# Real-Time API Latency Buckets

---

## Problem Statement

**You are building a monitoring system for a web API. Each API request has a latency (response time in milliseconds) and an endpoint name.**

**Your task is to maintain real-time counts of requests in latency buckets for each endpoint. The buckets are defined as:**

1. `<100ms`
2. `<500ms`
3. `<1s`
4. `>1s`

**For each incoming request, you must update the counts per endpoint. This allows the system to quickly see the distribution of request latencies for different API endpoints.**

---

## Input Format

- **Each line represents one API request with a timestamp, endpoint name, and latency in milliseconds.**
- **Lines arrive in order of timestamps.**

**Example Input Stream:**

```
1 /login 45

2 /login 120
```

```
3 /get_user 320

4 /login 550

5 /get_user 1100

6 /get_user 200
```

- `<timestamp>` → integer seconds
- `<endpoint>` → string without spaces
- `<latency>` → integer milliseconds

---

## Output Format

- For each endpoint, maintain current counts of requests in each bucket.
- After processing each request, output the updated counts for that endpoint.

**Example Output (after each line processed):**

```
/login: <100ms=1, <500ms=0, <1s=0, >1s=0>

/login: <100ms=1, <500ms=1, <1s=0, >1s=0>

/get_user: <100ms=0, <500ms=1, <1s=0, >1s=0>

/login: <100ms=1, <500ms=1, <1s=1, >1s=0>

/get_user: <100ms=0, <500ms=1, <1s=0, >1s=1>

/get_user: <100ms=0, <500ms=2, <1s=0, >1s=1>
```

---

## Requirements / Deliverables

1. **Maintain real-time counts per latency bucket for each endpoint.**
2. **Output counts immediately after processing each request.**
3. **Handle multiple endpoints simultaneously.**
4. **Efficiently process high-volume request streams (millions of requests).**
5. **Memory usage must scale with number of endpoints, not total requests.**

---

## Constraints

- **Maximum 10,000 endpoints.**
- **Maximum stream length: unbounded, potentially millions of requests.**
- **Latency values are positive integers (milliseconds).**
- **Requests arrive ordered by timestamp.**
- **Must handle real-time updates, not batch processing.**

---

## Notes / Edge Cases

1. **Latencies exactly on bucket boundaries (e.g., 100ms, 500ms, 1000ms) should be included in the higher bucket (`<500ms` includes 100ms, `<1s` includes 500ms, `>1s` includes 1000ms).**
2. **Endpoints may start receiving requests at different times.**
3. **Endpoints may stop sending requests; counts should remain accurate for the next request.**
4. **System should support dynamic addition of new endpoints.**