Probabilistic Data Structure

Scribe: Min Zeng
Class date: Oct 23

Efficient data structure : for creating them and accessing them

Hashing : Good Probability. Each x is stored in location $h(x)$. If there is a collision,we chain them.

Assumption : $h()$ uniformly random. n items, m slots to place them.

False position : Some thing is not in the array but we report it.

1. $h(s)$ is its position
2. n other items $i_1, i_2, \cdots, i_n$

   ① no collision with $h(x)$

   Pr[i_1 is not at $h(x)$]=$\frac{m-1}{m}$

   Pr[i_r is not at $h(x)$]=$\frac{m-1}{m}$

   Pr[neither i_1 nor i_r is at $h(x)$]=$\frac{m-1}{m} * \frac{m-1}{m} = (1-\frac{1}{m})^2$

   Pr[none of the n items is hashed into h(x)] =$\frac{m-1}{m} * \cdots * \frac{m-1}{m} = (1-\frac{1}{m})^n \approx (1-\frac{1}{m})^{m*\frac{n}{m}} \approx e^{-\frac{n}{m}}$ (we have : $(1-\frac{1}{m})^m \approx e^{-1}$)

   ② False positive with $h(x)$[1] $= 1-(1-\frac{1}{m})^{m*\frac{n}{m}} \approx 1-e^{-\frac{n}{m}}$

   when $\frac{n}{m} \to 0$,which is small (good for one hash function)

Bloom Filter : uses many hash functions to reduce errors

Insert(x) set $A_i(h_i(x)) = 1$ $1 \le i \le k$

Query(x) return $A_1(h_1(x)) \wedge A_2(h_2(x)) \wedge \cdots \wedge A_k(h_k(x))$

if all the items are 1,return 1,else return 0

False positive rate : to choose best parameter k, m according to n

False positive probability for one array=$1-e^{-\frac{n}{m}} = 1-(1-\frac{1}{m})^n$

for k arrays? $(1-e^{-\frac{n}{m}})^k$

it further reduce the false positive error , best choice of k , m.

[1] somebody hit $h(x)$

Counting-Min Sketching : Some thing is not in the array but we report it. return min of the counts.we may have errors,but error is one sided $\geq$ the real count.

An $l \times b$ array to support two operations: Inc(x) and Count(x)

1. return frequency of x.
2. inc(x):invoked each time when x is entered.

two parameters : b (the bucket) =(2000) , l (the number of hash functions) =5

1. b to compress the array $A : b \ll n$, may create errors.
2. implement independent trials to reduce errors.
3. initiate CMS to all zeros.
4. for $i = 1$ to l increment $CMS[i][h_i(x)]$.
5. running time $O(l)$.

6.return $min_{i=1} CMS[i][h_i(x)]$

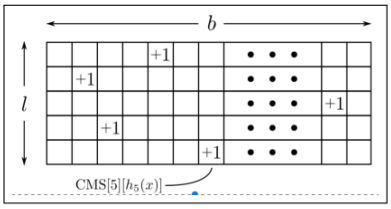

Figure 1: 1