背景

- **問題点**：RAPPORではあらかじめ全URLを辞書化しておく必要があるが、ウェブ上のURLは無数に存在しリスト化は不可能。

1. 入力文字列を複数のハッシュ関数で Bloom フィルタにマッピング
2. フィルタに対して永続的ランダム化（Permanent Randomized Response; PRR）
3. 報告ごとに即時ランダム化（Instantaneous Randomized Response; IRR）
4. サーバー側で多数の報告を集約し、候補文字列集合（辞書）を用いてビットパターンを逆引きし頻度を推定

提案手法：Count–Min Sketch（行×列の小さな表）でURLごとにただ１セルだけを＋1し、送信前にランダム応答（IRR）でノイズを混ぜる。

利点：大規模なURL候補リスト不要で、サーバー側ではSketchから任意のURLの訪問頻度を近似推定できる。

Count-Min Sketch (CMS) の数理的仕組み

- N = ストリーム総件数
- ε = 許容相対誤差
- δ = 失敗確率

1 データ構造

深さ	**行数 *d***	$\lceil \ln(1/\delta) \rceil$ **行**
幅	列数 *w*	$\lceil e/\varepsilon \rceil$ 列

d × *w* の整数表を持ち、各行に独立ハッシュを 1 個だけ割り当てます。
(dimacs.rutgers.edu, dsf.berkeley.edu)

2 更新

要素 x が到来したら、すべての行 i について

$$j = h_i(x), \quad C[i, j] \ \mathrel{+}= 1$$

を実行します（計 *d* 回）。操作は O(*d*) 時間、メモリは O(*w d*)。(barnasaha.net)

3 点クエリ（頻度推定）

頻度推定は要素 x の頻度推定 $\tilde{a}_x$ は、該当セルの最小値を取る：

$$\tilde{a}_x = \min_{1 \le i \le d} C[i, h_i(x)].$$

簡単に言うと

```
min_{行} 表[行, ハッシュ(要素)]
```

—つまり d 個の値の最小を返すだけ
→過小評価は起こらず、過大評価のみ。(dsf.berkeley.edu)

4 誤差保証

$$\Pr\left[\text{推定} \le \text{真値} + \varepsilon N\right] \ge 1 - \delta.$$

理由（概略）

1. 1 行での余計な上乗せの期待値は $N/w \approx \varepsilon N/e$。
2. マルコフ不等式で「誤差 > εN」の確率を ≤ 1/e に抑制。
3. 行を $d = \ln(1/\delta)$ 回独立に取り、全行で失敗しない確率を $(1 - 1/e)^d \ge 1 - \delta$ とする。(dimacs.rutgers.edu, dsf.berkeley.edu)

パラメータ設計例

- 例：ε = 0.01, δ = 0.001
  ⇒ $w \approx 272$, $d \approx 7$,
  表サイズ 1.9 k カウンタで 99.9 % の確率で ±1 % 誤差以内。(barnasaha.net)

Count–Min Sketch（CMS）具体例

以下では、具体的な数値例を使って Count–Min Sketch（CMS）がどのように動作するかをステップごとに示します。

CMS
①固定サイズの行列（カウンタ）と複数のハッシュ関数を使い、
②ストリーム上の要素を分散してインクリメントし、
③「最小値」を取ることで要素の頻度を推定します。これにより、正確な頻度を数えるには大きなメモリが必要な場合でも、限られたメモリで高速に「およその回数」を得られるしくみです (ウィキペディア).

具体例の設定

行列とハッシュ関数の準備

- 行数（depth） $d = 2$、列数（width） $w = 5$ の行列 C を用意 (ウィキペディア).
- 2 つのハッシュ関数 h_1, h_2 は、ここでは簡単化のために次のように定義します：
  - $h_1(x) \equiv (\text{文字数 of } x) \bmod 5$
  - $h_2(x) \equiv (\text{先頭文字の ASCIIコード}) \bmod 5$
    （実際にはペアワイズ独立なハッシュ関数を用います） (Computer Science Stack Exchange).

カウント対象のストリーム

以下の順序でアイテムが到来すると仮定：

```
apple, banana, apple, orange, banana, apple
```

ステップ1：初期状態

最初の行列はすべてゼロです。

	col0	col1	col2	col3	col4
h1	0	0	0	0	0
h2	0	0	0	0	0

この行列は、2×5 の $d \times w$ サイズであり、メモリ使用量はカウンタ数に比例 (ウィキペディア).

ステップ2：更新操作（Inc）

ストリーム中の各アイテムについて、以下のように行列をインクリメント

1. **apple**
   - $h_1(\text{"apple"}) = (5) \bmod 5 = 0$
   - $h_2(\text{"apple"}) = (97) \bmod 5 = 2$
     $\rightarrow C[1, 0] + +,\ C[2, 2] + +$
2. **banana**
   - $h_1(\text{"banana"}) = (6) \bmod 5 = 1$
   - $h_2(\text{"banana"}) = (98) \bmod 5 = 3$
     $\rightarrow C[1, 1] + +,\ C[2, 3] + +$
3. 以下同様に **apple, orange, banana, apple** を順に処理します。

最終的に得られる行列は例えば下記のようになります（数値は例示） (Medium):

	col0	col1	col2	col3	col4
h1	3	2	0	1	0
h2	0	1	4	1	0

- 行1（h1）は文字数ベース、行2（h2）は先頭文字 ASCII ベースで分散カウントしています ([Medium][5]).
- 同じアイテムは必ず同じセルがインクリメントされるため、一貫性があります ([florian.github.io][6]).

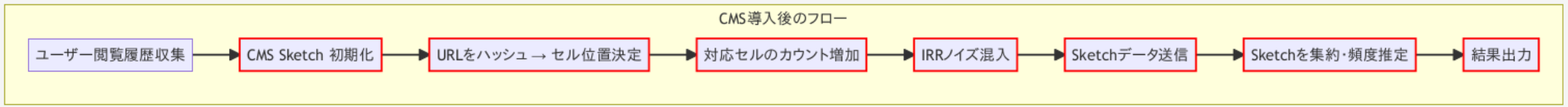
ステップ3：問い合わせ操作（Count）

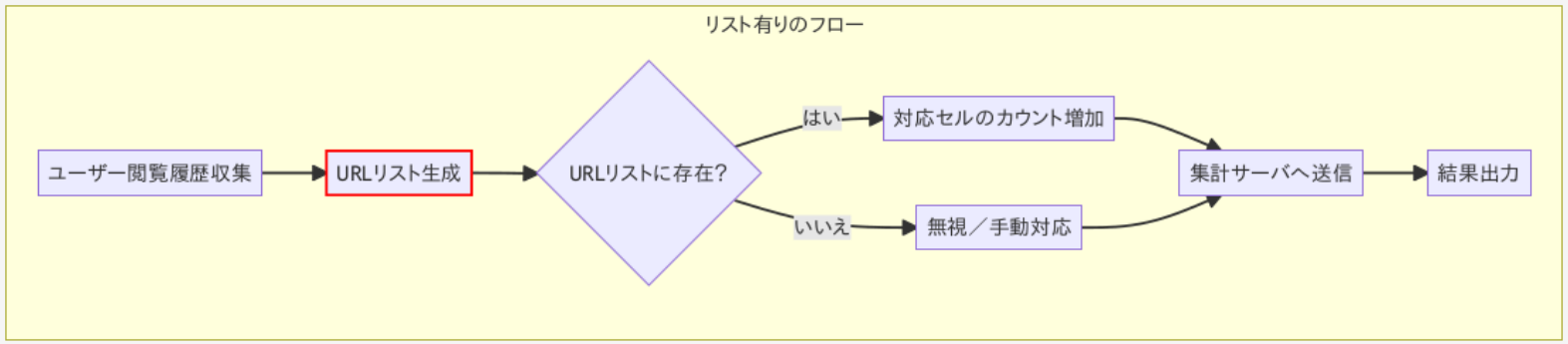
たとえば「banana の回数」は次のように推定します (ウィキペディア):

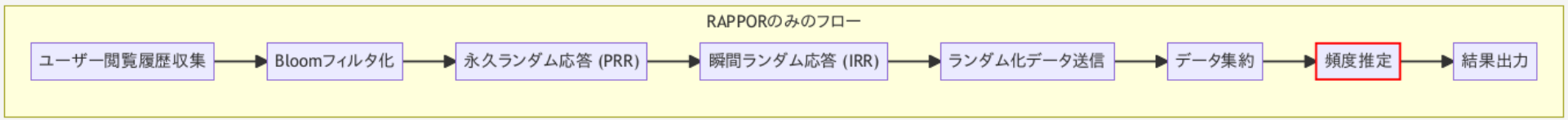
1. 行1（h1）→ col1 = 2
2. 行2（h2）→ col3 = 1

→ **推定値** = $\min\{2, 1\} = 1$

実際の banana の真値は 2 回ですが、CMS は過大評価のみを許容し、過小評価は起こしません。したがって最小値で見積もることで誤差を抑えられます (ウィキペディア).


CMS導入後のフロー
ユーザー閲覧履歴収集
CMS Sketch 初期化
URLをハッシュ → セル位置決定
対応セルのカウント増加
IRRノイズ混入
Sketchデータ送信
Sketchを集約・頻度推定
結果出力


リスト有りのフロー
ユーザー閲覧履歴収集
URLリスト生成
URLリストに存在?
はい
対応セルのカウント増加
いいえ
無視／手動対応
集計サーバへ送信
結果出力


RAPPORのみのフロー
ユーザー閲覧履歴収集
Bloomフィルタ化
永久ランダム応答 (PRR)
瞬間ランダム応答 (IRR)
ランダム化データ送信
データ集約
頻度推定
結果出力