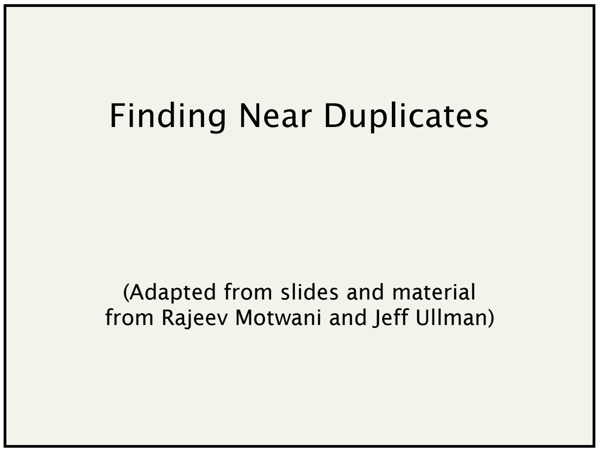
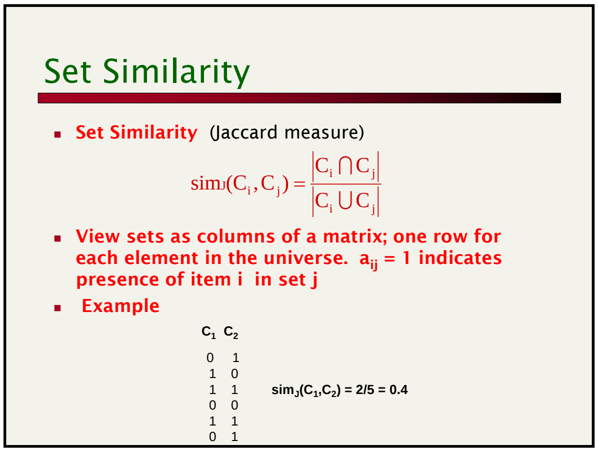


Finding Near Duplicates

(Adapted from slides and material
from Rajeev Motwani and Jeff Ullman)



Set Similarity

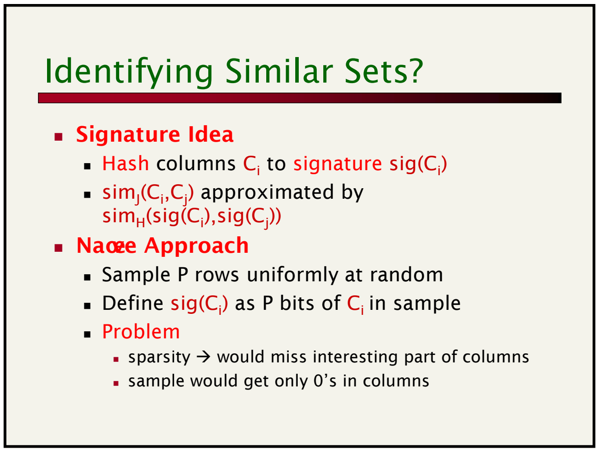
- **Set Similarity** (Jaccard measure)

$$\mathrm{sim}_J(C_i, C_j) = \frac{|C_i \cap C_j|}{|C_i \cup C_j|}$$

- **View sets as columns of a matrix; one row for each element in the universe. a_{ij} = 1 indicates presence of item i in set j**
- **Example**

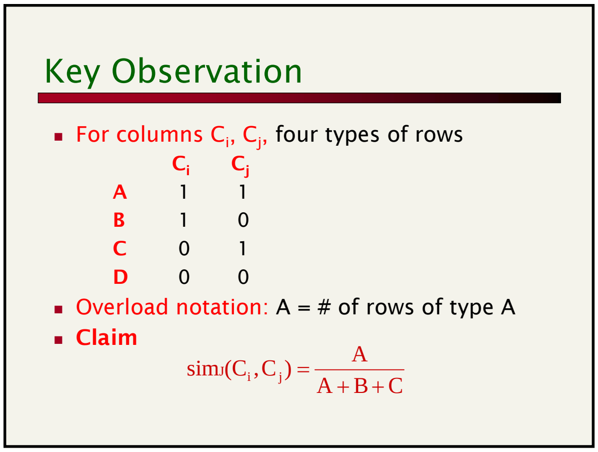
C_1	C_2
0	1
1	0
1	1
0	0
1	1
0	1

$\mathrm{sim}_J(C_1,C_2)$ = 2/5 = 0.4



Identifying Similar Sets?

- **Signature Idea**
  - Hash columns C_i to signature $\mathrm{sig}(C_i)$
  - $\mathrm{sim}_J(C_i,C_j)$ approximated by $\mathrm{sim}_H(\mathrm{sig}(C_i),\mathrm{sig}(C_j))$
- **Naœe Approach**
  - Sample P rows uniformly at random
  - Define $\mathrm{sig}(C_i)$ as P bits of C_i in sample
  - Problem
    - sparsity → would miss interesting part of columns
    - sample would get only 0's in columns



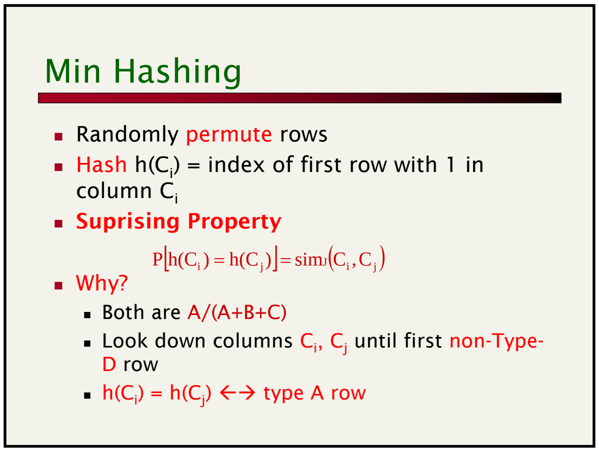
Key Observation

- For columns C_i, C_j, four types of rows

	C_i	C_j
A	1	1
B	1	0
C	0	1
D	0	0

- Overload notation: A = # of rows of type A
- **Claim**

$$\mathrm{sim}_J(C_i, C_j) = \frac{A}{A+B+C}$$

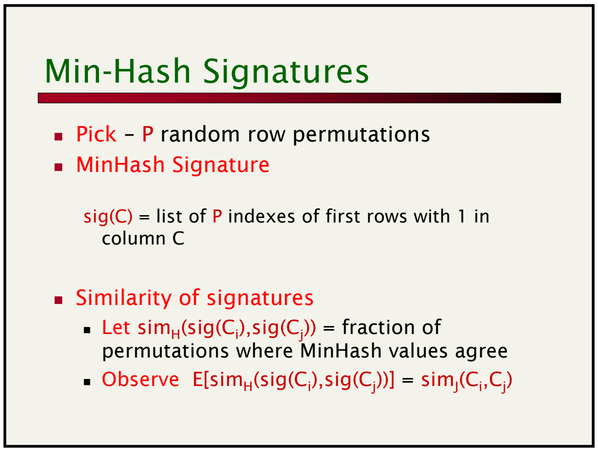


Min Hashing

- Randomly permute rows
- Hash $h(C_i)$ = index of first row with 1 in column C_i
- **Suprising Property**

$$P\left[h(C_i) = h(C_j)\right] = \mathrm{sim}_J\left(C_i, C_j\right)$$

- Why?
  - Both are A/(A+B+C)
  - Look down columns C_i, C_j until first non-Type-D row
  - $h(C_i) = h(C_j)$ ←→ type A row



Min-Hash Signatures

- Pick – P random row permutations
- MinHash Signature

sig(C) = list of P indexes of first rows with 1 in column C

- Similarity of signatures
  - Let $\mathrm{sim}_H(\mathrm{sig}(C_i),\mathrm{sig}(C_j))$ = fraction of permutations where MinHash values agree
  - Observe $E[\mathrm{sim}_H(\mathrm{sig}(C_i),\mathrm{sig}(C_j))] = \mathrm{sim}_J(C_i,C_j)$

Example

	C_1	C_2	C_3
R_1	1	0	1
R_2	0	1	1
R_3	1	0	0
R_4	1	0	1
R_5	0	1	0

Signatures

	S_1	S_2	S_3
Perm 1 = (12345)	1	2	1
Perm 2 = (54321)	4	5	4
Perm 3 = (34512)	3	5	4

Similarities

	1-2	1-3	2-3
Col-Col	0.00	0.50	0.25
Sig-Sig	0.00	0.67	0.00

Implementation Trick

- Permuting rows even once is prohibitive
- **Row Hashing**
  - Pick P hash functions h_k: {1,…,n}→{1,…,O(n)}
  - Ordering under h_k gives random row permutation
- **One-pass Implementation**
  - For each C_i and h_k, keep “slot” for min-hash value
  - Initialize all slot(C_i,h_k) to infinity
  - Scan rows in arbitrary order looking for 1’s
    - Suppose row R_j has 1 in column C_i
    - For each h_k,

Example

	C_1	C_2
R_1	1	0
R_2	0	1
R_3	1	1
R_4	1	0
R_5	0	1

h(x) = x mod 5
g(x) = 2x+1 mod 5

	C_1 slots	C_2 slots
h(1) = 1	1	-
g(1) = 3	3	-
h(2) = 2	1	2
g(2) = 0	3	0
h(3) = 3	1	2
g(3) = 2	2	0
h(4) = 4	1	2
g(4) = 4	2	0
h(5) = 0	1	0
g(5) = 1	2	0

Comparing Signatures

- **Signature Matrix S**
  - Rows = Hash Functions
  - Columns = Columns
  - Entries = Signatures
- Compute – Pair-wise similarity of signature columns
- **Problem**
  - MinHash fits column signatures in memory
  - But comparing signature-pairs takes too much time
- Technique to limit candidate pairs?
  - A-Priori does not work
  - Locality Sensitive Hashing (LSH)