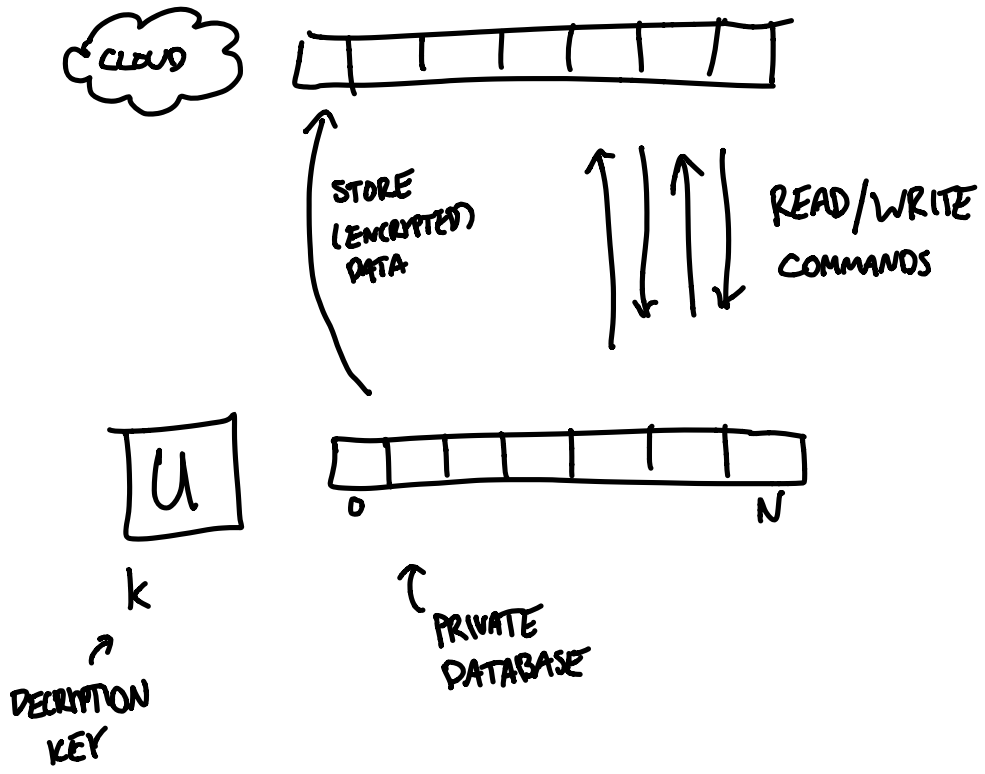
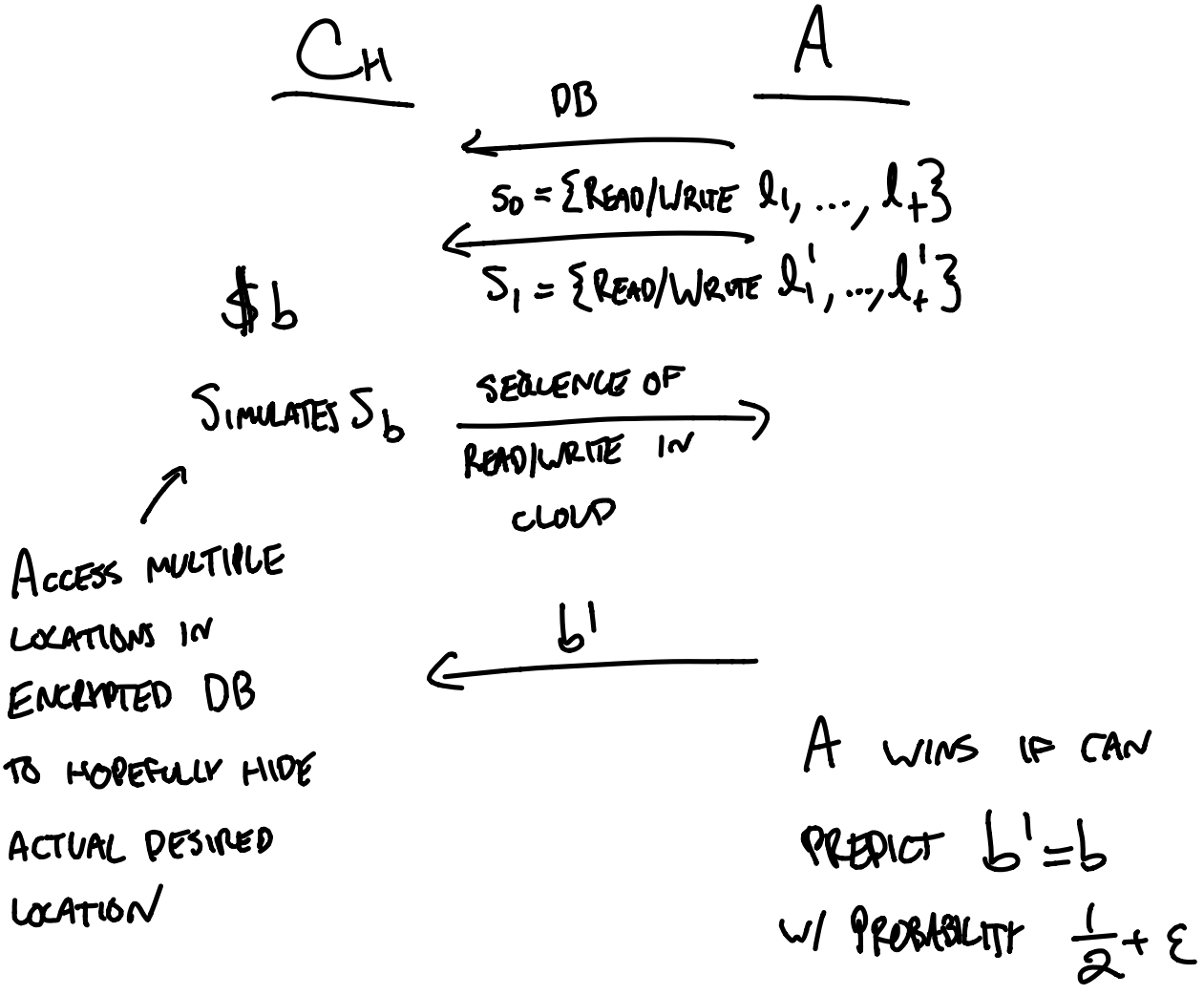
OBLIVIOUS RAM :



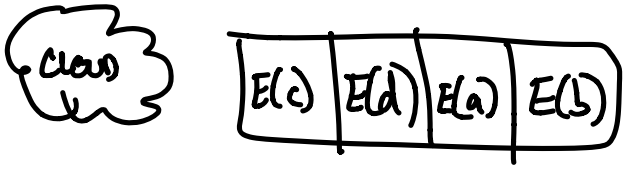
OBLIVIOUS DEFN: (INTUITIVE) ANY TWO SEQUENCES OF READ/WRITE COMMANDS ARE INDISTINGUISHABLE

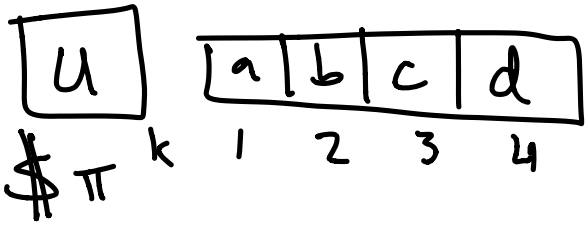
FORMAL DEFN :



A WINS IF CAN PREDICT $b' = b$ W/ PROBABILITY $\frac{1}{2} + \varepsilon$

SPECIAL CASE: S_0 + S_1 BOTH PERMUTATIONS OF ALL N ELEMENTS





TO READ INDEX i OF DB, READ INDEX $\pi(i)$ OF ENCRYPTED DB.

AS LONG AS $S_0 \neq S_1$ PERMUTATIONS, SINCE π IS HIDDEN, $(S_0 \circ \pi) \neq (S_1 \circ \pi)$ ARE INDISTINGUISHABLE, BOTH ARE RANDOM PERMUTATIONS.

BUT IF $S_0 = 1, 1, 1, 1$

$S_1 = 1, 2, 3, 4$

A KNOWS $b = 0$ IF $\mathcal{U}$ ACCESSES SAME ENCRYPTED LOCATION EVERY TIME.

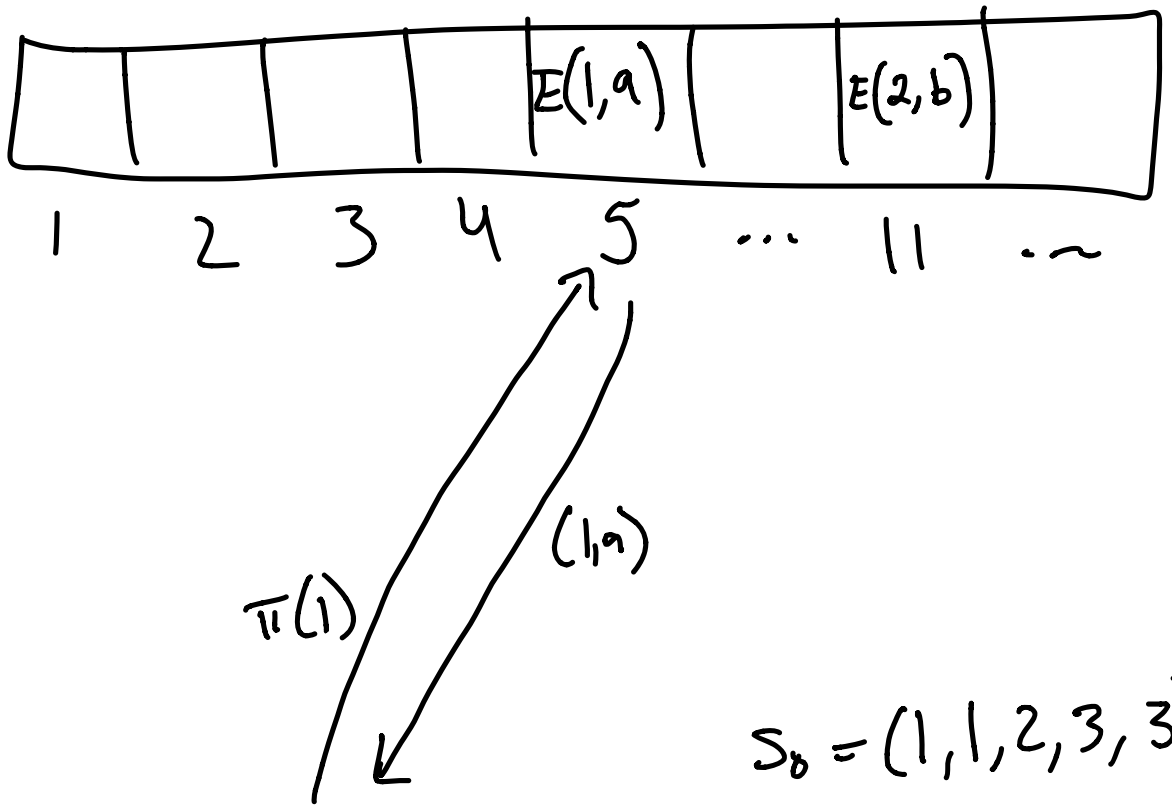
$b = 1$ IF $\mathcal{U}$ ACCESSES DIFFERENT LOCATIONS.

PROBLEM BECOMES INTERESTING WHEN USER'S MEMORY IS SMALL.

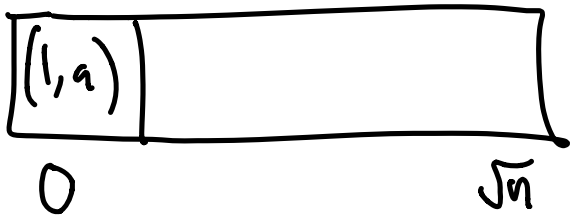
OTHERWISE, CAN JUST STORE DATABASE LOCALLY. BORING!

IDEALLY: $O(1) * \text{poly}(k)$

WHAT IF USER'S LOCAL MEMORY IS $O(\sqrt{n})$?



$S_6 = (1,1,2,3,3)$



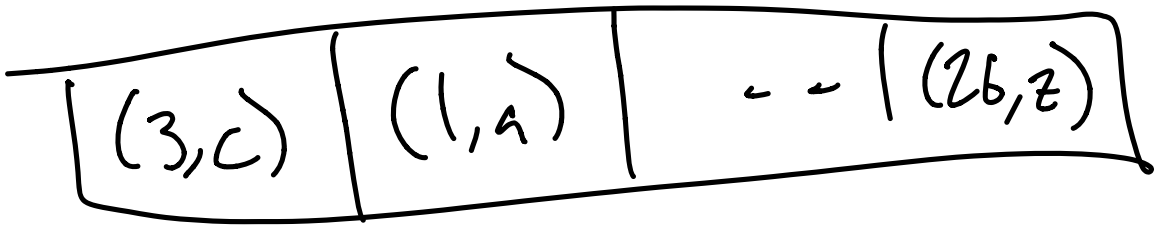
$1 \to 5$
$2 \to 11$
$\vdots$
$26 \to 3$

INSTEAD OF READING LOCATION TWICE, STORE VALUE IN LOCAL CACHE AND MAKE "FAKE" QUERY TO UNREAD LOCATION IN ENCRYPTED DB.

WHAT DO WE DO WHEN CACHE RUNS OUT OF SPACE?

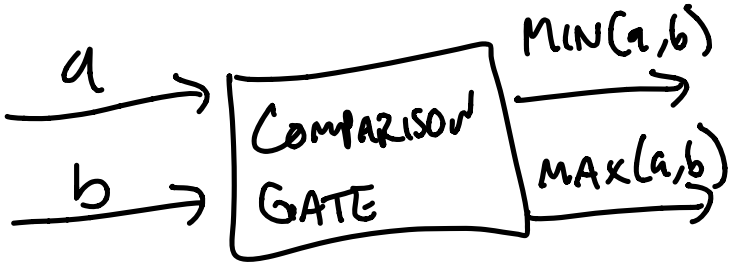
IDEA: PICK NEW π', REARRANGE ENCRYPTED DB, & STORE PREVIOUS CACHE INFO IN CLOUD.

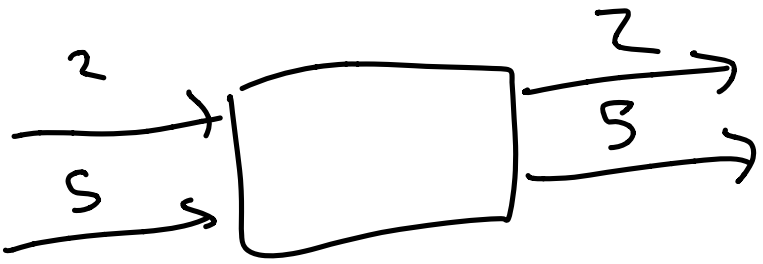
RELATED PROBLEM

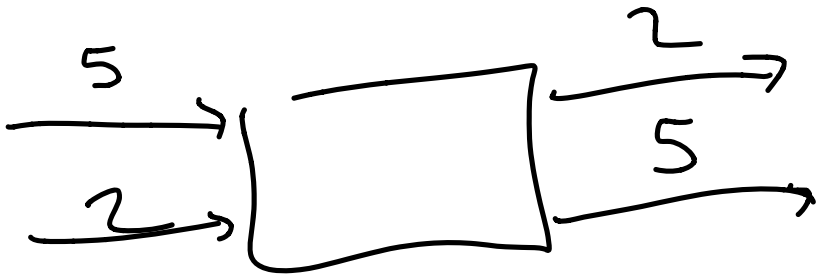


HOW TO SORT BY KEY WITHOUT REVEALING DATA?

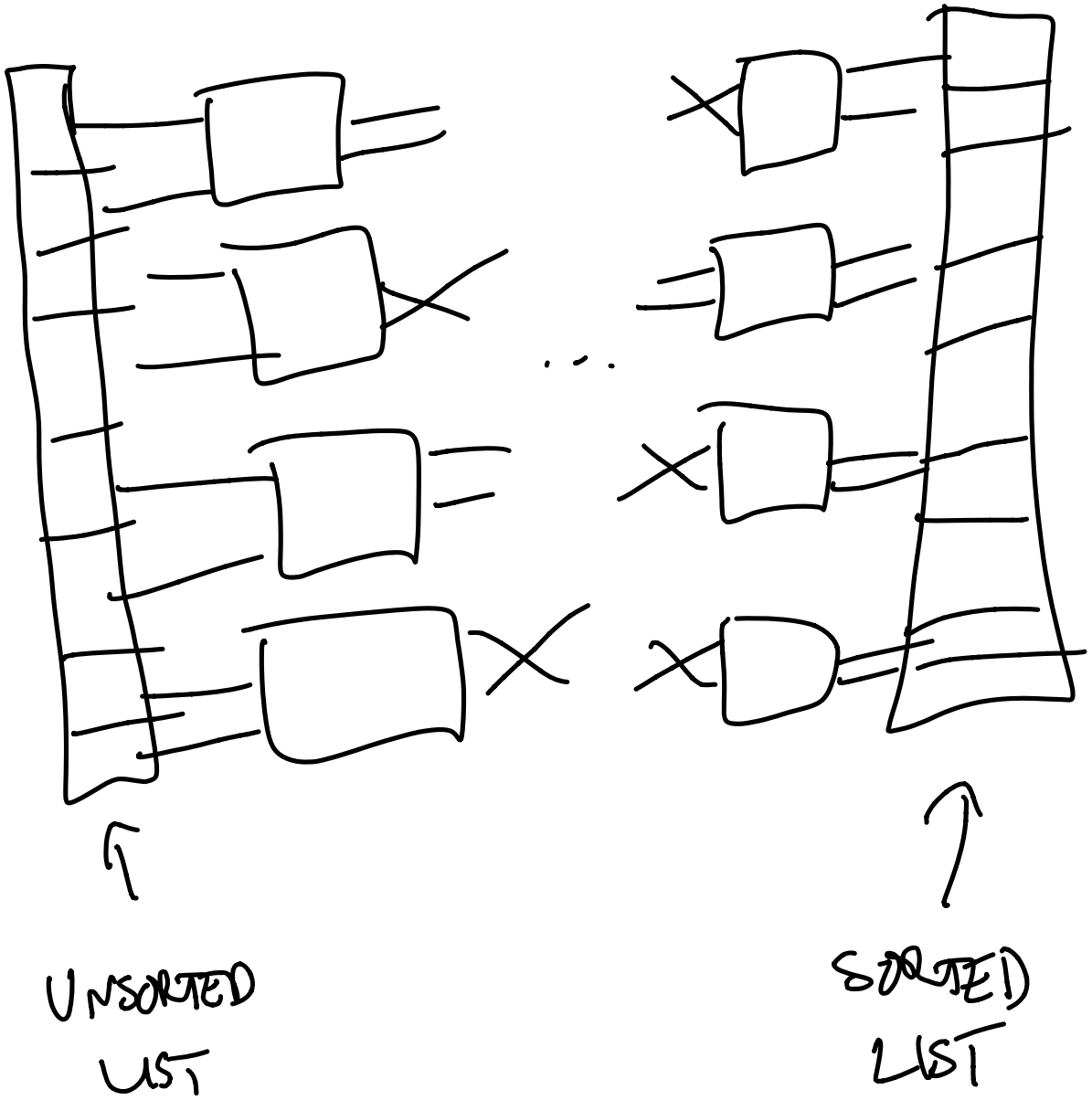
USE SORTING NETWORK!







CAN CONSTRUCT NETWORK OF COMPARISON GATES W/ N INTEGER INPUT WIRES + N OUTPUT WIRES, + RETURNS A SORTED LIST!


UNSORTED
LIST
SORTED
LIST

BATCHER SORTING NETWORK:

SMALL CONSTANT
4

$O(N \log^2 N)$ GATES

$O(\log^2 N)$ DEPTH

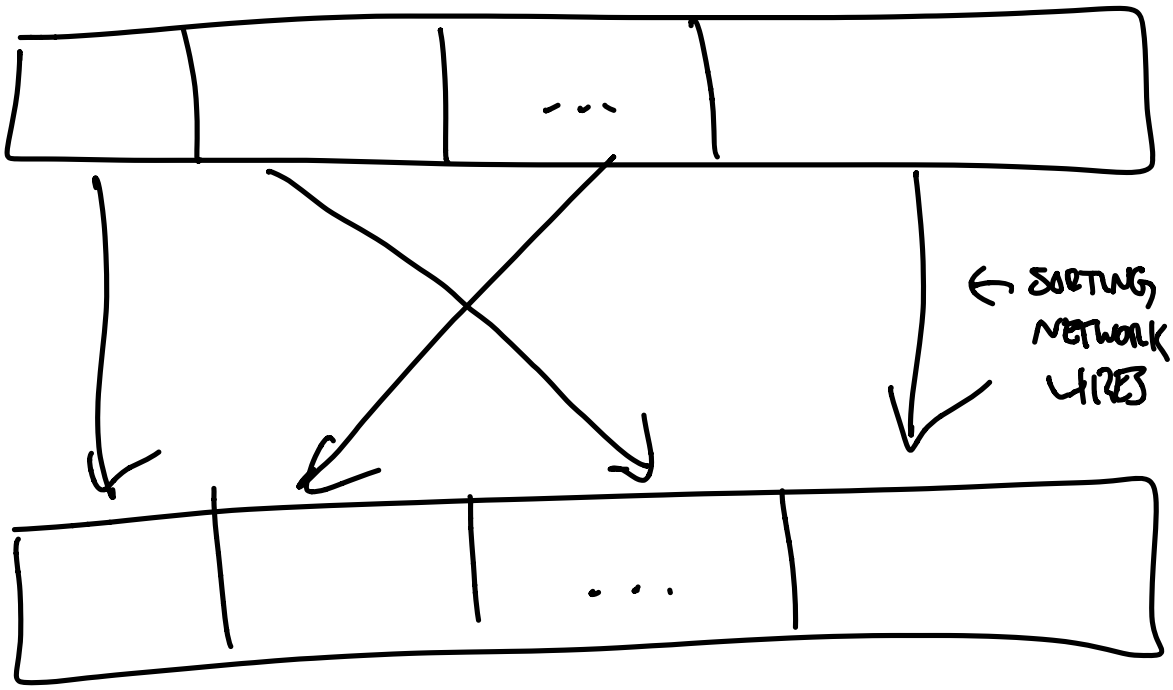
AKS SORTING NETWORK:

LARGE CONSTANT, BILLIONS

$O(N \log N)$ GATES

$O(\log N)$ DEPTH

TO SORT KEYS WITHOUT REVEALING ACCESS PATTERN TO ADVERSARY, JUST FOLLOW SORTING NETWORK!



USER DOWNLOADS 2 ELEMENTS AT A TIME
+ RE-UPLOADS IN NEW LOCATIONS, LOW MEMORY.

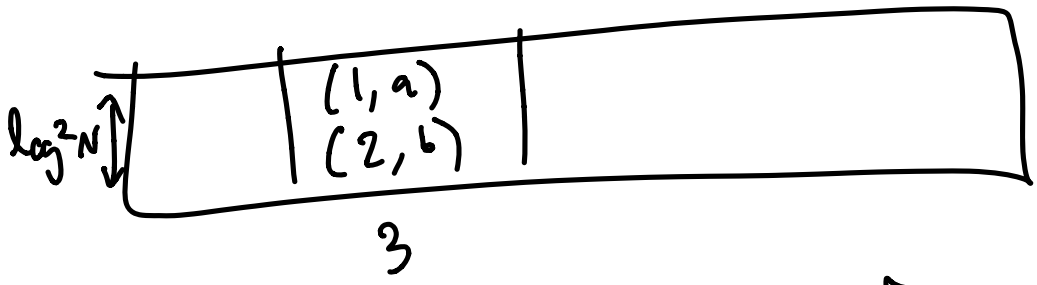
ACCESS PATTERN DETERMINED BY NETWORK WIRING, WHICH IS FIXED! REVEALS NOTHING.

HOW DOES OBLIVIOUS SORTING HELP OUR $O(\sqrt{N})$ SOLUTION?

Aside: How do we store it with limited memory?

Use PRF to create hash table instead.

Make hash table of $f_s(i) \bmod 10N$.



$f_s(1) \bmod 10N = 3$

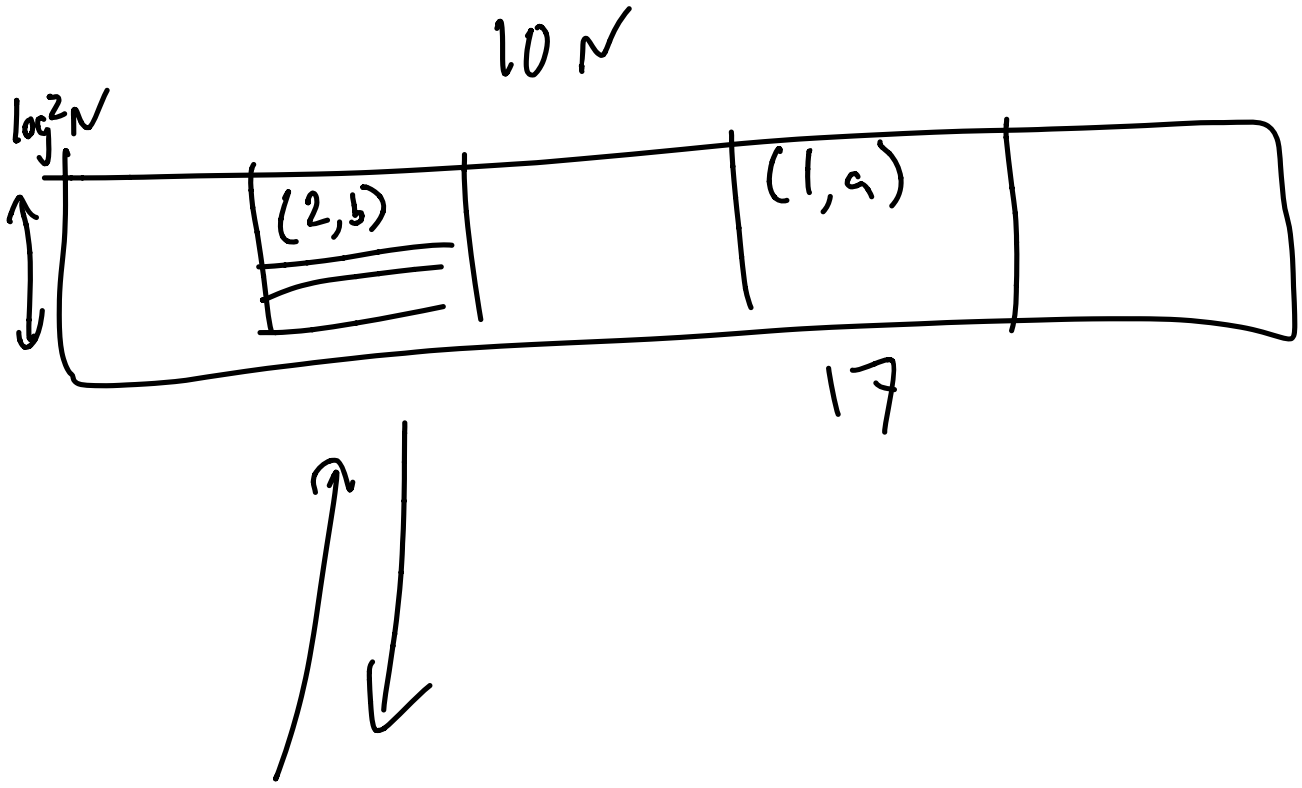
$f_s(2) \bmod 10N = 3$

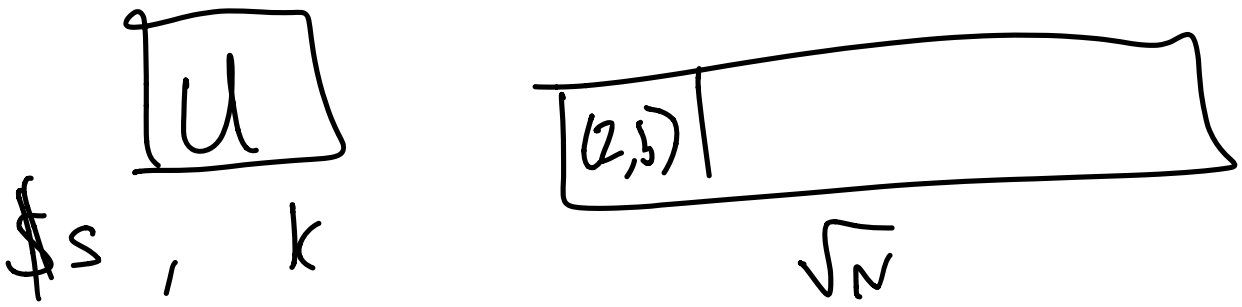
Pr[overflow] = negligible

To access item 5, compute

$f_s(5) \longrightarrow 17$ + download everything in bucket 17, look for $(5, e)$.

Full $\sqrt{N}$ solution

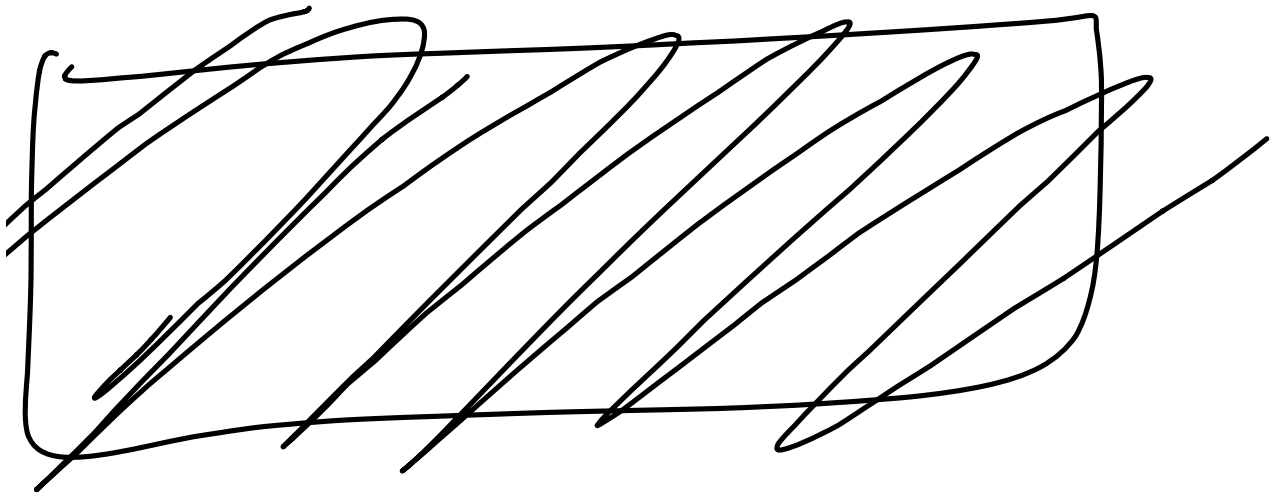

10 N
$\log^2 N$
(2,5)
(1,9)
17


U
$s, k
(2,5)
$\sqrt{N}$

- READ UNTIL BUFFER IS FULL
- PICK NEW S'
- REARRANGE CLOUD HASH TABLE USING SORTING NETWORK
- CAN STORE BUFFER IN CLOUD, PAY $\sqrt{n}$ FOR EACH READ.

EVERY $\sqrt{n}$ STEPS, MUST "REDO" ENTIRE DATA STRUCTURE WHICH COSTS $O(n \log^2 n)$. PER STEP: $O(\sqrt{n} \log^2 n)$

MEMORY: CONSTANT $\cdot$ poly(k)



LIST OF TOPICS FOR FINAL EXAM:

HARDCORE BITS

INDISTINGUISHABILITY

PRG, PRF

DIGITAL SIGNATURES
UNIQUE SIGNATURES

VRF

MERKLE TREES

ZK: 3 TYPES (COMPUTATIONAL, STATISTICAL, PERFECT)

ZK PROOF VS. ARGUMENT

PKE:

- SEMANTIC SECURITY
- CCA-1
- CCA-2

) DEFNS. ONLY

EL-GAMAL ENCRYPTION

PROOF OF SECURITY BASED ON DDH

COMMITMENTS:

- NAOR COMMITMENTS
- Com BASED ON CLAW-FREE PERMUTATIONS

OT, 1-2-OT, CORRELATED RANDOMNESS

BEAVER'S PRE-COMPUTED OT

OT $\Rightarrow$ HONEST-BUT CURIOUS MPC

MALICIOUS MPC, THRESHOLD BOUNDS

YAO'S GARBLED CIRCUITS (PROOF OF SECURITY)

BGW PROTOCOL (INFO-THEORETIC MPC)

BYZANTINE AGREEMENT

(IMPOSSIBILITY FOR $t \geq n/3$)

DEF OF PUBLIC LEDGER FOR BLOCKCHAINS

POW, POS

PIR, PIR $\Rightarrow$ CRHF

ORAM