Mathematical Olympiad

To the geometry and beyond...

Mirek Olšák

- 2017: AlphaGo
- 2018: Who cares about Euclidean Geometry?
- 2020: Grand IMO Challenge
- 2021: MiniF2F benchmark
- 2022: chatGPT
- 2024
  - AlphaGeometry
  - AI-MO challenge
  - Math Olympiad solver (Numina)
  - PutnamBench
  - AlphaProof

- 2017: AlphaGo
- 2018: Who cares about Euclidean Geometry?
- 2020: Grand IMO Challenge
- 2021: MiniF2F benchmark
- 2022: chatGPT
- 2024
  - AlphaGeometry
  - AI-MO challenge
  - Math Olympiad solver (Numina)
  - PutnamBench
  - AlphaProof

IMO (International Mathematical Olympiad)

- Prestigious competition for pre-university students
- since 1959
- max 6 competitors per country, 108 participant countries last year
- 4 domains:
  - Geometry
  - Number Theory
  - Algebra
  - Combinatorics

Why IMO?

- The most curated problem set
  - Theoretically solvable
  - Novel problems
- Mathematicians will understand you

Geometry

- The easiest IMO domain
- 1996 / 2000: Deduction database / Full Angle (Chou et al)
  - ATP geometry methods
- 2018: AITP talk
- 2020: GeoLogic:
  - ITP for IMO-style geometry
- 2024: AlphaGeometry (DeepMind, Trinh et al)
  - Solves 83% of all historical IMO geometry problems from the past 25 years

Geometry key components

- GeoLogic
  - Semi-formal logic
  - Conveniently strong automation
  - Lemmata
- AlphaGeometry
  - Semi-formal logic
  - Even stronger automation
  - Training on synthetic data

- Why didn't I get to AlphaGeometry?
  - I was just a mathematician / idealist

Geometry key components

- GeoLogic
  - Semi-formal logic
  - Conveniently strong automation
  - Lemmata
- AlphaGeometry
  - Semi-formal logic
  - Even stronger automation
  - Training on synthetic data

- Why didn't I get to AlphaGeometry?
  - I was just a mathematician / idealist

Geometry key components

- GeoLogic
  - Semi-formal logic
  - Conveniently strong automation
  - Lemmata
- AlphaGeometry
  - Semi-formal logic
  - Even stronger automation
  - Training on synthetic data

- Why didn’t I get to AlphaGeometry?
  - I was just a mathematician / idealist

So, is geometry done?

Geometry is a toy domain

- The main purpose is a playground for experiments with ML / logic

Geometry	General math
Construction	Functional program
Predicate description	Logical program
Using a diagram	Using a model
Semiformal to formal	Informal to formal
Compositionality (lemmata)	Compositionality (lemmata)

AlphaProof

- Tactic prediction for Lean
- Trained on ~1M autoformalized examples
- Reinforcement-learning based
- RL loop also involved while solving a particular problem
- Solved Algebra & Number theory problems from IMO 2024 (P1, P2, P6)

Are we close to singularity :-)

3 years after AlphaGo? …

Are we close to singularity :-)

3 years after AlphaGo? … didn't work out

Are we close to singularity :-)

3 years after AlphaGo? … didn’t work out

3 years after IMO? …

Are we close to singularity :-)

3 years after AlphaGo? … didn’t work out

3 years after IMO? …

It is hard to make predictions,

especially about the future.

Are we close to singularity :-)

3 years after AlphaGo? … didn’t work out

3 years after IMO? …

It is hard to make predictions,

especially about the future.

Important note: Winning gold ≠ superhuman

IMO categories in a nutshell

(quoting Štěpán Šimsa)

- Geometry = Imagination
- Algebra = Calculation
- Number theory = Knowledge
- Combinatorics = Thinking

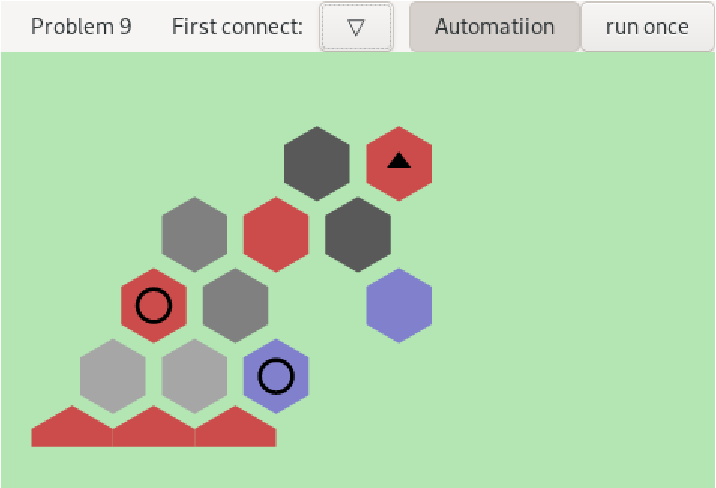
Combinatorics still hard

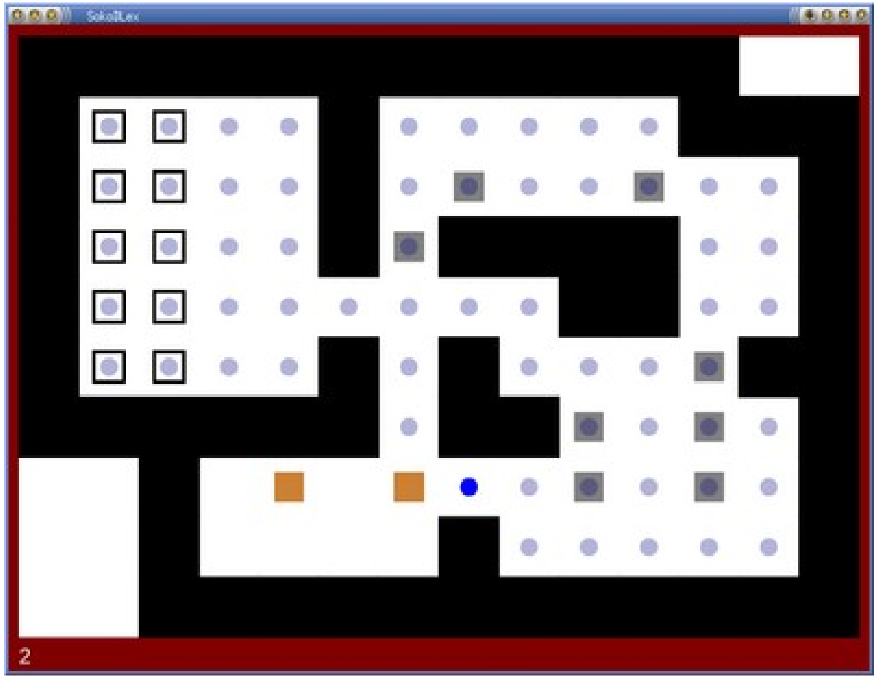
I am still a mathematician, idealist...

Let me do what I did with Geometry

Games

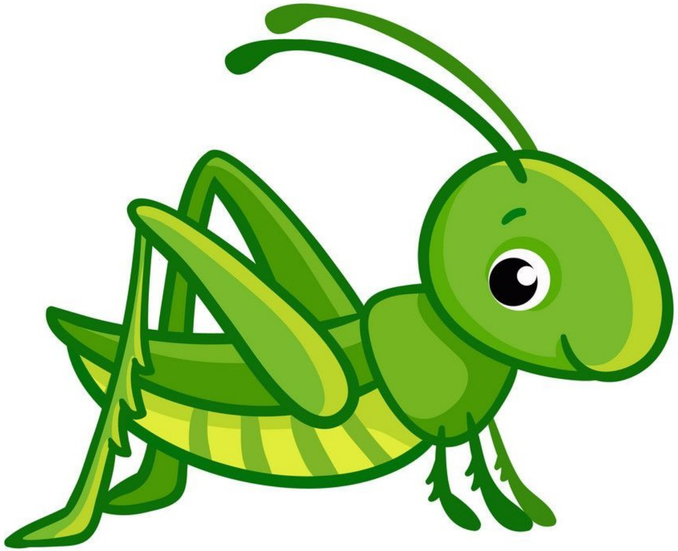
- Hex, Sokoban (PSPACE-complete)
- Case split
- CDCL style





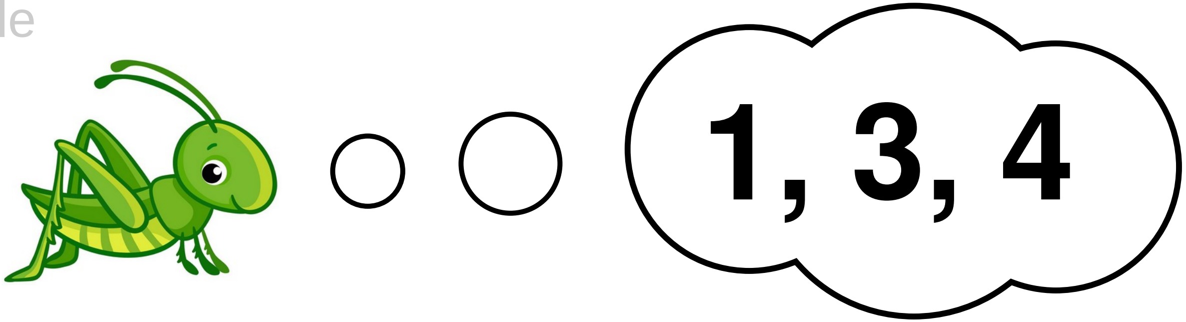
Grasshopper problem

- n available jumps – a finite (distinct) set of positive integers
- n-1 mines – a subset of points {1, 2, ..., sum(jumps)-1}
- Grasshopper wants to
  - get from 0 to sum(jum
  - use each available ju
  - do not hit any mine
- Task: Prove it is always



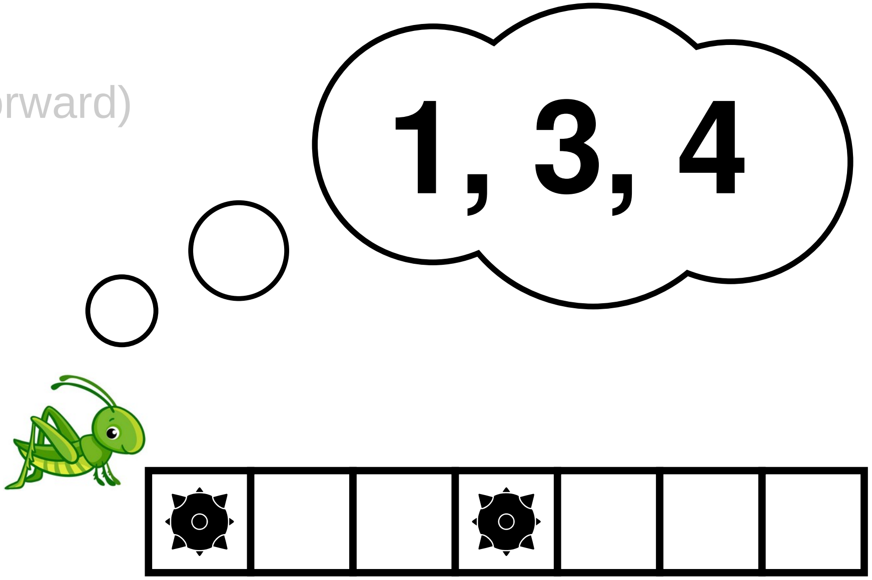
Grasshopper problem

- n available jumps – a finite (distinct) set of positive integers
- n-1 mines – a subset of points {1, 2, ..., sum(jumps)-1}
- Grasshopper wants to
  - get from 0 to sum(jumps)
  - use each available jump exactly once (forward)
  - do not hit any mine
- Task: Prove it is always possible



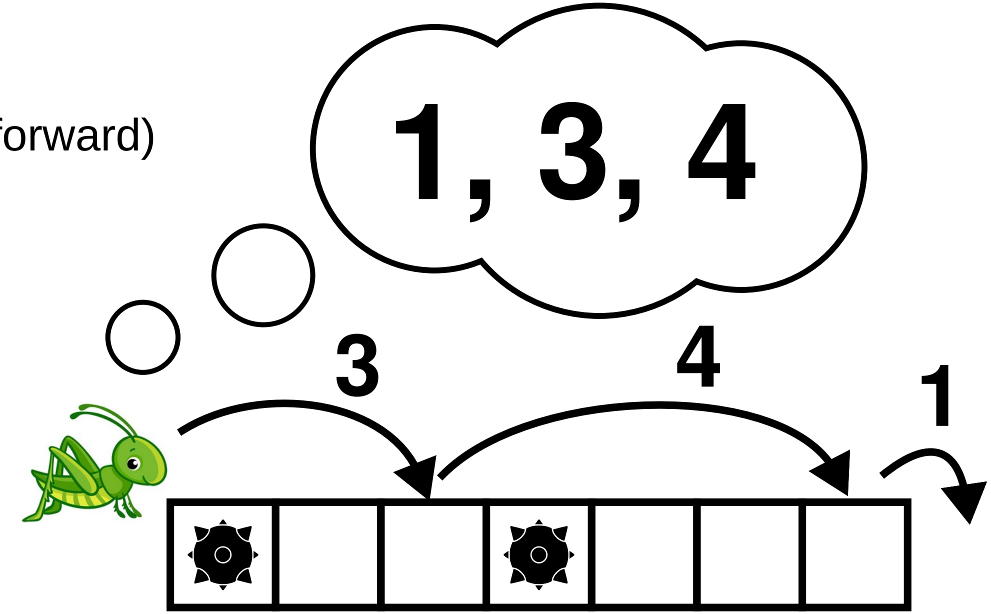
Grasshopper problem

- n available jumps – a finite (distinct) set of positive integers
- n-1 mines – a subset of points {1, 2, ..., sum(jumps)-1}
- Grasshopper wants to
  - get from 0 to sum(jumps)
  - use each available jump exactly once (forward)
  - do not hit any mine
- Task: Prove it is always possible



Grasshopper problem

- n available jumps – a finite (distinct) set of positive integers
- n-1 mines – a subset of points {1, 2, ..., sum(jumps)-1}
- Grasshopper wants to
  - get from 0 to sum(jumps)
  - use each available jump exactly once (forward)
  - do not hit any mine
- Task: Prove it is always possible

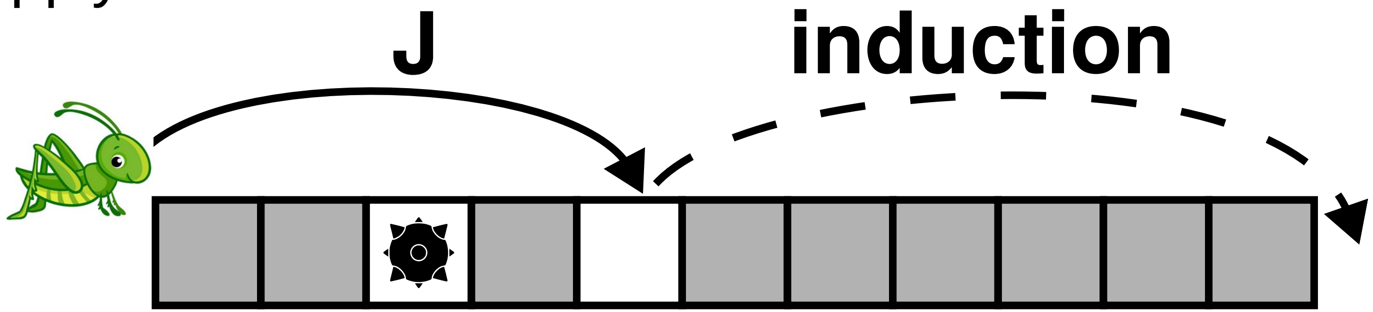


Solution process

- (high level) guess induction step at the start
- Since then, we play a “minigame”
  - Construction-based (like geometry)
  - Convenient enough automation
    - Custom instantiation & SMT LIA
  - Automatic case split when automation fails
  - CDCL not as essential?
  - Model is useful at least for rendering

Grasshopper solution

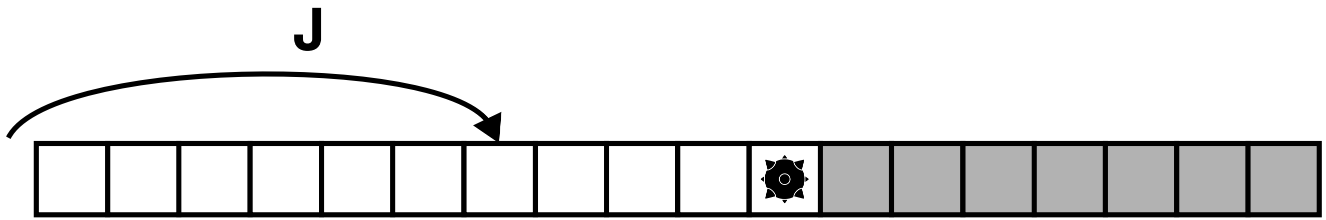
- Start with the biggest jump J
  - if we jump over at least one mine and don't land on any, we can apply induction



- Two possible problems
  - all mines far away
  - biggest jump lands on a mine

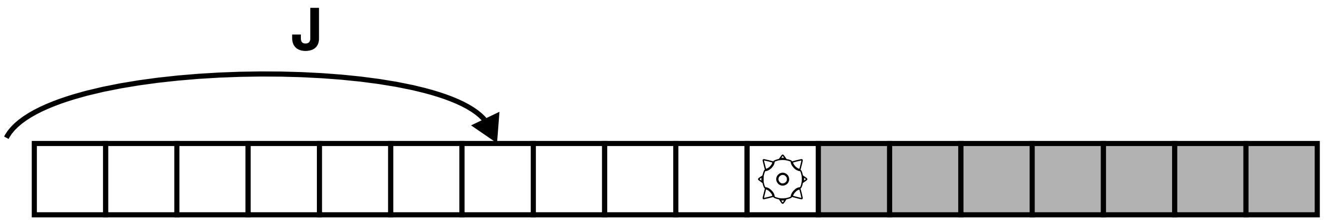
Solution – far away

- Remove first mine
- Apply induction
- Insert the largest jump to fix the solution



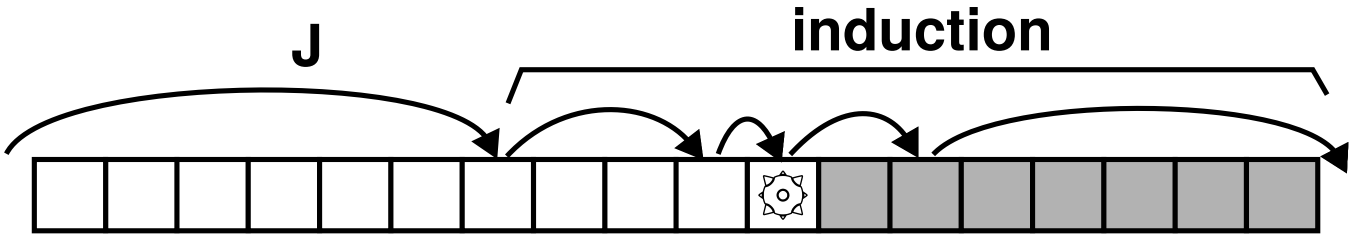
Solution – far away

- Remove first mine
- Apply induction
- Insert the largest jump to fix the solution



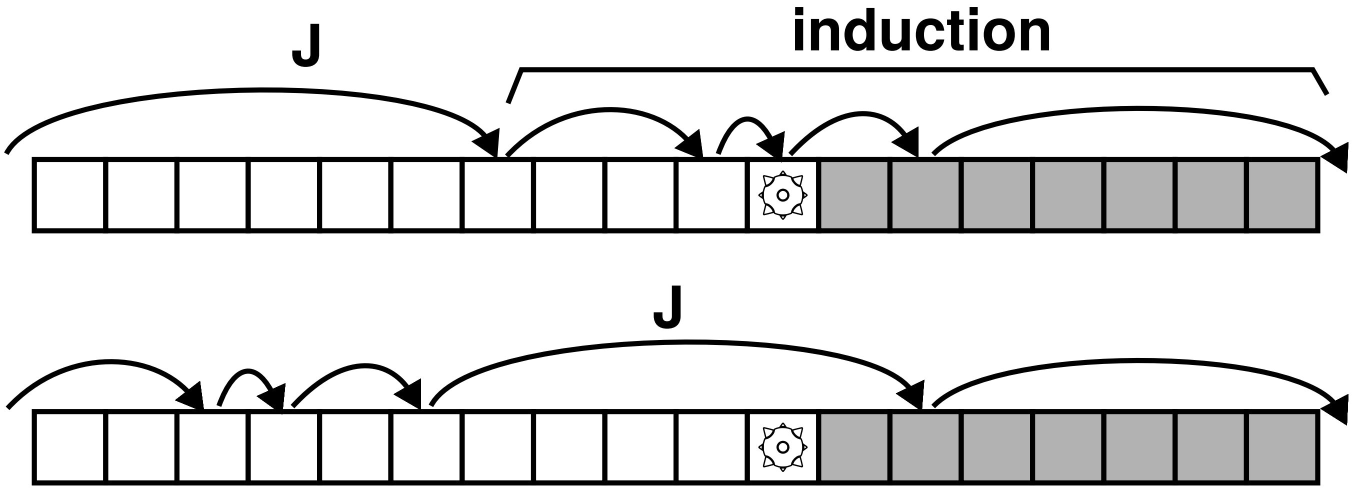
Solution – far away

- Remove first mine
- Apply induction
- Insert the largest jump to fix the solution



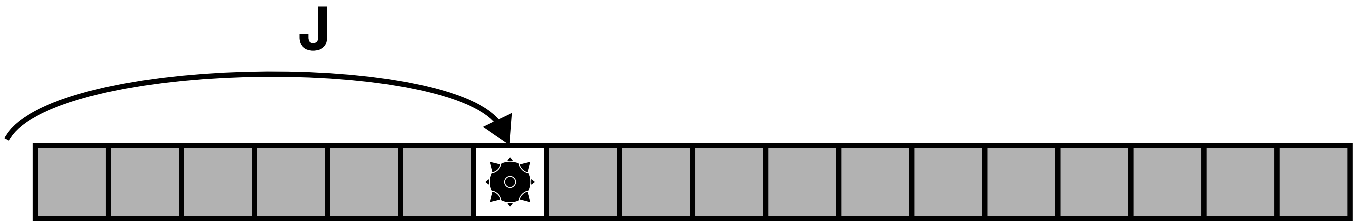
Solution – far away

- Remove first mine
- Apply induction
- Insert the largest jump to fix the solution



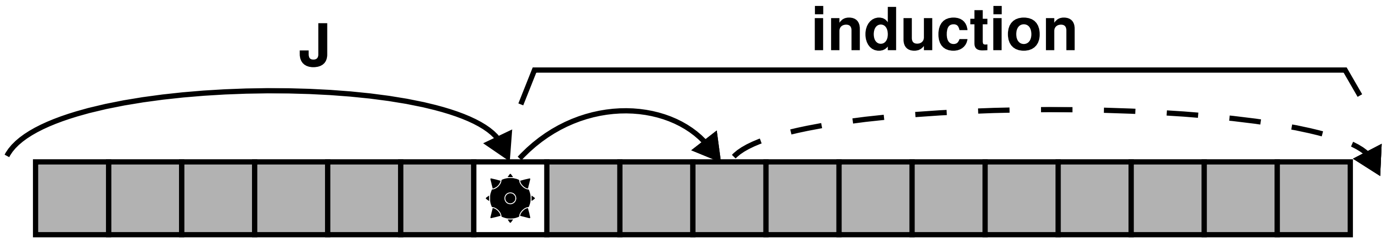
Solution – J lands on mine

- Try an analogous approach



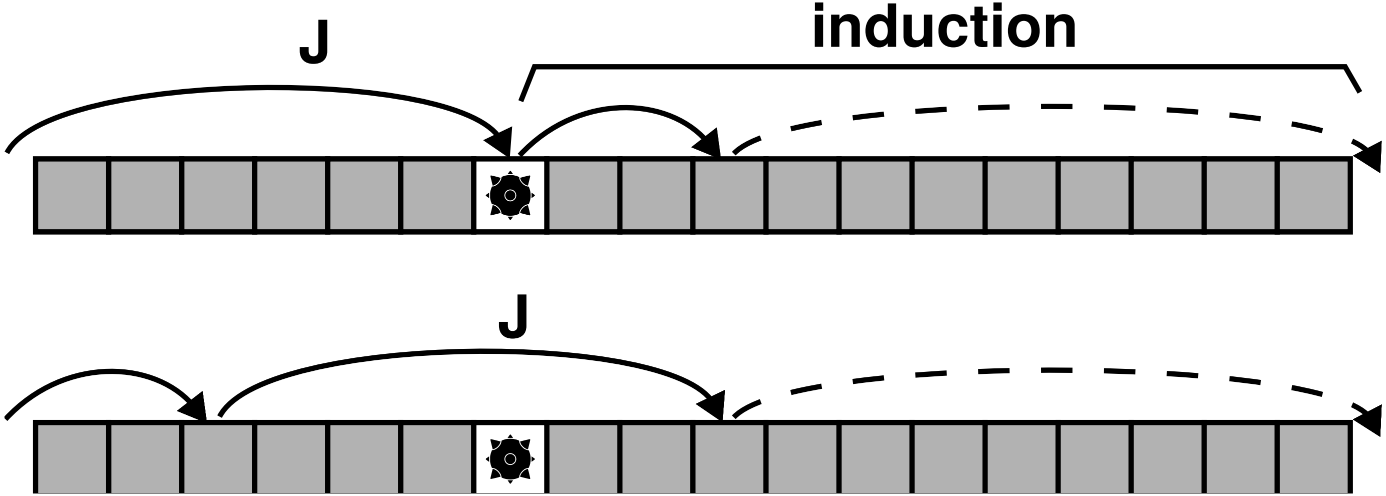
Solution – J lands on mine

- Try an analogous approach



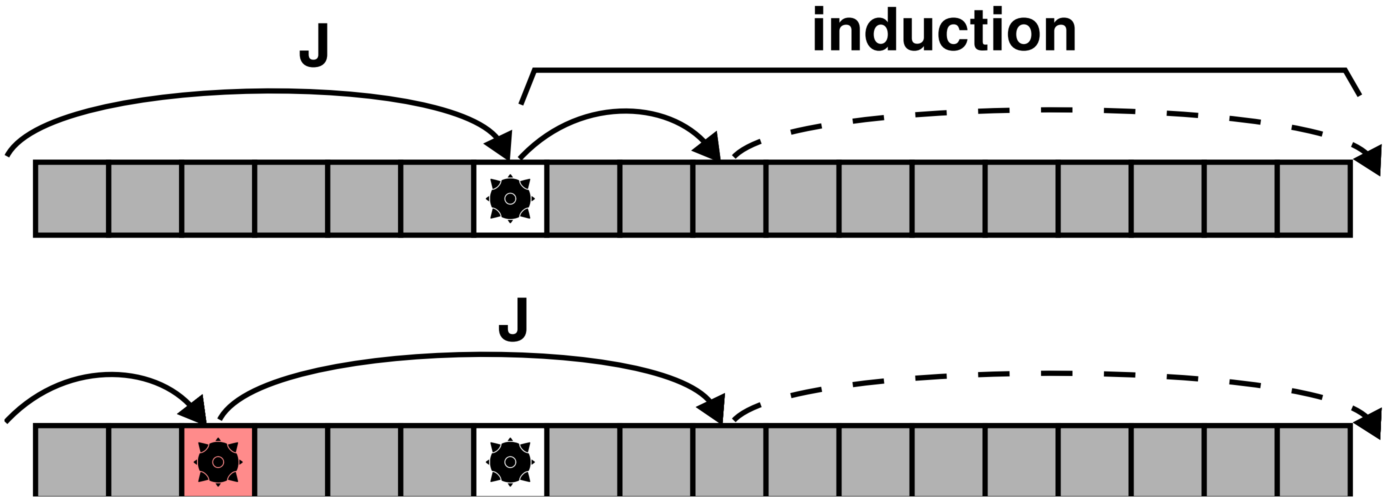
Solution – J lands on mine

- Try an analogous approach



Solution – J lands on mine

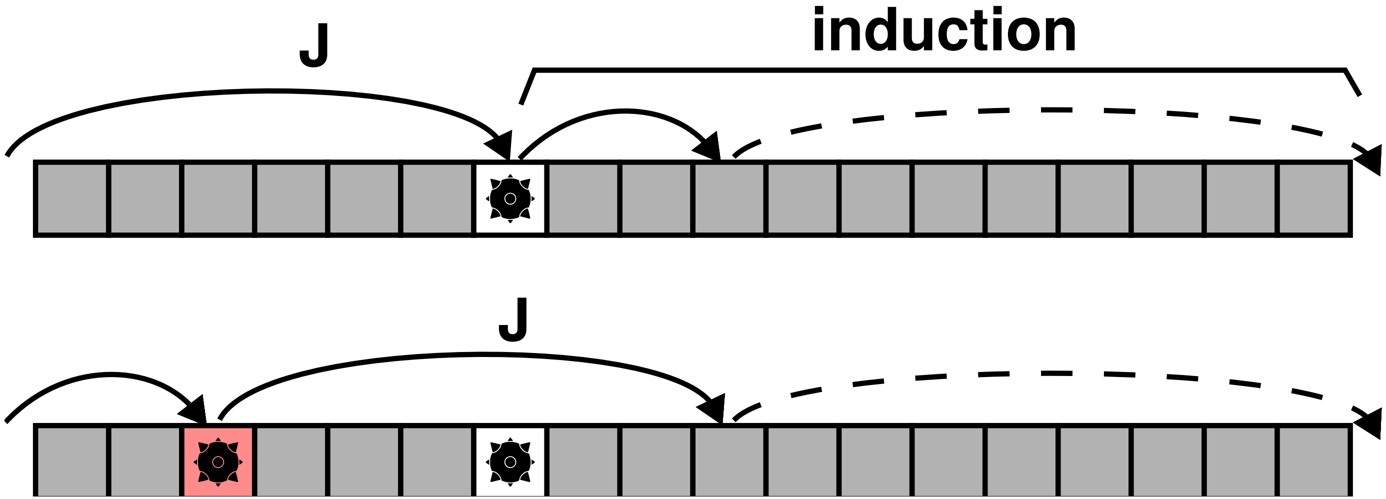
- Try an analogous approach



- What to do with mines before?

Solution – J lands on mine

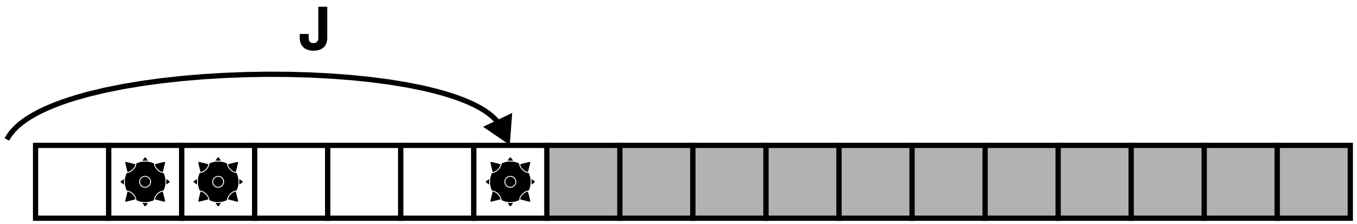
- Try an analogous approach



- What to do with mines before?
  - Use them to restrict the IH

Solution – J lands on mine

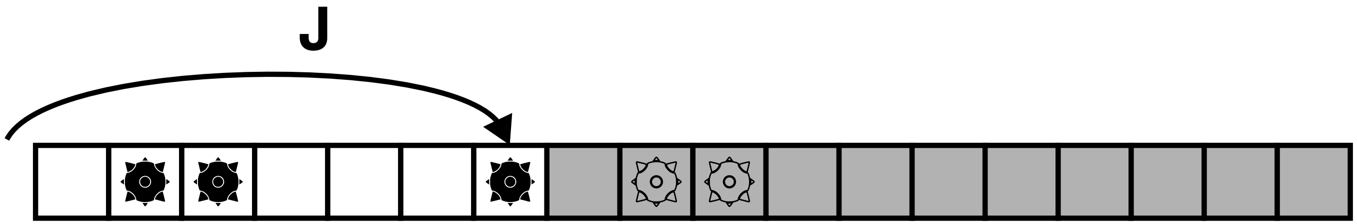
- Try an analogous approach



- What to do with mines before?
  - Use them to restrict the IH

Solution – J lands on mine

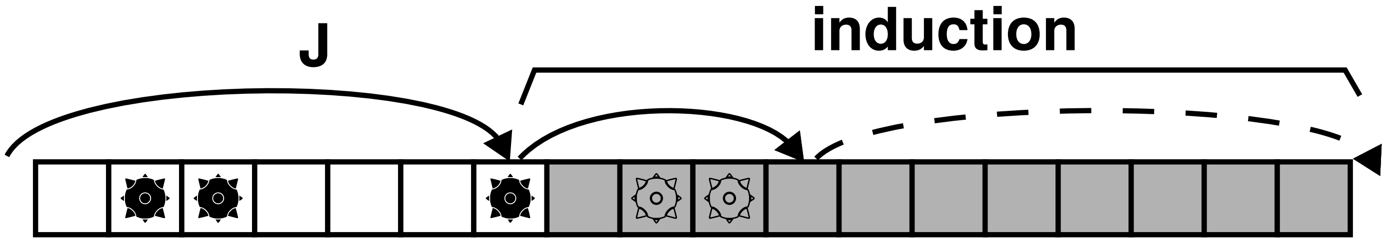
- Try an analogous approach



- What to do with mines before?
  - Use them to restrict the IH

Solution – J lands on mine

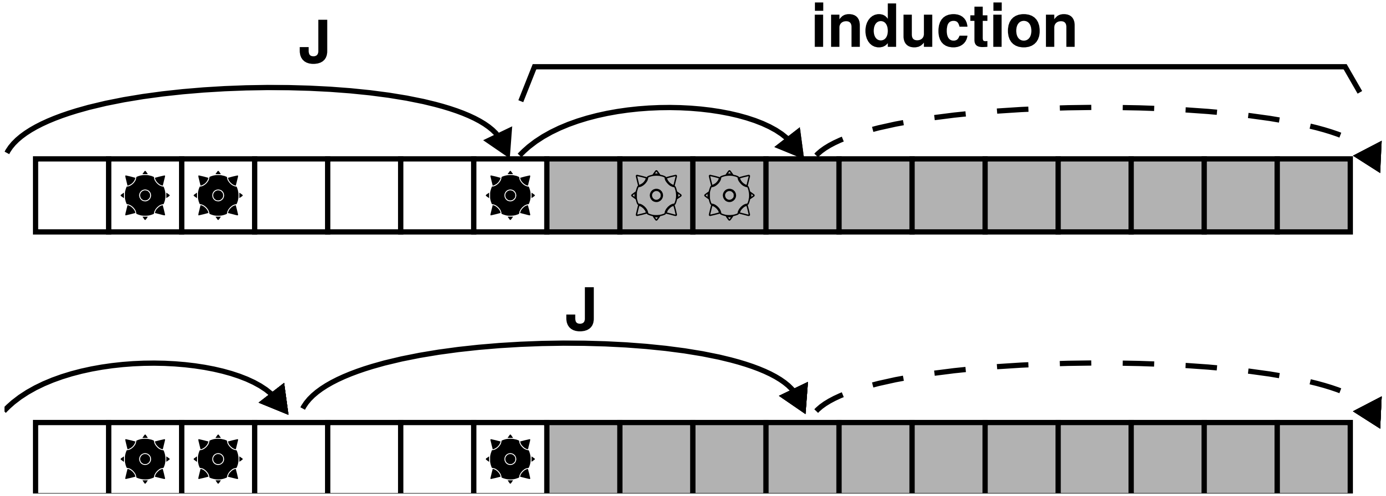
- Try an analogous approach



- What to do with mines before?
  - Use them to restrict the IH

Solution – J lands on mine

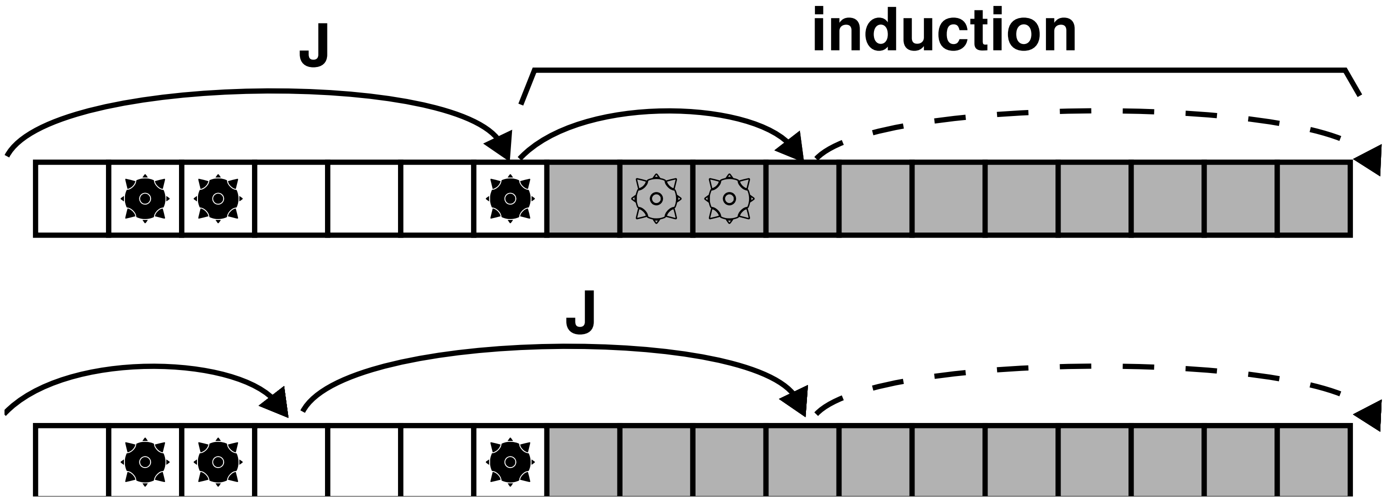
- Try an analogous approach



- What to do with mines before?
  - Use them to restrict the IH

Solution – J lands on mine

- Try an analogous approach



Problem Solved!