🔍 Basic Concept of the Flajolet-Martin (FM) Algorithm

(as used in Intrudex project for network anomaly detection)

🧠 Problem It Solves:

In large-scale network data, it's often important to know:

> "How many **unique IP addresses** are hitting our system over time?"

But simply counting every distinct IP (using sets or hash maps) is:

- **Memory-intensive**
- **Slow** for high-volume, real-time data

⚙️ What is the FM Algorithm?

The **Flajolet-Martin algorithm** is a **probabilistic method** for estimating the **number of distinct elements** in a data stream — efficiently and with minimal memory.

📊 Core Idea:

Every IP address is **hashed** into a binary number. We then observe **how many trailing zeros** appear in the binary result.

Why? Because:

- Rare events (like a hash ending in many zeros) are more likely when there are many **unique items**.
- So, more unique items → higher chance of seeing more trailing zeros.

🧩 Step-by-Step (in simple terms):

1. **Hash the IP address** (using MD5 in the code).

   Example:

```
IP: 192.168.1.1 → hash: 100101000...0000
```

2. **Count trailing zeros** in the binary hash.

```
100100000 → 4 trailing zeros
```

3. **Track the maximum number of trailing zeros** seen.

```
Max so far = 7 → Estimate ≈ 2^7 = 128
```

4. **Repeat** using multiple hash seeds for accuracy.
5. Use a **correction constant** (φ ≈ 0.77351) for bias:

$$\text{Estimate} = \frac{2^{\text{max trailing zeros}}}{\phi}$$

6. **Average the results** from all hash functions.

📦 In this Project:

Apply this FM estimate over **sliding windows** of data (e.g., every 100 packets).

For each window:

- **Estimate** how many distinct Source IPs are present.
- Compare it to the **actual count** to measure accuracy.
- Flag it as **suspicious** if the estimate is unusually high → Possible DDoS or scan.

🛡️ Why It's Useful for Security:

If suddenly a lot of **new unique IPs** appear in a short time:

- It may be a **botnet attack**, **network scan**, or **reconnaissance**.

Since FM is lightweight, you can:

- Run it on **live traffic**
- Detect **anomalies early** without storing full IP logs

🪟 What is a Sliding Window?

A **sliding window** is a technique used in stream or time-series data processing where you analyze a **fixed-sized chunk** (window) of data, then **slide the window forward** by a certain step, and repeat the analysis.

🔁 How It Works

Think of your data as a long row of items:

```
[ Packet1, Packet2, Packet3, ..., Packet1000 ]
```

Now imagine looking at **100 packets at a time**:

- First window: [1 - 100]
- Slide forward 50: [51 - 150]

- Slide forward 50: [101 - 200]
- … and so on.

We are analyzing network traffic to detect anomalies in unique source IPs.

```
window_size = 100
step = 50
```

This means:

- To **analyze every 100 rows (packets)** of the CSV.
- Then **slide forward by 50 rows** for the next window.
- This creates **overlapping windows**, allowing for smoother detection of trends or anomalies.

📌 **Example:** Let’s say the data looks like this:

Row	Source IP
1	192.168.1.1
2	10.0.0.5
…	…
100	172.16.0.1
101	203.0.113.10
…	…

With

window_size = 100

,

step = 50

:

- **Window 1** → rows 1 to 100
- **Window 2** → rows 51 to 150
- **Window 3** → rows 101 to 200

🧠 Why Use a Sliding Window?

- ✅ **Real-Time Detection:** It helps catch changes in the stream as they happen.
- ✅ **Resource Efficiency:** Only a portion of data is processed at a time.
- ✅ **Localized Analysis:** You can spot **short-term spikes** or anomalies.
- ✅ **Smooth Monitoring:** Overlapping windows reduce the chance of missing patterns at window boundaries.

🛡️ In Security Context

In the anomaly detection use case:

- Each window gets a **distinct IP estimate** using the FM algorithm.
- If a window shows **a sudden spike** in distinct IPs, it might indicate:
  - Botnet activity
  - Network scan
  - DDoS attempt

Term	Meaning
Window Size	How many rows (packets) you analyze at once
Step	How many rows forward the window moves after each iteration
Sliding Window	Repeated analysis on these windows to detect trends or outliers

⚠️ Disclaimer

This project is intended **solely for research and educational purposes**. It demonstrates the use of probabilistic algorithms (like the Flajolet-Martin algorithm) to estimate unique elements in a data stream and detect anomalies in a simulated or restricted network environment.

If you plan to **deploy this tool in a real-time production network**, please be advised:

- 🔐 Additional **security protocols**, **data validation layers**, and **compliance measures** must be implemented to meet organizational and legal standards.
- 📊 Live deployment should be accompanied by **real-time traffic monitoring tools**, **alerting systems**, and **fail-safes** to prevent false positives or misclassification of benign traffic.
- 📁 Ensure the **privacy and confidentiality** of all data used. Do not apply this tool to live or sensitive data without proper authorization.

By using this project, you acknowledge that:

- The author (RakshithKK) are **not responsible for any misuse** of the code.
- The tool is provided **as-is**, with **no guarantees or warranties** regarding accuracy or performance in a real-world setting.

Always consult with **network security professionals, Data Engineers** and **compliance officers** before integrating such tools into critical systems.