Teacher Guide: Are Bugs Intelligent?

Jake Braunberger

Revised December 2025

How to Use This Guide

This guide is written as if a colleague is walking you through the project step-by-step. This project is meant to be playful, surprising, and mathematically rich. It includes:

- what you actually *do* with students each day,
- the math behind the scenes (explained at a teacher level),
- anticipated misconceptions,
- CCSS-M alignment.

The student workbook is separate and written in a narrative, exploratory tone. This guide is your "behind the curtain" explanation of what is happening mathematically and pedagogically.

Learning Objectives

By the end of this project, students will:

- Understand random walks as a mathematical model.
- Represent a maze as a graph with nodes and edges.
- Construct probability tables describing movement rules.
- Compute multi-step probabilities using recursive updates.
- Compare human reasoning to random processes.
- Apply the scientific method to an open-ended question.

Standards Alignment (CCSS-M)

- **HSS.MD.A.1** — Define a random variable for a probability distribution.
- **HSS.MD.A.2** — Calculate probabilities for simple random processes.
- **HSS.MD.B.5** — Use probability to evaluate outcomes of decisions.
- **HSA.REI.D.10** — Understand relationships between representations.
- **MP1** — Make sense of problems and persevere.
- **MP4** — Model with mathematics.
- **MP5** — Use appropriate tools strategically.

Anticipated Misconceptions

- Students often think "random" means "50-50." Emphasize that probabilities depend on the number of available moves.
- Students may think probability tables and matrices are different ideas. They are the same information in different formats.
- Students may assume a maze is "too hard" if the bug takes a long time. Random walks are slow and inefficient by nature.
- Students may confuse the geometric drawing of a maze with the abstract graph. Reinforce that a graph is just nodes and edges.

Optional Extensions

- Introduce transition matrices for advanced students.
- Compare probability-table recursion to matrix multiplication.
- Explore absorbing Markov chains.
- Use Python or Sage to simulate thousands of random walks.

The Math Behind the Project (Teacher-Level)

This section explains the mathematics you will guide students through, at a teacher-friendly level.

1. Graph Representation of a Maze

A maze is converted into a graph:

- Each "decision point" (where a path splits) becomes a node.
- Each corridor between decision points becomes an edge.

The bug experiences the maze as "from here, I can go to these neighbors," not as a visual picture.

2. Probability Tables (Instead of Matrices)

For each node i, we list:

- all possible next nodes j,
- the probability $P(i \to j)$ of moving to each.

Example:

Current Node	Next Node	Probability
4	3	1/2
4	5	1/2

This is equivalent to a row of a transition matrix, but much more approachable for students.

3. Multi-Step Probabilities

For a path

$$i_0 \to i_1 \to i_2 \to \cdots \to i_n,$$

with step probabilities $P(i_k \to i_{k+1})$, the path probability is

$$P(\text{path}) = \prod_{k=0}^{n-1} P(i_k \to i_{k+1}).$$

To find the probability of being at node j after n steps, you:

1. list all paths of length n that end at j,

2. compute the probability of each path,
3. add those probabilities.

This is exactly what matrix powers would give in a Markov chain model, but here we stay in the probability-table representation.

4. Recursive Probability Updates

We track a probability distribution p_n over nodes after n steps. If $p_n(i)$ is the probability of being at node i after n steps, then:

$$p_{n+1}(j) = \sum_i p_n(i)\, P(i \to j).$$

This is the compact form of the path logic above.

5. SageTeX Example: Updating a Distribution

Below is a SageTeX example that computes distributions with the recursive rule.

```
# Example probability table for a tiny graph
transitions = {
    4: [(3, 1/2), (5, 1/2)],
    3: [(4, 1/2), (5, 1/2)],
    5: [(4, 1/3), (3, 1/3), (6, 1/3)],
    6: []
}

def step(dist):
    new = {node: 0 for node in transitions}
    for node, prob in dist.items():
        for nxt, p in transitions[node]:
            new[nxt] += prob * p
    return new

p0 = {4: 1, 3: 0, 5: 0, 6: 0}
p1 = step(p0)
p2 = step(p1)
```

$$\text{Step 1 distribution: } \left\{4 : 0, 3 : \frac{1}{2}, 5 : \frac{1}{2}, 6 : 0\right\}$$

$$\text{Step 2 distribution: } \left\{4 : \frac{5}{12}, 3 : \frac{1}{6}, 5 : \frac{1}{4}, 6 : \frac{1}{6}\right\}$$

6. Probability Trees

Probability trees give a visual way to understand multi-step probabilities before students move to the more compact distribution method.

A probability tree of depth n shows:

- every possible path the bug can take in n steps,
- the probability of each step along each path,
- how probabilities multiply along a path,
- how probabilities add when multiple paths end at the same node.

Each level of the tree corresponds to one step in the random walk. A branch represents a single move from one node to another.

If a path consists of moves

$$i_0 \to i_1 \to i_2 \to \cdots \to i_n,$$

then the probability of the entire path is

$$P(\text{path}) = \prod_{k=0}^{n-1} P(i_k \to i_{k+1}).$$

To compute the probability that the bug is at node j after n steps:

1. List all paths of length n that end at j.
2. Compute the probability of each path.
3. Add those probabilities.

Once this is understood visually, the recursive update rule

$$p_{n+1}(j) = \sum_i p_n(i)\, P(i \to j)$$

feels like a natural summary: every path ending at j must come from some previous node i.

7. Drawing a Probability Tree

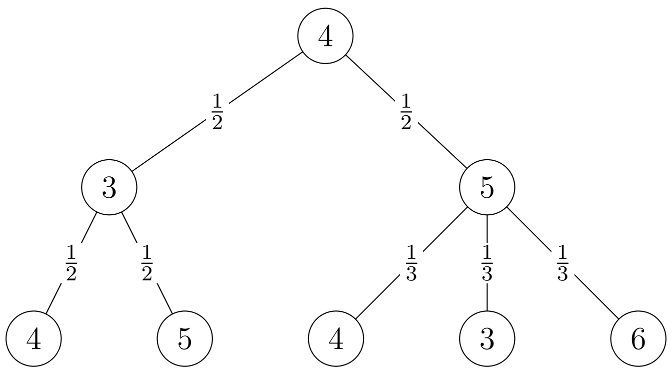

Figure 1: Automatically generated depth-2 probability tree starting at node 4

This diagram can be used for your own understanding or as a student-facing visual. Students will typically draw a simplified tree by hand. The branches are labeled with their respective, normalized probabilities, and the nodes are labeled with their node name. For example, the node named "3" in the left of the graph has two possible outcomes of a "move". If a bug were at node "3", it could move either to node "4" or node "5". Since their probabilities of occurence is assumed to be equal, their probability of occuring is 1/2.

How to Teach Probability Trees (Teacher Mini-Script)

This mini-script gives you a practical way to introduce probability trees.

1. Set the Stage (2–3 minutes). Teacher says: "Today we're going to look at how a bug might move through a maze if it makes random choices. Instead of guessing where it might end up after several steps, we're going to draw a picture that shows *every possible path* the bug could take."

Teacher move: Draw a single starting node on the board (e.g., "4").

2. Build the First Level (3–4 minutes). Teacher says: "From node 4, the bug has two choices: 3 or 5. Each choice has a probability. Let's draw those as branches."

Teacher move: Draw two branches labeled "3 ($\frac{1}{2}$)" and "5 ($\frac{1}{2}$)".

Key idea: A branch represents a possible move and its probability.

3. Build the Second Level (5 minutes). Teacher says: "Now imagine the bug is standing at each of these new nodes. From each one, it has its own set of choices. We draw those too."

Teacher move: Extend the tree one more level using the probability table.

Teacher tip: Narrate the structure: "From 4, it can go to 3 or 5. From 5, it can go to 4, 3, or 6."

4. Explain Path Probabilities (3 minutes). Teacher says: "Each path down the tree represents one possible sequence of moves. To find the probability of a path, we multiply the probabilities along the branches."

Teacher move: Pick one path and compute it aloud:

$$P(4 \to 3 \to 5) = \tfrac{1}{2} \times \tfrac{1}{2} = \tfrac{1}{4}.$$

Teacher tip: Emphasize why we multiply: "The bug has to make both moves in a row. Each move is an 'and', and with 'and' you multiply."

5. Explain How to Find the Probability of Ending at a Node (3–4 minutes). Teacher says: "To find the probability that the bug ends at a certain node after two steps, we look at all the paths that end there and add their probabilities."
Teacher move: Circle all paths ending at node 4. Add their probabilities.
Teacher tip: Emphasize why we add: "These are different ways to end up in the same place. That's an 'or', and with 'or' you add."

6. Connect Trees to Distributions (2 minutes). Teacher says: "Probability trees are great for small numbers of steps, but they get big fast. Instead of drawing the whole tree, we can summarize everything in a table that tracks the probability of being at each node after each step."
Teacher move: Show the distribution update rule:

$$p_{n+1}(j) = \sum_i p_n(i)\, P(i \to j).$$

Teacher tip: Say explicitly: "The distribution method is just the tree method written in a compact form."

7. Let Students Try a Small Tree (10 minutes). Teacher says: "Now you'll build a small probability tree of depth 2 for a simple graph. Don't worry about making it perfect—focus on the structure."

Teacher move: Circulate and check that:

- branches match the probability table,
- probabilities multiply correctly,
- students add paths ending at the same node.

8. Wrap-Up (1–2 minutes). Teacher says: "Probability trees help us see the structure of randomness. Once you understand the tree, the distribution method will feel natural."

Matrices (Teacher Notes)

This section is for teachers only. Students do not need matrix notation to complete the project, but understanding the matrix structure behind the random walk can help you see how the entire unit fits together mathematically. A random-moving bug is following a **transition matrix**: a table that tells you the probability of moving from one node to another in a single step. Updating the bug's probability distribution is equivalent to multiplying by this matrix.

Transition Matrices

For any maze or graph, we can build a **transition matrix** T. Each row corresponds to a current node, and each column corresponds to the next node.

- T_{ij} is the probability of moving from node i to node j.
- Each row sums to 1.
- Nodes with no outgoing edges have a row of zeros (or a 1 on the diagonal if you want them to be absorbing).

For example, if node 4 goes to nodes 3 and 5 with probability 1/2 each, then row 4 of the matrix is:

$$[0,\ 0,\ 1/2,\ 0,\ 1/2,\ 0,\ 0,\ 0]$$

(assuming nodes are ordered 1–8).

Distributions as Vectors

A probability distribution over nodes can be written as a **row vector**:

$$\mathbf{p} = [p_1,\ p_2,\ p_3,\ \ldots]$$

where p_i is the probability that the bug is at node i.
At the start, the distribution is "100% at the start node," for example:

$$\mathbf{p}_0 = [0,\ 0,\ 1,\ 0,\ 0,\ 0,\ 0,\ 0]$$

if the bug begins at node 3.

Updating the Distribution

The key mathematical idea is:

$$\mathbf{p}_{\text{new}} = \mathbf{p}_{\text{old}}\, T$$

This is exactly the rule students use verbally:

New probability at a node = sum over all previous nodes of (previous probability) × (probability of moving into the new node).

Matrix multiplication is simply the compact version of that rule.

Multi-Step Movement

After 2 steps:

$$\mathbf{p}_2 = \mathbf{p}_0 T^2$$

After 3 steps:

$$\mathbf{p}_3 = \mathbf{p}_0 T^3$$

And so on.
This is the matrix version of a **probability tree**:

- the tree shows all paths explicitly,
- the matrix raises the transition structure to a power,
- both produce the same distribution.

Why This Matters for Teachers

You do not need to teach matrices to students, but knowing the structure helps you:

- understand why the distribution method works,
- see how the random walk "settles" into patterns,
- diagnose student misconceptions,
- and connect the project to high-school and college-level probability.

It also gives you a way to check your own examples quickly using a calculator or software.

Optional Extensions (Teacher-Only)

- Show how T^n stabilizes as n grows (steady-state behavior).
- Compare different mazes by comparing their transition matrices.
- Let advanced students explore a small matrix example digitally.

These are entirely optional and should not appear in the student workbook.

Example

This example shows how the matrix method works on a concrete maze. We begin with the graph shown below. The original nodes are labeled `a` through `m`. For the matrix method, we relabel them as integers so that the transition matrix has a clean, numerical indexing.

Letter	Number	Letter	Number	Letter	Number
a	0	f	5	l	10
b	1	g	6	m	11
c	2	h	7		
d	3	i	8		
e	4	j	9		

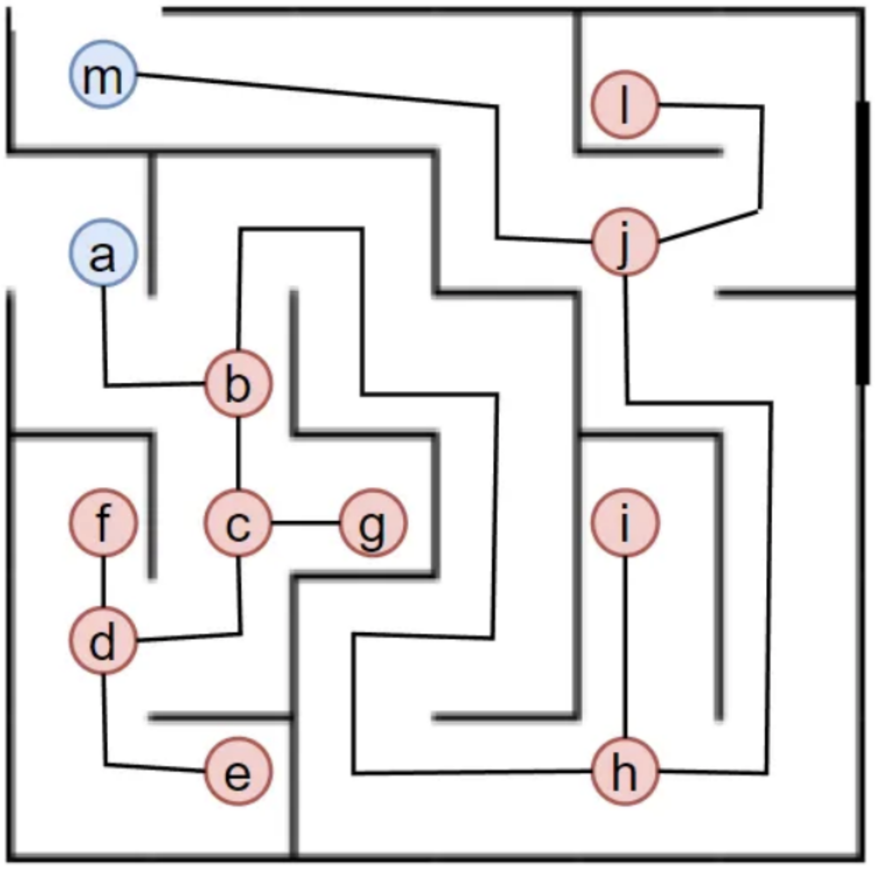

Figure 2: Maze graph with nodes labeled `a` through `m`. Start node is `a`, exit node is `m`. Integer labels used in the matrix correspond to these letters.

The transition matrix for this graph, using the integer relabeling above, is:

$$T = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

To help interpret this matrix, consider **row 2** (corresponding to node `c`, now node 2). In the graph, node `c` has two neighbors: `d` and `g`. Each is equally likely, so row 2 contains a $\frac{1}{2}$ in the columns for nodes 3 and 6, and zeros elsewhere:

$$\text{Row } 2 = [0,\ 0,\ 0,\ \tfrac{1}{2},\ 0,\ 0,\ \tfrac{1}{2},\ 0,\ 0,\ 0,\ 0,\ 0].$$

Every row of the matrix is constructed in this same way: each nonzero entry represents a possible move and its probability.

The initial probability distribution is the row vector

$$\mathbf{p}_0 = [1,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0],$$

meaning the bug begins at node 0 (formerly `a`) with probability 1.

To compute the probability distribution after seven moves, we multiply the initial distribution by the seventh power of the transition matrix:

$$\mathbf{p}_7 = \mathbf{p}_0 T^7 = (0.0,\ 0.34,\ 0.0,\ 0.23,\ 0.0,\ 0.0,\ 0.1,\ 0.0,\ 0.1,\ 0.15,\ 0.0,\ 0.09)$$

The entry corresponding to node 11 (formerly node `m`) gives the probability that the bug has reached the end of the maze after seven moves:

$$\mathbf{p}_7[11] = 0.09$$

Finally, we can visualize how the probability of reaching the exit grows over time. The plot below shows the probability of having completed the maze as a function of the number of steps taken. The slow rise reflects the maze's branching structure and the possibility of looping before reaching the exit.

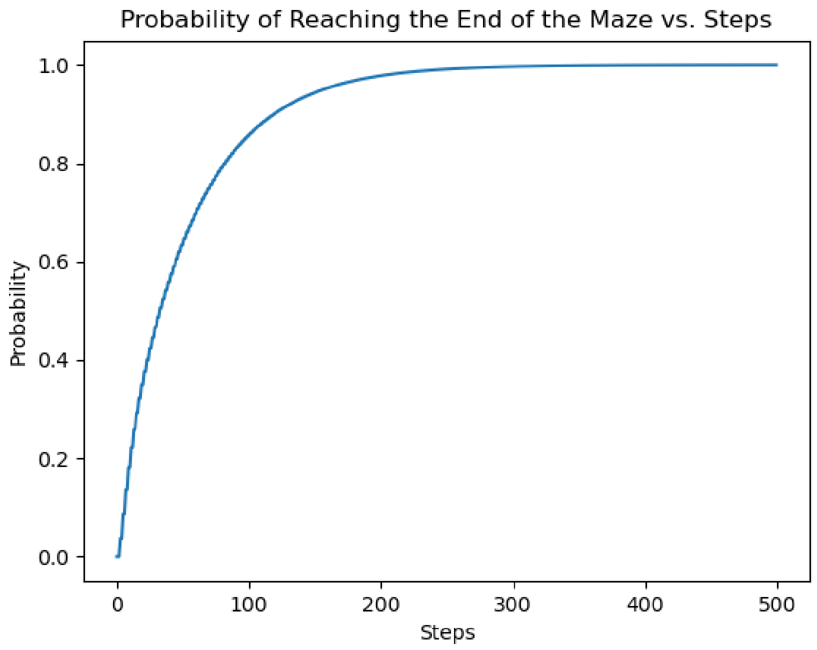

Figure 3: Probability of reaching the end of the maze as a function of the number of steps elapsed.

This example shows how the matrix method summarizes all possible paths compactly, and why it becomes the most efficient tool once the maze grows beyond a few steps.

Day-by-Day Teaching Script (Expanded)

The following daily plan shows how these ideas unfold in the classroom. This section gives you concrete teacher moves, sample questions, and approximate timing. Adjust as needed for your class length and context.

Day 0: Solve a Tricky Maze

Goal: Surface students' assumptions about intelligence and problem-solving.

Launch (5 minutes)

- **Teacher move:** Hand out a maze that is intentionally tricky. Ask students to work silently at first.
- **Script:** "Today you're going to solve a maze, but I'm just as interested in *how* you solve it as whether you finish."
- Have students write their start and end times on the workbook page.

Work Time (5–10 minutes)

- Circulate and **listen** for strategies: tracing walls, backtracking, guessing, scanning.
- **Ask quietly:** "What are you paying attention to right now?" or "How do you know which way to go?"

Discussion (10 minutes)

- **Whole-class share:** Invite 2–3 students to describe their strategies.
- Record on the board under headings like "Look ahead," "Systematic," "Guess and check."
- **Prompt:** "Do any of these strategies feel more 'intelligent' than others? Why?"

Reflection (5 minutes)

- Have students answer the "Before and After" questions in their workbook.
- **Closing prompt:** "We'll come back to these answers later when we compare you to a very dumb bug."

Day 1: Create a Maze

Goal: Shift students from maze-solvers to maze-designers and start them thinking structurally.

Launch (5 minutes)

- **Script:** "Yesterday you solved a maze. Today you get to be the designer. Designing a maze is a different kind of thinking."
- Discuss briefly: "What makes a maze hard?" Elicit ideas and record them.

Brainstorm (5 minutes)

- Have students list features that make mazes harder or easier in their workbooks.
- **Prompt:** "Think about dead ends, long paths, loops, choices, distractions."

Maze Design (20–25 minutes)

- Students sketch mazes on graph paper.
- **Teacher moves:**
  - Encourage at least a few decision points and one clear start/finish.
  - Ask: "Where is the first important choice in your maze?" "Could someone get stuck looping?"
- Optionally run a quick "most difficult maze" gallery walk at the end.

Reflection (5 minutes)

- Students respond in writing: "What was harder: solving or creating a maze? Why?"

Day 2–3: OpenSCAD Modeling

Goal: Introduce basic 3D modeling tools and connect spatial reasoning to maze structure.

Launch (5 minutes)

- **Script:** "You're going to learn how to build 3D shapes on the computer using code. Eventually, we'll use this to build a version of your maze."
- Briefly demo the OpenSCAD interface if students haven't seen it.

Part 1: Basic Shapes (15–20 minutes)

- Teacher demonstrates commands for cubes, spheres, cylinders.
- Students copy one example of each into their workbook and then into OpenSCAD.

- **Check for understanding:** Ask, “What happens if we change this number?” and let them experiment.

Part 2: Transformations (15–20 minutes)

- Demonstrate translate, rotate, scale, mirror with simple examples.
- **Prompt:** “Which transformation feels most useful for building a maze? Why?”
- Give students a small challenge, e.g., “Make a hallway with two rooms off to the sides.”

Part 3: Start the 3D Maze (Day 3) (20–30 minutes)

- Students begin turning their 2D maze into a simple 3D structure.
- Circulate to help with:
  - aligning corridors,
  - choosing a consistent scale,
  - troubleshooting small syntax errors.

Day 4: Neuroscience and Memory

Goal: Connect maze navigation to memory and physical limits of brains.

Launch (5 minutes)

- **Script:** “If a bug wanted to memorize a maze, what would it need to store in its brain?”
- Have students brainstorm ways to represent a map (picture, list of connections, directions, etc.).

Mini-Lesson (10–15 minutes)

- Give a simple explanation of neurons, bits/bytes, and the idea that brains have finite capacity.
- Optionally show a short clip (e.g., 3Blue1Brown) about simple neural nets and pattern storage.
- **Prompt:** “Do you think an ant can store a detailed picture of a maze? Why or why not?”

Discussion + Writing (15–20 minutes)

- Discuss the tradeoff between detailed maps vs. simple rules (e.g., “always follow the left wall”).
- Students respond to workbook questions about tiny brains and map storage.

Day 5: Experimental Design

Goal: Prepare students to think about fair experiments and how to measure "bug performance."

Launch (5 minutes)

- **Script:** "If we want to test whether bugs are intelligent, we need an experiment that's actually fair."

Mini-Lesson (10–15 minutes)

- Read or summarize a short excerpt from Feynman on good vs. bad experiments.
- Highlight: control, clear measurement, repeatability.
- Ask: "What would be an unfair test of bug intelligence?" and collect 2–3 examples.

Design Discussion (15–20 minutes)

- Pose: "If we put food at the end of a maze, what might we be testing instead of intelligence?"
- In small groups, have students propose possible measures of bug performance (time, steps, success/failure, etc.).
- Share out and identify 1–2 measures to use later in the project.

Day 6: A Very Dumb Bug (Random Walks)

Goal: Give students a physical sense of randomness and path variability.

Launch (5 minutes)

- **Script:** "Today we'll watch what happens if a bug follows a very simple, very dumb rule: flip a coin to decide where to go."

Part 1: Coin-Flip Walk (20 minutes)

- Set up a simple number line or board walk; define "left" and "right" moves.
- Have students or pairs perform several walks of a fixed number of steps, recording the final position.
- **Prompts while circulating:**
  - "Were you ever surprised by where you ended up?"
  - "Did you ever return to where you started?"

Quick Debrief (10 minutes)

- Collect a few final positions on the board and look for patterns (clustering near the middle, occasional far-out results).
- Connect this to the idea of ensembles: many random walkers vs. one.

Part 2: Fractional Bugs (10 minutes)

- Introduce the idea of tracking "many identical bugs at once" using fractions or probabilities.
- **Script:** "Instead of following one bug, imagine we have 100 identical bugs. We can talk about what fraction end up where."

Day 7: Probability Tables

Goal: Help students encode movement rules in a structured, numeric way.

Launch (5 minutes)

- **Script:** "We're going to describe how our very dumb bug moves using tables. These tables are the language our bug follows."

Part 1: Simple Example (10–15 minutes)

- Present the tiny example with nodes 3, 4, 5, 6 (as in the workbook).
- Ask: "From 4, what are all the places the bug can go? With what probabilities?"
- Build the table together on the board.

Part 2: Students Build a Table (20 minutes)

- Show a simple graph or part of a maze.
- Have students fill out a probability table for each node:
  - list neighbors,
  - assign equal probabilities if movement is random.
- **Check for understanding:** Walk around and ask, "Do your probabilities in this row add to 1?"

Wrap-Up (5 minutes)

- Emphasize that these tables are the core rule set for the bug's movement.

Day 8: Multi-Step Probabilities and Trees

Goal: Move from single-step rules to multi-step paths using trees, then connect to distributions.

Launch (5 minutes)

- **Script:** "Our bug doesn't just move once; it takes many steps. A probability tree shows every path it could take."

Part 1: Bug World Rules (5 minutes)

- Review the tiny bug-world table from the workbook (nodes 2, 3, 4, 5, 6).
- Check that students understand the meaning of each row.

Part 2: What Is a Probability Tree? (10 minutes)

- Sketch a small fragment on the board: start at 4, then two branches to 3 and 5.
- **Prompt:** "What would the next level look like if we keep going from 3? From 5?"

Part 3: Students Build a 2-Step Tree (15–20 minutes)

- Students draw their own depth-2 tree starting at node 4.
- Circulate and check:
  - Are all allowed moves present?
  - Are probabilities correctly copied from the table?

Part 4: Paths and Path Probabilities (10–15 minutes)

- Have students list all 2-step paths and compute the probability of each.
- Model one example on the board.
- **Prompt:** "Why do we multiply along a path?" (emphasize "and")

Part 5: Where Does the Bug End Up? (10 minutes)

- Students add probabilities of paths that end at the same node.
- **Prompt:** "Why do we add here?" (emphasize "or")

Part 6: From Trees to Distributions (10–15 minutes)

- Show the worked example with A, B, C.
- Walk through Step 0, Step 1, Step 2 distributions.
- Then have students write the 1-step and 2-step distributions for *their* tree.

Day 9: Graphs of Student Mazes

Goal: Help students abstract their own mazes into graphs and assign probabilities.

Launch (5 minutes)

- **Script:** "We're going to turn your maze into a graph: circles for decision points, lines for corridors."

Part 1: Identify Decision Points (10–15 minutes)

- Students circle choice points on their maze drawings.
- They label each decision point with a number.
- Check that each label corresponds to a meaningful choice, not just a corner.

Part 2: Sketch the Graph (15–20 minutes)

- Students draw a node-and-edge version of their maze: just circles and lines.
- **Prompt:** "If you ignore the walls, what stays the same about how you can move?"

Part 3: Probability Tables (10–15 minutes)

- For at least one node, students build a probability table:
  - list neighbors,
  - assign equal probabilities if the bug chooses randomly.
- Encourage students to complete tables for all nodes if time allows.

Day 10: Probability of Completion

Goal: Use distributions (ensembles of bugs) to estimate the chance of finishing the maze after n steps.

Launch (5 minutes)

- **Script:** "Imagine 100 bugs starting in your maze. How many do you expect to reach the exit after a few steps?"

Part 1: Starting Distribution (10 minutes)

- Students choose a start node and an exit node in their graph.
- They write the starting distribution: 100% at the start, 0% elsewhere.
- **Check:** "Does your distribution add up to 100%?"

Part 2: Updating the Distribution (20–25 minutes)

- Model one update on the board using a small example:
  - Show how probability "flows" out of each node according to the table.
- Students update their distribution for 1 step, then a few more.
- They record the probability of being at the exit node after each step.
- Optionally, use a spreadsheet or Python/Sage to support the calculations.

Part 3: Interpreting the Results (10 minutes)

- **Prompt:** "How does the probability of being at the exit change as steps increase?"
- If available, show a probability vs. steps graph (as in the example).

Day 11: Running the "Very Dumb Bug" Experiment

Goal: Tie together theoretical probability and physical random experiments.

Launch (5 minutes)

- **Script:** "Now we'll let a physical bug (a bead or token) follow the random rules you've been modeling and see what happens."

Part 1: Set Up (5–10 minutes)

- Choose the maze model (3D, 2D, or graph-only).
- Review the "very dumb bug" rule: always choose randomly at decision points.

Part 2: Demonstrate One Trial (5–10 minutes)

- Model a single run in front of the class.
- Narrate: "We're at node 4. There are 3 choices. We'll roll a die..." and so on.

Part 3: Student Trials (20–25 minutes)

- In pairs or groups, students run at least 5 trials, recording:
  - path taken,
  - whether they reached the exit,
  - number of steps.
- Circulate and watch for:
  - loops,

  - long unsuccessful runs,
  - surprising quick successes.

Part 4: Discussion and Connection (10–15 minutes)

- **Prompts:**
  - "How often did your bug reach the exit?"
  - "Did some paths show up more often than others?"
  - "How do your experimental results compare to your theoretical probabilities?"
- Emphasize that randomness in real experiments approximates, but does not exactly match, the theoretical distributions.

Reflection (5–10 minutes)

- Have students complete the workbook reflection about randomness, models, and intelligence.
- **Closing script:** "You've just compared human reasoning to a very dumb bug, and used math to understand the difference."

Encourage students to connect their physical results to the mathematical structures they have built throughout the project.