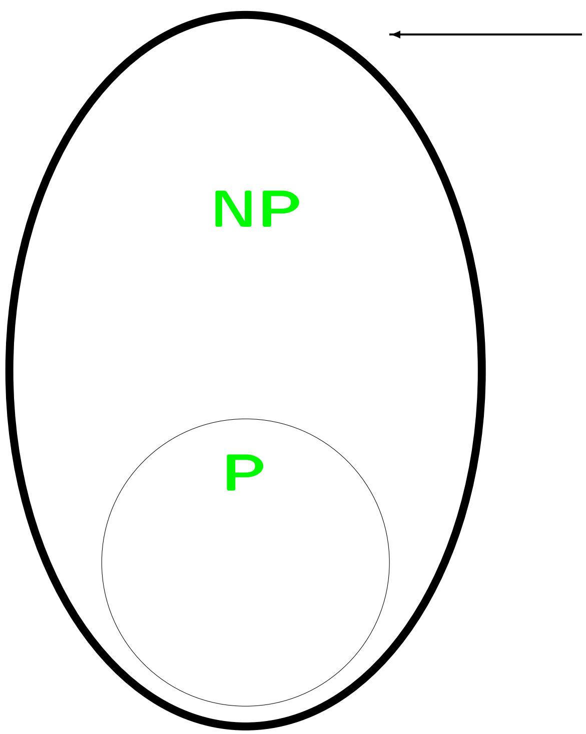
We do not know which is true!



- One of the greatest unsolved problems in theoretical computer science (and all of mathematics!).
- Clay Mathematics Institute offers $1 million for solution.
- Is it easier to verify a correct solution to a problem than it is to solve the problem from scratch?
- Huge impact on cryptography, airline scheduling, factory layout, UPS truck packing, drug design, etc., etc., etc., etc.

NP-completeness

First major step on **P** vs. **NP** problem [Cook and Levin, 1970s]:



An **NP**-complete problem:

1. is in **NP**
2. is as hard as the hardest problem in **NP**.

- If we can quickly decide one NP-complete problem, then we can quickly decide them all. (**P**=**NP**)
- If any problem is in **NP** and not **P**, then every **NP**-complete problem is in **NP** and not **P**. (**P**≠**NP**)

Technique #1: Divine Inspiration

Theorem [Cook/Levin]: SATISFIABILITY is **NP**-complete.

Technique #1: SAT is NP-complete

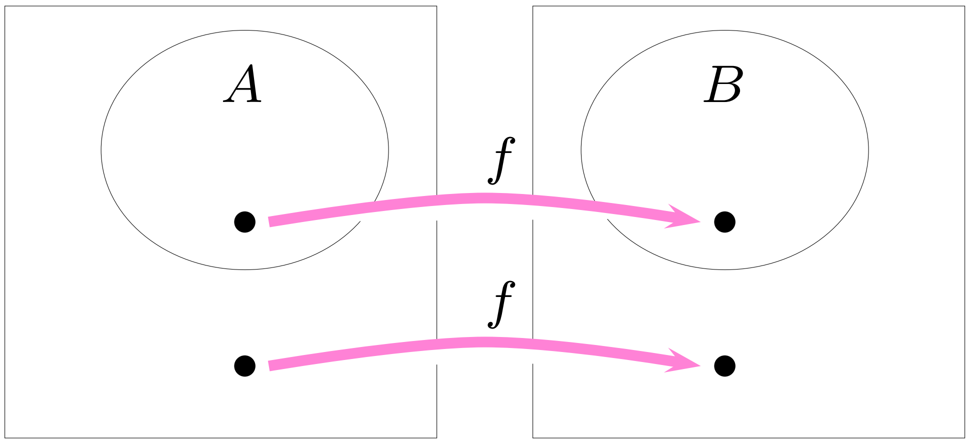
SATISFIABILITY = { formulas ϕ : there is a way of setting the variables of ϕ so that ϕ is true }

e.g., $x \wedge (\neg x \vee \neg y) \in$ SATISFIABILITY.
(set $x :=$ true and $y :=$ false)

Theorem [Cook/Levin]: SATISFIABILITY is NP-complete.

Technique #2: Reduction

A reduction f is a mapping from one problem to another.



$x \in A$
if and only if
$f(x) \in B$.

We're interested in *efficient* f.

- If there's a quick algorithm for B, then there's one for A.
- If there's no quick algorithm for A, then there's none for B.
  - "B is as hard as A"

NP-Completeness Summary

Theorem: SATISFIABILITY is **NP**-complete.

— Cook/Levin.

Theorem: If: (1) A is **NP**-complete,

(2) B is in **NP**, and

(3) there is an efficient reduction from A to B

then B is **NP**-complete. — Karp.

Enough already! I came for the Tetris ...

To show that TETRIS is **NP**-complete:

- Define efficient mapping m from instances of known **NP**-Complete problem L to TETRIS.
- Show that if $x \in L$, then $m(x) \in$ TETRIS.
- Show that if $x \notin L$, then $m(x) \notin$ TETRIS.

From 3-PARTITION to TETRIS

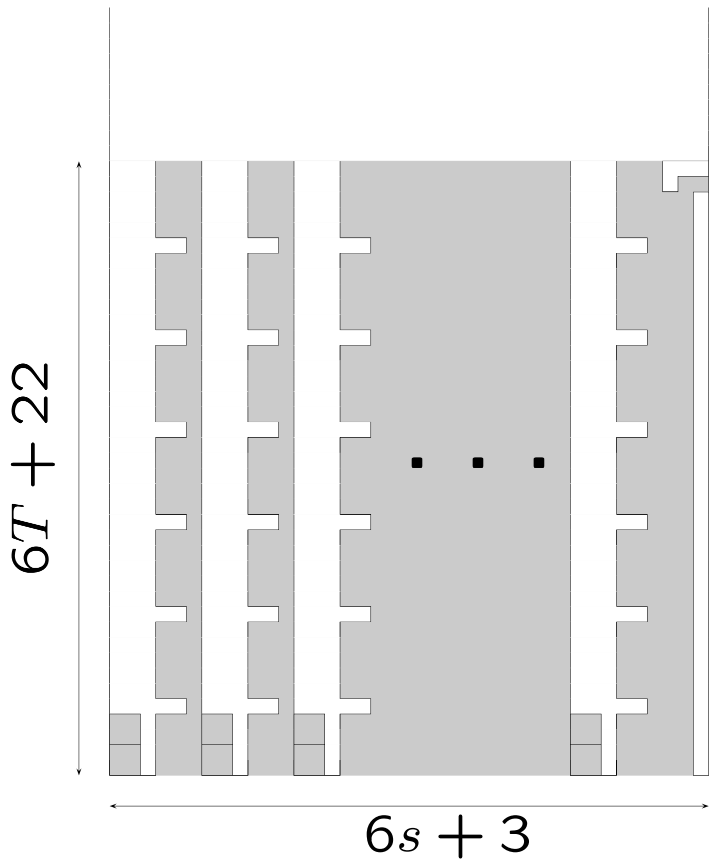
3-PARTITION = {sets of integers that can be separated into piles of three integers each, where each pile has the same sum}

> Theorem [Garey and Johnson]: 3-PARTITION is **NP**-complete.

Our result:
There is an efficient reduction from 3-PARTITION to TETRIS.

Our Reduction ...

Given integers $a_1, \ldots, a_{3s}$ and the target sum T for each pile:



a_1 initiator:

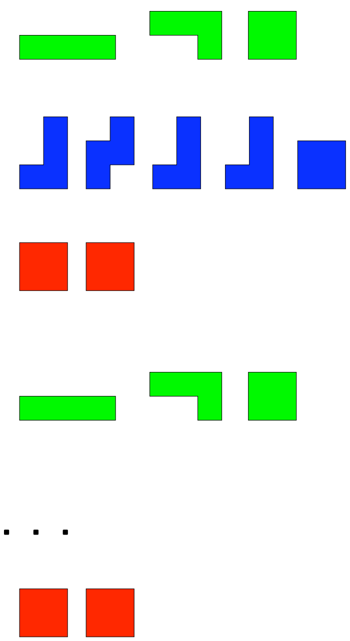
filler (a_1 times):

terminator:

a_2 initiator:

...

a_{3s} terminator:



Finally: $\times\ s$, , and $\times(\frac{3T}{2} + 5)$.

Reminder: NP-Completeness

To show that Tetris is NP-complete:

√ Define efficient mapping m from instances of 3-PARTITION to TETRIS.

- Show that if $x \in$ 3-PARTITION, then $m(x) \in$ TETRIS.

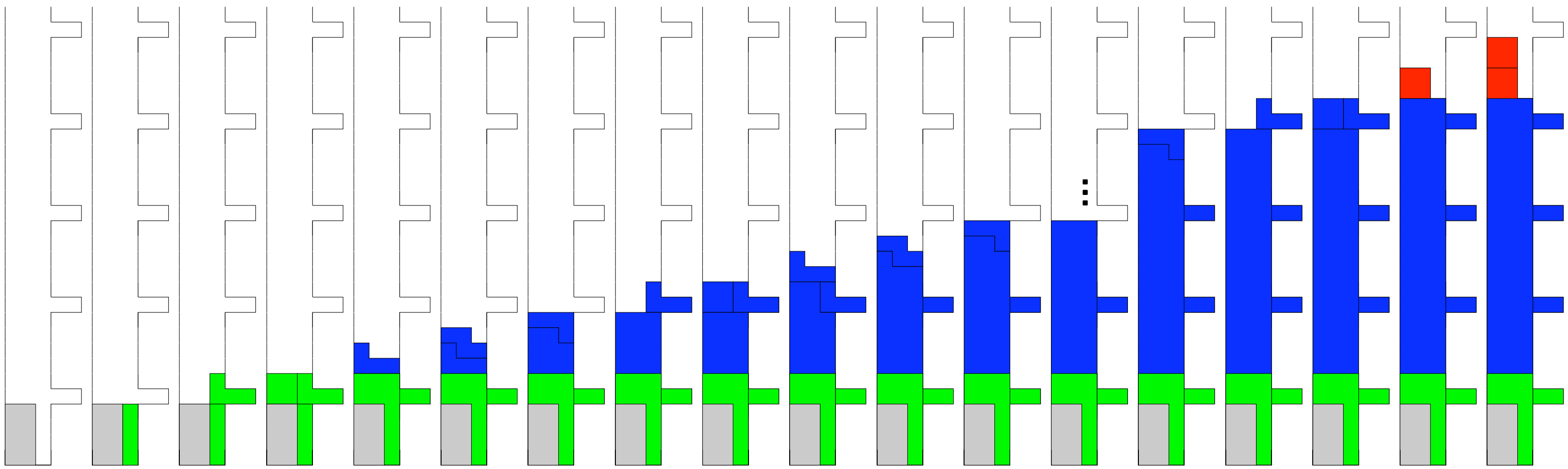
- Show that if $x \notin$ 3-PARTITION, then $m(x) \notin$ TETRIS.

For $x \in L$, we need $m(x) \in$ TETRIS

"Yes" means "yes."

If we have $x \in$ 3-PARTITION, then need $m(x) \in$ TETRIS.

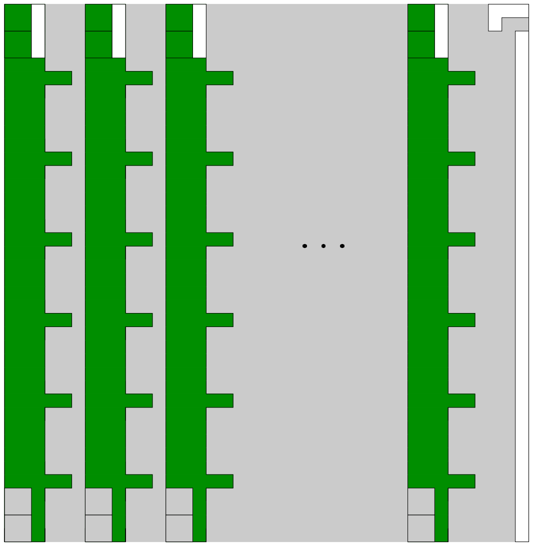
- Pile the integers according to their 3-PARTITION piles.



The "yes" case, continued

Piles each have the same sum, so they have the same height:

- Each bucket is filled exactly to the top.



- Remaining pieces will exactly clear entire gameboard.

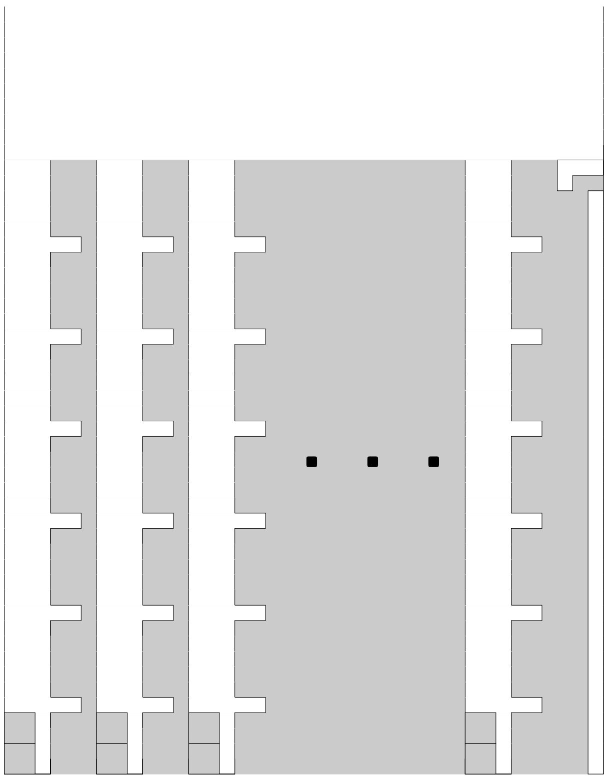
Reminder: NP-Completeness

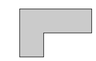
To show that TETRIS is **NP**-complete:

- √ Define efficient mapping m from instances of 3-PARTITION to TETRIS.
- √ Show that if $x \in$ 3-PARTITION, then $m(x) \in$ TETRIS.
- ➡ Show that if $x \notin$ 3-PARTITION, then $m(x) \notin$ TETRIS.

Why this reduction?

We want to make sure that “no” means “no.”



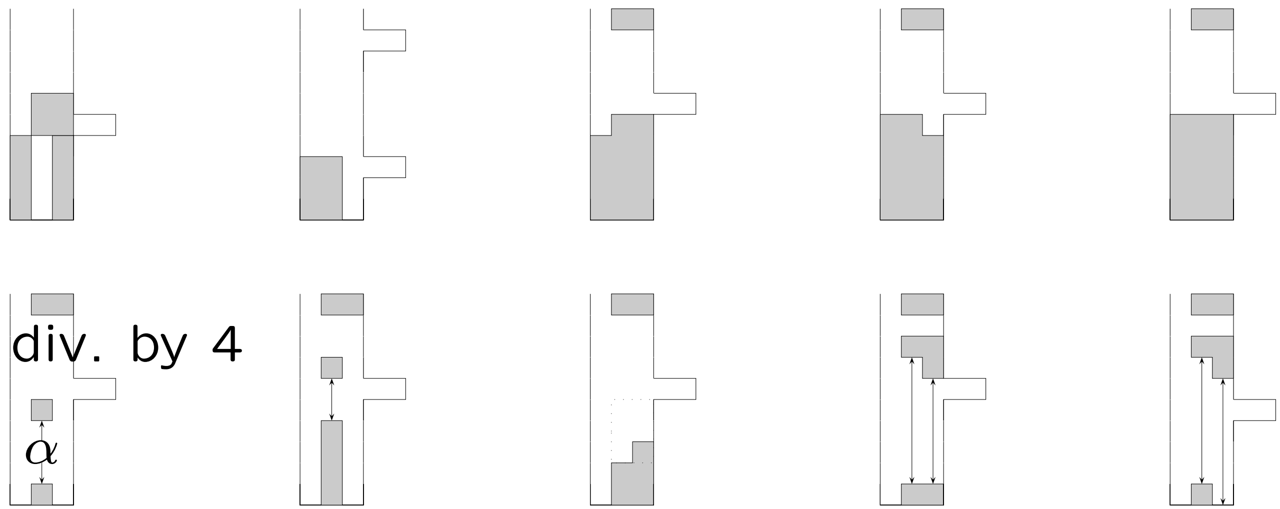
- can’t clear any rows until the  .
- have to completely fill each bucket or it’s all over.
- notches make it easy to get stuck.

For $x \notin L$, we need $m(x) \notin$ TETRIS

"No" means "no."

Step #1: if you don't (or can't) make the moves from two slides ago, you get into one of the following configurations.

Step #2: if you get into one of these, you're hosed.



α not div. by 4

Reminder: NP-Completeness

To show that Tetris is NP-complete:

√ Define efficient mapping m from instances
of 3-PARTITION to TETRIS.

√ Show that if $x \in$ 3-PARTITION, then $m(x) \in$ TETRIS.

√ Show that if $x \notin$ 3-PARTITION, then $m(x) \notin$ TETRIS.

Theorem: TETRIS is NP-complete.

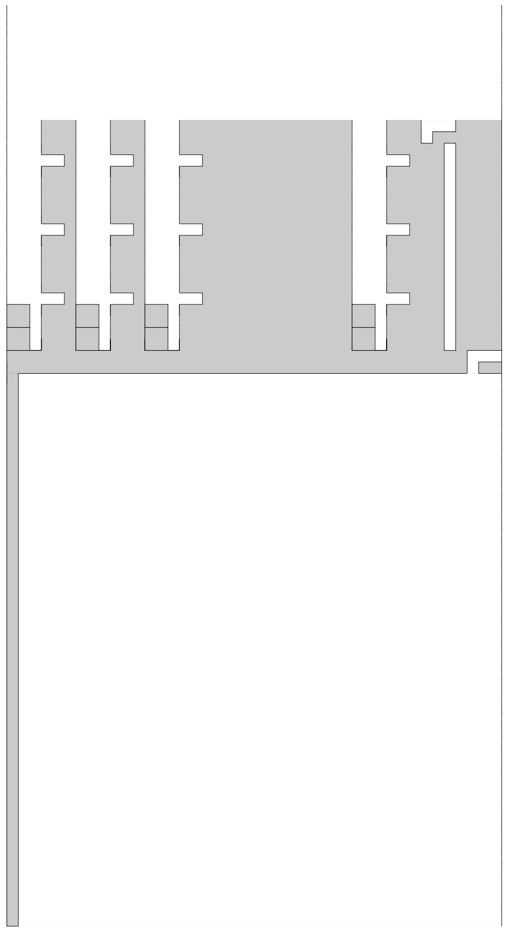
Statement of Results

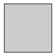
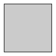
Our results actually apply for some other Tetris questions, too:

- can we clear at least k rows?
- can we place at least p pieces without losing?
- can we place all pieces without filling a square at height h?
- can we achieve at least t tetrises?

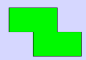
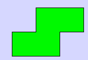
All of these are **NP**-complete.

Other Objectives and Inapproximability



- Can strengthen our results further: add a boatload of  pieces to our sequence.
- Can get into (and clear) lower reservoir (using extra  pieces) only if can clear the top using original.
- Choosing a large reservoir yields inapproximability results.

Other Work on Tetris

Theorem: A (sufficiently long) alternating sequence of  s and  s will cause a loss.

[Brzustowski/Burgiel]

Implies that probability of a real Tetris game lasting infinitely long is zero (regardless of player's skill).

On *the* Tetris gameboard that you know and love.

(Our complexity results are as the gameboard grows.)

Conclusion and Open Questions

- Many interesting problems are **NP**-complete.
  (Traveling Salesman, Minesweeper, Tetris, Satisfiability, ...)
  - So what? Does **P** = **NP**?
  - Is it easier to verify that a solution is correct than it is to come up with it from scratch?
- Games are interesting because they're hard (?)
- Easy way to make a million bucks:
  give an efficient algorithm for TETRIS!

Recap: Proving Membership in NP

SATISFIABILITY = $\{$ formula ϕ : there is a way of setting the variables of ϕ so that ϕ is true $\}$

Certificate: assignment to variables

Verifier: plugs in assignment and tests that ϕ is true

Recap: Proving Membership in NP

SATISFIABILITY = { formula ϕ : there is a way of setting the variables of ϕ so that ϕ is true }

Certificate: assignment a to variables
Verifier: tests that ϕ is true with assignment a

CLIQUE = { graph G and integer k : there is a clique of size k in G}

Certificate: nodes in the clique C
Verifier: tests that each node in C is connected to every other node in C

Recap: Optimization vs Decision

SATISFIABILITY = { formula ϕ : there is a way of setting the variables of ϕ so that ϕ is true }

Suppose P = NP.

How can you find a satisfying assignment in polynomial time?

Recap: Optimization vs Decision

CLIQUE $= \{$ graph G and integer k : there is a clique of size k in $G\}$

Suppose P = NP.

How can you find a k-clique in polynomial time?