Appeasing the Cherry Blossom Horde

Vamshi Jandhyala

4/4/2025

A nice puzzle by Xavier Durawa for the week of 3/30.

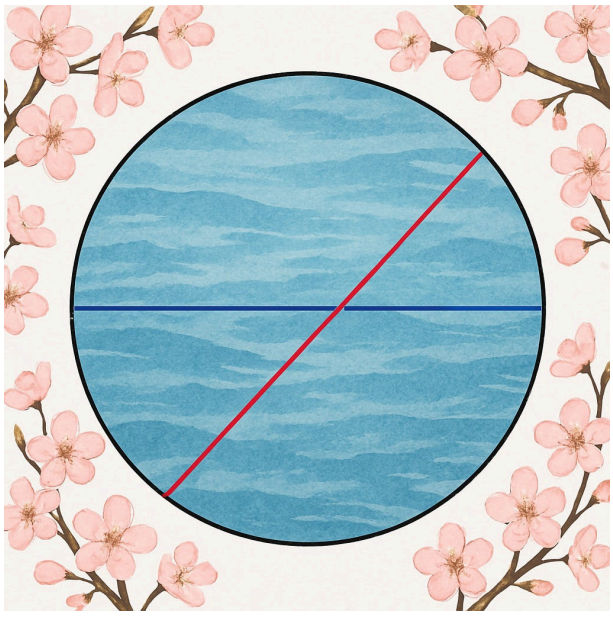
Contents

1 Puzzle

This weekend marks the peak bloom for D.C.'s cherry blossoms 🌸, and as always, there's a ton of foot traffic around the Tidal Basin—a circular-ish body of water just off the Potomac.

Now, imagine that in an alternate universe, the D.C. government decided (questionably) to build a floating walking path straight across the middle of the basin. The idea was to help with pedestrian traffic flow — even if it meant ruining some of the scenic views 😅.

While the path was being constructed, the parks department needed to fence it off to prevent premature use. So they put up a temporary barrier somewhere across the basin. The only rule? The barrier had to intersect the floating path, effectively blocking it. Other than that, the barrier's placement was completely random. (For simplicity, let's model the Tidal Basin as a perfect circle, the floating path as a diameter, and the barrier as a random chord that intersects the diameter.)



Now that construction is done, you — a humble parks worker — are on your way to remove the barrier and finally open the path to the public. But halfway there, you realize…you forgot the map! The map would've told you which of the two ends of the floating path the barrier is closer to, so you could have picked the shorter walk.

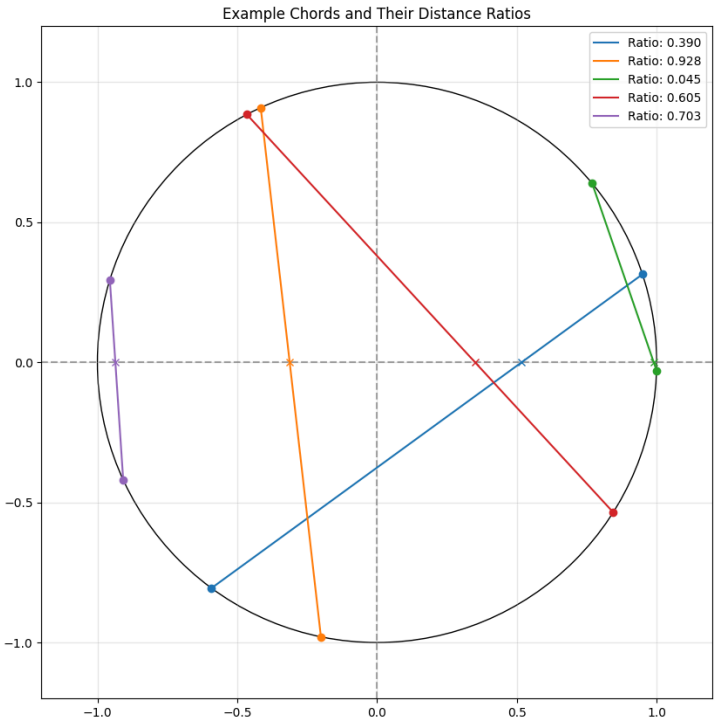
You're not sure if it's worth turning back, so instead you ask: What's the expected ratio of the shorter segment of the path to the longer one, given that the barrier randomly intersects the path?

In other words, if the barrier cuts the floating path into two segments, what is the expected value of the ratio:

$$\frac{\text{length of shorter segment}}{\text{length of longer segment}}$$

2 Brute force simulation

As we are dealing with ratios, we can assume that the radius of the circle is 1. We can also assume that the path is on the x −axis. Given that the barrier intersects the path, we can randomly generate a point that lies on the upper semi-circle and one on the lower semi-circle, calculate the point of intersection and the lengths of the two segments of the chord. We can then repeat this process a million times to estimate the expected value of the ratio of the shorter segment to the longer segment. The Python code below performs the above simulation and we see that the expected value of the ratio is **0.5619**.



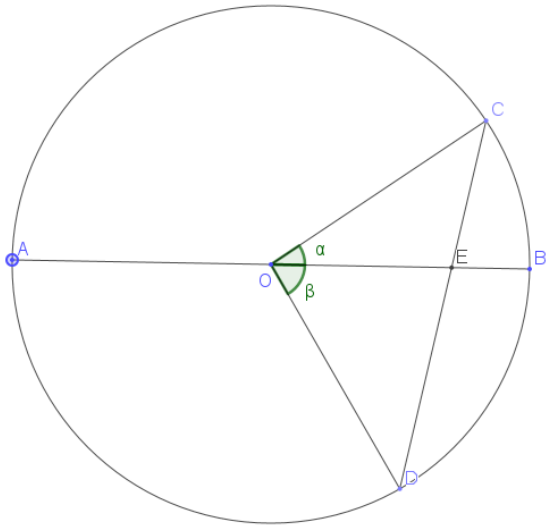
2.1 Python code

```
import numpy as np

def generate_points_and_calculate_ratio(num_samples=1000000):
```

```
    ratios = np.zeros(num_samples)

    for i in range(num_samples):
        theta1 = np.random.uniform(0, np.pi)
        theta2 = np.random.uniform(np.pi, 2*np.pi)
        x1, y1 = np.cos(theta1), np.sin(theta1)  # Upper point
        x2, y2 = np.cos(theta2), np.sin(theta2)  # Lower point
        if x2 - x1 != 0:  # Avoid division by zero
            m = (y2 - y1) / (x2 - x1)
            b = y1 - m * x1

            # Find intersection with x-axis (where y = 0)
            x_intersect = -b / m
            y_intersect = 0
        else:
            # Vertical line case
            x_intersect = x1
            y_intersect = 0

        d1 = np.sqrt((x_intersect - x1)**2 + (y_intersect - y1)**2)
        d2 = np.sqrt((x_intersect - x2)**2 + (y_intersect - y2)**2)
        shorter = min(d1, d2)
        longer = max(d1, d2)
        ratios[i] = shorter / longer
    return ratios

ratios = generate_points_and_calculate_ratio(num_samples=1000000)
mean_ratio = np.mean(ratios)
print(f"Expected value (mean) of the ratio: {mean_ratio:.6f}")
```

3 Analytical solution

In the figure below, AB is the walking path and CD is the floating barrier. It is easy to see that for every chord CD, we have the corresponding triangle OCD, where $0 < \alpha, \beta \leqslant \frac{\pi}{2}$.



Using **Sine Rule**, the ratio of the shorter segment to the longer segment is

$$\frac{\sin(\alpha)}{\sin(\beta)}, \alpha < \beta. \tag{3.1}$$

Let A be the random variable for the $\angle\alpha$ and B be the random variable for the $\angle\beta$. A and B are independent uniform random variables on $\left(90, \frac{\pi}{2}\right]$. We have $f_A(\alpha) = \frac{2}{\pi}$, $f_B(\beta) = \frac{2}{\pi}$ and $f_{AB}(\alpha, \beta) = \left(\frac{2}{\pi}\right)\left(\frac{2}{\pi}\right)$. The expected value of the ratio is

$$\begin{aligned}
\mathbb{E}\left[\frac{\sin(A)}{\sin(B)} \middle| A < B\right] &= \frac{\mathbb{E}\left[\frac{\sin(A)}{\sin(B)} \cdot \mathbb{1}_{A<B}\right]}{\mathbb{P}[A < B]} \\
&= \frac{\int_0^{\frac{\pi}{2}} \int_0^{\beta} \frac{\sin(\alpha)}{\sin(\beta)} \left(\frac{2}{\pi}\right)\left(\frac{2}{\pi}\right) d\alpha d\beta}{\frac{1}{2}} \\
&= \frac{8}{\pi^2} \int_0^{\frac{\pi}{2}} \frac{1 - \cos(\beta)}{\sin(\beta)} d\beta \\
&= \frac{8}{\pi^2} \log(2) \\
&= 0.5618.
\end{aligned} \tag{3.2}$$

4 Simulation using a simplified approach

Using the Python code and carrying out Monte Carlo simulation, we see that the expected value of the ratio is 0.5625 which is very close to the theoretical value as expected.

4.1 Python code

```python
import numpy as np

def monte_carlo_conditional_expectation(num_samples=1000000):
    x = np.random.uniform(0, np.pi/2, num_samples)
    y = np.random.uniform(0, np.pi/2, num_samples)
    indicator = x < y
    ratio = np.zeros_like(x)
    ratio[indicator] = np.sin(x[indicator]) / np.sin(y[indicator])
    return np.mean(ratio[indicator])

print(monte_carlo_conditional_expectation())
```