6.006	Lecture 21	Nov. 29, 2018

TODAY: Computational Complexity
- P, EXP, R - pseudopoly.
- most problems are uncomputable
- NP
- hardness & completeness
- reductions

Recall: $O(nt)$-time algorithm for Subset Sum
(given n pos. integers $A = \{a_0, a_1, \ldots, a_{n-1}\}$ & target sum t, does any subset $S \subseteq A$ sum to t)
- is this fast?
- cf. obvious $O(2^n n)$-time brute force alg.

Polynomial time = polynomial in input size
- here $\Theta(n)$ if number t fits in a word
  $O(n \lg t)$ in general (can assume $a_i \leq t$)
- t is exponential in $\lg t$ (not polynomial)

Pseudopolynomial time = polynomial in the problem size AND the numbers in input
(here: t & a_i's)
- $\Theta(nt)$ is pseudopolynomial

Polynomial: GOOD; Exponential: BAD; Pseudopoly: so so

$\underline{P}$ = {problems solvable in polynomial time}

$n^{O(1)}$ time where n = INPUT SIZE

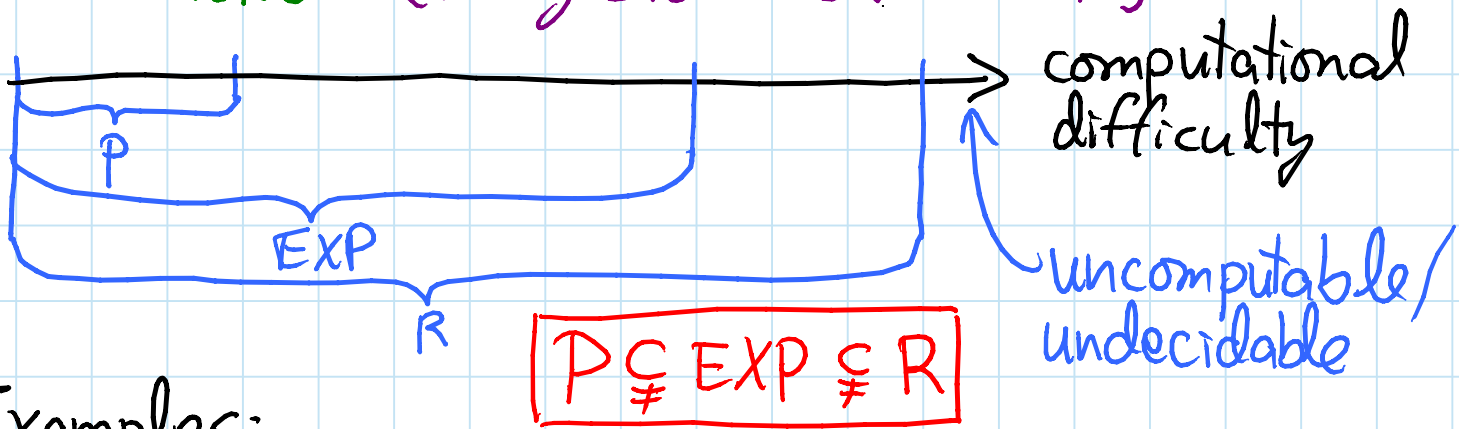
(what this class is all about)

$\underline{EXP}$ = {problems solvable in exponential time}

$2^{n^{O(1)}}$

$\underline{R}$ = {problems solvable in finite time}

"recursive" [Turing 1936; Church 1941]



$P \subsetneq EXP \subsetneq R$

Examples:

- negative-weight cycle detection $\in P$
- $n \times n$ Chess $\in EXP$ but $\notin P$
  - who wins from given board config.?
- Tetris $\in EXP$ but don't know whether $\in P$
  - survive given pieces from given board

Halting problem: given a computer program, does it ever halt (stop)?

- uncomputable ($\notin R$): no algorithm solves it (correctly in finite time on all inputs)
- decision problem: answer is YES or NO

Most decision problems are uncomputable:

- program $\approx$ binary string $\approx$ nonneg. integer $\in \mathbb{N}$
- decision problem = a function from binary strings ($\approx$ nonneg. integers) to {YES, NO} ($\approx \{0,1\}$)

$\approx$ infinite sequence of bits $\approx$ real number $\in \mathbb{R}$

- $|\mathbb{N}| \ll |\mathbb{R}|$: no assignment of unique nonneg. integers to real numbers ($\mathbb{R}$ uncountable) (proof by "diagonalization" argument [6.042])

$\Rightarrow$ not nearly enough programs for all problems

- each program solves only one problem

$\Rightarrow$ almost all problems cannot be solved

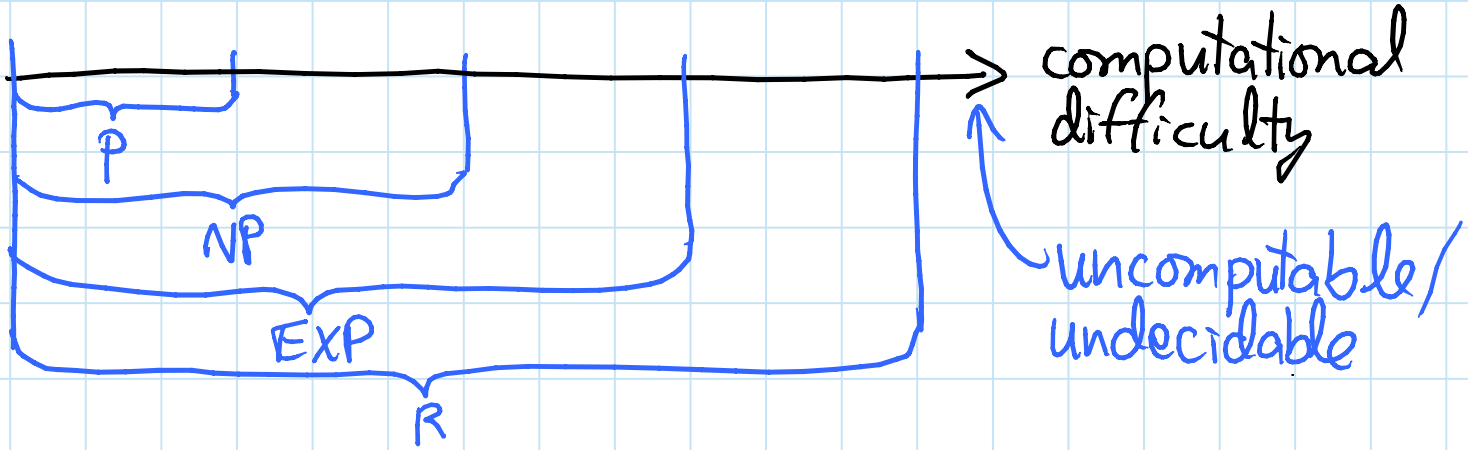
NP = {decision problems solvable in poly. time via a "lucky" algorithm}

↳ can make lucky guesses, always "right", without trying all options

- nondeterministic model: algorithm makes guesses & then says YES or NO
- guesses guaranteed to lead to YES outcome if possible (NO otherwise)

= {decision problems with solutions that can be "checked" in polynomial time}

- when answer = YES, can "prove" it with a polynomial-length proof/certificate checkable in poly. time



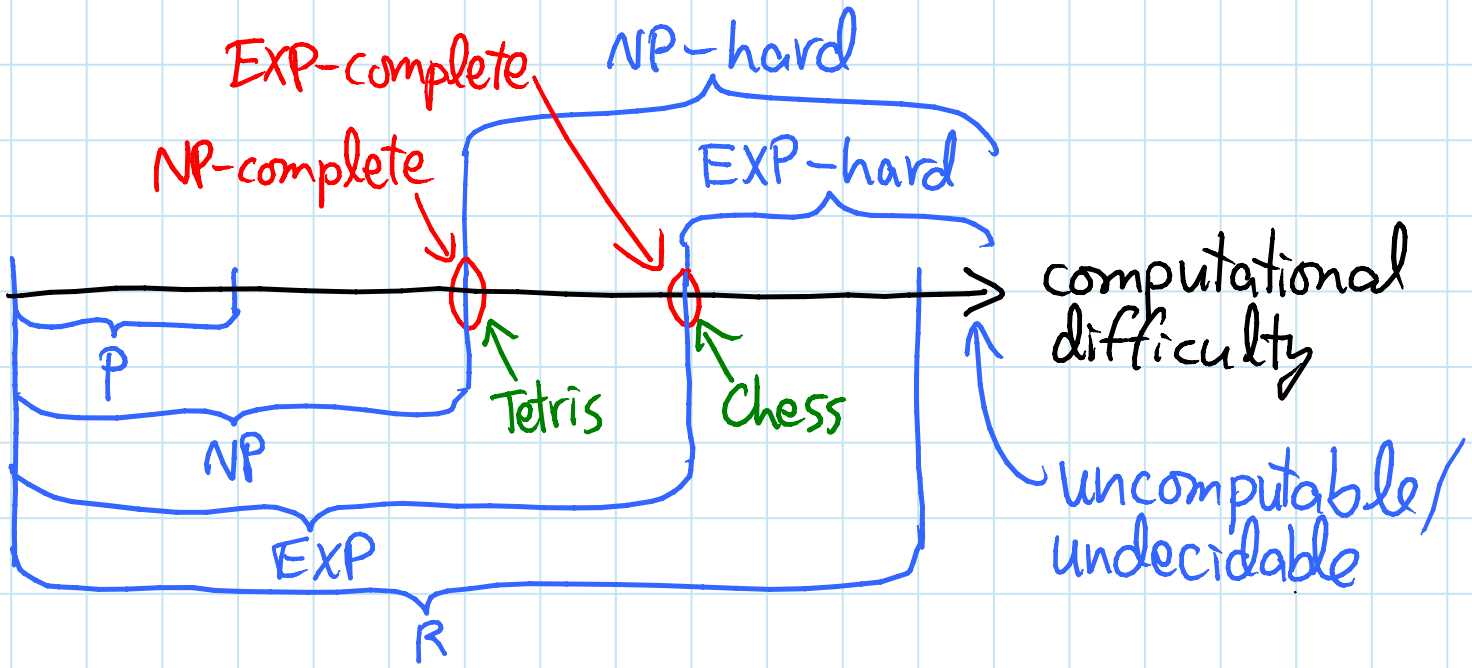
Example: Tetris ∈ NP

- nondeterministic alg: - guess each move
  - did I survive?
- proof of YES: list what moves to make (rules of Tetris are easy)

P≠NP: big conjecture (worth $1,000,000)
≈ can't engineer luck
≈ generating (proofs of) solutions is sometimes harder than checking them

Claim: if P≠NP, then Tetris ∈ NP∖P
[Breukelaar, Demaine, Hohenberger, Hoogeboom, Kosters, Liben-Nowell 2004]

Why? Tetris is NP-hard
= "as hard as" every problem ∈ NP
– in fact NP-complete = NP ∩ NP-hard

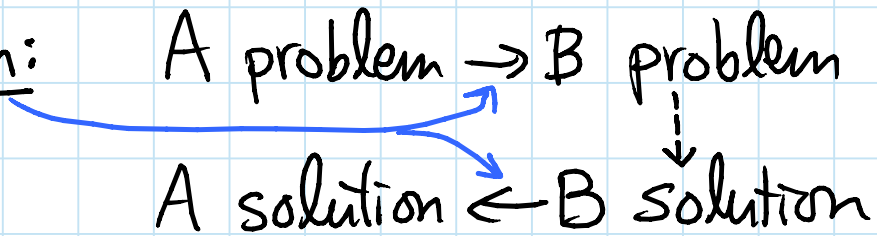


Similarly: Chess is EXP-complete
= EXP ∩ EXP-hard
as hard as every problem in EXP
⇒ if NP≠EXP, then Chess ∉ EXP∖NP
also open, but less famous/"important"

Reductions: convert your problem into a problem you already know how to solve (instead of solving from scratch)

- most common algorithm design technique
- unweighted shortest path → weighted
  (set weights = 1)
- integer-weighted shortest path → unweighted
  (subdivide edges)
- longest path → shortest path
  (negate weights)
- ≤k-dangerous path → graph reachability
  (k copies of the graph) [PS6-5]

A→B reduction:



⇒ A is at least as easy as B
i.e. B is at least as hard as A

- NP-complete problems are all interreducible using polynomial-time reductions (same difficulty)

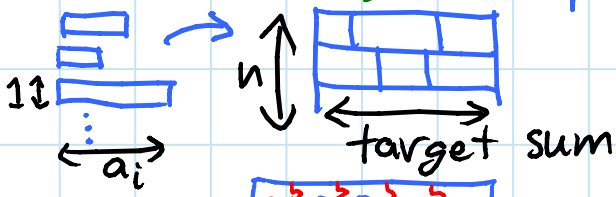
⇒ can use reductions to prove NP-hardness
e.g. 3-Partition → Tetris

Examples of NP-complete problems:

- Subset Sum (pseudopoly, not poly) "weakly NP-complete"
- 3-Partition: given n integers, can you divide them into triples of equal sum? "strongly NP-complete" (not even pseudopoly. possible if P≠NP)

REDUCTIONS:

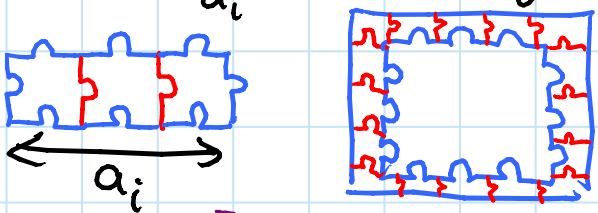
→ rectangle packing



→ jigsaw puzzles

ambiguous

unique

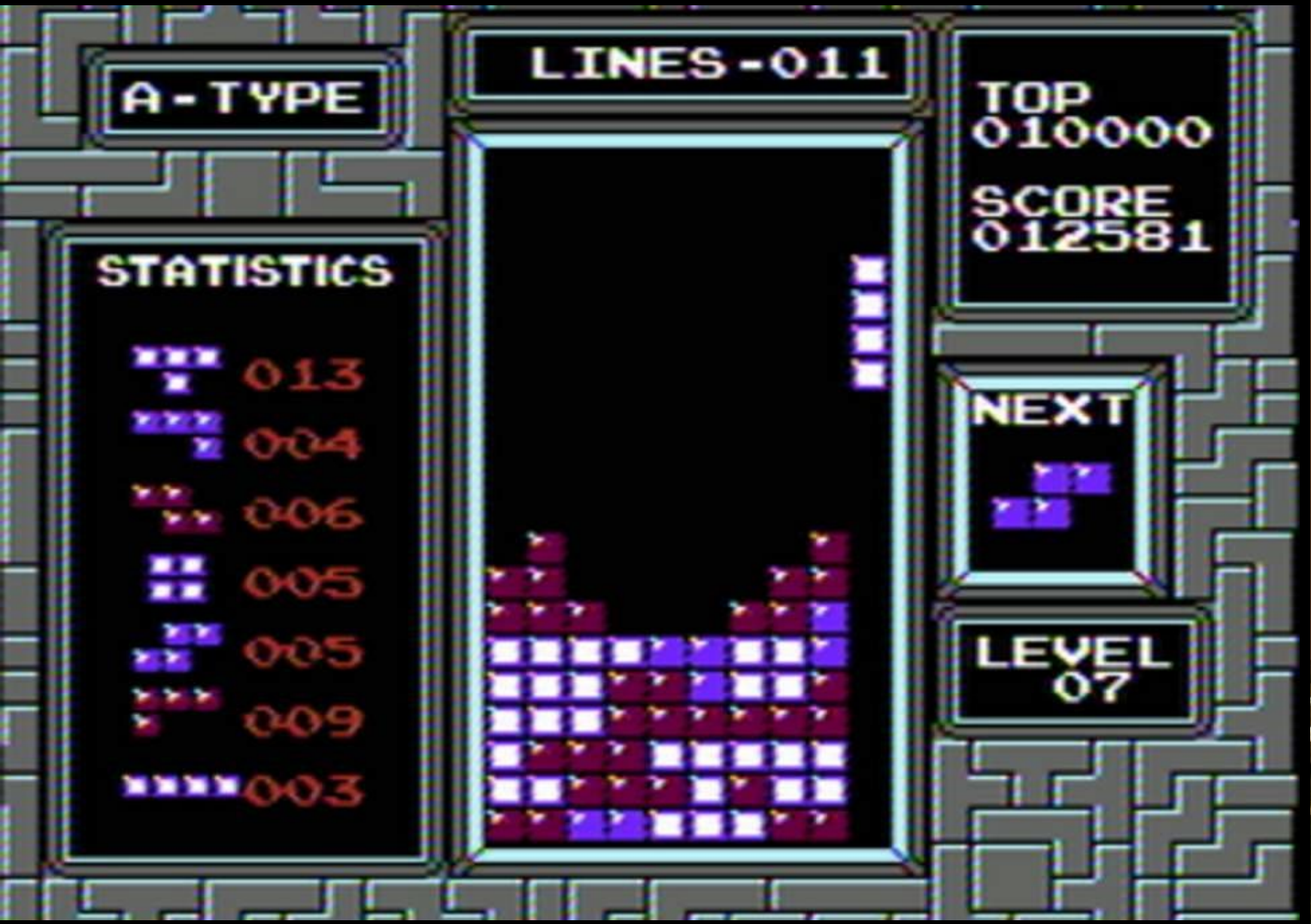


[Demaine & Demaine 2007]

- Minesweeper, Sudoku, & most puzzles
- Super Mario Bros., Legend of Zelda, Pokémon, ...
  [Aloupis, Demaine, Guo 2012]
- longest simple path in a graph
  → deciding whether you've already won in Settler's of Catan is NP-complete!
- Traveling Salesman Problem: shortest path that visits all vertices of a given graph
  - decision version: is min weight ≤ x?
- longest common subsequence of k strings
- SAT: given a Boolean formula (and, or, not), is it ever true? x and not x → NO
- shortest paths amidst obstacles in 3D
- 3-coloring a given graph
- find largest clique in a given graph

“Horses Running Endlessly”
by Gabriel Orozco, 1995

http://www.museoamparo.com/collections/piece/2534/horses-running-endlessly

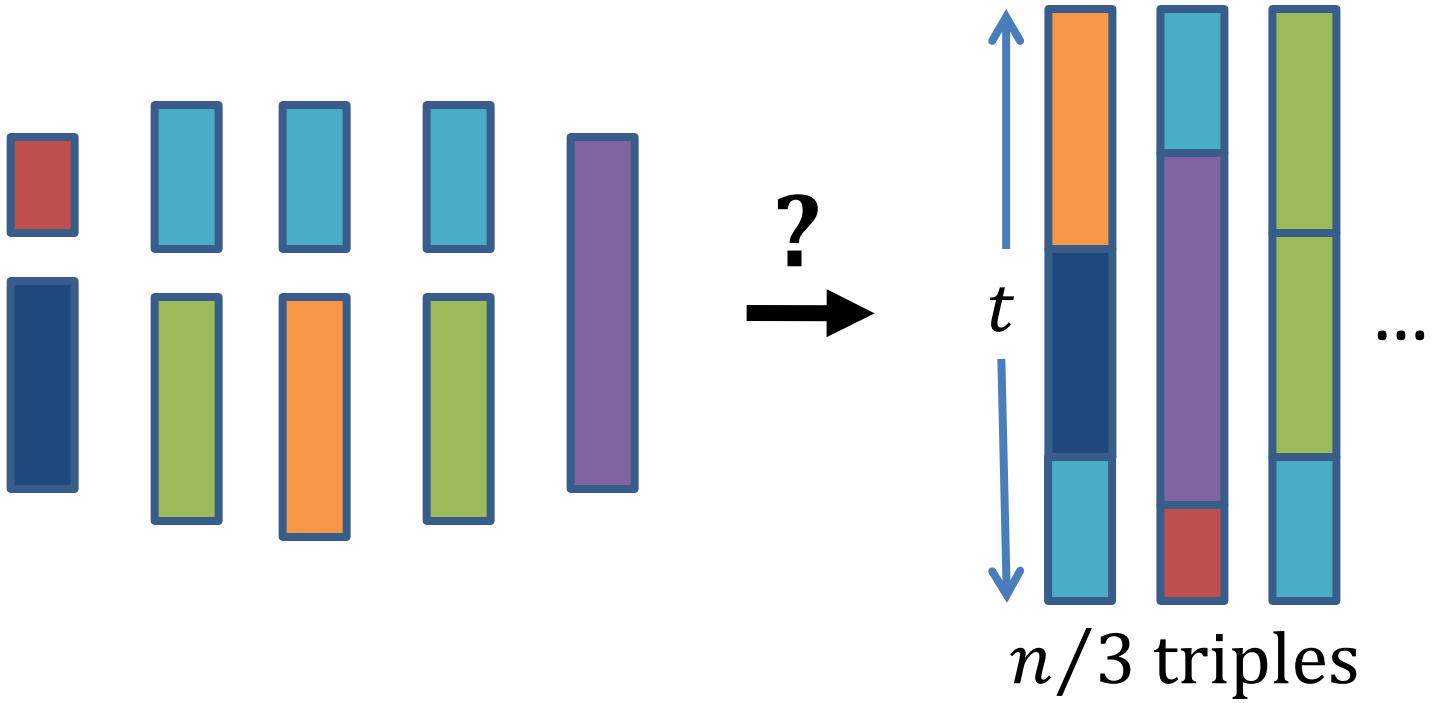

A-TYPE
LINES-011
TOP
010000
SCORE
012581
STATISTICS
013
004
006
005
005
009
003
NEXT
LEVEL
07


MIT Green Bldg.
April 2012

Classic NP-complete Problem: *3-Partition*

[Garey & Johnson 1975]

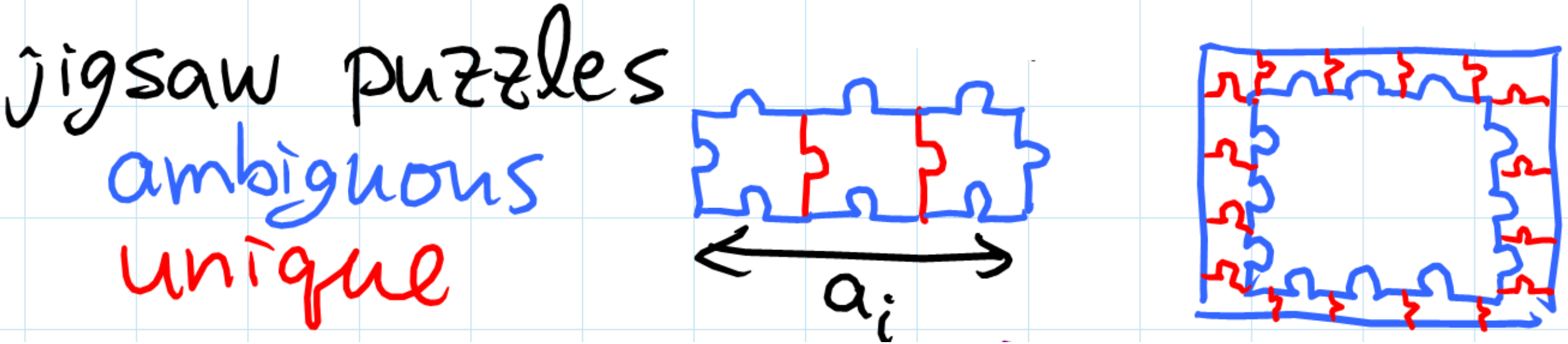
- Given n integers $a_1, a_2, \ldots, a_n$ between 0 and n, can you partition them into $n/3$ triples with the same sum? $\left[t = \frac{(\sum_i a_i)}{n/3}\right]$



Reduction from 3-Partition to Jigsaw Puzzles

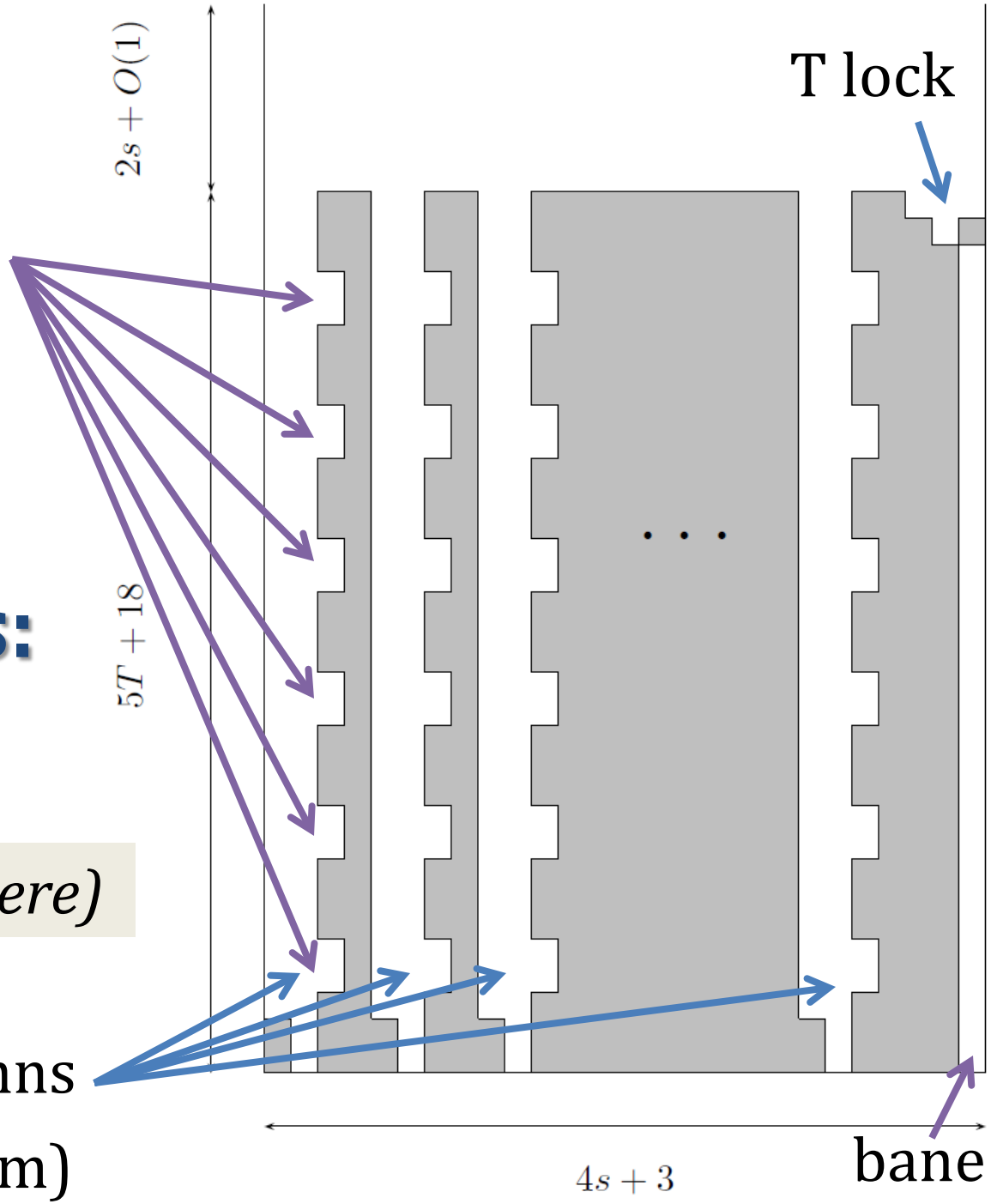
[Demaine & Demaine 2007]

- Given n integers $a_1, a_2, \ldots, a_n$ between 0 and n, can you partition them into $n/3$ triples with the same sum? $\left[t = \frac{(\sum_i a_i)}{n/3}\right]$



Reduction from 3-Partition to Tetris: Initial Board

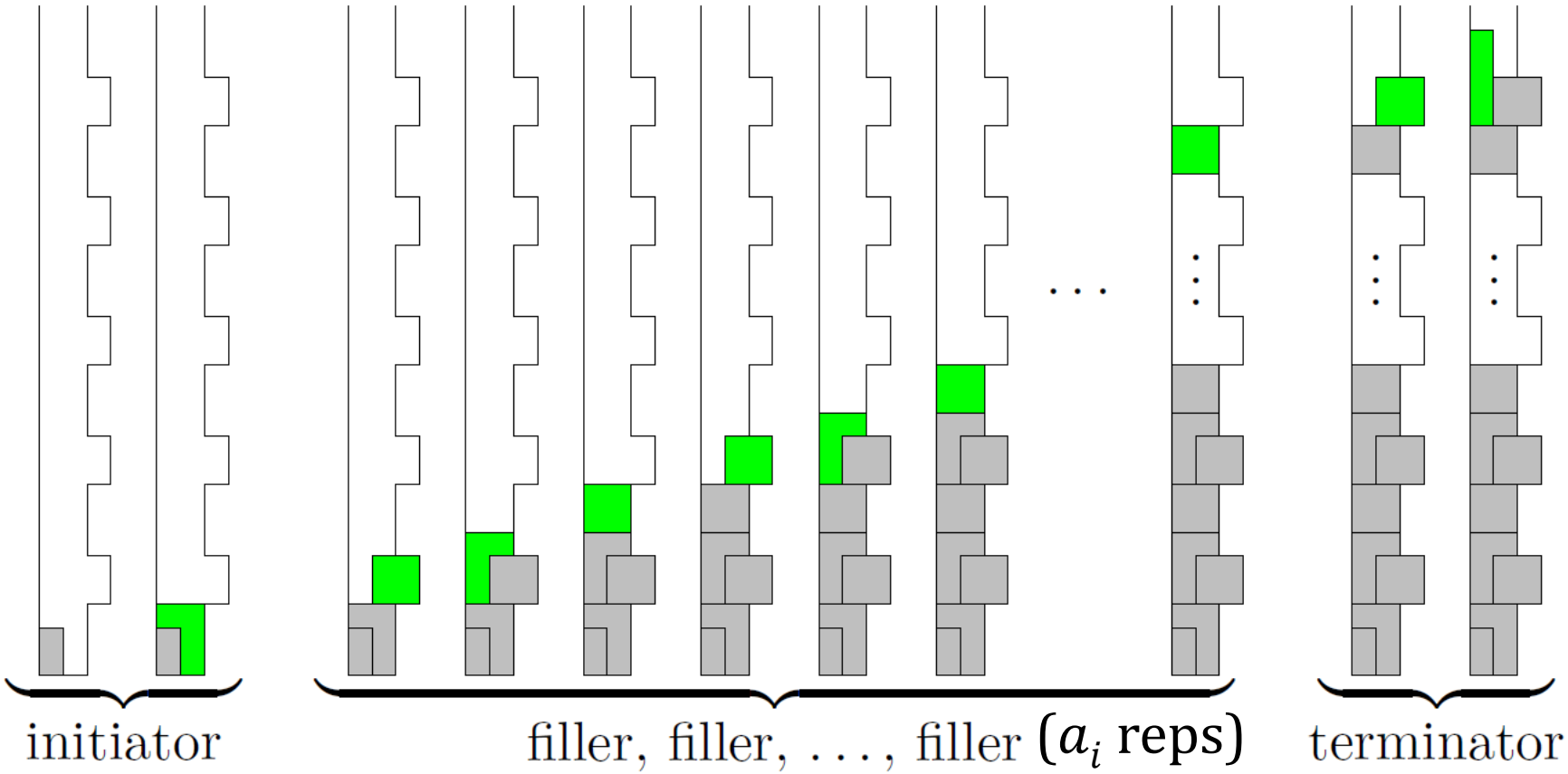
(it is possible to actually get here)



Reduction from 3-Partition to Tetris: Piece Sequence

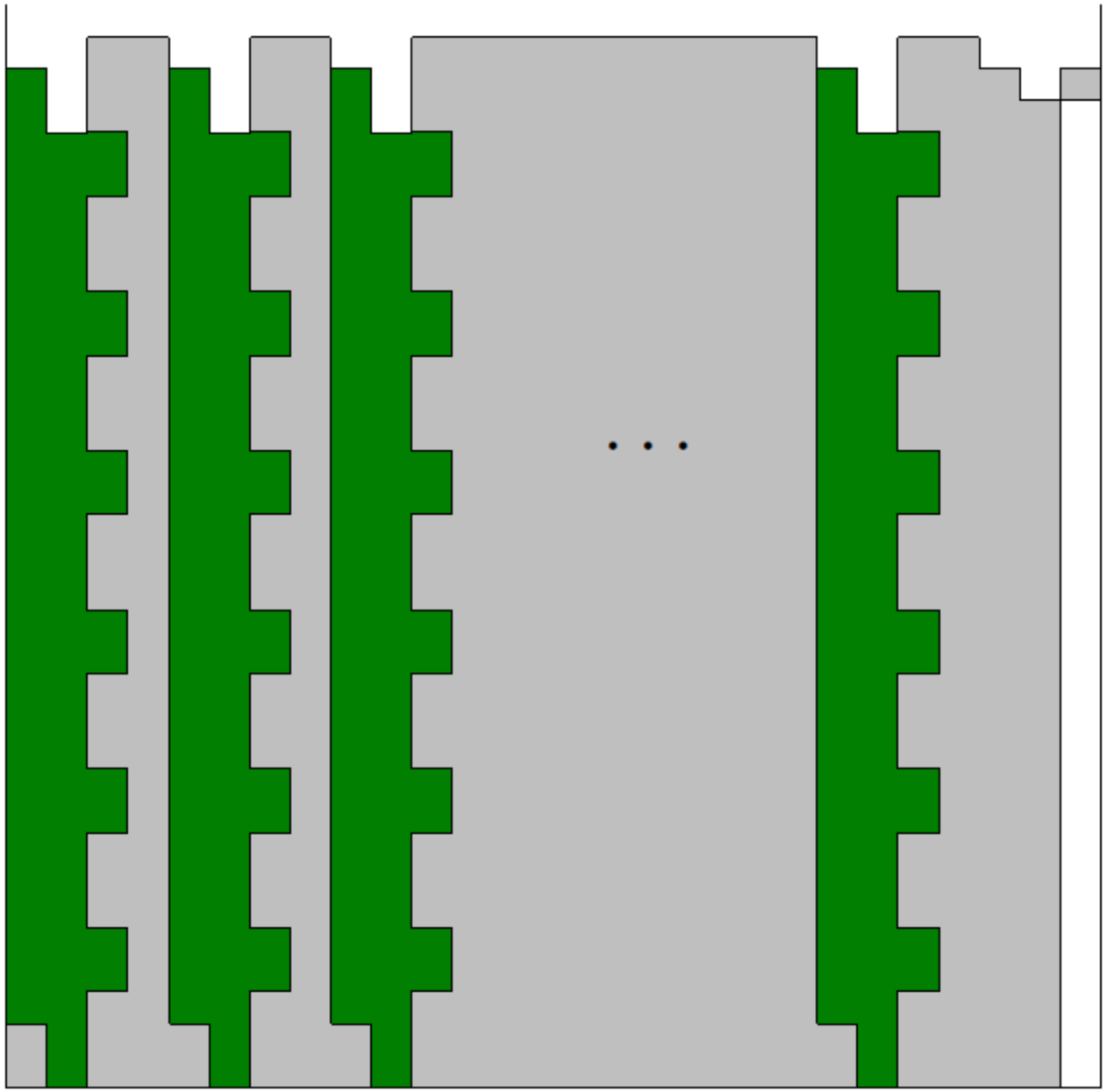
[Breukelaar, Demaine, Hohenberger, Hoogeboom, Kosters, Liben-Nowell 2003]

- For each input a_i:



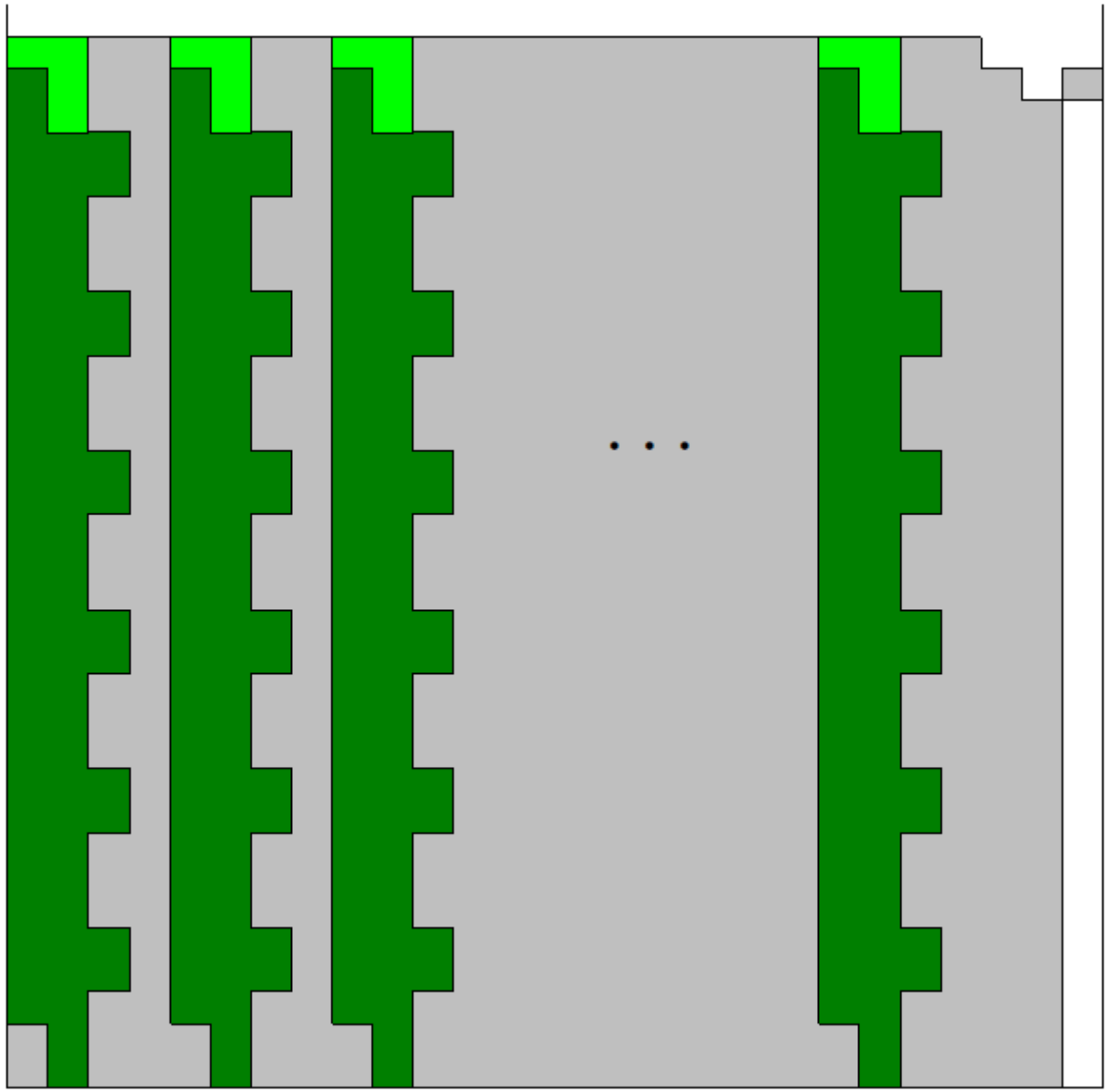
Finale Pieces

[Breukelaar, Demaine, Hohenberger, Hoogeboom, Kosters, Liben-Nowell 2003]



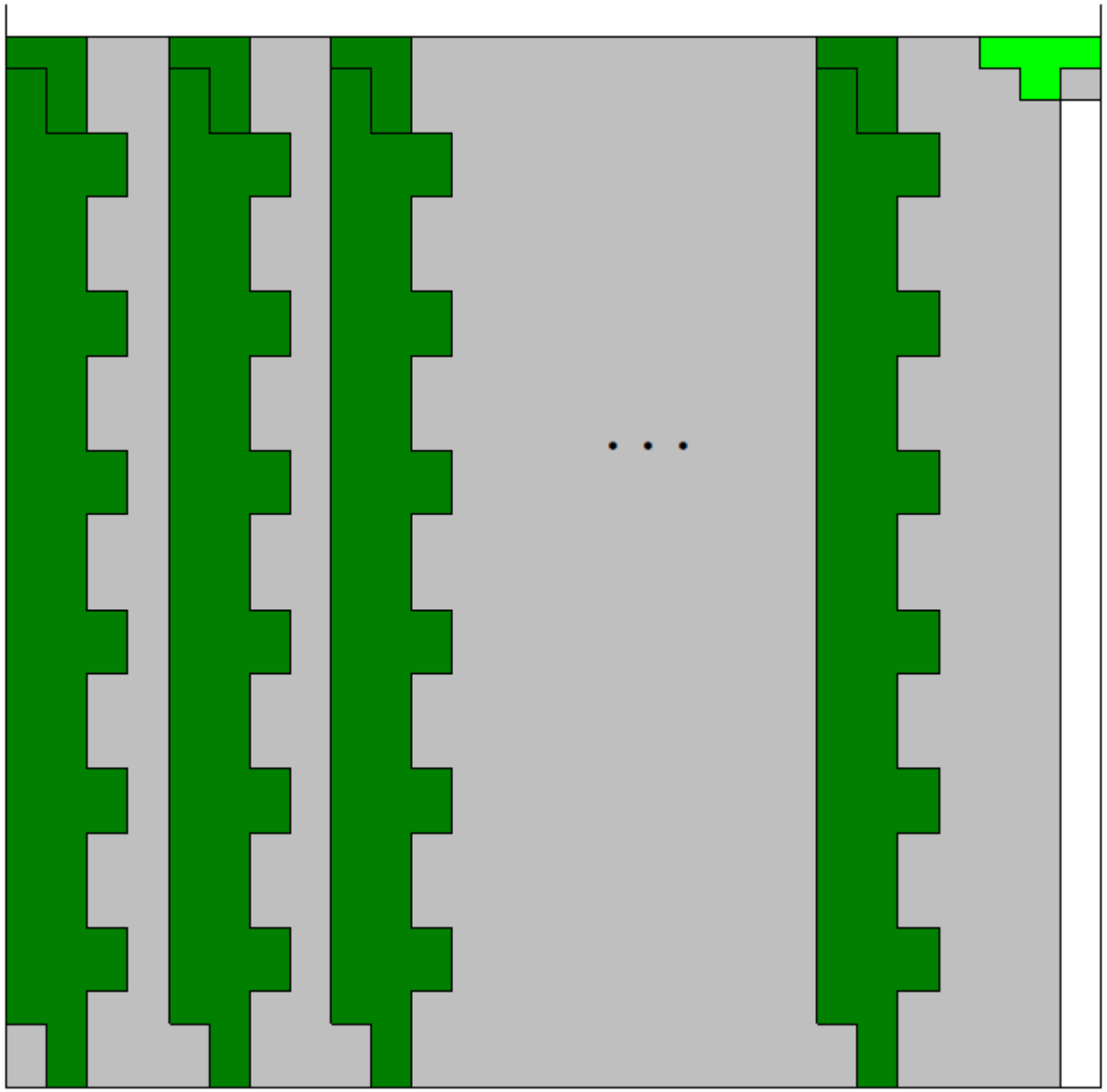
Finale Pieces

[Breukelaar, Demaine, Hohenberger, Hoogeboom, Kosters, Liben-Nowell 2003]



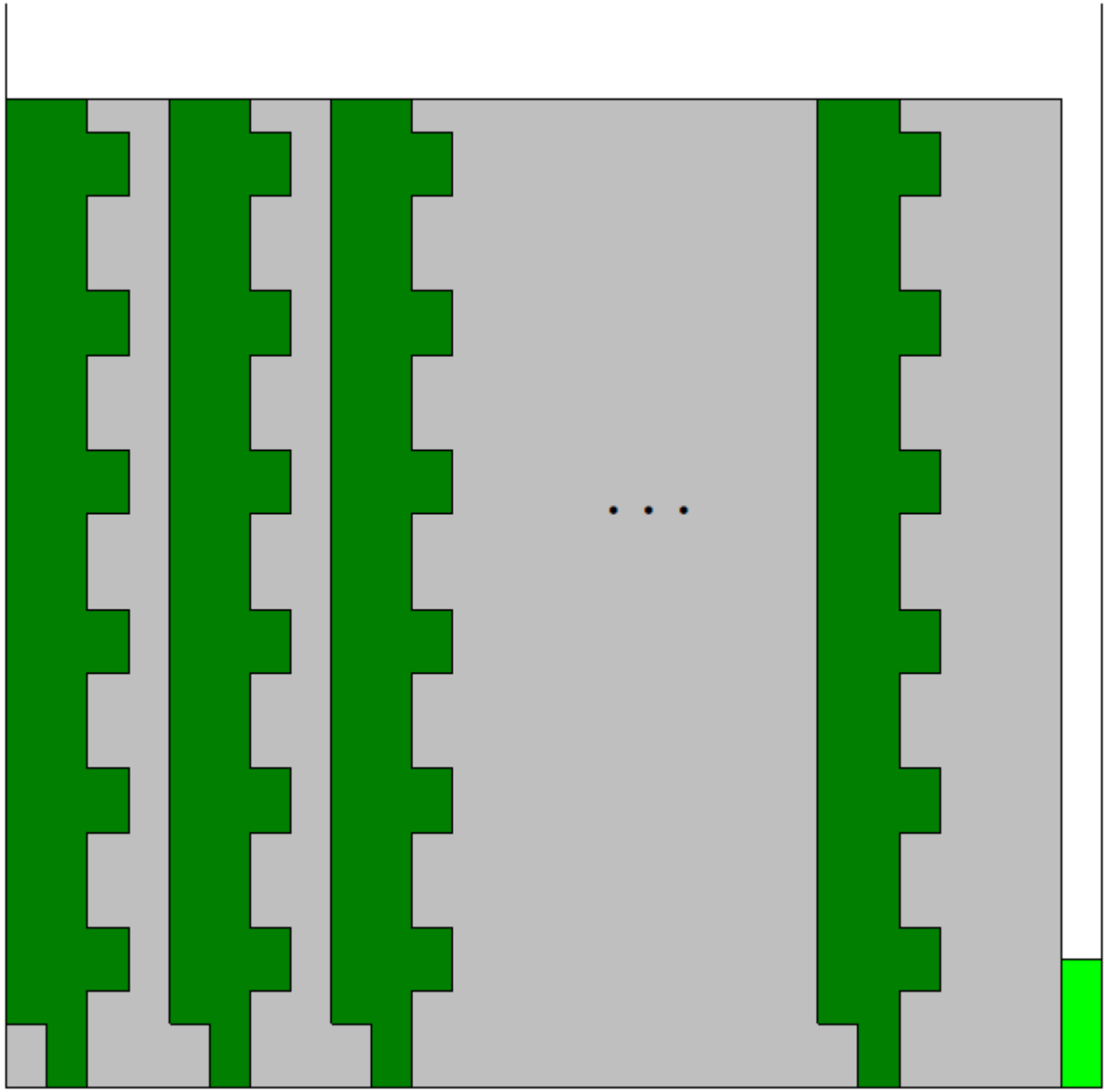
Finale Pieces

[Breukelaar, Demaine, Hohenberger, Hoogeboom, Kosters, Liben-Nowell 2003]



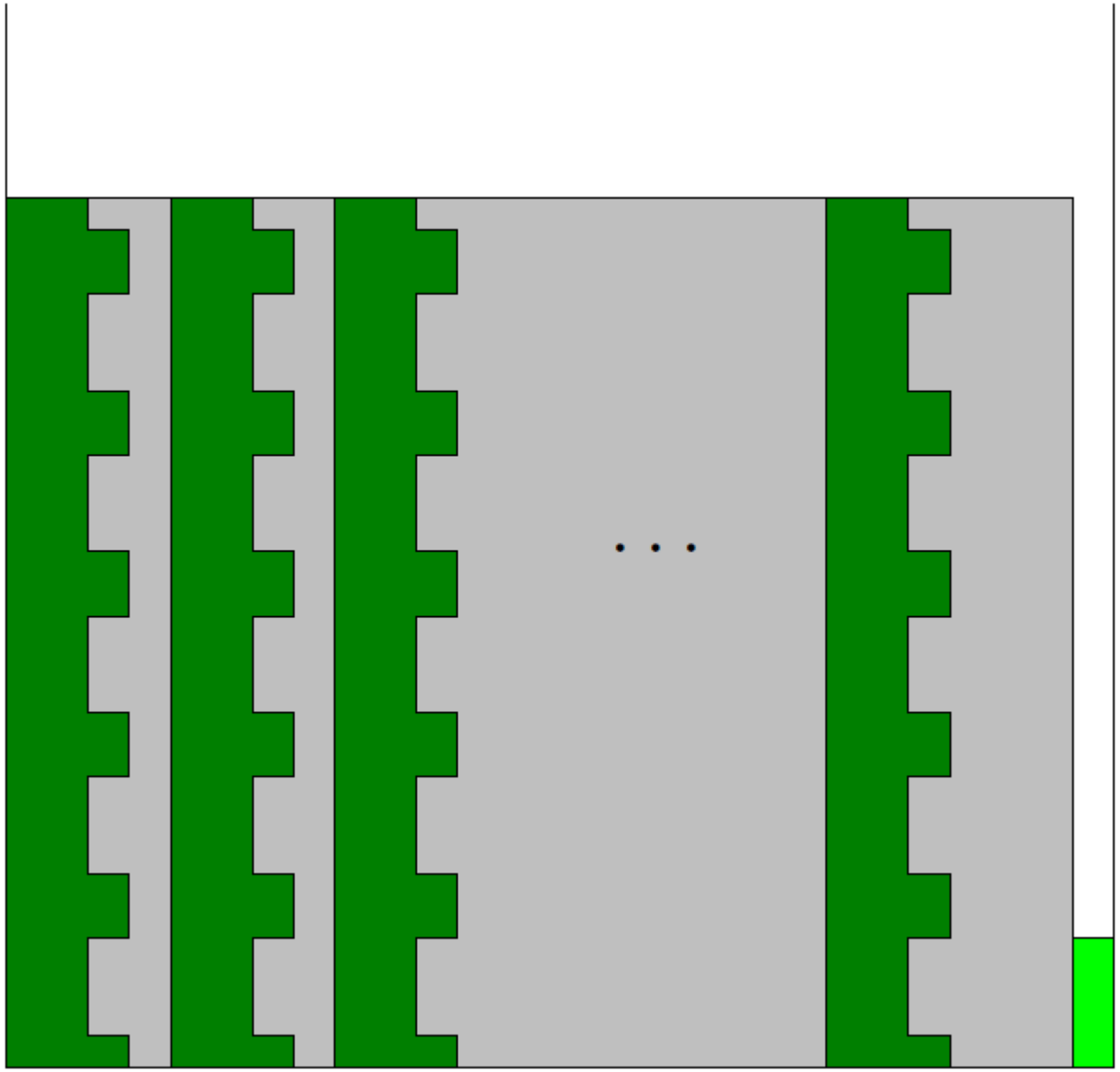
Finale Pieces

[Breukelaar, Demaine, Hohenberger, Hoogeboom, Kosters, Liben-Nowell 2003]



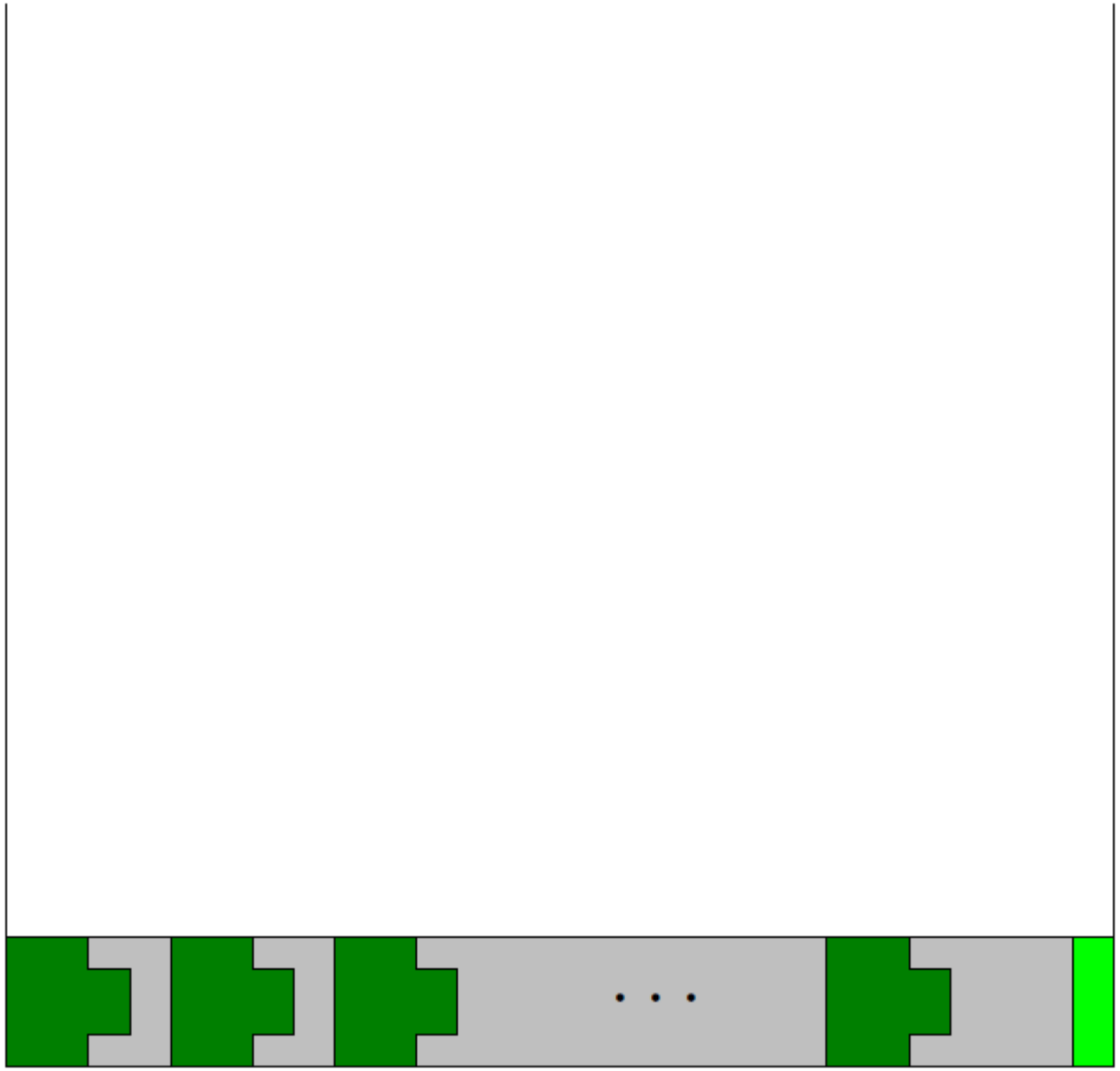
Finale Pieces

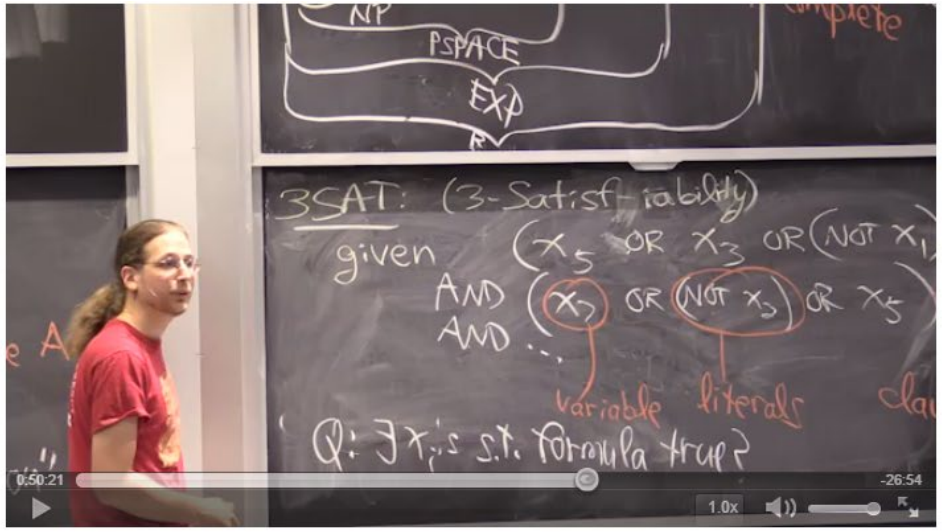
[Breukelaar, Demaine, Hohenberger, Hoogeboom, Kosters, Liben-Nowell 2003]



Finale Pieces

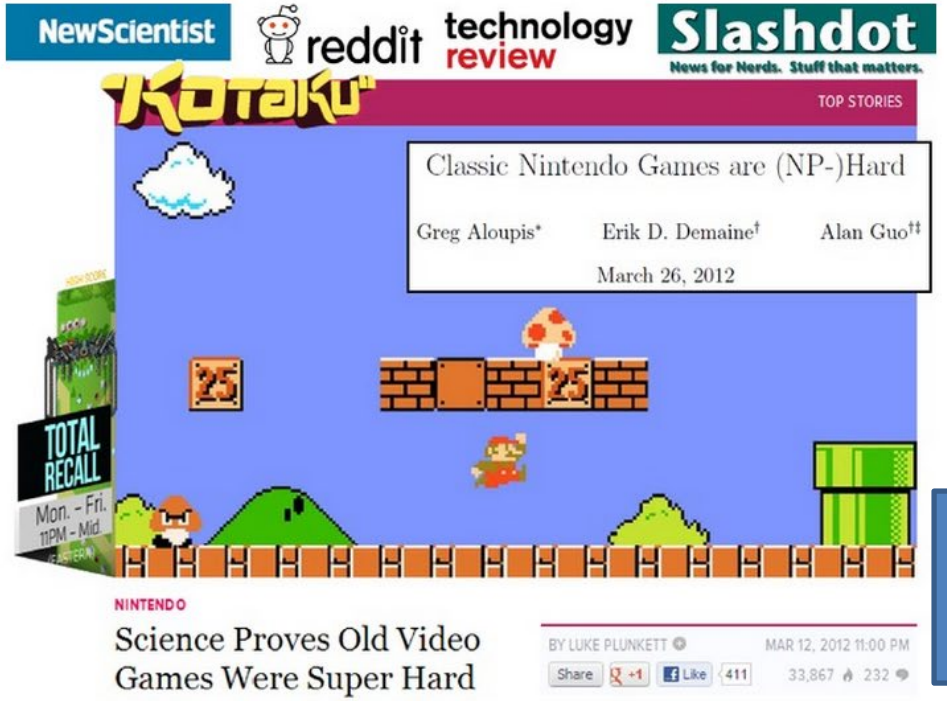
[Breukelaar, Demaine, Hohenberger, Hoogeboom, Kosters, Liben-Nowell 2003]





Download Video: 360p, 720p

Slides, page 5/21 • [previous page] • [next page] • [PDF]
Video times: • 47:27–50:56



Handwritten notes, page 8/8 • [previous p
Video times: • 47:27-

Examples of hardness proofs:

① Super Mario Bros. is

- reduction from 3SAT:
(make true) a formul
(x1 OR
AND (x5 OR
AND ...
3 lite

② Rush Hour is PSPACE-co

- reduction from NCL (
given directed graph w
find sequence of edge
reverse a target edge
maintaining total in-w
- only need AND verte

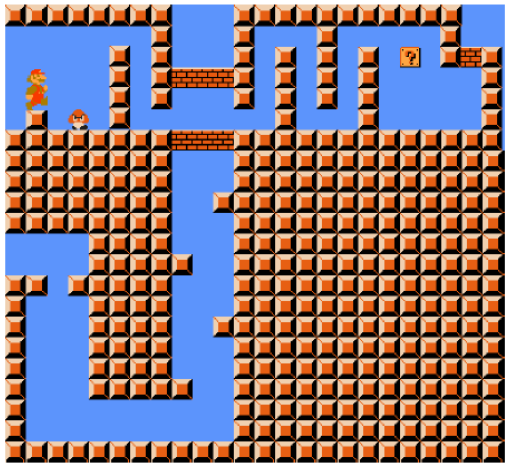
(in fact: OR can be
input active
- Constraint Logic is a
proving hardness of

Handwritten notes, page 8/8 • [previous p

RETURNING SPRING 2019!

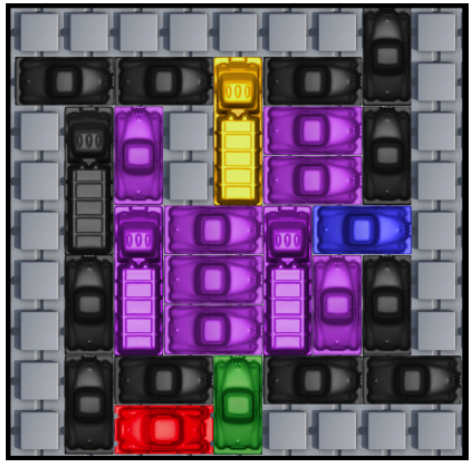
Algorithmic Lower Bounds: Fun with Hardness Proofs

Super Mario Bros.



Crossover gadget for NP-hardness

Rush Hour



AND gadget for PSPACE-hardness

Minesweeper

OR gadget for NP-hardness

Hardness Made Easy*

Learn **when to give up** the search for efficient algorithms; see **connections** between computational problems; **solve puzzles** to prove theorems, solve **open problems,** and write papers.

Topics: NP, PSPACE, EXPTIME, EXPTIME, EXPSPACE, 3SUM, approximation, fixed parameter, games & puzzles, 3SUM, key problems, gadgets, and proof styles.

Fall 2014

6.890 taught by Professor Erik Demaine
Grad H, AUS, and Theoretical CS Concentration
Tuesday & Thursday 3:30-5:00pm in room 2-105
http://courses.csail.mit.edu/6.890/
sign up for our mailing list to join the class

** Easiness not guaranteed. Side effects such as open problems and a heightened sense of complexity may occur. Ask your advisor if 6.890 is right for you!*