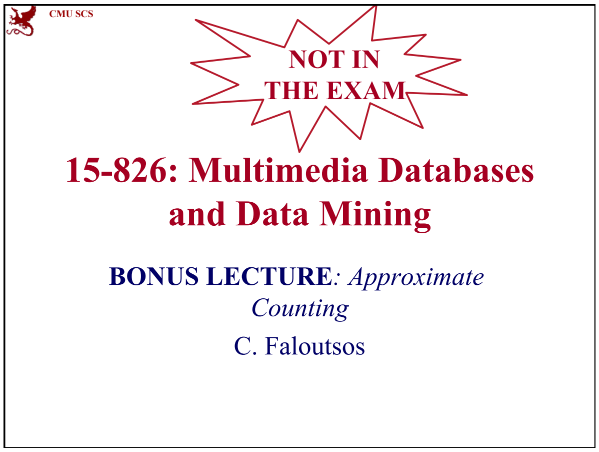
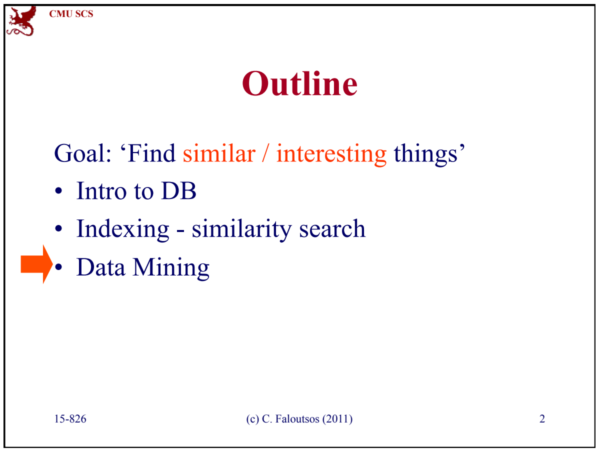

CMU SCS

NOT IN THE EXAM

15-826: Multimedia Databases and Data Mining

BONUS LECTURE: *Approximate Counting*

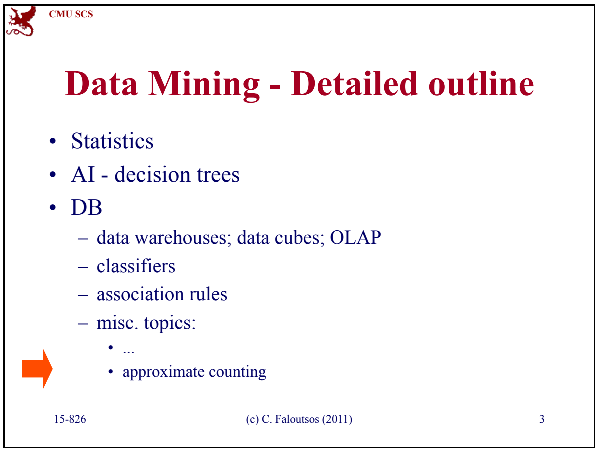
C. Faloutsos


CMU SCS

Outline

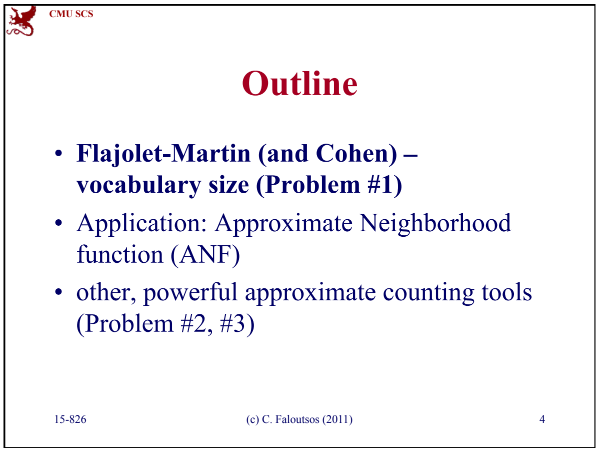
Goal: 'Find similar / interesting things'

- Intro to DB
- Indexing - similarity search
- Data Mining


CMU SCS

Data Mining - Detailed outline

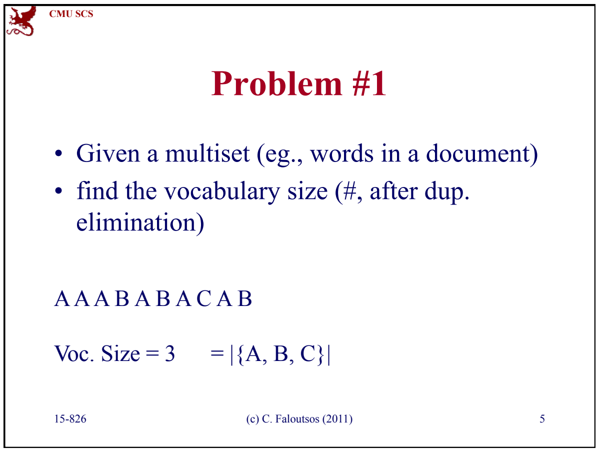
- Statistics
- AI - decision trees
- DB
  - data warehouses; data cubes; OLAP
  - classifiers
  - association rules
  - misc. topics:
    - ...
    - approximate counting



CMU SCS

Outline

- **Flajolet-Martin (and Cohen) – vocabulary size (Problem #1)**
- Application: Approximate Neighborhood function (ANF)
- other, powerful approximate counting tools (Problem #2, #3)



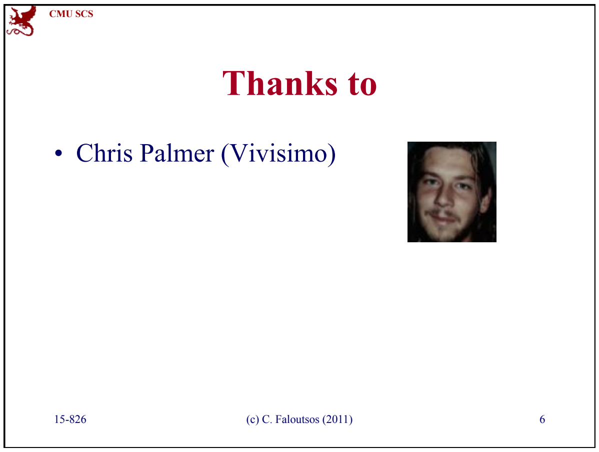
CMU SCS

Problem #1

- Given a multiset (eg., words in a document)
- find the vocabulary size (#, after dup. elimination)

A A A B A B A C A B

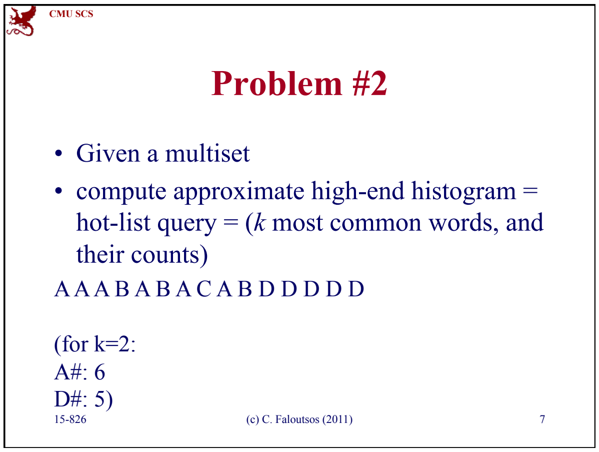
Voc. Size = 3 = |{A, B, C}|



CMU SCS

Thanks to

- Chris Palmer (Vivisimo)

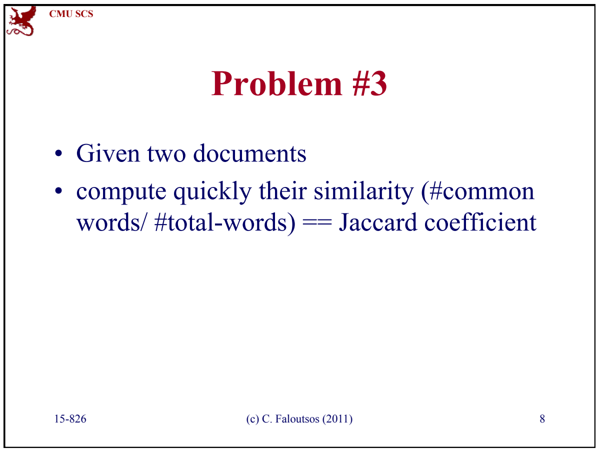


Problem #2

- Given a multiset
- compute approximate high-end histogram = hot-list query = (*k* most common words, and their counts)

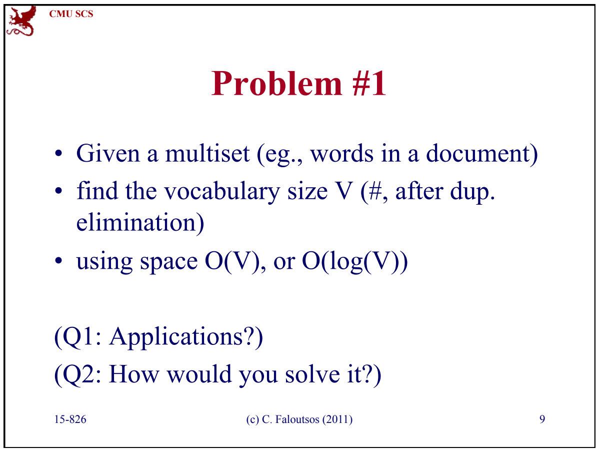
A A A B A B A C A B D D D D D

(for k=2:
A#: 6
D#: 5)



Problem #3

- Given two documents
- compute quickly their similarity (#common words/ #total-words) == Jaccard coefficient

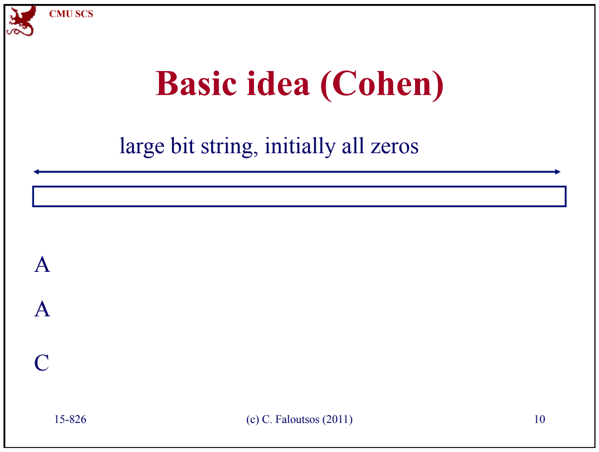


Problem #1

- Given a multiset (eg., words in a document)
- find the vocabulary size V (#, after dup. elimination)
- using space O(V), or O(log(V))

(Q1: Applications?)

(Q2: How would you solve it?)



CMU SCS

Basic idea (Cohen)

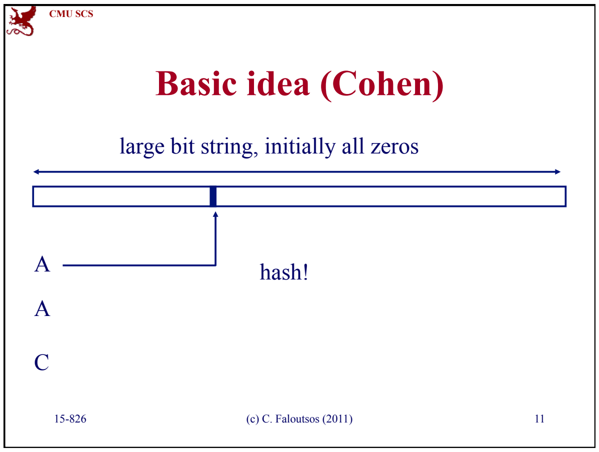
large bit string, initially all zeros

A

A

C

15-826 (c) C. Faloutsos (2011) 10



CMU SCS

Basic idea (Cohen)

large bit string, initially all zeros

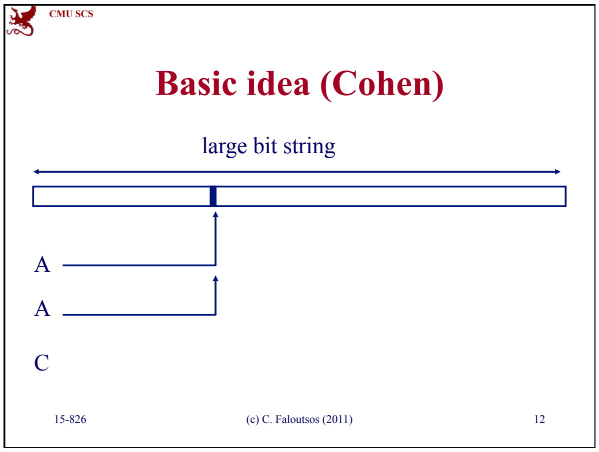
A

hash!

A

C

15-826 (c) C. Faloutsos (2011) 11



CMU SCS

Basic idea (Cohen)

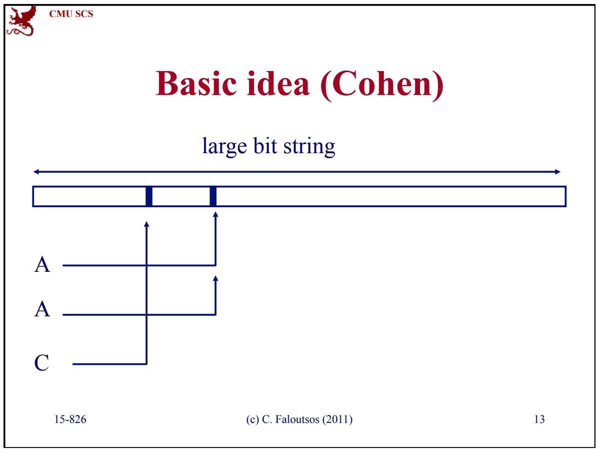
large bit string

A

A

C

15-826 (c) C. Faloutsos (2011) 12



CMU SCS

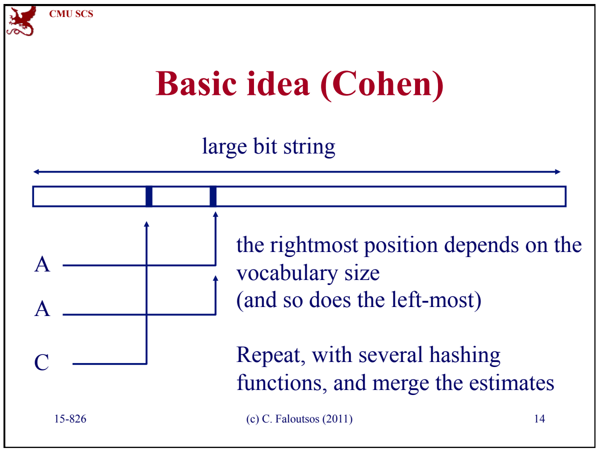
Basic idea (Cohen)

large bit string

A

A

C



CMU SCS

Basic idea (Cohen)

large bit string

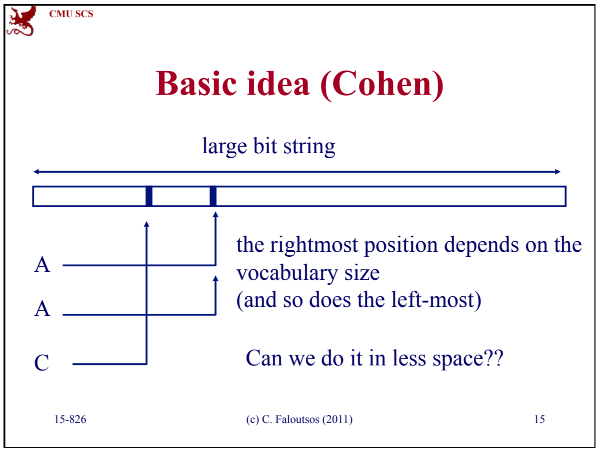
A

A

C

the rightmost position depends on the vocabulary size
(and so does the left-most)

Repeat, with several hashing functions, and merge the estimates



CMU SCS

Basic idea (Cohen)

large bit string

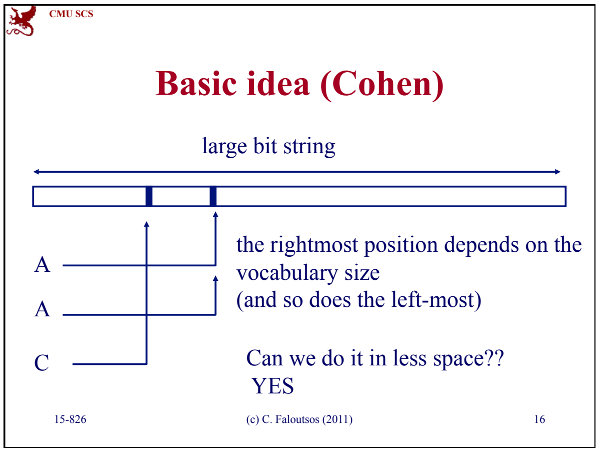
A

A

C

the rightmost position depends on the vocabulary size
(and so does the left-most)

Can we do it in less space??

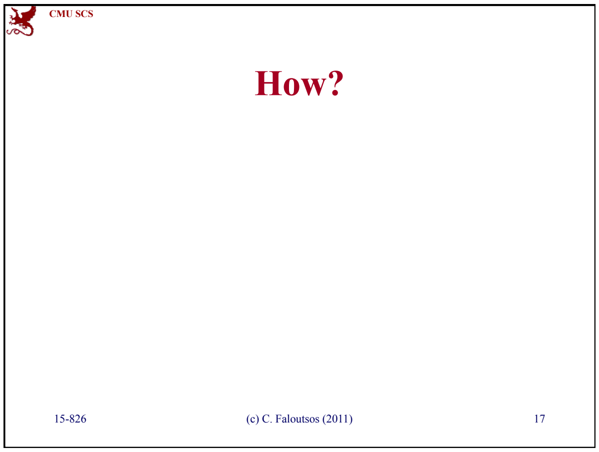


CMU SCS

Basic idea (Cohen)

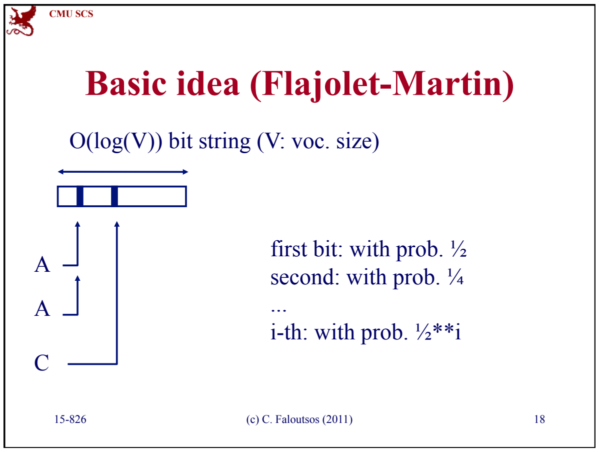
the rightmost position depends on the vocabulary size
(and so does the left-most)

Can we do it in less space??
YES



CMU SCS

How?



CMU SCS

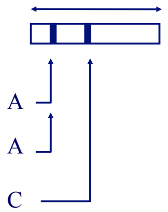
Basic idea (Flajolet-Martin)

first bit: with prob. ½
second: with prob. ¼
...
i-th: with prob. ½**i

CMU SCS

Basic idea (Flajolet-Martin)

O(log(V)) bit string (V: voc. size)



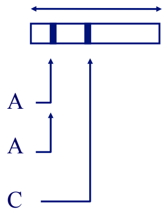
again, the rightmost bit
'reveals' the vocabulary size

15-826 (c) C. Faloutsos (2011) 19

CMU SCS

Basic idea (Flajolet-Martin)

O(log(V)) bit string (V: voc. size)



again, the rightmost bit
'reveals' the vocabulary size

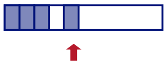
Eg.: V=4, will probably set
the 2nd bit, etc

15-826 (c) C. Faloutsos (2011) 20

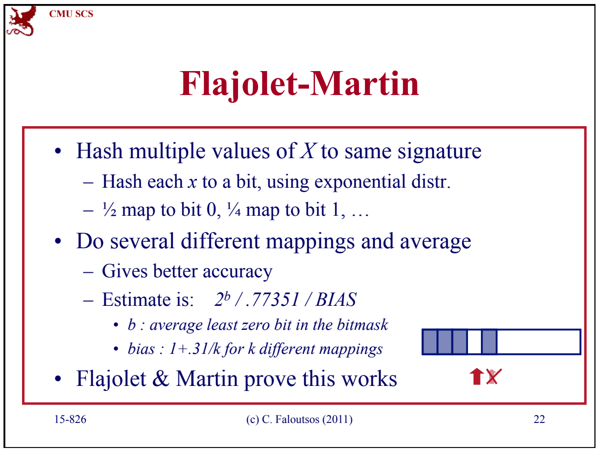
CMU SCS

Flajolet-Martin

- Hash multiple values of *X* to same signature
  - Hash each *x* to a bit, using exponential distr.
  - ½ map to bit 0, ¼ map to bit 1, …
- Do several different mappings and average
  - Gives better accuracy
  - Estimate is: 2^b */ .77351 / BIAS*
    - *b ~ rightmost '1', and actually:*



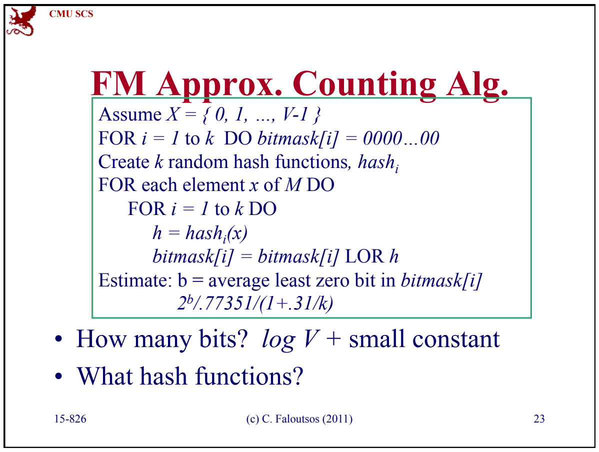
15-826 (c) C. Faloutsos (2011) 21



CMU SCS

Flajolet-Martin

- Hash multiple values of *X* to same signature
  - Hash each *x* to a bit, using exponential distr.
  - ½ map to bit 0, ¼ map to bit 1, …
- Do several different mappings and average
  - Gives better accuracy
  - Estimate is: $2^b / .77351 / BIAS$
    - *b : average least zero bit in the bitmask*
    - *bias : 1+.31/k for k different mappings*
- Flajolet & Martin prove this works

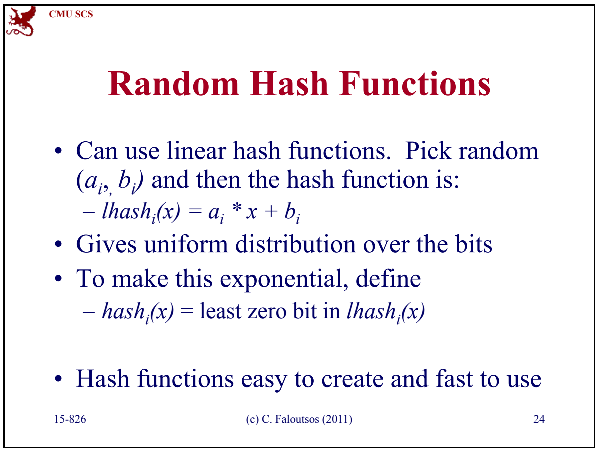


CMU SCS

FM Approx. Counting Alg.

```
Assume $X = \{ 0, 1, ..., V-1 \}$
FOR $i = 1$ to $k$ DO $bitmask[i] = 0000...00$
Create $k$ random hash functions, $hash_i$
FOR each element $x$ of $M$ DO
    FOR $i = 1$ to $k$ DO
        $h = hash_i(x)$
        $bitmask[i] = bitmask[i]$ LOR $h$
Estimate: b = average least zero bit in $bitmask[i]$
        $2^b/.77351/(1+.31/k)$
```

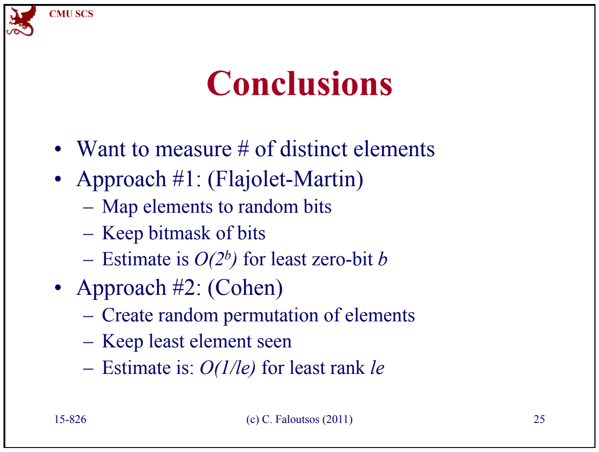
- How many bits? *log V* + small constant
- What hash functions?



CMU SCS

Random Hash Functions

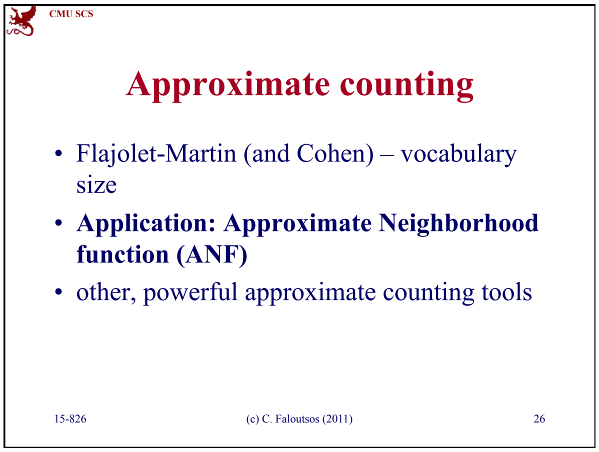
- Can use linear hash functions. Pick random (a_i, b_i) and then the hash function is:
  - $lhash_i(x) = a_i * x + b_i$
- Gives uniform distribution over the bits
- To make this exponential, define
  - $hash_i(x)$ = least zero bit in $lhash_i(x)$
- Hash functions easy to create and fast to use



CMU SCS

Conclusions

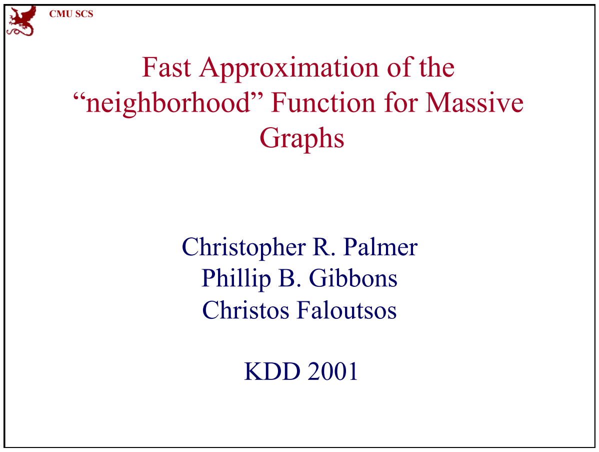
- Want to measure # of distinct elements
- Approach #1: (Flajolet-Martin)
  - Map elements to random bits
  - Keep bitmask of bits
  - Estimate is $O(2^b)$ for least zero-bit *b*
- Approach #2: (Cohen)
  - Create random permutation of elements
  - Keep least element seen
  - Estimate is: $O(1/le)$ for least rank *le*



CMU SCS

Approximate counting

- Flajolet-Martin (and Cohen) – vocabulary size
- **Application: Approximate Neighborhood function (ANF)**
- other, powerful approximate counting tools

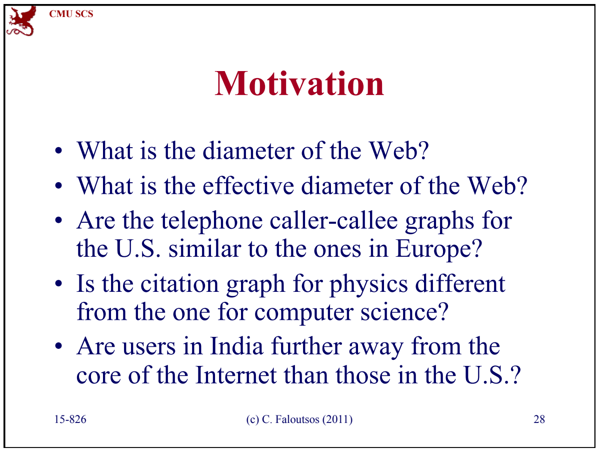


CMU SCS

Fast Approximation of the “neighborhood” Function for Massive Graphs

Christopher R. Palmer
Phillip B. Gibbons
Christos Faloutsos

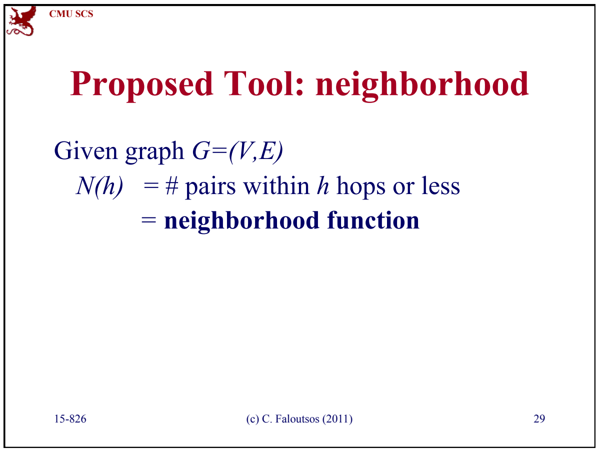
KDD 2001



CMU SCS

Motivation

- What is the diameter of the Web?
- What is the effective diameter of the Web?
- Are the telephone caller-callee graphs for the U.S. similar to the ones in Europe?
- Is the citation graph for physics different from the one for computer science?
- Are users in India further away from the core of the Internet than those in the U.S.?



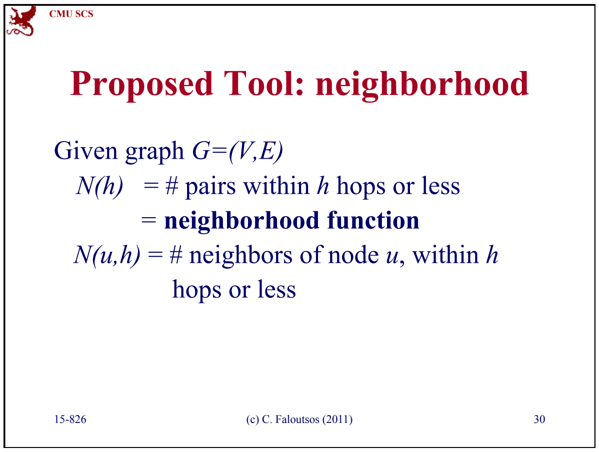
CMU SCS

Proposed Tool: neighborhood

Given graph $G=(V,E)$

$N(h)$ = # pairs within h hops or less

= **neighborhood function**



CMU SCS

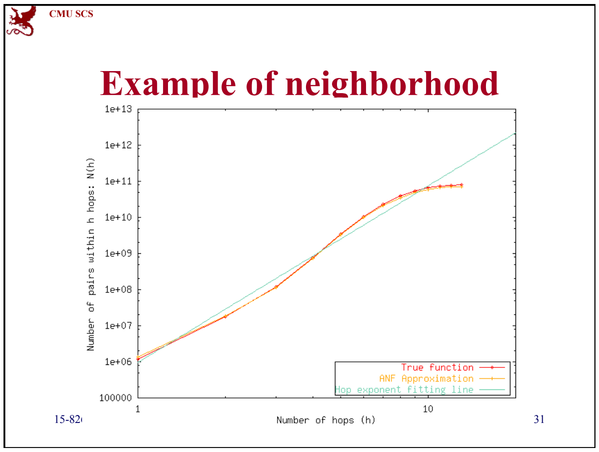
Proposed Tool: neighborhood

Given graph $G=(V,E)$

$N(h)$ = # pairs within h hops or less

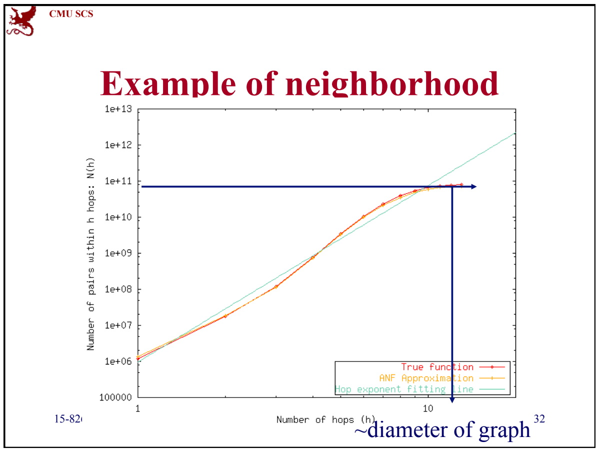
= **neighborhood function**

$N(u,h)$ = # neighbors of node u, within h hops or less



CMU SCS

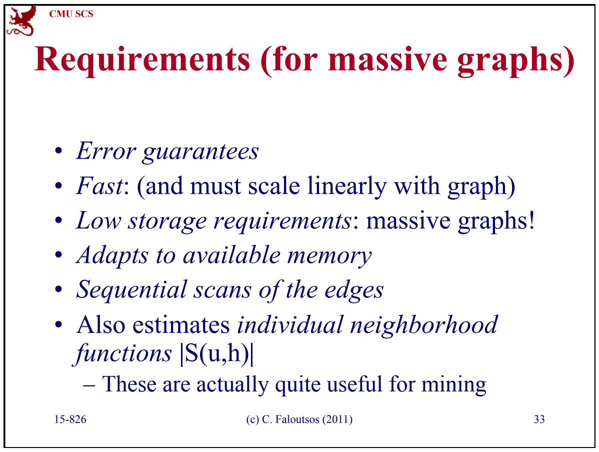
Example of neighborhood



CMU SCS

Example of neighborhood

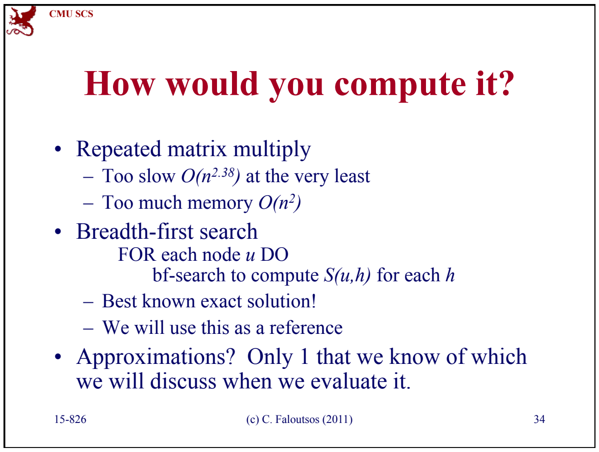
~diameter of graph



CMU SCS

Requirements (for massive graphs)

- *Error guarantees*
- *Fast*: (and must scale linearly with graph)
- *Low storage requirements*: massive graphs!
- *Adapts to available memory*
- *Sequential scans of the edges*
- Also estimates *individual neighborhood functions* |S(u,h)|
  - These are actually quite useful for mining



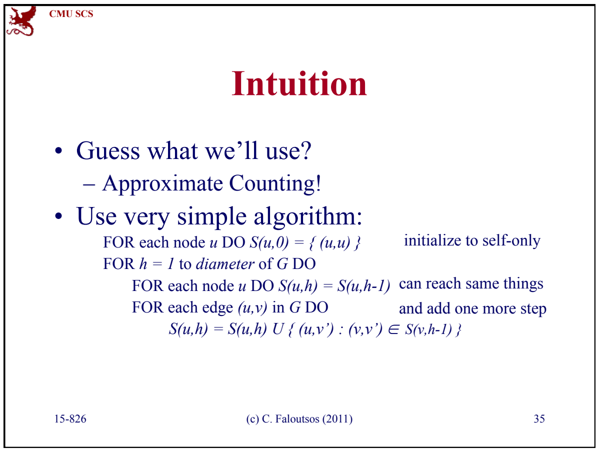
CMU SCS

How would you compute it?

- Repeated matrix multiply
  - Too slow $O(n^{2.38})$ at the very least
  - Too much memory $O(n^2)$
- Breadth-first search

```
FOR each node $u$ DO
    bf-search to compute $S(u,h)$ for each $h$
```

  - Best known exact solution!
  - We will use this as a reference
- Approximations? Only 1 that we know of which we will discuss when we evaluate it.

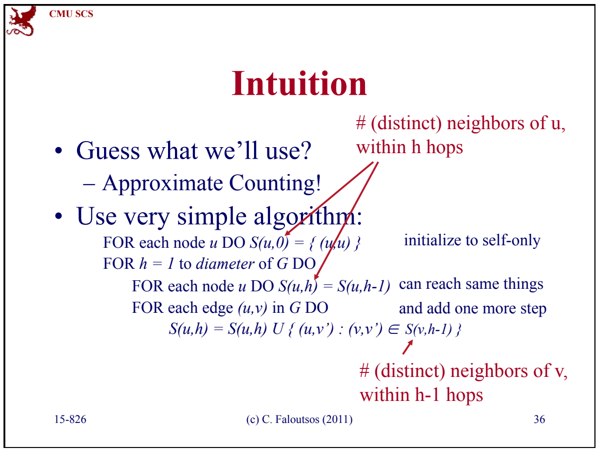


CMU SCS

Intuition

- Guess what we'll use?
  - Approximate Counting!
- Use very simple algorithm:

```
FOR each node $u$ DO $S(u,0) = \{ (u,u) \}$          initialize to self-only
FOR $h = 1$ to $diameter$ of $G$ DO
    FOR each node $u$ DO $S(u,h) = S(u,h-1)$         can reach same things
    FOR each edge $(u,v)$ in $G$ DO                  and add one more step
        $S(u,h) = S(u,h) \cup \{ (u,v') : (v,v') \in S(v,h-1) \}$
```



CMU SCS

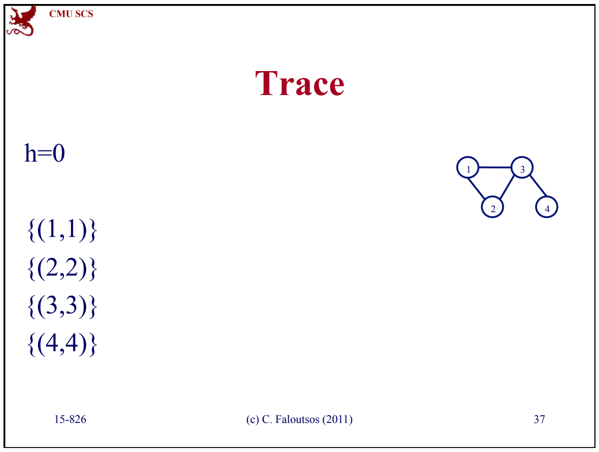
Intuition

# (distinct) neighbors of u, within h hops

- Guess what we'll use?
  - Approximate Counting!
- Use very simple algorithm:

```
FOR each node $u$ DO $S(u,0) = \{ (u,u) \}$          initialize to self-only
FOR $h = 1$ to $diameter$ of $G$ DO
    FOR each node $u$ DO $S(u,h) = S(u,h-1)$         can reach same things
    FOR each edge $(u,v)$ in $G$ DO                  and add one more step
        $S(u,h) = S(u,h) \cup \{ (u,v') : (v,v') \in S(v,h-1) \}$
```

# (distinct) neighbors of v, within h-1 hops



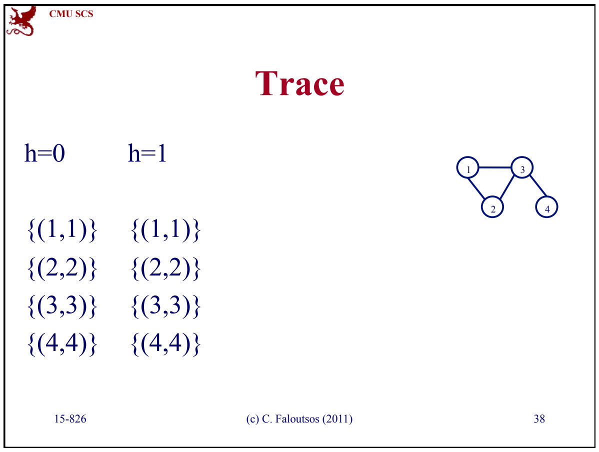
CMU SCS

Trace

h=0

{(1,1)}
{(2,2)}
{(3,3)}
{(4,4)}

15-826 (c) C. Faloutsos (2011) 37

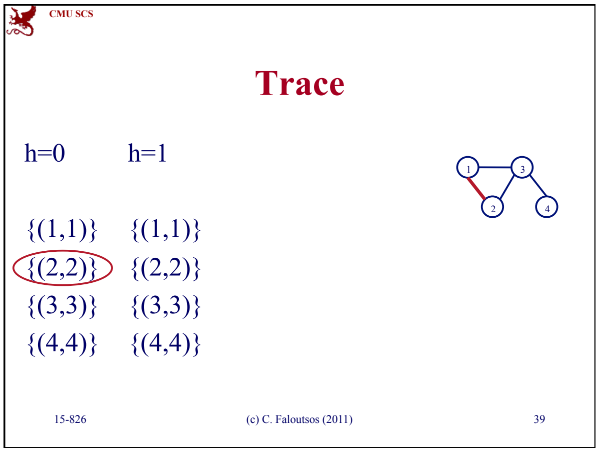


CMU SCS

Trace

h=0	h=1
{(1,1)}	{(1,1)}
{(2,2)}	{(2,2)}
{(3,3)}	{(3,3)}
{(4,4)}	{(4,4)}

15-826 (c) C. Faloutsos (2011) 38

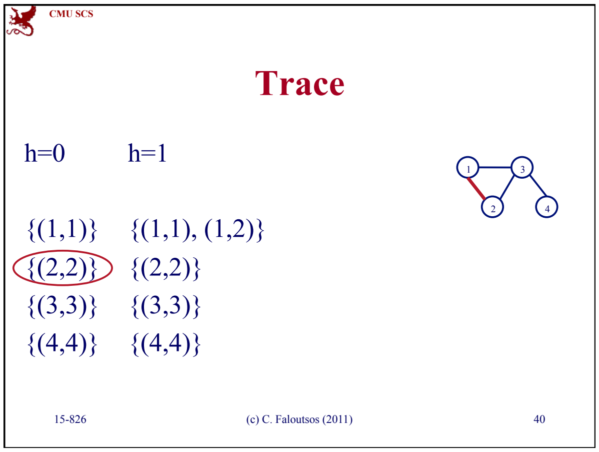


CMU SCS

Trace

h=0	h=1
{(1,1)}	{(1,1)}
{(2,2)}	{(2,2)}
{(3,3)}	{(3,3)}
{(4,4)}	{(4,4)}

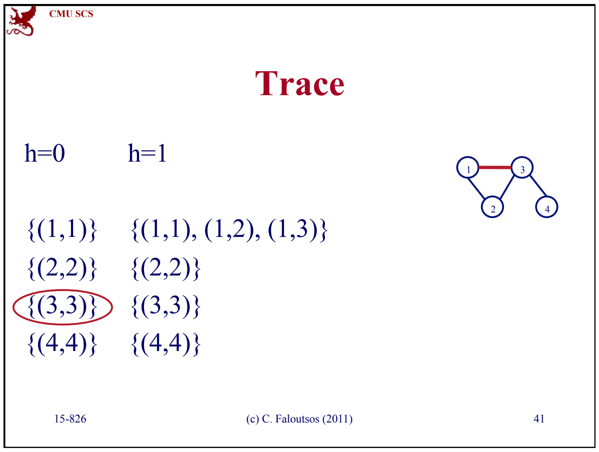
15-826 (c) C. Faloutsos (2011) 39



CMU SCS

Trace

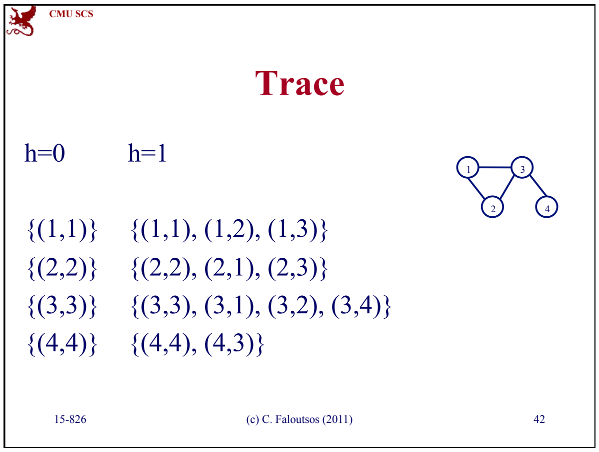
h=0	h=1
{(1,1)}	{(1,1), (1,2)}
{(2,2)}	{(2,2)}
{(3,3)}	{(3,3)}
{(4,4)}	{(4,4)}



CMU SCS

Trace

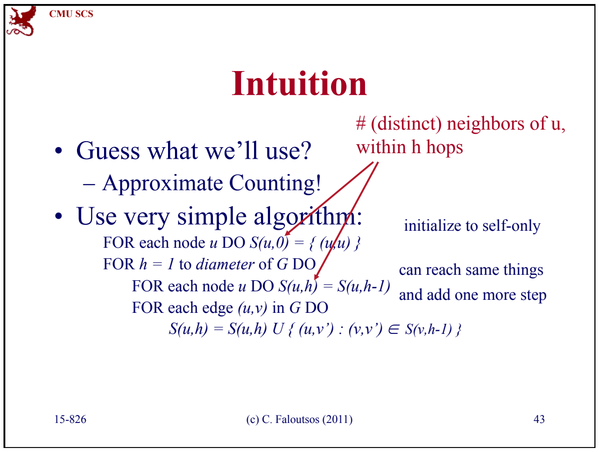
h=0	h=1
{(1,1)}	{(1,1), (1,2), (1,3)}
{(2,2)}	{(2,2)}
{(3,3)}	{(3,3)}
{(4,4)}	{(4,4)}



CMU SCS

Trace

h=0	h=1
{(1,1)}	{(1,1), (1,2), (1,3)}
{(2,2)}	{(2,2), (2,1), (2,3)}
{(3,3)}	{(3,3), (3,1), (3,2), (3,4)}
{(4,4)}	{(4,4), (4,3)}



Intuition

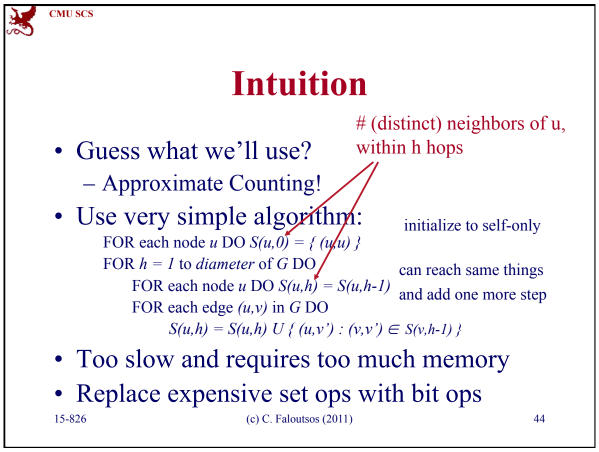
# (distinct) neighbors of u, within h hops

- Guess what we'll use?
  - Approximate Counting!
- Use very simple algorithm:

```
FOR each node u DO $S(u,0) = \{ (u,u) \}$
FOR $h = 1$ to diameter of $G$ DO
    FOR each node u DO $S(u,h) = S(u,h-1)$
    FOR each edge $(u,v)$ in $G$ DO
        $S(u,h) = S(u,h) \cup \{ (u,v') : (v,v') \in S(v,h-1) \}$
```

initialize to self-only

can reach same things and add one more step



Intuition

# (distinct) neighbors of u, within h hops

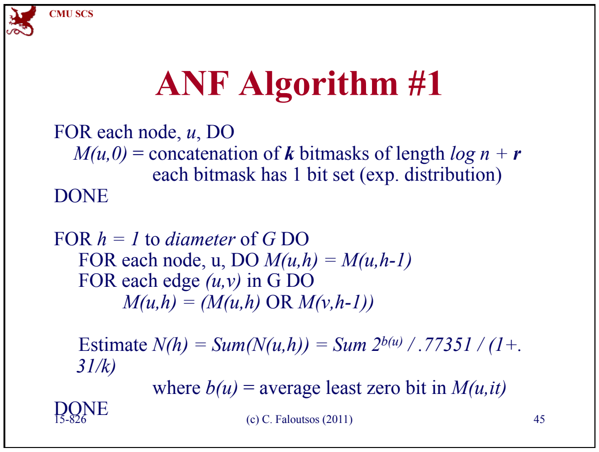
- Guess what we'll use?
  - Approximate Counting!
- Use very simple algorithm:

```
FOR each node u DO $S(u,0) = \{ (u,u) \}$
FOR $h = 1$ to diameter of $G$ DO
    FOR each node u DO $S(u,h) = S(u,h-1)$
    FOR each edge $(u,v)$ in $G$ DO
        $S(u,h) = S(u,h) \cup \{ (u,v') : (v,v') \in S(v,h-1) \}$
```

initialize to self-only

can reach same things and add one more step

- Too slow and requires too much memory
- Replace expensive set ops with bit ops

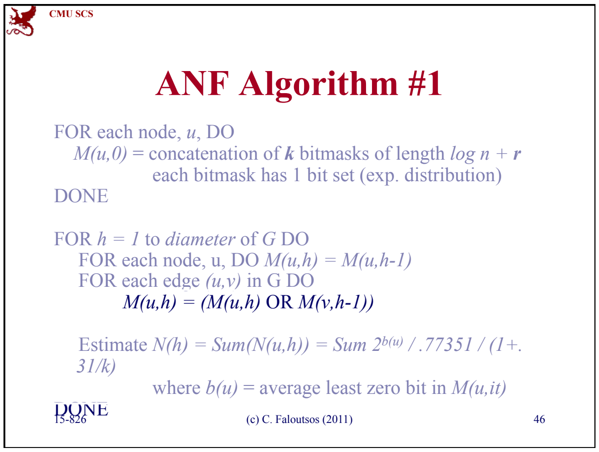


ANF Algorithm #1

```
FOR each node, $u$, DO
   $M(u,0)$ = concatenation of $\boldsymbol{k}$ bitmasks of length $\log n + \boldsymbol{r}$
             each bitmask has 1 bit set (exp. distribution)
DONE

FOR $h = 1$ to diameter of $G$ DO
    FOR each node, u, DO $M(u,h) = M(u,h-1)$
    FOR each edge $(u,v)$ in G DO
         $M(u,h) = (M(u,h)$ OR $M(v,h-1))$

    Estimate $N(h) = Sum(N(u,h)) = Sum\ 2^{b(u)} / .77351 / (1+.31/k)$
             where $b(u)$ = average least zero bit in $M(u,it)$
DONE
```



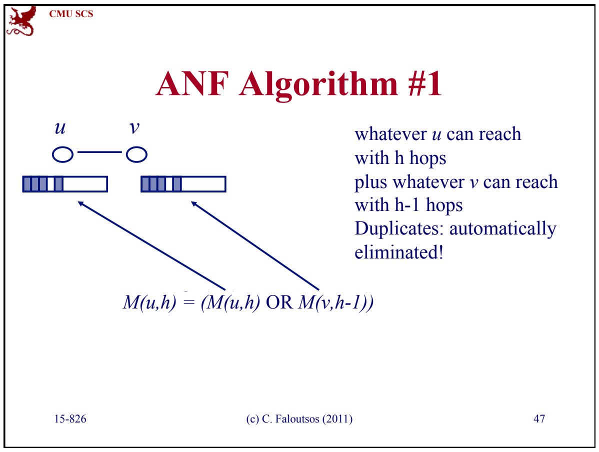
CMU SCS

ANF Algorithm #1

```
FOR each node, $u$, DO
   $M(u,0)$ = concatenation of $\boldsymbol{k}$ bitmasks of length $log\ n + \boldsymbol{r}$
            each bitmask has 1 bit set (exp. distribution)
DONE

FOR $h = 1$ to $diameter$ of $G$ DO
    FOR each node, u, DO $M(u,h) = M(u,h\text{-}1)$
    FOR each edge $(u,v)$ in G DO
          $M(u,h) = (M(u,h)$ OR $M(v,h\text{-}1))$

    Estimate $N(h) = Sum(N(u,h)) = Sum\ 2^{b(u)}\ /\ .77351\ /\ (1+.31/k)$
            where $b(u)$ = average least zero bit in $M(u,it)$
DONE
```

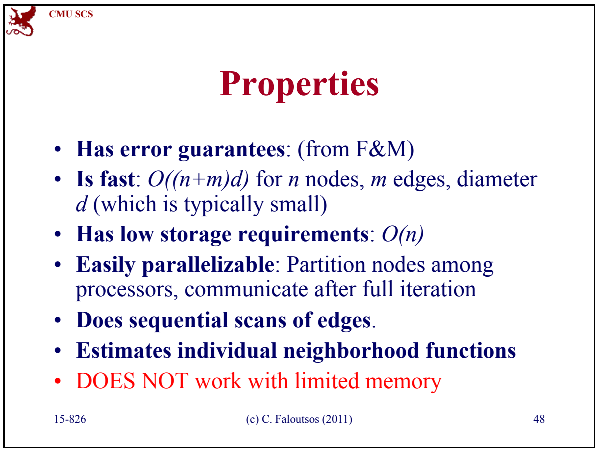


CMU SCS

ANF Algorithm #1

whatever u can reach with h hops plus whatever v can reach with h-1 hops Duplicates: automatically eliminated!

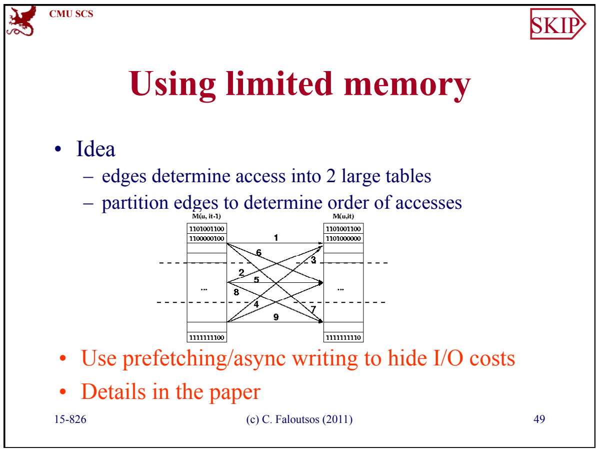
$M(u,h) = (M(u,h)$ OR $M(v,h\text{-}1))$



CMU SCS

Properties

- **Has error guarantees**: (from F&M)
- **Is fast**: $O((n+m)d)$ for n nodes, m edges, diameter d (which is typically small)
- **Has low storage requirements**: $O(n)$
- **Easily parallelizable**: Partition nodes among processors, communicate after full iteration
- **Does sequential scans of edges**.
- **Estimates individual neighborhood functions**
- DOES NOT work with limited memory



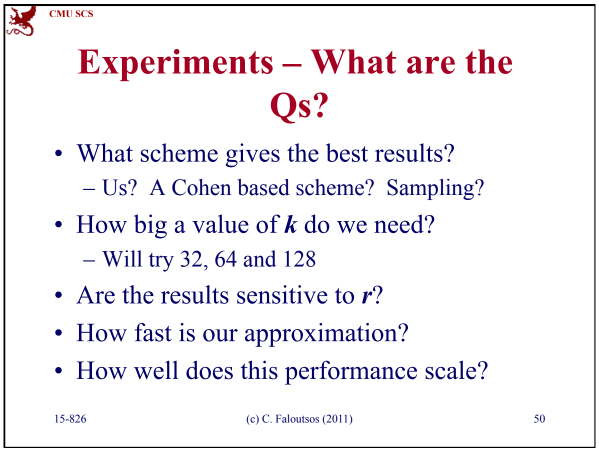
CMU SCS

SKIP

Using limited memory

- Idea
  - edges determine access into 2 large tables
  - partition edges to determine order of accesses

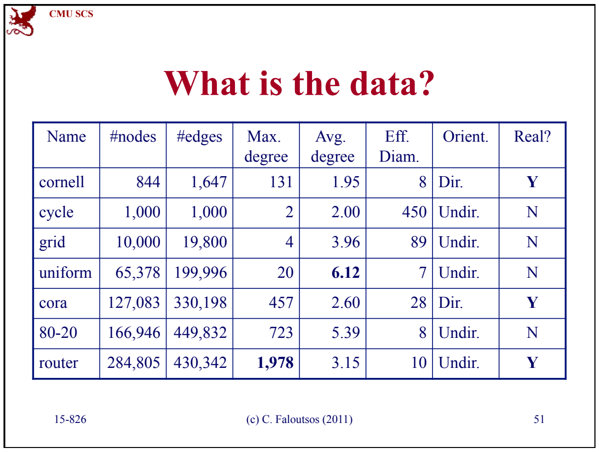
- Use prefetching/async writing to hide I/O costs
- Details in the paper



CMU SCS

Experiments – What are the Qs?

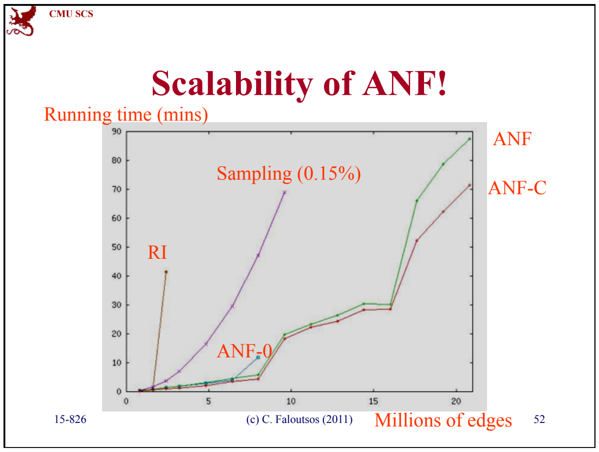
- What scheme gives the best results?
  - Us? A Cohen based scheme? Sampling?
- How big a value of ***k*** do we need?
  - Will try 32, 64 and 128
- Are the results sensitive to ***r***?
- How fast is our approximation?
- How well does this performance scale?



CMU SCS

What is the data?

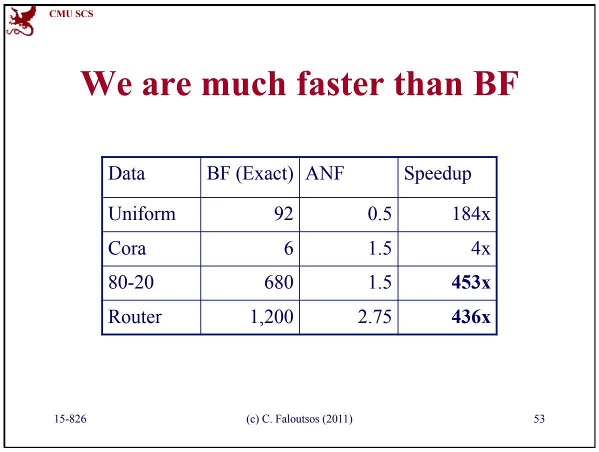
Name	#nodes	#edges	Max. degree	Avg. degree	Eff. Diam.	Orient.	Real?
cornell	844	1,647	131	1.95	8	Dir.	**Y**
cycle	1,000	1,000	2	2.00	450	Undir.	N
grid	10,000	19,800	4	3.96	89	Undir.	N
uniform	65,378	199,996	20	**6.12**	7	Undir.	N
cora	127,083	330,198	457	2.60	28	Dir.	**Y**
80-20	166,946	449,832	723	5.39	8	Undir.	N
router	284,805	430,342	**1,978**	3.15	10	Undir.	**Y**



CMU SCS

Scalability of ANF!

15-826 (c) C. Faloutsos (2011) 52

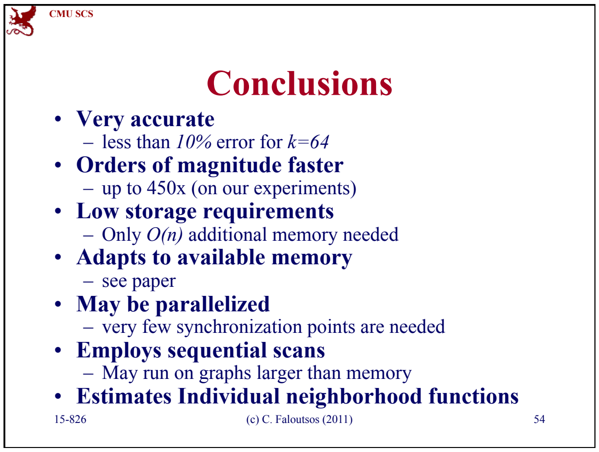


CMU SCS

We are much faster than BF

Data	BF (Exact)	ANF	Speedup
Uniform	92	0.5	184x
Cora	6	1.5	4x
80-20	680	1.5	**453x**
Router	1,200	2.75	**436x**

15-826 (c) C. Faloutsos (2011) 53

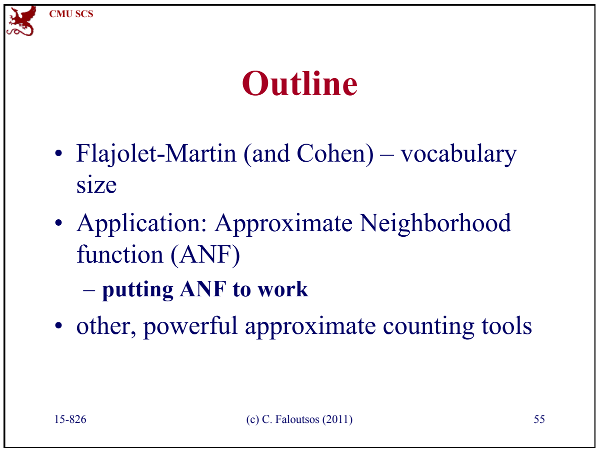


CMU SCS

Conclusions

- **Very accurate**
  - less than *10%* error for *k=64*
- **Orders of magnitude faster**
  - up to 450x (on our experiments)
- **Low storage requirements**
  - Only *O(n)* additional memory needed
- **Adapts to available memory**
  - see paper
- **May be parallelized**
  - very few synchronization points are needed
- **Employs sequential scans**
  - May run on graphs larger than memory
- **Estimates Individual neighborhood functions**

15-826 (c) C. Faloutsos (2011) 54

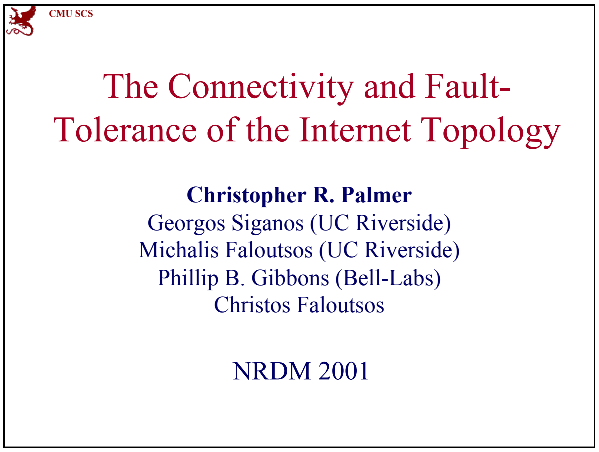


CMU SCS

Outline

- Flajolet-Martin (and Cohen) – vocabulary size
- Application: Approximate Neighborhood function (ANF)
  - **putting ANF to work**
- other, powerful approximate counting tools

15-826 (c) C. Faloutsos (2011) 55

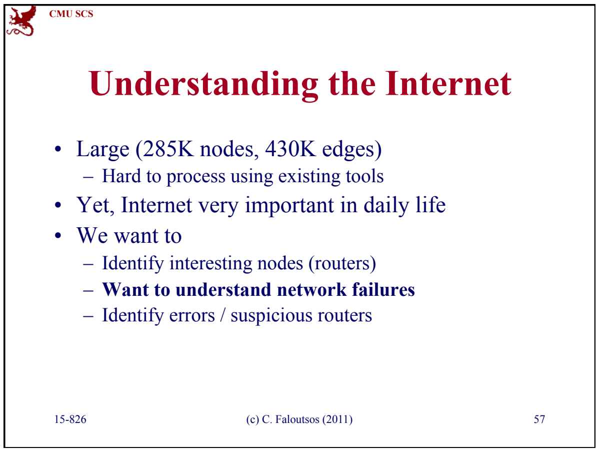


CMU SCS

The Connectivity and Fault-Tolerance of the Internet Topology

Christopher R. Palmer
Georgos Siganos (UC Riverside)
Michalis Faloutsos (UC Riverside)
Phillip B. Gibbons (Bell-Labs)
Christos Faloutsos

NRDM 2001

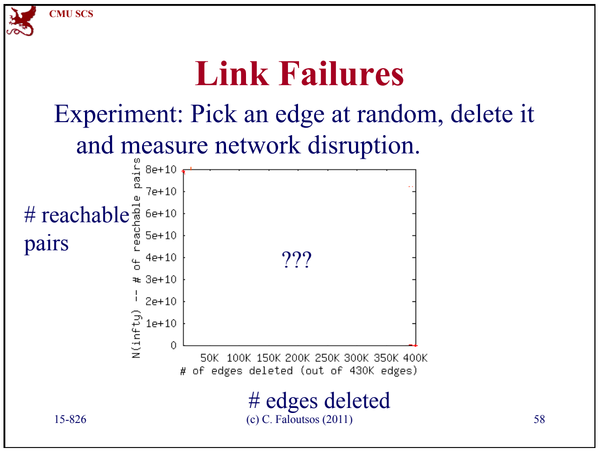


CMU SCS

Understanding the Internet

- Large (285K nodes, 430K edges)
  - Hard to process using existing tools
- Yet, Internet very important in daily life
- We want to
  - Identify interesting nodes (routers)
  - **Want to understand network failures**
  - Identify errors / suspicious routers

15-826 (c) C. Faloutsos (2011) 57



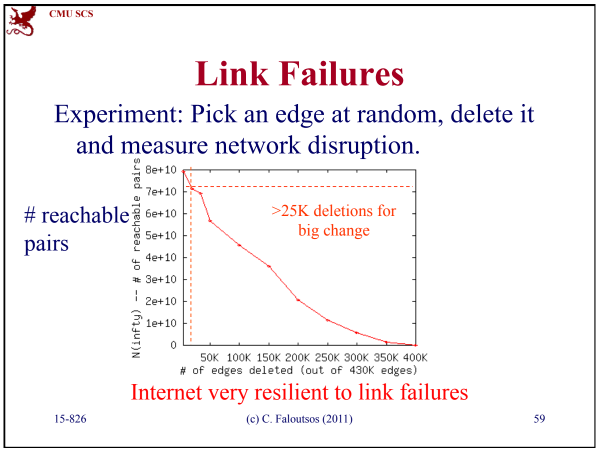
CMU SCS

Link Failures

Experiment: Pick an edge at random, delete it and measure network disruption.

# reachable pairs
N(infty) -- # of reachable pairs
8e+10
7e+10
6e+10
5e+10
4e+10
3e+10
2e+10
1e+10
0
???
50K 100K 150K 200K 250K 300K 350K 400K
# of edges deleted (out of 430K edges)
# edges deleted

15-826 (c) C. Faloutsos (2011) 58



CMU SCS

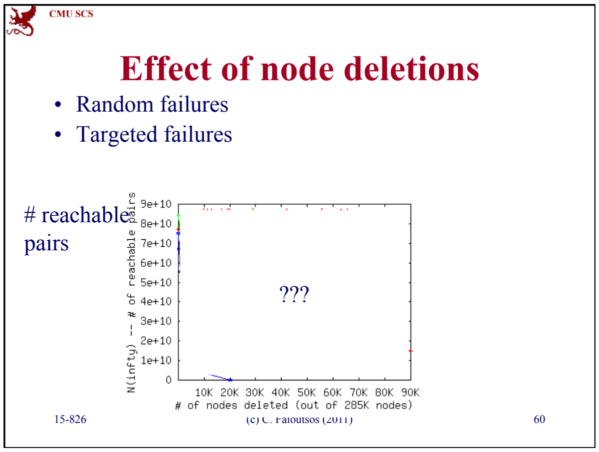
Link Failures

Experiment: Pick an edge at random, delete it and measure network disruption.

# reachable pairs
N(infty) -- # of reachable pairs
8e+10
7e+10
6e+10
5e+10
4e+10
3e+10
2e+10
1e+10
0
>25K deletions for big change
50K 100K 150K 200K 250K 300K 350K 400K
# of edges deleted (out of 430K edges)

Internet very resilient to link failures

15-826 (c) C. Faloutsos (2011) 59



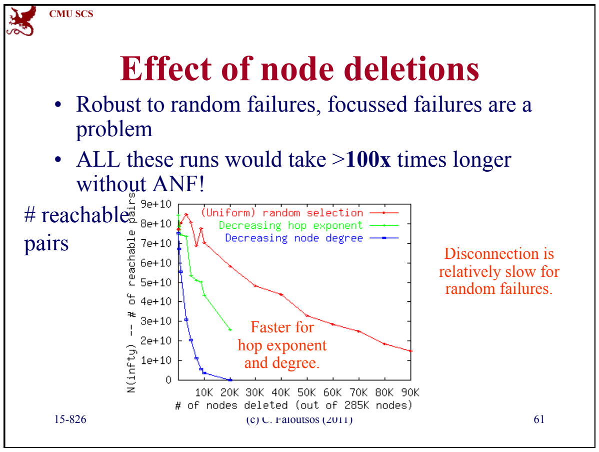
CMU SCS

Effect of node deletions

- Random failures
- Targeted failures

# reachable pairs
N(infty) -- # of reachable pairs
9e+10
8e+10
7e+10
6e+10
5e+10
4e+10
3e+10
2e+10
1e+10
0
???
10K 20K 30K 40K 50K 60K 70K 80K 90K
# of nodes deleted (out of 285K nodes)

15-826 (c) C. Faloutsos (2011) 60



CMU SCS

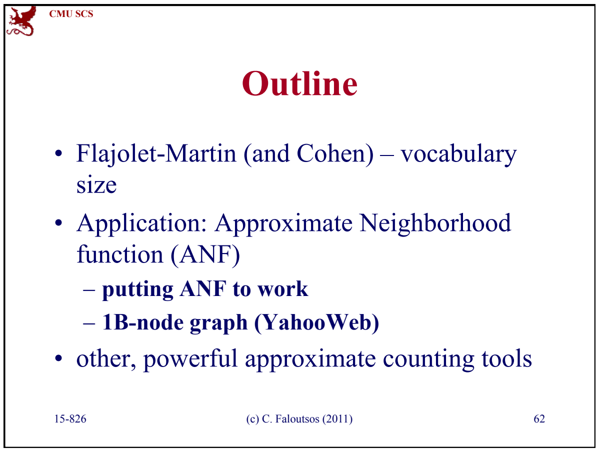
Effect of node deletions

- Robust to random failures, focussed failures are a problem
- ALL these runs would take >**100x** times longer without ANF!

# reachable pairs

Disconnection is relatively slow for random failures.

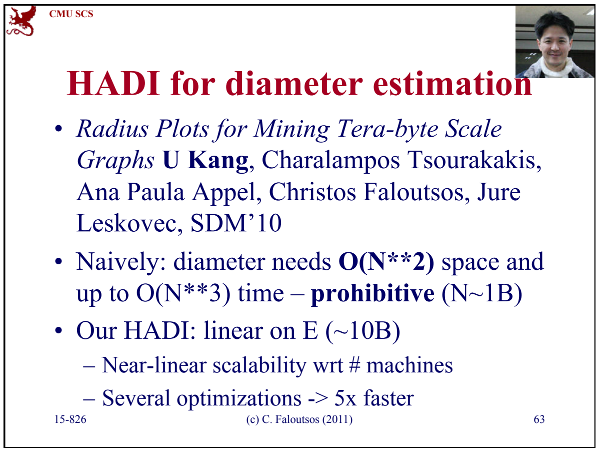
Faster for hop exponent and degree.



CMU SCS

Outline

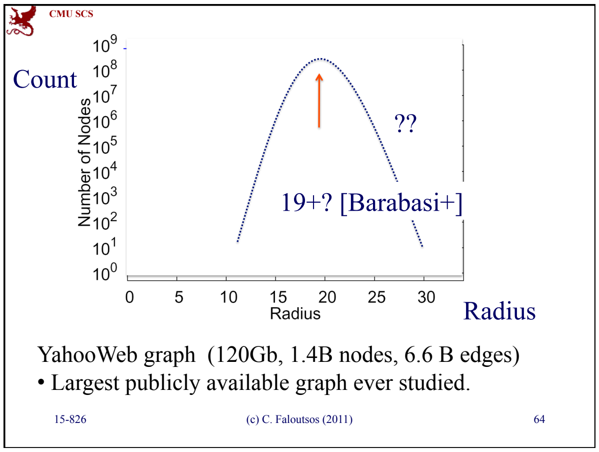
- Flajolet-Martin (and Cohen) – vocabulary size
- Application: Approximate Neighborhood function (ANF)
  - **putting ANF to work**
  - **1B-node graph (YahooWeb)**
- other, powerful approximate counting tools



CMU SCS

HADI for diameter estimation

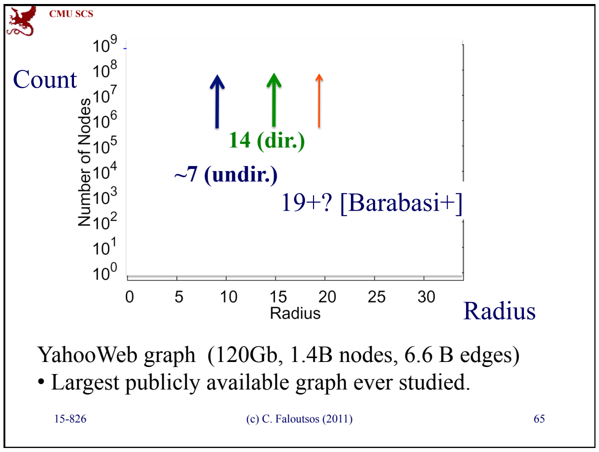
- *Radius Plots for Mining Tera-byte Scale Graphs* **U Kang**, Charalampos Tsourakakis, Ana Paula Appel, Christos Faloutsos, Jure Leskovec, SDM’10
- Naively: diameter needs **O(N**2)** space and up to O(N**3) time – **prohibitive** (N~1B)
- Our HADI: linear on E (~10B)
  - Near-linear scalability wrt # machines
  - Several optimizations -> 5x faster



YahooWeb graph (120Gb, 1.4B nodes, 6.6 B edges)

- Largest publicly available graph ever studied.

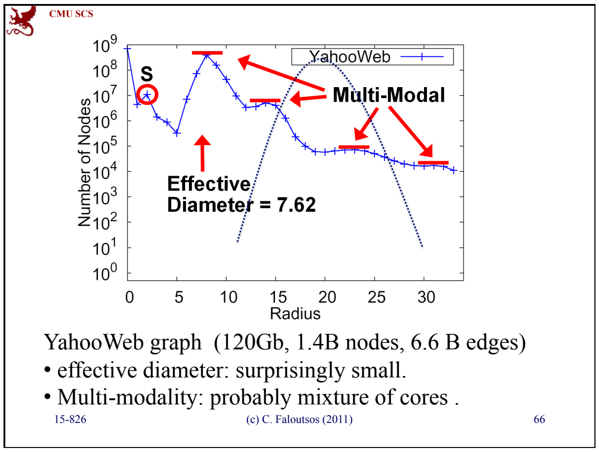
15-826 (c) C. Faloutsos (2011) 64



YahooWeb graph (120Gb, 1.4B nodes, 6.6 B edges)

- Largest publicly available graph ever studied.

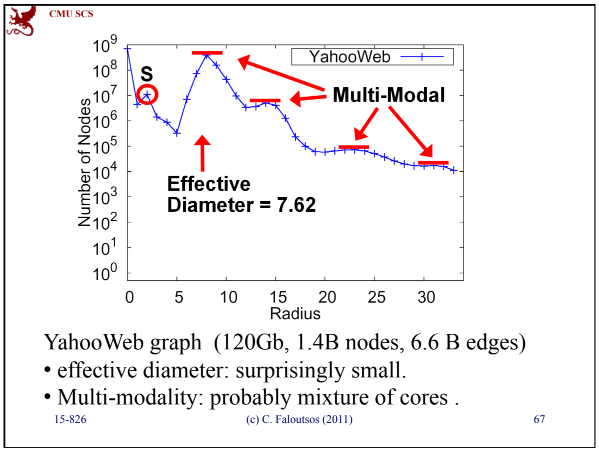
15-826 (c) C. Faloutsos (2011) 65



YahooWeb graph (120Gb, 1.4B nodes, 6.6 B edges)

- effective diameter: surprisingly small.
- Multi-modality: probably mixture of cores .

15-826 (c) C. Faloutsos (2011) 66

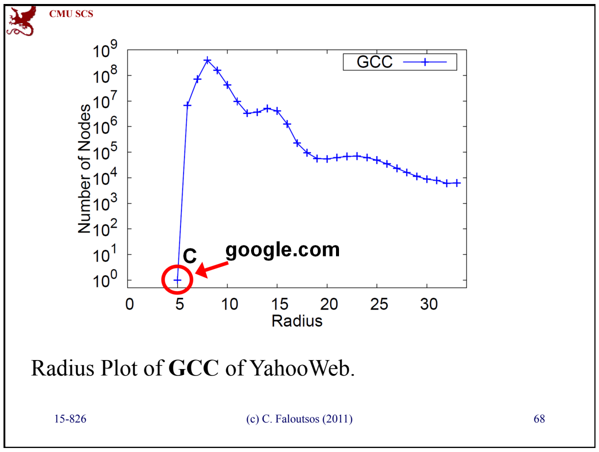


CMU SCS

YahooWeb graph (120Gb, 1.4B nodes, 6.6 B edges)

- effective diameter: surprisingly small.
- Multi-modality: probably mixture of cores .

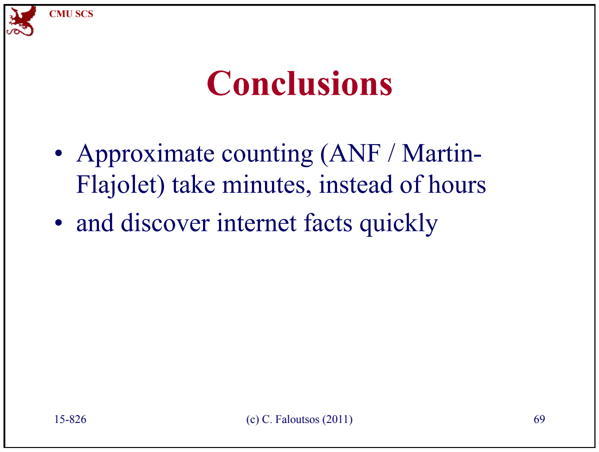
15-826 (c) C. Faloutsos (2011) 67



CMU SCS

Radius Plot of **GCC** of YahooWeb.

15-826 (c) C. Faloutsos (2011) 68

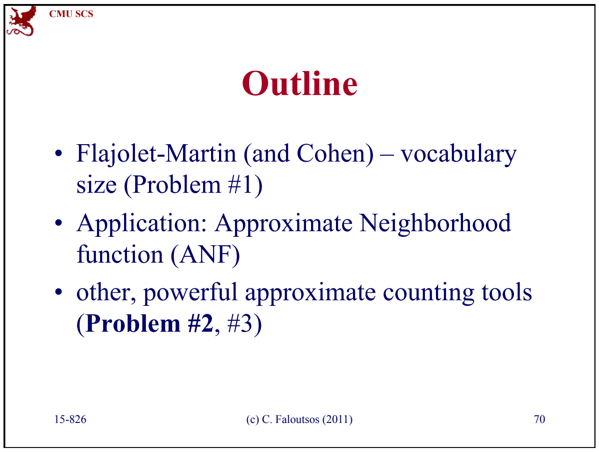


CMU SCS

Conclusions

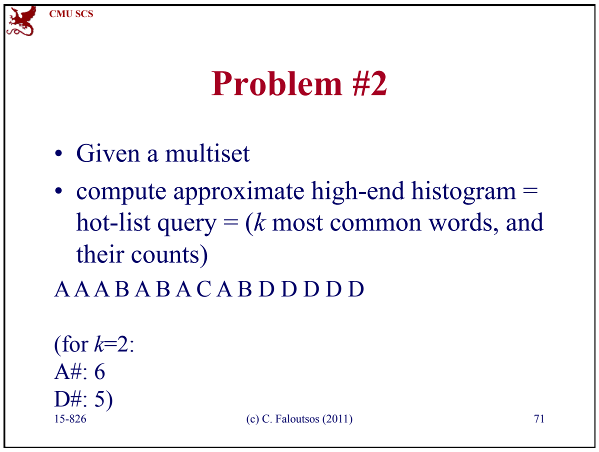
- Approximate counting (ANF / Martin-Flajolet) take minutes, instead of hours
- and discover internet facts quickly

15-826 (c) C. Faloutsos (2011) 69



Outline

- Flajolet-Martin (and Cohen) – vocabulary size (Problem #1)
- Application: Approximate Neighborhood function (ANF)
- other, powerful approximate counting tools (**Problem #2**, #3)

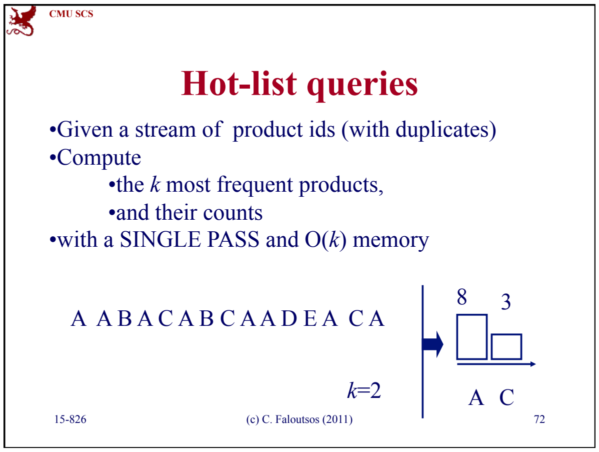


Problem #2

- Given a multiset
- compute approximate high-end histogram = hot-list query = (*k* most common words, and their counts)

A A A B A B A C A B D D D D D

(for *k*=2:
A#: 6
D#: 5)

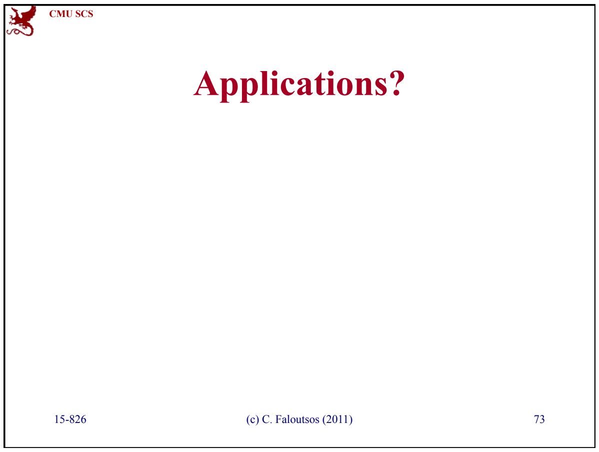


Hot-list queries

•Given a stream of product ids (with duplicates)
•Compute
 •the *k* most frequent products,
 •and their counts
•with a SINGLE PASS and O(*k*) memory

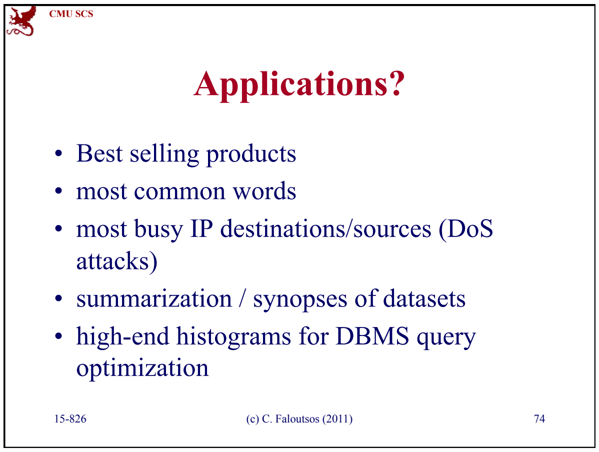
A A B A C A B C A A D E A C A

k=2



CMU SCS

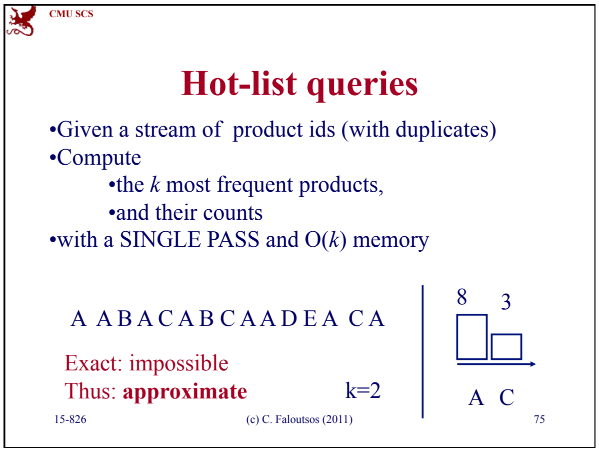
Applications?



CMU SCS

Applications?

- Best selling products
- most common words
- most busy IP destinations/sources (DoS attacks)
- summarization / synopses of datasets
- high-end histograms for DBMS query optimization



CMU SCS

Hot-list queries

•Given a stream of product ids (with duplicates)

•Compute

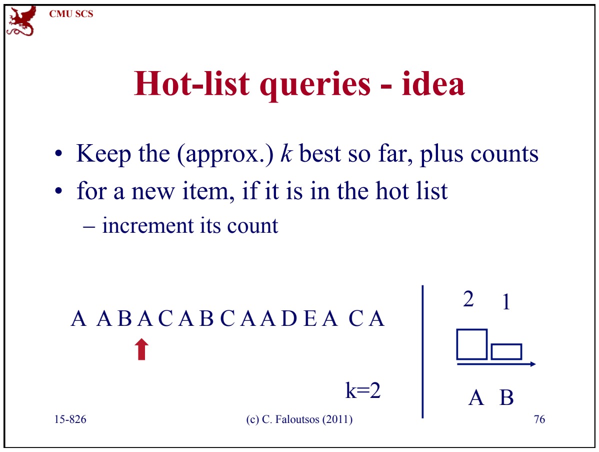
- •the *k* most frequent products,
- •and their counts

•with a SINGLE PASS and O(*k*) memory

A A B A C A B C A A D E A C A

Exact: impossible

Thus: **approximate** k=2



CMU SCS

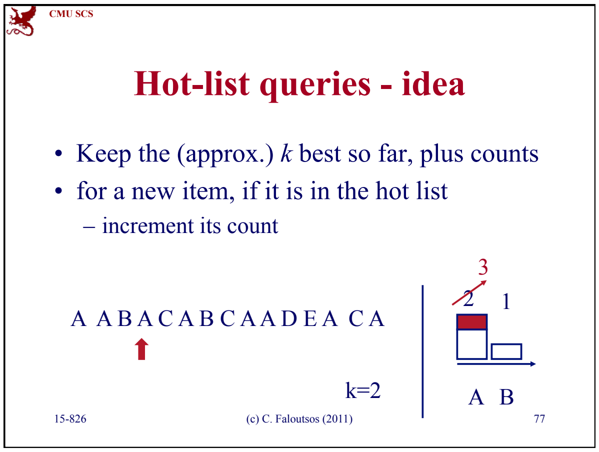
Hot-list queries - idea

- Keep the (approx.) *k* best so far, plus counts
- for a new item, if it is in the hot list
  - increment its count

A ABACABCAADEA CA

k=2

15-826 (c) C. Faloutsos (2011) 76



CMU SCS

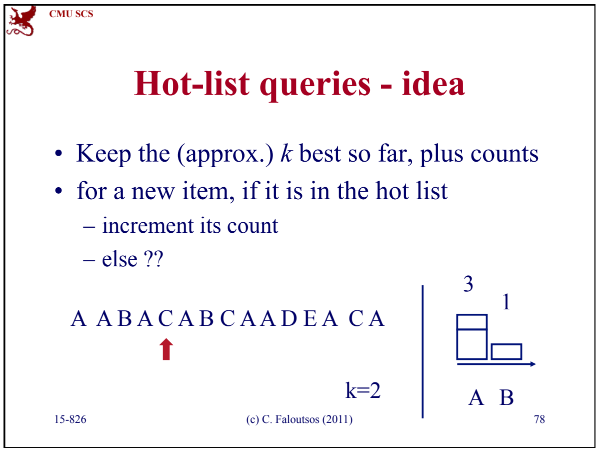
Hot-list queries - idea

- Keep the (approx.) *k* best so far, plus counts
- for a new item, if it is in the hot list
  - increment its count

A ABACABCAADEA CA

k=2

15-826 (c) C. Faloutsos (2011) 77



CMU SCS

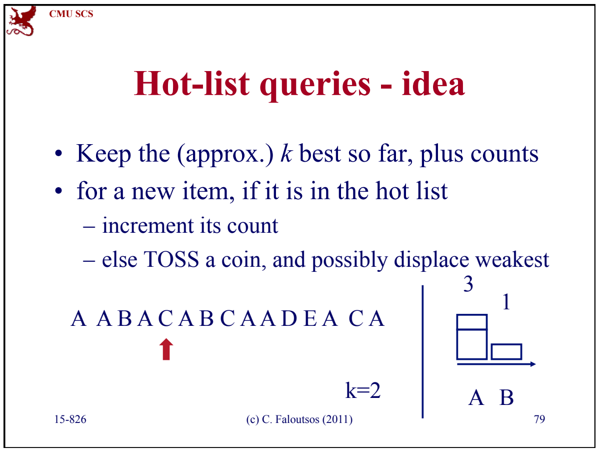
Hot-list queries - idea

- Keep the (approx.) *k* best so far, plus counts
- for a new item, if it is in the hot list
  - increment its count
  - else ??

A ABACABCAADEA CA

k=2

15-826 (c) C. Faloutsos (2011) 78



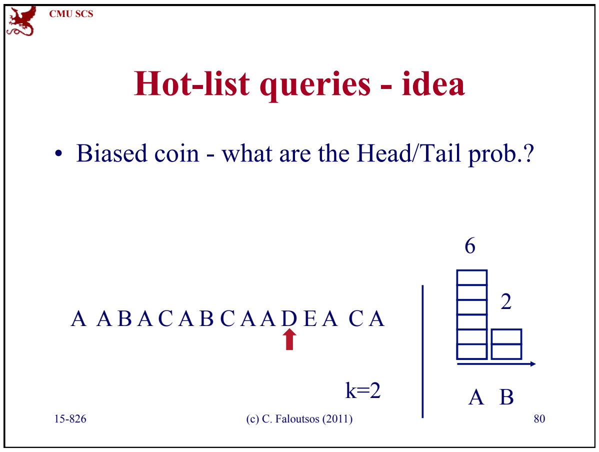
CMU SCS

Hot-list queries - idea

- Keep the (approx.) *k* best so far, plus counts
- for a new item, if it is in the hot list
  - increment its count
  - else TOSS a coin, and possibly displace weakest

A ABACABCAADEA CA

k=2



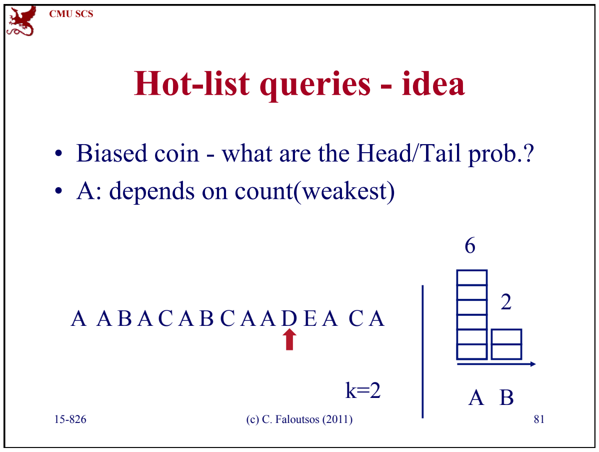
CMU SCS

Hot-list queries - idea

- Biased coin - what are the Head/Tail prob.?

A ABACABCAADEA CA

k=2



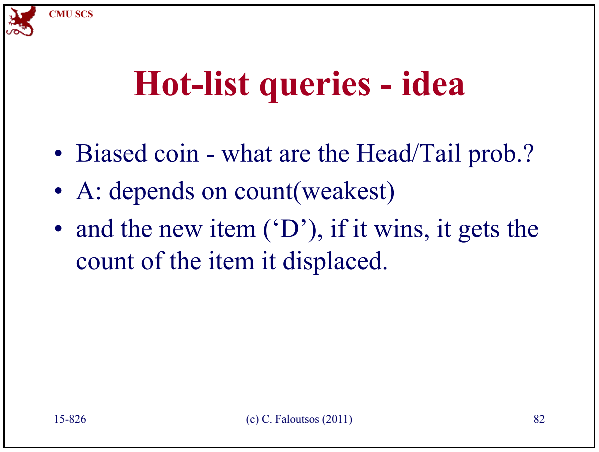
CMU SCS

Hot-list queries - idea

- Biased coin - what are the Head/Tail prob.?
- A: depends on count(weakest)

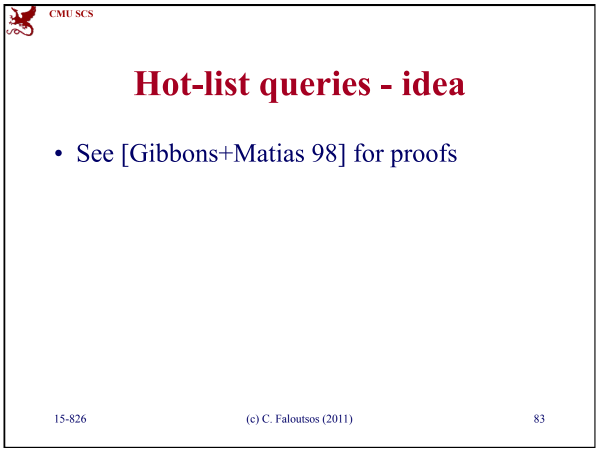
A ABACABCAADEA CA

k=2



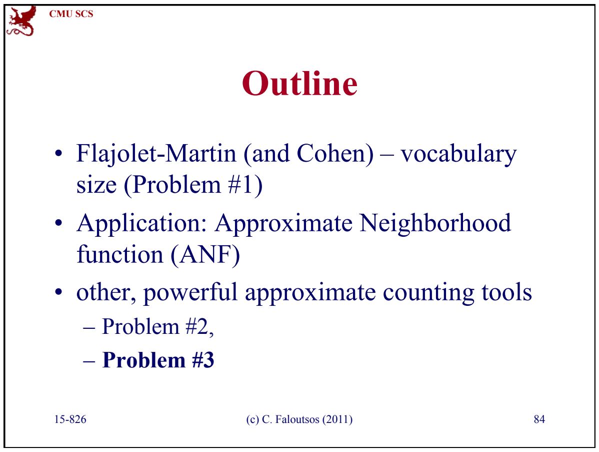
Hot-list queries - idea

- Biased coin - what are the Head/Tail prob.?
- A: depends on count(weakest)
- and the new item (‘D’), if it wins, it gets the count of the item it displaced.



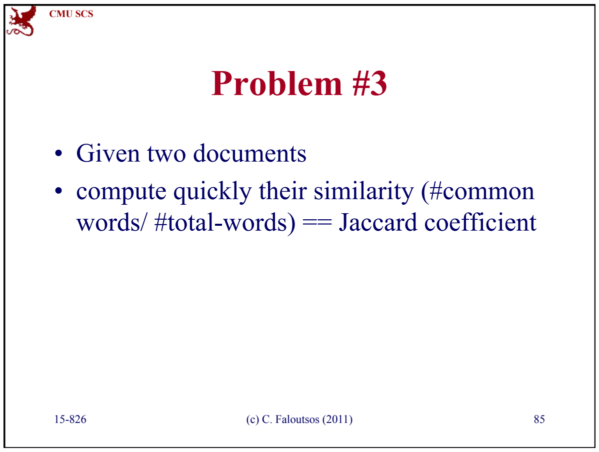
Hot-list queries - idea

- See [Gibbons+Matias 98] for proofs



Outline

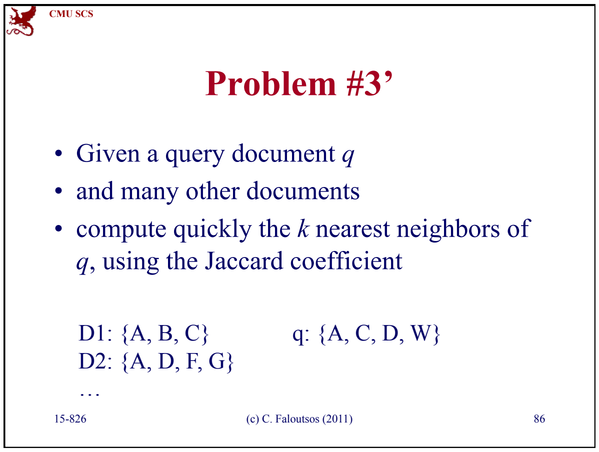
- Flajolet-Martin (and Cohen) – vocabulary size (Problem #1)
- Application: Approximate Neighborhood function (ANF)
- other, powerful approximate counting tools
  - Problem #2,
  - **Problem #3**



CMU SCS

Problem #3

- Given two documents
- compute quickly their similarity (#common words/ #total-words) == Jaccard coefficient

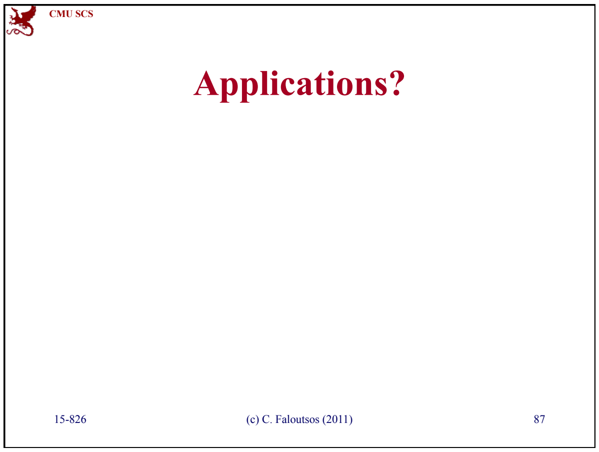


CMU SCS

Problem #3’

- Given a query document *q*
- and many other documents
- compute quickly the *k* nearest neighbors of *q*, using the Jaccard coefficient

D1: {A, B, C} q: {A, C, D, W}
D2: {A, D, F, G}
…



CMU SCS

Applications?

Applications?

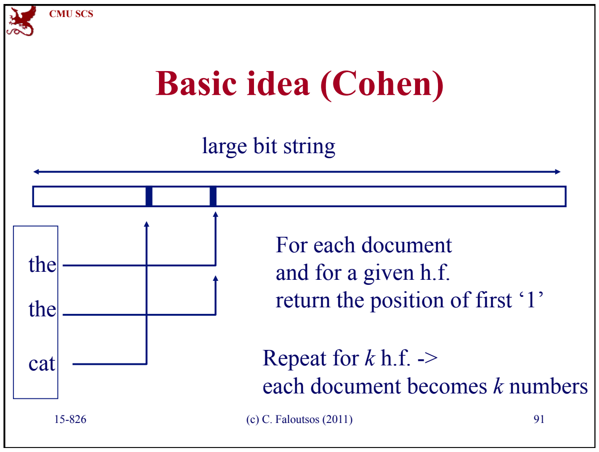
- Set comparisons eg.,
  - snail-mail address (set of trigrams)
- search engines - ‘similar pages’
- social networks: people with many joint friends (facebook recommendations)

Problem #3’

- Given a query document *q*
- and many other documents
- compute quickly the *k* nearest neighbors of *q*, using the Jaccard coefficient
- Q: how to extract a fixed set of numerical features, to index on?

Answer

- Approximation / hashing - Cohen:



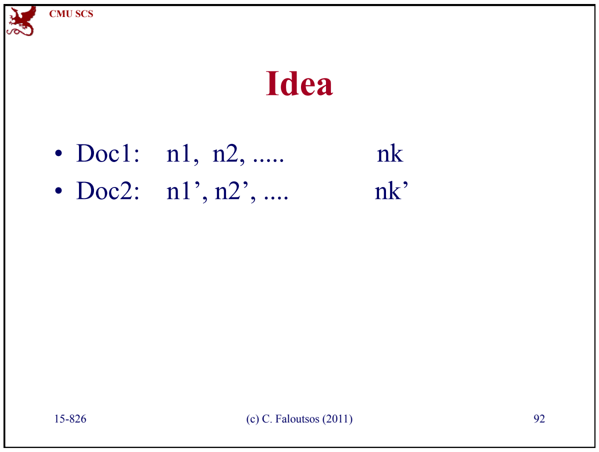
CMU SCS

Basic idea (Cohen)

large bit string
the
the
cat

For each document
and for a given h.f.
return the position of first ‘1’

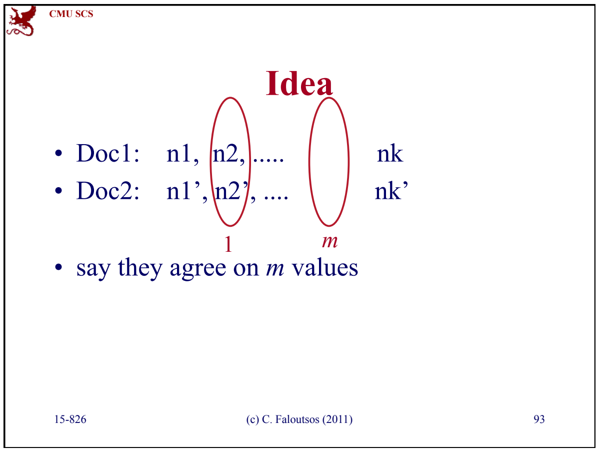
Repeat for *k* h.f. ->
each document becomes *k* numbers



CMU SCS

Idea

- Doc1: n1, n2, ... nk
- Doc2: n1’, n2’, ... nk’

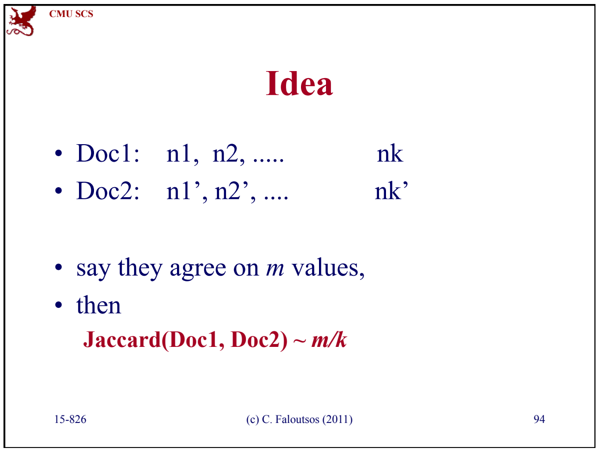


CMU SCS

Idea

- Doc1: n1, n2, ... nk
- Doc2: n1’, n2’, ... nk’

- say they agree on *m* values



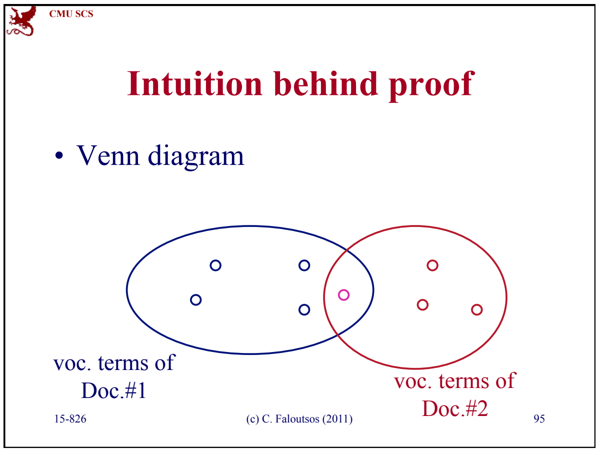
CMU SCS

Idea

- Doc1: n1, n2, ... nk
- Doc2: n1’, n2’, ... nk’

- say they agree on *m* values,
- then

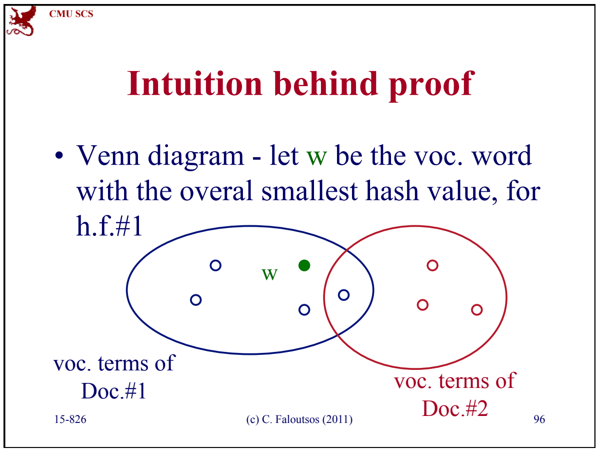
Jaccard(Doc1, Doc2) ~ *m/k*



CMU SCS

Intuition behind proof

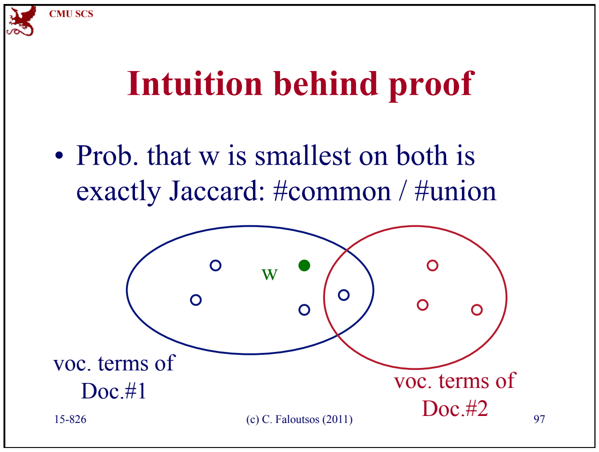
- Venn diagram



CMU SCS

Intuition behind proof

- Venn diagram - let w be the voc. word with the overal smallest hash value, for h.f.#1



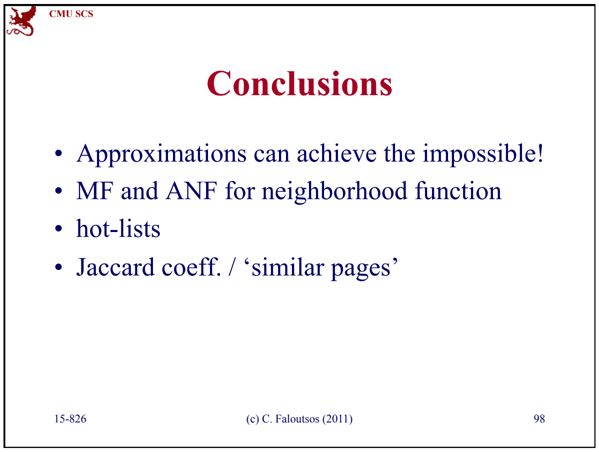
CMU SCS

Intuition behind proof

- Prob. that w is smallest on both is exactly Jaccard: #common / #union

voc. terms of Doc.#1

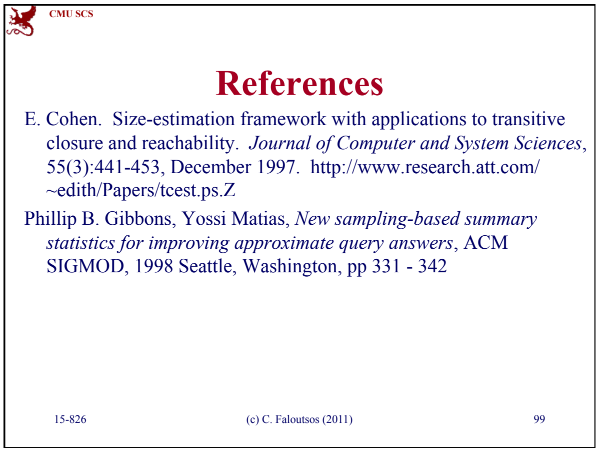
voc. terms of Doc.#2



CMU SCS

Conclusions

- Approximations can achieve the impossible!
- MF and ANF for neighborhood function
- hot-lists
- Jaccard coeff. / ‘similar pages’

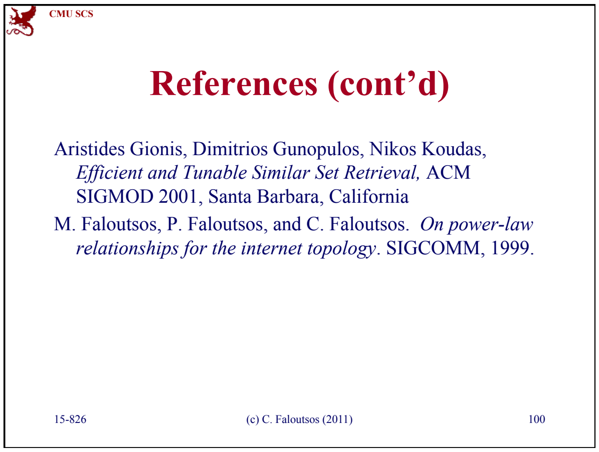


CMU SCS

References

E. Cohen. Size-estimation framework with applications to transitive closure and reachability. *Journal of Computer and System Sciences*, 55(3):441-453, December 1997. http://www.research.att.com/~edith/Papers/tcest.ps.Z

Phillip B. Gibbons, Yossi Matias, *New sampling-based summary statistics for improving approximate query answers*, ACM SIGMOD, 1998 Seattle, Washington, pp 331 - 342

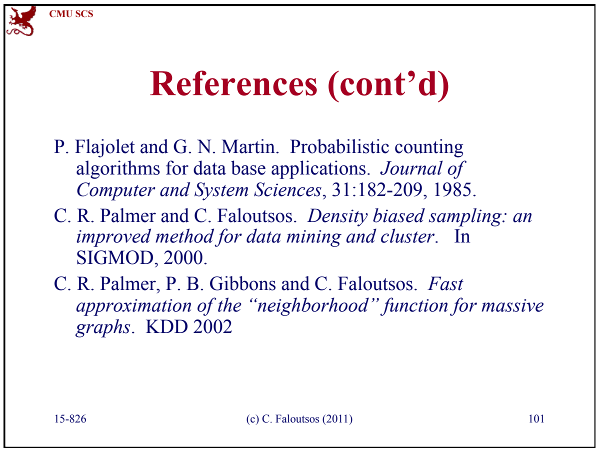


CMU SCS

References (cont'd)

Aristides Gionis, Dimitrios Gunopulos, Nikos Koudas, *Efficient and Tunable Similar Set Retrieval,* ACM SIGMOD 2001, Santa Barbara, California

M. Faloutsos, P. Faloutsos, and C. Faloutsos. *On power-law relationships for the internet topology*. SIGCOMM, 1999.



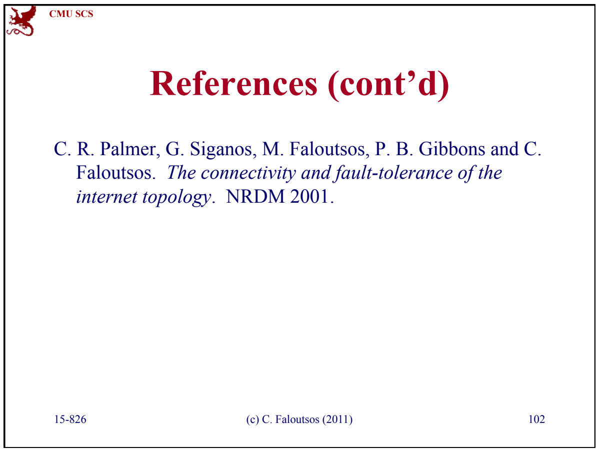
CMU SCS

References (cont'd)

P. Flajolet and G. N. Martin. Probabilistic counting algorithms for data base applications. *Journal of Computer and System Sciences*, 31:182-209, 1985.

C. R. Palmer and C. Faloutsos. *Density biased sampling: an improved method for data mining and cluster*. In SIGMOD, 2000.

C. R. Palmer, P. B. Gibbons and C. Faloutsos. *Fast approximation of the "neighborhood" function for massive graphs*. KDD 2002



CMU SCS

References (cont'd)

C. R. Palmer, G. Siganos, M. Faloutsos, P. B. Gibbons and C. Faloutsos. *The connectivity and fault-tolerance of the internet topology*. NRDM 2001.