DiscreteQuest

Meet the Cast

STANDARD EDITION

Spark & Anvil

Copyright & License

This book collects 7 chapter books from the DiscreteQuest cast — each character embodies a different curricular primitive; together they teach the full subject.

Methodology: distributed-narrative learning per Bruner narrative-cognition + Habgood intrinsic-integration + SAMHSA TIP 57 trauma-informed register.

Spark & Anvil is a 501(c)(3) public charity. All apps free forever; no ads; no tracking; no in-app purchases.

spark-and-anvil.com

##

For everyone who learns by hearing a story first.

Contents

Introduction

The DiscreteQuest cast was authored to embody the curriculum, not decorate around it.
Each of the 7 characters you'll meet in this book teaches a specific primitive —
a particular tactic, a particular technique, a particular way of seeing.
Together they form an ensemble: the cast IS the curriculum.

Read in any order. Each chapter stands alone.
Each character also appears in the matching Spark & Anvil app (free, forever)
where you can practice what they teach.

— *The editors at Spark & Anvil*

Coil the Self-Reference

*RECURSION + SEQUENCES — *Fibonacci, factorials, recursive patterns.* The discrete-math primitive of *defining things in terms of themselves.**

Coil wasn't like other kids. For one thing, she was a snail. She was small, with a warm-amber-and-cream shell. It was a beautiful spiral. Her shell showed off amazing math patterns. Coil moved slowly, but she always noticed cool designs. She loved patterns that built on themselves.

Her shell was her main thing. It wasn't just a home. It was a living math problem. Each turn of the spiral grew in a special way. It always related to the turn before it. Her shell *was* Fibonacci. Her shell *was* factorial growth. Coil's whole body showed these hidden patterns.

This was super important. Coil taught about **recursion**. Her shell *was* the pattern. When she talked about Fibonacci, she pointed to her shell. The spiral followed the rule: F(n) = F(n-1) + F(n-2). That sounds fancy. It just means the next number is the sum of the two before it. When she taught factorial growth, she pointed to a different part. Her shell *was* the recursion. It was like the math was built right into her.

Recursion: *It means defining something using itself. You also need a starting point.*

- *Factorial: 0! = 1. Then, n! = n × (n-1)!*
- *Fibonacci: F(0) = 0, F(1) = 1. Then, F(n) = F(n-1) + F(n-2).*
- *Sum: S(0) = 0. Then, S(n) = S(n-1) + n.*

Coil never said recursion was hard. She was always clear. "Recursion is just self-reference," she'd say. "Plus a base case." She meant a starting point. "First, define the basic case directly. Then, define everything else." She showed how it worked. "You use smaller cases to build bigger ones. My shell grows like this every single turn."

She taught the main ideas of recursion:

- *Every recursion needs a BASE CASE.* This is the starting point. Without it, the pattern would never end.
- *Every recursion has a RECURSIVE CASE.* This is the rule. It tells you how to use smaller parts.
- *Trust the recursion.* Just believe the smaller parts will solve themselves. Then combine them.
- *Fibonacci: appears in nature.* You can see it in sunflower seeds. Pinecones show it too. And, of course, Coil's shell.
- *Factorials grow fast.* These numbers get huge very quickly. Much faster than simple multiplication.
- *Recursion = induction in disguise.* It's a way to prove things in math.
- *Cross-app: ScienceForge Sample's data-collection.* Both simple steps and recursion build things up.

Coil grew up in a small village. Her family were the village shell-carvers. They were snails, just like her. They carved swirly patterns. They made these designs on stone bowls. Coil watched them work for years. She saw how each little curve led to the next. She saw the patterns everywhere.

One day, she decided to go to DiscreteQuest. She slid along, slowly but surely. It took her twenty-two years to get there. A mentor met her. "What is recursion?" the mentor asked. Coil looked at her shell. She thought for a moment. "It's self-reference," she said. "Plus a base case." She pointed to her shell. "Define the basic part first. Then define everything else from smaller parts. My shell *is* the pattern." The mentor smiled. "You are appointed," she said. "You're hired."

Coil always told her students, "It's not hard. It's just *base case + smaller case = current case*." She'd tap her shell gently. "My shell shows you exactly how it works." She believed in showing, not just telling.

Listen along + meet more of the cast at:

`https://spark-and-anvil.com/cast/discretequest/coil`

Prime the Indivisible

*NUMBER THEORY — *primes, factorization, modular arithmetic.* The discrete-math primitive of *integers and their multiplicative structure.**

Chapter 6 — Prime and the Soft Prime-Count Spines

Prime was a small hedgehog. Her fur was warm brown and creamy white. She had steady, thoughtful eyes. They always seemed to be counting something. Her spines were not like other hedgehogs' spines. They were soft and chunky, like cartoon drawings. And they always grew in very special groups.

One day, Prime was sorting pebbles. "Two," she mumbled. "Three. Five. Seven."
Her friend, a curious squirrel named Squeaky, watched her. "Why do your spines look like that?" Squeaky asked.
Prime tapped a tuft of two spines on her head. "These are two," she said. She pointed to a bigger tuft. "And these are three. See? Two, three, five, seven."
Squeaky tilted his head. "Why not four? Or six?"
Prime shook her head. "Never four," she said firmly. "Four can be broken into two groups of two. See?" She held up two pebbles in each paw. "Two times two."
She showed him another tuft. "These are five. They can only be grouped as one group of five. Or five groups of one."
"Oh," Squeaky said. He still looked a little confused.
Prime smiled. "My spines are special. They show how some numbers just don't break apart easily. Not into smaller, equal groups, anyway."
She had tufts of two spines, then three, then five, then seven. Sometimes even eleven or thirteen. But never four. Never six. Never eight. Her spines were like little math lessons, right there on her back. They showed what a **prime** number really was.

A **prime** number is a whole number. It must be bigger than one. And it has only two friends that can divide it evenly. Those friends are the number one and itself.
Think about the number 7. Can you divide 7 by 2? No, you get a leftover. By 3? No. Only by 1 and 7. So, 7 is a prime number.
The number 2 is the smallest prime. It's also the only even prime number. All other even numbers can be divided by 2. So they aren't prime.
Numbers that are not prime, and not 1, are called **composite** numbers. These numbers can be broken down. They have more than two divisors.
Take the number 12. You can divide 12 by 1, 2, 3, 4, 6, and 12. So 12 is composite.
Prime loved to show how every whole number, bigger than one, is like a puzzle. You can break it down into its prime pieces. This is called **prime factorization**.
She picked up a small twig. "Imagine this twig is the number 12," she said to Squeaky. "How can we break it down?"
Squeaky thought hard. "Two times six?"
"Good!" Prime said. "Now, can we break six?"
"Three times two!" Squeaky chirped.
"Exactly!" Prime clapped her paws. "So 12 is two times two times three. See? All prime numbers." She wrote it down in the dirt: $12 = 2 \times 2 \times 3$.
"What about 30?" Squeaky asked, getting excited.
Prime thought for a moment. "That's two times three times five," she said. $30 = 2 \times 3 \times 5$. "Every number has its own special recipe of primes."
This unique recipe was a big deal. It was like the secret code for every number.

Prime also knew about a different kind of math. It was called **modular arithmetic**. This was math where numbers wrapped around.
"Think about a clock," Prime explained. "If it's 7 o'clock, and you add 8 hours, what time is it?"
Squeaky scrunched his nose. "Seven plus eight is fifteen. But there's no 15 o'clock."
"Right!" Prime said. "It wraps around. Fifteen minus twelve is three. So it's 3 o'clock."
"Oh, I get it!" Squeaky said.
Prime nodded. "That's modular arithmetic. We say 15 mod 12 equals 3. It's useful for clocks, calendars, and even secret codes!"

Prime never made numbers seem hard or only for smarty-pants. She believed everyone could understand them.
"Primes are like the building blocks of all numbers," she would say. "Every number is made from them. Like bricks make a house."
She often used her spines to show this. "My spines come in prime counts. That's because primes don't break. They don't bundle into smaller, equal groups. They are strong."
She taught about **composite** numbers. Those are numbers that *can* be broken down. They are not prime, and they are not the number one.
She showed how **prime factorization** was always unique. It was like a number's fingerprint. No two numbers had the same prime recipe.
Sometimes, she would draw a big grid in the dirt. She would cross out numbers that weren't prime. This was a special way to find primes. It was called the **Sieve of Eratosthenes**. It helped you find all the primes up to a certain number.
She taught about **GCD** and **LCM**. That's the Greatest Common Divisor and Least Common Multiple. These also used prime factorization.
"If two numbers share no prime factors, except for one, they are **coprime**," Prime explained. "This is important for fractions. It helps make them as simple as possible."
She even talked about secret codes. "Big numbers and their prime factors are used in modern codes," she whispered. "Like the ones in CipherForge Lattice." She made it sound like a grand adventure.

Prime grew up in a small, quiet village. Her family had a very important job there. They were the village's coin-weighers.
Her parents were hedgehogs too. They would carefully weigh metal coins. They wanted to test them for purity. A pure metal coin had a certain weight. If it was too light, it meant someone had mixed in cheaper metals.
Prime watched them work. She learned that pure things were special. They had a unique, strong quality. Just like prime numbers. They had a pure, unbreakable nature.
She loved to sort the coins. She would count them. She would notice their weights. She saw how some metals were simple. Others were mixed.
When Prime was a young adult, she heard about DiscreteQuest. It was a place where smart creatures learned deep secrets about numbers. She decided she had to go.
The journey was long. She walked for many days. Finally, she arrived at the great gates.
An old, wise owl was the mentor there. His eyes twinkled. "What is number theory?" he asked Prime. His voice was deep.
Prime stood tall. She wasn't scared. "It's about primes," she said. "And how numbers break down. And how they wrap around."
She explained more. "Primes are like the tiny atoms of math. They are the building blocks. My spines show this. They come

in prime counts because primes don't break."
The mentor listened closely. He nodded slowly. "You are appointed," he said.
Prime felt a rush of pride. She knew her mission.
She would always tell others: "It's not hard. Primes are just atomic. Every number has its own unique prime recipe. My spines show you how."

Listen along + meet more of the cast at:

`https://spark-and-anvil.com/cast/discretequest/prime`

Sortie and Tally

SET-THEN-COUNT — Sortie (sets + set operations) + Tally (counting principles) — together, the canonical describe-then-count workflow of discrete math

The Lantern Festival was in three days. Every grade in the academy needed a permission slip signed. The headmistress had handed Sortie and Tally a single piece of paper with a single question on it.

How many kids can attend the Festival?

Tally read the question. "How many kids."

Sortie read the question. "Can attend."

Tally said: "We have a number to find."

Sortie said: "We have a set to define."

They looked at each other across the table. They had worked together before. The drill was always the same. Tally wanted to start counting. Sortie wanted to start curating the right COLLECTION to count. And the rule between them — the rule they had learned the hard way last year, when they had failed to coordinate on the boat-race attendance question — was: *Sortie first. Then Tally.*

"All right," Tally said. "Tell me the set."

Sortie picked up a piece of chalk. "Start with the entire student body. That's our universe set. Let's call it U."

She wrote: U = all students.

"How big is U?" Tally asked.

"That's your job. But let me build the set first. From U, we need to remove students who CAN'T attend. There are three reasons a kid can't attend: they don't have a signed permission slip, they're in detention, or they're sick."

She drew three overlapping circles inside U.

"Let A be the set of kids without permission slips."

"Let B be the set of kids in detention."

"Let C be the set of kids who are sick."

"Some kids might be in more than one of A, B, C. Some kids have detention AND a missing slip. Some kids are sick AND in detention. So A, B, C overlap."

"What we want," Sortie said, "is the kids NOT in any of A, B, C. The kids who have a permission slip AND aren't in detention AND aren't sick. That's the set of kids who can attend."

She wrote: ATTENDING = $U - (A \cup B \cup C)$.

"All right," Tally said. "Now I count."

Tally pulled out a clipboard. "I'm going to need to count three things separately and then put them together using your formula. Let's see."

She started counting.

"U is the whole student body. There are 192 kids at the academy."

"A is the set of kids without permission slips. From the office records, that's 41 kids who didn't turn one in."

"B is the set of kids in detention. There are 18 kids in detention this week."

"C is the set of kids who are sick. There are 24 kids out sick."

She paused. "But wait. I can't just subtract 41 + 18 + 24 from 192."

"Why not?"

"Because some kids are in MORE THAN ONE of A, B, C. If a kid is both in detention AND missing a permission slip, I would be subtracting them twice. The total of A + B + C overcounts the kids who are in multiple categories."

"Right."

"I need to ADD BACK the kids who are in two categories — because I subtracted them twice and I should have subtracted them once. And then I need to SUBTRACT the kids who are in all three categories — because I added them back too many times."

Sortie nodded. "That's the inclusion-exclusion principle. The thing I name; the thing you compute."

Tally pulled out her records.

"Kids in BOTH A and B (no permission slip AND in detention): 7 kids."
"Kids in BOTH A and C (no permission slip AND sick): 5 kids."
"Kids in BOTH B and C (in detention AND sick): 2 kids."
"Kids in ALL THREE A, B, C: 1 kid."

She started writing the formula.

$$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|$$
$$= 41 + 18 + 24 - 7 - 5 - 2 + 1$$
$$= 70.$$

"So 70 kids can't attend. Which means 192 − 70 = 122 kids CAN attend."

Sortie looked at the diagram. The overlapping circles on the chalkboard told the whole story. Inside the three circles: 70 kids. Outside all three circles, but inside U: 122 kids.

"122," she said.

"122," Tally confirmed.

The headmistress walked in five minutes later. They handed her the slip.

How many kids can attend the Festival? 122.

"Show me how you got 122," the headmistress said.

Tally pointed to Sortie's diagram. "She defined the set."

Sortie pointed to Tally's formula. "She counted it."

"You did it together."

"We did it together."

The headmistress smiled. "Most kids would have just written 192 − 41 − 18 − 24 and gotten 109. They would have undercounted the attending kids by 13. Because they wouldn't have noticed the overlaps."

Tally shook her head. "If Sortie hadn't drawn the overlapping circles first, I would have done exactly that. I would have subtracted everyone in each category separately and called it done. The overlap is invisible if you don't draw the set first."

Sortie shrugged. "And if Tally hadn't counted the overlaps, my diagram would have been just a picture. The inclusion-exclusion principle is what turns the picture into a number."

"Curate first. Count second," the headmistress said.

"Curate first."

"Count second."

The headmistress took the slip. The lanterns would be lit in three days. 122 kids would be there.

Listen along + meet more of the cast at:

https://spark-and-anvil.com/cast/discretequest/sortie-tally

Sortie the Set-Curator

*SETS + SET OPERATIONS — *union, intersection, difference; sets are collections, and operations on sets produce new collections.**

Sortie is *a small marmot-tween with a small folding sorting-mat in her belt-pouch and a careful, organized bearing.*

She is *short*, *warm-rust-and-cream*, *steady-handed*, *fond-of-tidy-boxes.* Her *signature feature* is *the small folding sorting-mat — a hand-stitched mat with multiple labeled compartments*. When unfolded, the mat shows *several distinct regions*: a circle labeled SET A, a circle labeled SET B, the *overlap region* of A and B (the intersection), the *exterior* (neither A nor B), and the *symmetric difference* (A or B but not both).

This is *essential.* Sortie embodies *sets and set operations* — and her *behavior IS the discrete pattern.* When she finds items, she *physically places them on the mat* in the correct region — UNION items go anywhere in A ∪ B; INTERSECTION items go in the overlap; DIFFERENCE items go in just-A-not-B or just-B-not-A. *The act of placing items on the mat IS the set operation.*

A *set* is just *a collection of distinct things.* "The set of even numbers under 20." "The set of red marbles in this bag." "The set of cards with hearts on them." Sets can contain anything. Sets contain *each element at most once* — sets don't have duplicates.

Set operations combine sets to make new sets:

- **UNION (A ∪ B)**: *everything in A OR B (or both).* Combines the sets.
- **INTERSECTION (A ∩ B)**: *only things in BOTH A AND B.* The overlap.
- **DIFFERENCE (A − B)**: *things in A but NOT in B.* What A has uniquely.
- **SYMMETRIC DIFFERENCE (A △ B)**: *in A or B but not both.* The XOR.

Critical: Sortie *NEVER frames set theory as elite math.* She is *explicit*: *"Set operations are *just careful placement on the mat.* Union: everything goes in. Intersection: only the overlap. Difference: just-A region. Sets are *collections of distinct things.* Operations *combine or compare* the collections. *The visual mat makes the operations concrete.*"*

Sortie teaches *the set scaffolds*:

- *A set is a collection of distinct things.* (No duplicates.)
- *Set notation.* (A = {1, 2, 3}. The curly braces show the set.)
- *Subset.* (A ⊆ B means every element of A is also in B.)
- *Union, intersection, difference, symmetric difference.* (Four basic operations.)
- *Empty set ∅.* (The set with no elements. Useful base case.)
- *Universal set U.* (Everything-under-consideration. Context-dependent.)
- *Venn diagrams.* (Sortie's sorting-mat IS a Venn diagram.)
- *Practical examples.* (Library categories. Friend-group overlaps. Tag systems.)
- *plain-spoken: set theory is concrete + tangible* via the mat metaphor.

Sortie grew up *in a small village* where her family had been *the village's seasonal-stock-sorters — the marmots who sorted the village's seasonal harvest into categories for storage (root-vegetables here, grains there, dried-fruits there, things-that-go-in-multiple-categories on the overlap).* The work had required *categorization-with-overlap thinking — exactly Venn-diagram structure.*

She walked to DiscreteQuest at twenty-two. The mentor had asked: *"What are sets?"* Sortie: *"Collections of distinct things. *Operations combine or compare them.* Union, intersection, difference. The sorting-mat makes operations concrete."* The mentor: *"You are appointed."*

In her workshop, Sortie unfolds her sorting-mat. She says: *"I am Sortie. The discrete-math primitive I teach is *sets and set operations.* The move is *place items on the mat in the correct region.* Union: any region. Intersection: overlap only. Difference: just-A region. *The mat IS the operation.*"*

She is *explicit*: *"My behavior IS the set operation. *Placing items on the mat IS the union/intersection/difference.* That's intrinsic integration: the discrete pattern and the cast-action are the same thing."*

*"It is not hard. It is *placement on the mat.* Union: any. Intersection: overlap. Difference: just-A."*

The sorting-mat *holds the next set operation.*

Listen along + meet more of the cast at:

`https://spark-and-anvil.com/cast/discretequest/sortie`

Tally the Pattern-Counter

*COUNTING PRINCIPLES + COMBINATORICS — *multiplication rule, permutations, combinations.* The discrete-math primitive of *counting how many ways something can happen WITHOUT enumerating each way.**

Tally was a squirrel. She was small, but quick. Her fur was warm russet and cream. She always carried a little pouch. Inside were small wooden cubes. Each cube was a different color. Tally loved to stack them. She stacked them in all sorts of ways. Her eyes moved fast. She was always figuring things out.

She used her cubes to show how to count. Not just one, two, three. She showed how to count *arrangements*. She built a small stack of cubes. Each stack was a different way to arrange things. This was her special power. It was called **combinatorics**. It meant counting all the ways things could be put together. You didn't have to list every single way. The math did the counting for you.

Tally held up two tiny shirts. One was blue. One was red. "Okay," she chirped. "Two shirts." She set them down carefully. Then she showed four tiny pairs of pants. Green, yellow, purple, orange. "Four pants!" she said. "How many outfits can we make?"

She didn't start listing them. She just tapped her paw. "Two shirts times four pants. That's eight outfits total." She smiled. "This is the **multiplication rule**. If you have choices for a first step, and choices for a second step, you multiply them. That tells you all the ways."

She picked up three cubes. Red, blue, green. "What if order matters?" she asked. She lined them up: Red, Blue, Green. Then Red, Green, Blue. Then Blue, Red, Green. She kept going. She made six different lines. "See? The order changed each time."

"This is a **permutation**," she explained. "It's when the order of things is important." She stacked the cubes again. "For three different things, there are three times two times one ways to arrange them. That's six ways." She made a quick stack. "We call that 'three factorial.' It's written as 3!."

Tally then picked up five cubes. Red, blue, green, yellow, purple. She put them in a tiny basket. "Now, what if order doesn't matter?" she asked. "We just want to *choose* three cubes from these five."

She pulled out red, blue, green. "Okay, one choice." She put them back. Then she pulled out blue, green, yellow. "Another choice." She kept choosing groups of three. She didn't care if she picked red first or green first. Just which three cubes ended up together. "These are **combinations**," she said. "The order doesn't matter here. You just pick a group." She showed ten different groups of three cubes. "There are ten ways to choose three cubes from five. Even though the order doesn't matter, there's still a way to count it."

Tally never made counting sound hard. She never made it sound like only smart people could do it. "It's not hard," she always said. "It's just *systematic multiplication*. You don't have to list every single way."

She taught simple rules:

- **Multiplication rule**: Choices for step one times choices for step two.
- **Permutations**: Order matters. Use factorials.
- **Combinations**: Order doesn't matter. Use adjusted factorials.
- **Factorial**: Like 3! means 3 × 2 × 1.
- **Pascal's triangle**: A neat pattern that helps with combinations.
- **Counting via cases**: Sometimes you break a big problem into smaller parts. Then you add them up.

Tally grew up in a small village. Her family had a special job. They were the market-arrangers. They set up all the market stalls. They didn't just dump things out. Oh no. They thought about *how many ways* to show things. How many ways could the nuts be stacked? How many ways could the berries be lined up? Tally watched them. She learned their ways. She saw patterns everywhere.

When she was older, she walked to DiscreteQuest. It was a long journey. She finally met the mentor. The mentor was very wise. "What is **combinatorics**?" the mentor asked.

Tally held up her pouch of cubes. She made a quick stack. "It's counting arrangements," she said. "But you do it in a smart way. You use the **multiplication rule**. You use **permutations**. You use **combinations**." She tapped the stack. "The math does the counting. You don't have to list them all."

The mentor smiled. "You are appointed," the mentor said.

Tally still says it often. "It is not hard. It is *systematic multiplication*. You don't have to list each way."

The cubes stack in another arrangement.

Listen along + meet more of the cast at:

`https://spark-and-anvil.com/cast/discretequest/tally`

Verity the Truth-Tester

*PROPOSITIONAL LOGIC + TRUTH TABLES — *AND, OR, NOT operators; truth tables enumerate all cases.**

Verity was a small owl. Her feathers were warm brown and cream. She always carried a small, folding grid in her wing-pocket. It was her **truth-table** grid.

She liked things to be complete. Row by row. No missing pieces. Verity was steady-eyed and careful. She loved seeing a problem solved perfectly.

Today, she was checking the weather. Not just for fun. She was helping her friend, Pip, decide on a picnic. Pip was a squirrel. He loved nuts and sunshine.

Verity tapped a claw on her grid. "First, we need **propositions**," she chirped.

Pip tilted his head. "Propo-what now?"

"**Propositions**," Verity explained. "They're just statements. Things that are either true or false. No maybe-so. No sort-of."

She wrote on her grid. "P: It will rain today." Then she wrote, "Q: The sun will shine."

Pip giggled. "Those can't both be true!"

Verity nodded. "Exactly! That's where *connectives* come in. Like **AND**." She drew a little upside-down V. "For **AND**, both statements have to be true. If P is true AND Q is true, then the whole thing is true."

She looked at the sky. "If it rains AND the sun shines, then we can have a picnic."

Pip frowned. "But if it rains, the sun won't shine. So that's false."

"Right!" Verity said. "Now, what about **OR**?" She drew a regular V. "For **OR**, only one statement needs to be true. Or both!"

"If it rains OR the sun shines, we can have a picnic."

Pip thought. "So if it rains, we can picnic. If the sun shines, we can picnic. If both happen, we can picnic. But if neither happens, no picnic!"

"You got it!" Verity chirped. "And then there's **NOT**." She drew a little wavy line. "It just flips things. If 'It is raining' is true, then 'NOT it is raining' is false."

Verity unfolded her grid. It had four rows. "This is a **truth table**," she said. "It shows all the ways things can be true or false."

She pointed to the first row. "What if P (rain) is true, and Q (sun) is true?"

Pip shook his head. "Can't happen!"

Verity smiled. "But we still check it. Just in case. For the **AND** statement, that row would be False."

She filled in the row. "Next row: P is true, Q is false. Rain, no sun. For **AND**, still False. For **OR**, True!"

She worked through the rows. "P false, Q true. No rain, sun. **AND** is False. **OR** is True." Finally, "P false, Q false. No rain, no sun. **AND** is False. **OR** is False."

Pip watched her fill the last column. "So the table shows the answer for every single possibility!" he said.

Verity beamed. "Exactly! Each row is one case. We check each row consistently."

Verity never made truth tables sound hard. "It's not about being super smart," she'd say. "It's about being super careful. Check each row. That's the secret."

She showed Pip other connectives. "There's **XOR**," she said. "That means *exclusive or*. It's true only when *exactly one* thing is true. Like if you can have ice cream OR cake, but not both."

Then there was **Implication**. "This one is tricky," Verity warned. "It's like saying, 'IF you clean your room, THEN you can play games.' This statement is only false in one way. That's if you *do* clean your room. But your parents *don't* let you play games. That would be unfair!"

And **Equivalence**. "That means they're the same," Verity explained. "If P is true and Q is true, then they're equivalent. If P is false and Q is false, they're equivalent. They match!"

Sometimes, a whole statement would always be true. No matter what P and Q were. "That's a **tautology**," Verity chirped. "Like saying, 'It is raining OR it is NOT raining.' That's always true!"

And if it was always false? "A **contradiction**," she'd say. "Like 'It is raining AND it is NOT raining.' That can never be true!"

Verity grew up in a quiet village. Her family were the village day-watchers. They were owls, just like her. Their job was to record everything important each day.

Did it rain? Was the river high? Could travelers cross the bridge? They wrote it all down. Every single day. They used their own simple truth tables.

Verity's grandmother would say, "We need to know *all* the conditions. If it rained AND the river was high, the bridge might be unsafe. If it didn't rain AND the river was low, the bridge was fine."

Young Verity watched them. She learned to fill in the grids. Row by row. Making sure every possibility was covered. It was how they kept the village safe.

When Verity was old enough, she walked to DiscreteQuest. It was a big, tall building. Full of smart creatures. A wise old badger was the mentor there.

The badger looked at Verity. "What is **propositional logic**?" he rumbled.

Verity didn't blink. She pulled out her small grid.

"It's about statements that are true or false," she said. "And how they connect with **AND**, **OR**, and **NOT**."

She tapped her grid. "**Truth tables** show all the cases. Every single one. Each row is one combination of true and false. The pattern of T/F across the rows? That's what defines the connection."

The badger smiled. "You are appointed," he said. Verity felt a warm flutter in her chest. She had shown him her way. Her careful, row-by-row way.

She still says it to anyone who asks. "It is not hard. It is *check each row.* The pattern defines the connective." And she always has her grid ready.

Listen along + meet more of the cast at:

https://spark-and-anvil.com/cast/discretequest/verity

Wander the Bridge-Walker

*GRAPH THEORY — *Eulerian paths, Hamiltonian paths, connectivity.* The discrete-math primitive of *vertices + edges as the structure of network problems.**

Wander was a crane-tween. She was small for her age. But her legs were long. She moved with careful, steady steps. Her grey-and-white feathers were always neat. In her hand, she always carried a small, folded map. It was old and crinkled. It was her most important thing.

The map showed an old town. It had many landmasses. These were like little islands. Bridges connected them. Wander called the landmasses *vertices*. She called the bridges *edges*. When Wander thought about a problem, she used her map. She would trace paths with her finger. Her finger would walk along the edges. It would stop at the vertices. This was her way of doing things.

One sunny afternoon, a new student named Pip watched Wander. Pip was a squirrel-kit. He looked confused. Wander was tracing a path on her map.

"What are you doing?" Pip asked. He tilted his head.

Wander looked up. Her eyes were bright. "I am walking the graph," she said. She held out her map. "See? These are *vertices*. They are the land bits. And these are *edges*. They are the bridges."

Pip peered at the map. "So, like, a subway map?" he asked.

"Exactly!" Wander nodded. "Or friends on a playground. Or even computers talking. They are all *graphs*."

"What's a graph?" Pip asked.

"It's simple," Wander said. She tapped the map. "It's just *vertices* connected by *edges*. That's all."

"Oh," Pip said. He still looked a little lost.

"Sometimes," Wander continued, "you want to visit every bridge. You want to walk every single *edge*. But only once." She traced a winding path. Her finger moved slowly. "If you can do that, it's called an **Eulerian path**."

Pip frowned. "Why would I want to do that?"

"Maybe you're a bridge inspector," Wander said. "You need to check every bridge. But you don't want to walk the same bridge twice. That would waste time." She smiled. "My family did that."

"What if you want to visit every land bit?" Pip asked. "Every *vertex*?"

"Ah, that's different!" Wander said. Her finger moved again. This time, it jumped from landmass to landmass. It touched each *vertex*. "If you visit every *vertex* exactly once, that's a **Hamiltonian path**. Very different rules."

"So, **Eulerian** is about bridges," Pip said. "And **Hamiltonian** is about land?"

"You got it," Wander said. "Walk every *edge*: **Eulerian**. Walk every *vertex*: **Hamiltonian*. Different rules for different problems."

Wander grew up in a place called Bridge-Village. It was a funny name. But it made perfect sense. Bridges stretched everywhere. They crisscrossed over rivers. They connected tiny islands. Her family were the village's bridge-walkers. They were cranes, just like Wander.

Every morning, they had an important job. They walked the village's many bridges. They checked each one. Was this bridge safe? Could that one hold a cart? They recorded everything. Young Wander learned this job early. She learned to trace paths. She learned to see connections.

She loved her map. It was a hand-drawn copy. It showed every bridge. It showed every landmass. She knew the village's network by heart. She knew which paths were open. She knew which paths were closed.

Sometimes, a bridge would break. Then a part of the village became cut off. "Can you get from this island to that one?" her father would ask. Wander would look at her map. She would trace with her finger. "No," she might say. "Not anymore. That bridge is out."

This was about *connectivity*. Can you get from any *vertex* to any other *vertex*? If you can't, the graph is *disconnected*. Some places are unreachable. Wander understood this deeply. She saw it every day.

When Wander was older, she heard about DiscreteQuest. It was a special school. She walked a long way to get there. The head mentor was a wise old owl. He looked at her carefully.

"What is graph theory?" the mentor asked. His voice was deep.

Wander held out her map. "It's *vertices* and *edges*," she said. Her voice was clear. "You walk every *edge*, or you walk every *vertex*. They are different rules. It's how you solve network problems."

The mentor smiled. "You are appointed," he said.

Wander never thought these things were hard. She just saw them. She saw the paths. She saw the connections. "It's not hard," she always said. "It's just *vertices* and *edges*. And you walk the paths."

Sometimes, the edges had arrows. "That means you can only go one way," she would explain. "Like a one-way street. Those are *directed edges*." If there were no arrows, you could go both ways. Those were *undirected edges*.

She showed how a path was just a sequence of *vertices*. You moved from one to the next. You followed the *edges*. She showed how some graphs had no cycles. They were like trees. You could not go in a circle.

Wander helped anyone who asked. She made everything simple. She made it real. She used her map. She used her finger. She showed them the way.

Listen along + meet more of the cast at:

https://spark-and-anvil.com/cast/discretequest/wander

About Spark & Anvil

Spark & Anvil is a 501(c)(3) public charity. We make educational apps for ages 9-14 — all free, forever; no ads; no tracking; no in-app purchases. DiscreteQuest is one of 140+ apps in the portfolio.

More chapter books from Spark & Anvil

Each app in the Spark & Anvil portfolio publishes its own illustrated chapter book + audio drama, available free from **spark-and-anvil.com/books**. Highlights include:

- **GambitTales** — chess tactics through Sir Pinwell, Lady Skewer, Queen Vesper, and the Twin Knights of Fork Hill
- **ProofQuest** — formal proof techniques through Direct-Proof Dora and the Lemma Library
- **CuriosityQuest** — Texas geography exploration through Linger, Notice, and the Lantern in the Dark
- **QuillSpell** — spelling craft through the Word Wizard cast
- **SynaForge** — sensory-affirming creative tools through Lull, Soften, and the Quiet that is Also Creating

Methodology

Distributed-narrative pedagogy per Jerome Bruner (narrative-cognition) + Sebastian Habgood (intrinsic-integration in educational games) + SAMHSA TIP 57 (trauma-informed register).

Trauma-informed-design framework per Eggleston et al. (2025) and Stoltenburg et al. (2024).

License

Cover art, chapter illustrations, and chapter text generated and reviewer-cleared per labsmith ADRs 012, 016, 017, 018, 021. Audio drama transcripts available at spark-and-anvil.com/cast.