APPROXIMATION ALGORITHMS

CARDINALITY ESTIMATION

RASMUS PAGH

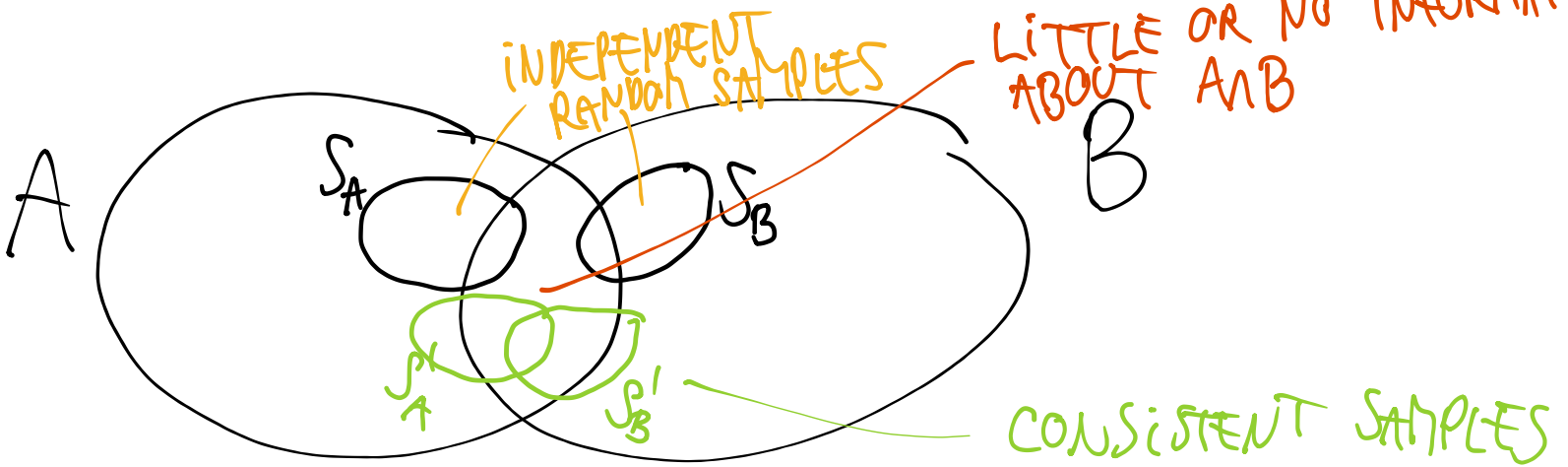
UNIVERSITY OF COPENHAGEN

TODAY

- COORDINATED SAMPLING (KMV SUMMARY)
  - APPLICATION: DISTRIBUTED COUNTING
- HYPERLOGLOG CARDINALITY ESTIMATOR
  - APPLICATION: APPROXIMATE NEIGHBORHOOD FUNCTION
- BLOOM FILTERS
  - APPLICATION: APPROXIMATING INTERSECTION SIZE

COORDINATED SAMPLING

ISSUE: UNLIKELY THAT SAME ELEMENT APPEARS IN TWO INDEPENDENT SAMPLES



OVERLAPPING DATA SETS

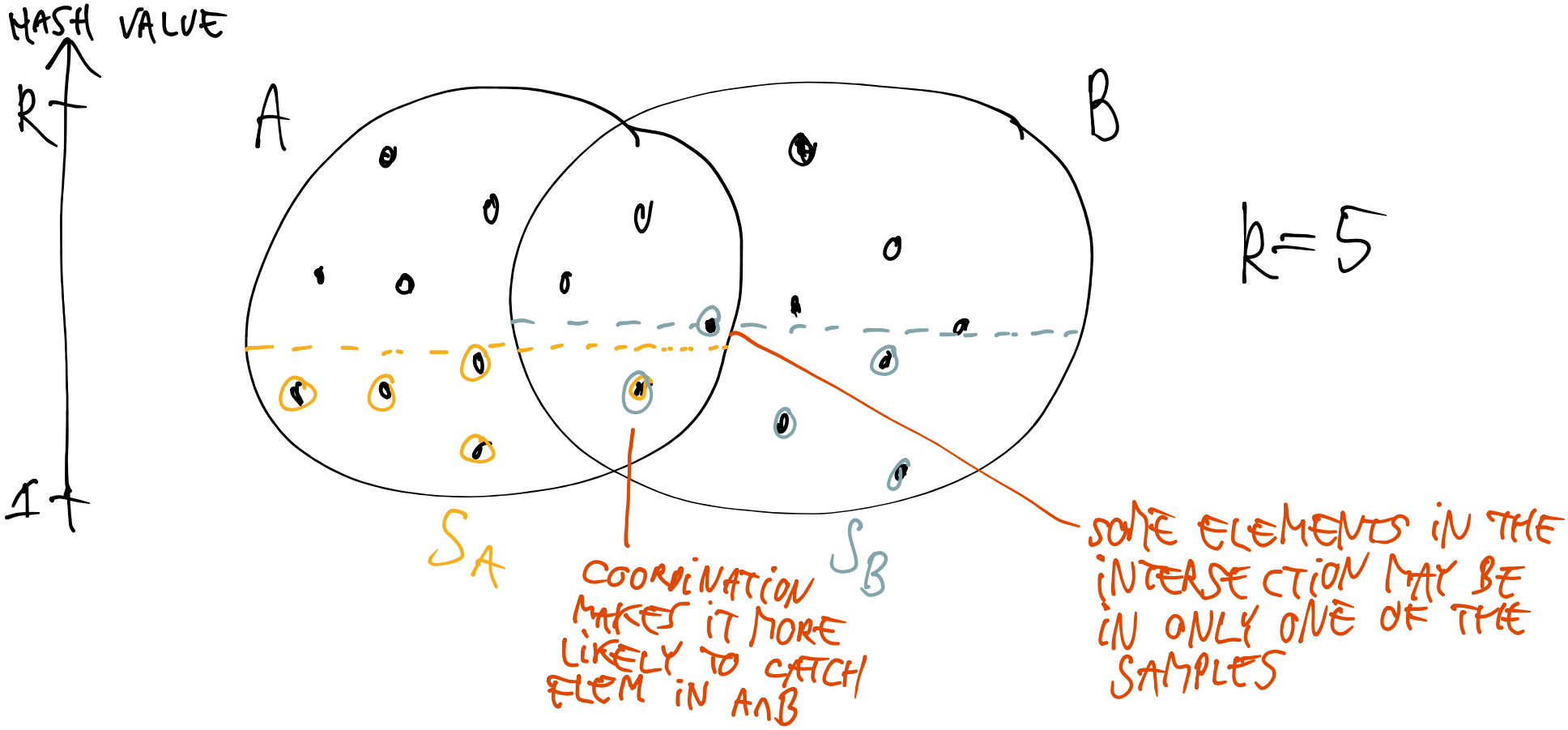
IDEA: MAKE THE DECISION OF WHETHER x SHOULD BE SAMPLED "CONSISTENT" BY BASING IT ON A RANDOM HASH VALUE h(x)

SAME HASH FUNCTION USED FOR CHOOSING S_A AND S_B

K-MINIMUM VALUE (KMV) SUMMARY

LARGE VALUE, LIKE 2^{64}

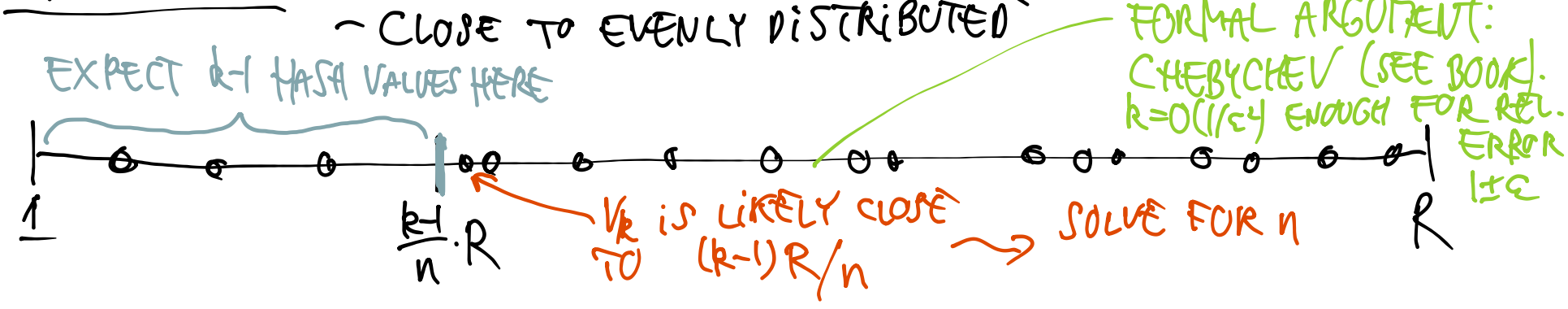
- PICK RANDOM HASH FUNCTION h, MAPPING TO $\{1, \ldots, R\}$
- SAMPLE S_A CONSISTS OF THE PAIRS $(x, h(x))$, $x \in A$ THAT HAVE THE k SMALLEST HASH VALUES.



KMV PROPERTIES

- EFFICIENT UPDATES ($O(\log k)$ OR EVEN $O(1)$)
  - INSERTING ELEMENT x MORE THAN ONCE DOES NOT CHANGE S_A.
- EFFICIENT MERGING ($O(k)$ BY COMPUTING MEDIAN OF $S_A \cup S_B$)
- SUPPORTS ESTIMATORS FOR SET CARDINALITIES:
  - LET V_R BE THE LARGEST HASH VALUE IN S_A
  - $\hat{n} = (k-1)R/V_R$ IS A GOOD ESTIMATOR FOR $n = |A|$

INTUITION: CONSIDER THE HASH VALUES $\{h(x) \mid x \in A\}$
- CLOSE TO EVENLY DISTRIBUTED



APPLICATION OF KMV

- SETS $A_1, A_2, \ldots, A_m$ (VERY LARGE)
- COLLECT KMV SUMMARIES $S_{A_1}, \ldots, S_{A_m}$
- MERGE TO FORM KMV SUMMARY S_A OF $A = \bigcup_i A_i$

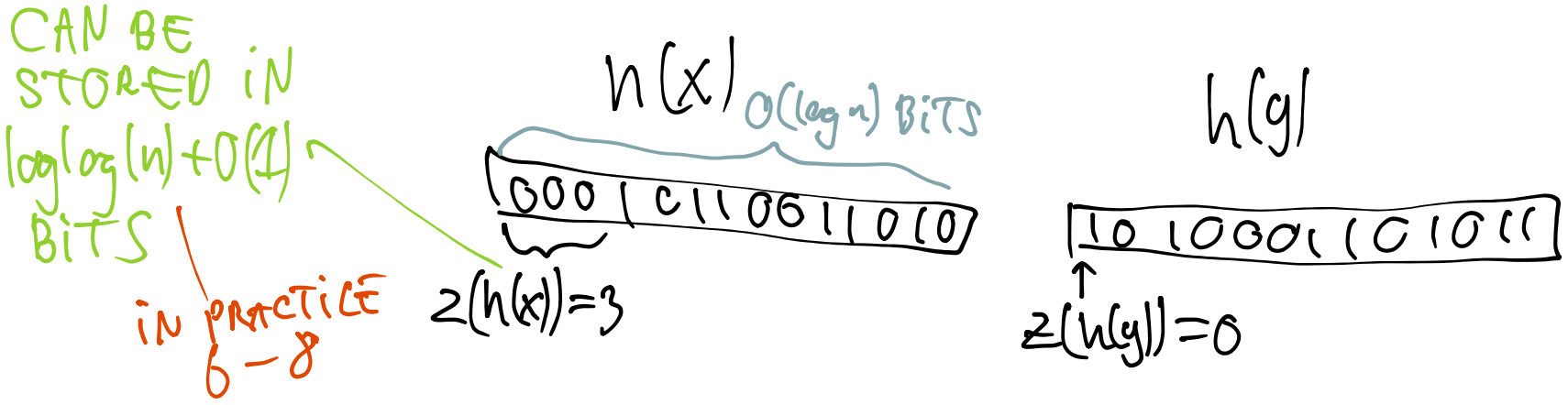
  ← DEPENDS ONLY ON SET OF ELEMENTS, NOT THE MULTIPLICITY OF EACH ELEMENT

- CAN ESTIMATE:
  - SIZE $|A|$ WITH RELATIVE ERROR $1 \pm O(1/\sqrt{k})$, WITH CONSTANT PROBABILITY $> \frac{3}{4}$
  - TAKE MEDIAN OF $O(\log(1/\delta))$ ESTIMATES TO GET ERROR PROB $\leq 1-\delta$
  - FOR GIVEN SET Q, CAN ESTIMATE $|A \cap Q|$ FROM $|S_A \cap Q|$ AND v_k

HYPERLOGLOG

- SPACE-EFFICIENT ALTERNATIVE TO KMV SUMMARY
  - DOES NOT KEEP A SAMPLE, ONLY SUPPORTS INSERTION, MERGING, CARDINALITY ESTIMATION.
- IDEA: MANY "CRUDE" ESTIMATORS USING FEW BITS

OBSERVATION. - HASH VALUES CAN BE CRUDELY ORDERED BY THE NUMBER OF LEADING ZEROS IN BIT REP.



HYPERLOGLOG DETAILS

$2^{w} \gg n$

- USE HASH FUNCTIONS h, g WHERE $h(x) \in \{1, \dots, k\}$ AND $g(x) \in \{0,1\}^{w}$
- MAINTAIN COUNTERS $C_1, \dots, C_k$ WHERE C_i IS A CRUDE COUNTER FOR $A_i = \{x \in A \mid h(x) = i\}$: $C_i = \max_{x \in A_i} (z(g(x)))$.
  - $z(\cdot)$: NUMBER OF LEADING ZEROS
- MERGING SKETCHES $C_1 C'$ (SAME h, g):

  $C_i'' = \max(C_i, C_i')$ ← IDENTICAL TO THE SKETCH THAT WOULD RESULT FROM INSERTING ALL ELEMENTS DIRECTLY

- FIRST ATTEMPT
- ESTIMATION: $2^{w - C_i} \approx$ SMALLEST HASH VALUE IN A_i

  $\rightsquigarrow$ ESTIMATE $|A_i| \approx 2^w / 2^{w - C_i} = 2^{C_i}$

  $\rightsquigarrow$ ESTIMATE $|A| = \sum_i |A_i| \approx \sum_i 2^{C_i}$

  VERY SENSITIVE TO A FEW LARGE COUNTERS

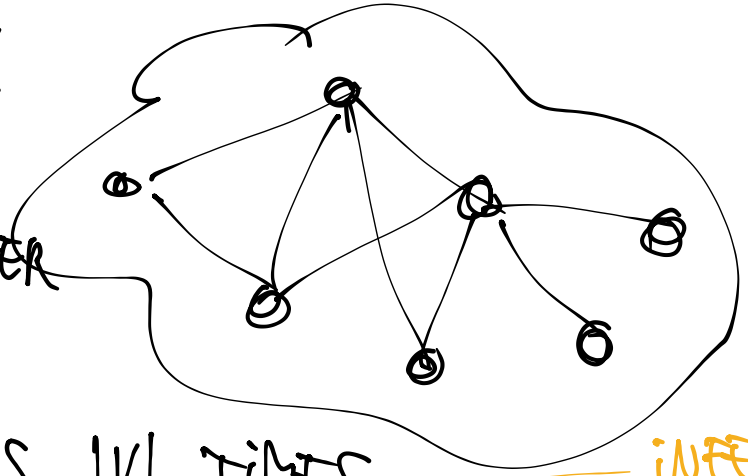
BETTER: TAKE HARMONIC MEAN OF ESTIMATORS 2^{C_i} — SEE DISCUSSION IN BOOK

OUTLIER ROBUST — AVERAGE REL. ERROR $1.06/\sqrt{k}$

HYPERLOGLOG APPLICATION: ANF

SETTING: LARGE, UNDIRECTED GRAPH (V, E), CONNECTED

TASK: FOR ALL $v \in V$, COMPUTE THE AVERAGE DISTANCE TO THE OTHER VERTICES IN V



$|V| = 10^9$

$|E| = 10^{11}$ ← "SPARSE", $|E| \ll |V|^2$

BASELINE: RUN BFS $|V|$ TIMES. ← INFEASIBLE ON A SINGLE MACHINE

ALTERNATIVE: SETS!

- S_i^v: SET OF VERTICES AT DISTANCE $\leq i$ FROM v.
- $S_0^v = \{v\}$, $S_{i+1}^v = \left(\bigcup_{\{v,w\} \in E} S_i^w\right) \cup S_i^v$
- $HLL(S_{i+1}^v) = \text{merge}\left(S_i^v, [S_i^w \mid \{v,w\} \in E]\right)$

FEW ITERATIONS NEEDED IF GRAPH HAS SMALL DIAMETER

- COMPUTE IN TIME $O(k|E|)$ FOR ALL $v \in V$
- USE TO ESTIMATE $|S_i^v|$ FOR $i = 1, \ldots$?

"APPROXIMATE NEIGHBORHOOD FCT."

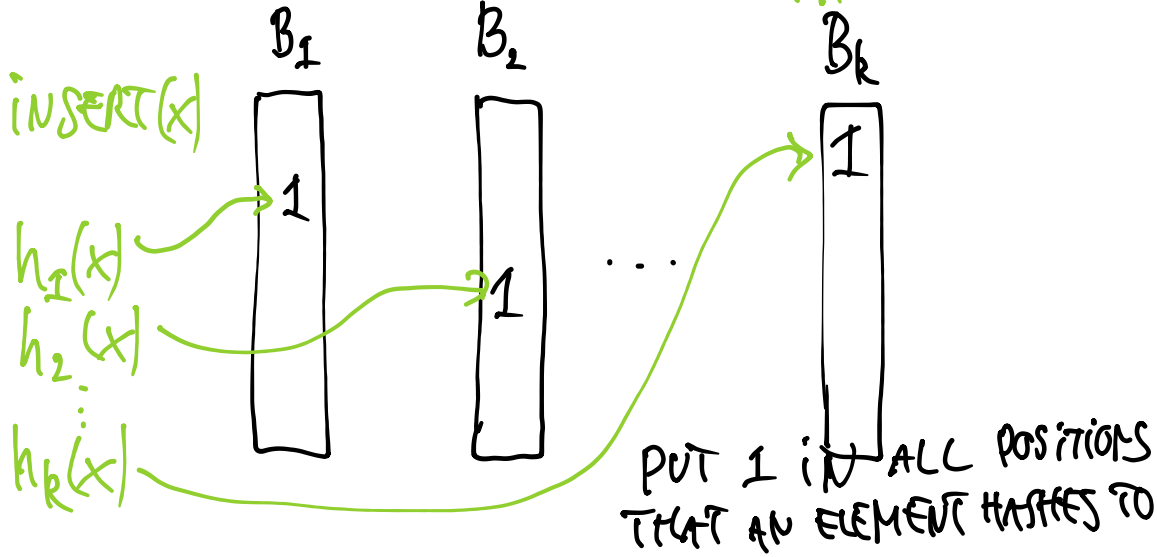
SPLIT

BLOOM FILTER

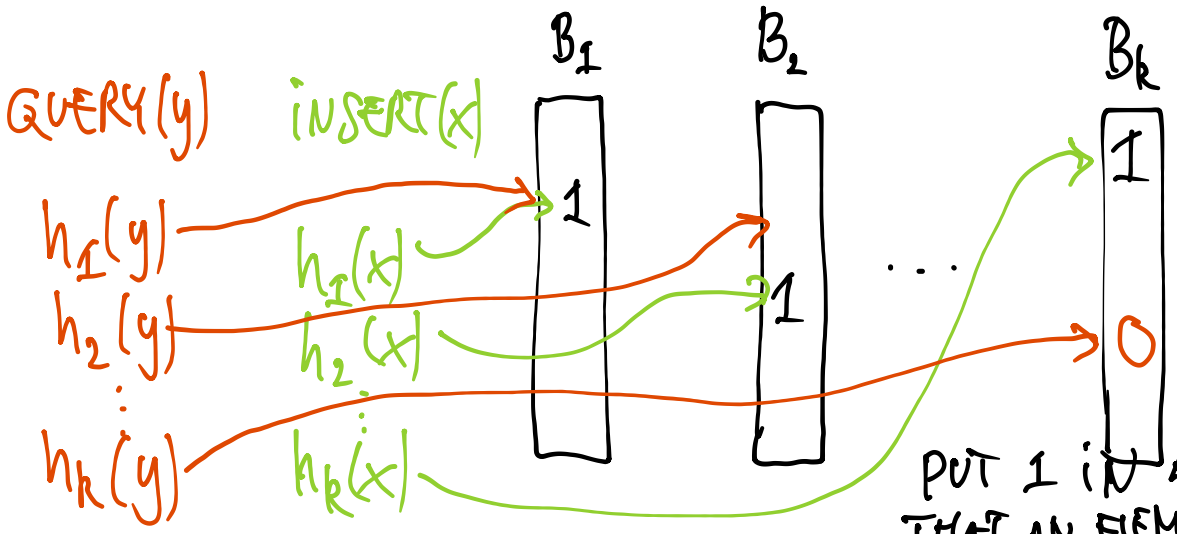
- APPROXIMATE VERSION OF A HASHSET, PARAMETER $\varepsilon > 0$
- CAN INSERT ELEMENTS INTO SET S
- CAN ASK ABOUT MEMBERSHIP QUERY$(x) = \begin{cases} 1 & \text{IF } x \in S \\ 0 & \text{IF } x \notin S \text{ WITH PROBABILITY} \geq 1-\varepsilon \\ 1 & \text{IF } x \notin S \text{ WITH PROB.} \leq \varepsilon \end{cases}$ FALSE POSITIVE
- SPACE USAGE, $|S| = n$: $m = O(n \log(1/\varepsilon))$ BITS

OFTEN MUCH LESS THAN REQUIRED TO STORE ELEMENTS

- HOW IT WORKS. PARAM.
  - HASH FUNCTIONS $h_1, \ldots, h_k$
  - MAPPING ELEMENTS TO $\{1, \ldots, m/k\}$ (SPACE)
  - BIT VECTORS $B_1, \ldots, B_k$, EACH OF SIZE m/k



BLOOM FILTERS, CONTINUED



PUT 1 IN ALL POSITIONS THAT AN ELEMENT HASHES TO

IF $B_i[h_i(y)]=0$ WE KNOW THAT y WAS NEVER INSERTED → ANSWER 0

ANSWER 1

IF $B_i[h_i(y)]=1$ FOR $i=1,\dots,k$, THEN y MAY HAVE BEEN INSERTED

(NON-OPTIMAL) PARAMETER CHOICE: LET $k=\log_2(1/\varepsilon)$, $m=2kn=2n\log_2(1/\varepsilon)$.

NEED TO KNOW (BOUND ON) SET SIZE IN ADVANCE

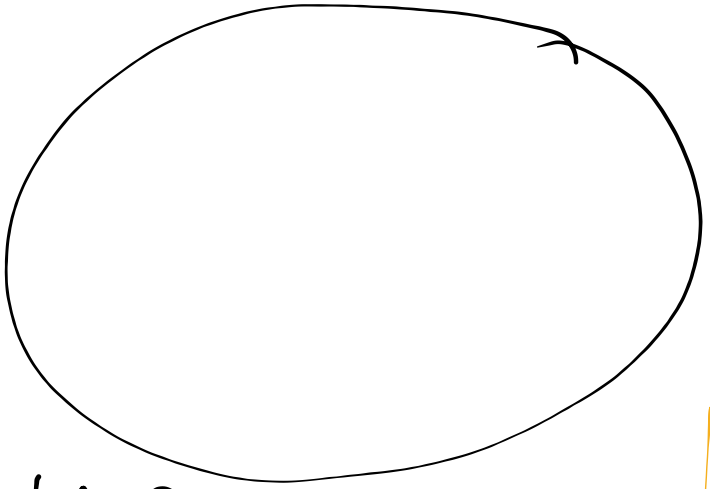
FOR y NEVER INSERTED:

$$\Pr[B_i[h_i(y)]=1 \text{ FOR } i=1,\dots,k]=\prod_{i=1}^{k}\Pr[B_i[h_i(y)]=1]\le\prod_{i=1}^{k}\frac{1}{2}=\varepsilon.$$

AT MOST n OUT OF 2n ENTRIES ARE 1

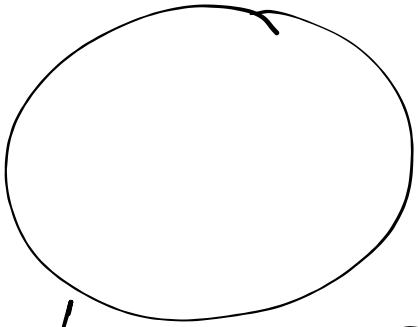
BLOOM FILTER APPLICATION

ESTIMATING THE NUMBER OF COMMON ELEMENTS

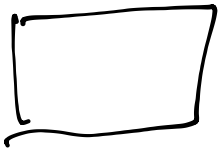


LARGE SET A

HOW LARGE IS $A \cap B$?



SMALLER SET B, BUT DOES NOT FIT IN MEMORY



BLOOM FILTER FOR B, FITS IN MEMORY

ALGORITHM

FOR EACH $x \in A$, LOOK UP x IN BLOOM FILTER FOR B, t POSITIVES

- ESTIMATE $|A \cap B| \approx t - \varepsilon |A|$

↑ EXPECTED NUMBER OF FALSE POSITIVES