By Hugh Zhang and Johnson Nguyen

<u>Review from Lecture 30</u>

Shingle: subsequence of length k. Given shingles, we most likely can reconstruct a sequence. A real life example would be that of a tree. Suppose we wanted to store the tree with a smaller representation, we could define the shingle to be a tree of width 3 and depth 2.

<u>New Problem: How do you find the number of distinct elements in a data stream?</u>

Application:

Given a list of credit card transactions, how many distinct numbers are there (namely, unique customers)?
What we need are the credit card numbers and a User Identification.

$a_1, a_2, a_3, \ldots, a_n$ n integers in range 1 to m (i.e. credit card #'s)

One solution:

Keep a vector 1 to m, set to 1 if the integer is seen

$$\underline{0}\ \underline{0}\ \underline{0}\ \underline{1}\ \underline{1}\ \underline{0}\ \underline{0}\ \underline{0}$$
$$\leftarrow m \rightarrow$$

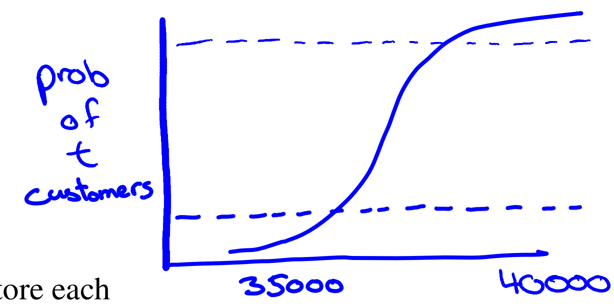
The problem with this is that it requires m bits of memory and every number must be processed.

Another Solution:

If the number of distinct elements is small:

Create a hash function:

d is the # of distinct elements



d*log(m) where log(m) is how long it takes to store each of the d distinct elements.

We may simply want to know, are there *t* distinct customers?
An approximation to this may be sufficient. Suppose we had 38, 451 – we could take the range from 35,000 to 40,000 as sufficient knowledge.

Approximation Algorithm:

if (# of distinct elements $\geq$ 2t)
  the algorithm answers yes with probability $\geq$ 0.865
if (# of distinct elements < t)
  the algorithm answers yes with probability $\leq$ 0.64

Let h: {1,2,…,m} → {1,2,…t}

set of credit card numbers

Compute $h(a_i)$ for each element in the sequence sequence
Answer yes if for any i, $h(a_i) = 1$

Proof of this algorithm:

For each i, the probability that $h(a_i) = 1$ is $\frac{1}{t}$

If there are *d* distinct elements, what is the probability none of them are hashed to 1?

$$(1-\frac{1}{t})^d$$

Aside: As d increases, $(1-\frac{1}{t})^d$ decreases.

if ($d \le t$)

the probability that the algorithm answers no is $\ge (1-\frac{1}{t})^t = \frac{1}{e} \approx 0.36$

Probability of yes $\le 1-\frac{1}{e} \approx 0.64$

if ($d > 2t$)

probability the algorithm answers no is $(1-\frac{1}{t})^d \le (1-\frac{1}{t})^{2t} = (\frac{1}{e})^2 \approx 0.135$

=> probability the algorithm answers yes $\ge 1 - 0.135 \approx 0.865$

CS 485 – Lecture 31 – April 12, 2006

By Hugh Zhang and Johnson Nguyen

Singular Value Decomposition

$$A = U \sum V^t$$
$$A^k = U \sum^k V^t$$

1) May not be able to interpret rows and columns. Possibility of negative elements.
2) A likely to be sparse, A^k likely to be dense.

$$(A) = (C)(\Sigma)(R)$$

$$\left|A - C \sum R^t\right|_F^2 \le \frac{1}{1-\varepsilon}\left|A - A^k\right| + \varepsilon\left|A\right|_F^2$$

$$\left|A\right|_F^2 = \lambda_1^2 + \lambda_2^2 + \ldots + \lambda_n^2$$