CSL 101 DISCRRETE MATHEMATICS

Dr. Barun Gorain
Department of CSE, IIT Bhilai
Email: barun@iitbhilai.ac.in

Strong Induction

Strong induction

Prove $P(0)$.

Then prove $P(n+1)$ assuming *all* of

$P(0), P(1), \ldots, P(n)$ (instead of just $P(n)$).

Conclude $\forall n.P(n)$

equivalent

Ordinary induction

$0 \to 1, 1 \to 2, 2 \to 3, \ldots, n\text{-}1 \to n$.

So by the time we got to $n+1$, already know *all* of

$P(0), P(1), \ldots, P(n)$

The point is: assuming P(0), P(1), up to P(n), it is often easier to prove P(n+1).

Prime Products

Claim: Every integer > 1 is a product of primes.

Proof: (by strong induction)

- Base case is easy.
- Suppose the claim is true for all 2 <= i < n.
- Consider an integer n.
- If n is prime, then we are done.
- So $n = k \cdot m$ for integers k, m where $n > k,m > 1$.
- Since k,m smaller than n,
- *By the induction hypothesis, both k and m are product of primes*

$$k = p_1 \cdot p_2 \cdots p_{94}$$

$$m = q_1 \cdot q_2 \cdots q_{214}$$

Prime Products

Claim: Every integer > 1 is a product of primes.

…So

$$n = k \cdot m = p_1 \cdot p_2 \cdots p_{94} \cdot q_1 \cdot q_2 \cdots q_{214}$$

is a prime product.

$\therefore$ This completes the proof of the induction step.

Postage by Strong Induction

Available stamps:

5¢

3¢

What amount can you form?

Theorem: Can form any amount ≥ 8¢

Prove by strong induction on n.

$P(n)$ ::= can form $(n + 8)$¢.

Postage by Strong Induction

Base case ($n = 0$):

$(0 + 8)$¢:

Inductive Step: assume $(m + 8)$¢ for $0 \le m \le n$,

then prove $((n + 1) + 8)$¢

cases:

$n + 1 = 1$, 9¢:

$n + 1 = 2$, 10¢:

Postage by Strong Induction

case $n+1 \geq 3$: let $m = n - 2$.

now $n \geq m \geq 0$, so by induction hypothesis have:

$+$

$= (n+1)+8$

3

$(n-2)+8$

We're done!

In fact, use at most two 5-cent stamps!

Postage by Strong Induction

Given an unlimited supply of 5 cent and 7 cent stamps,

what postages are possible?

Theorem: For all n >= 24,

it is possible to produce n cents of postage from 5¢ and 7¢ stamps.

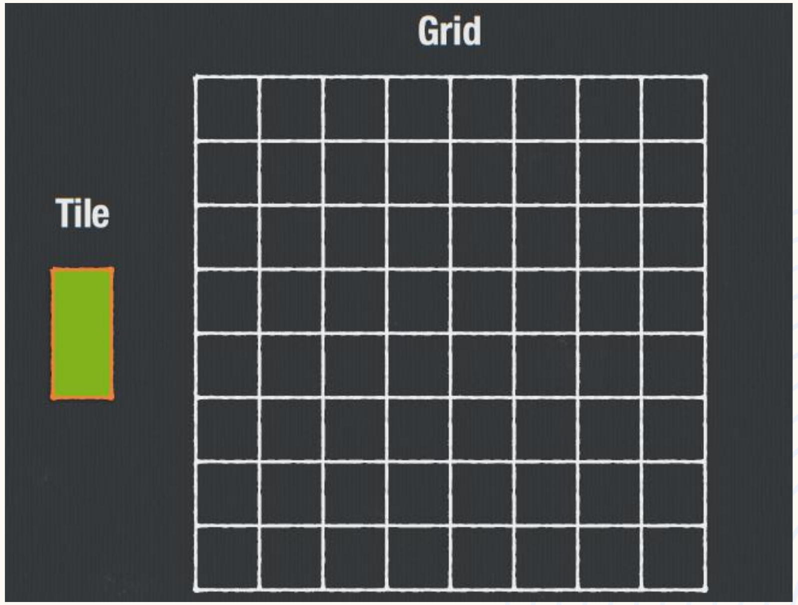
Invariants

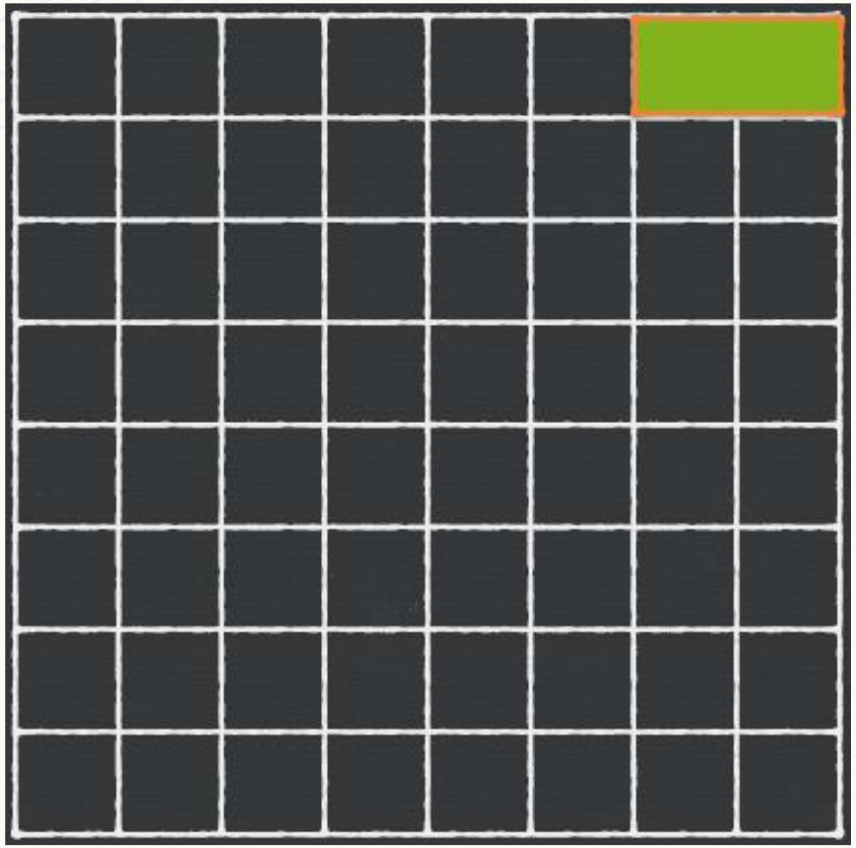
An warm-up example

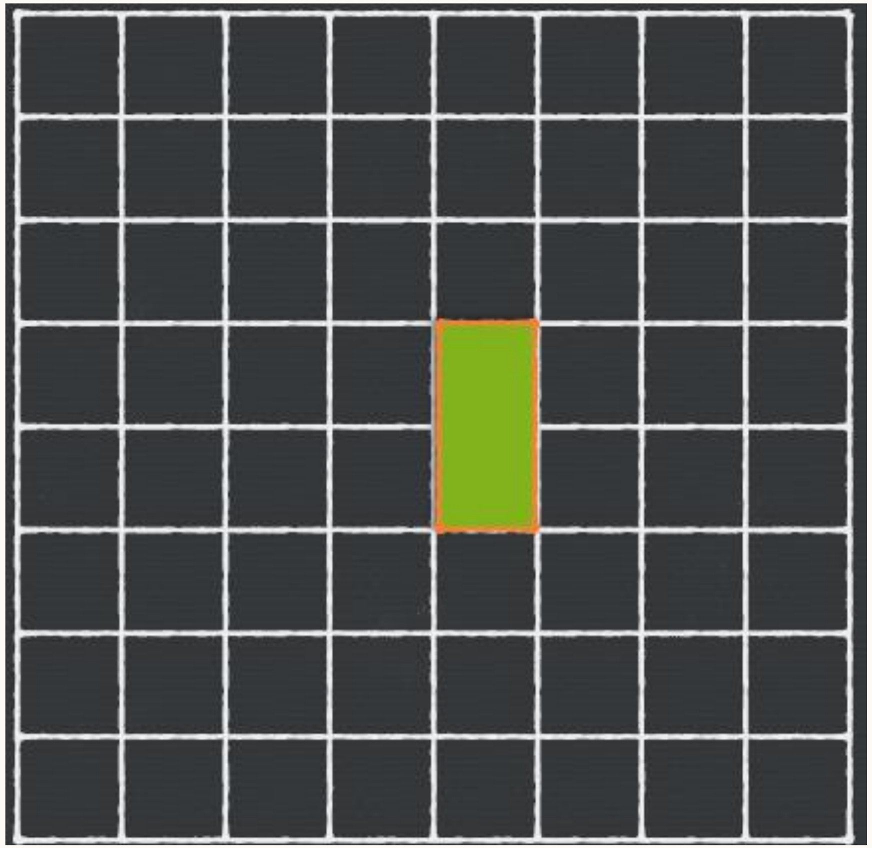
- Consider an 8 x 8 grid of squares.
- You are given a set of tiles. The size of each tile covers exactly 2 squares of the grid.
- You can only place a tile horizontally or vertically, and each tile must cover exactly 2 grid squares (e.g., can't have a tile hanging off the edge, can't cover half a grid square, etc)
- No overlapping tiles allowed.

An warm-up example

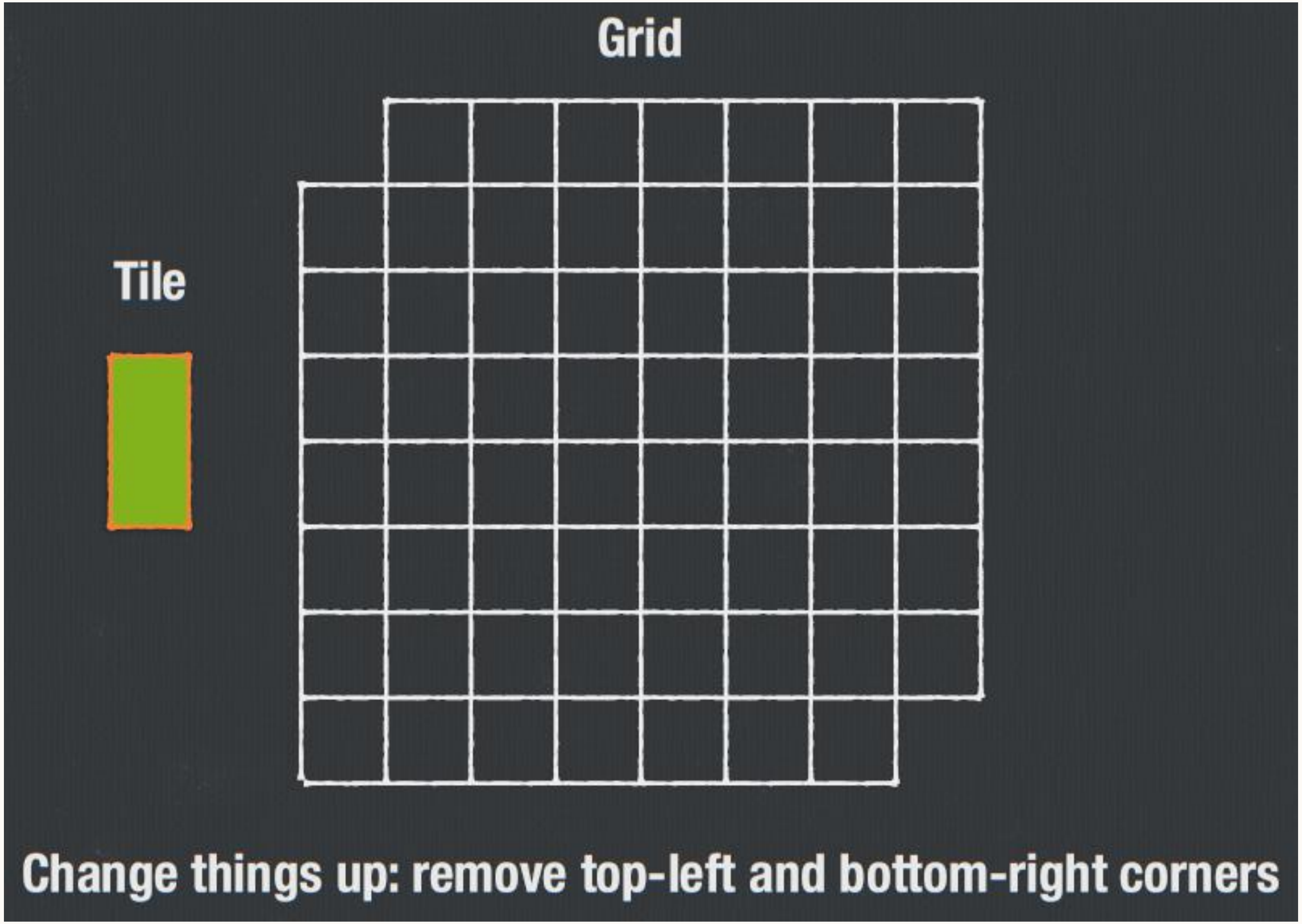
- Consider an 8 x 8 grid of squares.
- You are given a set of tiles. The size of each tile covers exactly 2 squares of the grid.
- You can only place a tile horizontally or vertically, and each tile must cover exactly 2 grid squares (e.g., can't have a tile hanging off the edge, can't cover half a grid square, etc)
- No overlapping tiles allowed.

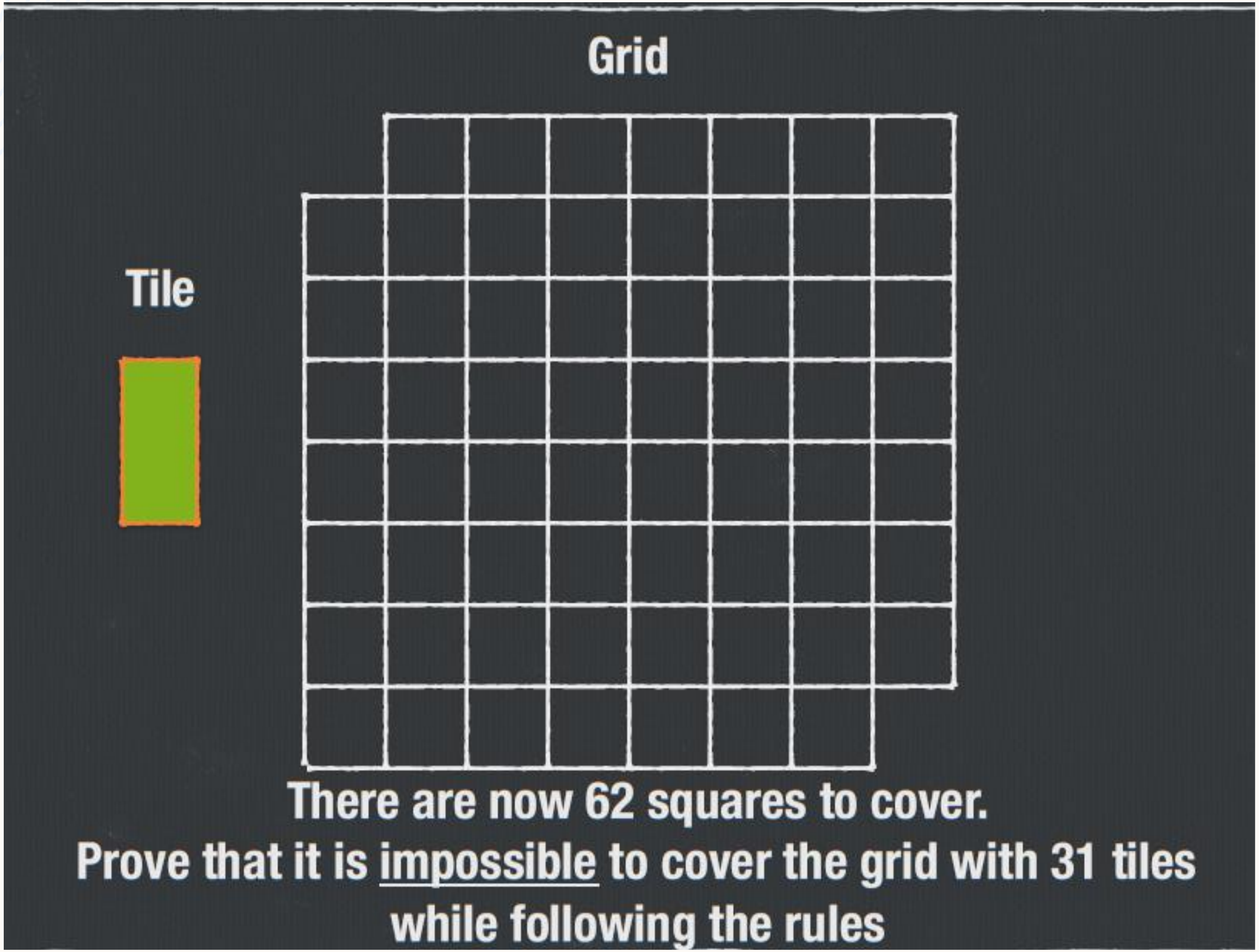


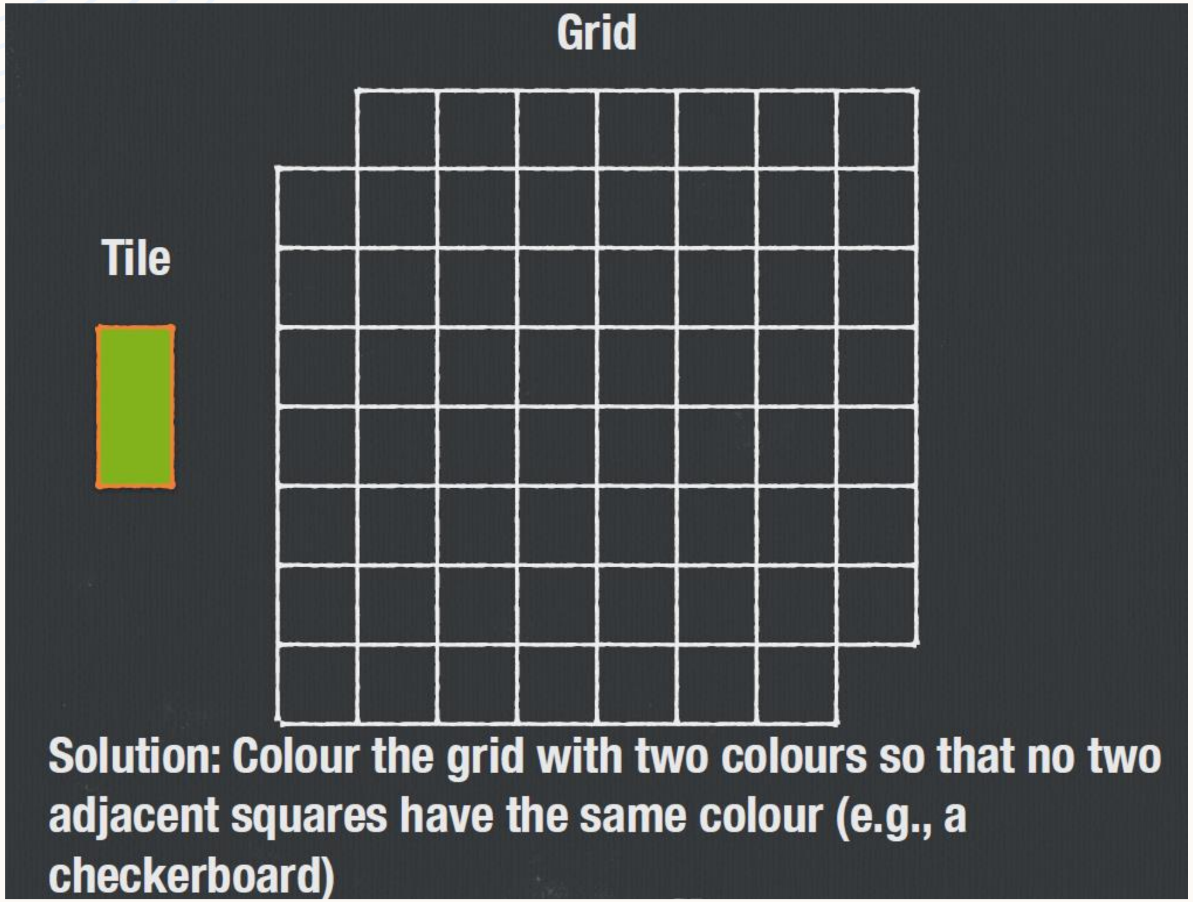


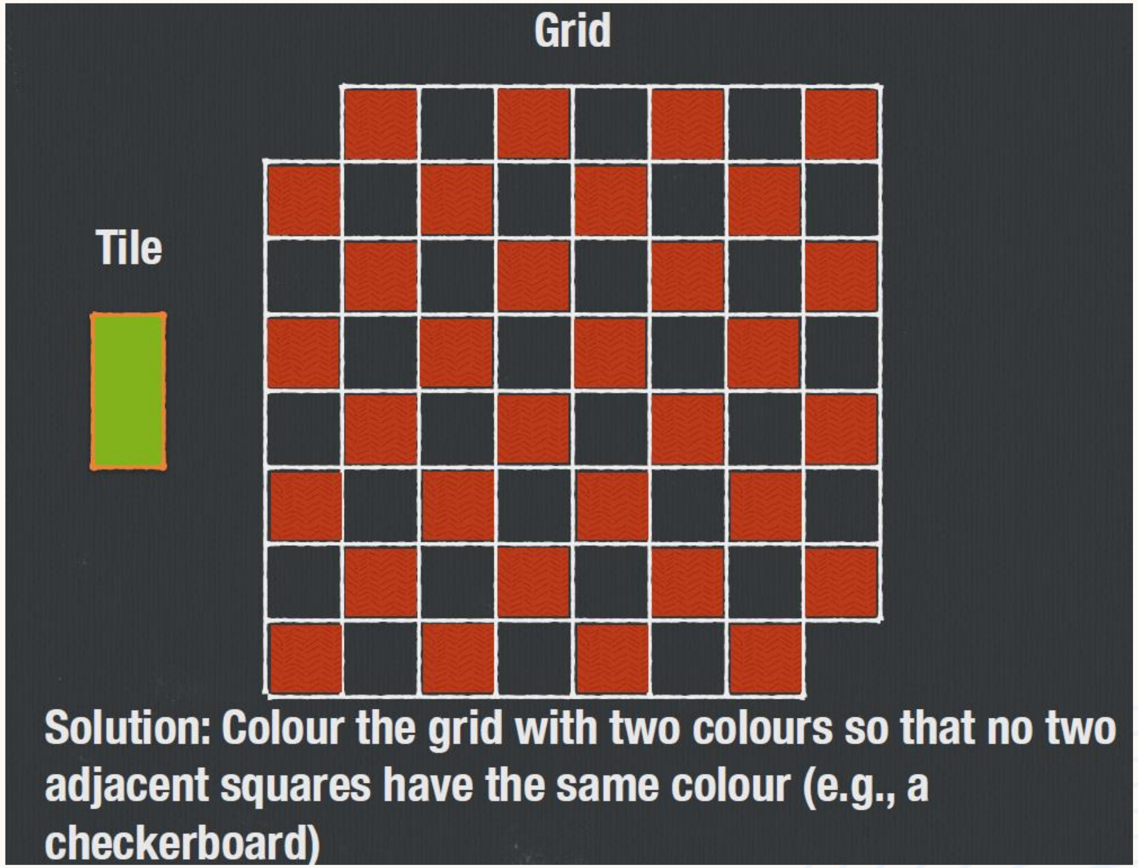


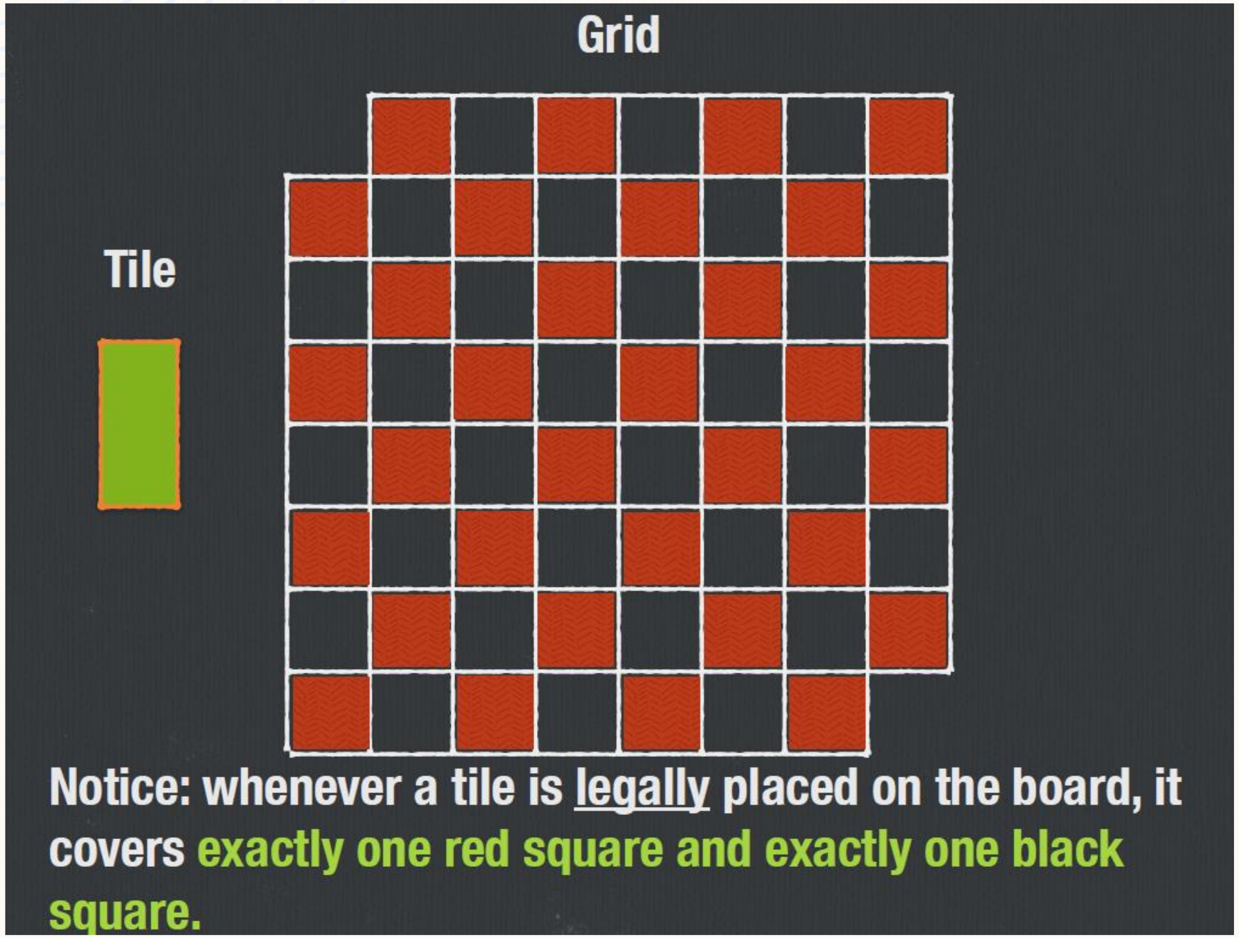
It’s easy to cover the entire grid with tiles

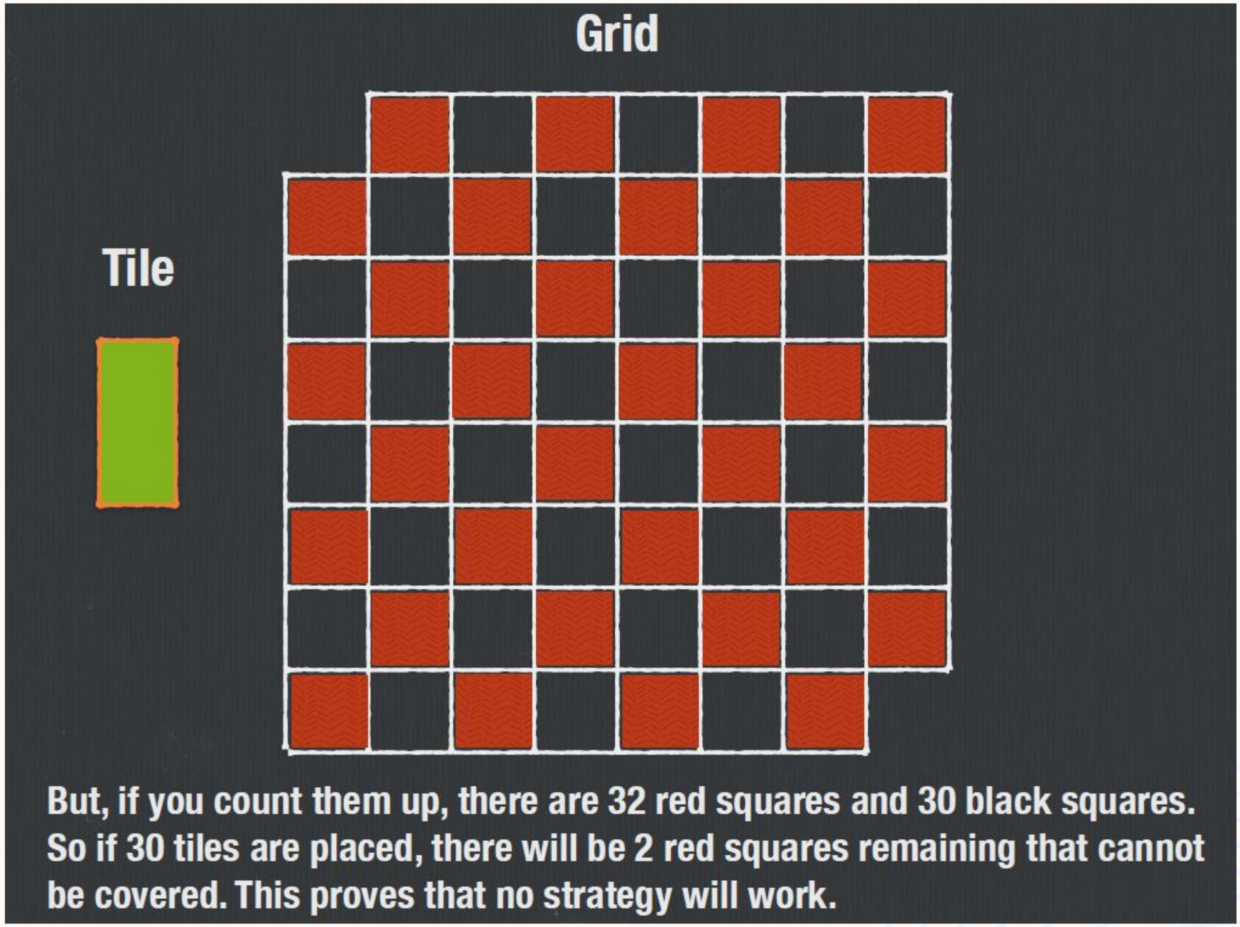

Grid
Tile
Change things up: remove top-left and bottom-right corners


Grid
Tile
There are now 62 squares to cover.
Prove that it is impossible to cover the grid with 31 tiles
while following the rules


Grid
Tile
Solution: Colour the grid with two colours so that no two adjacent squares have the same colour (e.g., a checkerboard)


Grid
Tile
Solution: Colour the grid with two colours so that no two adjacent squares have the same colour (e.g., a checkerboard)


Grid
Tile
Notice: whenever a tile is legally placed on the board, it covers exactly one red square and exactly one black square.


Grid
Tile
But, if you count them up, there are 32 red squares and 30 black squares. So if 30 tiles are placed, there will be 2 red squares remaining that cannot be covered. This proves that no strategy will work.

States of an algorithm

- Note: start fresh. We're currently not in any particular model.
- Consider any algorithm or process that is occurring in any system.
- At the beginning, nothing has happened yet, and we call this the <u>initial state</u> C_0 (or initial configuration)
- Then the algorithm starts.

  Referring back to the game analogy, an algorithm performs instructions that can be thought of as "legal moves" according to the rules of the model.

States of an algorithm

- Each time that a legal move is performed, we can think of the algorithm <u>transitioning</u> to a new state C_i
- So an execution of an algorithm looks like:

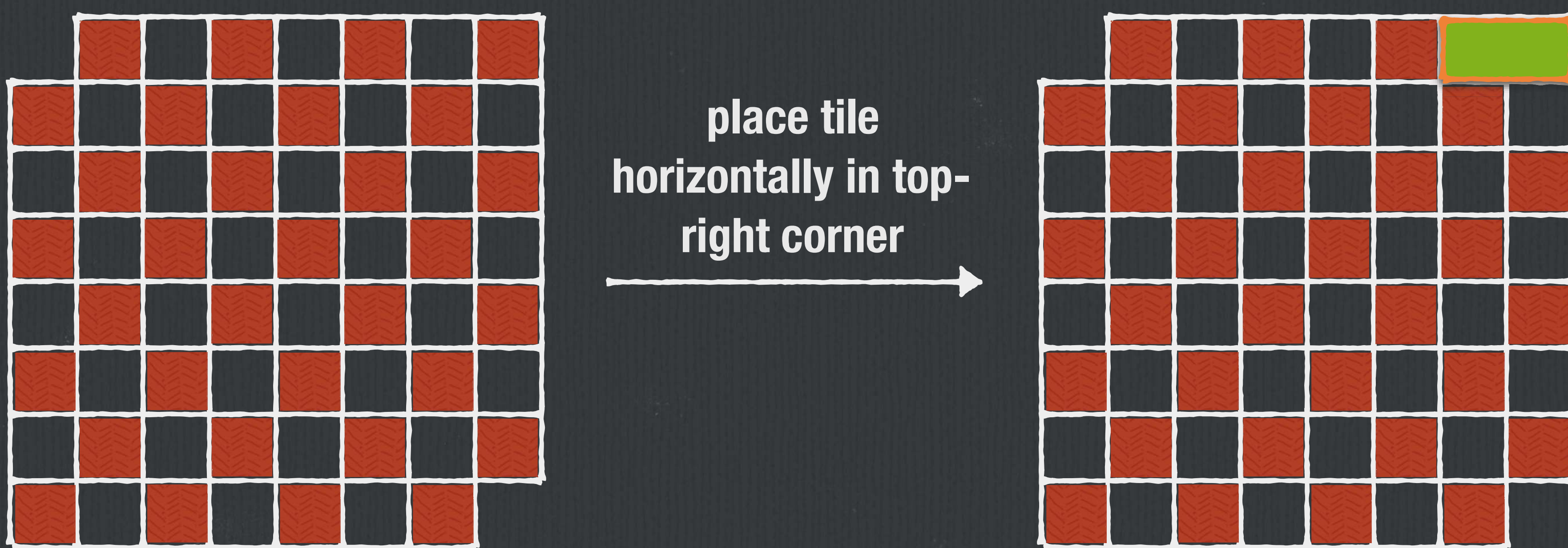
  $C_0 \xrightarrow{\alpha_1} C_1 \xrightarrow{\alpha_2} C_2 \xrightarrow{\alpha_3} \ldots$

  Where α_i was the move performed that made the algorithm transition from state C_{i-1} to state C_i

States of an algorithm

- Basically, it's a way of thinking about how the algorithm makes the system change, but one step at a time.
- Each state is a frozen moment in time, and we look at how changes happen one move at a time.
- Keep in mind: the next state doesn't always have to be different then previous state. Sometimes a legal move might have no effect on the system.

States of an algorithm



C_0 C_1

Details about the state

- When looking at a fixed state of the algorithm, we can consider various details about it.
- In the tiles-on-grid example, when looking at one state, we can ask
  - “How many uncovered squares are there?”
  - “How many rows have no squares covered?”
  - “Is the number of covered squares in each row odd or even?”
  - There are many possibilities.

Invariants

- As the algorithm makes moves, some of the details change.
- However, some might not! These are called <u>invariants</u>.
- An invariant is a statement that is <u>always</u> true. More specifically:
  - It is true about the initial state C_0.
  - For any state C_i in which the invariant holds, for any legal move α_{i+1}, the invariant holds for state C_{i+1}

Why invariants are useful

- If we can prove an invariant of the algorithm, and this invariant contradicts every correct solution to the problem, then the algorithm is not correct.
- But: it's not always obvious what that invariant should be.

Tiles-On-Grid

- Recall the problem: we have an 8x8 grid but with the top-left and bottom-right corners removed.
- Model: place one tile at a time, horizontally or vertically, so that each covers exactly two adjacent grid squares.
- To prove: It is impossible to cover the 62 grid squares using 31 tiles.

Tiles-On-Grid

- The proof I showed you in Lecture 2 is actually a proof using an invariant.
- First, colour the grid like a checkerboard using colours black and red.
- We prove the invariant: “the number of red squares minus the number of black squares is 2”.

Proof of the invariant
(using induction)

- Let red_i = number of uncovered red squares in state C_i

  Let $black_i$ = number of uncovered black squares in state C_i

- (Base Case)

  In the initial state, the number of uncovered red squares is 32 and the number of uncovered black squares is 30,

  so $red_0 - black_0 = 2$.

- (Induction Hypothesis)

  Assume that the invariant holds for state C_i, $i \geq 0$

  that is, $red_i - black_i = 2$

Proof of the invariant

- (Inductive step)

  For any legal move, a tile covers exactly two adjacent squares, so one is red and one is black. This means, in state C_{i+1}:

  red_{i+1} - $black_{i+1}$

  = (red_i -1) - ($black_i$ -1)

  = red_i - $black_i$

  = 2

- By induction, the variant holds for all configurations C_i

Finishing the proof

- From our proven invariant, we know:

  When the algorithm terminates, it is in some state C_i such that

  $red_i - black_i = 2$

- But, in any valid solution, $red_i = 0$ and $black_i = 0$ since all squares must be covered, so

  $red_i - black_i = 0$ in any correct solution.

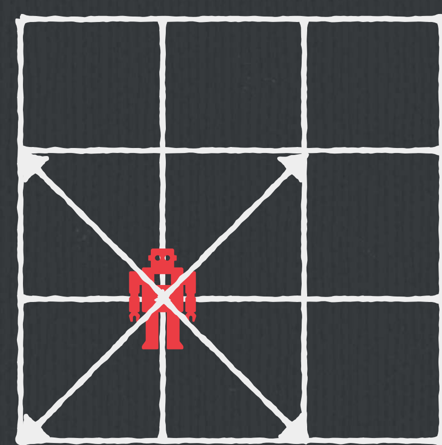
- This proves that the algorithm is incorrect.

New example: A diagonally-moving robot

- Model: a robot is standing at (0,0) in $\mathbb{R}^2$. The robot is only allowed to move diagonally, changing its coordinates by 1 unit each.

- In other words, if the robot is at position (x,y), in one move it must go to one of:

  1. (x+1,y+1)
  2. (x+1,y-1)
  3. (x-1,y+1)
  4. (x-1,y-1)



Impossibility result

- Prove: the robot cannot reach point (1,0)
- Incorrect proof:

  The robot starts at point (0,0), and going to (1,0) means that it changed its x-coordinate but not its y-coordinate, which isn't a legal move.
- The above would prove that it is impossible in <u>one</u> move, but we have to show it can <u>never</u> reach (1,0), no matter how many moves it makes!

Proof by invariant

- Find something about the robots state that doesn’t change, no matter which legal move it makes.
- The x-coordinate and y-coordinate can change, so they are not invariant.
- The sum or difference or product or quotient of x and y can change, so they are not invariant.

Proof by invariant

- Notice: both coordinates change by 1 with every legal move.
- So the <u>parity of the sum of x and y</u> is always the same. In other words, $x+y$ is always even or always odd.

Invariant: sum of x and y is even

- Base case: in the initial state C_0, the robot is at (0,0), and 0+0=0 is even.
- Assume that, for some state C_i, $i \geq 0$, the robot's position (x,y) satisfies the property that x+y is even.
- Consider any legal move and the resulting state C_{i+1}.

Invariant: sum of x and y is even

- Case 1: robot moves to $(x+1,y+1)$

  Check: $(x+1)+(y+1) = x+y+2$

  and since we assumed $x+y$ is even, it follows that $x+y+2$ is even.

- The other cases are:

  $(x+1)+(y-1) = x+y$, which is even

  $(x-1)+(y+1) = x+y$, which is even

  $(x-1)+(y-1) = x+y-2$, which is even.

Proof that (1,0) is unreachable

- By induction, for all states C_i, $i \geq 0$, we have the invariant "x+y is even".
- However, position (1,0) has $x+y = 1+0 = 1$, which is odd. This proves that the robot will never reach (1,0).

Two Jugs Puzzle

- Model: you are given two jugs, one holds 3 gallons of water, the other holds 9 gallons of water. You have a faucet which you can use to fill the jugs with water.
- Problem: Fill the 9-gallon jug with <u>exactly</u> 4 gallons of water.

Legal moves

- Other than filling a jug to the top, we have no way of using the faucet to directly fill a jug with an exact quantity. So we can:

    1. Fill or empty the 3-gallon jug completely
    2. Fill or empty the 9-gallon jug completely
    3. Pour the 3-gallon jug into the 9-gallon jug (and stop when the 9-gallon jug is full)
    4. Pour the 9-gallon jug into the 3-gallon jug (and stop when the 3-gallon jug is full)

States

- The state at any time is (B,L), where B is the amount of water in the big jug, and L is the amount of water in the little jug.
- Initially, both jugs are empty, so we are in state (0,0).

Transitions

- Fill little jug: (B,L) $\longrightarrow$ (B,3)
- Fill big jug: (B,L) $\longrightarrow$ (9,L)
- Empty little jug: (B,L) $\longrightarrow$ (B,0)
- Empty big jug: (B,L) $\longrightarrow$ (0,L)

Transitions

- Pour big jug into little jug:

  1. if $B+L \leq 3$, then $(B,L) \longrightarrow (0, B+L)$

  2. if $B+L > 3$, then $(B,L) \longrightarrow (B - (3-L), 3)$

- Pour little jug into big jug:

  1. if $B+L \leq 9$, then $(B,L) \longrightarrow (B+L,0)$

  2. if $B+L > 9$, then $(B,L) \longrightarrow (9, L - (9 - B))$

Impossibility

- So the question:

  "starting with empty jugs can we put exactly 4 gallons of water in the big jug?"

  becomes:

  "starting from state (B,L)=(0,0), is state (4,L) reachable for any L?

- We prove the following invariant:

  Both B and L are always a multiple of 3.

- Let B_i and L_i represent the values of B and L in state C_i.

- <u>Base case:</u> initial state is (0,0), and 0 is a multiple of 3, so B_0 and L_0 are both multiples of 3.

Impossibility

- (Induction Hypothesis)

  Assume, for some state C_i, $i \geq 0$, that B_i and L_i are multiples of 3.

- (Inductive Step)

  Consider each legal move and look at the state C_{i+1} reached:

  - If small jug filled completely: $(B_{i+1},L_{i+1}) = (B_i,3)$
  - If big jug filled completely: $(B_{i+1},L_{i+1}) = (9,L_i)$
  - If small jug emptied completely: $(B_{i+1},L_{i+1}) = (B_i,0)$
  - If big jug emptied completely: $(B_{i+1},L_{i+1}) = (0,L_i)$

- The invariant holds in these cases, since 0, 3, 9, B_i and L_i are multiples of 3.

Impossibility

- Pour big jug into little jug:

  1. if $B_i+L_i \leq 3$, then $(B_{i+1},L_{i+1}) \longrightarrow (0, B_i+L_i)$

  2. if $B_i+L_i > 3$, then $(B_{i+1},L_{i+1}) \longrightarrow (B_i - (3-L_i), 3)$

- 0, 3, and sums and differences involving only multiples of 3: all multiples of 3

Impossibility

- Pour little jug into big jug:

  1. if $B_i+L_i \leq 9$, then $(B_{i+1},L_{i+1}) \longrightarrow (B_i+L_i,0)$

  2. if $B_i+L_i > 9$, then $(B_{i+1},L_{i+1}) \longrightarrow (9, L_i - (9 - B_i))$

- 0, 3, and sums and differences involving only multiples of 3: all multiples of 3

Impossibility

- So in all cases, a legal move results in the new state (B_{i+1},L_{i+1}) where both entries are multiples of 3.
- Since 4 isn't a multiple of 3, we get that (4,L) is unreachable, so it is impossible to fill the 9-gallon jug with exactly 4 gallons of water.

If you're bored…

If the two jugs held 3 gallons and 5 gallons, it <u>is</u> possible to fill the 5-gallon jug with exactly 4 gallons of water.

Can you figure out how?

Can you generalize which cases are possible and which are impossible?