DSRV©

Count distinct items approximately and PoW

in-house lightening talk

Jeongho Jeon

2024-05-31

DSRV (All That Node, WELLDONE Studio)

Cardinality

To count distinct items in a set or stream exactly,
you need to remember all distinct items.

But if approximate count is enough,
we can count with a very small amount of memory.

Leading zeros in hash values

The probability of each bit in ideal hash values being 1 is 1/2.

$0\ 1\ \cdots$	1 out of 2
$0\ 0\ 1\ \cdots$	1 out of 4
$0\ 0\ 0\ 1\ \cdots$	1 out of 8
$0\ 0\ 0\ 0\ 1\ \cdots$	1 out of 16

If the number of leading zeros in hash values is n, the chance is $1/2^n$.

If the maximum number of leading zeros among all hash values is n, we may estimate that the count of distinct items is about 2^n.

But it can be misled by even single item that has a lot of leading zeros. Then we average many estimates.

HyperLogLog (2007)

There are m (for example, 16) counters that record the maximum number of leading zeros. First bits of hash values determine which counter is used, and consider leading zeros of the remaining part.

$$\underbrace{1\ 0\ 0\ 1}_{\text{counter}}\ \underbrace{0\ 0\ 1\ \cdots}_{\text{leading zeros}}$$ leading 2 zeros in counter #5

The cardinality estimate is,

$$E = a_m m H$$

where a_m corrects a hash collision bias ($a_{16} = 0.673$, $a_{32} = 0.697$, $a_{64} = 0.709$), and H is the harmonic mean of $2^{counter}$s, that is less sensitive to outliers.

HyperLogLog is available as one of Redis data structures.

PoW

PoW mining finds a hash value of a block header, that are equal to or lesser than target.

4-bytes `nBits` field in Bitcoin block headers tells target:

$$\text{3-bytes} \times 2^{8(\text{first 1-byte} - 3)}$$

Now `nBits` is 0x17 0355f0 and target is
$0\text{x}0355\text{f}0 \times 2^{8(0\text{x}17 - 3)} \approx 3.195 \times 10^{53}$.

Difficulty shows how much harder this target is than the minimum target defined by the network. Bitcoin adjusts difficulty every 2016 blocks (about 2 weeks) to meet 10 minutes block time.

extranonce

We heard that Bitcoin mining keeps increasing nonce to find a block header hash value that are equal to or lesser than target. Current target is 3.195×10^{53}, and the chance is $\frac{3.195 \times 10^{53}}{2^{256}} = 1$ out of 3.624×10^{23}.

But **nonce** field in Bitcoin headers is 4 bytes length, and there are different $2^{8 \cdot 4} \approx 4.295 \times 10^{9}$ nonces only. We are very unlikely to find a proper nonce even if we try all nonces. Nowadays single mining device that can perform 200 TH/s takes less than 1 ms to sweep all nonces.

Then we use input part of coinbase transaction (first transaction in a block, that give the miner the block reward) as well. This part does not affect coinbase transaction, and miner reveals their identity in this part. If you change this part, Merkle root field in a block header is changed, and block hash is changed, too. It is called “extranonce”.

Mining pool

It is virtually impossible that any single miner mine a block by chance. Miners gather to form a mining pool. If mining pool mines a block, block rewards are distributed according to miners' effort. How do we measure effort fairly without rewarding abusers?

Mining pool and miners communicate via Stratum protocol. Mining pool inform miners of less difficult target (demands less leading zeros). Whenever a miner find this less difficult target as well as real target, a miner reports it to mining pool. Mining pool evaluates each miner's effort based on the number of reported shares.

CVM algorithm (2022)

Another simple probabilistic algorithm was published recently.
Let's assume that there is a bucket that contains up to 100 items.

Round 0: Fill a bucket with distinct items.

Flip a coin per item. Keep about half the items only.

Round 1: Flip a coin per item. If it comes up a head, keep the item in a bucket. Otherwise, drop the item out of a bucket.

When a bucket is full, flip a coin per item again. Keep about half the items only.

Round 2: Flip two coins per item. If all comes up heads, keep the item. Otherwise, drop the item.

When a bucket is full, flip a coin per item again. Keep about half the items only.

...

Round k: Flip k coins per item. If all comes up heads, keep the item. Otherwise, drop the item.

The estimate is the number of elements in a bucket $\times\ 2^k$.

CVM algorithm in Python

```python
from random import random

bucket_size = 100
bucket = set()
pr = 1.0

with open('input.txt') as f:
  for x in f.readlines():
    bucket.discard(x)
    if random() <= pr:
      bucket.add(x)
    if len(bucket) == bucket_size:
      new_bucket = set()
      for x in bucket:
        if random() <= 0.5:
          new_bucket.add(x)
      bucket = new_bucket
      pr = pr / 2

print(len(bucket) / pr)
```

m – total number of items

R – real distinct count

E – estimate

If you want

$Pr[(1-\epsilon)\cdot R \leq E \leq (1+\epsilon)\cdot R] \geq 1-\delta$,

the bucket size is $\lceil \frac{12}{\epsilon^2} \log_2 (\frac{8m}{\delta}) \rceil$.

Why does it work? 🤔

References

Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier.
Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm.
Discrete Mathematics & Theoretical Computer Science, DMTCS Proceedings vol. AH, 2007 Conference on Analysis of Algorithms (AofA 07), 2007.

Sourav Chakraborty, N. V. Vinodchandran, and Kuldeep S. Meel.
Distinct elements in streams: An algorithm for the (text) book.
pages 34:1–34:6, 2022.