r/ThingsYouDidntKnow • 2 mo. ago
TheStocksGuy

# Number Theory: Dirichlet Theorem and Ulam Spiral

## Number Theory and Prime Visualization using HTML and JavaScript

In this post, I present a method to visualize prime numbers using an interactive HTML and JavaScript code snippet. This visualization aims to showcase primes in a spiral pattern, leveraging color coding and animation for an engaging experience.

## Code Explanation

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Enhanced Prime Visualization</title>
    <style>
        body {
            background-color: black;
            color: white;
        }
        canvas {
            display: block;
            margin: auto;
            background-color: black;
        }
    </style>
</head>
<body>
    <canvas id="primeCanvas" width="1200" height="1200"></canvas>
    <script>
        const canvas = document.getElementById('primeCanvas');
        const ctx = canvas.getContext('2d');
        const width = canvas.width;
        const height = canvas.height;
        const colors = ['#FF0000', '#FFA500', '#FFFF00', '#00FF00', '#00FFFF', '#0000FF', '#FF00FF
        let primes = [];
        let zoomLevel = 1;
        let offsetX = 0;
        let offsetY = 0;
        let animationFrameId;
        let isAnimating = true;
```

```
function isPrime(num) {
    if (num <= 1) return false;
    if (num <= 3) return true;
    if (num % 2 === 0 || num % 3 === 0) return false;
    for (let i = 5; i * i <= num; i += 6) {
        if (num % i === 0 || num % (i + 2) === 0) return false;
    }
    return true;
}

// Function to generate all primes up to a specified number
function generatePrimes() {
    for (let num = 2; num < numofPrimes; num++) {
        if (isPrime(num)) {
            primes.push(num);
        }
    }
}

// Function to draw the prime chart
function drawPrimeChart() {
    const centerX = width / 2;
    const centerY = height / 2;
    let angleIncrement = 2 * Math.PI / 360;
    let radius = 10;

    ctx.clearRect(0, 0, canvas.width, canvas.height);
    ctx.save();
    ctx.translate(offsetX, offsetY);
    ctx.scale(zoomLevel, zoomLevel);
    ctx.translate(centerX, centerY);

    primes.forEach((prime, index) => {
        let angle = index * angleIncrement;
        let x = radius * Math.cos(angle);
        let y = radius * Math.sin(angle);
        ctx.fillStyle = colors[(index % 9)];
        ctx.fillText(prime, x, y);
        if (index > 0) {
            let prevAngle = (index - 1) * angleIncrement;
            let prevX = radius * Math.cos(prevAngle);
            let prevY = radius * Math.sin(prevAngle);
            ctx.strokeStyle = colors[(index % 9)];
            ctx.beginPath();
            ctx.moveTo(prevX, prevY);
            ctx.lineTo(x, y);
```

```
            });

            ctx.restore();
        }

        // Function to animate the prime chart
        function animate() {
            primes.push(primes.shift()); // Rotate primes
            drawPrimeChart();
            if (isAnimating) {
                animationFrameId = requestAnimationFrame(animate);
            }
        }

        canvas.addEventListener('wheel', function(event) {
            const mouseX = event.offsetX;
            const mouseY = event.offsetY;

            if (event.deltaY < 0) {
                zoomLevel *= 1.1;
                offsetX = mouseX - (mouseX - offsetX) * 1.1;
                offsetY = mouseY - (mouseY - offsetY) * 1.1;
            } else {
                zoomLevel /= 1.1;
                offsetX = mouseX - (mouseX - offsetX) / 1.1;
                offsetY = mouseY - (mouseY - offsetY) / 1.1;
            }
            drawPrimeChart();
        });

        canvas.addEventListener('click', function() {
            if (isAnimating) {
                cancelAnimationFrame(animationFrameId);
            } else {
                animate();
            }
            isAnimating = !isAnimating;
        });

        // Generate primes and start the visualization
        generatePrimes();
        drawPrimeChart();
        animate();
    </script>
```

## Explanation of the Code

1. **HTML and CSS**: The `<style>` section sets up a black background and white text for a striking contrast. The canvas is centered and set to a black background.
2. **JavaScript**:
   - **Canvas Setup**: `const canvas = document.getElementById('primeCanvas');` and `const ctx = canvas.getContext('2d');` initialize the canvas and context.
   - **Color Array**: Defines an array of neon colors for visual appeal.
   - **Prime Generation**: `generatePrimes()` function generates all prime numbers up to `numofPrimes` (30,000 in this case) using the `isPrime` function.
   - **Prime Spiral Drawing**: `drawPrimeChart()` function visualizes the primes in a spiral pattern. It calculates positions based on angle and radius, and draws lines connecting successive primes.
   - **Zoom and Pan**: The `canvas.addEventListener('wheel', ...)` function allows zooming in and out using the mouse wheel, adjusting the scale and translation of the canvas.
   - **Animation**: The `animate()` function continuously rotates the primes for a dynamic effect. Clicking the canvas toggles the animation on and off.

This visualization provides a unique way to explore the distribution of prime numbers and observe interesting patterns. Feel free to try the code and experiment with different settings!

## Mathematical Concepts Behind the Visualization

1. **Prime Numbers**:
   - A prime number is a natural number greater than 1 that has no positive divisors other than 1 and itself. Examples include 2, 3, 5, 7, 11, etc.
2. **Prime Generation**:
   - To find prime numbers, we typically use an algorithm like the Sieve of Eratosthenes. This involves iteratively marking the multiples of each prime, starting with 2, and removing them from the list of numbers.
3. **Spiral Pattern**:
   - We arrange numbers in a spiral pattern starting from the center and moving outward. Each successive number is placed in the next position in the spiral.
   - Mathematically, this involves incrementing angles and radii in polar coordinates. For example, if you start at the origin (0, 0), you move to (1, 0), then to (1, 1), and so on, forming a spiral.
4. **Polar Coordinates**:
   - The position of each number in the spiral can be described using polar coordinates $(r, \theta)$, where $r$ is the radius (distance from the center) and $\theta$ is the angle.
   - For example, the nth prime is placed at $(r, \theta)$ where $r$ increases gradually, and $\theta$ is a multiple of a small angle increment.

- Draw a large grid on paper, starting from the center and extending outward. This grid will help you place each number in its correct position.

2. **Number Placement**:
   - Begin at the center of the grid. Place the number 1 at the center.
   - Move to the right and place the number 2, then up to place 3, left to place 4, down to place 5, and so on, forming a spiral. Continue this until you have filled the desired portion of the grid with numbers.
3. **Identifying Primes**:
   - As you place each number, check if it is a prime. You can use the Sieve of Eratosthenes or simple divisibility rules to determine primality.
   - Highlight prime numbers using a different color or marker. For instance, circle prime numbers or shade them with a bright color.
4. **Connecting Primes**:
   - Draw lines connecting successive prime numbers. This will help visualize the patterns in their distribution.
   - Ensure the lines are clear and do not overlap excessively, maintaining a neat and organized visual.
5. **Angle and Radius Calculation**:
   - Calculate the angle and radius for each step manually. For a more accurate placement, use a protractor to measure angles and a ruler for distances.
   - Increase the radius slightly for each new prime to maintain the spiral's shape and keep the numbers spaced out.

## Detailed Example

Suppose you want to visualize primes from 1 to 100:

1. Draw a grid with 10x10 squares, centered on the page.
2. Start at the center (origin) and place the number 1.
3. Move to the right for the number 2, then up for 3, left for 4, and down for 5, forming a spiral.
4. As you place each number, check if it is a prime:
   - 2: Prime, highlight it.
   - 3: Prime, highlight it.
   - 4: Not prime.
   - 5: Prime, highlight it.
5. Continue this process, checking for prime numbers and drawing lines to connect them.

By following these steps, you can create a visually appealing and mathematically accurate representation of prime numbers in a spiral on paper, similar to what the HTML and JavaScript code does digitally. This method combines both artistic and mathematical skills, offering a hands-on approach to exploring the fascinating world of prime numbers.

## Number Theory: Dirichlet Theorem and Ulam Spiral

# Dirichlet Theorem

The Dirichlet Theorem on arithmetic progressions states that for any two positive coprime integers (a) and (d), there are infinitely many primes of the form (a + nd) (where (n) is a non-negative integer). This theorem highlights the rich and predictable structure of prime numbers within arithmetic sequences. It underscores that primes are not merely random but exhibit a certain order even when distributed within seemingly simple linear patterns.

# Ulam Spiral

The Ulam Spiral, discovered by Stanislaw Ulam, is a graphical representation of the prime numbers in a spiral format. Starting from the center with the number 1, numbers are placed sequentially in a spiral pattern. Prime numbers, when highlighted, reveal surprising diagonal alignments and patterns, providing a visual clue to the underlying structure of primes.

# Visualization Approach

By combining the mathematical rigor of the Dirichlet Theorem with the visual clarity of the Ulam Spiral, we can explore prime numbers in a more profound way. Here's how you can visualize this on paper or digitally:

1. **Prime Generation**: Use algorithms to generate prime numbers up to a certain limit.
2. **Spiral Placement**: Arrange numbers in a spiral, starting from the center and moving outward, calculating positions using polar coordinates (radius (r) and angle (\theta)).
3. **Highlighting Primes**: Identify prime numbers and highlight them with distinct colors or markers.
4. **Pattern Observation**: Connect successive primes to observe patterns and alignments, providing both a numerical and visual exploration of primes.

# Conclusion

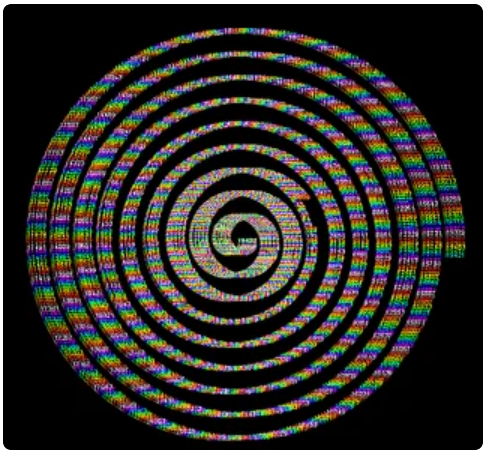
These methods showcase that prime numbers, far from being chaotic, exhibit intriguing structures and patterns that can be explored both mathematically and visually. The Dirichlet Theorem offers a theoretical framework for understanding prime distribution in arithmetic progressions, while the Ulam Spiral provides a compelling visual representation. Together, they enrich our understanding of the enigmatic world of prime numbers.

Add a comment

Sort by: Best

Search Comments

**TheStocksGuy** OP • 2mo ago



This showcases the code in motion just for your iinformation.

1

Approved 2 months ago