Some other polynomial-time reductions

proving NP-completeness is fun


25
25
25

Classic Nintendo Games are (Computationally) Hard

Greg Aloupis[*] Erik D. Demaine[†] Alan Guo[†‡] Giovanni Viglietta[§]

February 10, 2015

Abstract

We prove NP-hardness results for five of Nintendo's largest video game franchises: Mario, Donkey Kong, Legend of Zelda, Metroid, and Pokémon. Our results apply to generalized versions of Super Mario Bros. 1–3, The Lost Levels, and Super Mario World; Donkey Kong Country 1–3; all Legend of Zelda games; all Metroid games; and all Pokémon role-playing games. In addition, we prove PSPACE-completeness of the Donkey Kong Country games and several Legend of Zelda games.

Reduction from: 3-SAT

3-SAT

input: a Boolean formula consisting of 3-literal clauses over n variables

goal: does there exist a satisfying assignment (making all clauses true)?

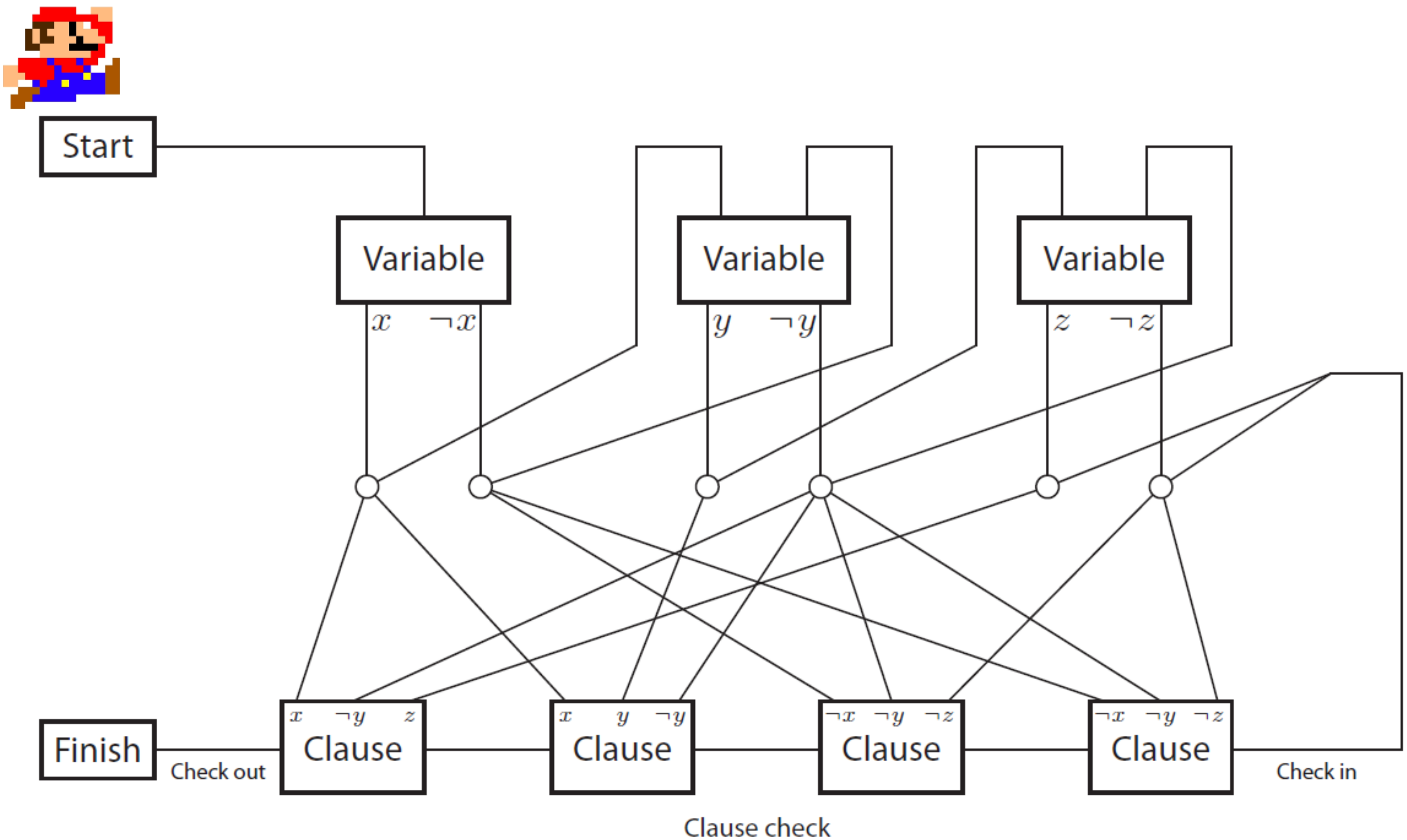
$$\Phi = (\overline{x_1} \lor x_2 \lor x_3) \land (x_1 \lor \overline{x_2} \lor x_3) \land (\overline{x_1} \lor x_2 \lor x_4)$$

yes instance: x_1 = true, x_2 = true, x_3 = false, x_4 = false

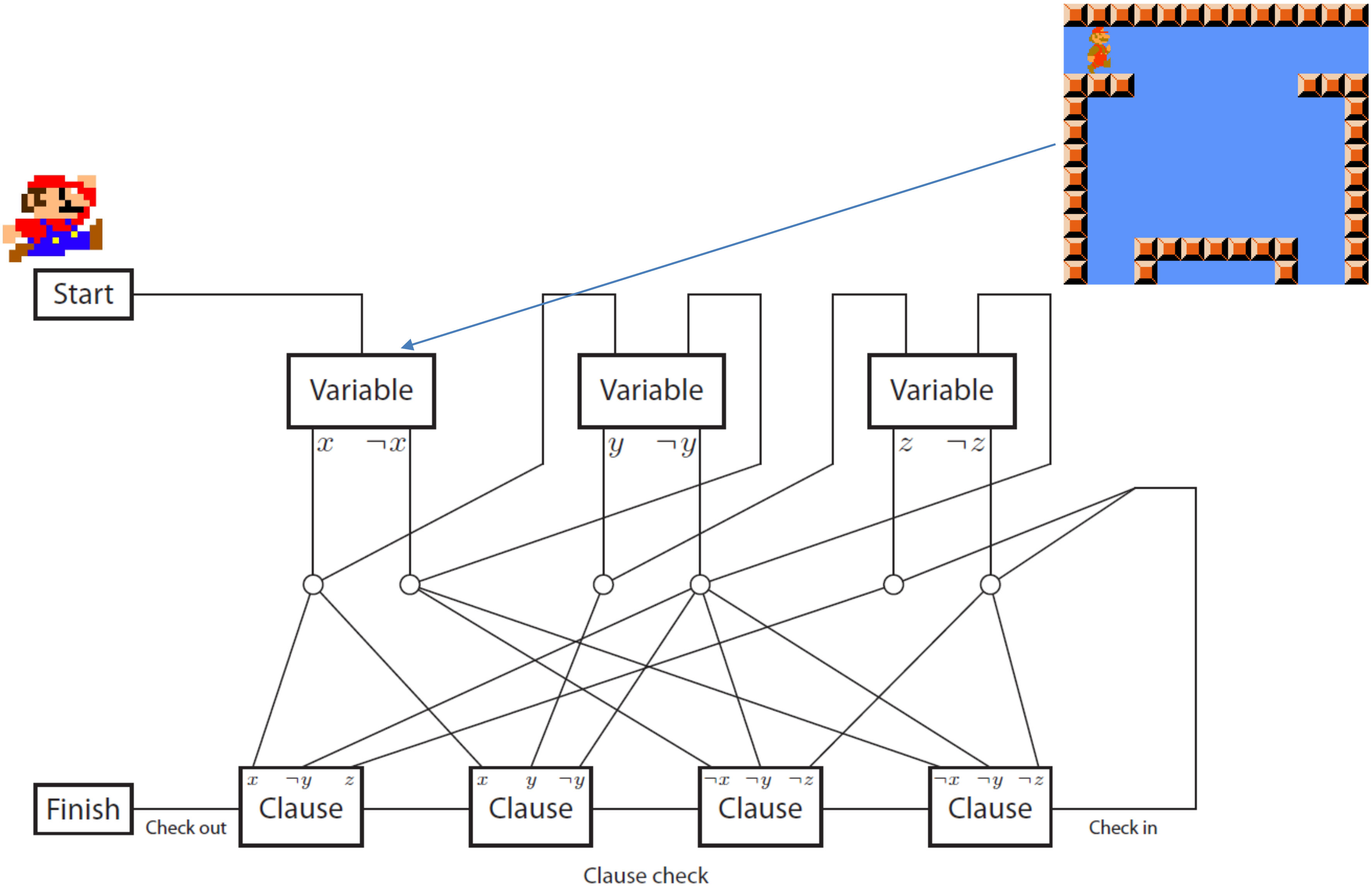
3-SAT is NP-complete

idea:

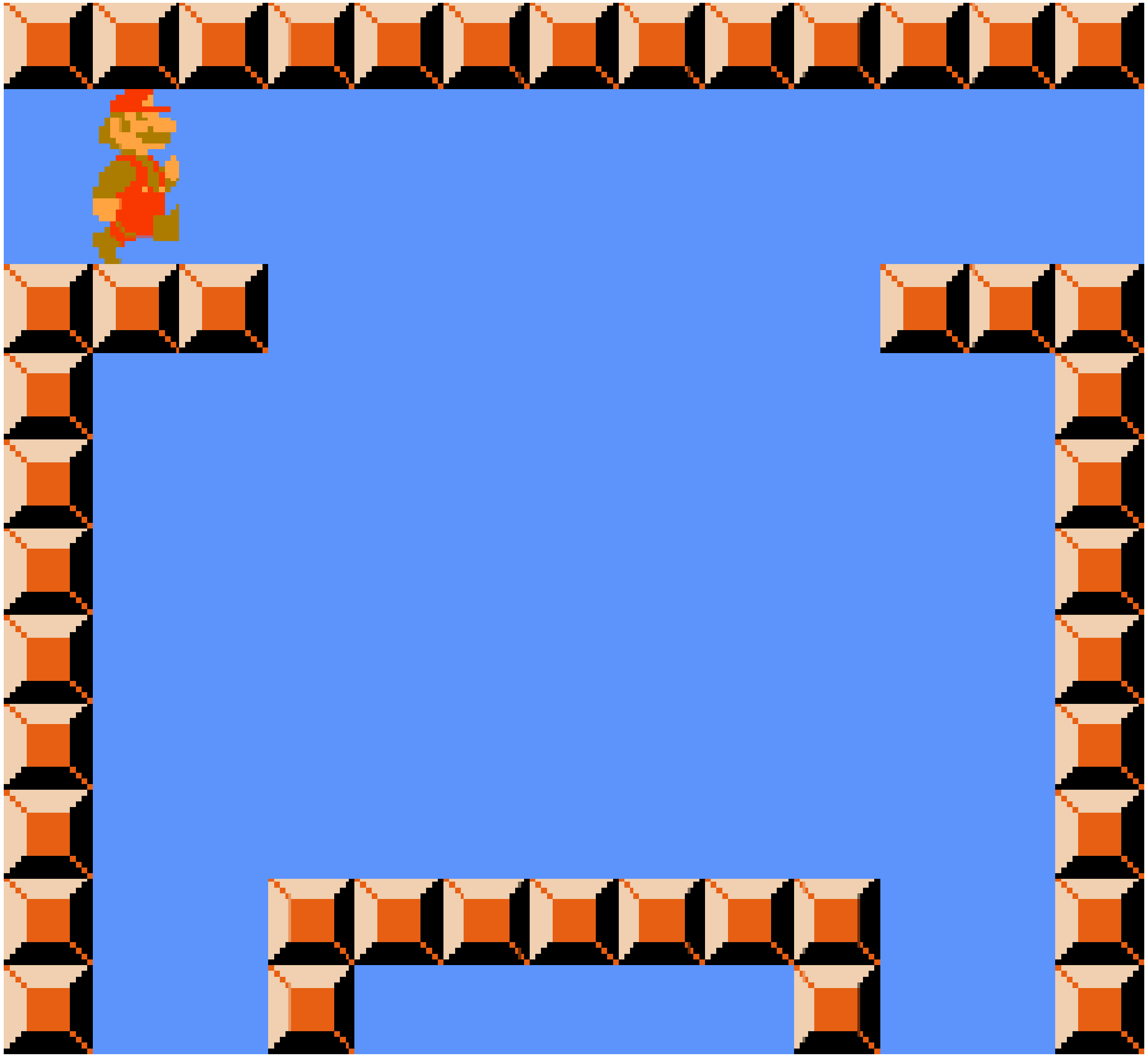
given a 3-SAT instance we build a level/instance of Super Mario that is solvable if and only if the formula is satisfiable



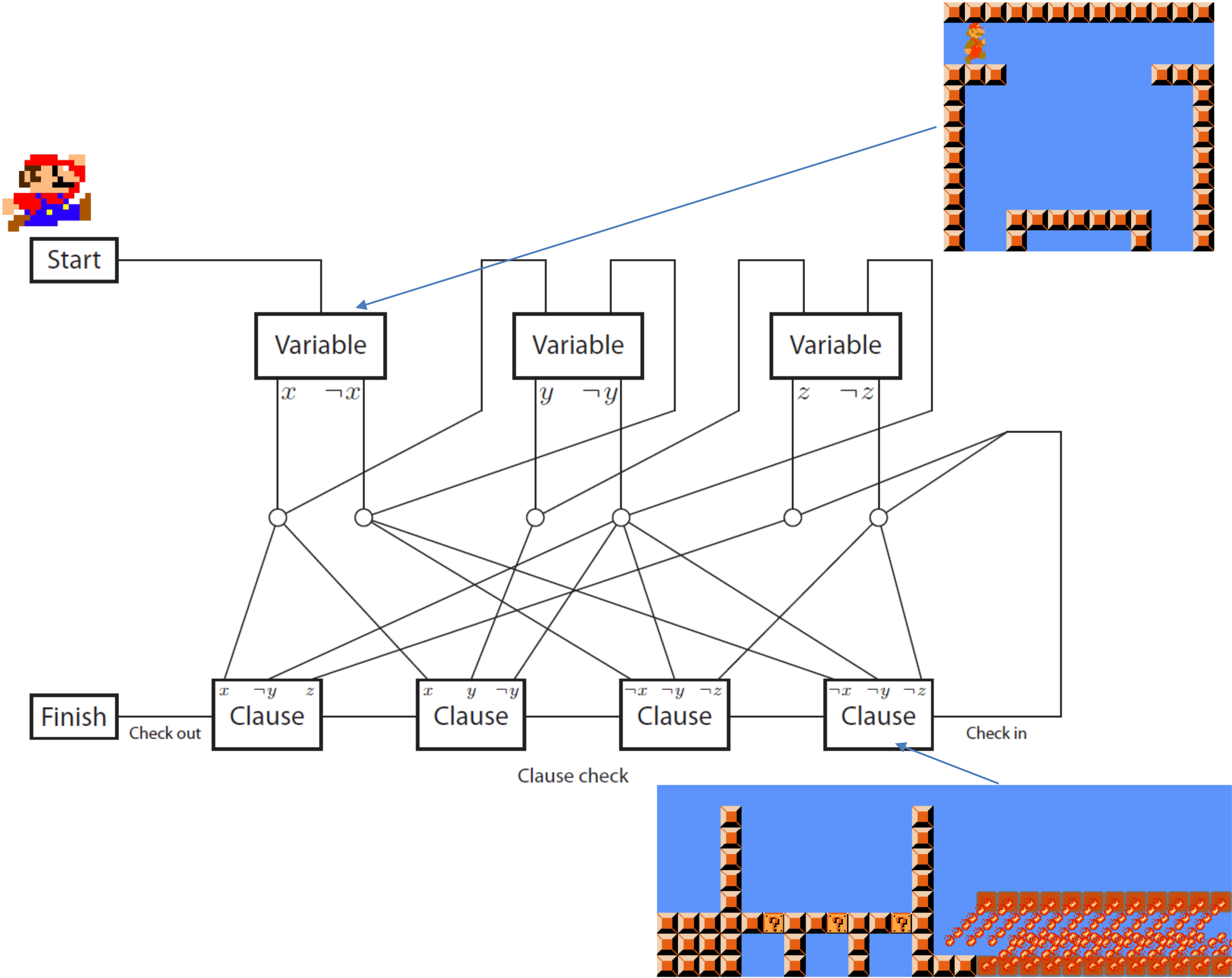
Start
Variable
x $\neg x$
Variable
y $\neg y$
Variable
z $\neg z$
Finish
Check out
x $\neg y$ z
Clause
x y $\neg y$
Clause
$\neg x$ $\neg y$ $\neg z$
Clause
$\neg x$ $\neg y$ $\neg z$
Clause
Check in
Clause check


Start
Variable
x $\neg x$
Variable
y $\neg y$
Variable
z $\neg z$
x $\neg y$ z
Clause
x y $\neg y$
Clause
$\neg x$ $\neg y$ $\neg z$
Clause
$\neg x$ $\neg y$ $\neg z$
Clause
Finish
Check out
Check in
Clause check

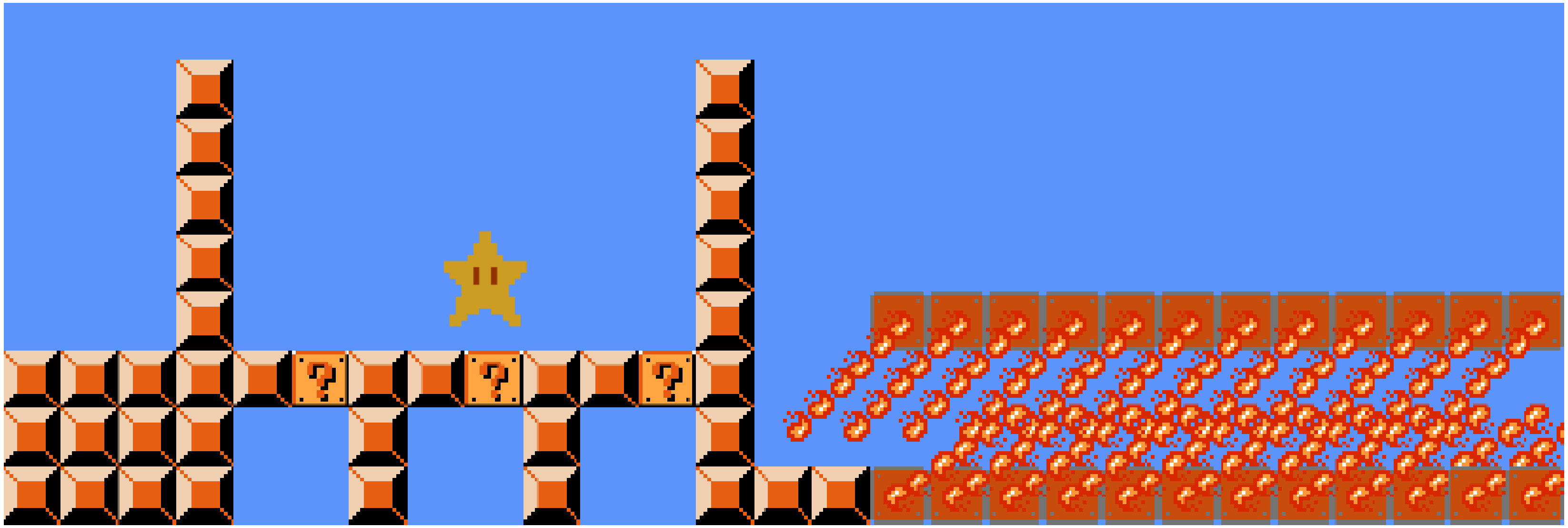
variable gadget

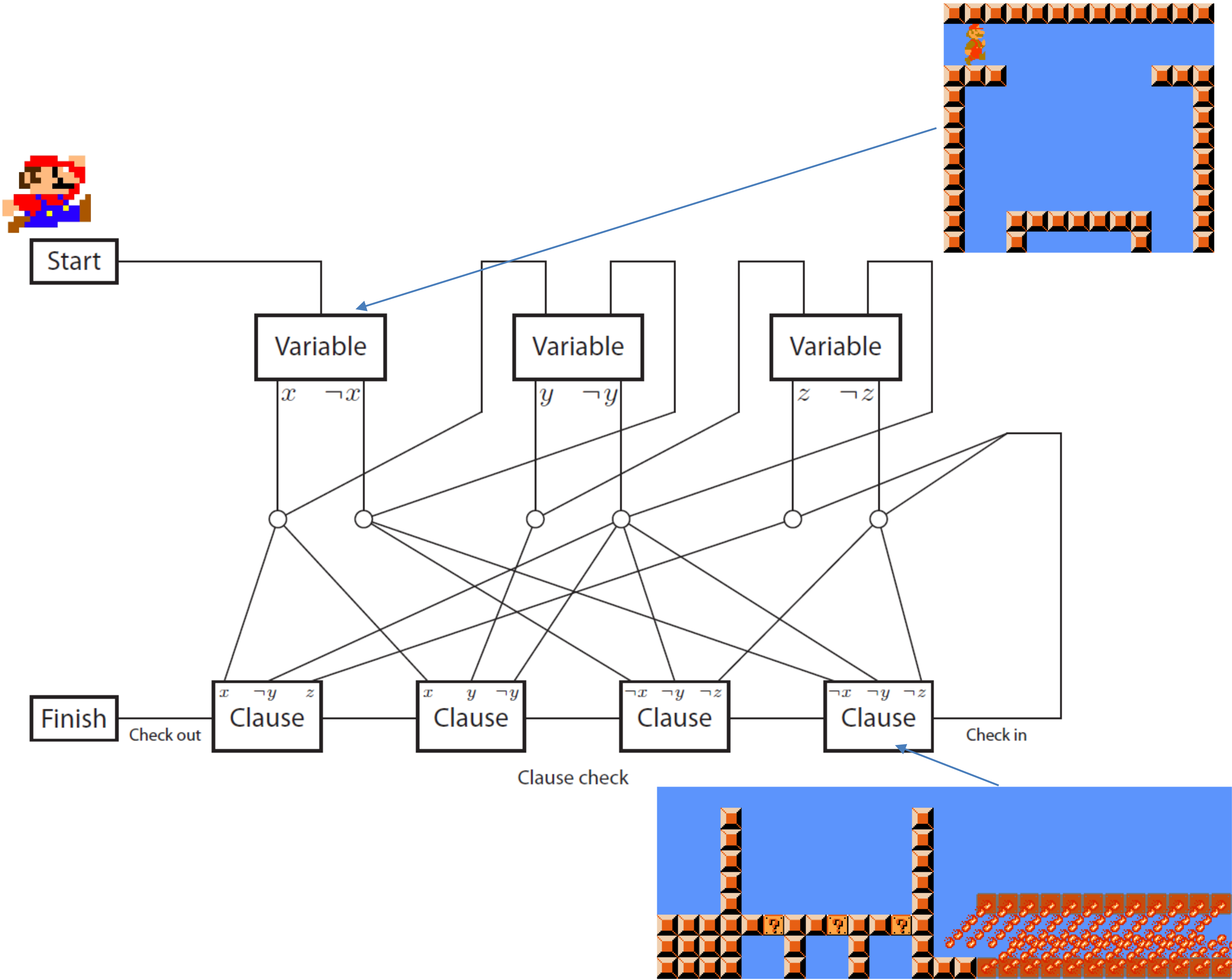


x_i $\neg x_i$

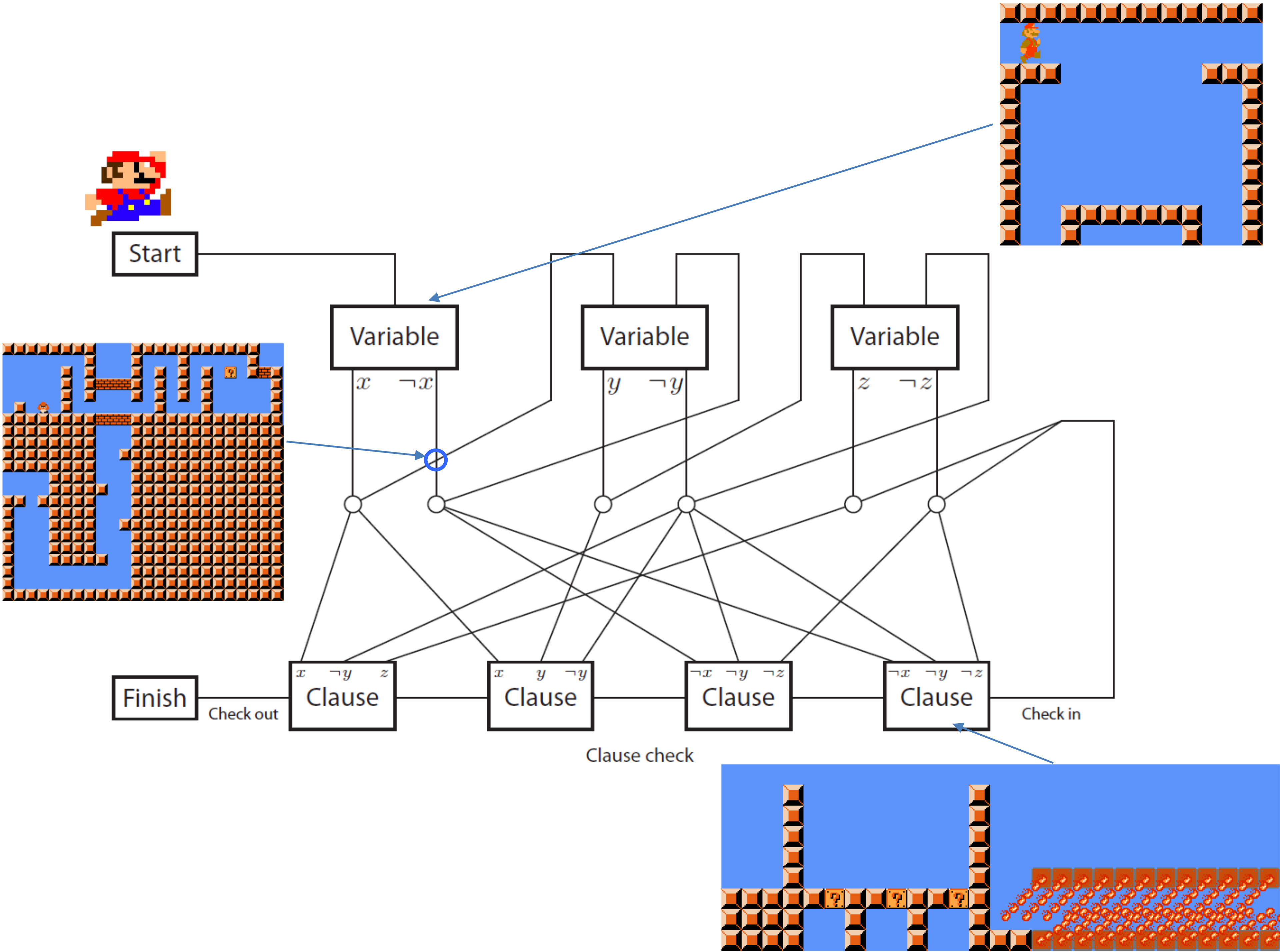

Start
Variable
x ¬x
Variable
y ¬y
Variable
z ¬z
x ¬y z
Clause
x y ¬y
Clause
¬x ¬y ¬z
Clause
¬x ¬y ¬z
Clause
Finish
Check out
Check in
Clause check

Clause gadget

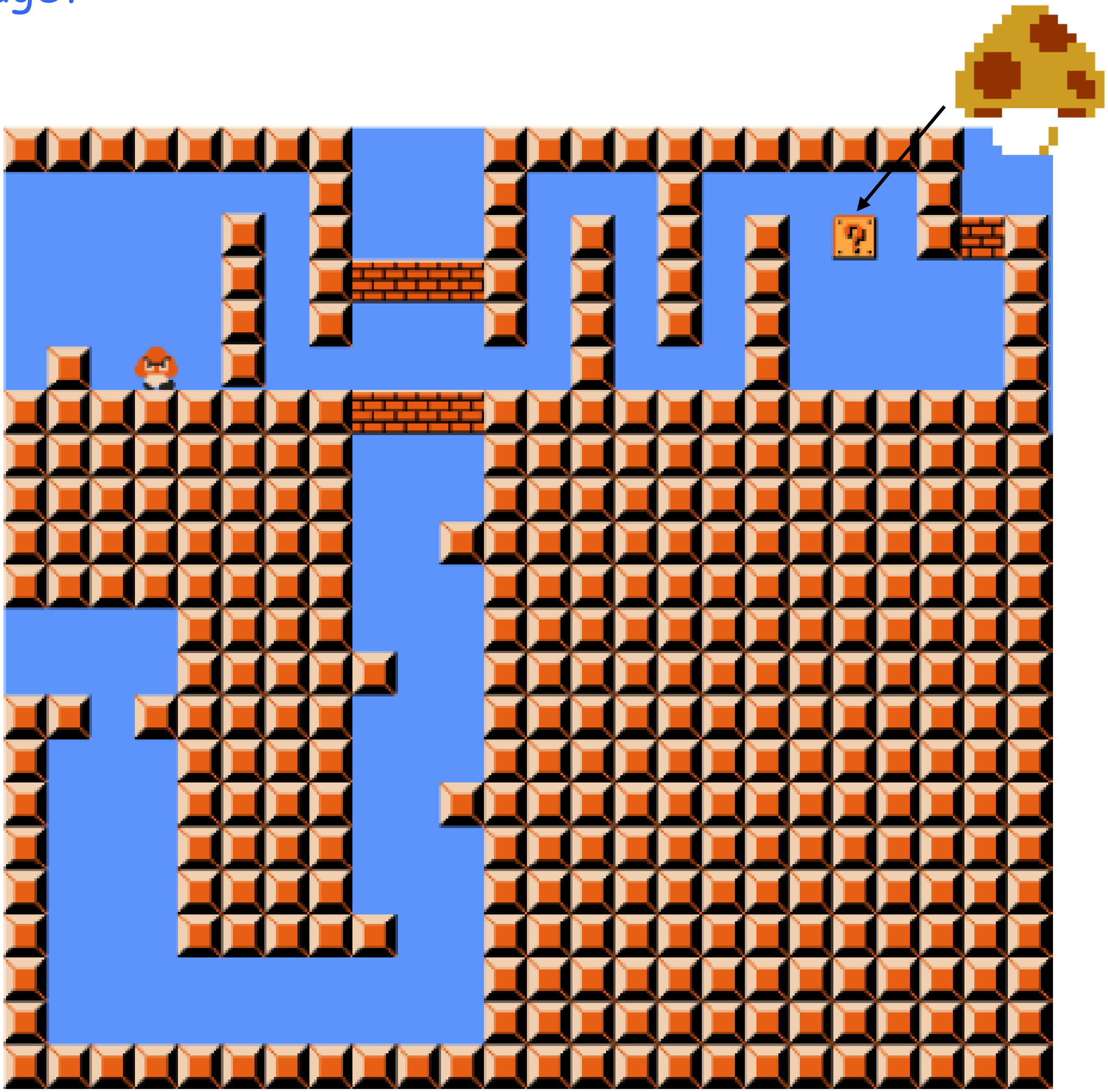


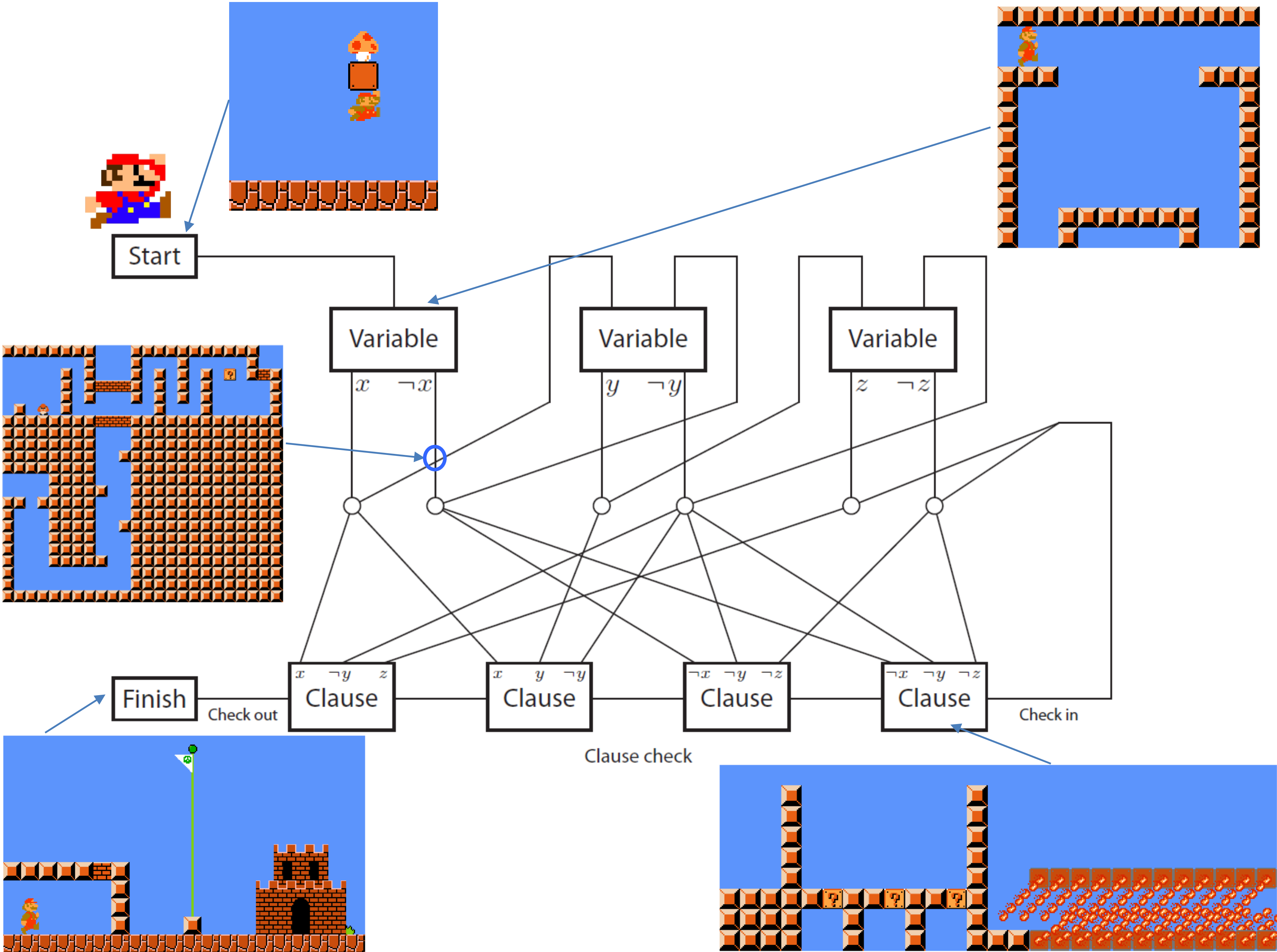


Start
Variable
Variable
Variable
x $\neg x$
y $\neg y$
z $\neg z$
x $\neg y$ z
Clause
x y $\neg y$
Clause
$\neg x$ $\neg y$ $\neg z$
Clause
$\neg x$ $\neg y$ $\neg z$
Clause
Finish
Check out
Check in
Clause check


Start
Variable
x $\neg x$
Variable
y $\neg y$
Variable
z $\neg z$
Finish
Check out
x $\neg y$ z
Clause
x y $\neg y$
Clause
$\neg x$ $\neg y$ $\neg z$
Clause
$\neg x$ $\neg y$ $\neg z$
Clause
Check in
Clause check

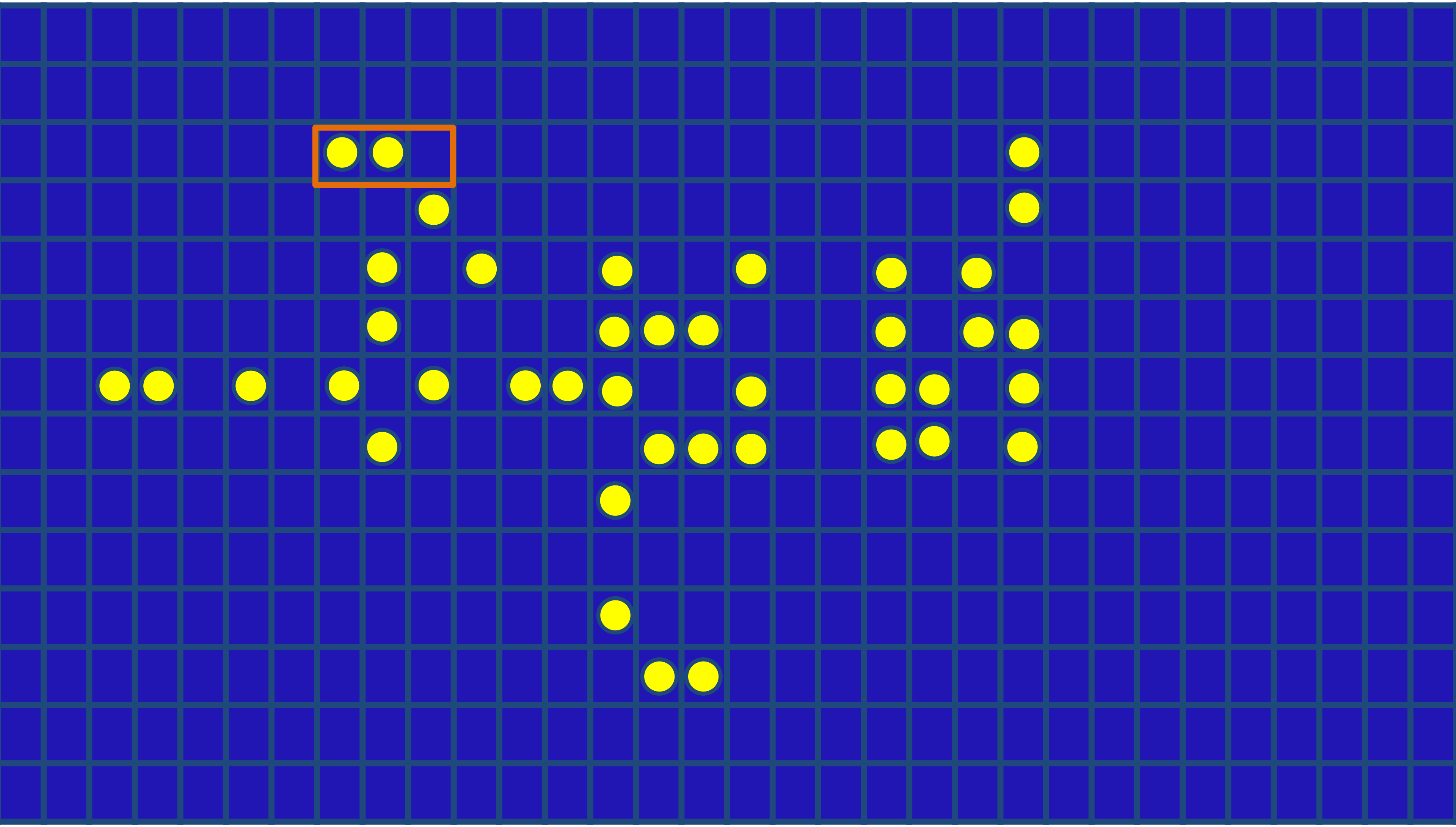
Crossover gadget



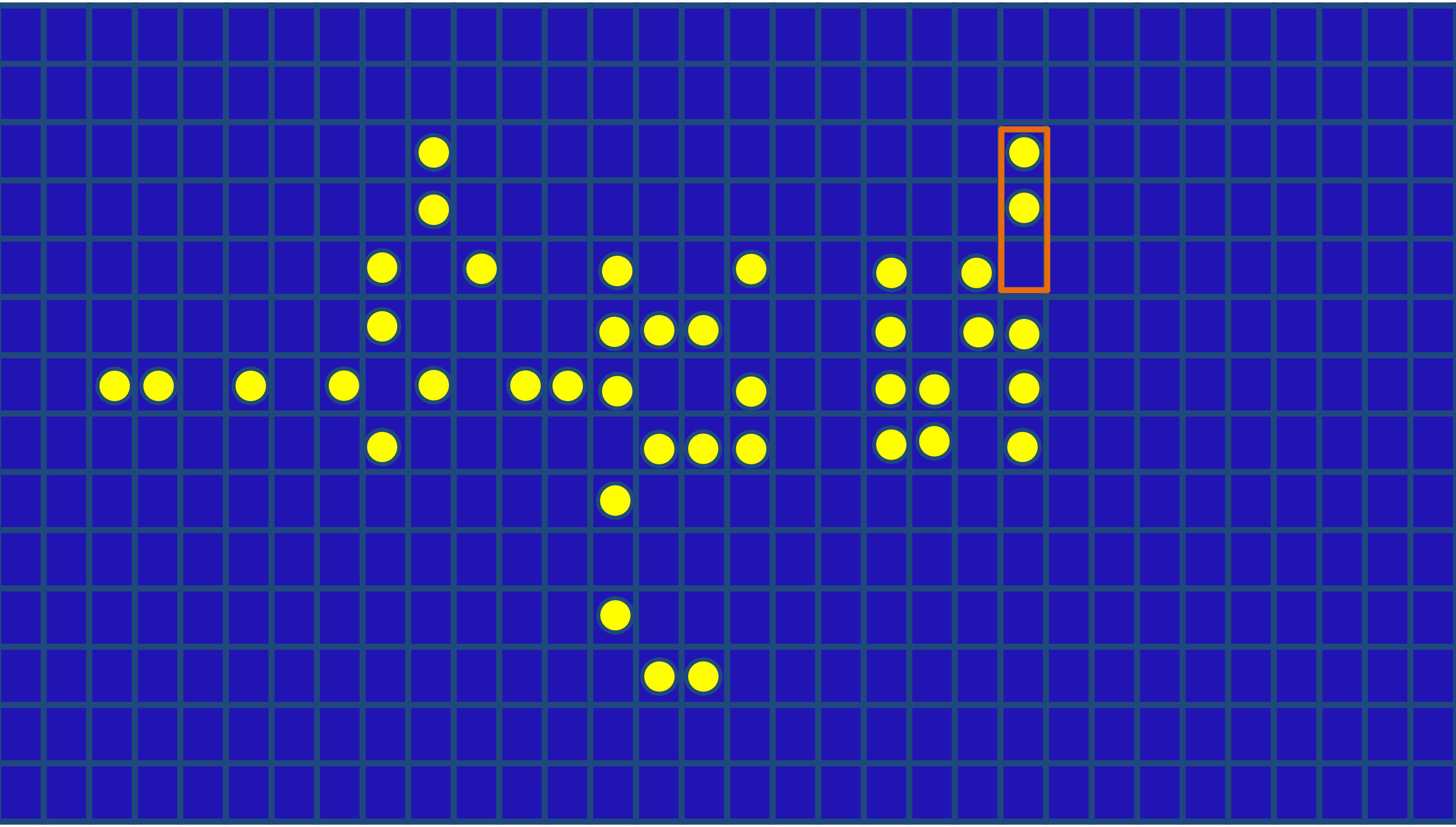

Start
Variable
x $\neg x$
Variable
y $\neg y$
Variable
z $\neg z$
Finish
Check out
x $\neg y$ z
Clause
x y $\neg y$
Clause
$\neg x$ $\neg y$ $\neg z$
Clause
$\neg x$ $\neg y$ $\neg z$
Clause
Check in
Clause check

Peg Solitaire

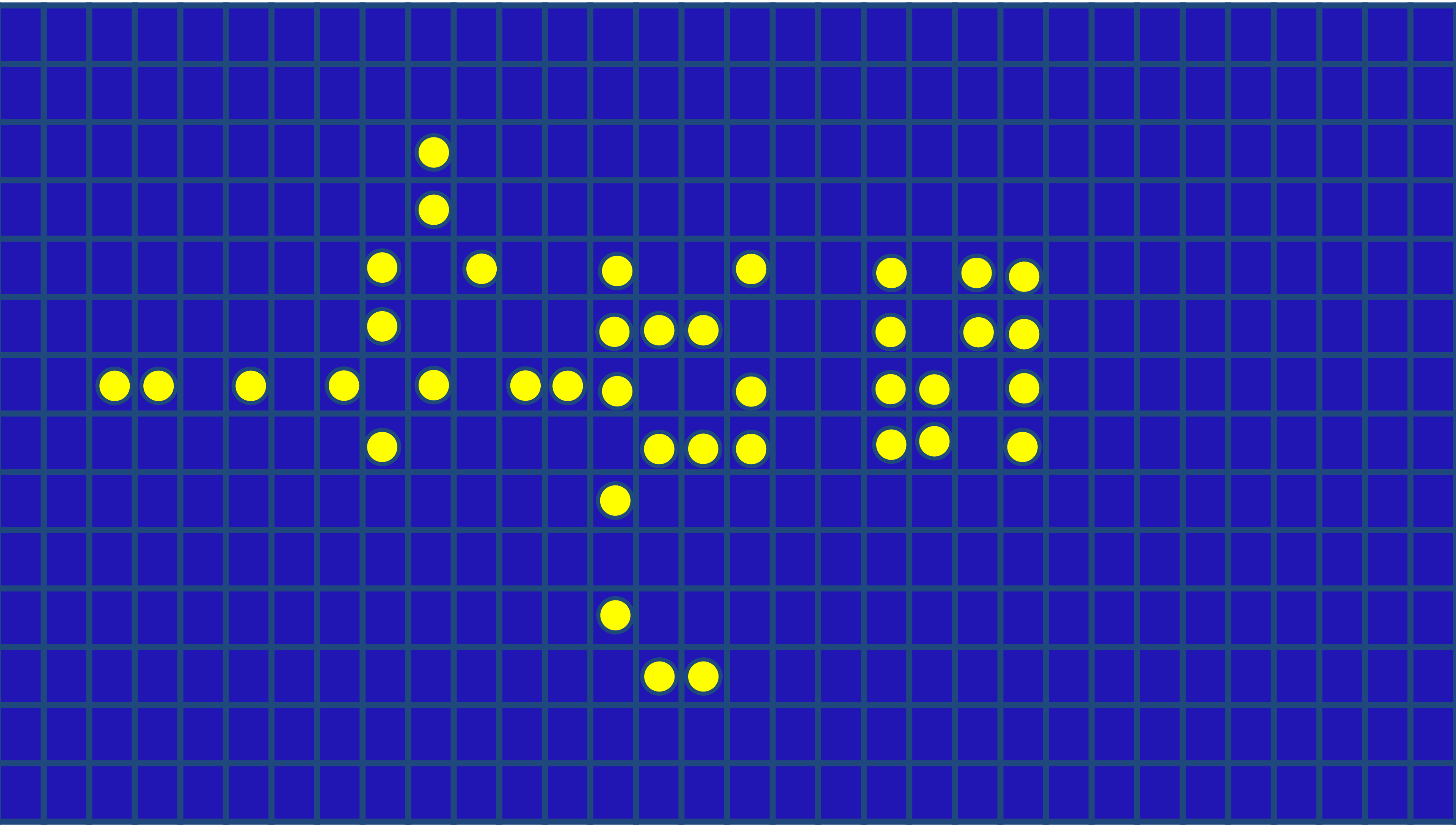
Peg Solitaire



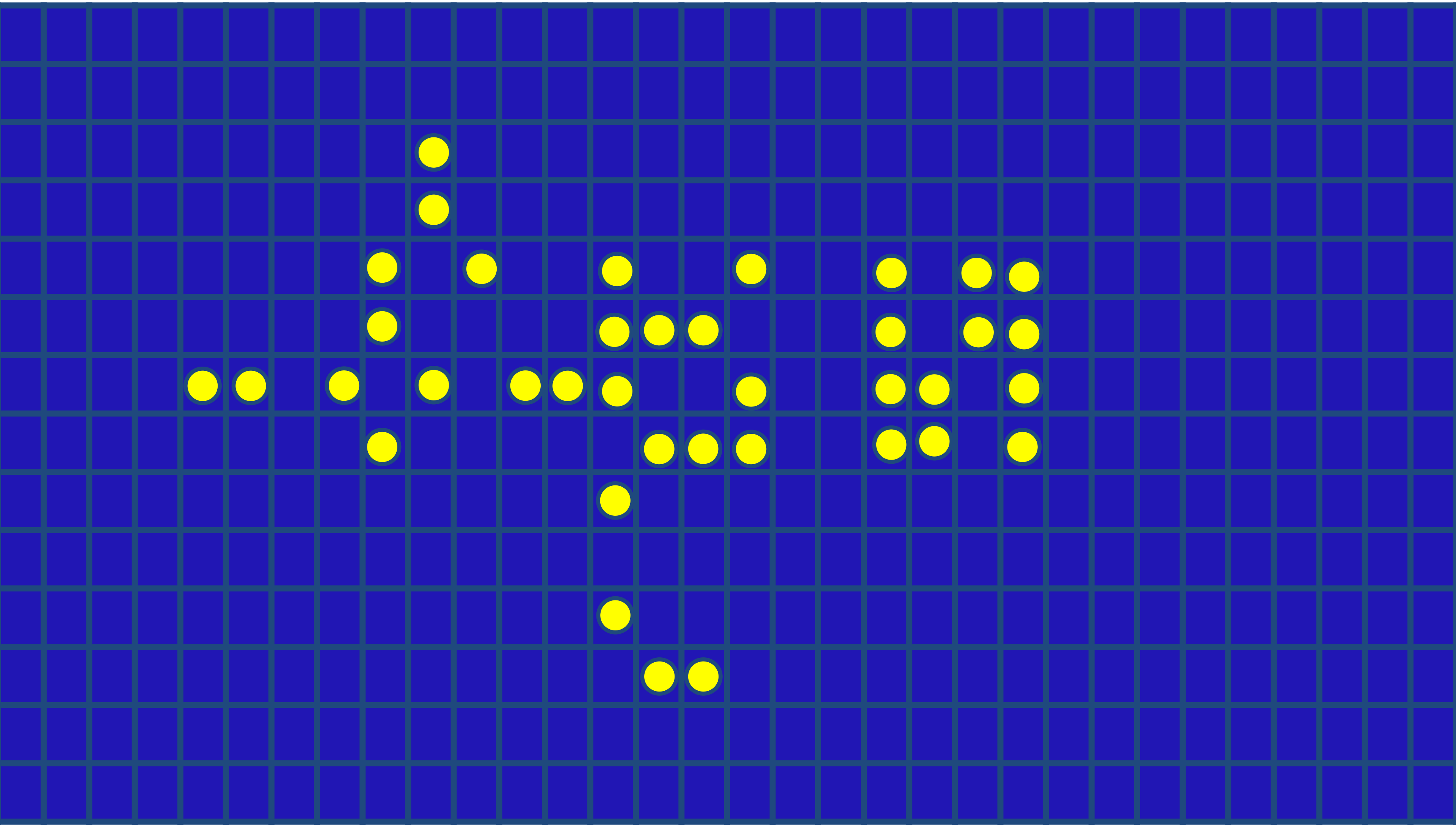
Peg Solitaire



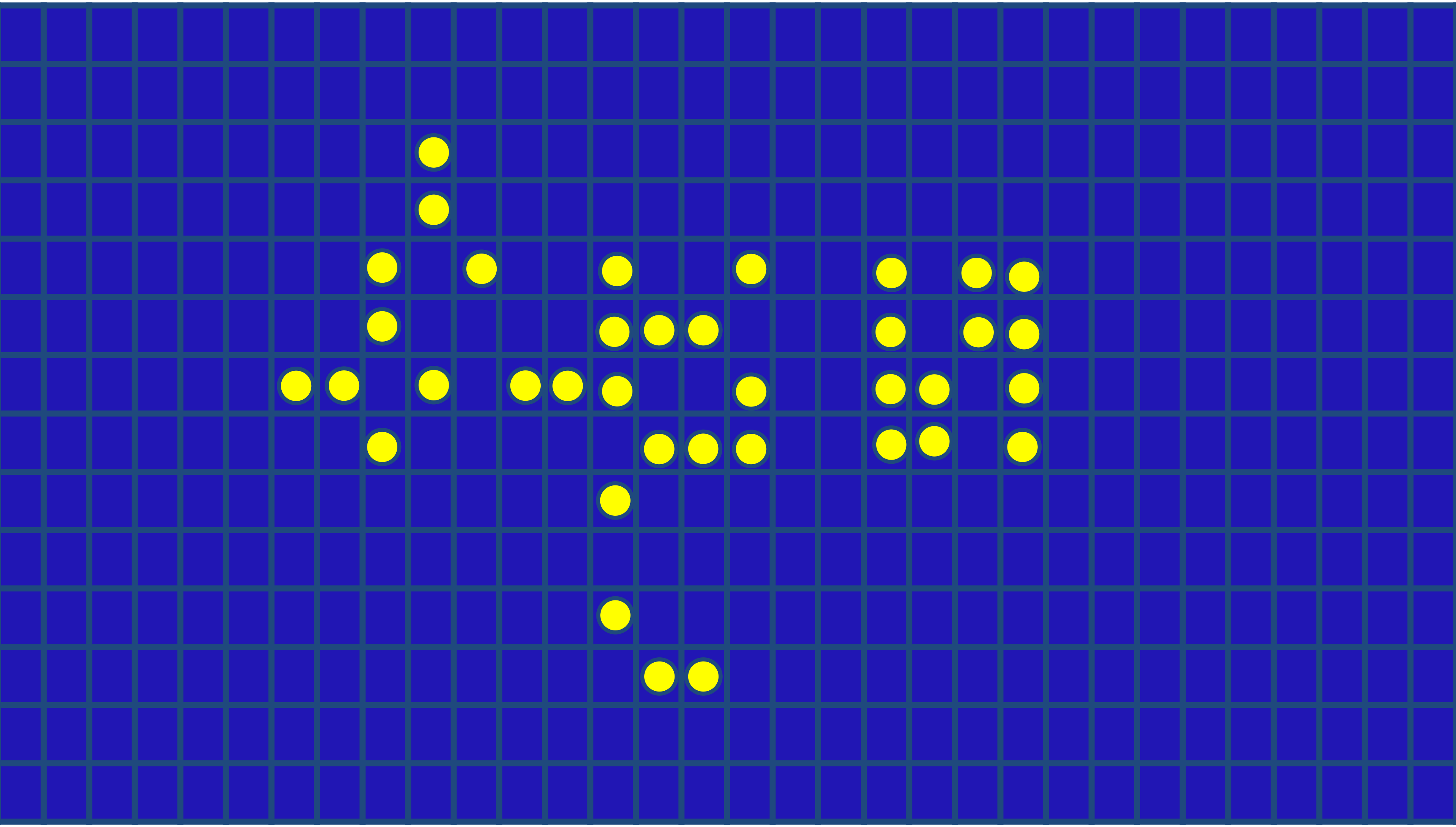
Peg Solitaire



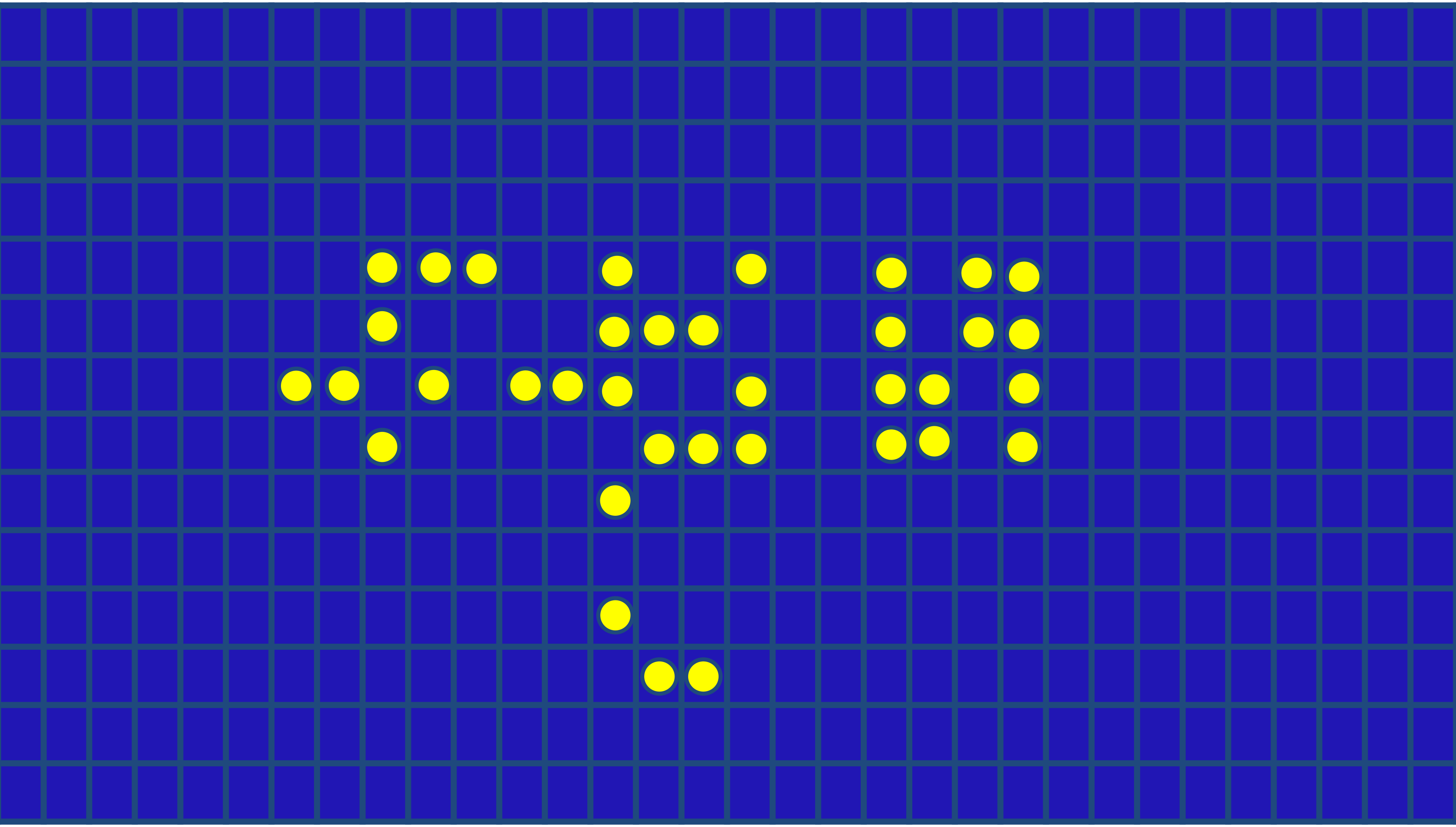
Peg Solitaire



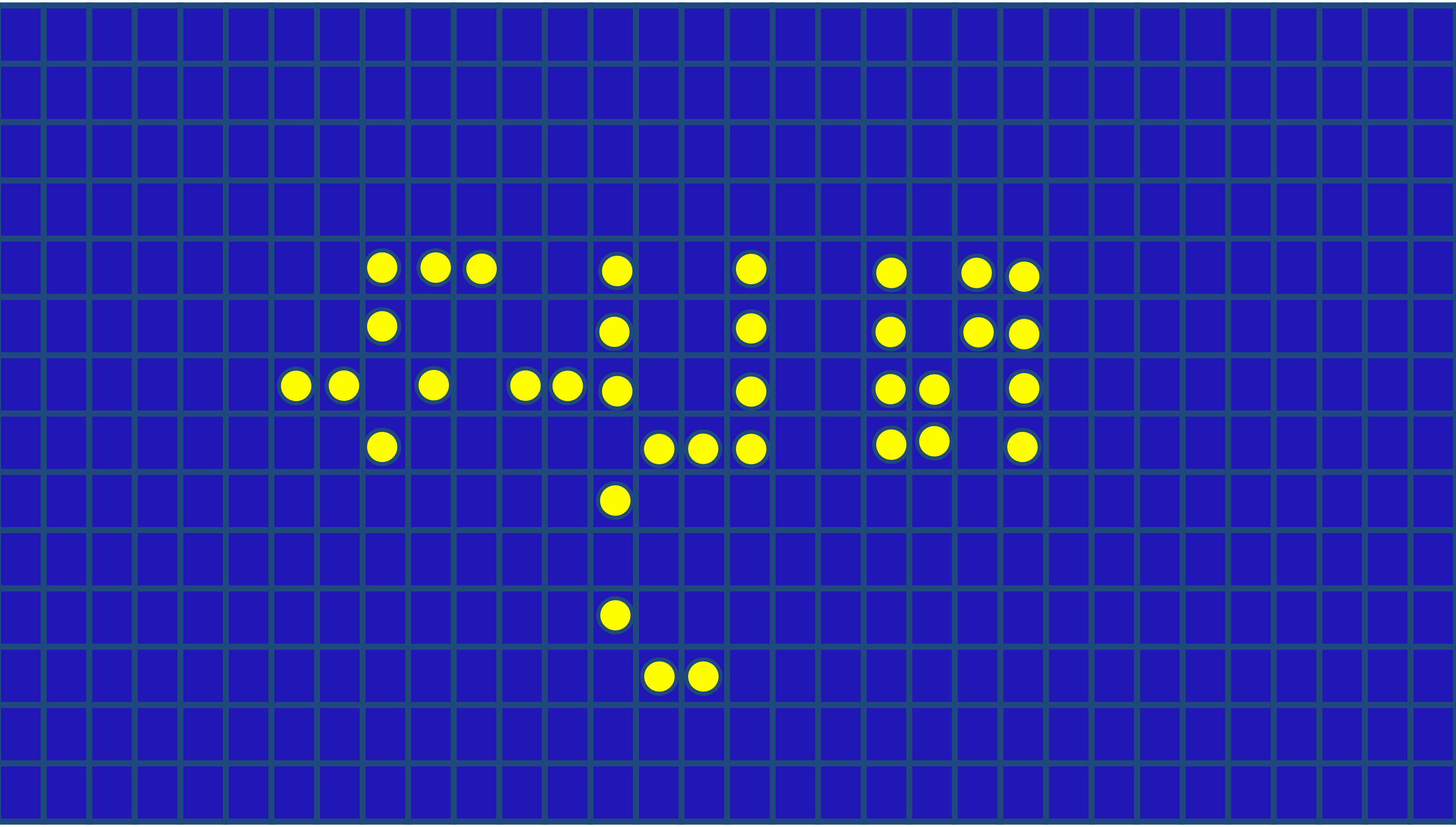
Peg Solitaire



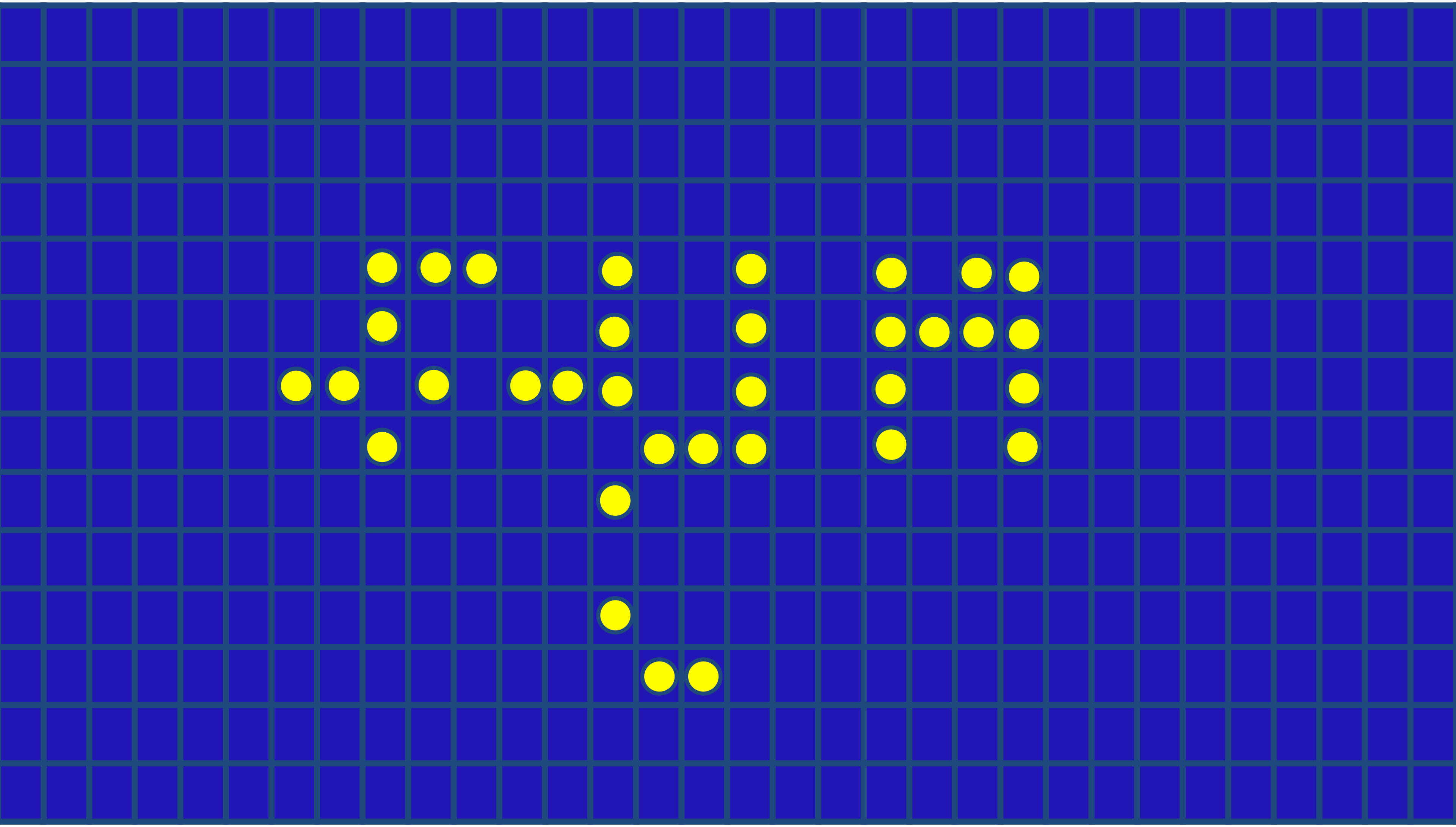
Peg Solitaire



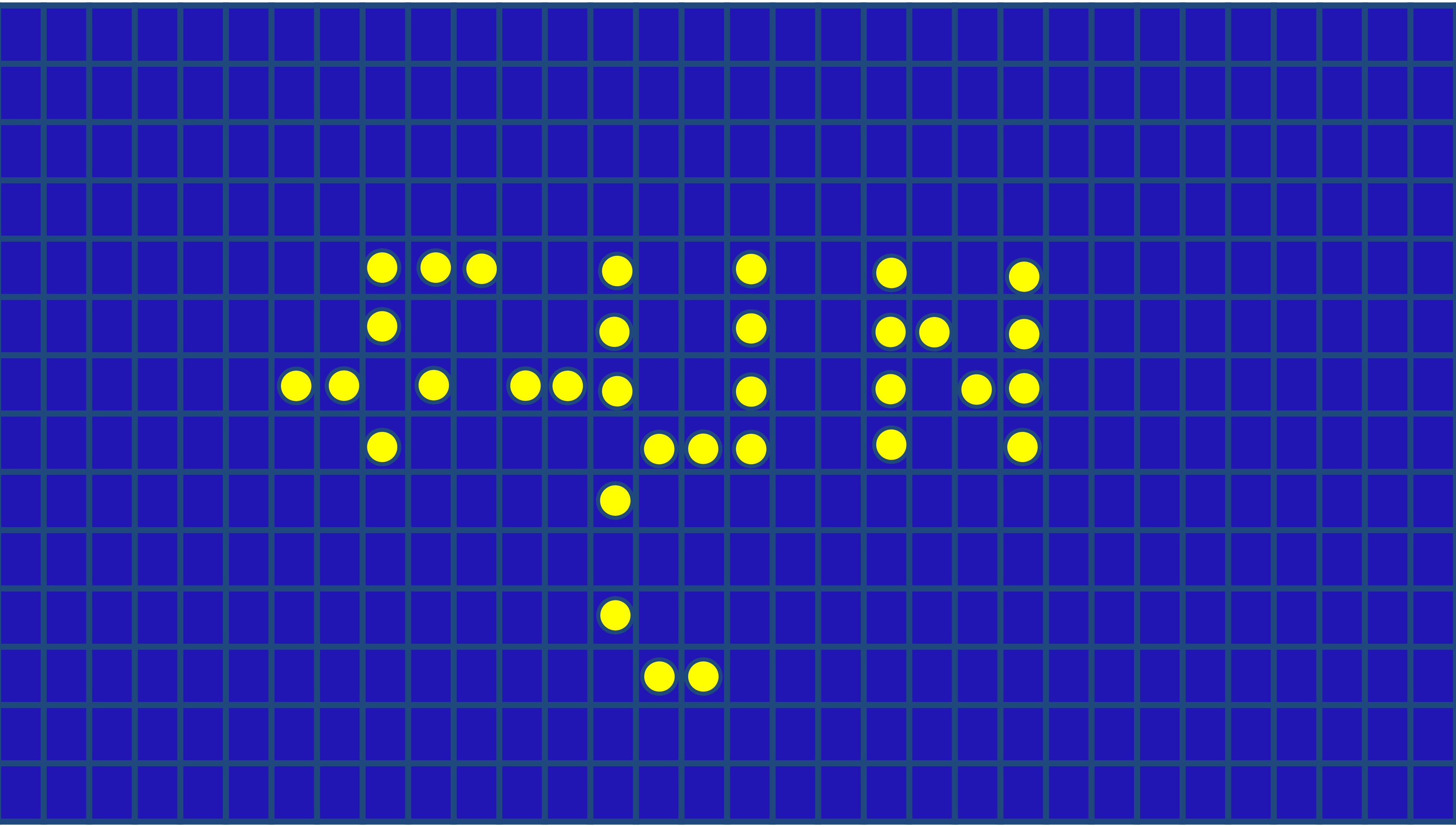
Peg Solitaire



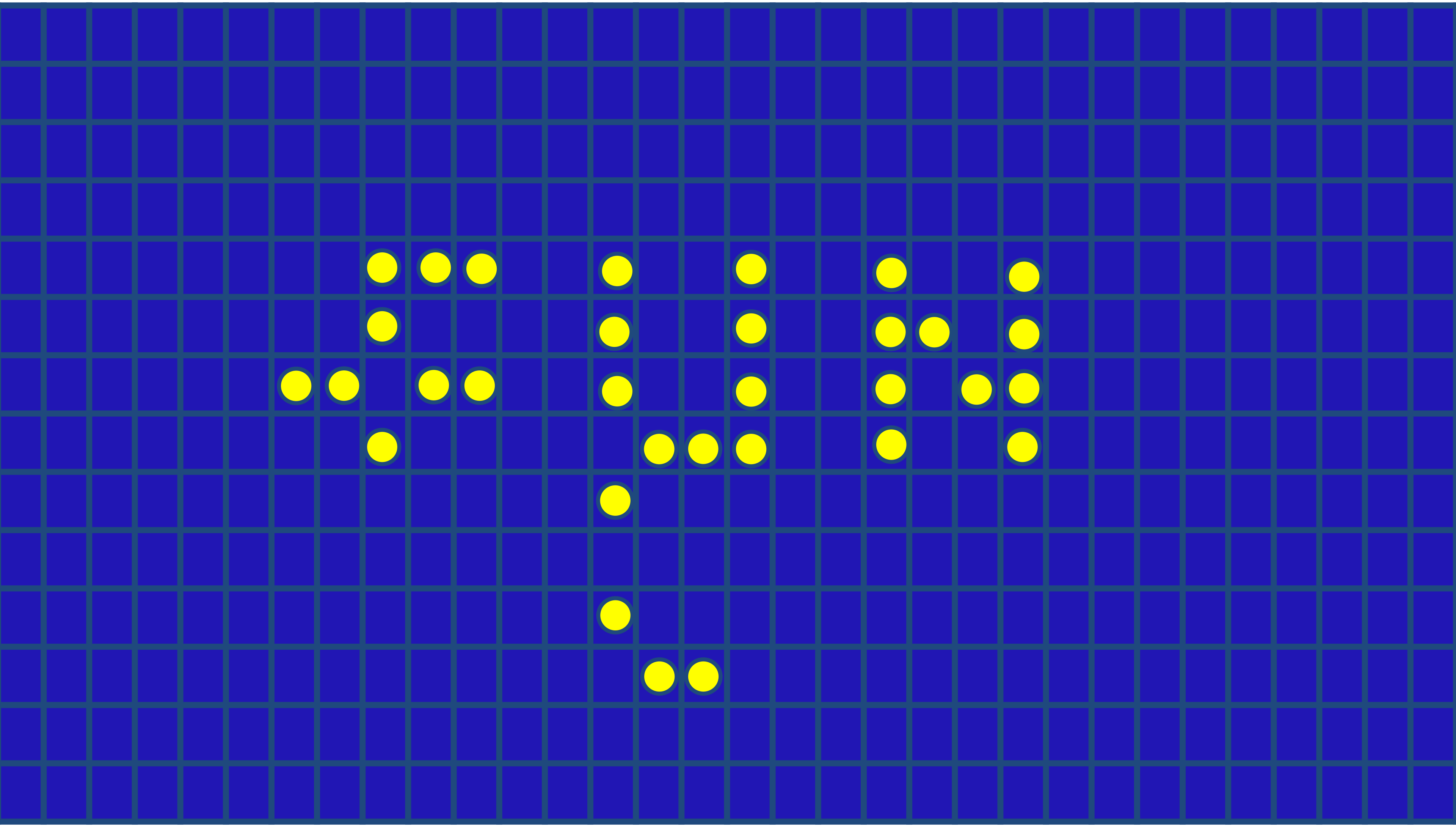
Peg Solitaire



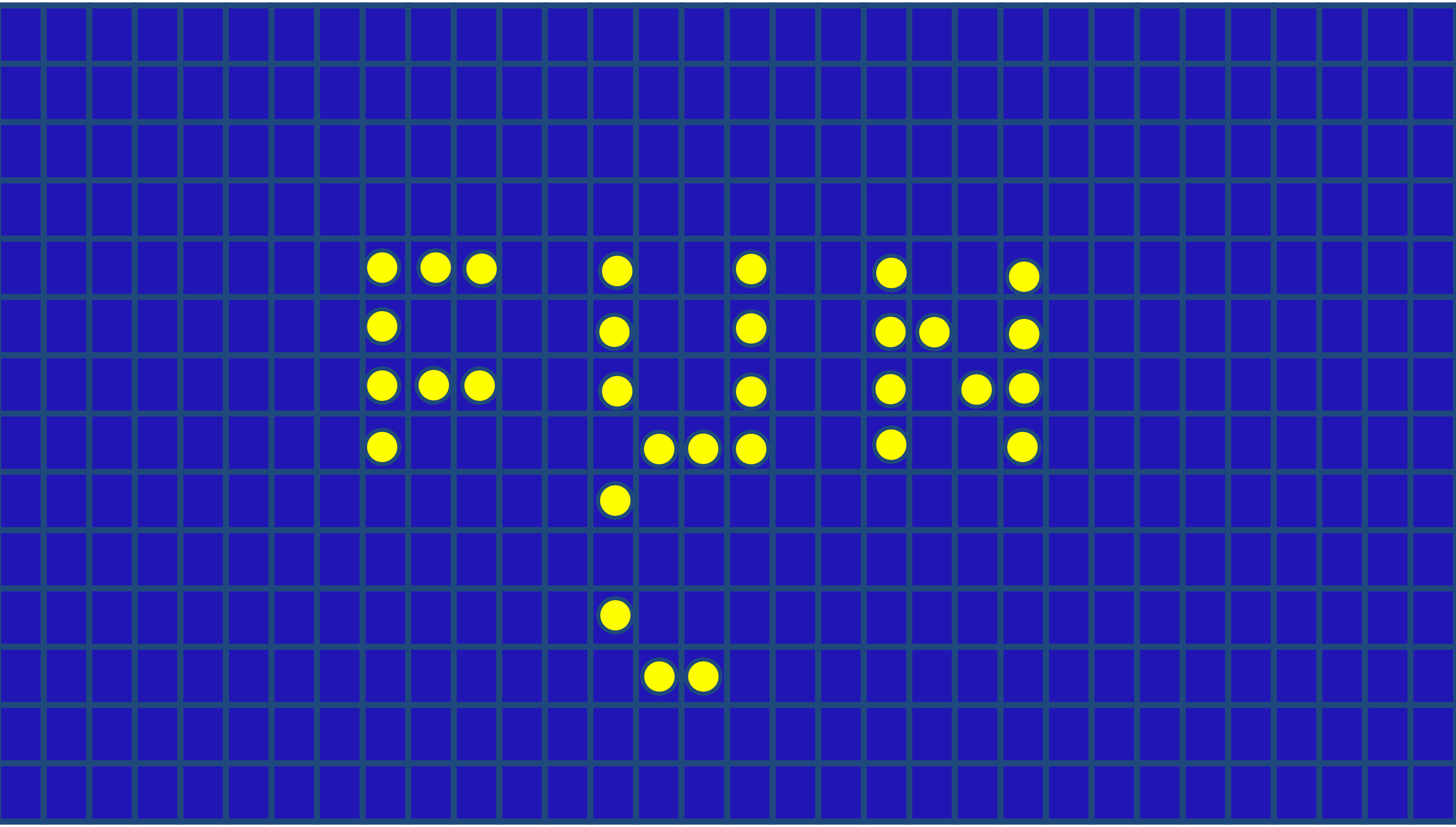
Peg Solitaire



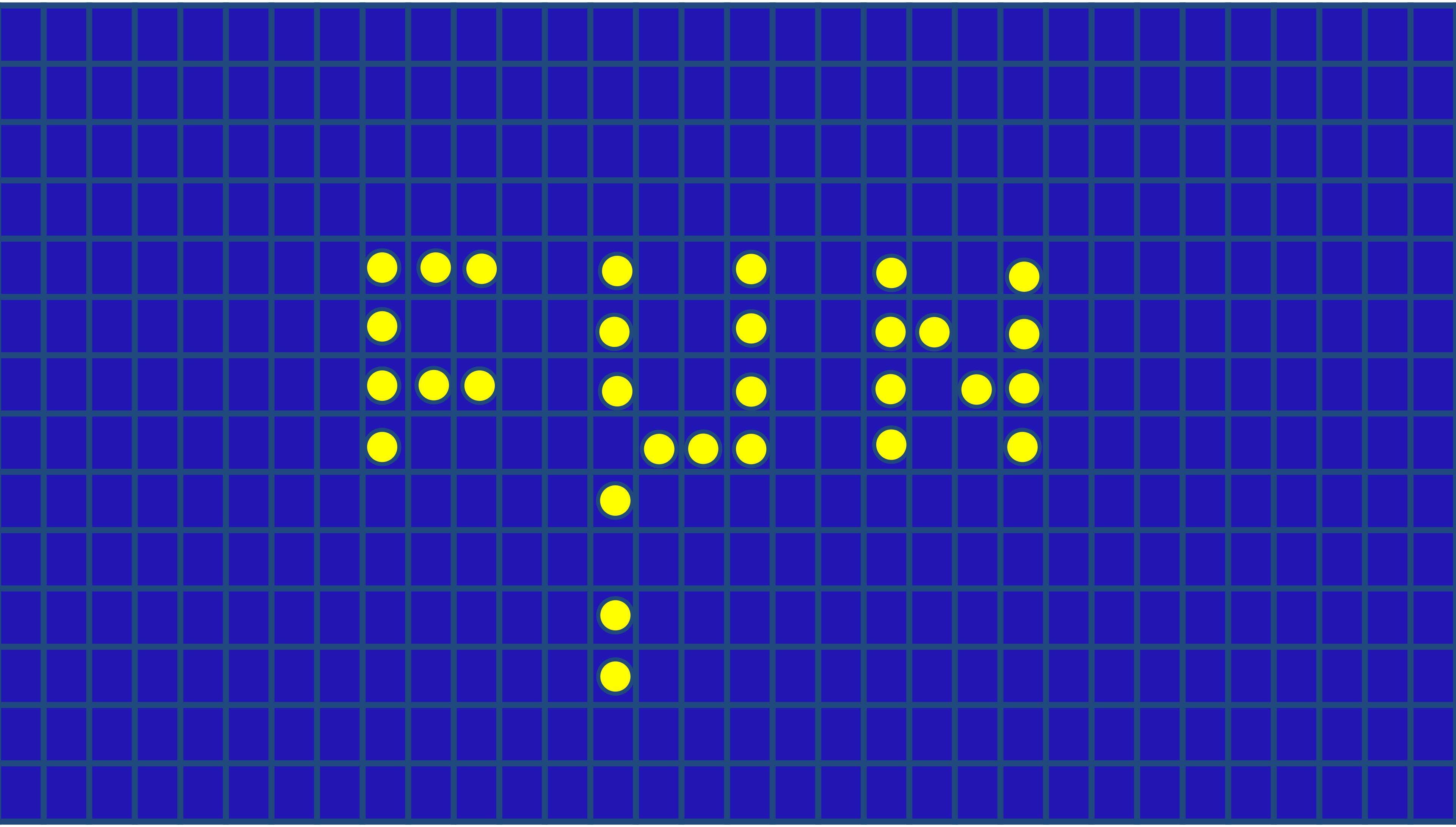
Peg Solitaire



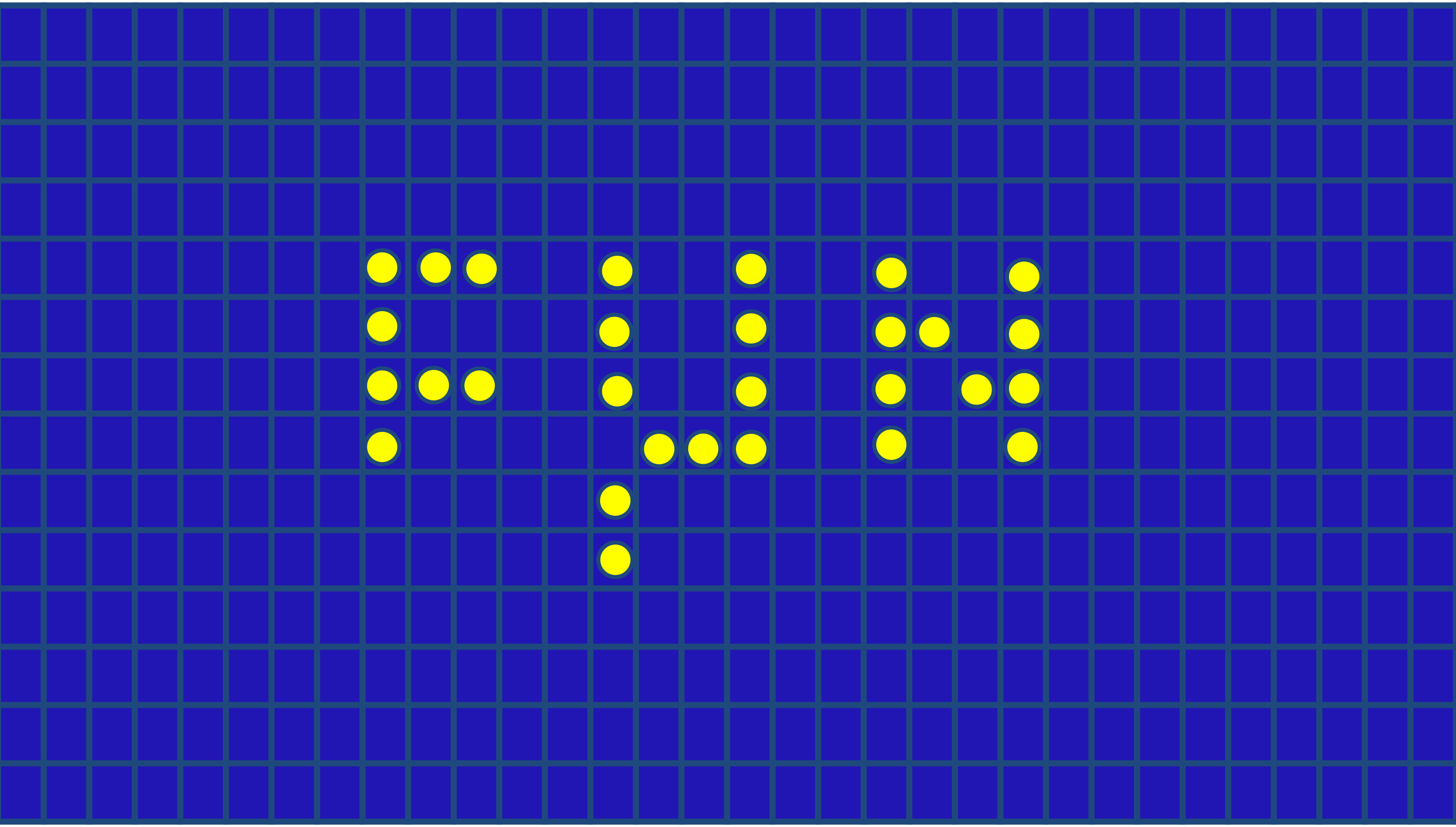
Peg Solitaire



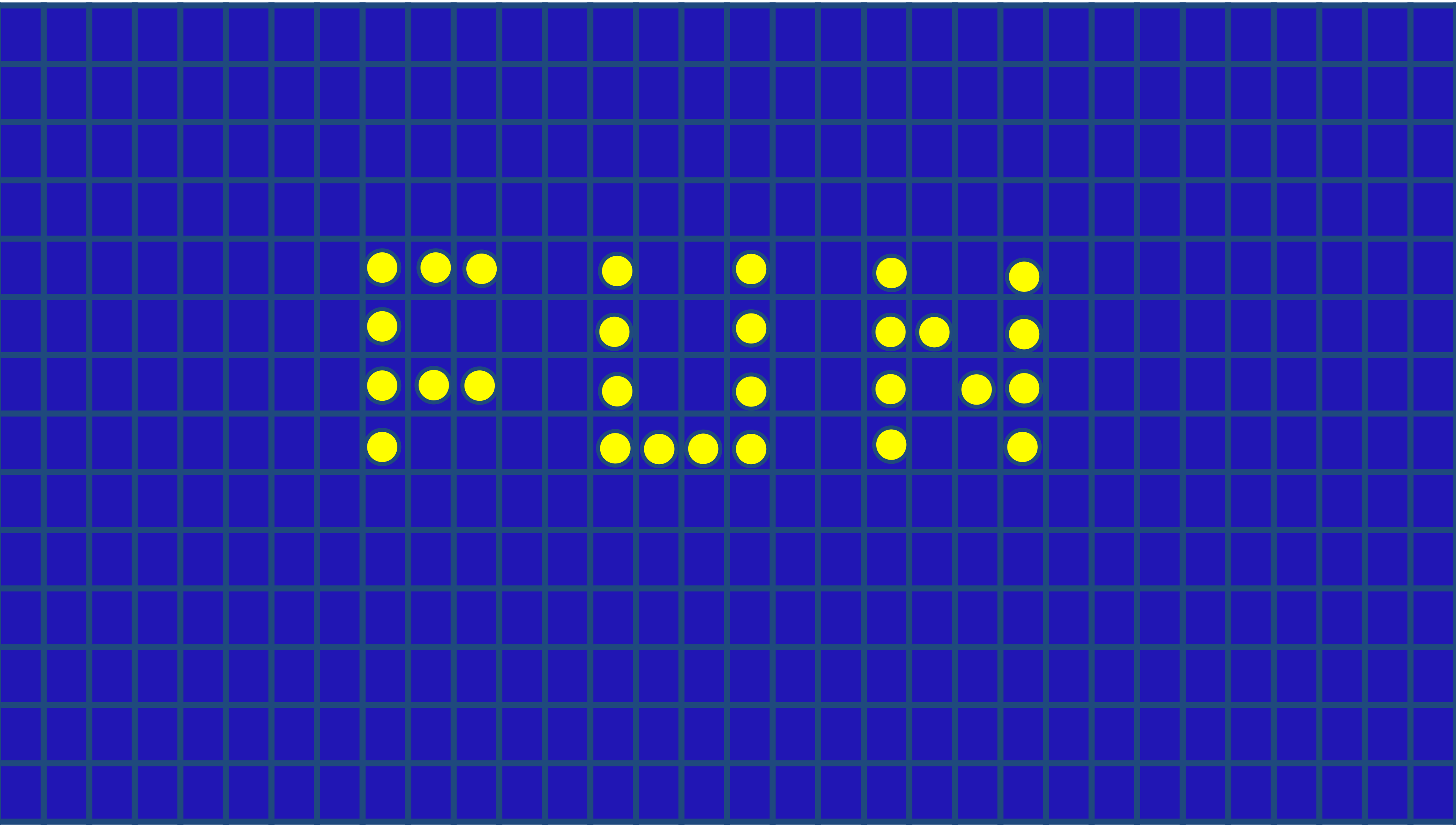
Peg Solitaire



Peg Solitaire



Peg Solitaire



goal: find a sequence of moves from an initial configuration to a target one

Peg Solitaire

Not so very long ago there became widespread an excellent kind of game, called Solitaire, where I play on my own, but as if with a friend as witness and referee to see that I play correctly. A board is filled with stones set in holes, which are to be removed in turn, but none (except the first, which may be chosen for removal at will) can be removed unless you are able to jump another stone across it into an adjacent empty place, when it is captured as in Draughts. He who removes all the stones right to the end according to this rule, wins; […]. This game can more elegantly be played backwards […]Thus we can either fill the board, or, what would be more clever, shape a predetermined figure from the stones; perhaps a triangle, a quadrilateral, an octagon, or some other, if this be possible; but such a task is by no means always possible: and this itself would be a valuable art, to foresee what can be achieved; and to have some way, particularly geometrical, of determining this.

Gottfried Wilhelm von Leibniz (1646-1716)

Not so very long ago there became widespread an excellent kind of game, called Solit… where I play on my own, but as if with a fri… itness and referee to see that I play co… is filled with stones set in holes … removed in turn, but non… ch may be chosen for rem… removed unless yo… ther stone across it in… place, when it is car… he who removes al… e end according to this r… ne can more eleg… ards […]Thus we can e… , what would be more cle… etermined figure from the stone… triangle, a quadrilateral, an octagon, … me other, if this be possible; but such a task is by no means always possible: and this itself would be a valuable art, to foresee what can be achieved; and to have some way, particularly geometrical, of determining this.

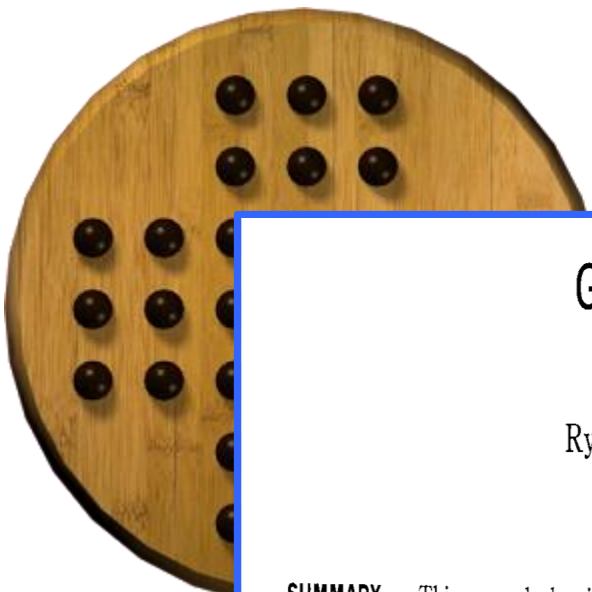
That's fun!

Gottfried Wilhelm von Leibniz (1646-1716)

Peg Solitaire

NP-complete to
decide whether the
board can be cleared
[Uehara&Iwata,90]

Peg Solitaire



Generalized Hi-Q is NP-Complete

Ryuhei UEHARA† *and* Shigeki IWATA††, *Members*

SUMMARY This paper deals with a popular puzzle known as Hi-Q. The puzzle is generalized: the board is extended to the size $n \times n$, an initial position of the puzzle is given, and a place is given on which only one token is finally placed. The complexity of the generalized Hi-Q is proved NP-complete.

1. Introduction

In general, combinatorial puzzles and games are hard to analyze, since we have to cope with enormous number of positions of the board. It is one of the main themes in artificial intelligence to solve these problems by heuristic methods. It is important at the same time to

initial position is given on the extended board of size $n \times n$, and a goal is also given on which only one token will finally be placed. We show that the problem to determine whether there is an answer for a given generalized Hi-Q is NP-complete. The NP-hardness can be obtained by reducing from a variation of the hamiltonian cycle problem.

2. Complexity Result

We extend the size of the board of Hi-Q to $n \times n$, and assume further that both a position and a goal of the

Reduction from:

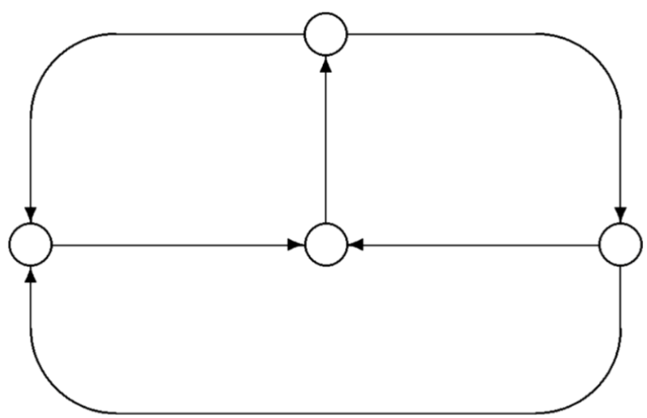
Hamiltonian Cycle in planar directed graphs with degree 3

it can be drawn in a plane in such a way that no edges cross each other

input: a directed planar graph G where each vertex has degree exactly 3

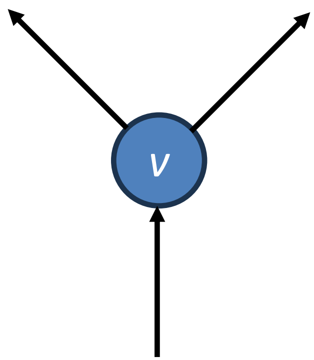
goal: does G has a directed Hamiltonian cycle?

G

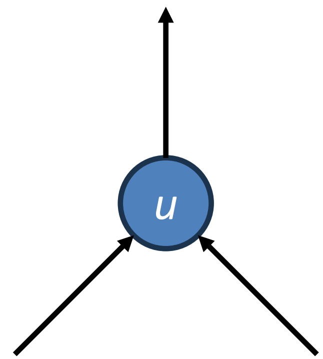


It is NP-complete

Only 2 types of vertices:

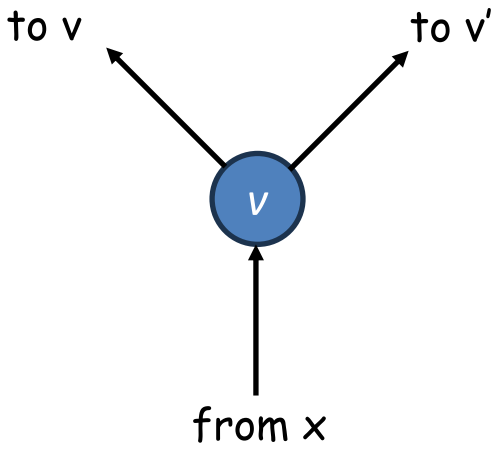


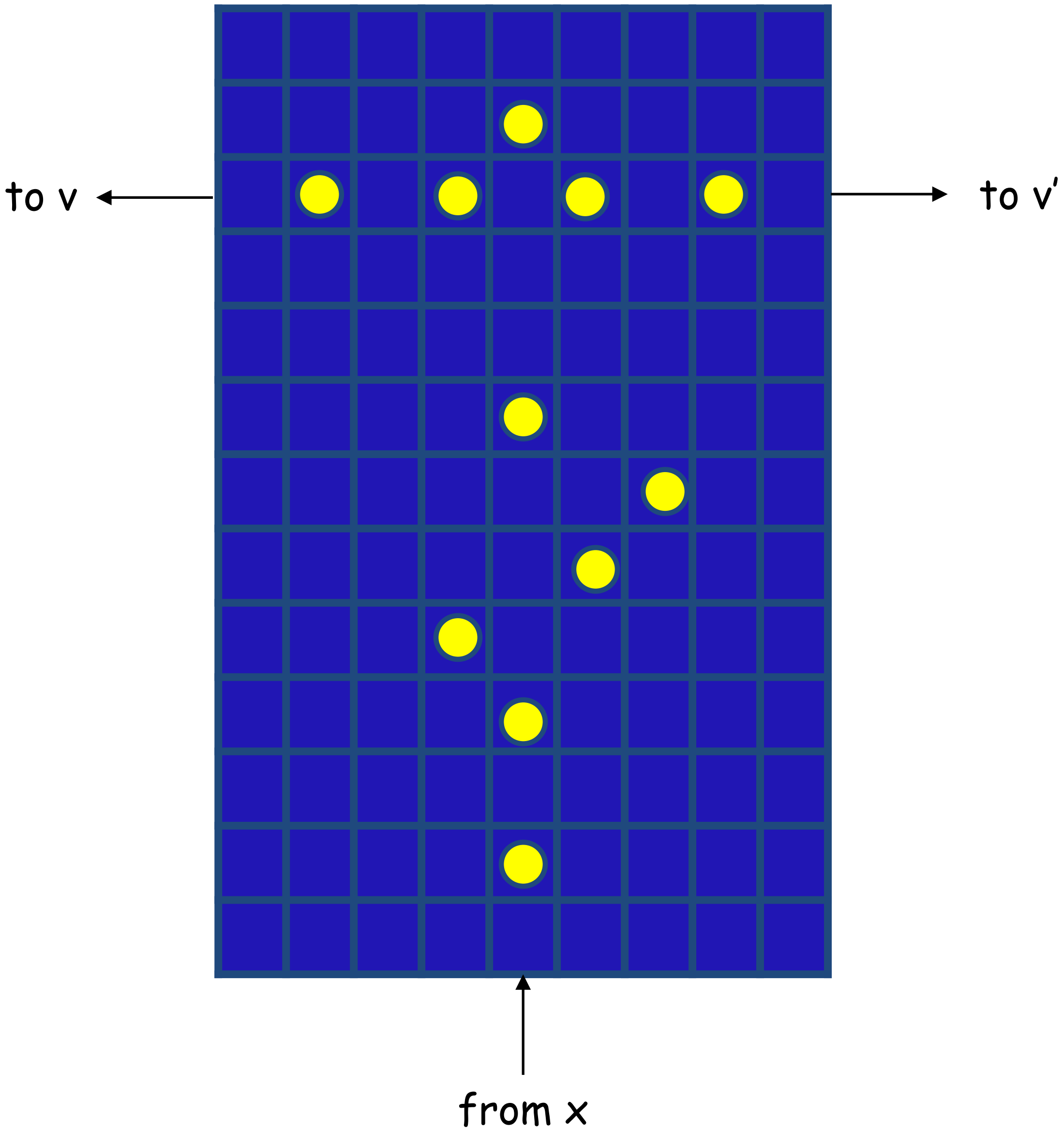
1-in 2-out
degree vertex



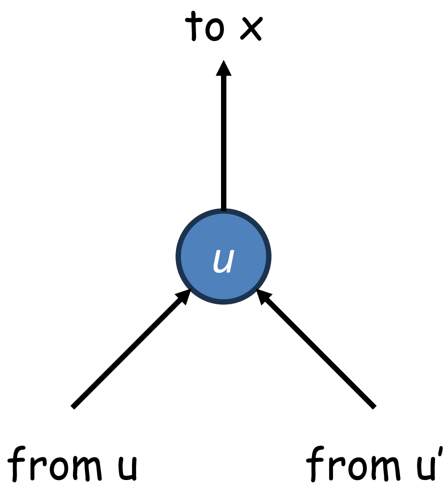
2-in 1-out
degree vertex

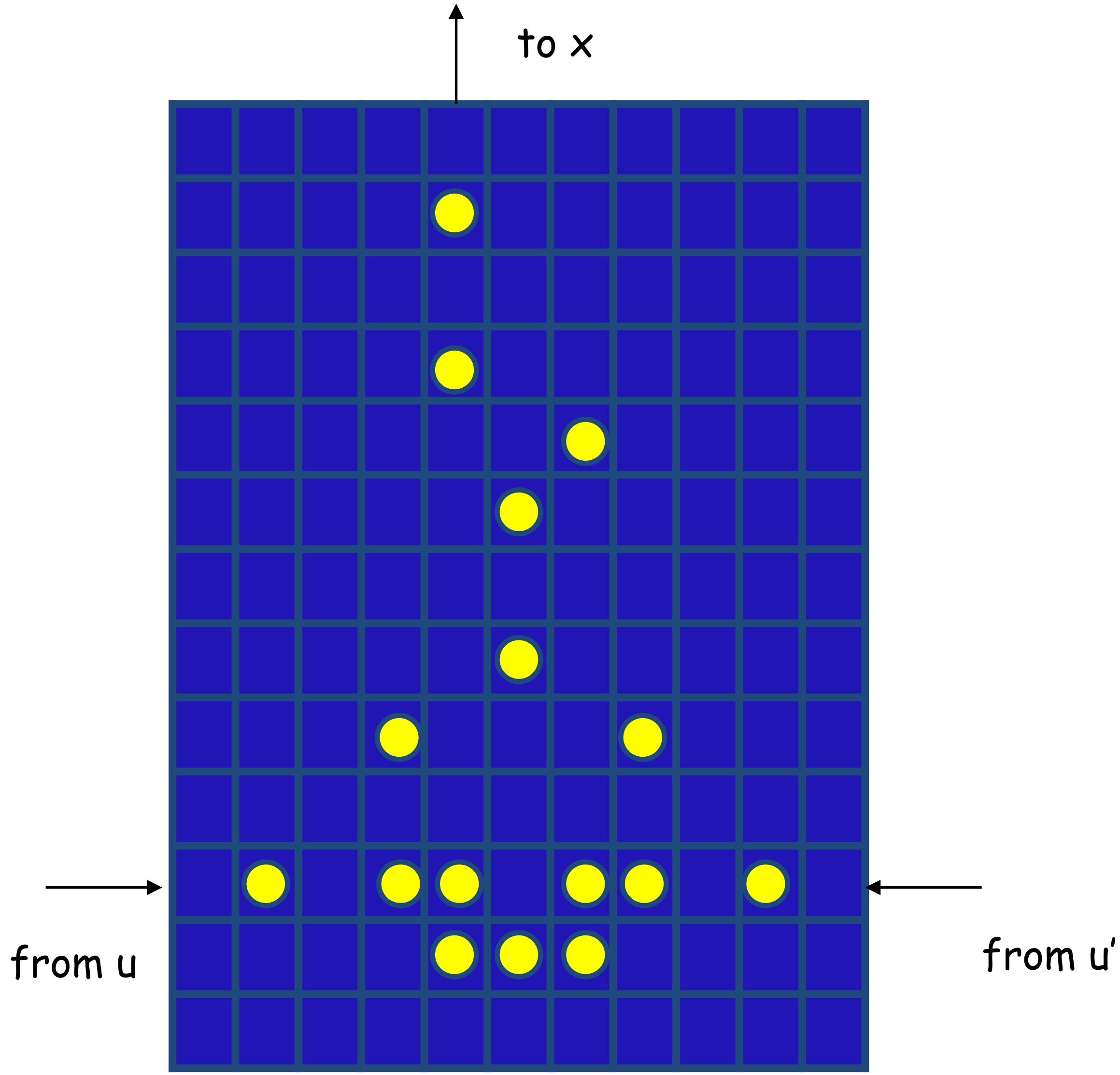
1-in 2-out degree vertex gadget



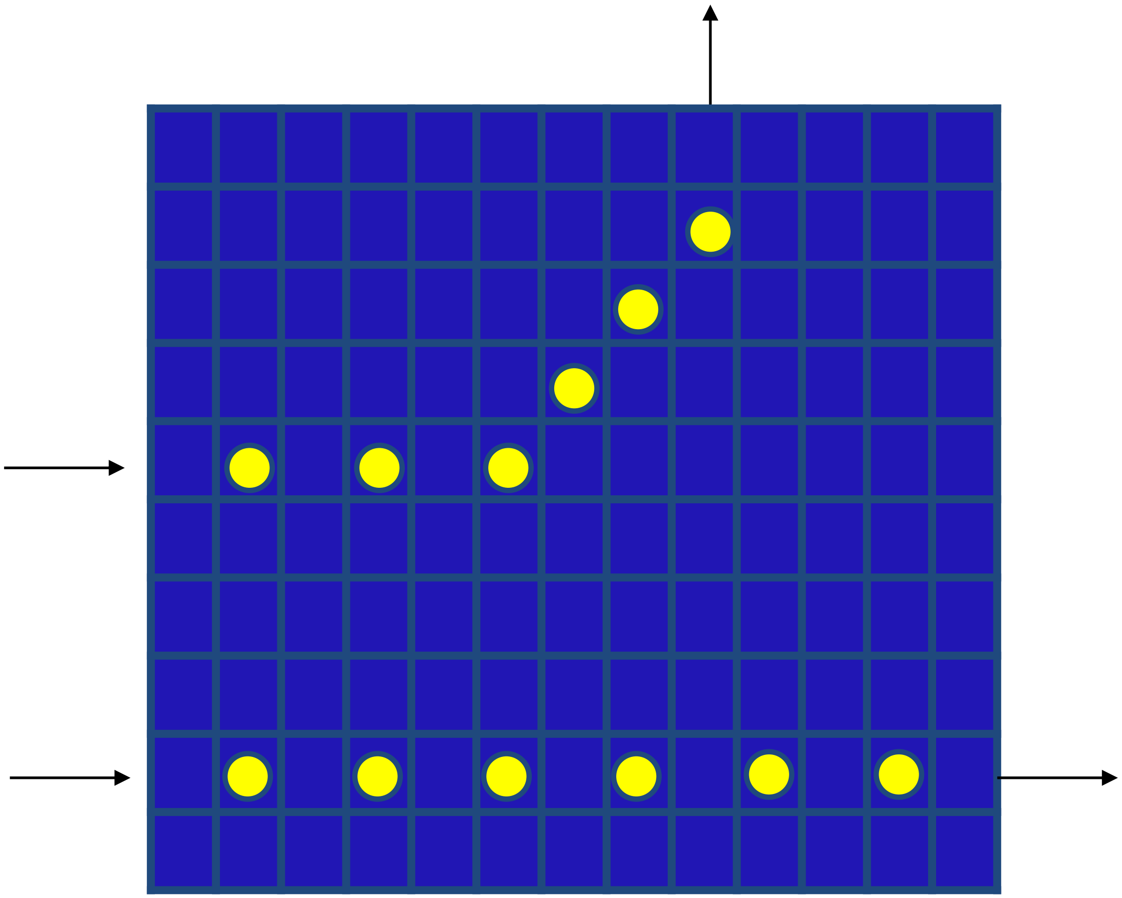


2-in 1-out degree vertex gadget

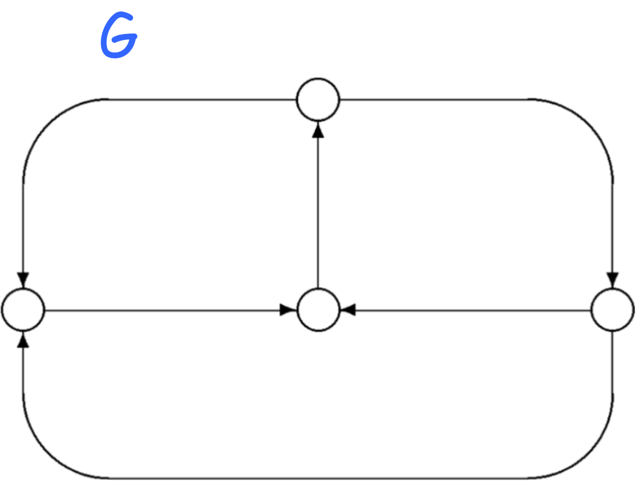


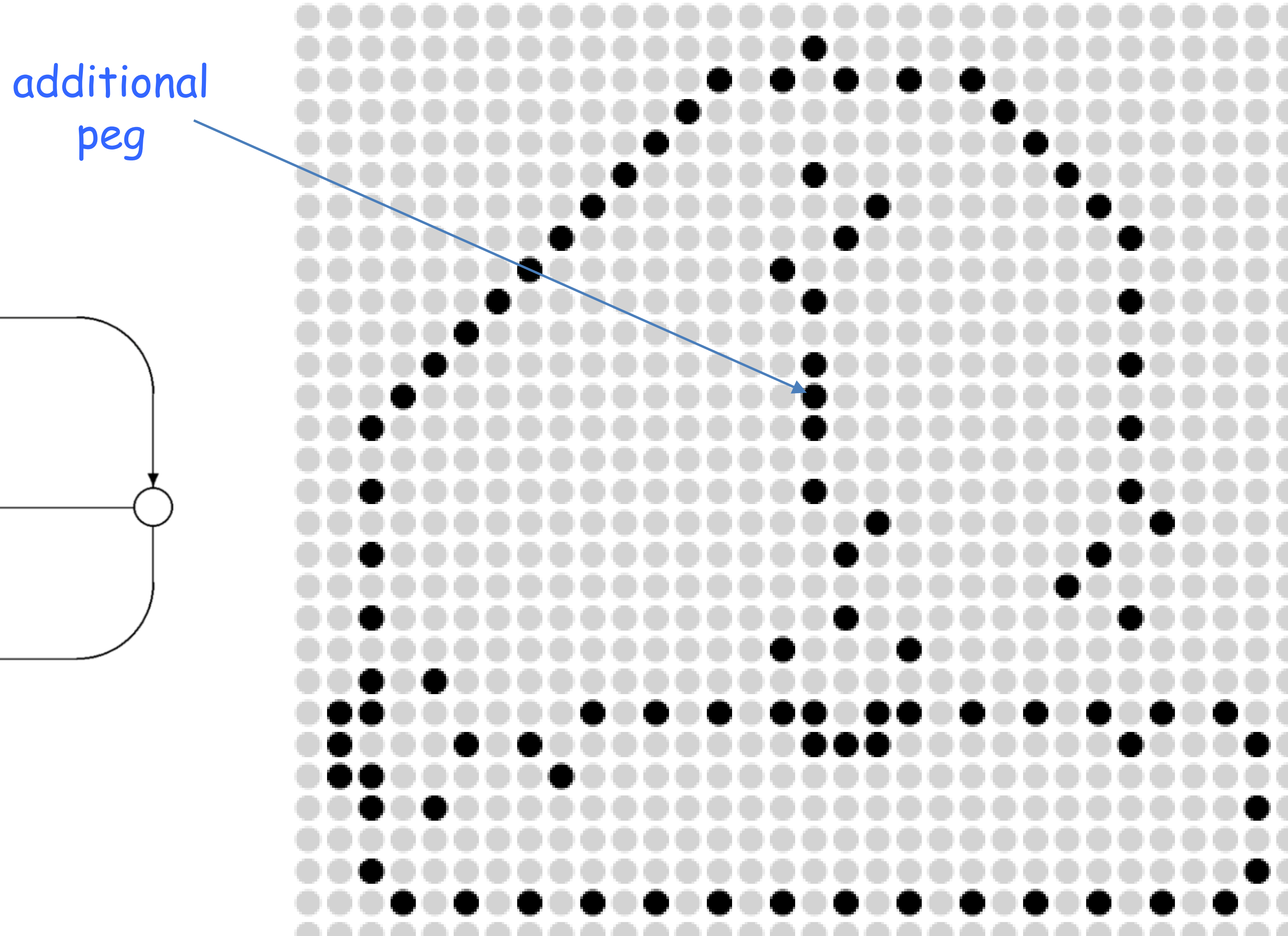


wire/edge gadgets

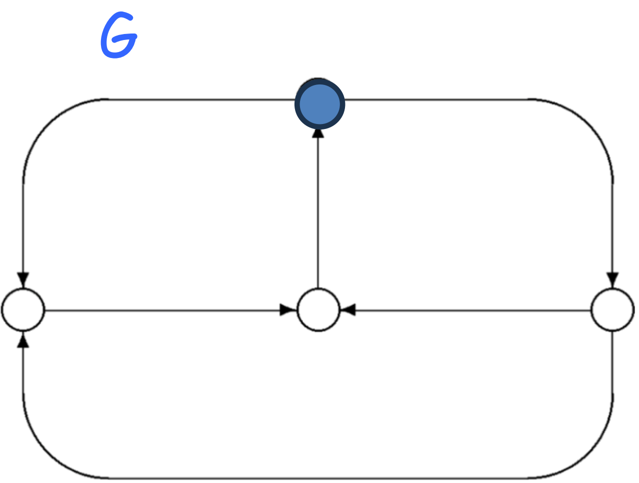


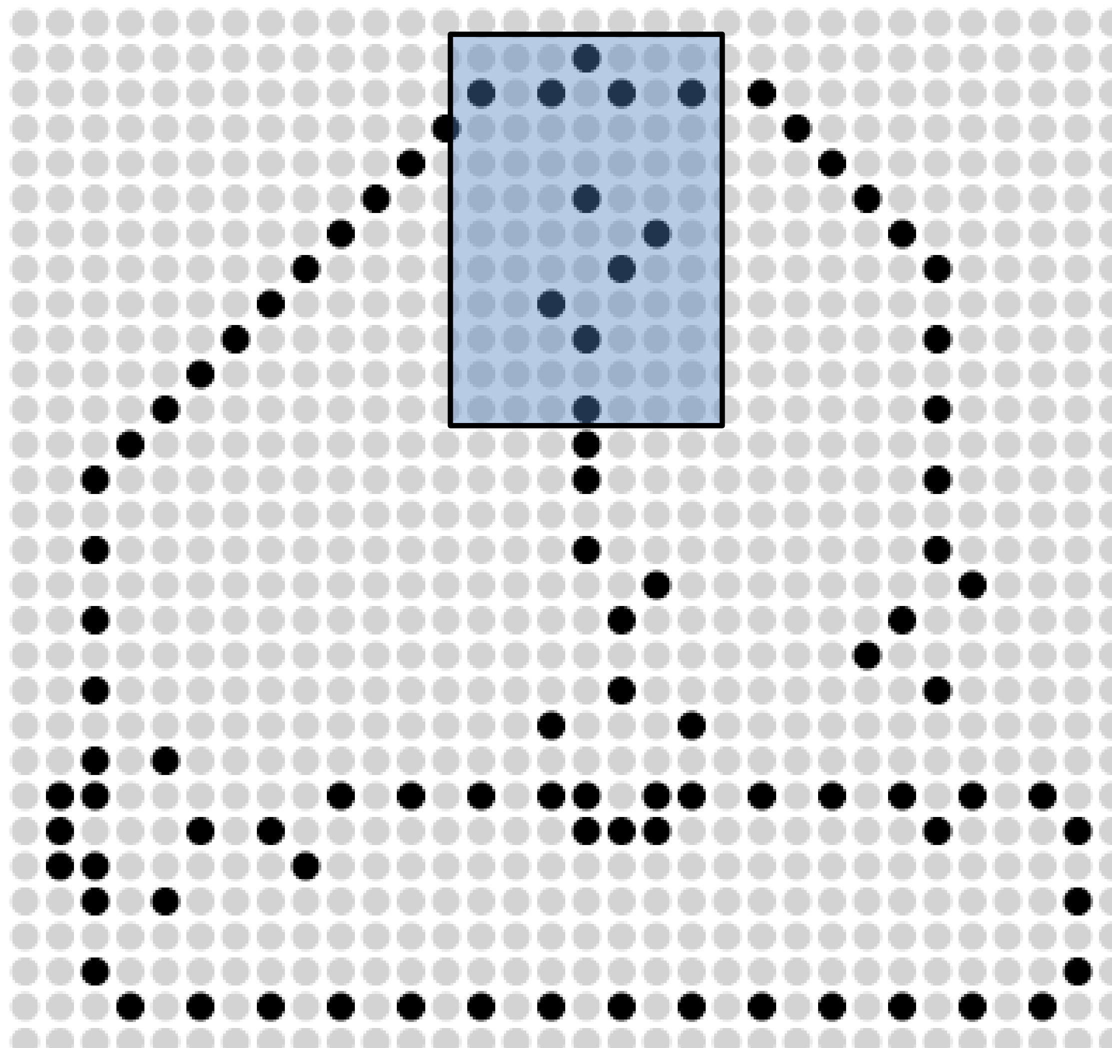
Connecting the gadgets



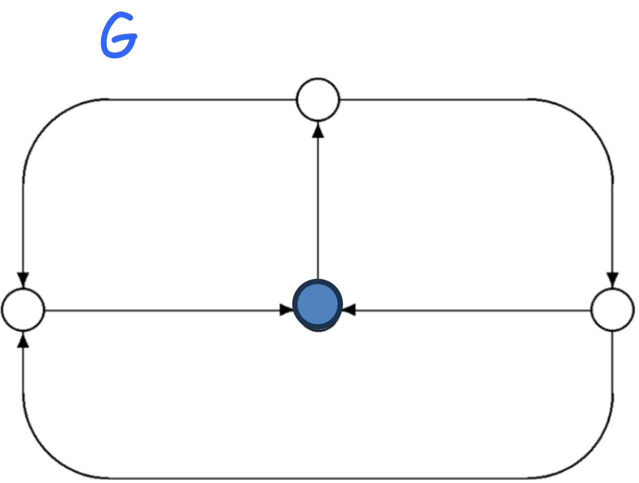


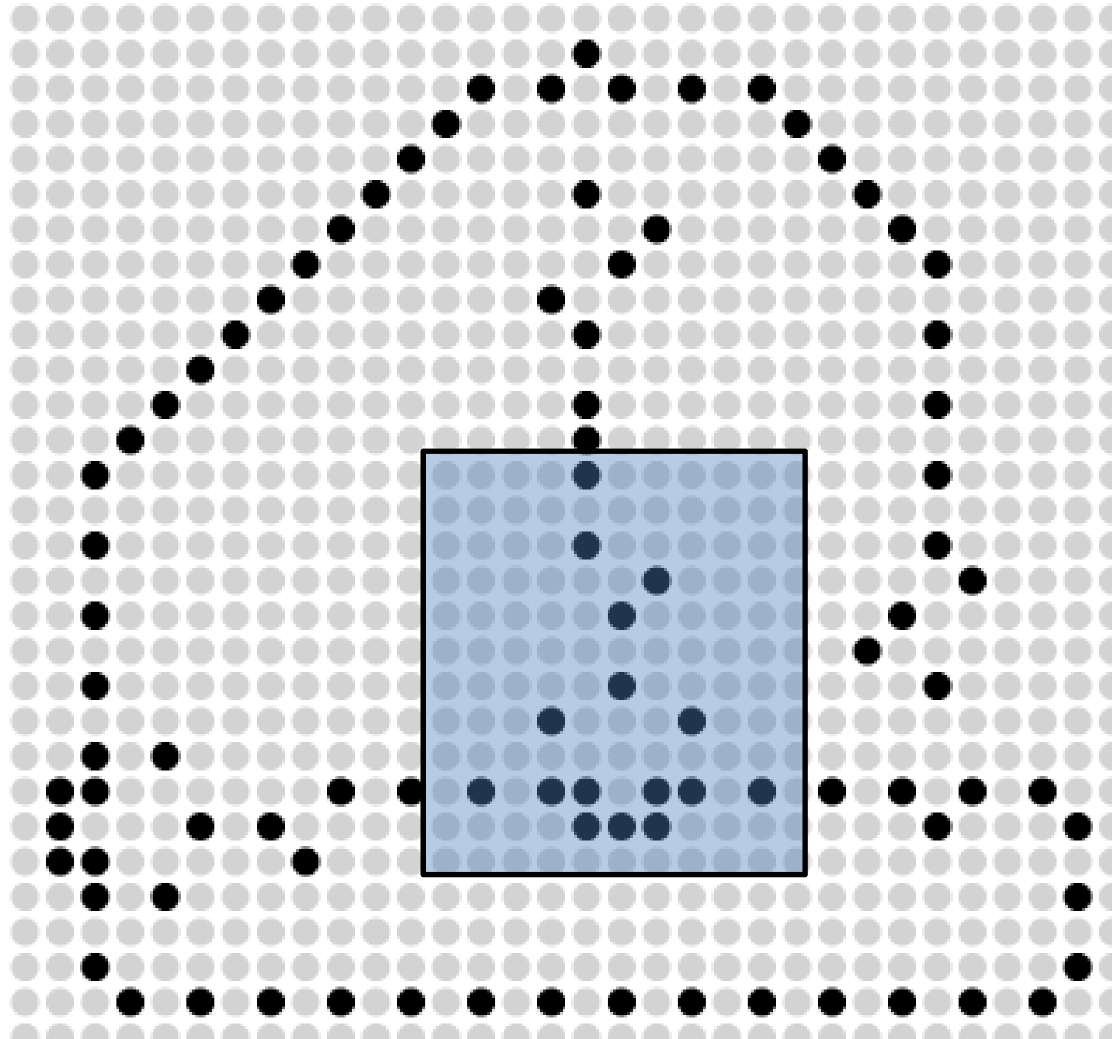
Connecting the gadgets



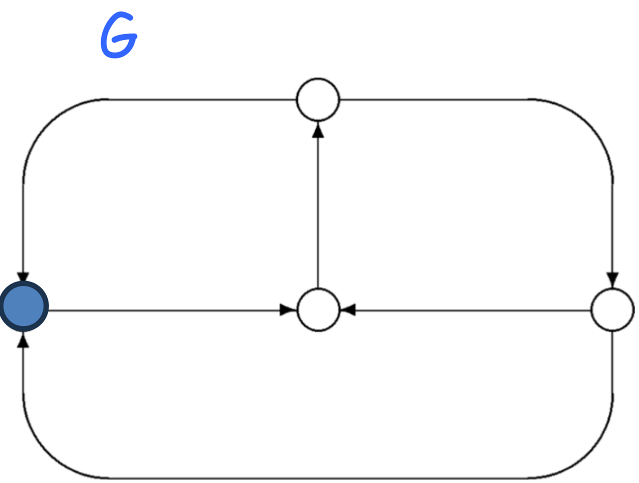


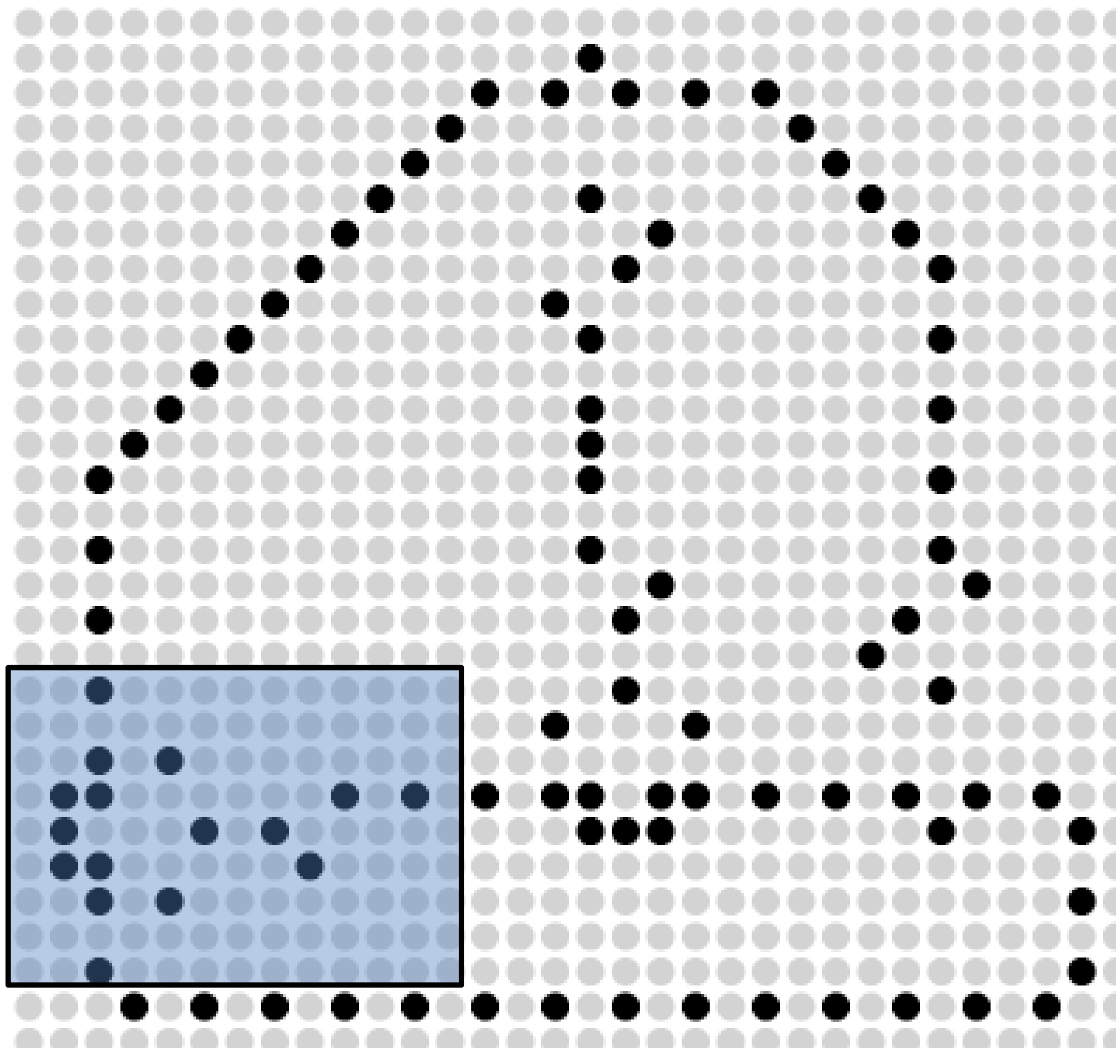
Connecting the gadgets





Connecting the gadgets

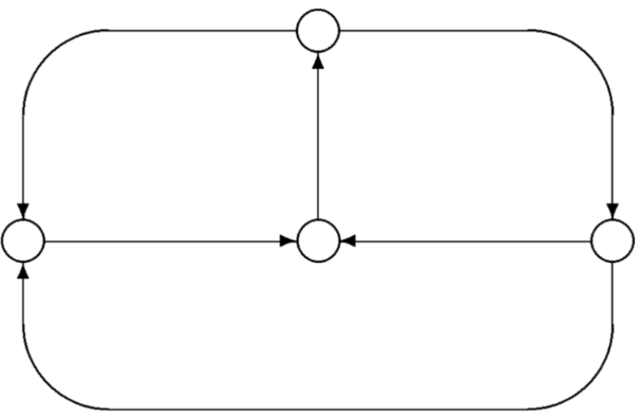


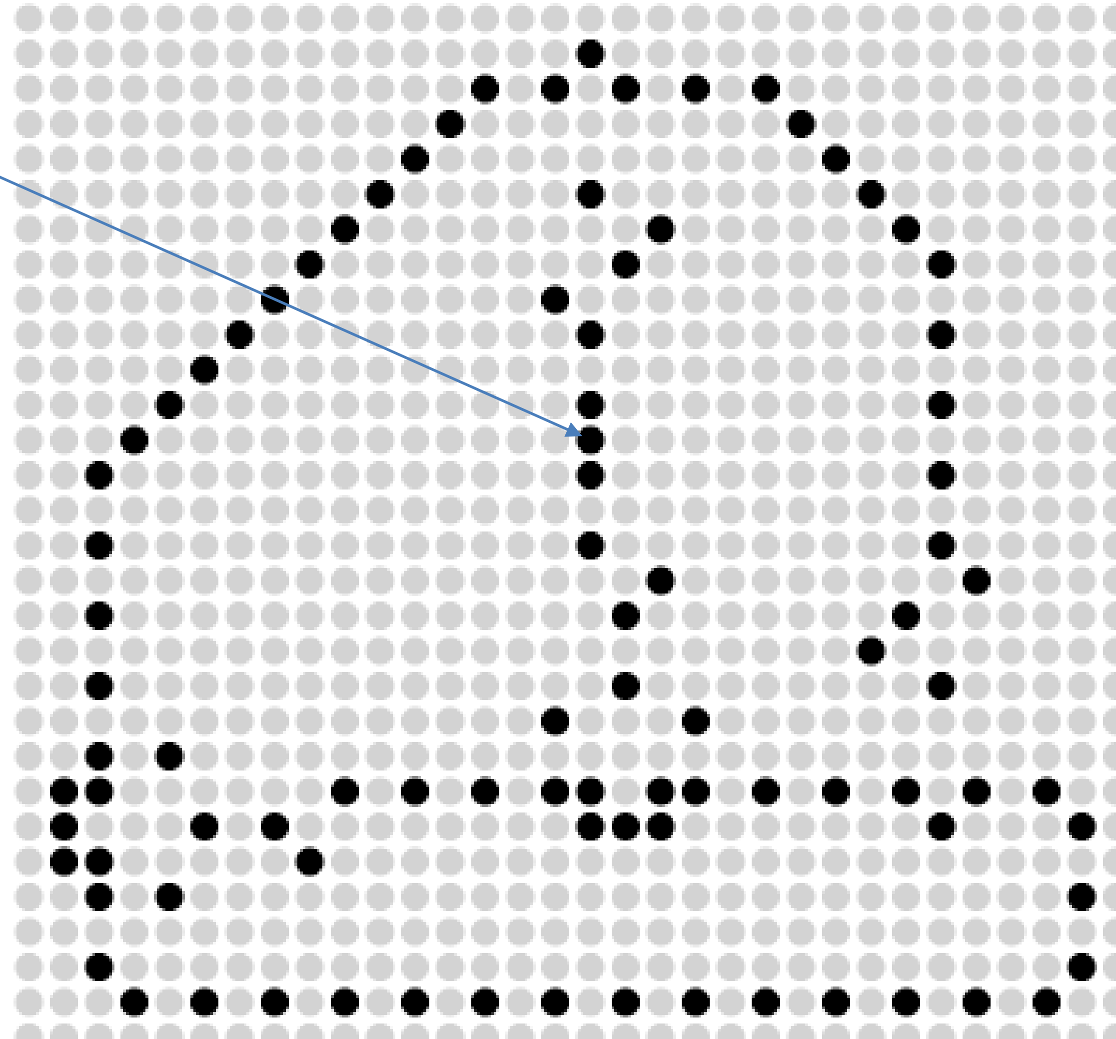


Connecting the gadgets

additional peg

G





claim:

you can clear the board if and only if G has a Hamiltonian cycle claim:

you can clear the board if and only if G has a Hamiltonian cycle

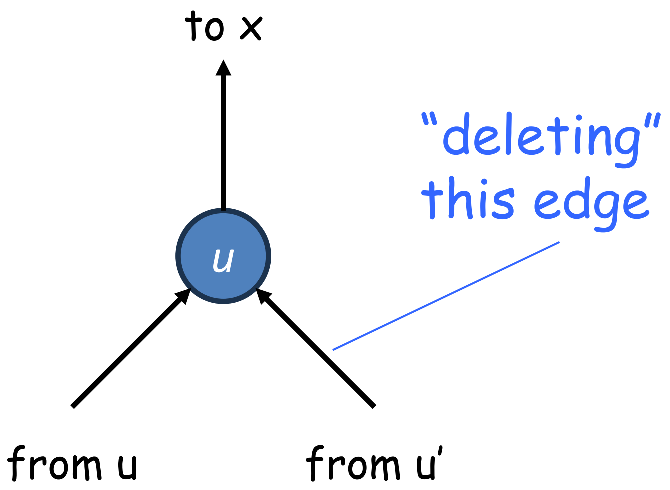
proof

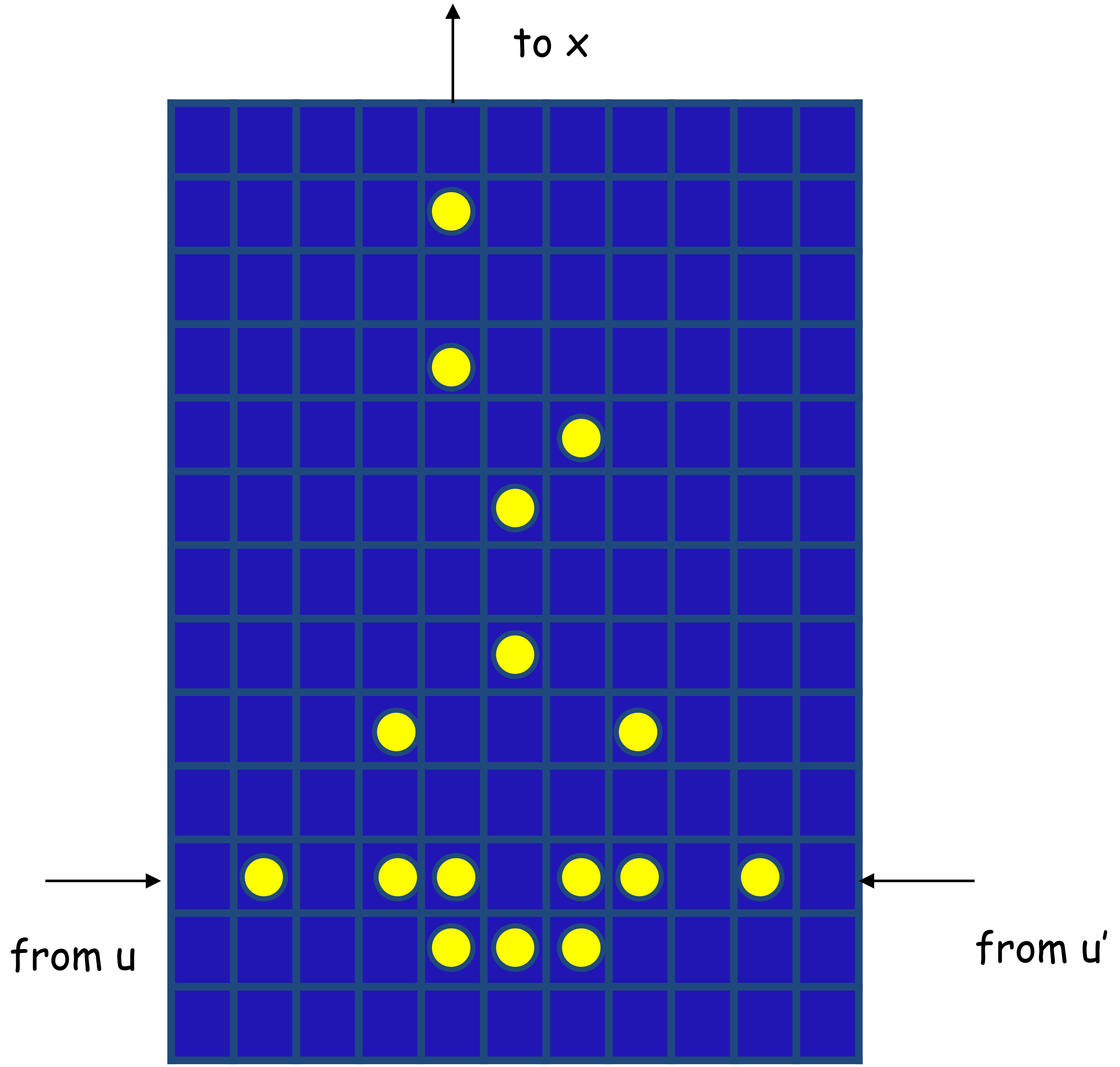
($\Leftarrow$)

if there is a Hamiltonian cycle C:

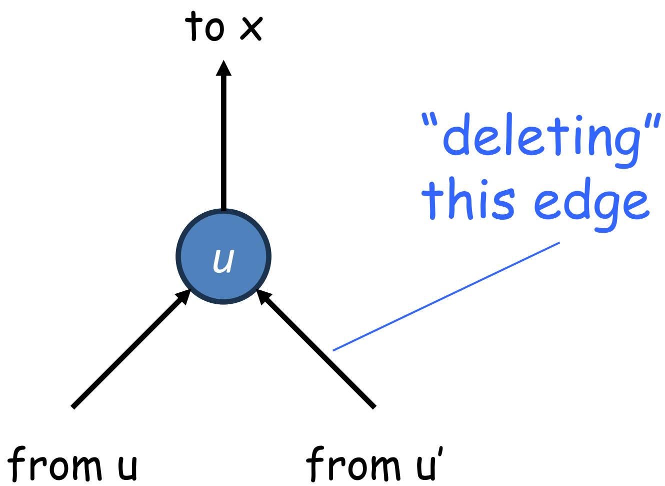
- first delete/clear the edges that do not belong to C
- clear the remaining pegs by going through C

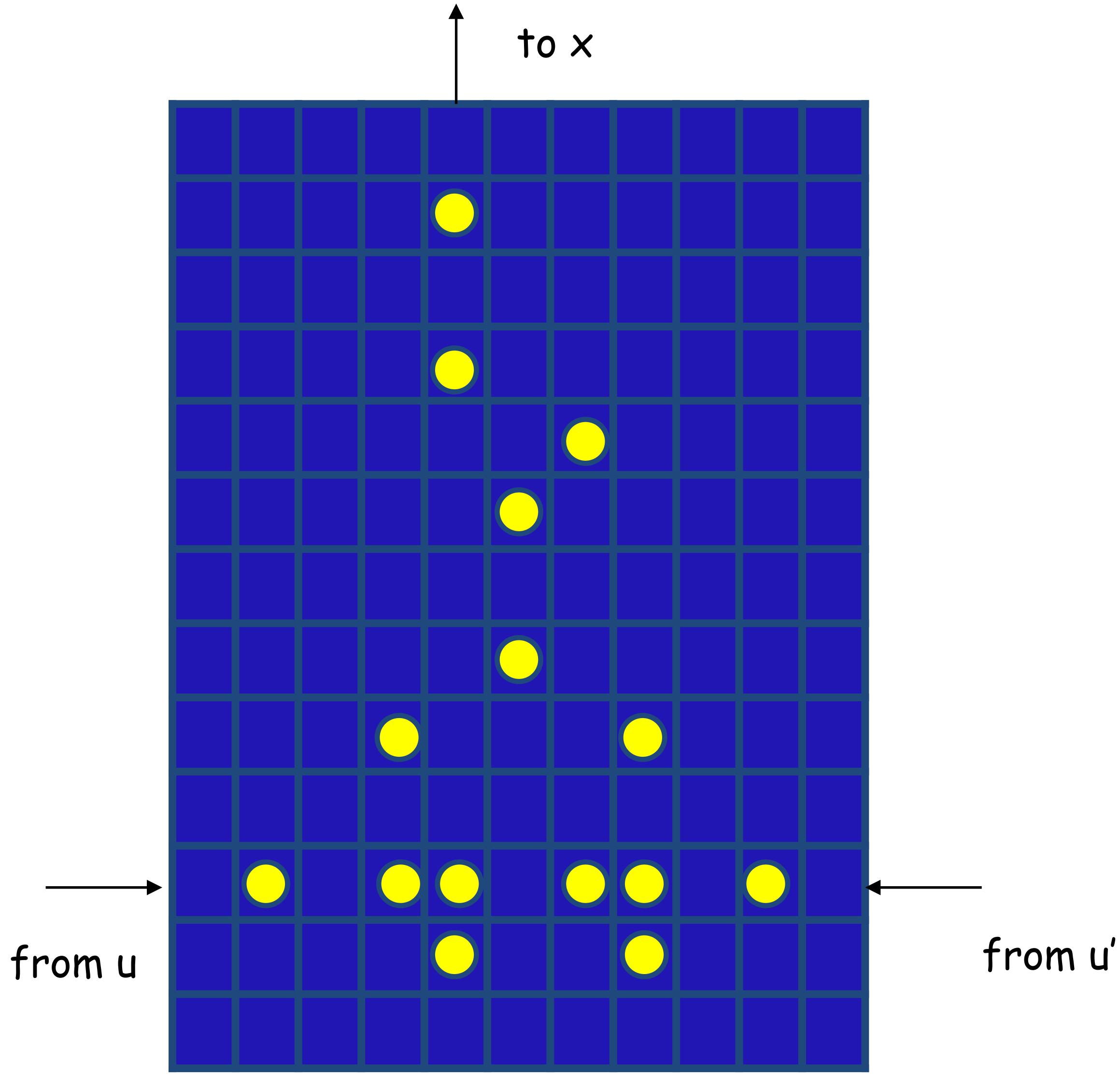
2-in 1-out degree vertex gadget: intended behavior



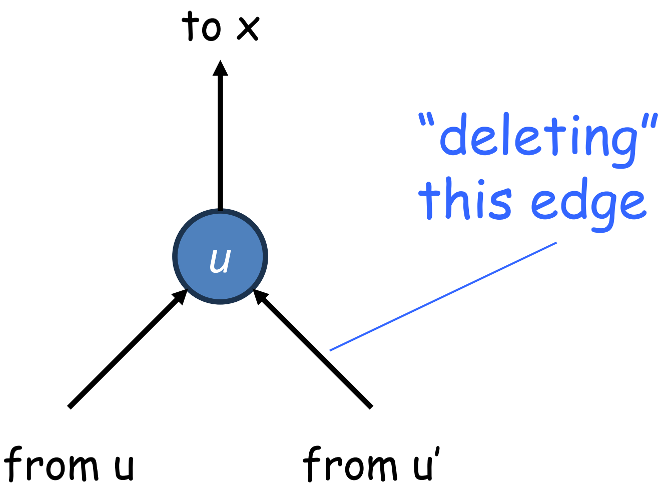


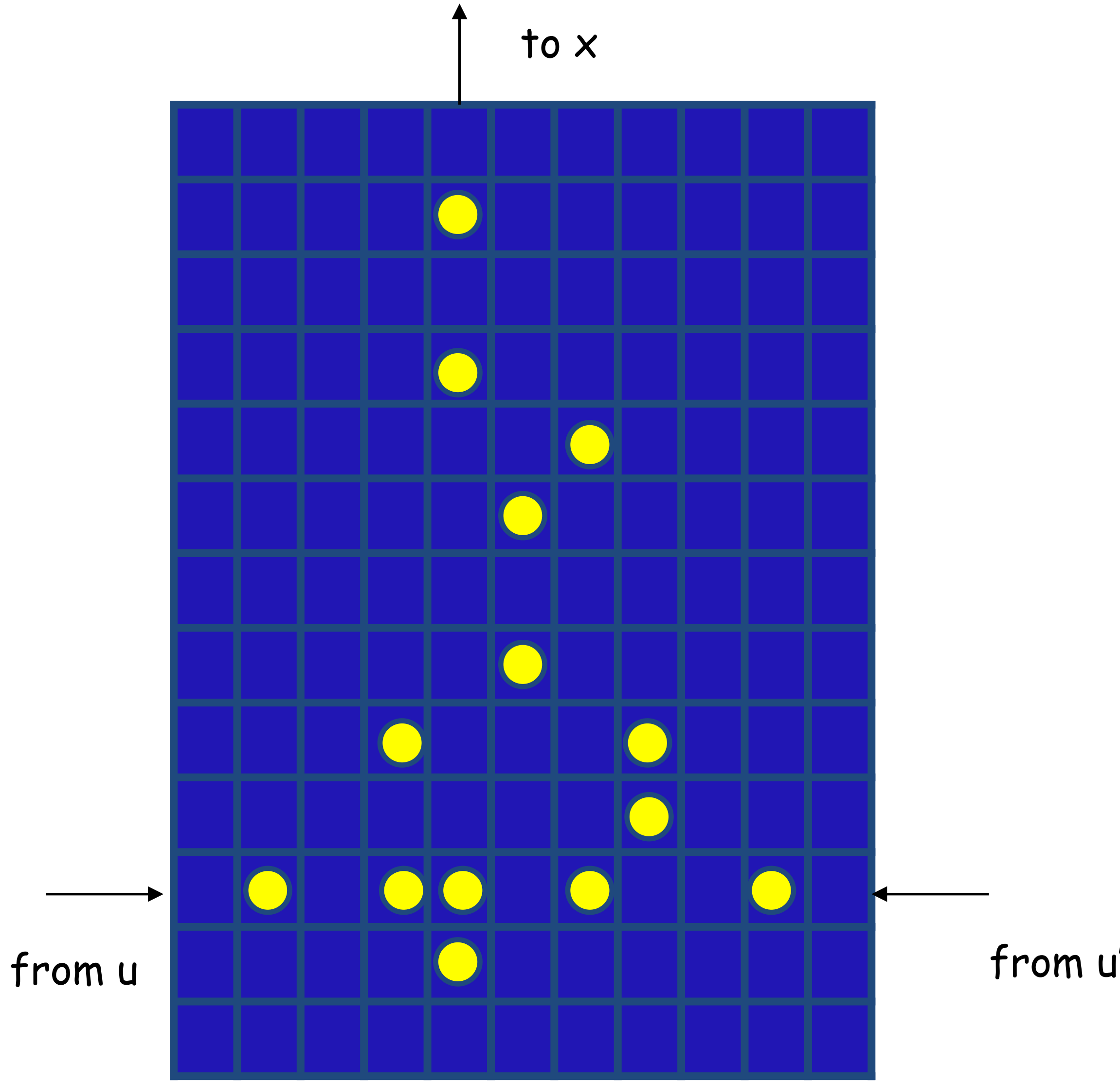
2-in 1-out degree vertex gadget: intended behavior



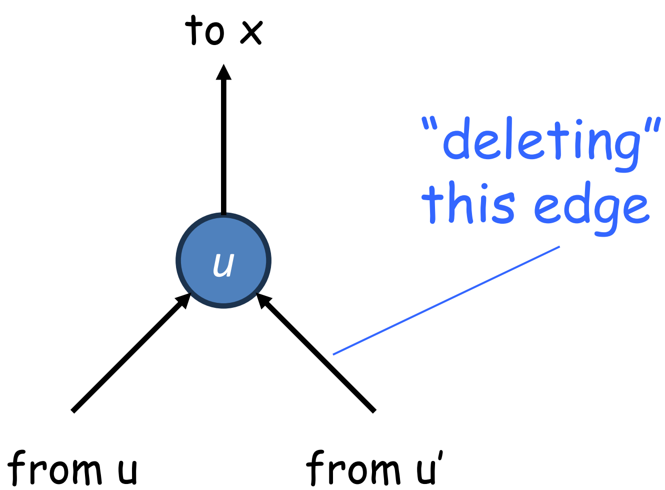


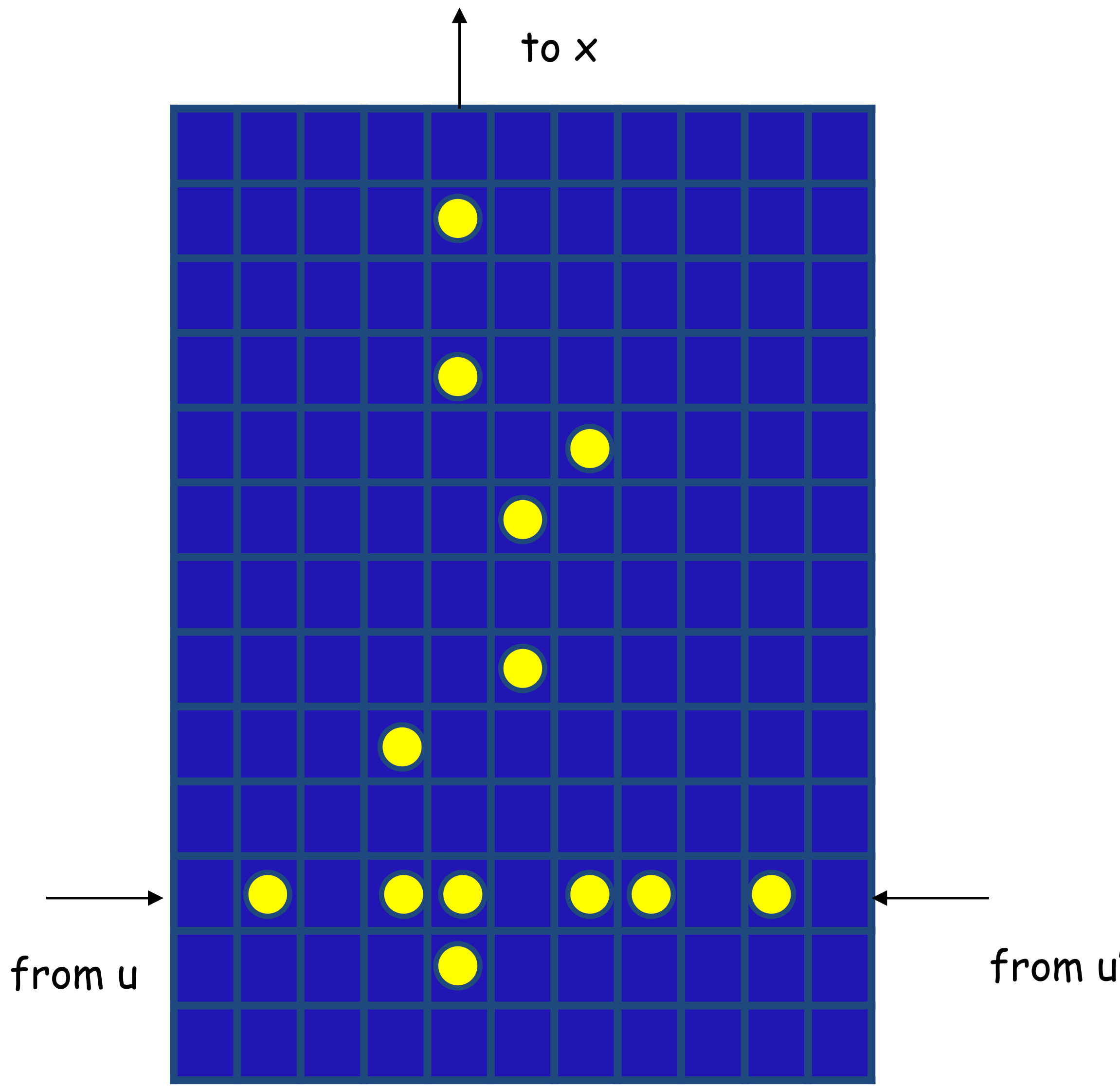
2-in 1-out degree vertex gadget: intended behavior



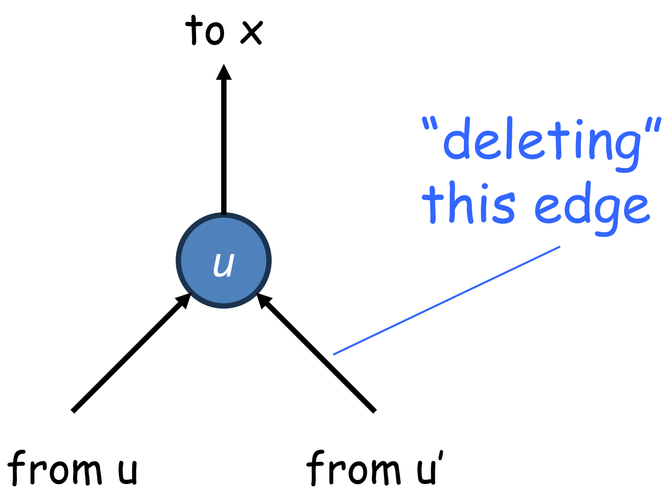


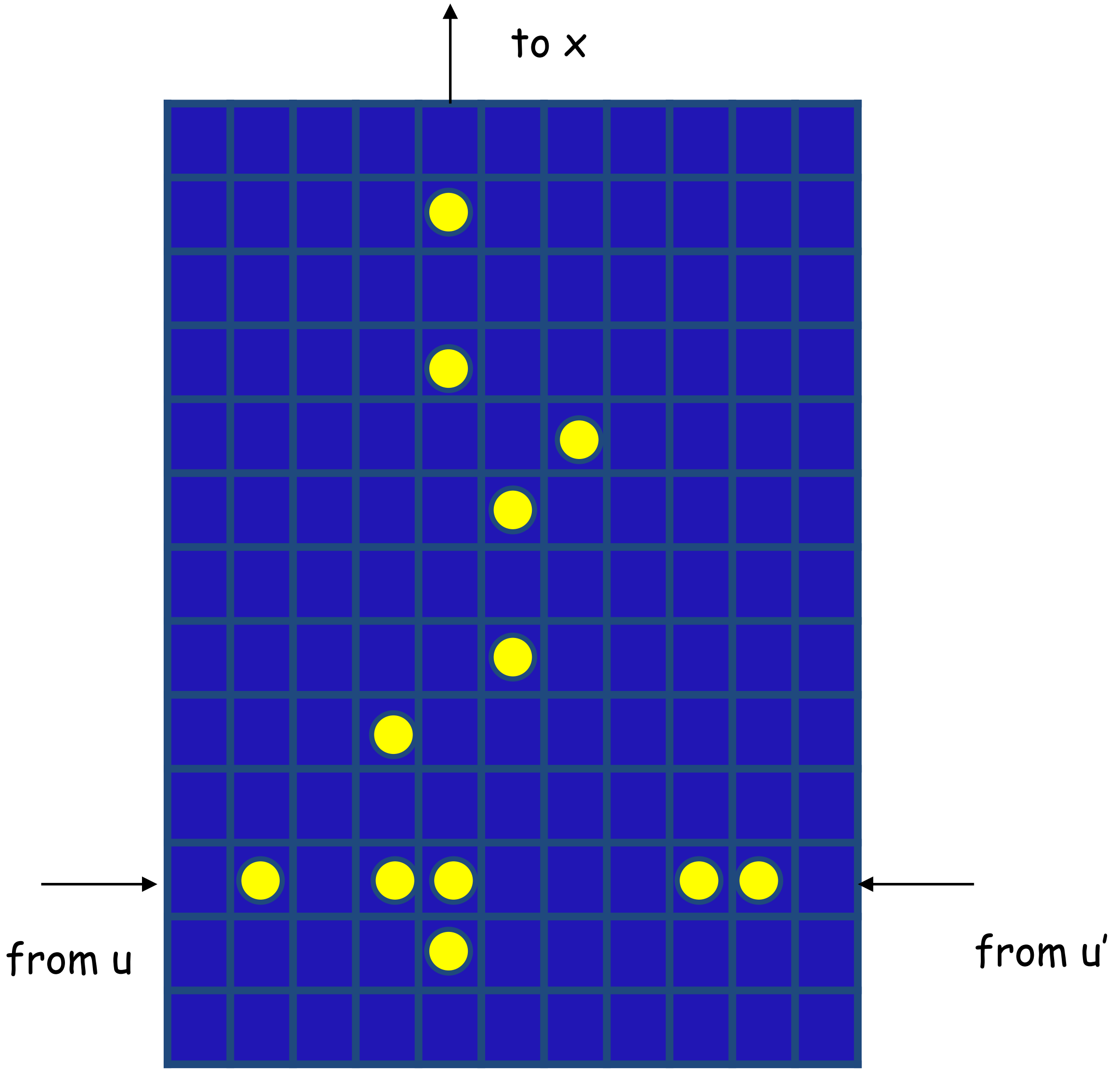
2-in 1-out degree vertex gadget: intended behavior



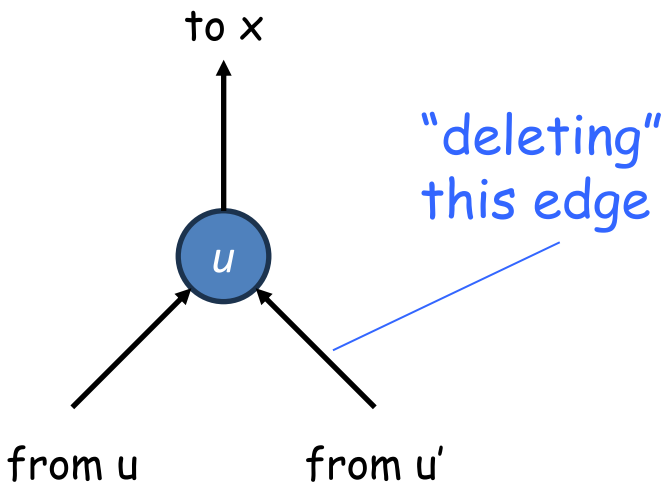


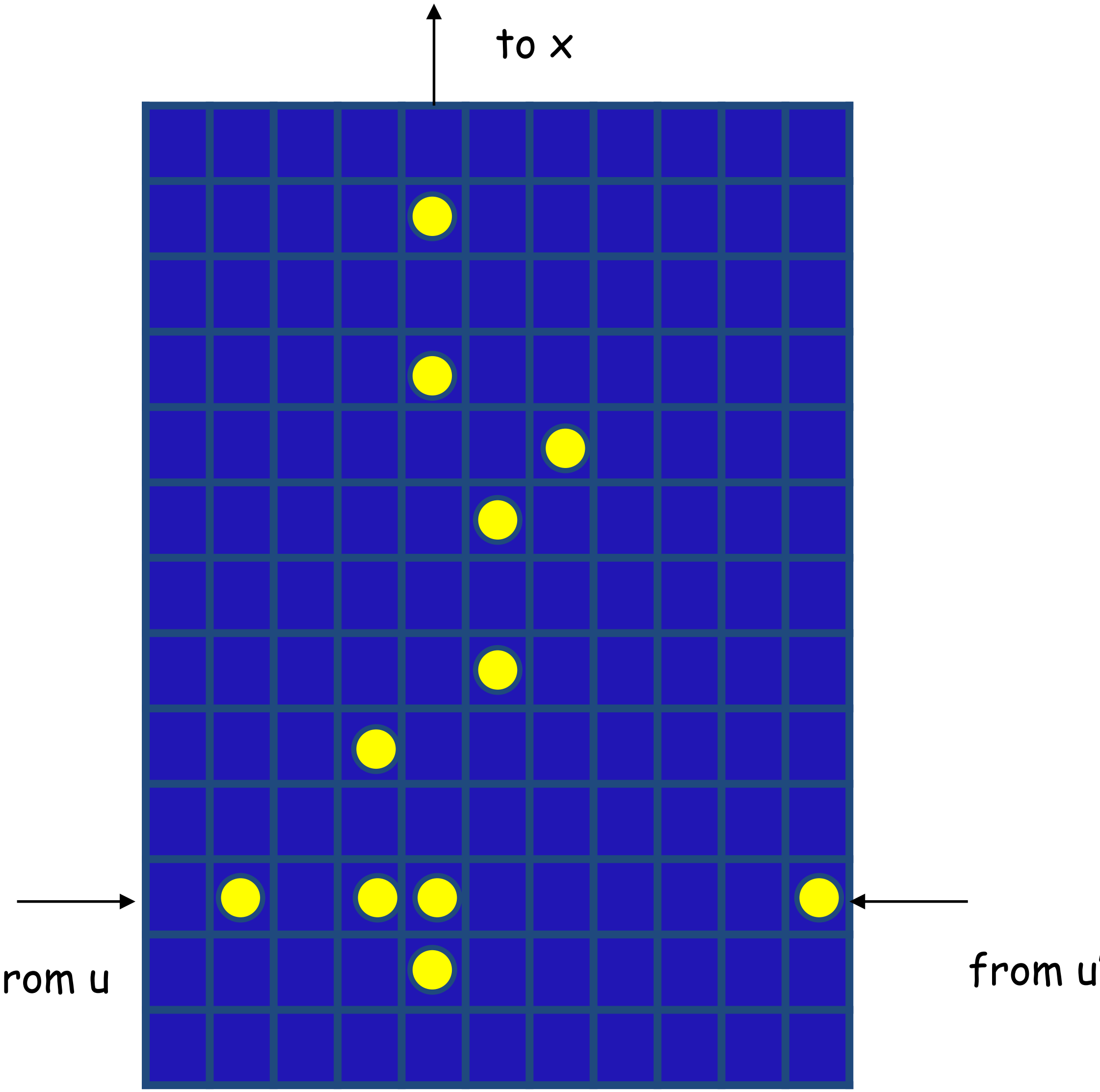
2-in 1-out degree vertex gadget: intended behavior



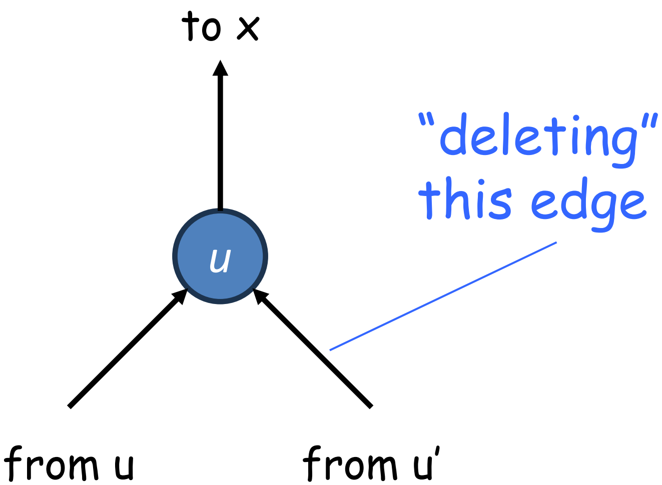


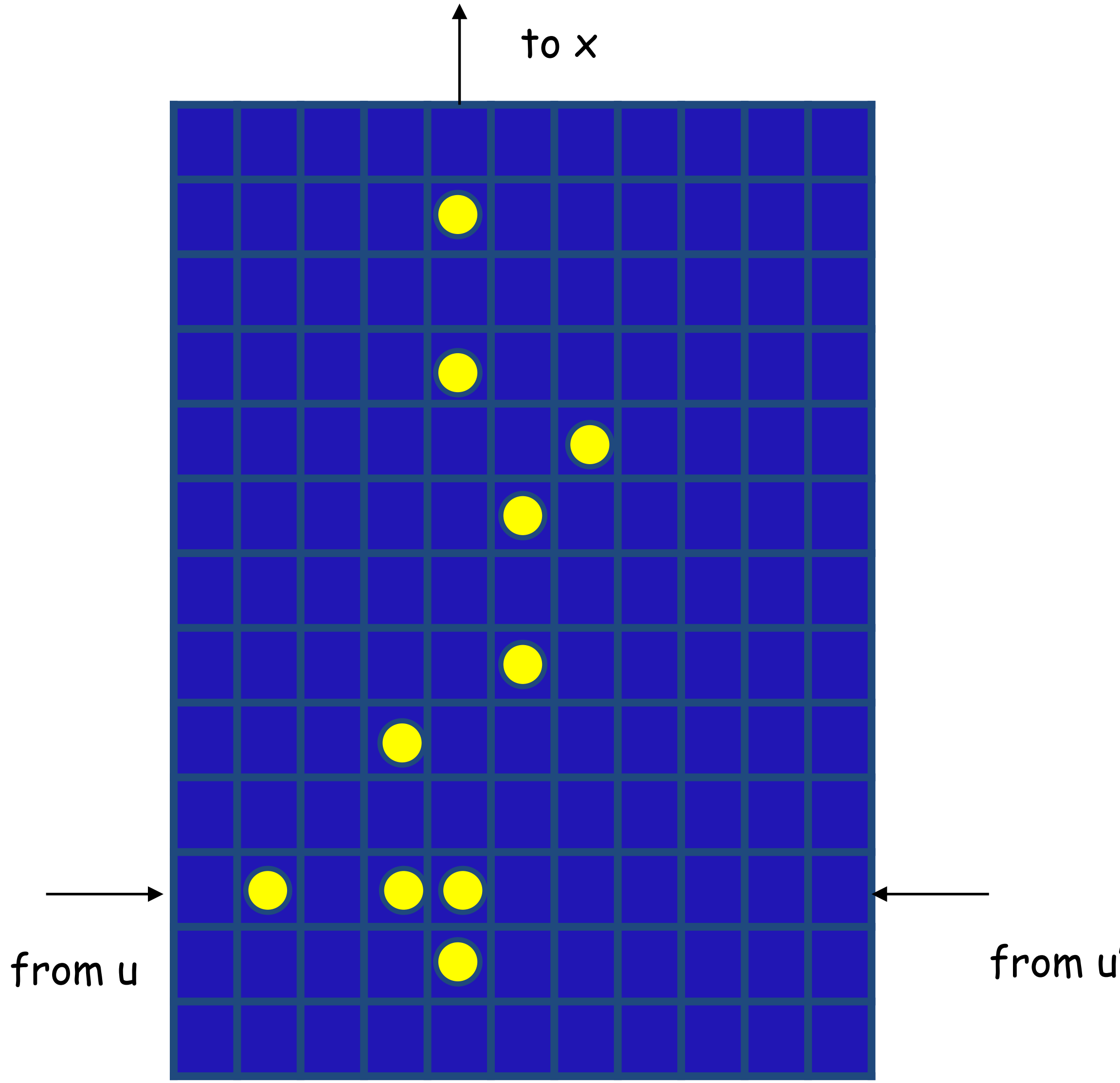
2-in 1-out degree vertex gadget: intended behavior



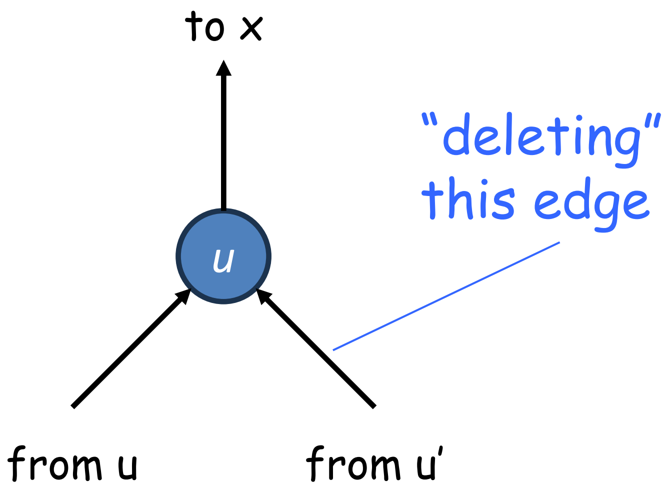


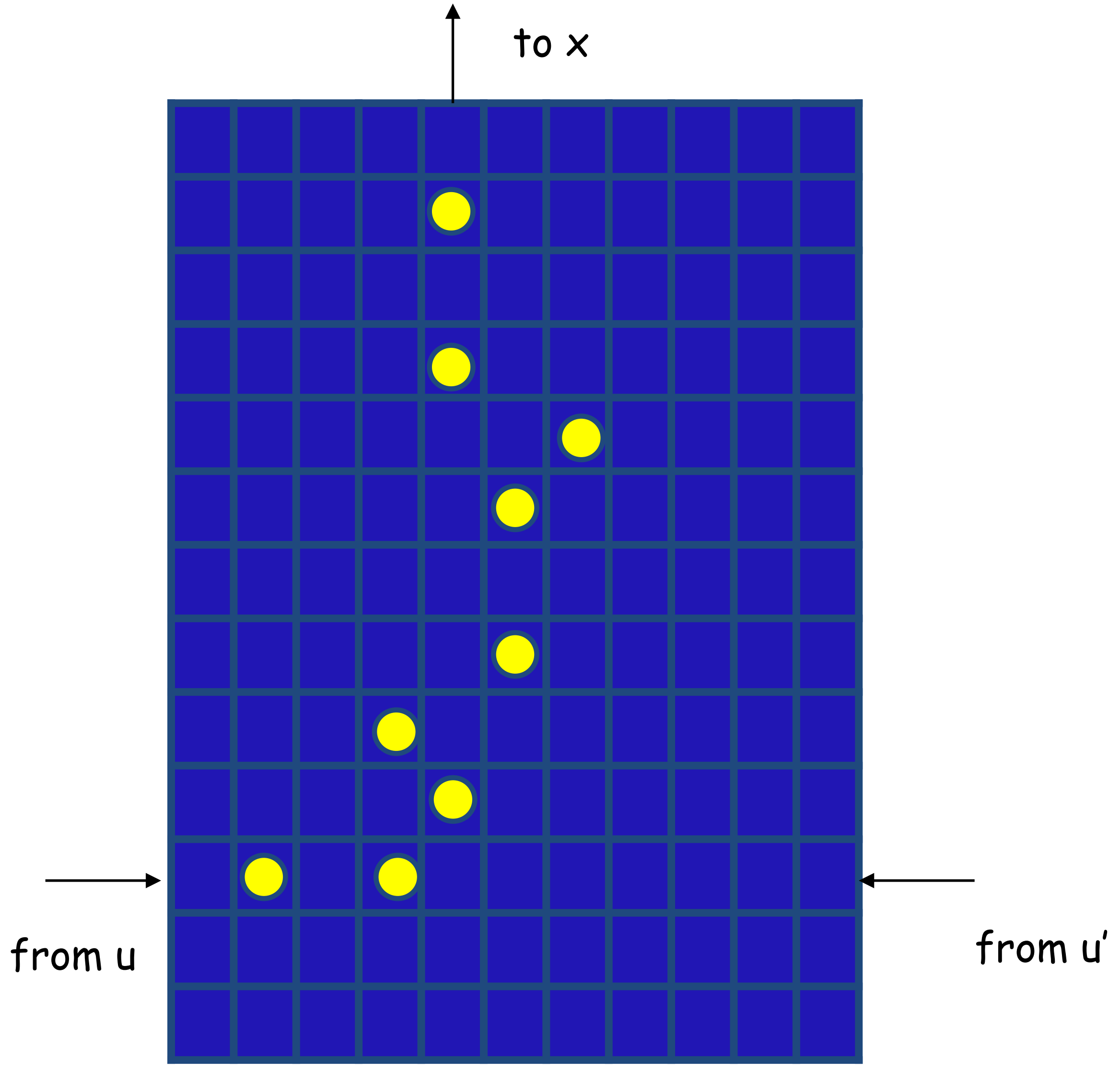
2-in 1-out degree vertex gadget: intended behavior



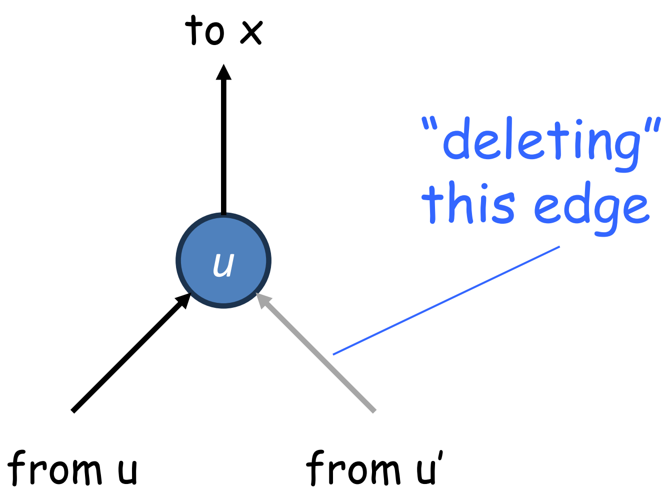


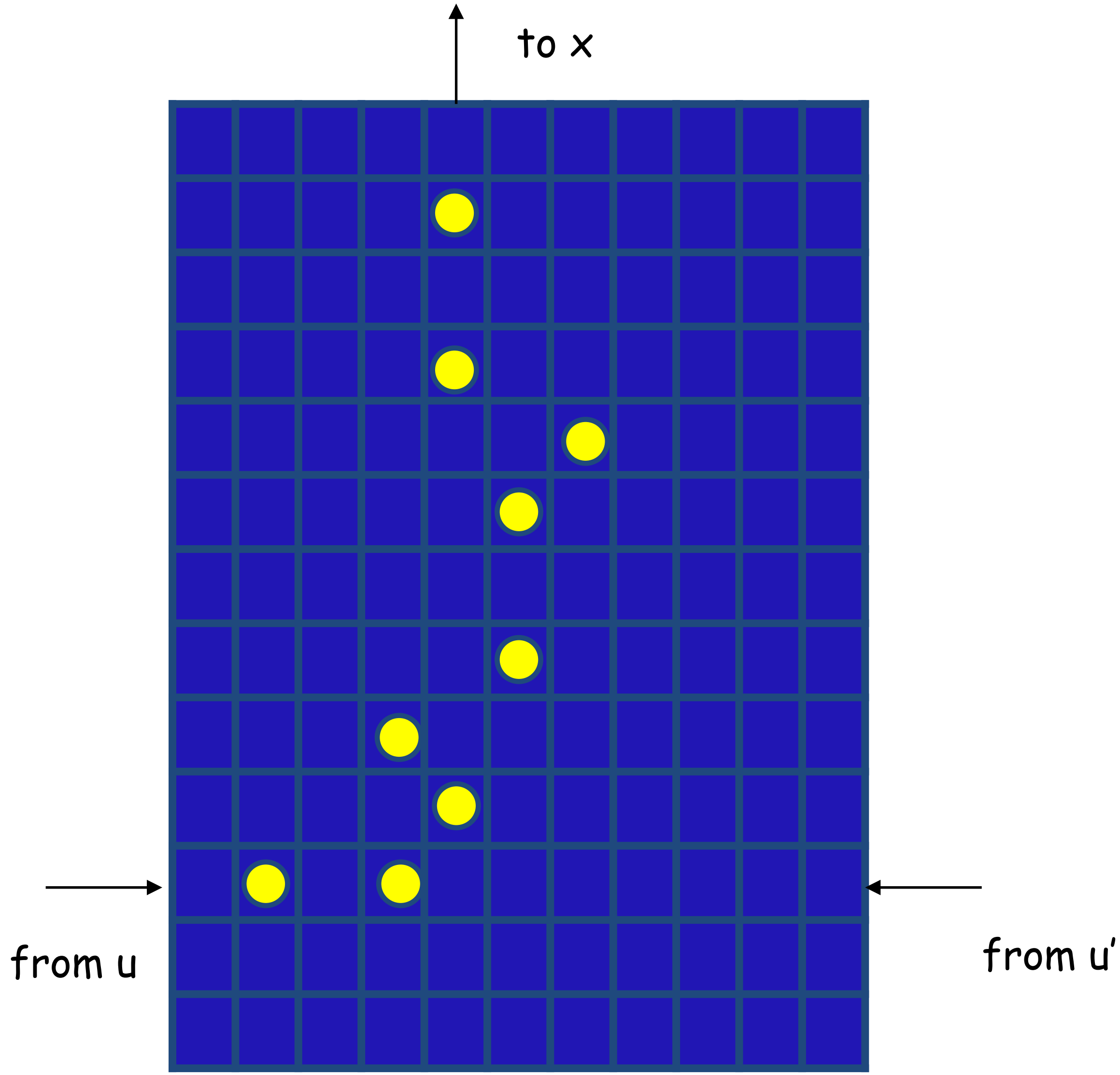
2-in 1-out degree vertex gadget: intended behavior



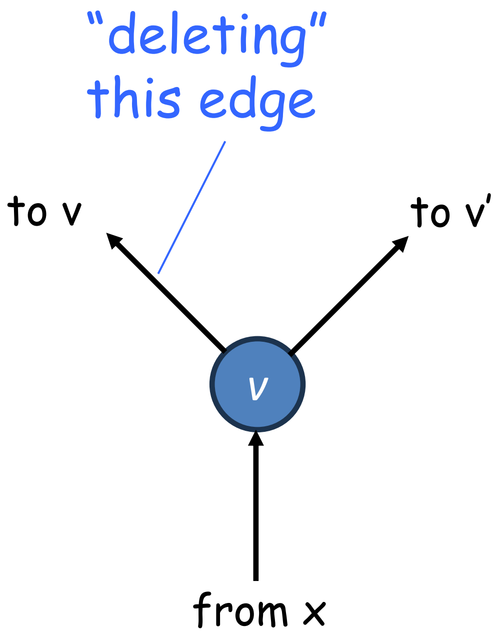


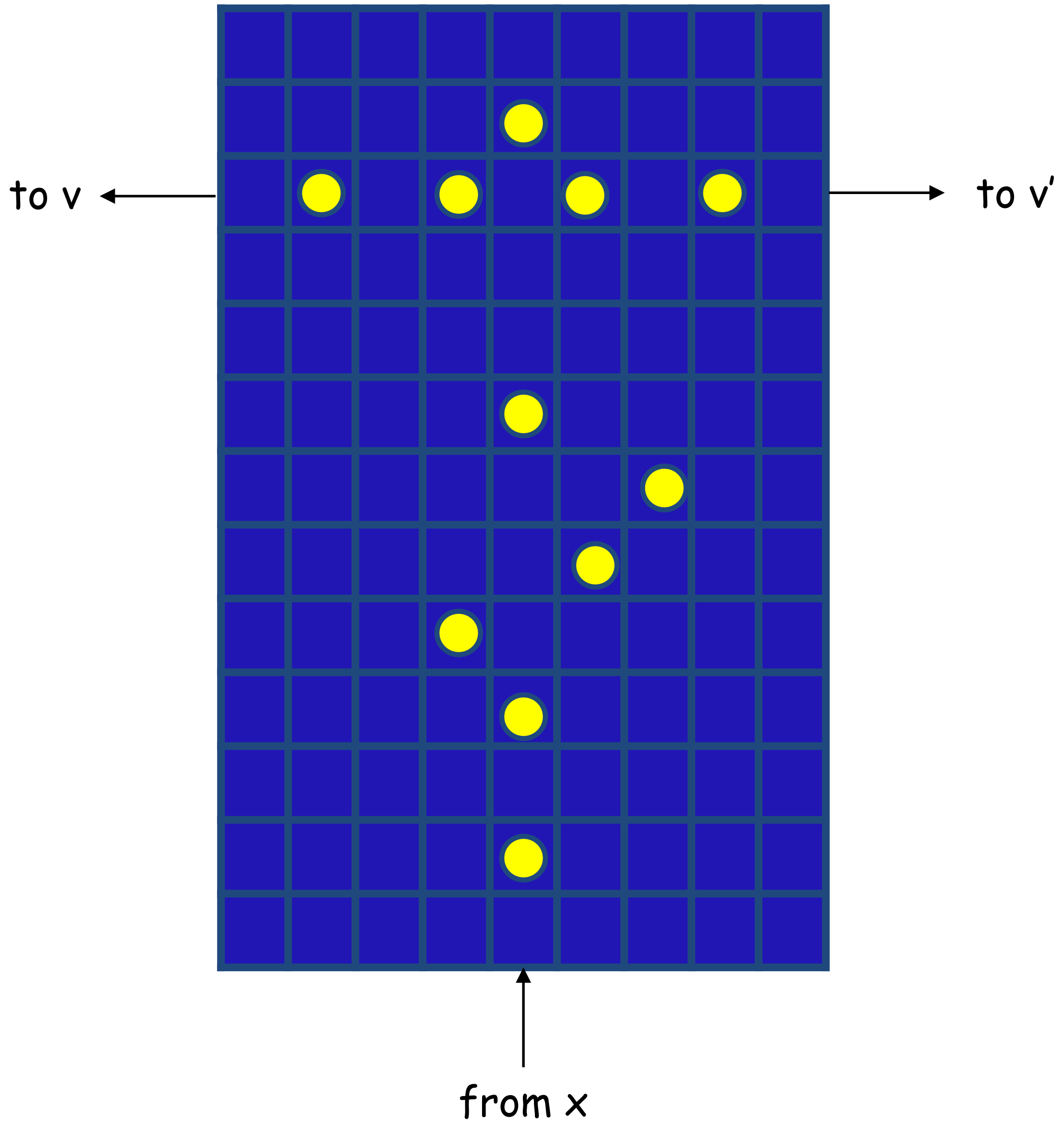
2-in 1-out degree vertex gadget: intended behavior



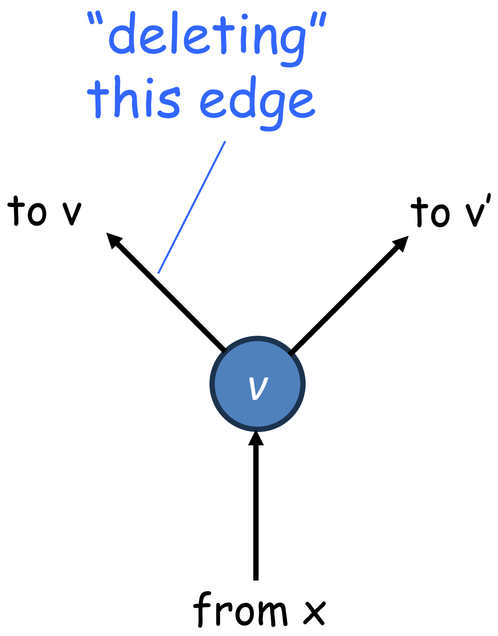


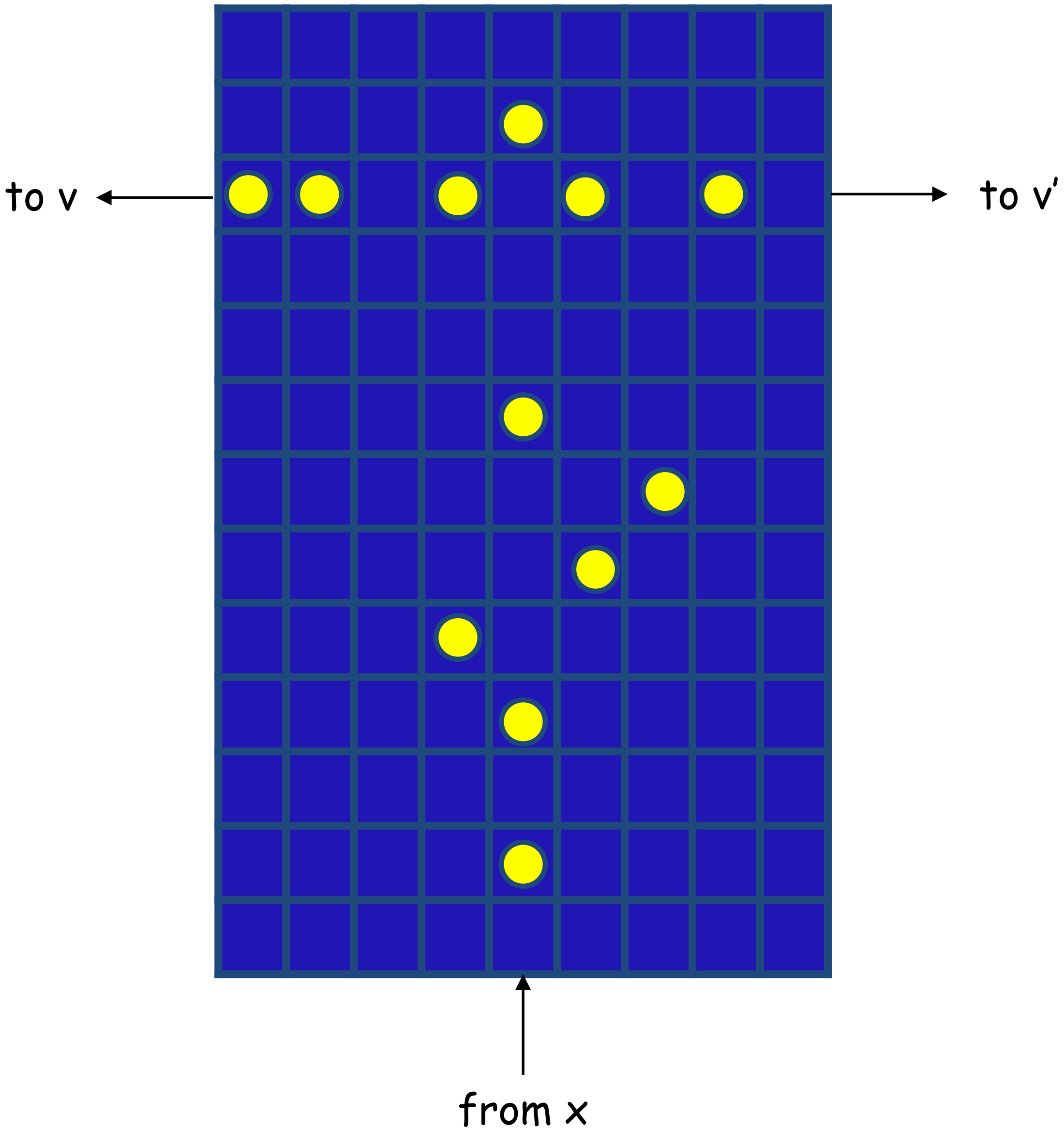
1-in 2-out degree vertex gadget: the intended behavior



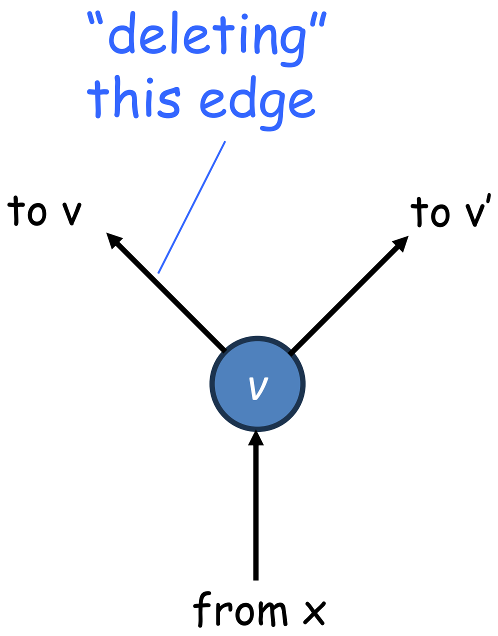


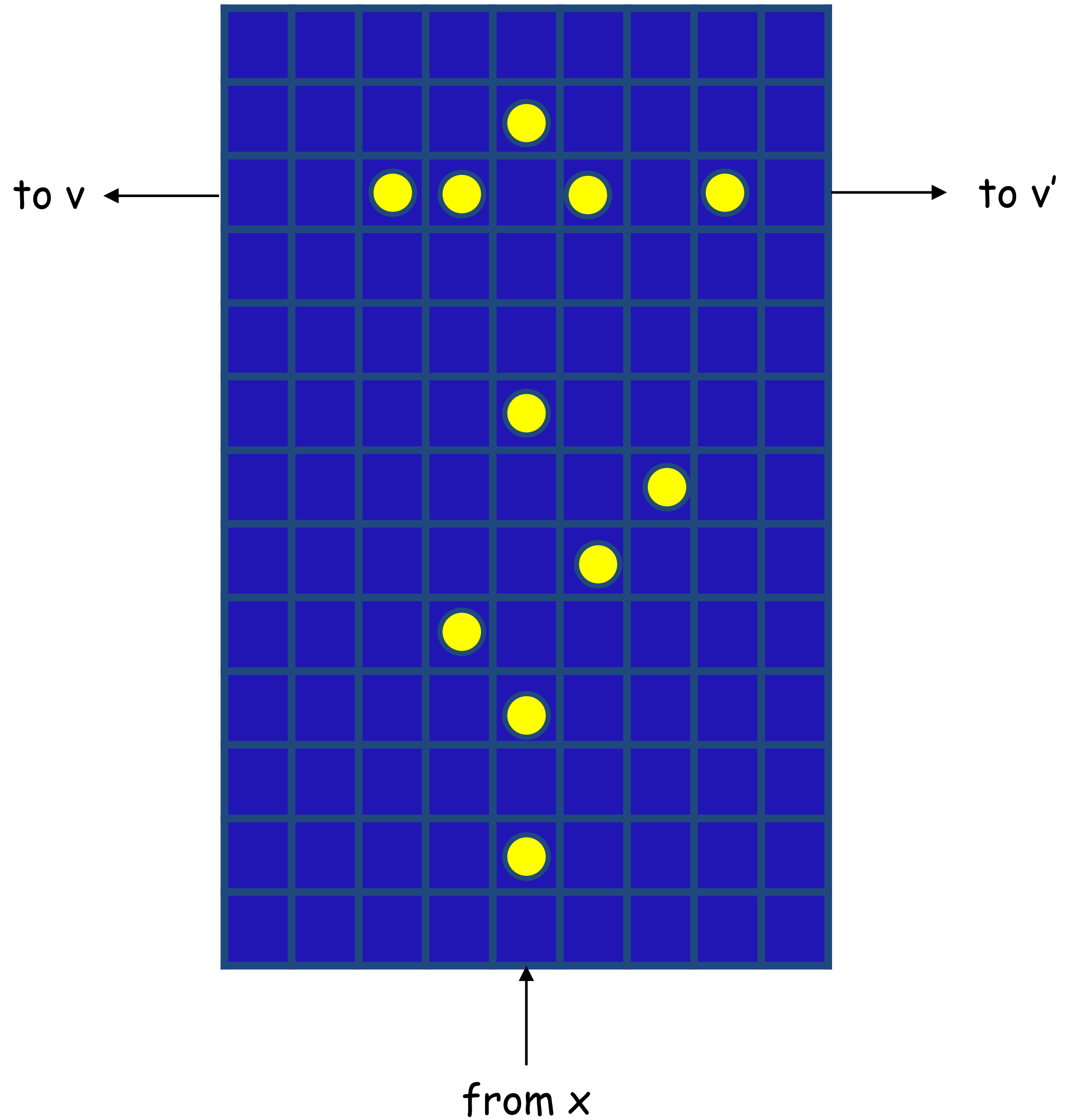
1-in 2-out degree vertex gadget: the intended behavior



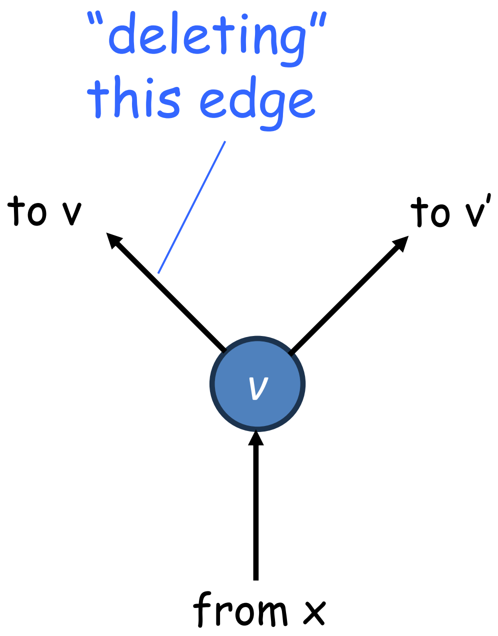


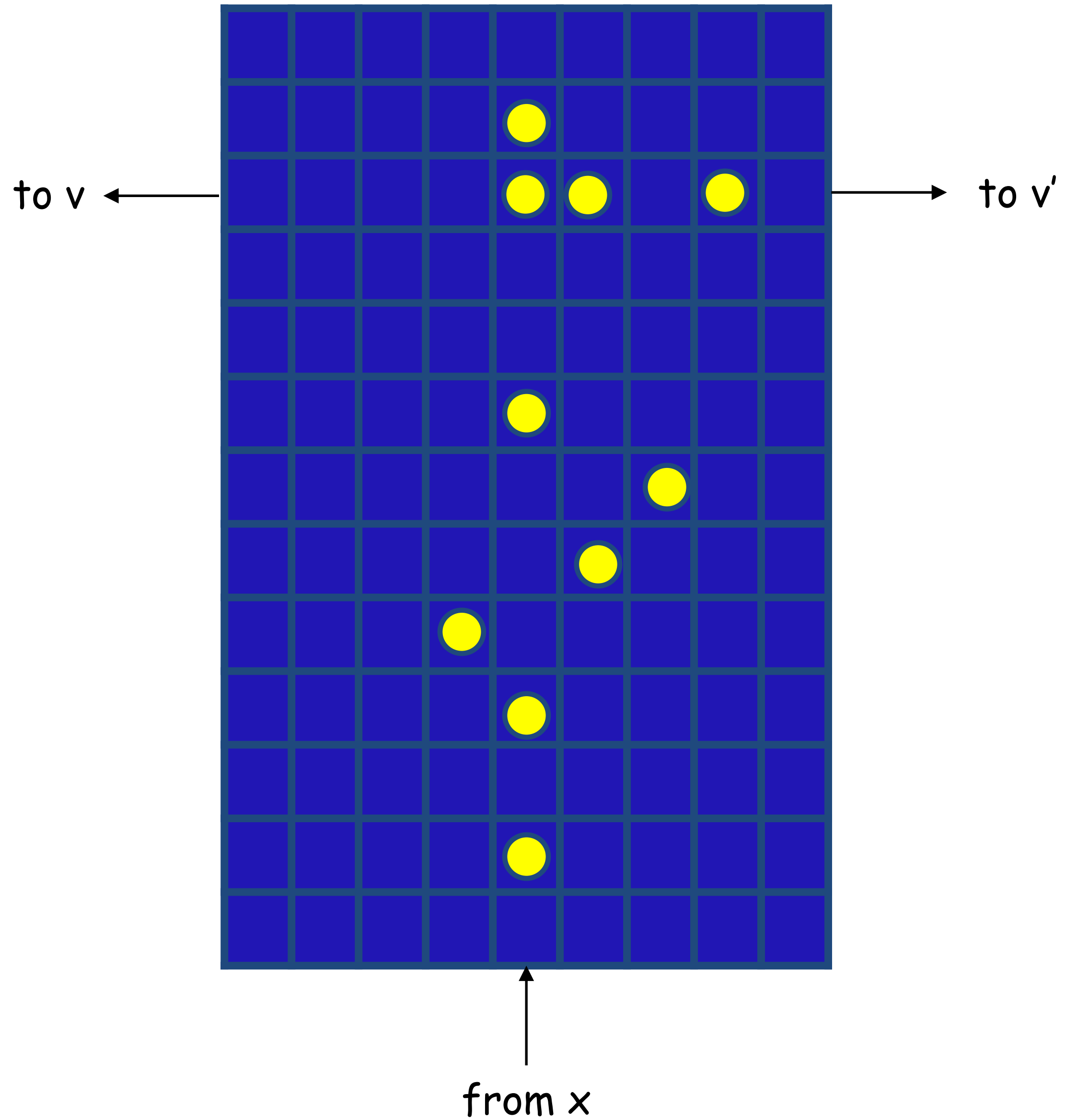
1-in 2-out degree vertex gadget: the intended behavior



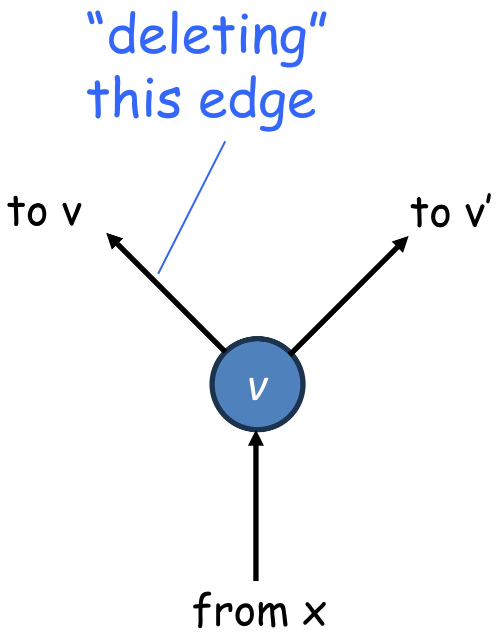


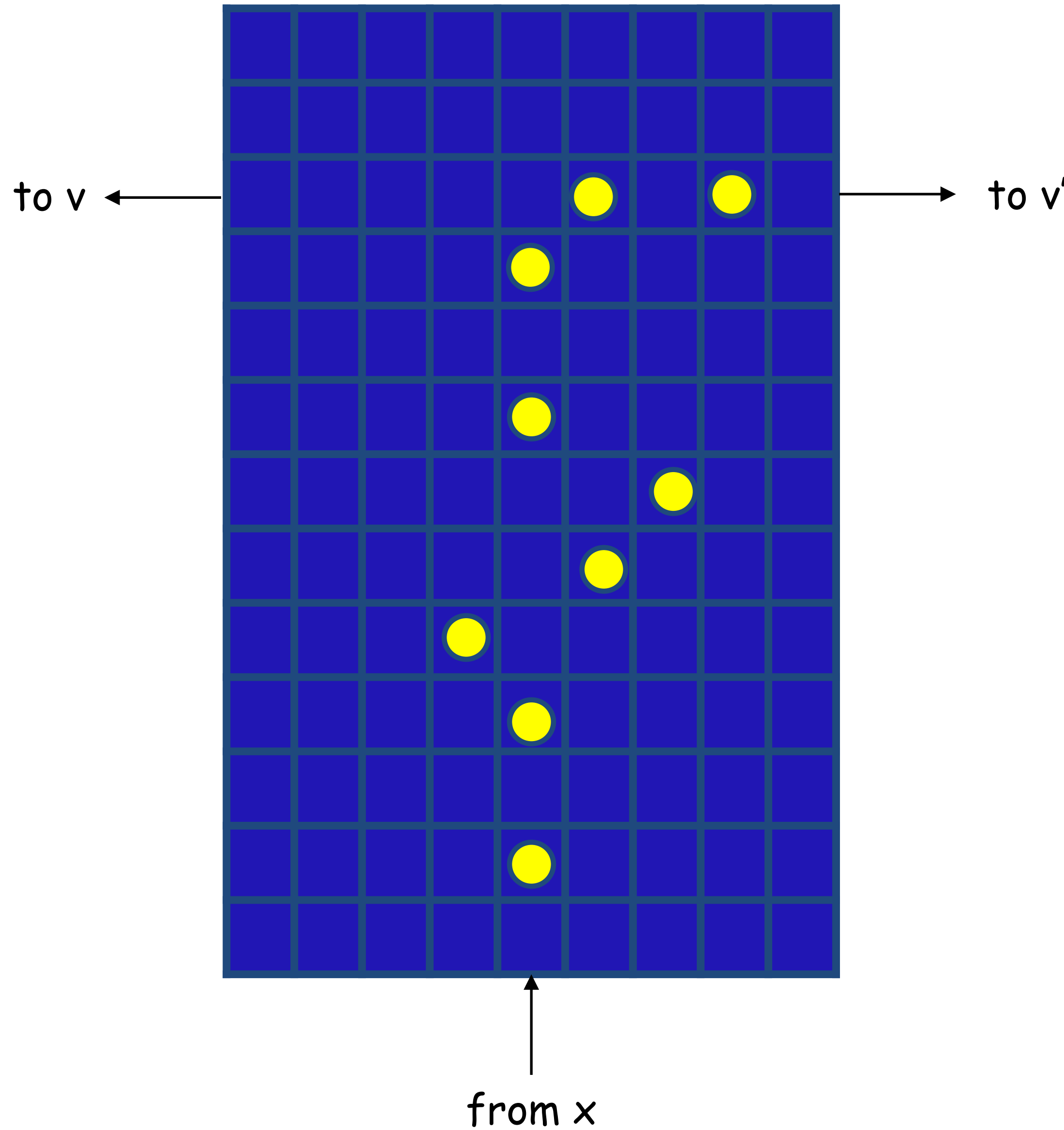
1-in 2-out degree vertex gadget: the intended behavior



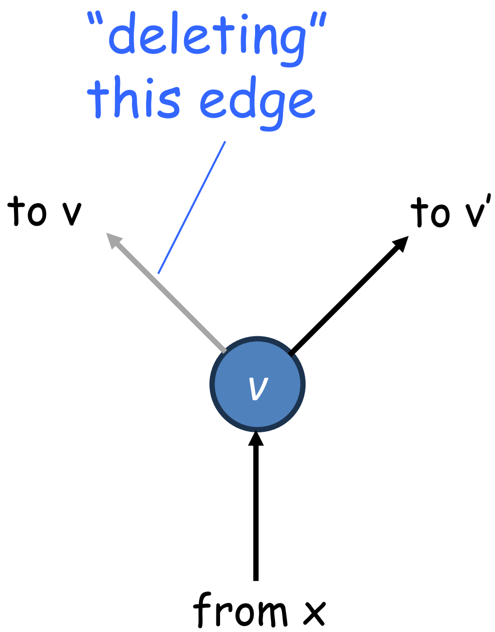


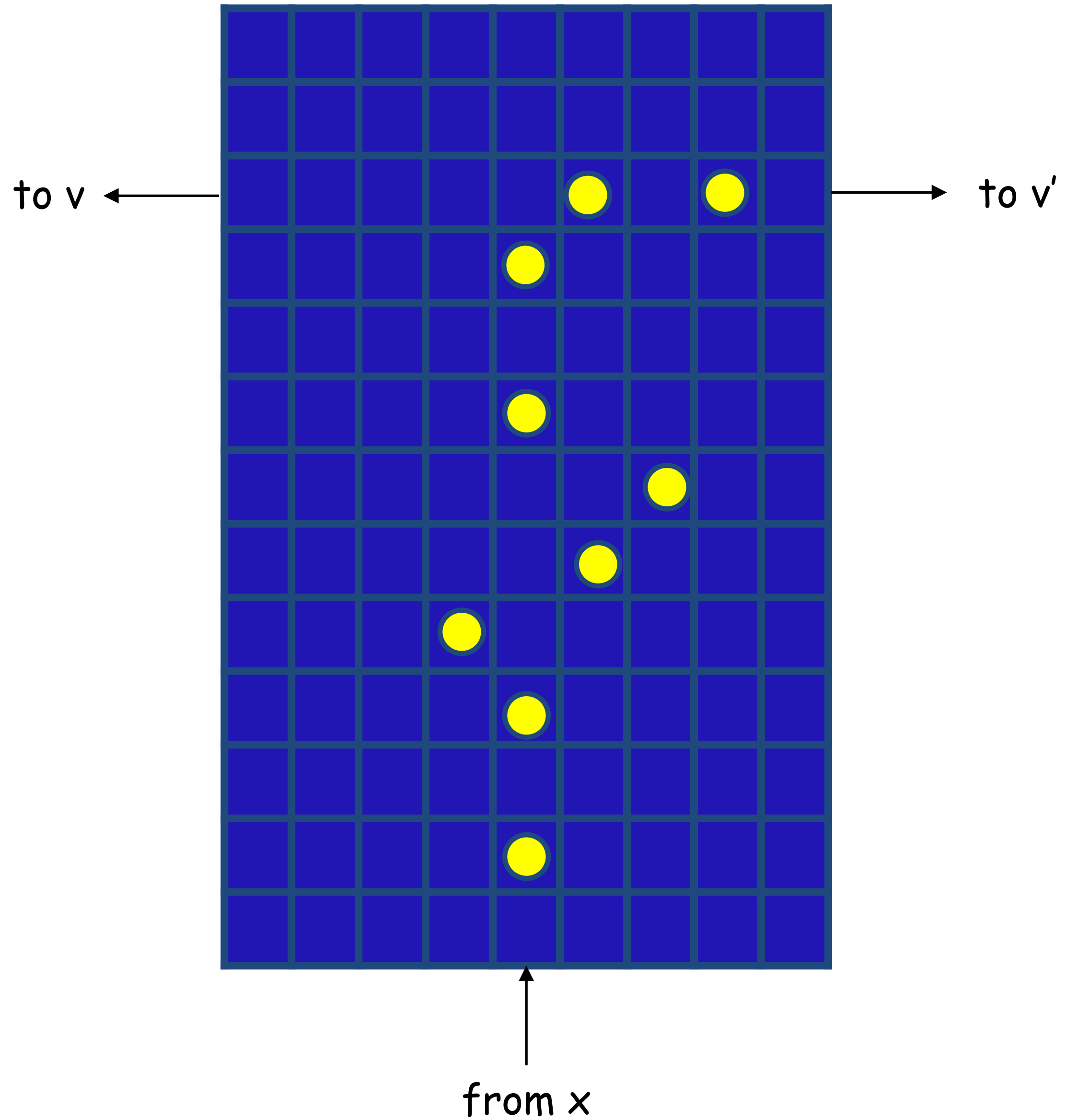
1-in 2-out degree vertex gadget: the intended behavior





1-in 2-out degree vertex gadget: the intended behavior





claim:
you can clear the board if and only if G has a Hamiltonian cycle

proof

($\Leftarrow$)

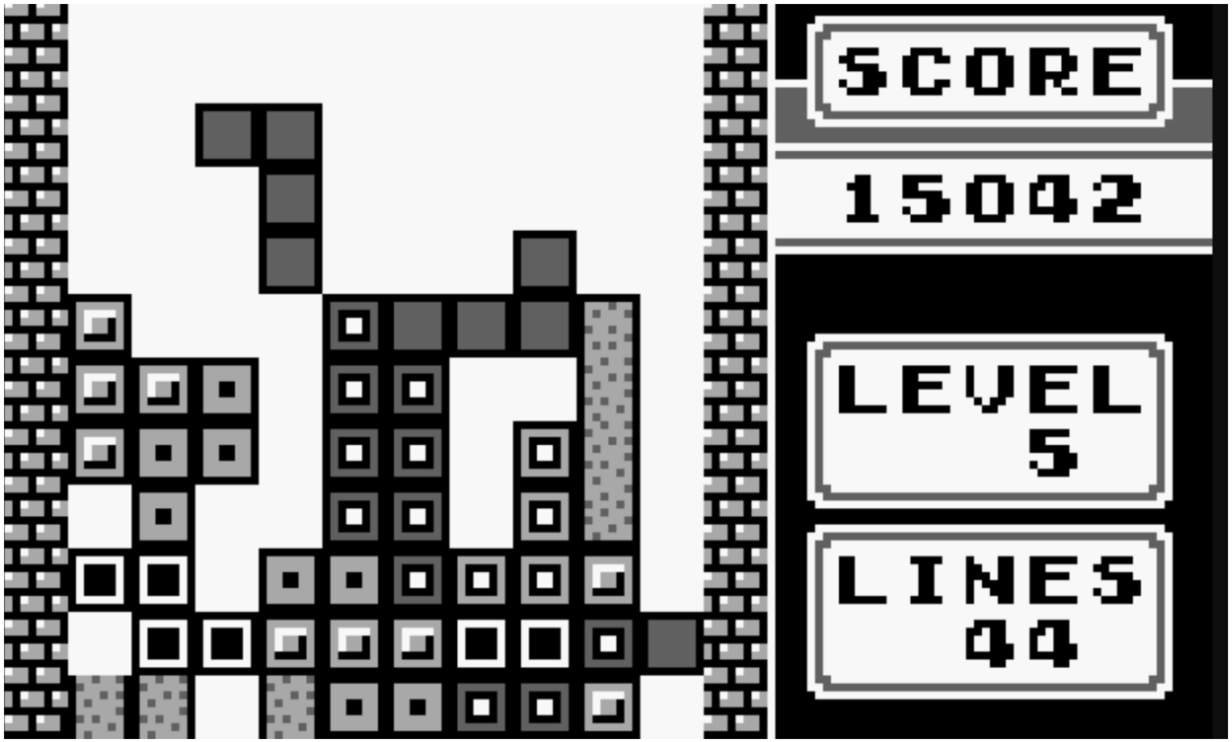
if there is a Hamiltonian cycle C:
- first delete/clear the edges that do not belong to C
- clear the remaining pegs by going through C

($\Rightarrow$)

intuitively: if you want to clear the board you must play the gadgets in the intended way (otherwise you loose)

Play the reduction here:
https://www.isnphard.com/g/peg-solitaire/

Tetris


SCORE
15042
LEVEL
5
LINES
44


Nintendo GAME BOY TM
TETRIS
DMG-TR-USA-1
MADE IN JAPAN
THIS SIDE OUT
Nintendo

Tetris


TETRIS
THE ABSOLUTE
THE GRAND MASTER 2
PLUS
PUSH START
SEGA
ASTRO CITY
SEGA
TETRIS
SUPER TETRIS 3
TETRIS 2
TETRIS DS
TETRIS PARTY
TETRIS ADVANCE
PlayStation 2
TETRIS 64
THE TETRIS EFFECT
DAN ACKERMAN

Tetris

Russian cosmonaut Aleksandr Serebrov became the first person to play a videogame in space when he packed a Game Boy and his personal copy of Tetris (Nintendo, 1989) for his trip to the MIR Space Station in 1993.

3-Partition problem

- **Input:** a collection A of n positive integers $a_1 \dots a_n$
- **question:** is it possible to partition A in $n/3$ collections $A_1 \dots A_{n/3}$ of equal sum, i.e.

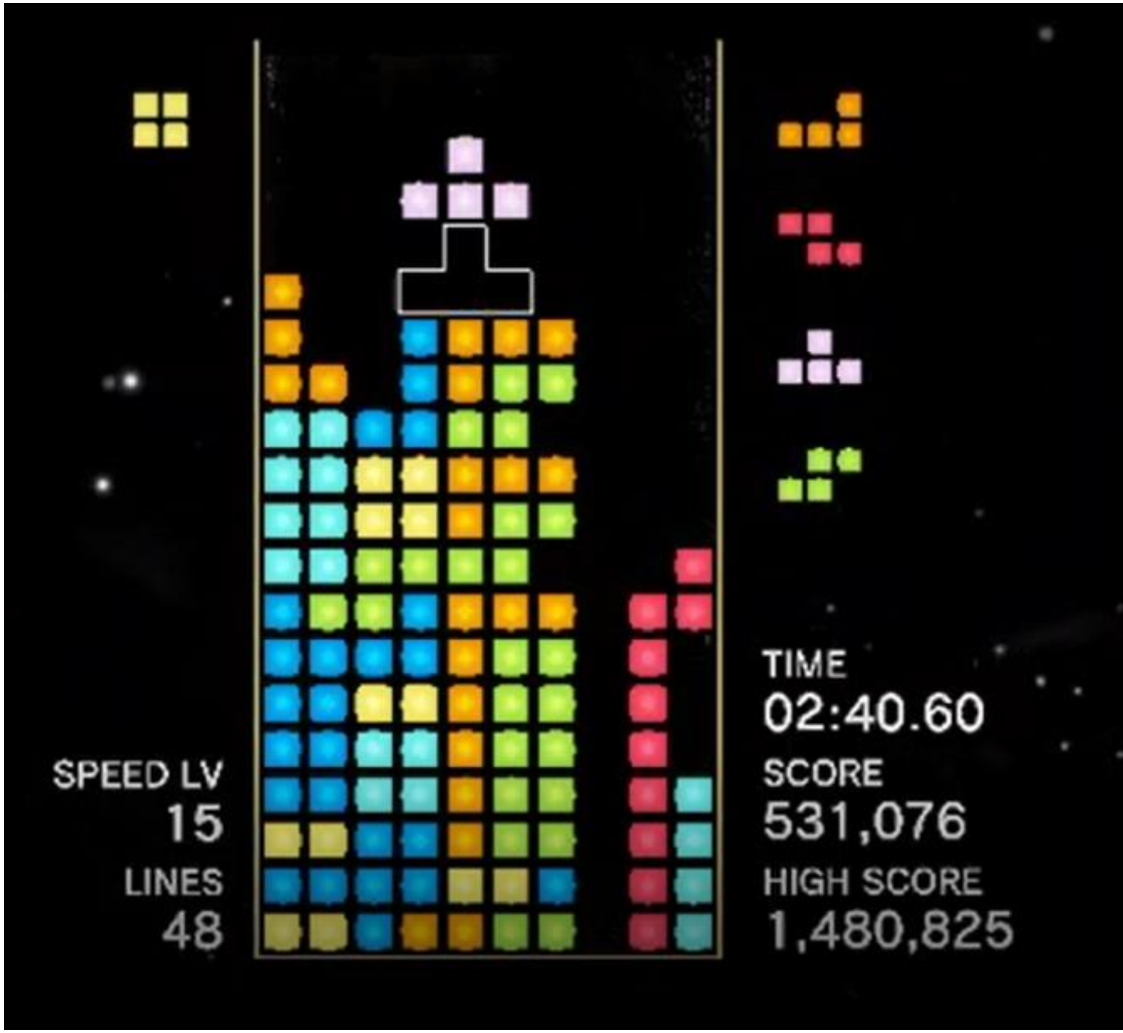
$$\sum_{a \in A_1} a = \cdots = \sum_{a \in A_{\frac{n}{3}}} a = \frac{\sum_{a \in A} a}{\frac{n}{3}} = t$$

- **Fact 1:** 3-Partition is NP-complete, even if $t/4 < a_i < t/2$.
- **Obs.:** if we assume $t/4 < a_i < t/2$ we have $|A_i| = 3$, for each A_i

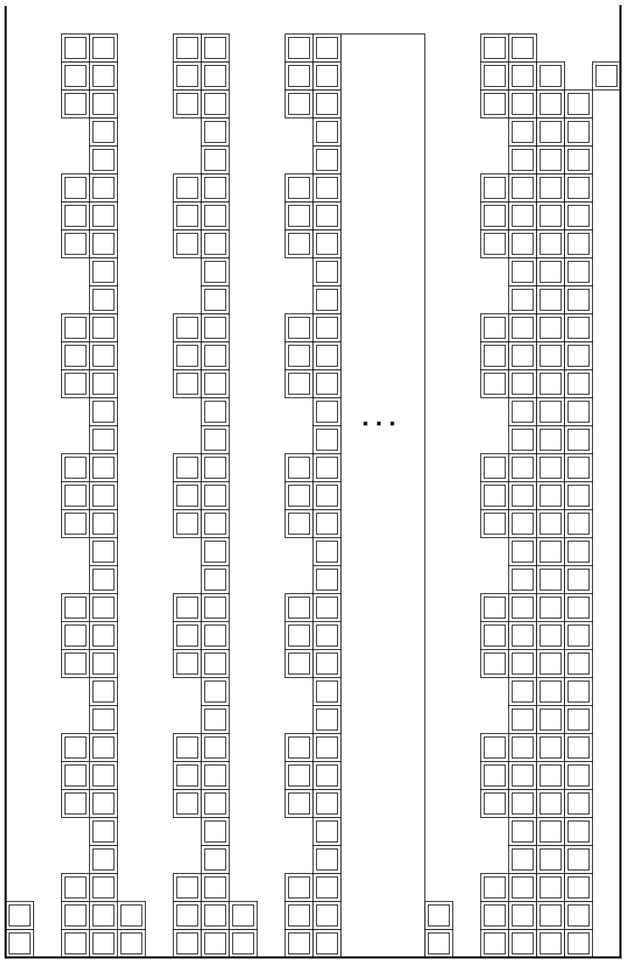
- **Fact 2:** 3-Partition is *strong* NP-hard, i.e. it is NP-complete even if *every a_i is polynomially bounded in n (n: the number of numbers).*

input: an initial configuration of the board, and the entire (offline) sequence of the pieces

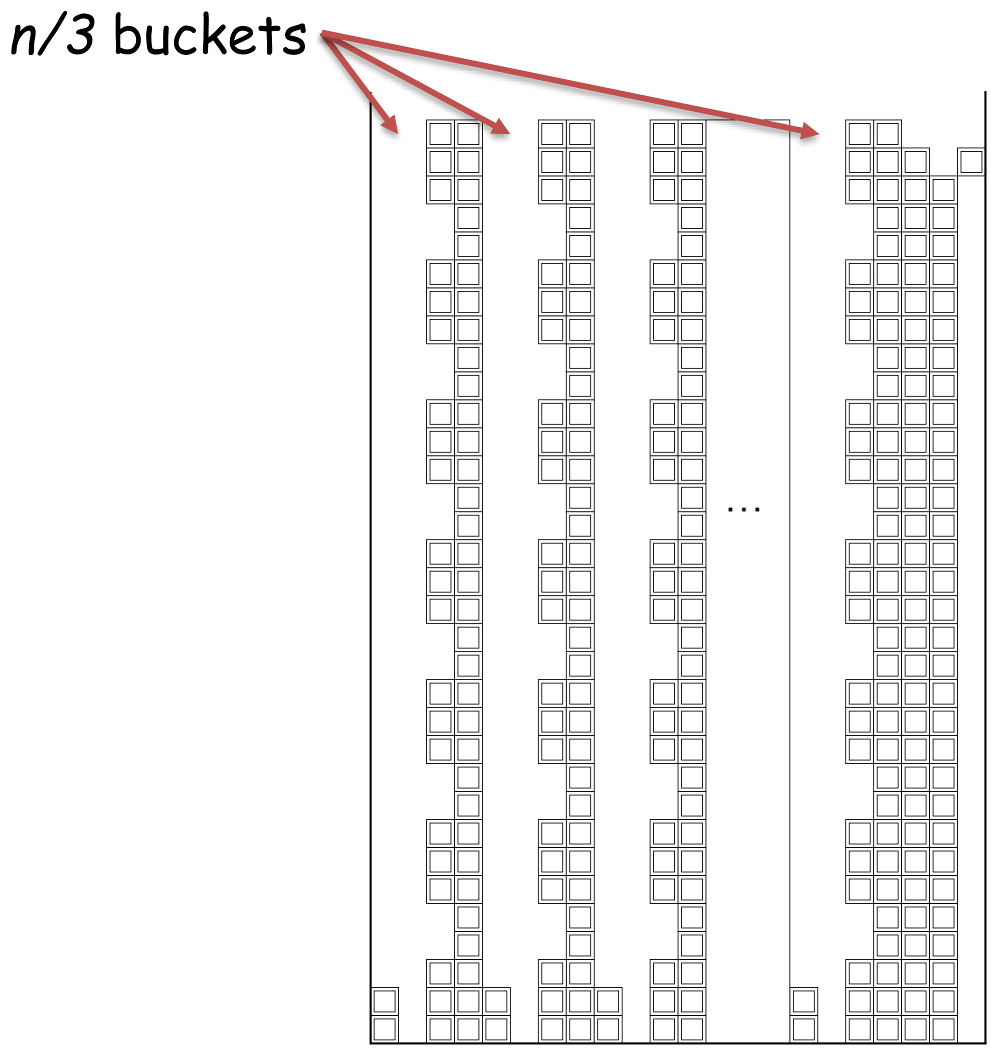
goal: can you clear all the board?



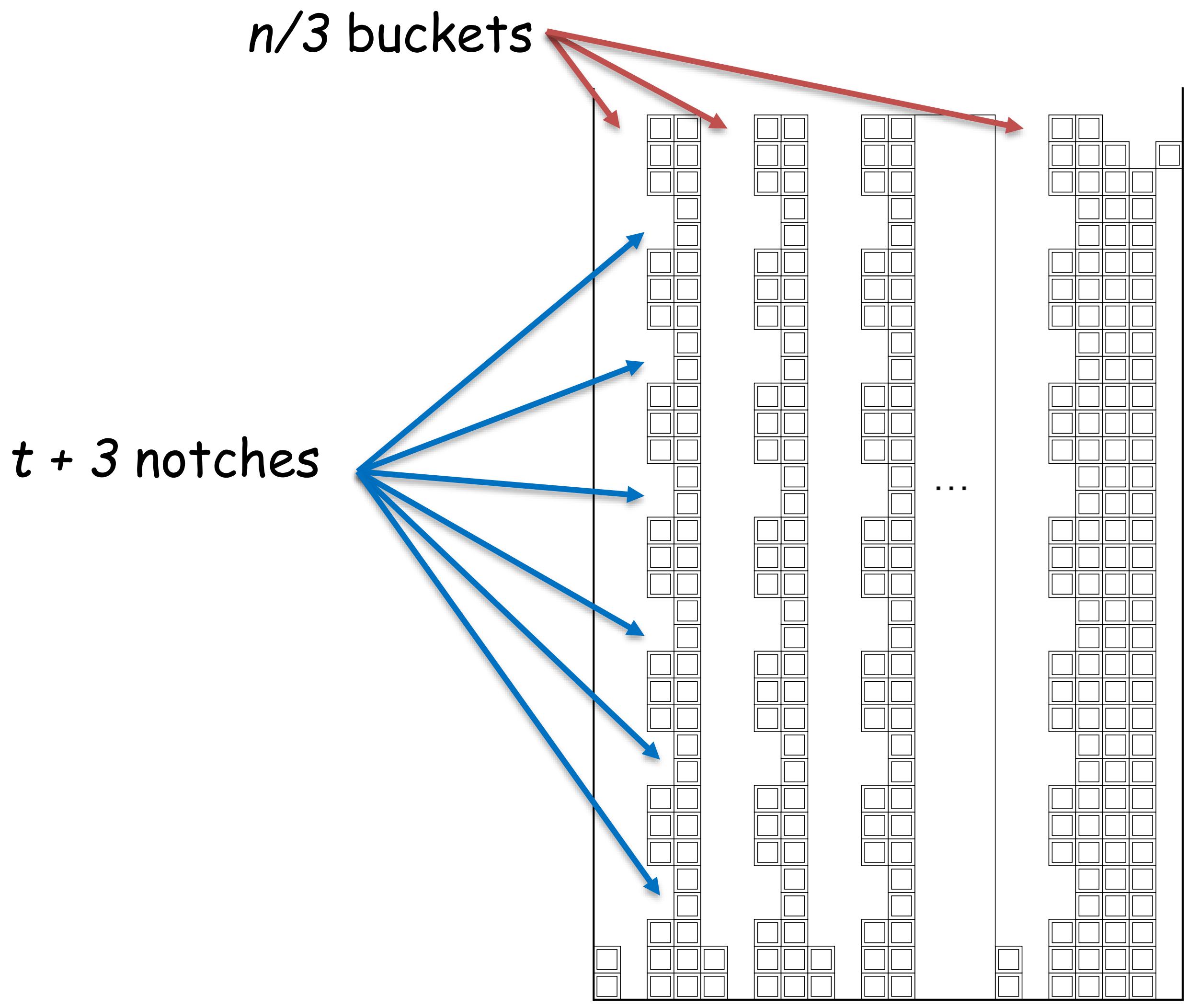
3-Partition to Tetris



3-Partition to Tetris

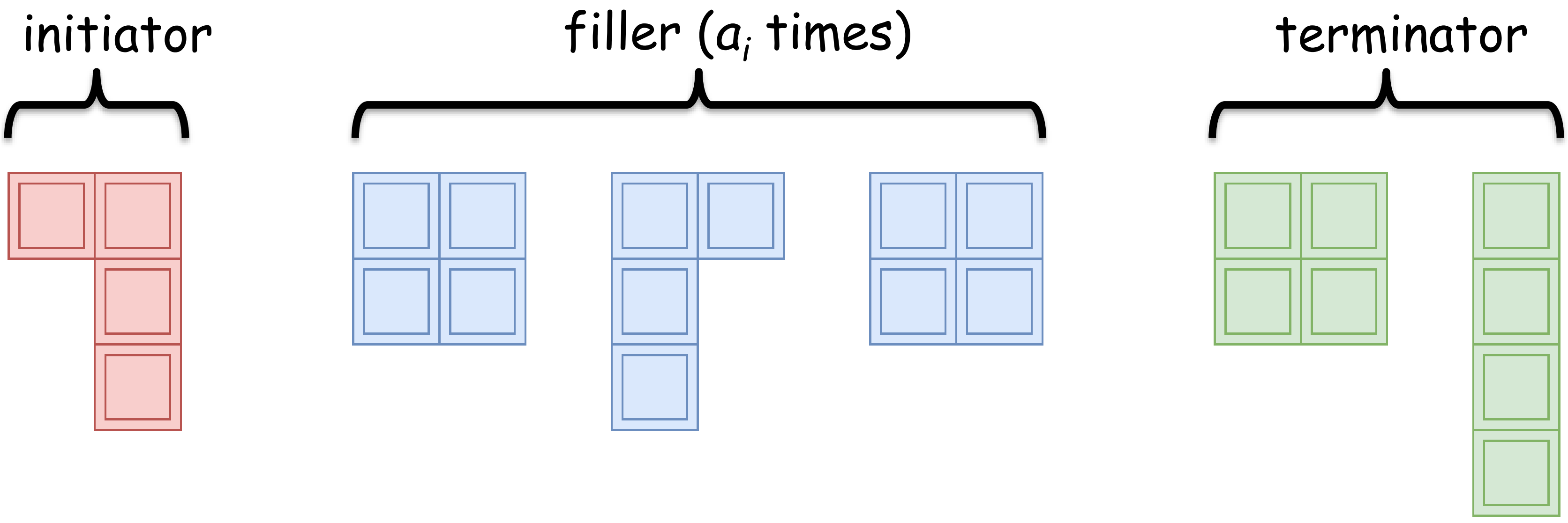


3-Partition to Tetris



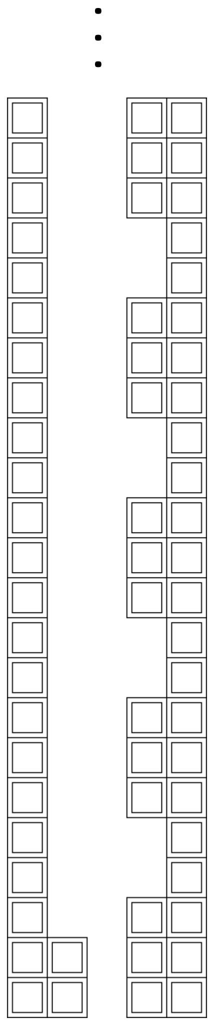
3-Partition to Tetris

a_i gadget



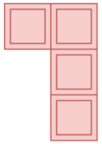
3-Partition to Tetris

$a_i = 3$

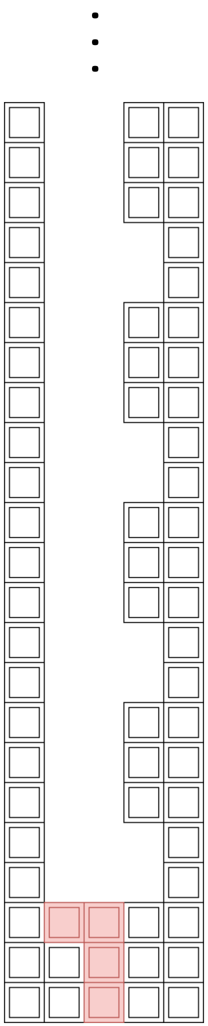


3-Partition to Tetris

$a_i = 3$

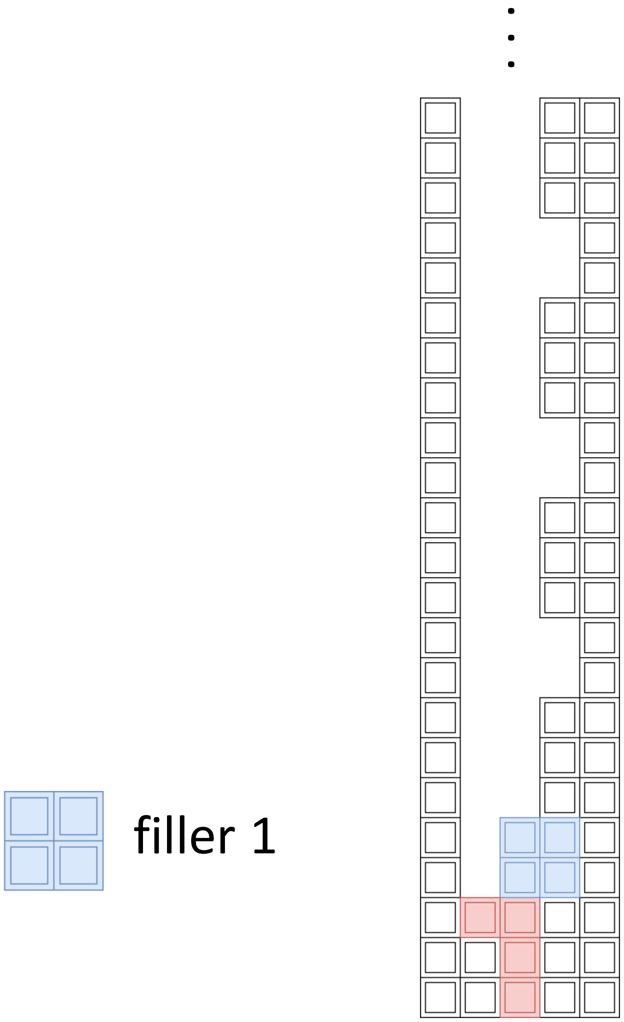


initiator



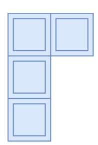
3-Partition to Tetris

$a_i = 3$

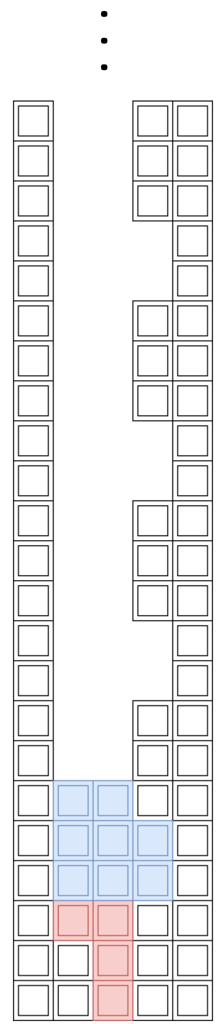


3-Partition to Tetris

$a_i = 3$

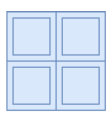


filler 1

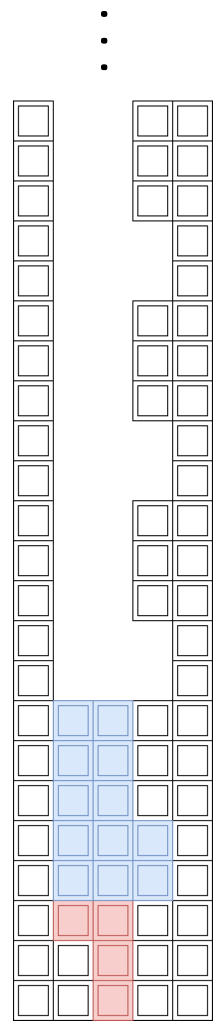


3-Partition to Tetris

$a_i = 3$

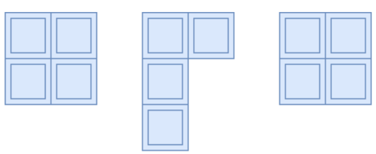


filler 1

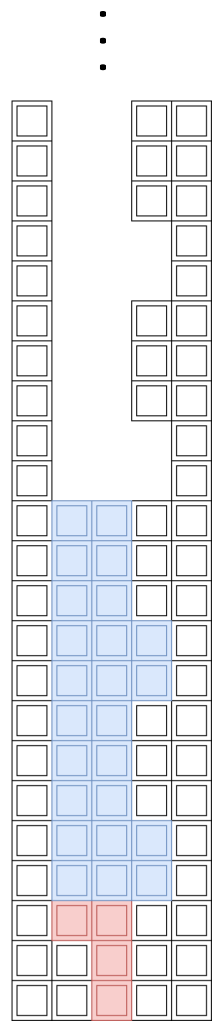


3-Partition to Tetris

$a_i = 3$

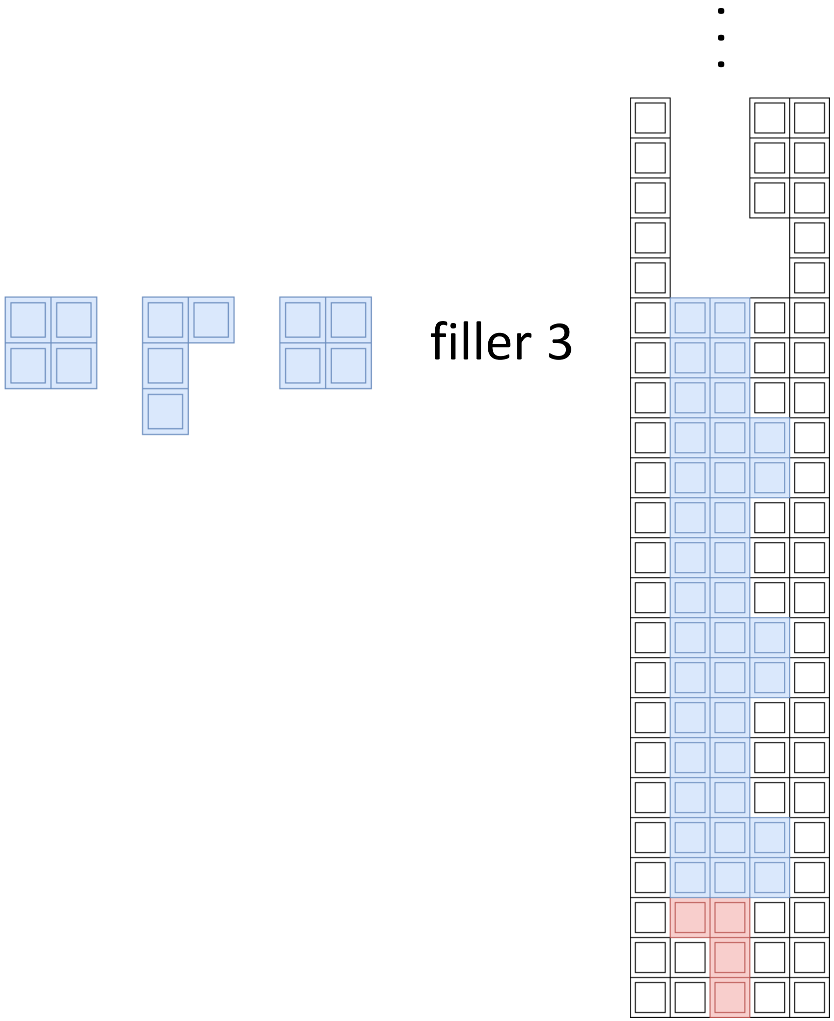


filler 2

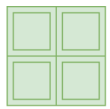


3-Partition to Tetris

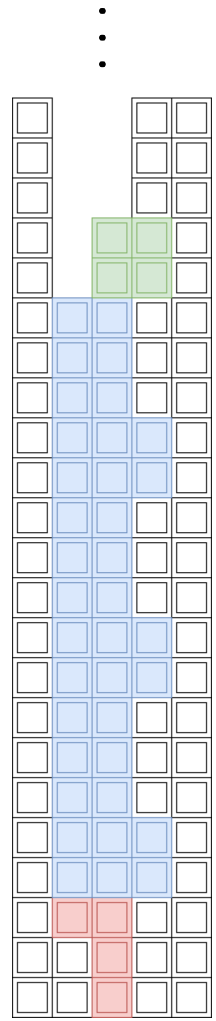
$a_i = 3$



3-Partition to Tetris



terminator

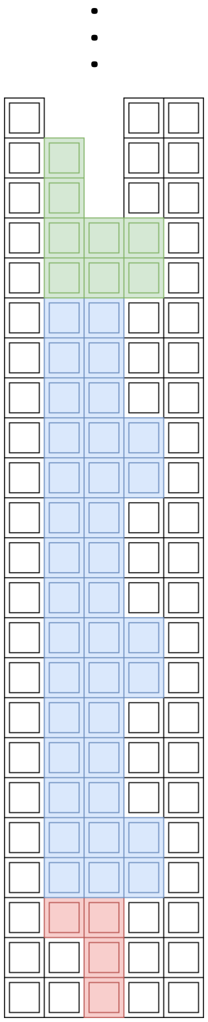


$a_i = 3$

3-Partition to Tetris

terminator

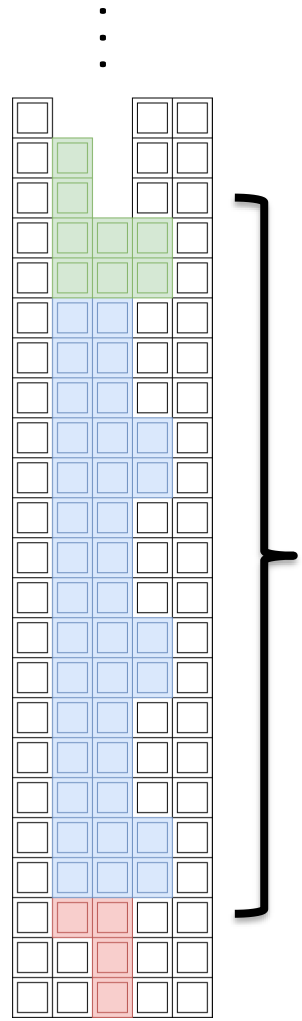
$a_i = 3$



3-Partition to Tetris

terminator

$a_i = 3$

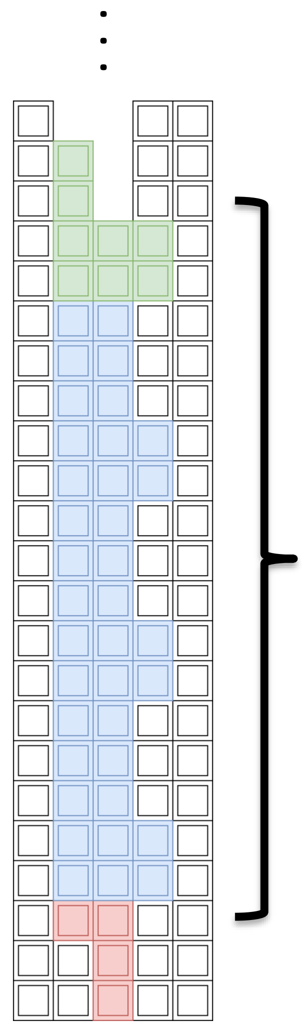


exactly $a_i + 1$ notches are used

3-Partition to Tetris

terminator

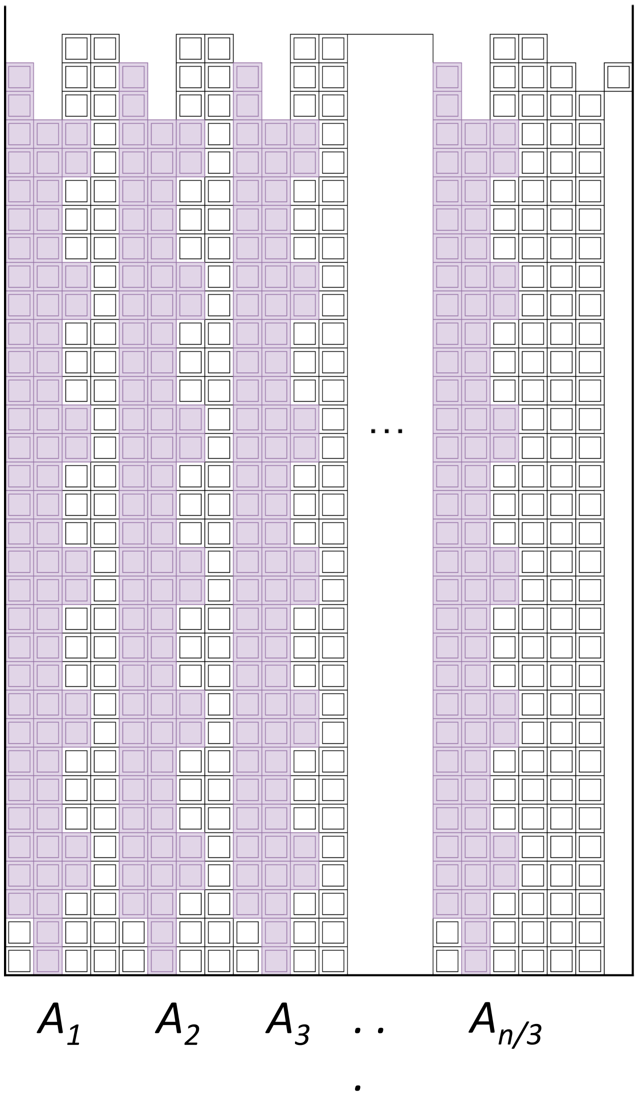
$a_i = 3$



exactly $a_i + 1$ notches are used

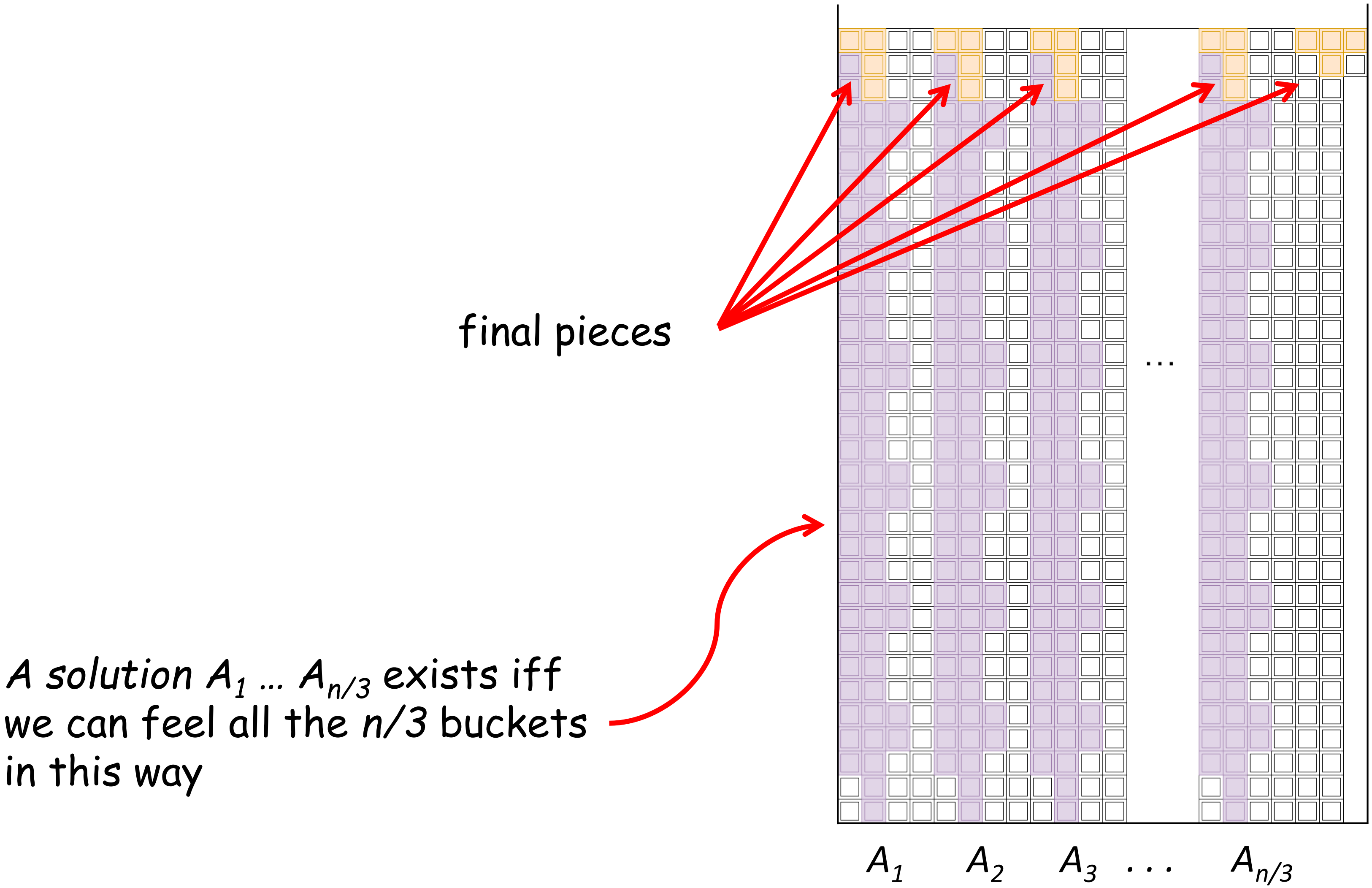
Since each $|A_k| = 3$ and $sum(A_k) = t$ then we need $t + 3$ notches.

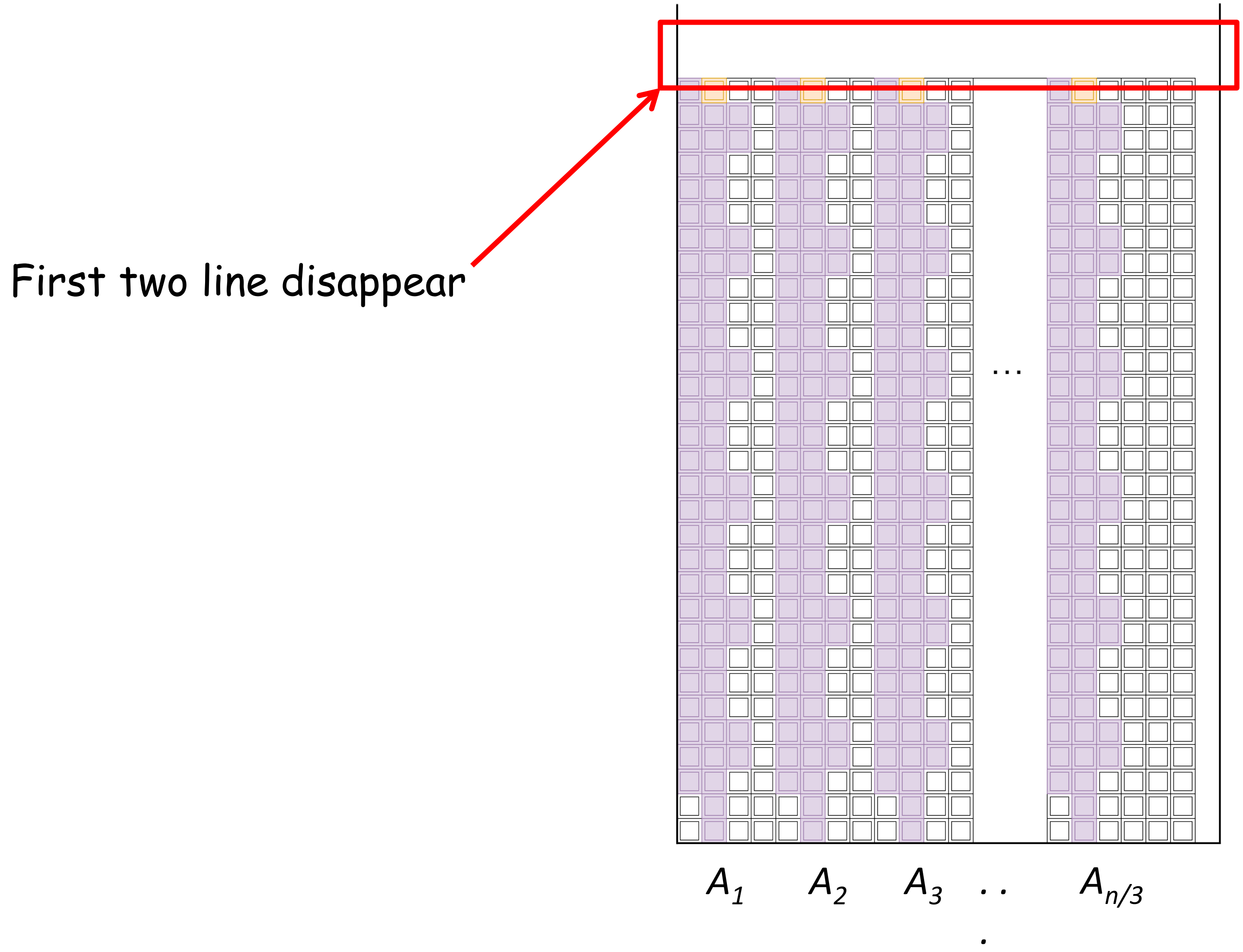
3-Partition to Tetris



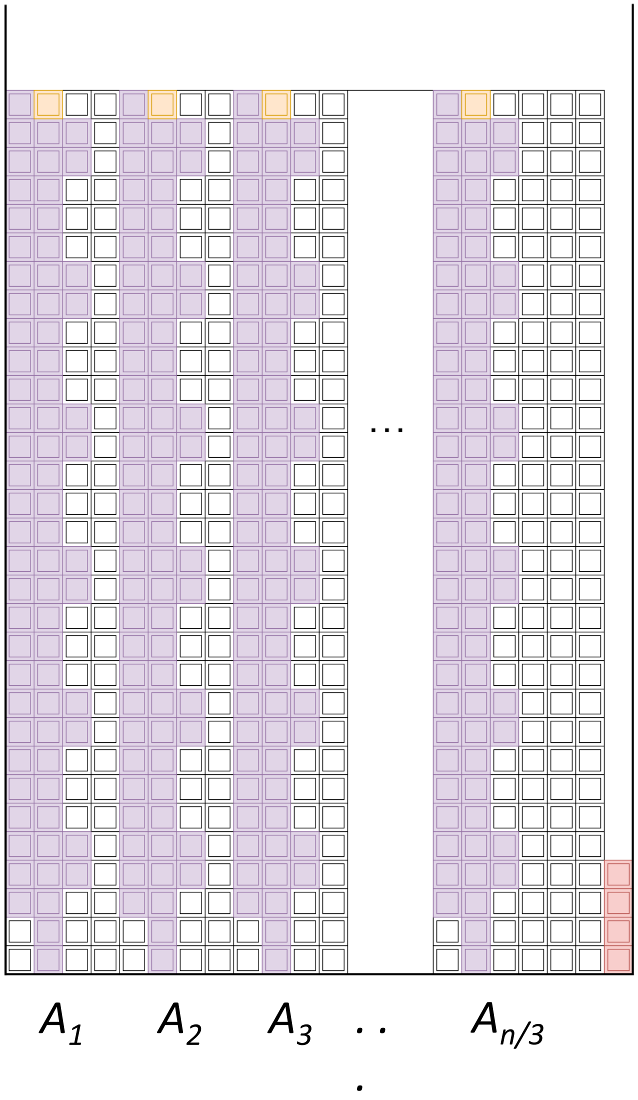
A solution $A_1 \ldots A_{n/3}$ exists iff we can feel all the *n/3* buckets in this way

3-Partition to Tetris

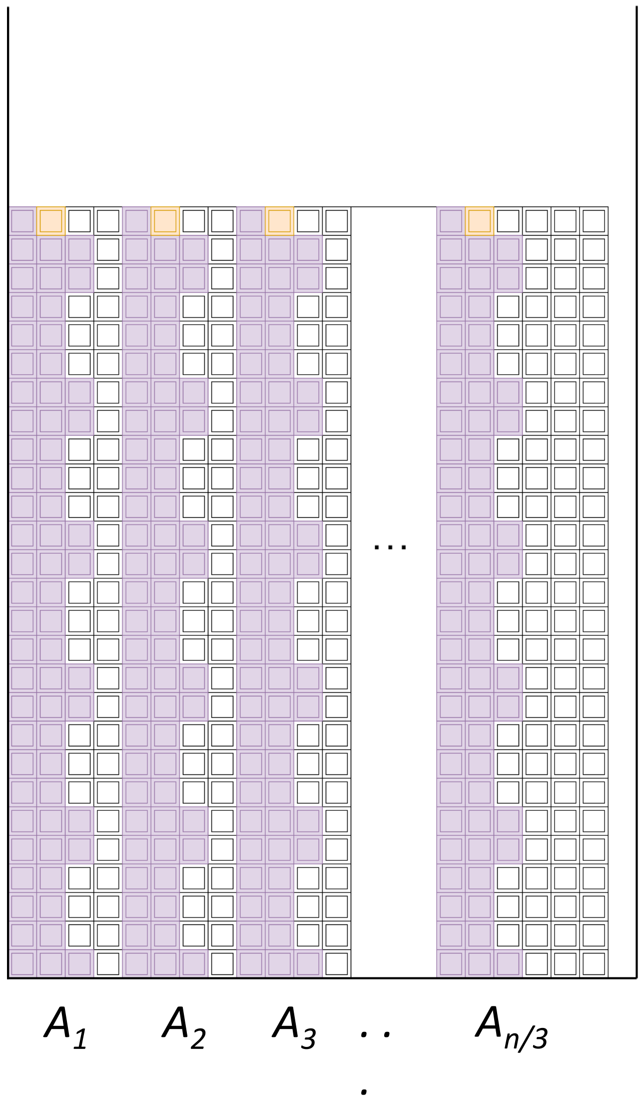



First two line disappear
...
A_1
A_2
A_3
...
$A_{n/3}$

3-Partition to Tetris

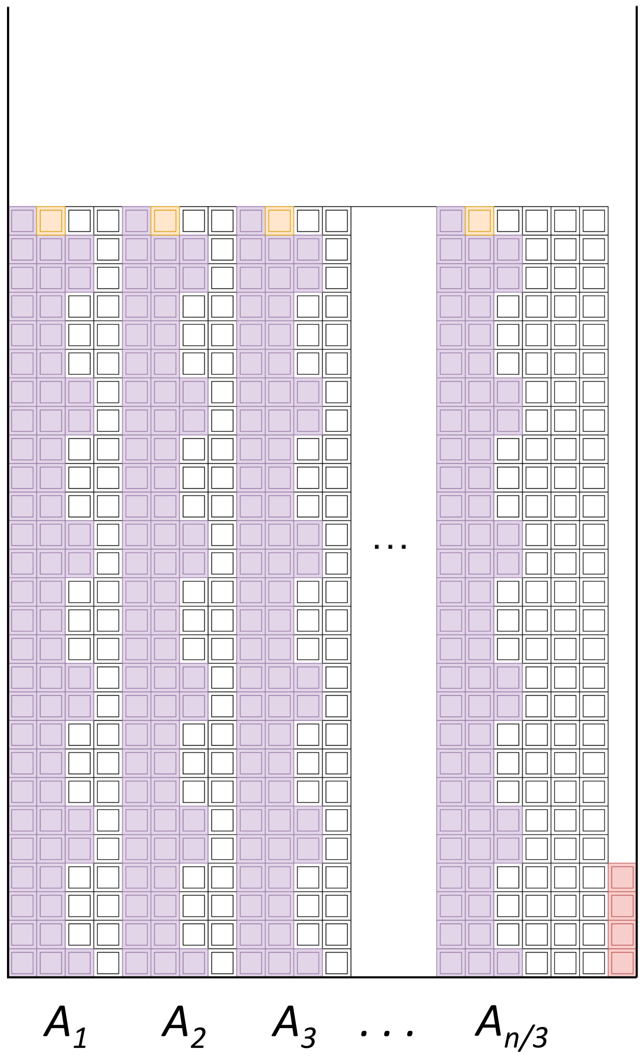


3-Partition to Tetris



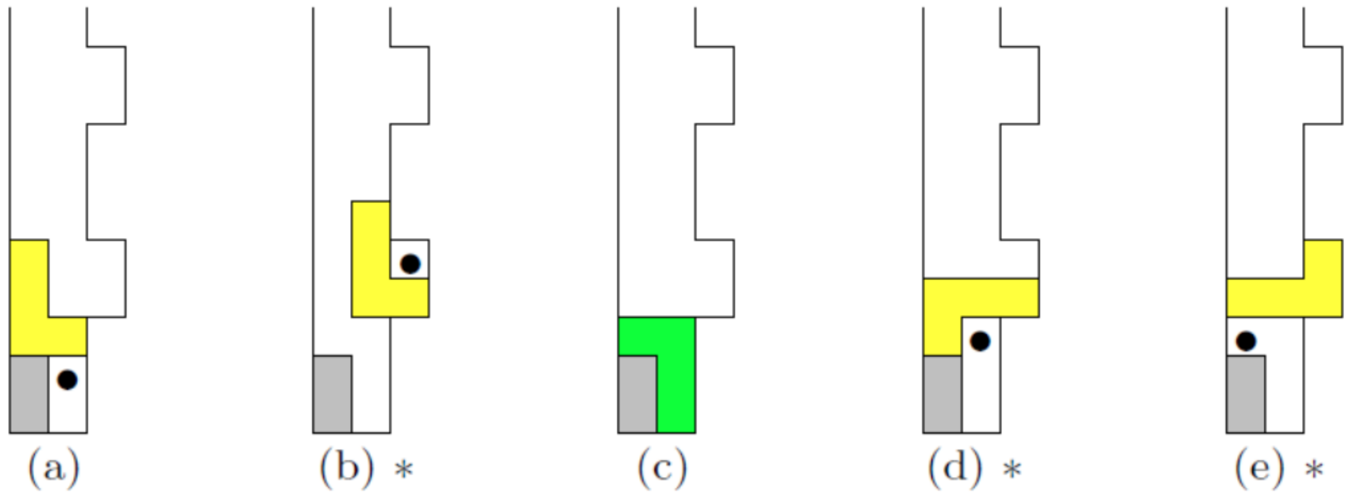
3-Partition to Tetris

Continue until the entire board is clean…



3-Partition to Tetris

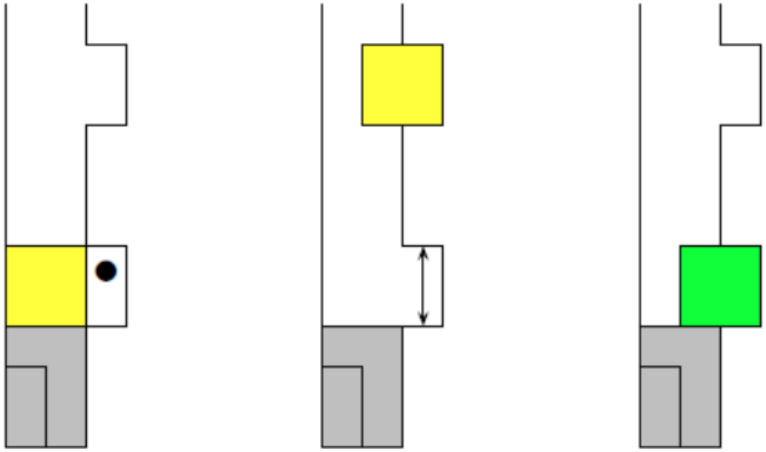
If you try to position blocks in a different way into a bucket



Forced moves - initiator

3-Partition to Tetris

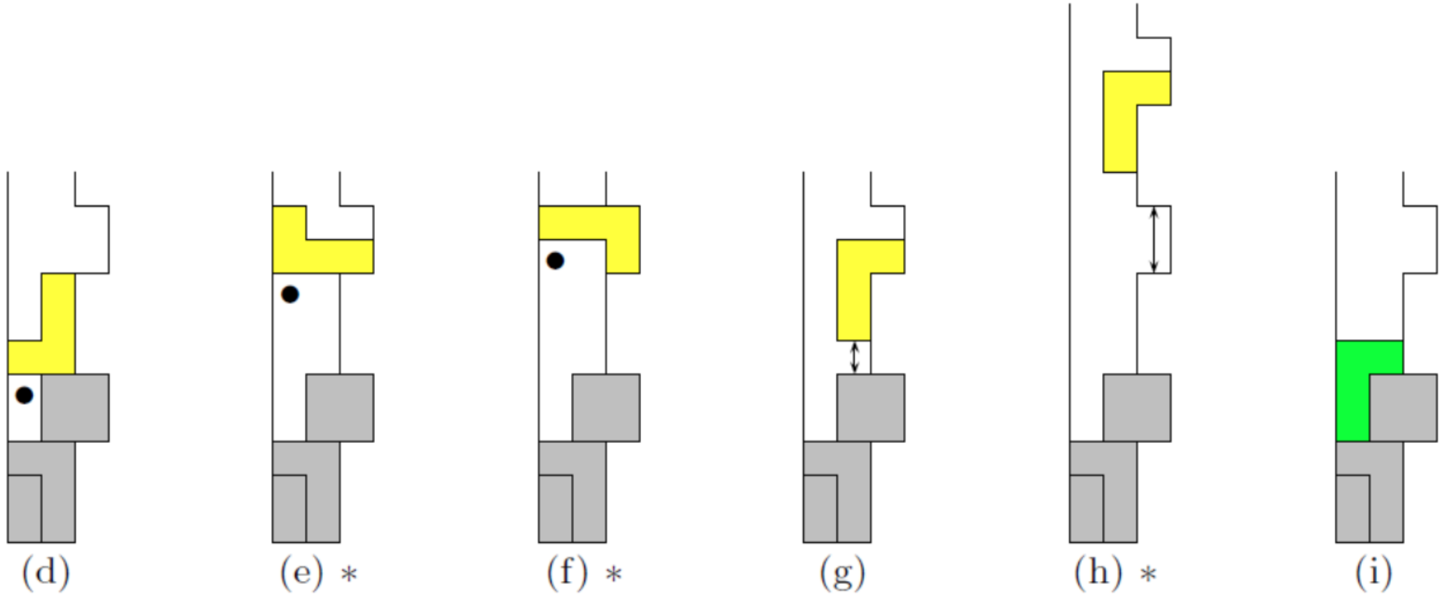
Forced moves - filler (first piece)



(a) (b) * (c)

3-Partition to Tetris

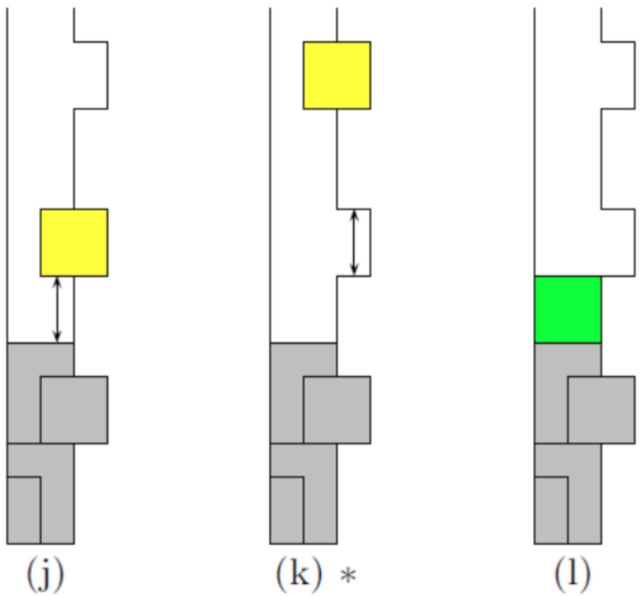
If you try to position blocks in a different way into a bucket



Forced moves – filler (second piece)

3-Partition to Tetris

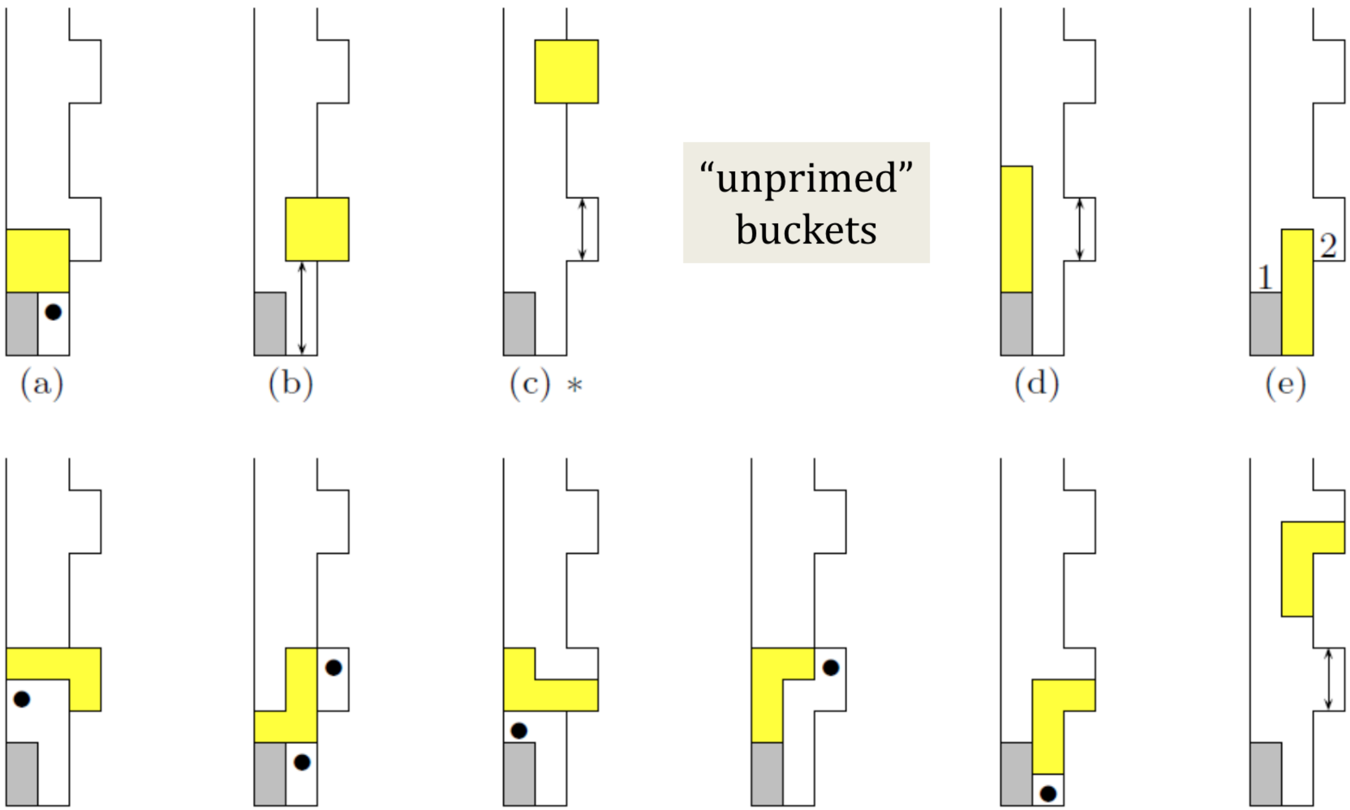
If you try to position blocks in a different way into a bucket



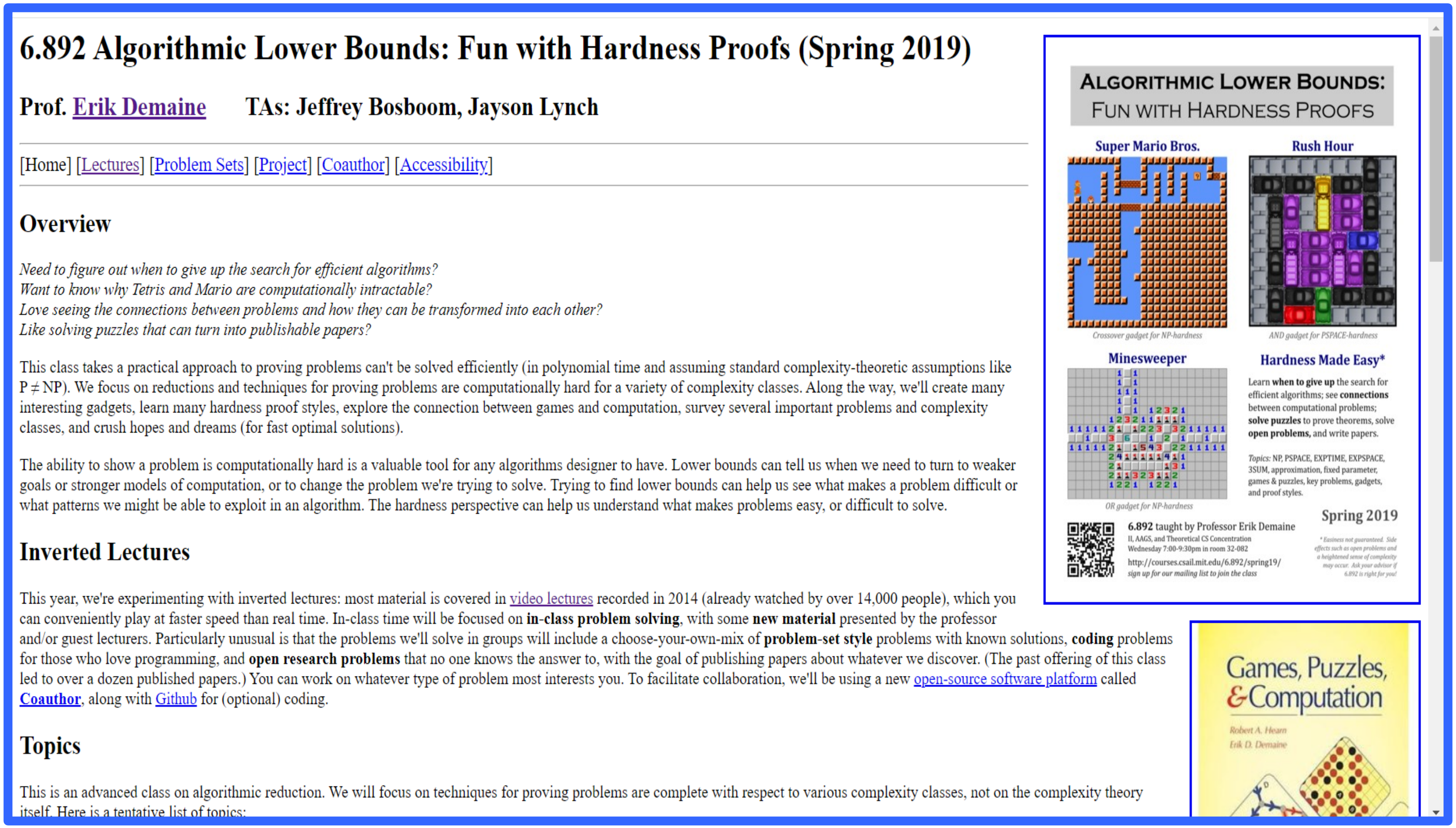
Forced moves – filler (third piece)

3-Partition to Tetris

If you try to break a element gadget



(a) (b) (c) * (d) (e)
(f) * (g) (h) * (i) (j) (k) *



6.892 Algorithmic Lower Bounds: Fun with Hardness Proofs (Spring 2019)

Prof. Erik Demaine   TAs: Jeffrey Bosboom, Jayson Lynch

[Home] [Lectures] [Problem Sets] [Project] [Coauthor] [Accessibility]

Overview

Need to figure out when to give up the search for efficient algorithms?
Want to know why Tetris and Mario are computationally intractable?
Love seeing the connections between problems and how they can be transformed into each other?
Like solving puzzles that can turn into publishable papers?

This class takes a practical approach to proving problems can't be solved efficiently (in polynomial time and assuming standard complexity-theoretic assumptions like $P \neq NP$). We focus on reductions and techniques for proving problems are computationally hard for a variety of complexity classes. Along the way, we'll create many interesting gadgets, learn many hardness proof styles, explore the connection between games and computation, survey several important problems and complexity classes, and crush hopes and dreams (for fast optimal solutions).

The ability to show a problem is computationally hard is a valuable tool for any algorithms designer to have. Lower bounds can tell us when we need to turn to weaker goals or stronger models of computation, or to change the problem we're trying to solve. Trying to find lower bounds can help us see what makes a problem difficult or what patterns we might be able to exploit in an algorithm. The hardness perspective can help us understand what makes problems easy, or difficult to solve.

Inverted Lectures

This year, we're experimenting with inverted lectures: most material is covered in video lectures recorded in 2014 (already watched by over 14,000 people), which you can conveniently play at faster speed than real time. In-class time will be focused on **in-class problem solving**, with some **new material** presented by the professor and/or guest lecturers. Particularly unusual is that the problems we'll solve in groups will include a choose-your-own-mix of **problem-set style** problems with known solutions, **coding** problems for those who love programming, and **open research problems** that no one knows the answer to, with the goal of publishing papers about whatever we discover. (The past offering of this class led to over a dozen published papers.) You can work on whatever type of problem most interests you. To facilitate collaboration, we'll be using a new open-source software platform called **Coauthor**, along with Github for (optional) coding.

Topics

This is an advanced class on algorithmic reduction. We will focus on techniques for proving problems are complete with respect to various complexity classes, not on the complexity theory itself. Here is a tentative list of topics:

MIT course (video lectures available):
https://courses.csail.mit.edu/6.892/spring19/

COMPLEXITY OF GAMES

Compendium Hosting Contributors

Compendium

Here is the list of games and puzzles that are currently in our index.
Is your favorite game missing? Are you aware of a new complexity result for one of the listed games? You are welcome add or edit the listed games by following the instructions on this page.

15-puzzle ($n^2 - 1$ puzzle)
$n^2 - 1$ numbered tiles can be slid in a $n \times n$ board with the goal of arranging them in increasing order.

Amazons
Two players move amazons on a square board. After moving, an amazon shoots an arrow that blocks movement. The last player to move wins.

Bejeweled
A player swaps adjacent items in a $n \times m$ grid in order to form as many matches of three as possible.

Boulder Dash
A single-player game in which the character digs through a rectangular grid to find diamonds within a time limit, while avoiding various dangers.

Candy Crush
A variant of Bejeweled.

Clickomania (SameGame)
A single player game in which the player removes groups of tiles of the same color in a rectangular board.

Deflektor

compendium on hardness for games and puzzles:
https://www.isnphard.com/i/