# ストリームデータアルゴリズム

アルゴリズム論
中野
Note 11　ストリームデータアルゴリズム
2020.6.15 作成
2020.7.3 update　7.6　2021.5.31　 6.27 6.28

# 近似カウンタ

https://en.wikipedia.org/wiki/Approximate_counting_algorithm

大量のイベントを少しのメモリでカウントしたい

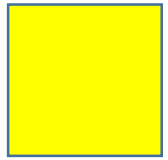
例　送信元IPアドレスごとのパケットのカウント

IPアドレスは32bit　or　8bit x 4

# 近似カウンタ

https://en.wikipedia.org/wiki/Approximate_counting_algorithm

大量のイベントを少しのメモリでカウントしたい



素朴な方法

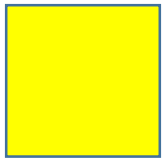
確率 1/1000でカウントを増加する

イベント1000万回　＝＞　カウンタの値　だいたい1万になる

イベント1500回のとき ＝＞　カウンタの値　1500？ 2？ 1.5? 1？ 0?

相対誤差（真の値の割合）大きいかも

# Morrisのアルゴリズム



1,2,4,8,16,…回のいずれであるかを記憶する。

8bitなら$2^{255}$まで（確率的に）カウントできる。$2^{255} > 5.78 \times 10^{76}$

つまり$2^{c-1}$のc=1,2,3,…を記憶する。ただしc=0のとき0回とする。

カウンタの増加は確率的にする。

例えば、cが

0=>1のときは確率1でカウンタを増加する。1回発生

1=>2のときは確率1でカウンタを増加する。2回発生

2=>3のときは確率1/2でカウンタを増加する。確率1/2でそのまま。4回発生

3=>4のときは確率1/4でカウンタを増加する。確率3/4でそのまま。8回発生

4=>5のときは確率1/8でカウンタを増加する。確率7/8でそのまま。　16回発生

カウンタの値が3になったとき
実際のイベントの発生回数の期待値は

| x回目 | 1 | 2 | 3 | 4 | 5… | |
|---|---|---|---|---|---|---|
| | o | o | o | | | 3 x 1/2　イベント3回の 確率は1/2 |
| | o | o | x | o | | 4 x 1/4　イベント4回の確率は1/4 |
| | o | o | x | x | o | 5 x 1/8 |
| | o | o | x | x | x | o　6 x 1/16 |
| | …… | | | | | |

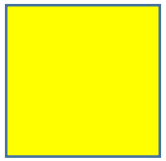
期待値

$$E = 3x\ 1/2 + 4x\ 1/4 + 5x\ 1/8 + 6x\ 1/16 + \cdots$$

$$E/2 = \quad 3x\ 1/4 + 4x\ 1/8 + 5x\ 1/16 + 6x\ 1/32 + \cdots$$

$$E/2 = 3x\ 1/2 + 1x\ 1/4 + 1x\ 1/8 + 1x\ 1/16 + 1x\ 1/32 + \cdots$$
$$= 3x\ 1/2 + 1/2$$
$$= 2$$
$$E = 4$$

イベントが2回発生のときカウンタが2（2回目）の確率



oo　1x1 =1

イベントが3回発生のときカウンタが2（2回目）の確率

oox　1x1x1/2 =1/2

イベントが4回発生のときカウンタが2（2回目）の確率

ooxx 1x1x1/2 x1/2=1/4

イベントが5回発生のときカウンタが2（2回目）の確率

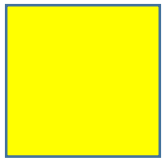
ooxxx　1x1x1/2 x1/2x1/2=1/8

。。。

イベントが10回発生のときカウンタが2（2回目）の確率

ooxxx…　1x1x1/2 x1/2x1/2…1/2=1/256

# イベントが7回発生のときカウンタが3の確率



7回発生のときカウンタの値が3, つまり$4=2^{3-1}$(4回目)になる確率は

(ooxxxxo) 1 x 1 x 1/2 x 1/2 x 1/2 x 1/2 x 1/2 +

(ooxxxox) 1 x 1 x 1/2 x 1/2 x 1/2 x 1/2 x 3/4 +

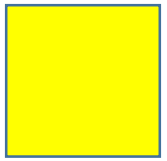
(ooxxoxx) 1 x 1 x 1/2 x 1/2 x 1/2 x 3/4 x 3/4 +

(ooxoxxx) 1 x 1 x 1/2 x 1/2 x 3/4 x 3/4 x 3/4 +

(oooxxxx) 1 x 1 x 1/2 x 3/4 x 3/4 x 3/4 x 3/4

= (16+24+36+54+81)/512 = 211/512

# イベントが7回発生のときカウンタが4の確率



(ooxxxoo) 1 x 1 x 1/2 x 1/2 x 1/2 x 1/2 x 1/4 +
(ooxxoxo) 1 x 1 x 1/2 x 1/2 x 1/2 x 3/4 x 1/4 +
(ooxoxxo) 1 x 1 x 1/2 x 1/2 x 3/4 x 3/4 x 1/4 +
(oooxxxo) 1 x 1 x 1/2 x 3/4 x 3/4 x 3/4 x 1/4 +
(ooxxoox) 1 x 1 x 1/2 x 1/2 x 1/2 x 1/4 x 3/4 +
(ooxoxox) 1 x 1 x 1/2 x 1/2 x 3/4 x 1/4 x 7/8 +
(oooxxox) 1 x 1 x 1/2 x 3/4 x 3/4 x 1/4 x 7/8 +
(ooxooxx) 1 x 1 x 1/2 x 1/2 x 1/4 x 7/8 x 7/8 +
(oooxoxx) 1 x 1 x 1/2 x 3/4 x 1/4 x 7/8 x 7/8 +
(ooooxxx) 1 x 1 x 1/2 x 1/4 x 7/8 x 7/8 x 7/8 = ??

# 次の問題を考えます

# Finding Frequent Items in Data Streams

http://archive.dimacs.rutgers.edu/Workshops/WGUnifyingTheory/Slides/cormode.pdf

大量のデータのストリーム。データn個。

メモリの制限あり。カウンタk個。

その時点までで n/k 回以上出現したデータを収集したい。

# Finding Frequent Items in Data Streams

http://archive.dimacs.rutgers.edu/Workshops/WGUnifyingTheory/Slides/cormode.pdf

大量のデータのストリーム。
メモリの制限あり。カウンタk個。
その時点までで n/k 以上出現したデータを収集したい。

高々k個のカウンタを維持する。
新データが計数中のとき　=> そのカウンタを+1。
新データが計数中でないとき
(場合1) カウンタの個数 < k　=> 新カウンタを準備して 1にする。
(場合2) カウンタの個数 = k　=> すべてのカウンタを -1

# Finding Frequent Items in Data Streams

http://archive.dimacs.rutgers.edu/Workshops/WGUnifyingTheory/Slides/cormode.pdf

高々k個のカウンタを維持する。
新データが計数中のとき　=> そのカウンタを+1。
新データが計数中でないとき
(場合1) カウンタの個数 < k　=> 新カウンタを準備して 1にする。
(場合2) カウンタの個数 = k　=> すべてのカウンタを -1

(解析)
-1 発生 => カウンタに記録されたちょうどk個分のデータと新データの合計k+1個のデータを削除
つまり -1 は 高々 n/(k+1) 回おこる
カウンタ中のデータは高々 n/(k+1) 回しか-1されない！
よって (n/(k+1)) + 1 ≦ n/k 回以上出現したデータは生きのこるはず

カウントのエラーは高々n/(k+1)

例
$n = 10^{12}$　　1テラ個
$k = 10^{6}$　　1メガ個

各カウンタ　5Byte=40bit だと
$2^{40} = 1.09 \times 10^{12}$ 回出現までカウントできる。
メモリサイズは $10^{6} \times 5 = 5$メガ Byte。
$10^{12} / 10^{6} = 10^{6} = 1$メガ回以上出現のデータを数える。
（ 0.0001%以上出現するデータを近似的に数える。）
エラーは最大でも $10^{12} / 10^{6} = 10^{6} = 1$メガ回　（1テラの$1/10^{6}$です）