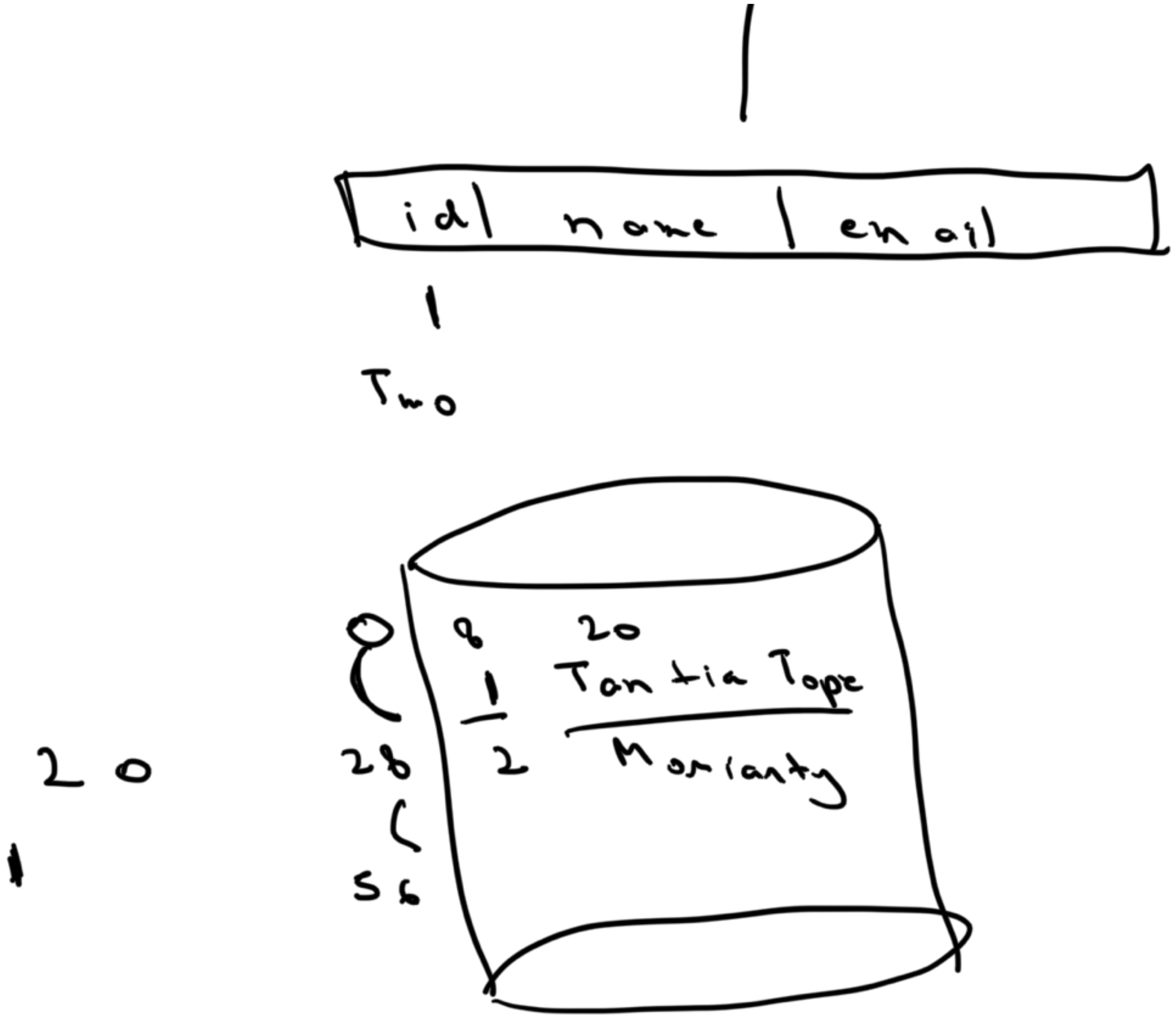
HLD - NoSQL internals

Tanmay Kacker

Agenda

- NoSQL storage
- Motivation
- Key-value store
- WAL
- LSM (SSTables + Memtable)
- Compaction

SQL	NoSQL
Structured	Unstructured



id
name
email
1
Two
0
8
20
1
Tantia Tope
28
2
Moriarty
56
2 0
1

→ ID NAME EMAIL

ID NAME

Key-value

- Redis

- get(key)
- update(key, value)

Solution 1 - File

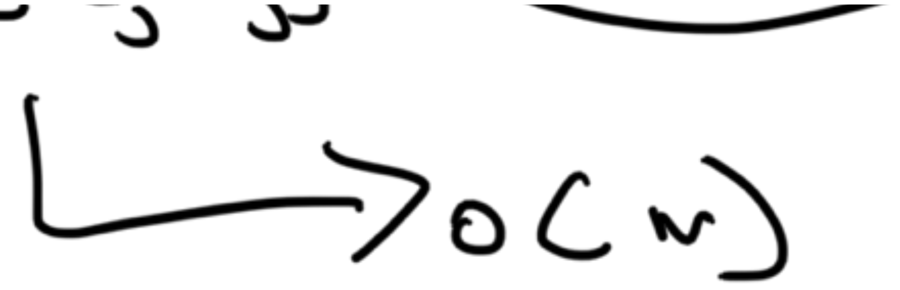
M → "1": "Tantia Tope"
"2": "Sherlock" }

ge(1)

Key	Value
1	Tantia
2	Sherlock

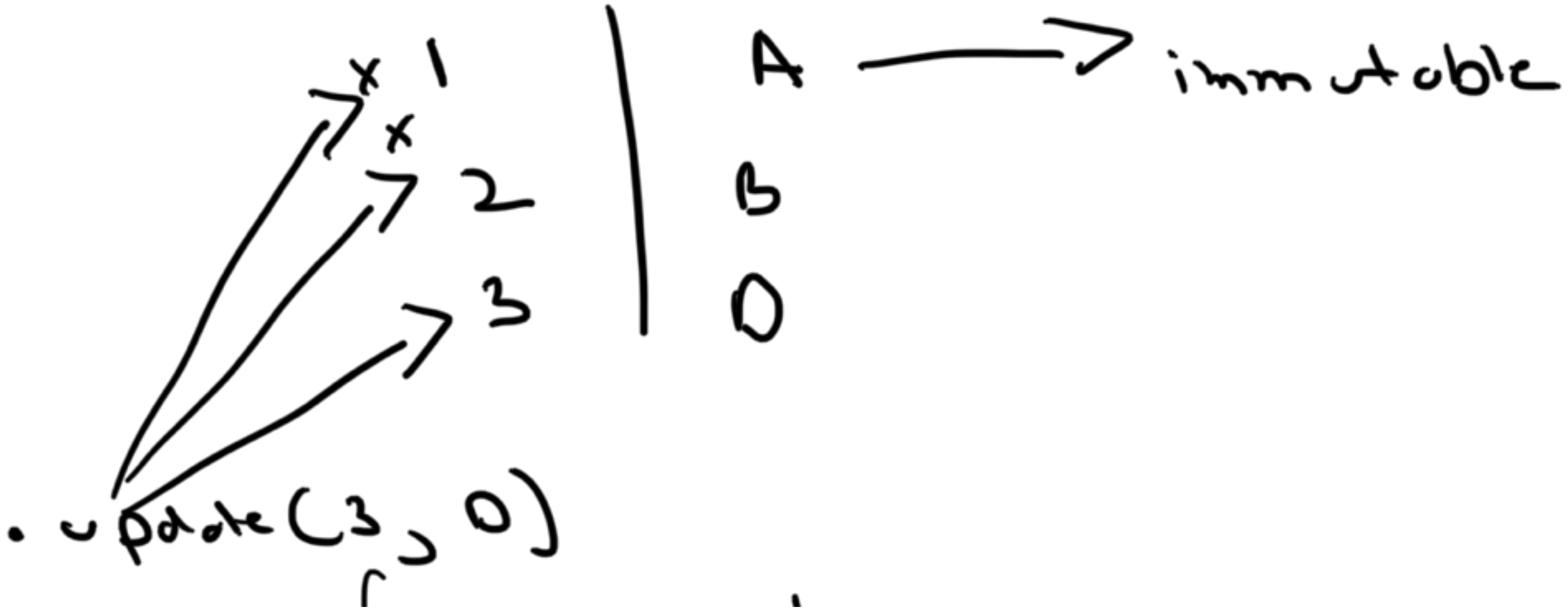
D

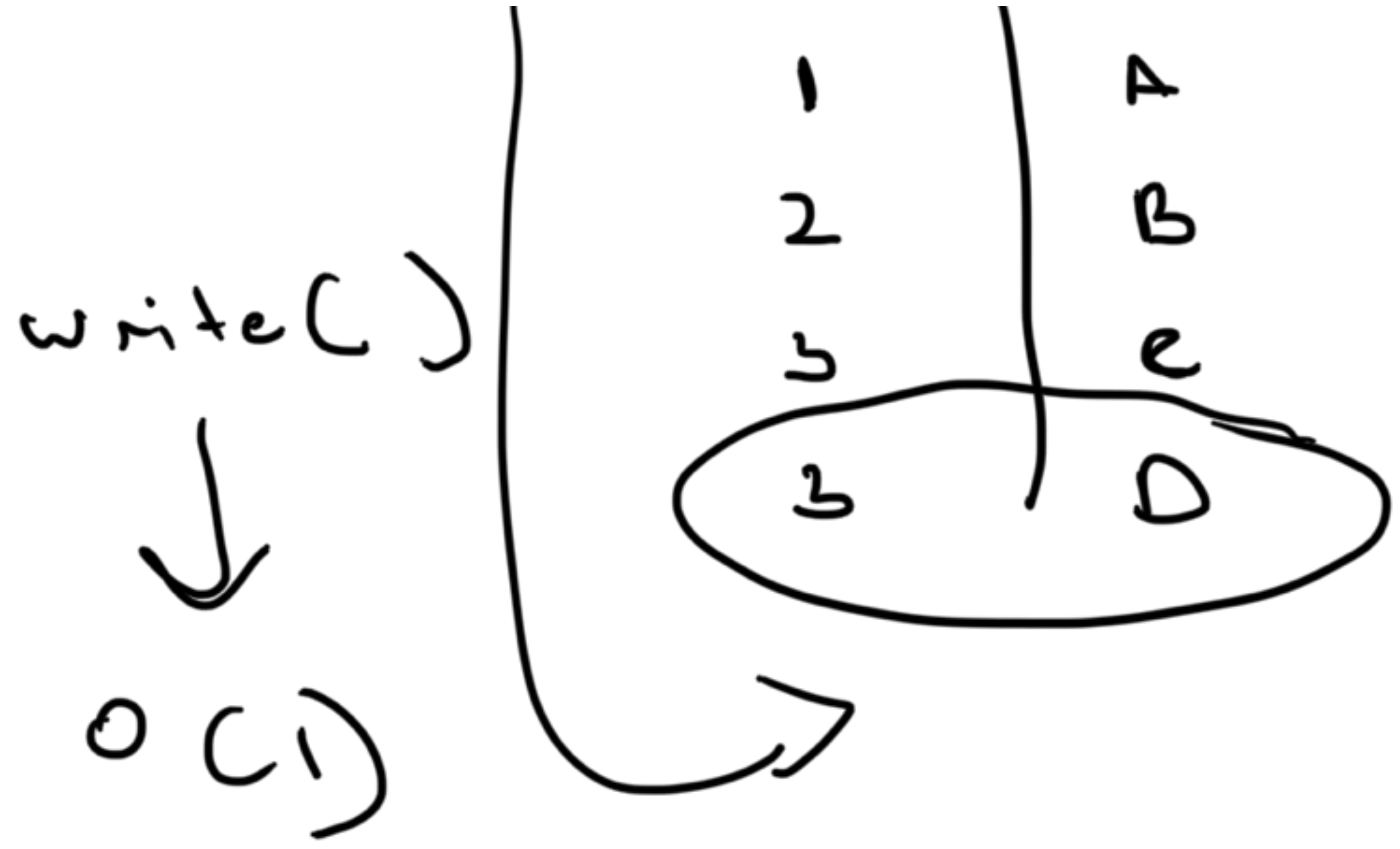
read (x) → O (N)
update (x, y)

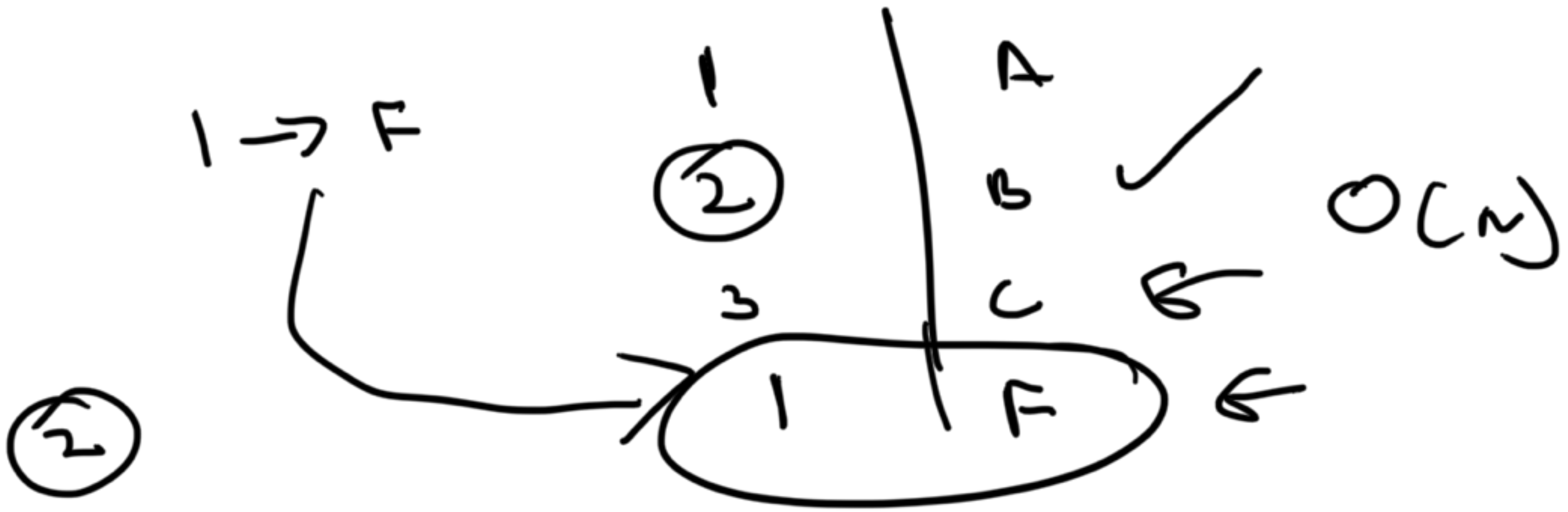


Solution 2 - O(1)

Write Ahead logging
- bin log





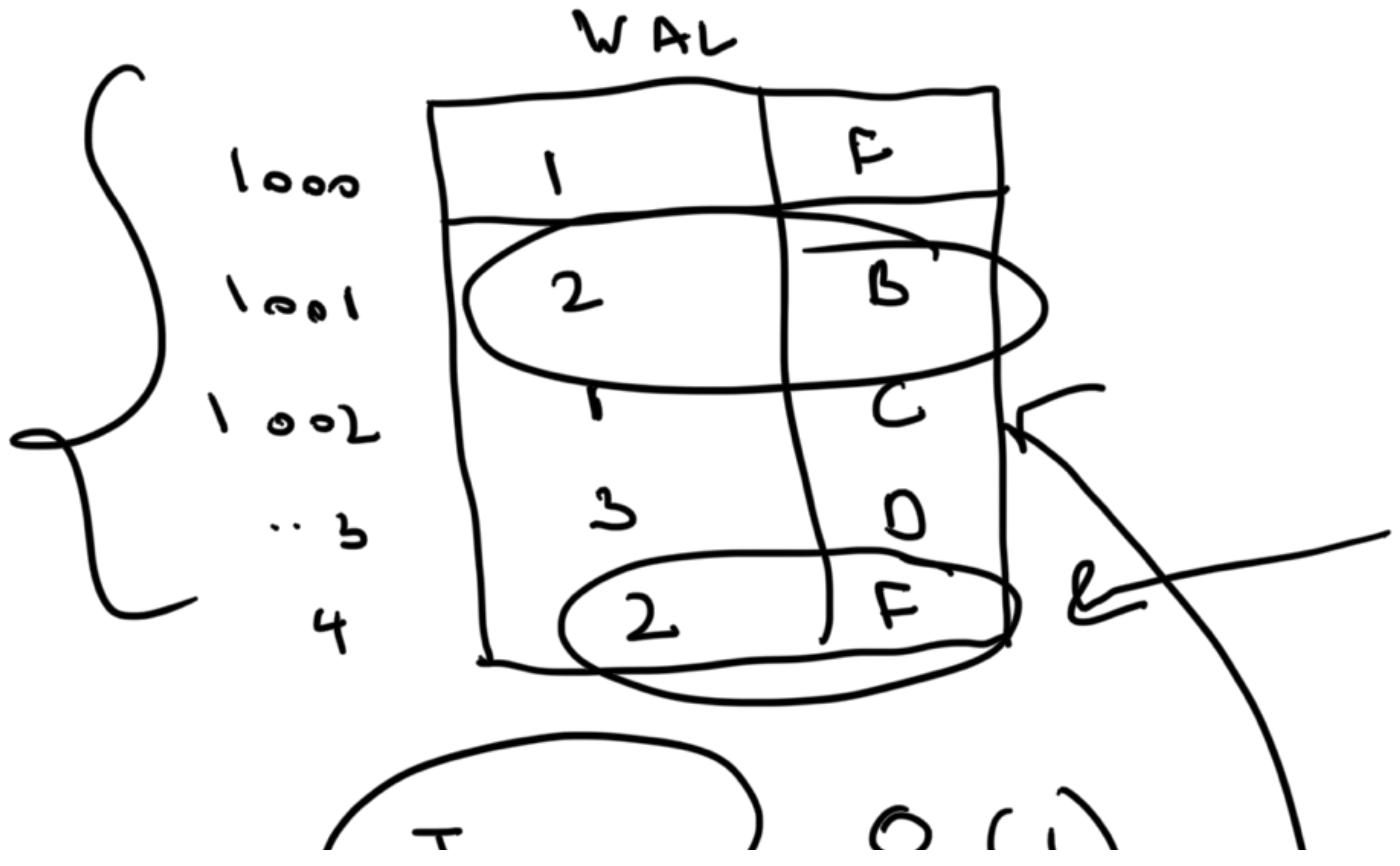


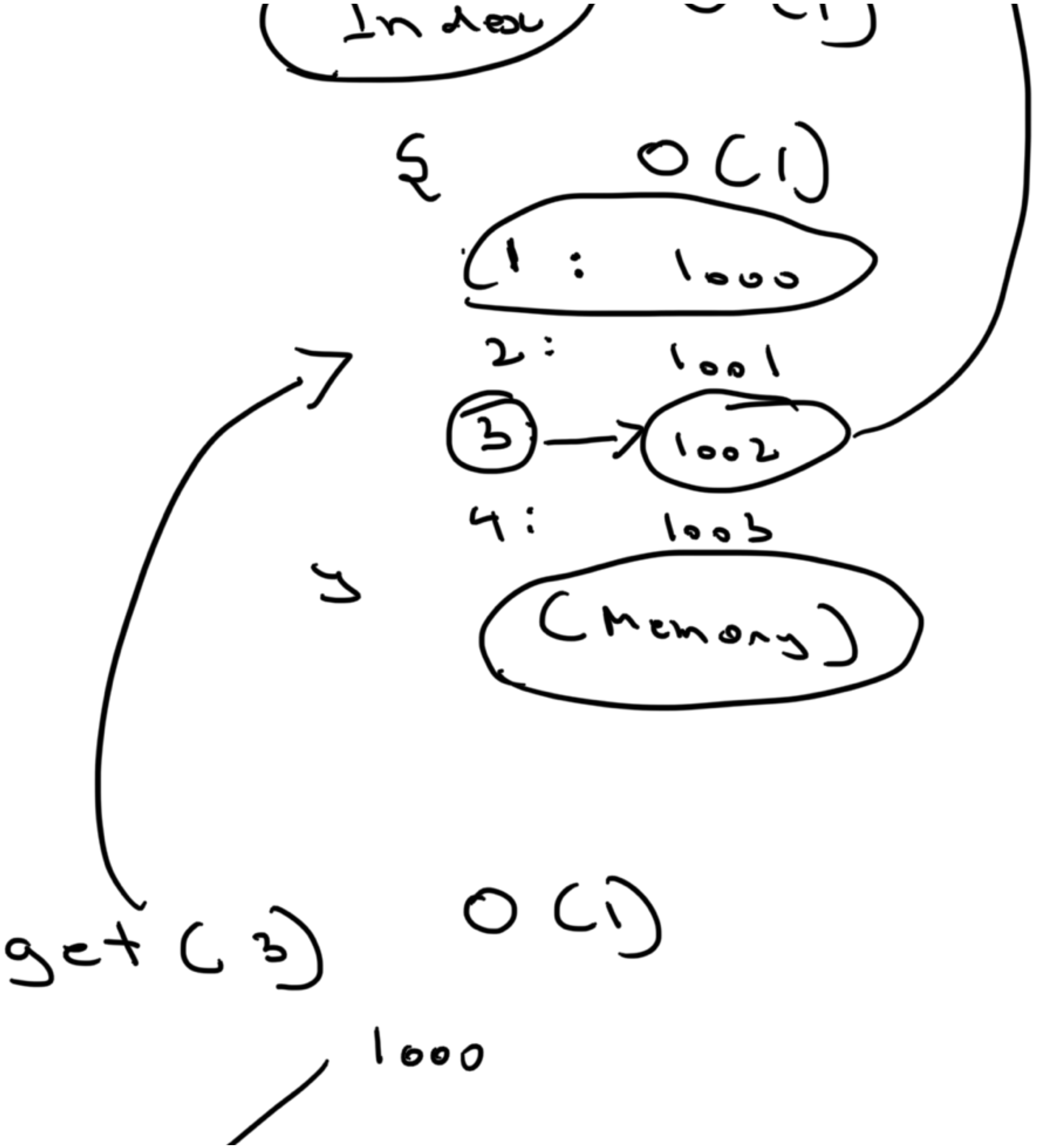
① Read complexity - O(N)

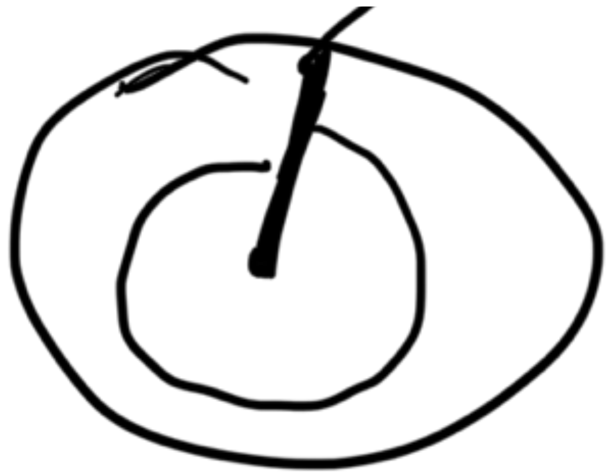
② Duplicate data

Solution 3 - read O(1)

$\boxed{\text{WAL + index}}$




Index
O(1)
{
O(1)
1 : 1000
2 : 1001
3
1002
4 : 1003
}
(Memory)
get(3)
O(1)
1000



update (1 , F)

get

→ Index → Address → Disk

→ O(1)

update

→ update (1 , F)

→ Append

→ Update the index

RAM − $\frac{8\ GB}{256\ GB}$ $\frac{32}{5\ TB}$
Disk −

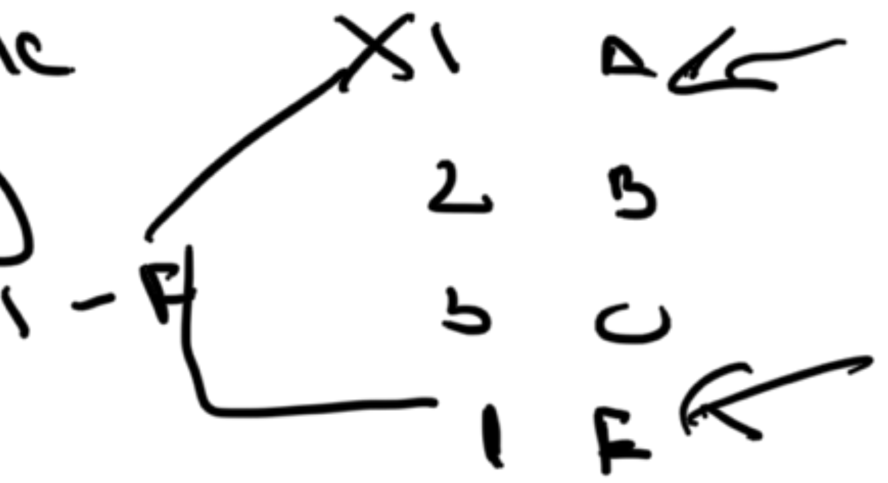
1 Trillion − 8 B + 8 B
$\boxed{16}$ −

$\boxed{1}$ Flat file

read + write - O(N)

② WAL

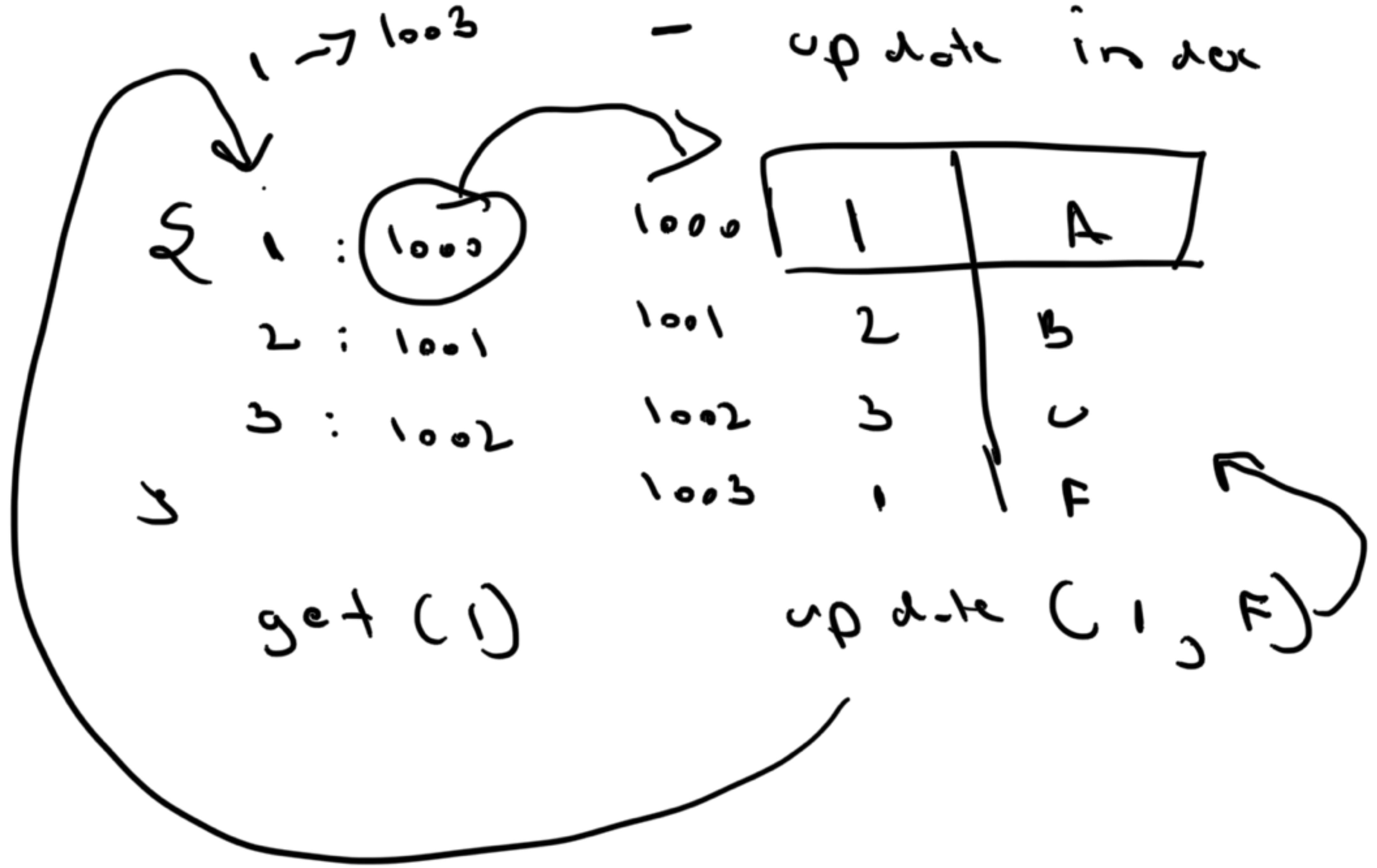
- append to the end
- immutable
- write - O(1)
- read - O(N)



③ WAL + index

- memory index
  - key to address
- get( ) - index
  - address
  - seek
- update

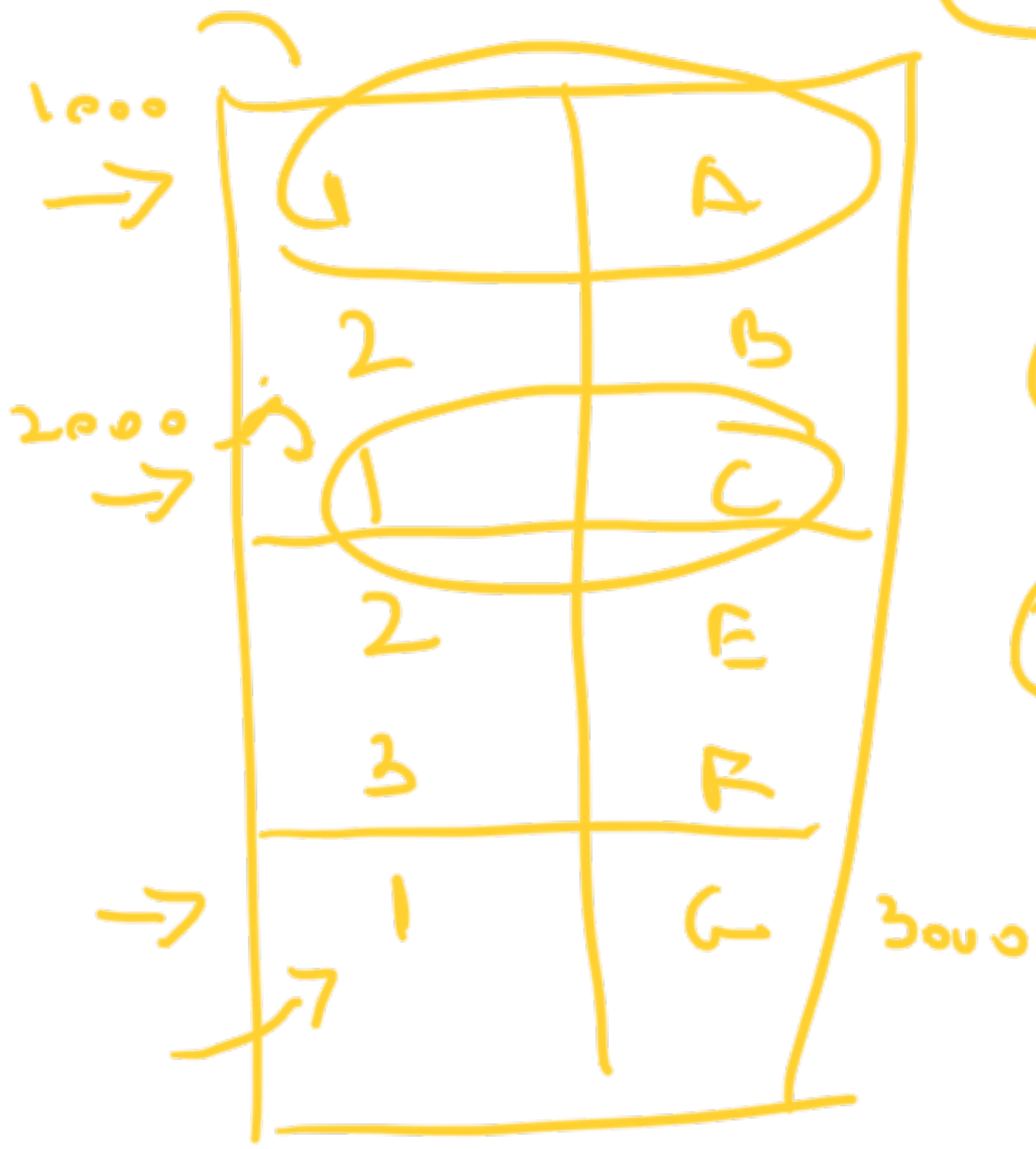
- append to the end
- update index



Solution 4

- Async process

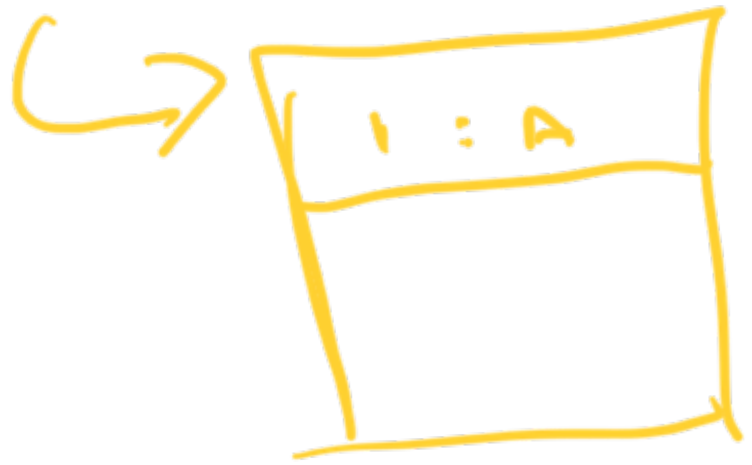
Compaction

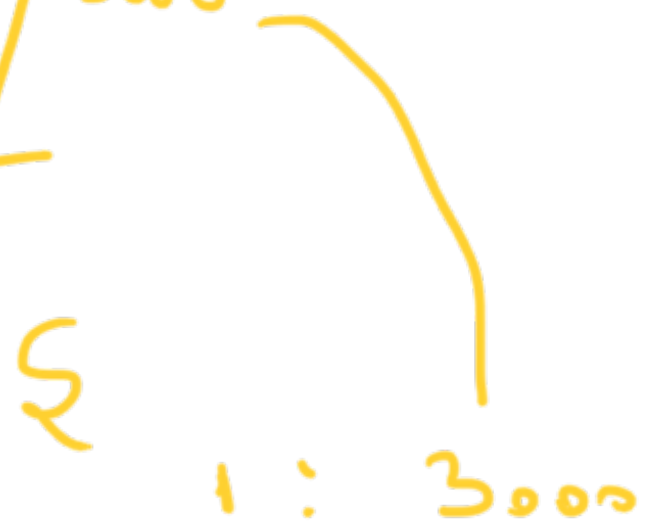


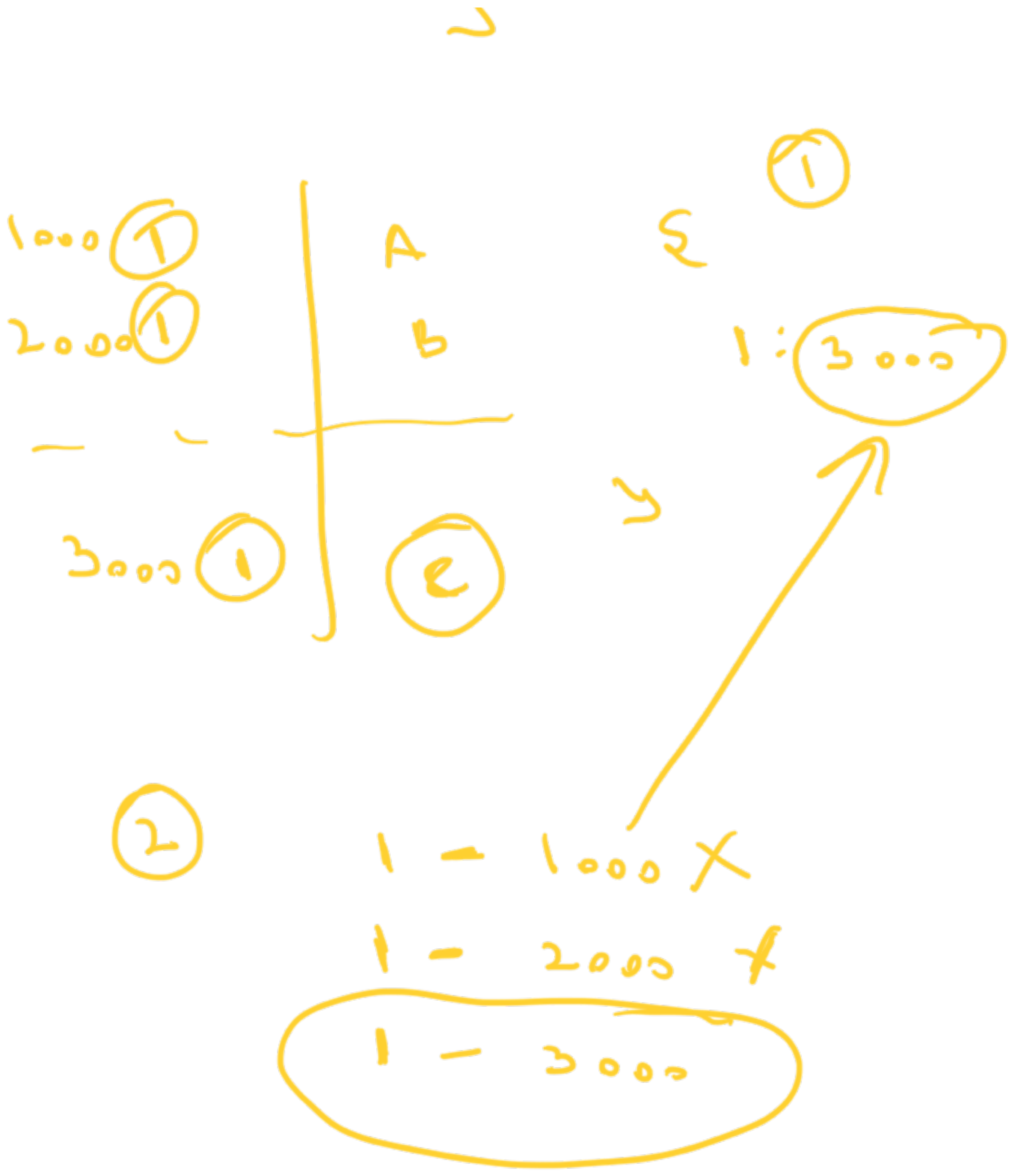
1. Periodic
2. Read the file
   - chunkify
3. De-duplication

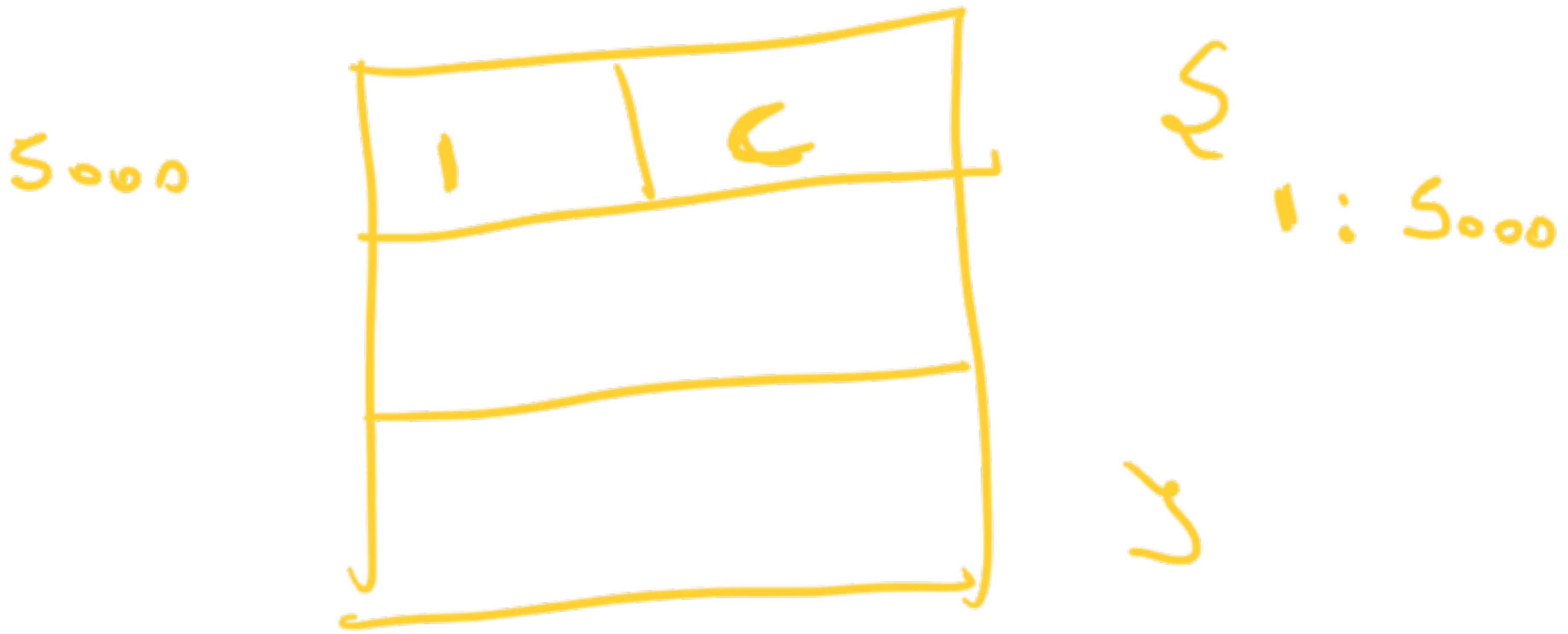
if (address != index)

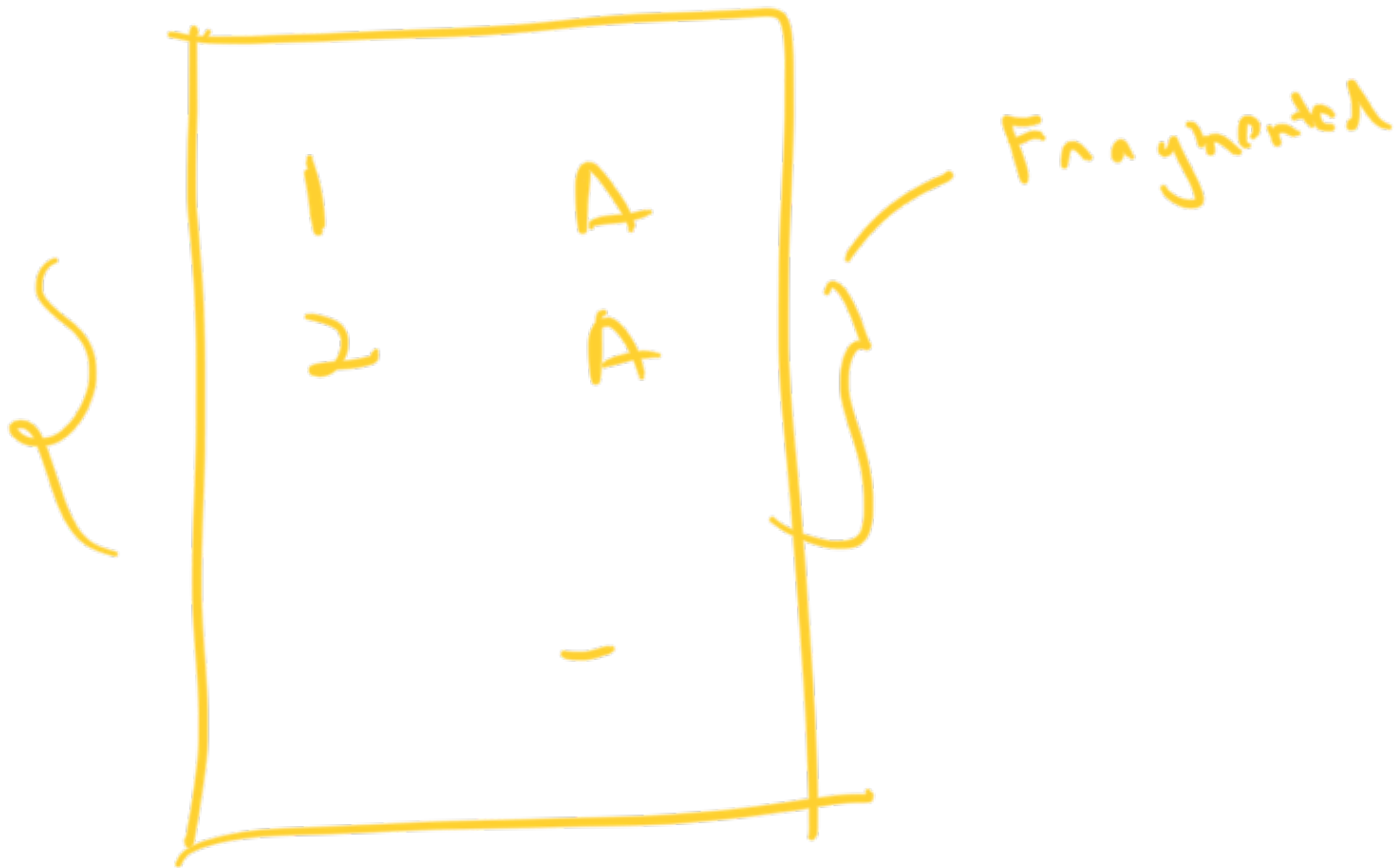
store






1000 1
2000 1
A
b
3000 1
e
1
1 : 3000
2
1 - 1000 X
1 - 2000 X
1 - 3000

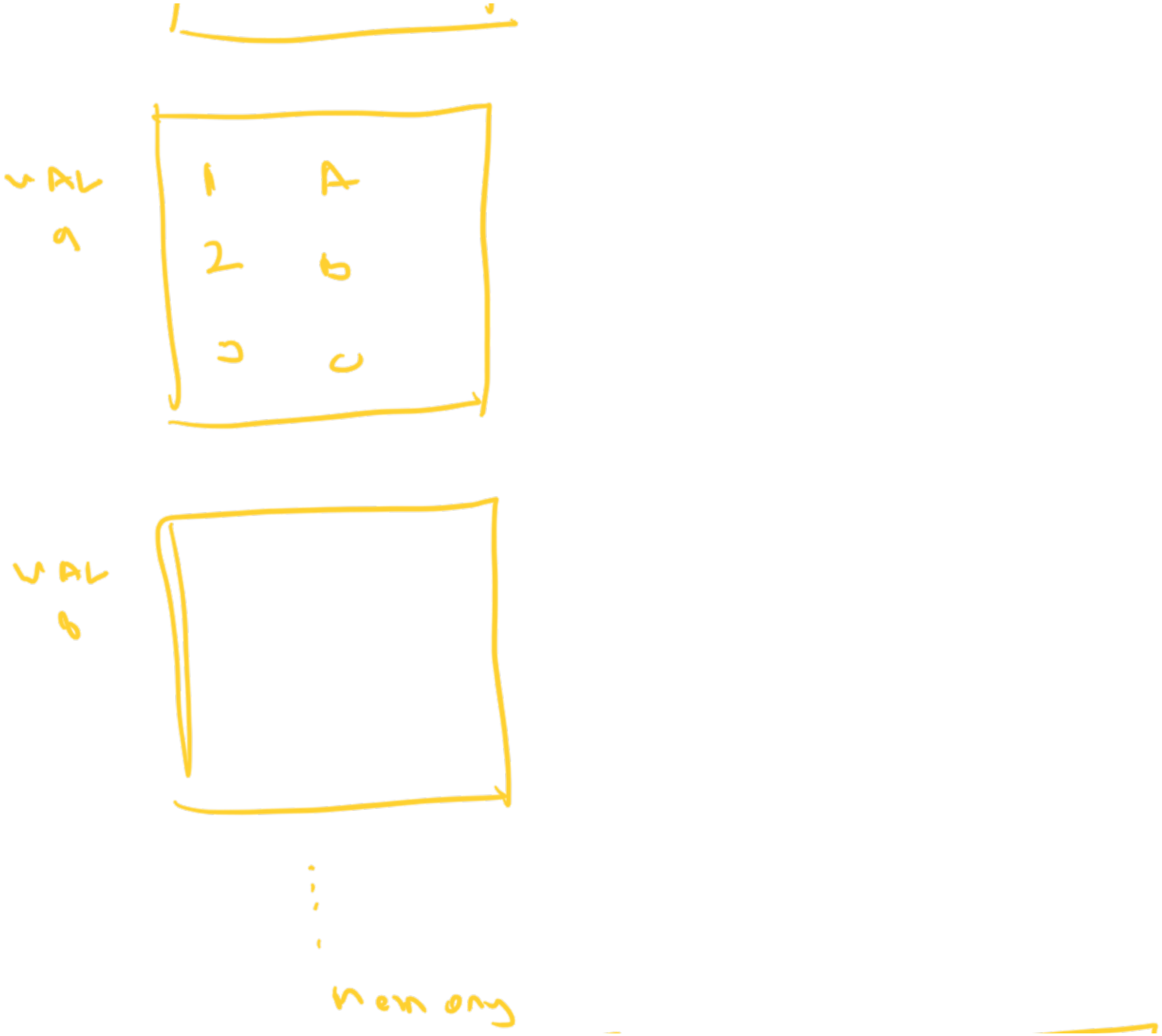

5000
1
C
1 : 5000

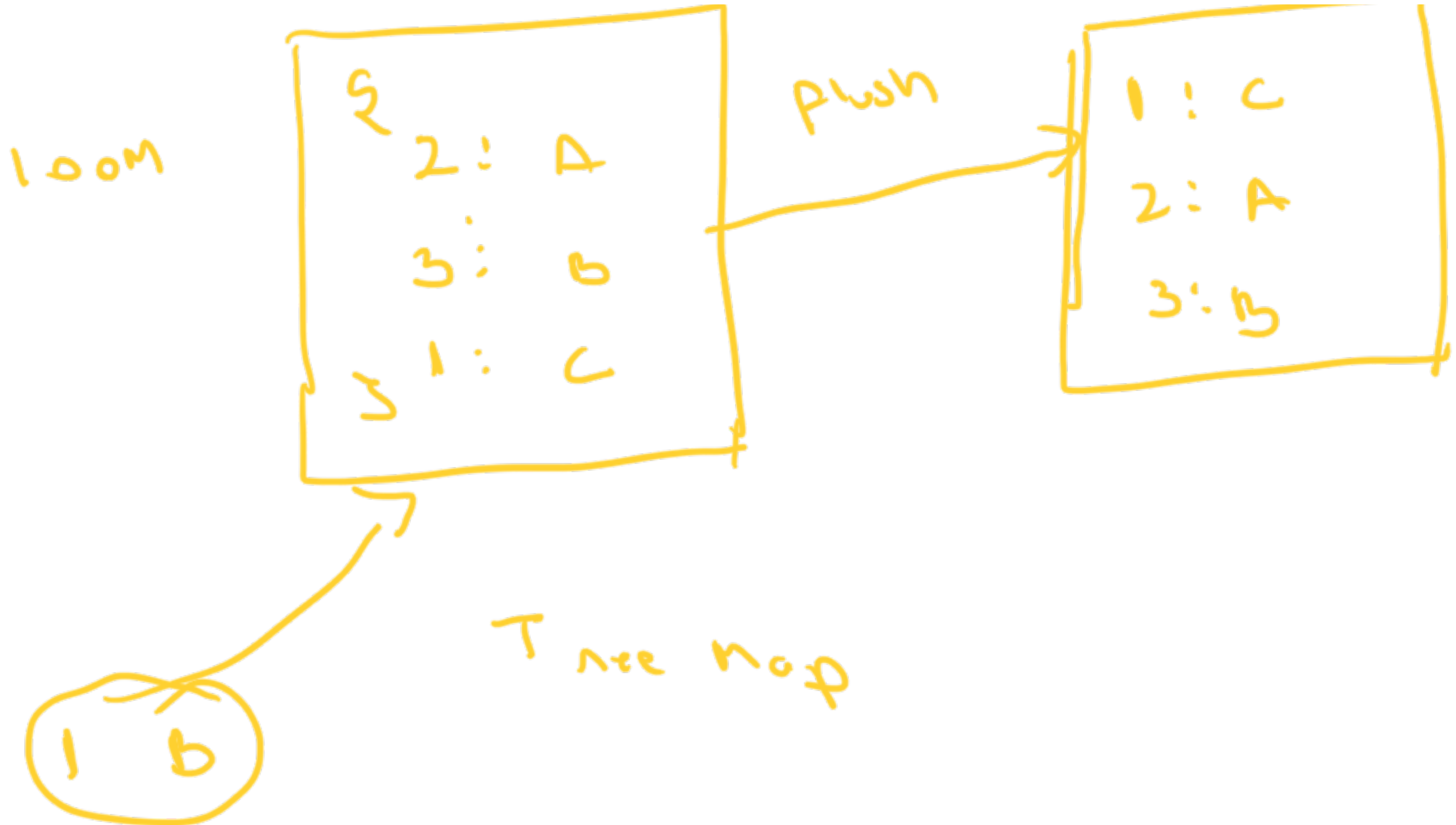

1 A
2 A
Fragmented

BREAK

5:54 - 6:00
- 10:30

Solution 5


val a
1 A
2 b
3 c
val b
memory

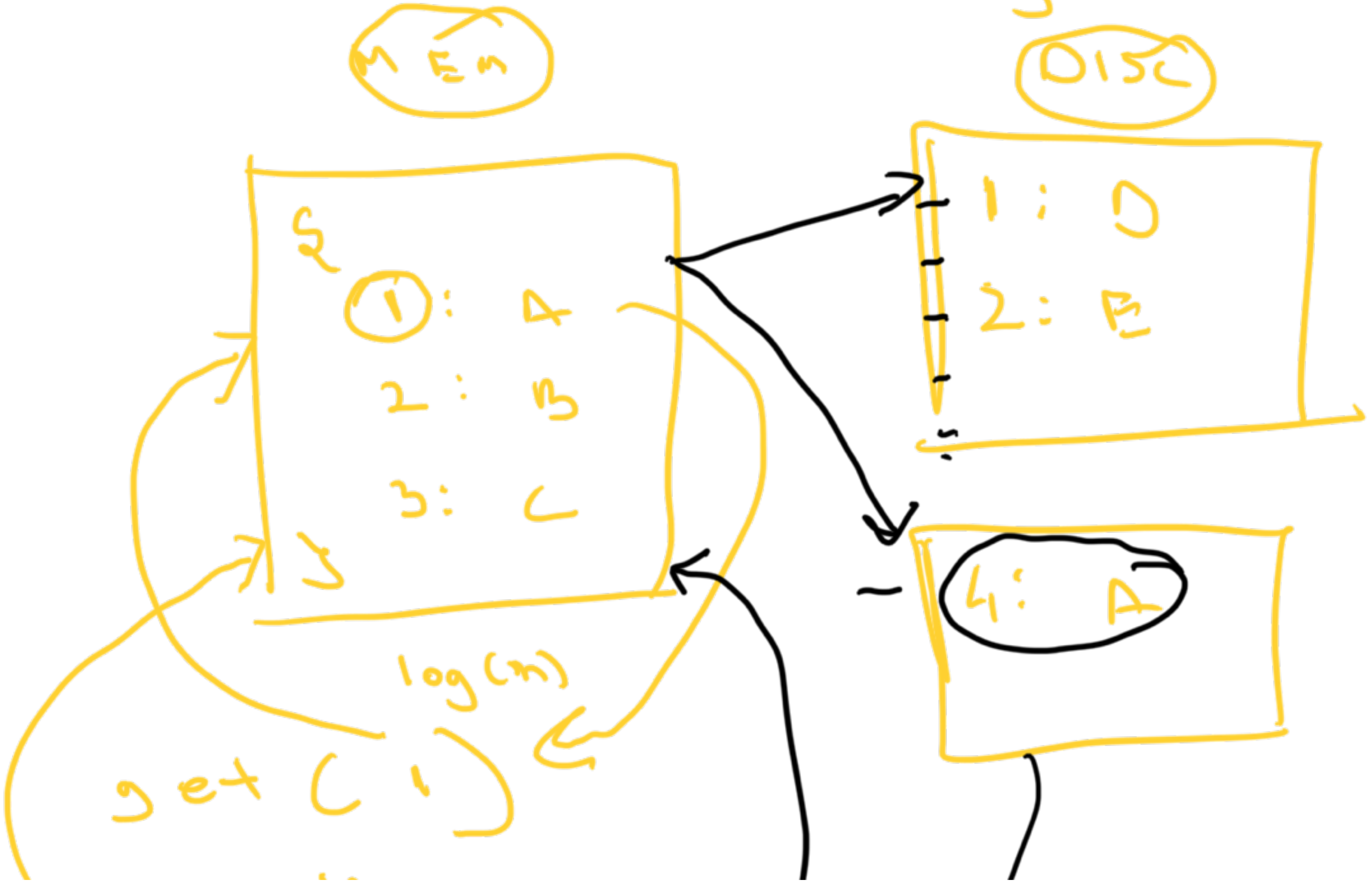


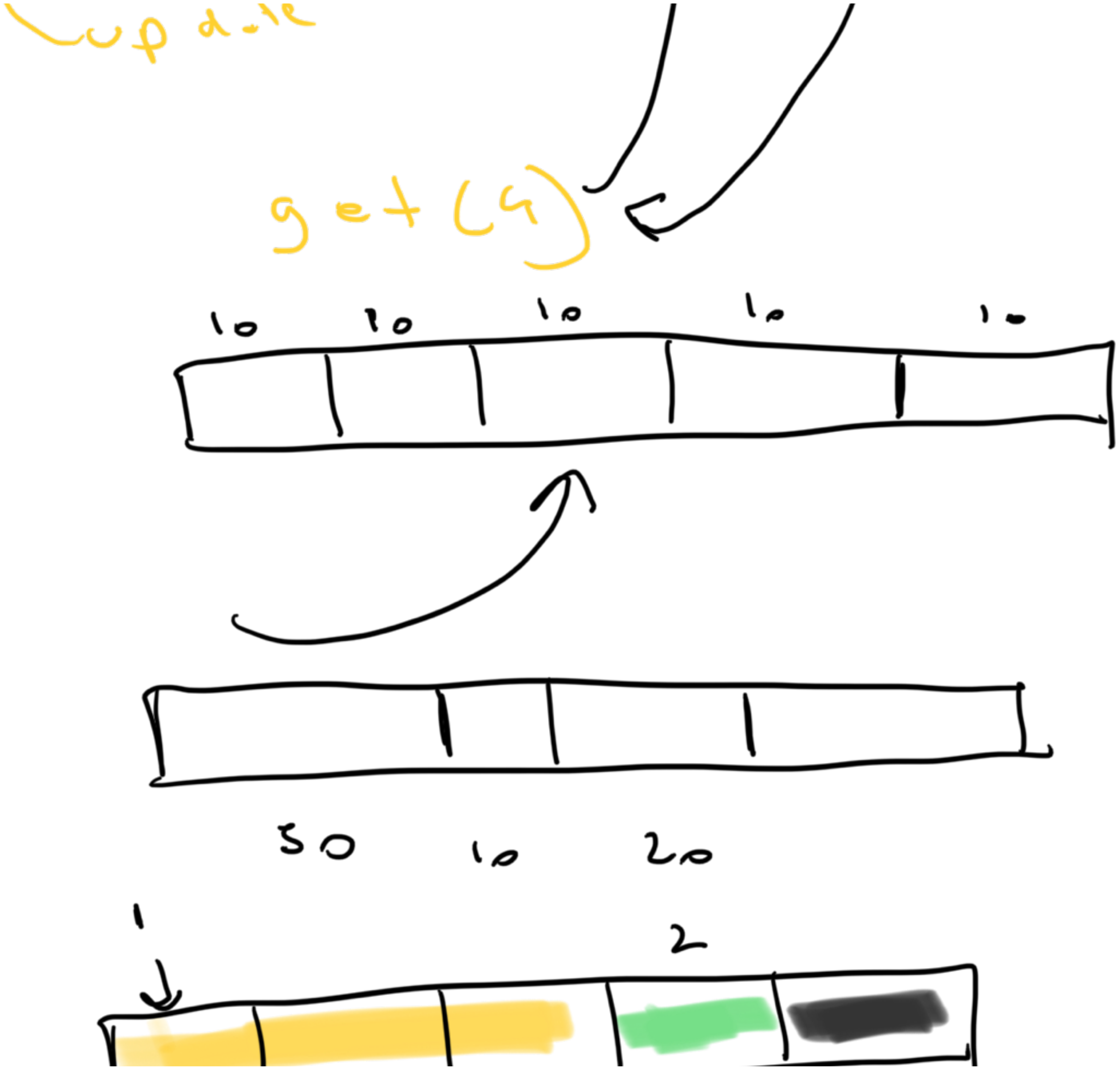
Update

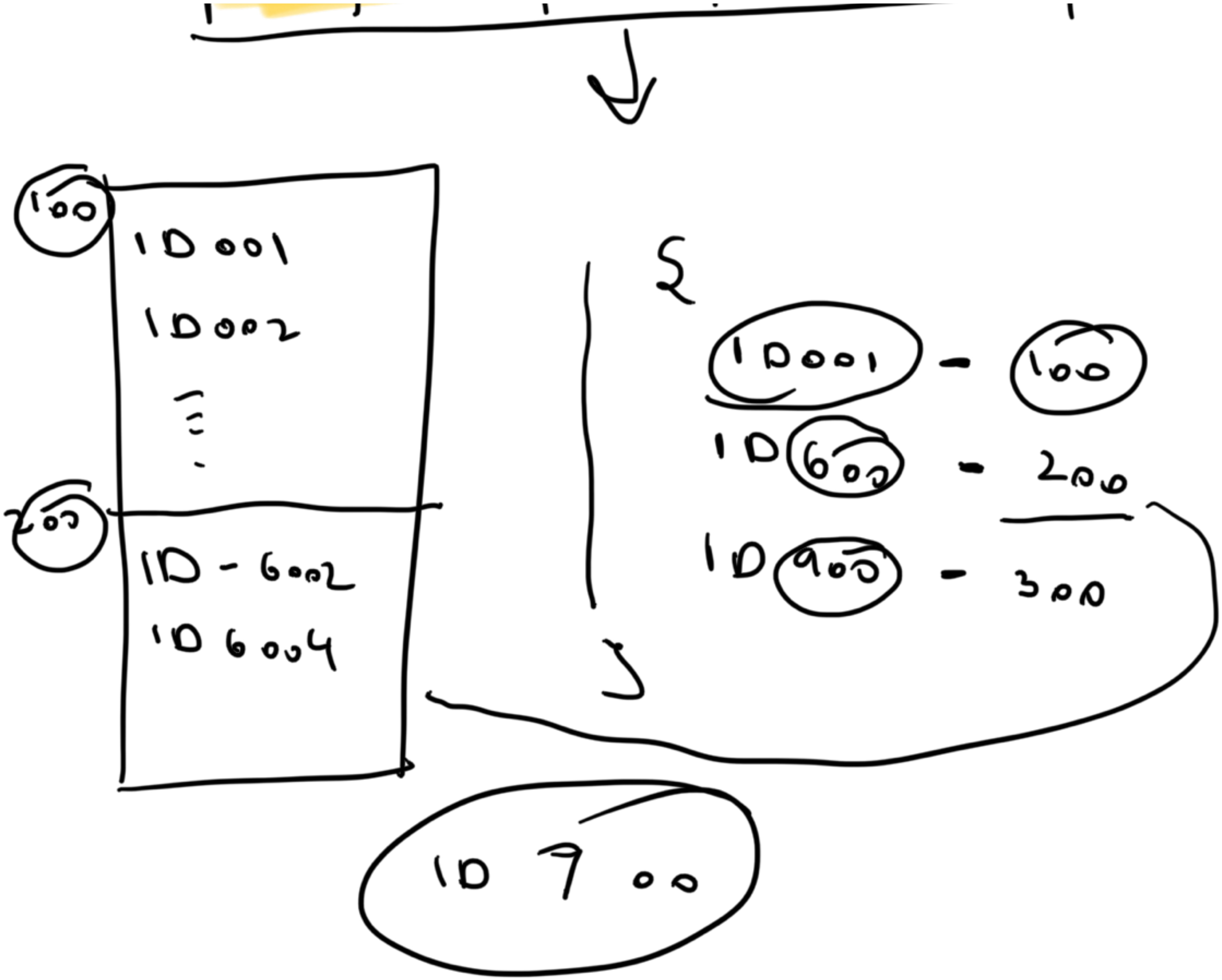
- O(log n)

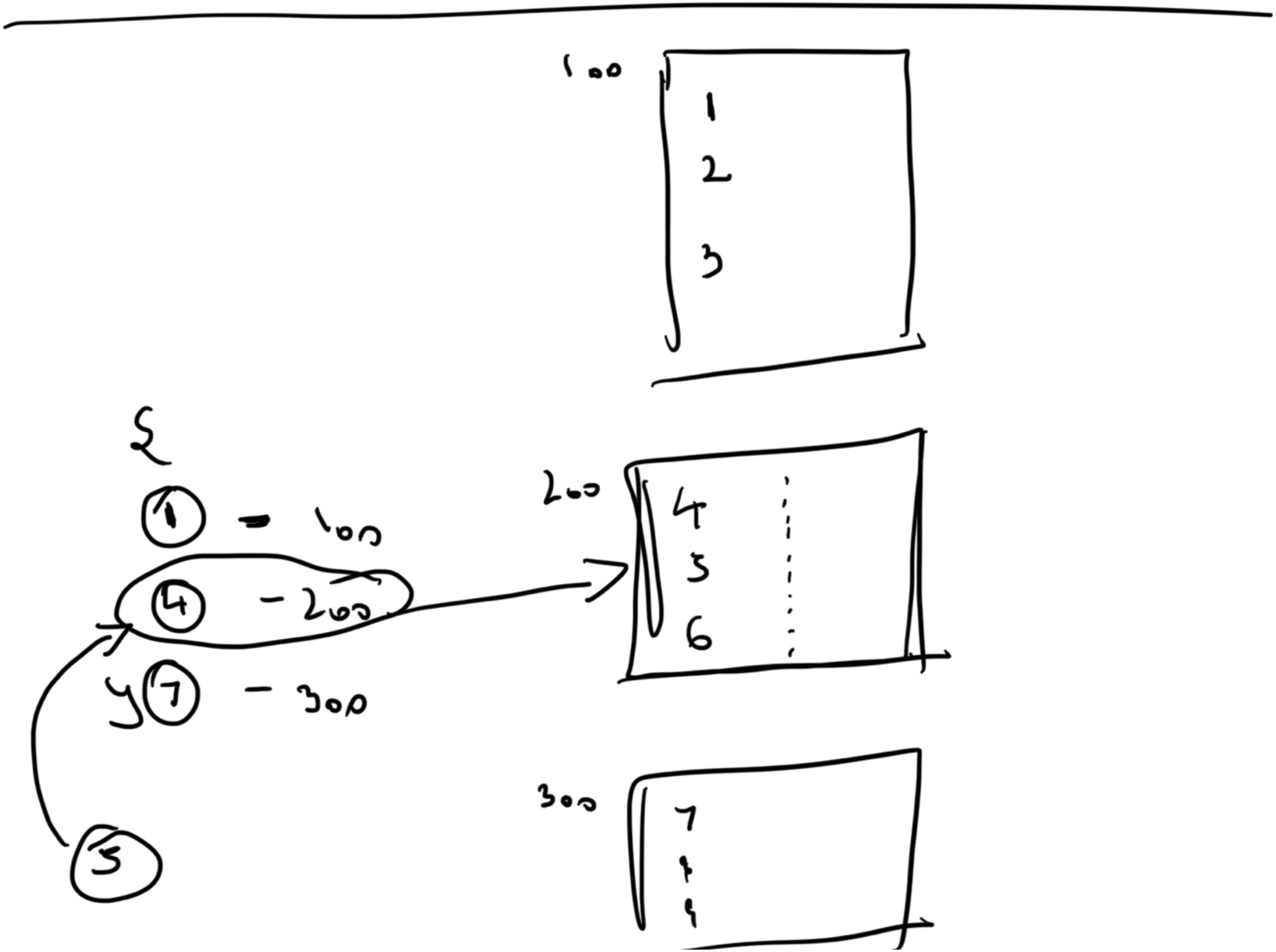
Get

- if key in memory
    - log(n)
- if the key is not in memory




update
get(4)
10
10
10
10
10
50
10
20
1
2


100
ID001
ID002
200
ID-6002
ID6004
{
ID001 - 100
ID600 - 200
ID900 - 300
}
ID700


100
1
2
3
200
4
5
6
300
7
{
1 - 100
4 - 200
7 - 300
}
5

LSM

- Log structured merge tree

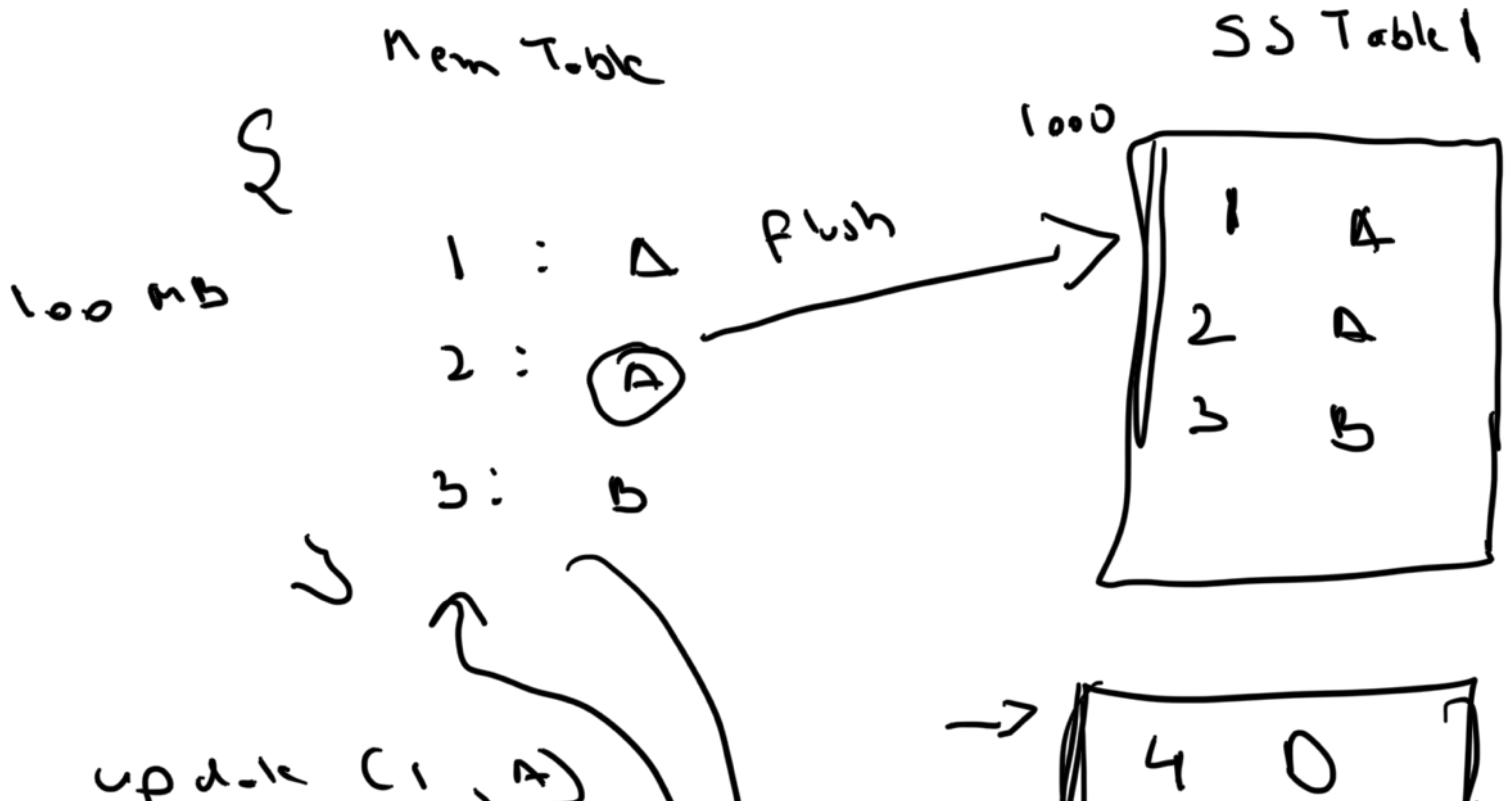
1. Tree map = Latest data
   - Memory RAM
   (MemTable) - 100 MB

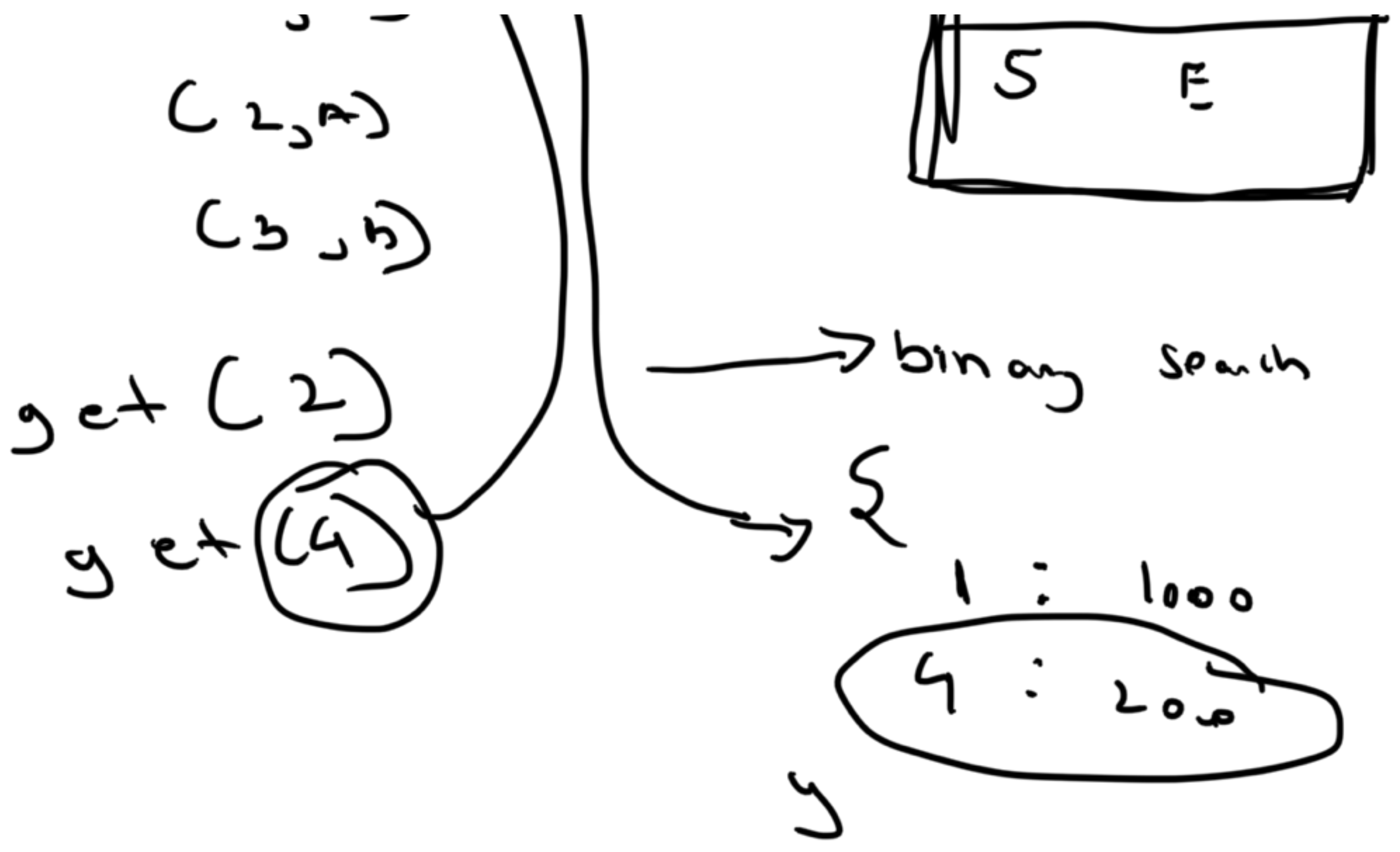
2. Chunks of WAL file
   (SS Table)
   ↓

Sorted String

(3) Index - memory

ID - Start of that





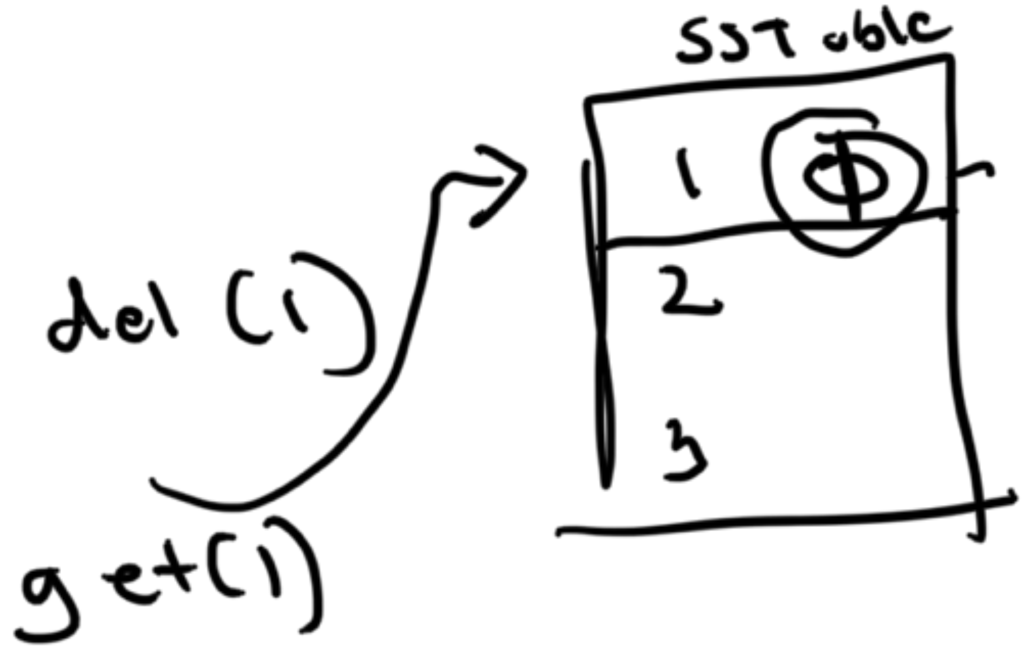
Volatile Memory - MemTable ?

→ WAL file

② Delete

→ Sof delete

→ Tomb storing



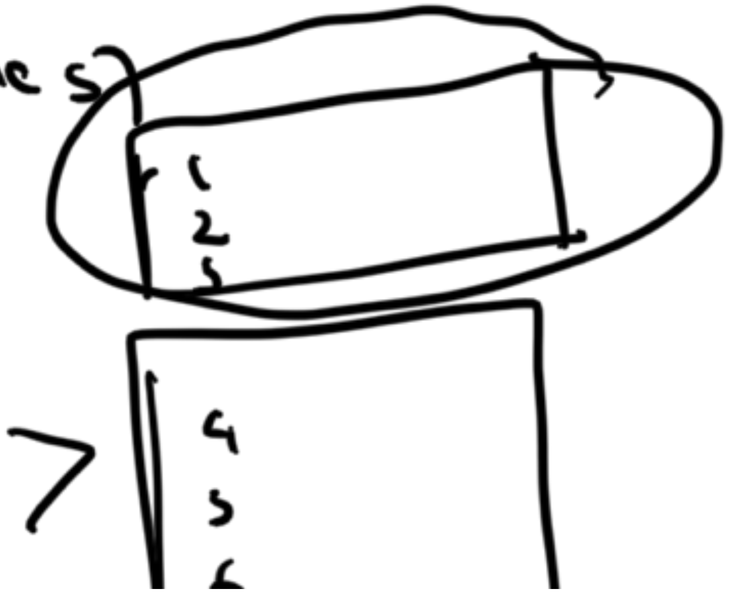
not present

③ Can a key be part of multiple SS Tables?

100

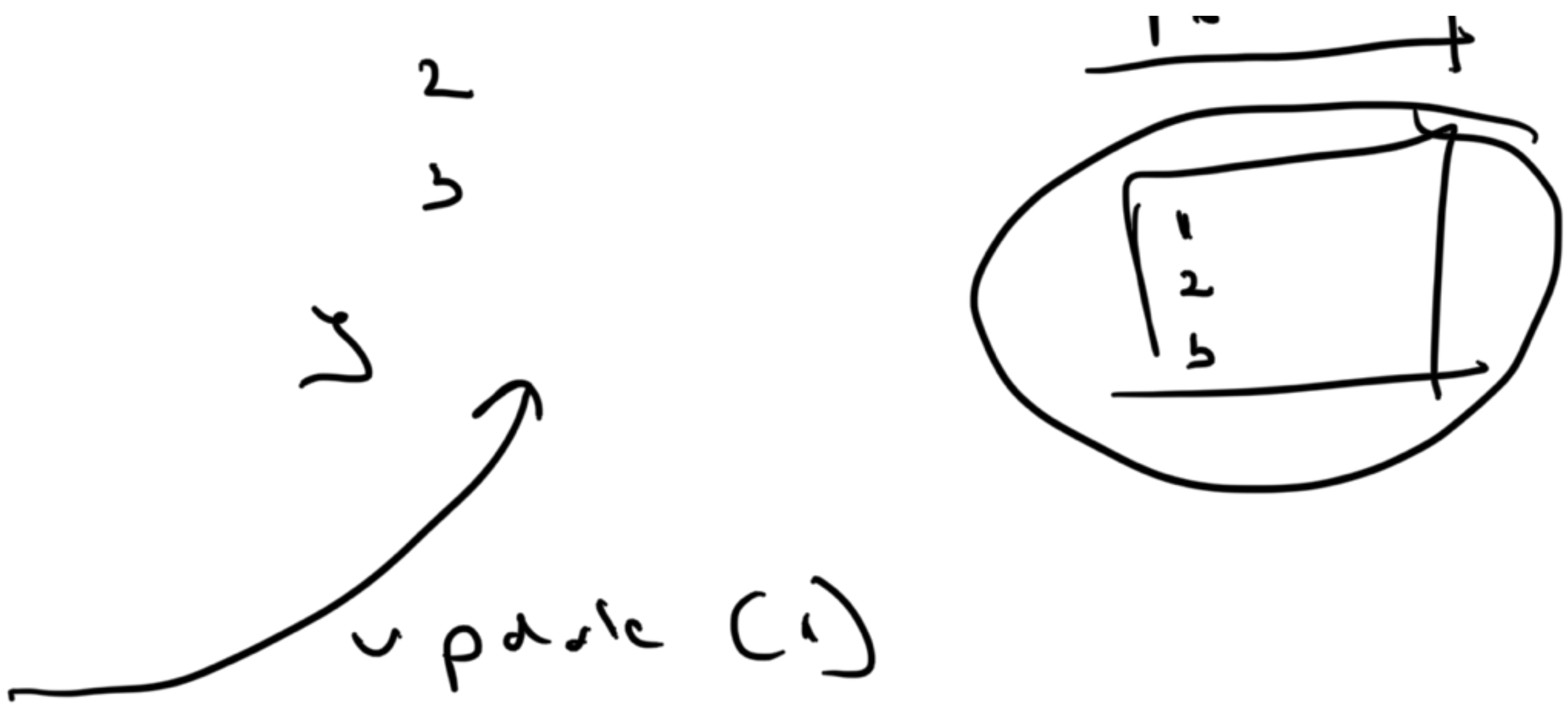
{

1



2

3

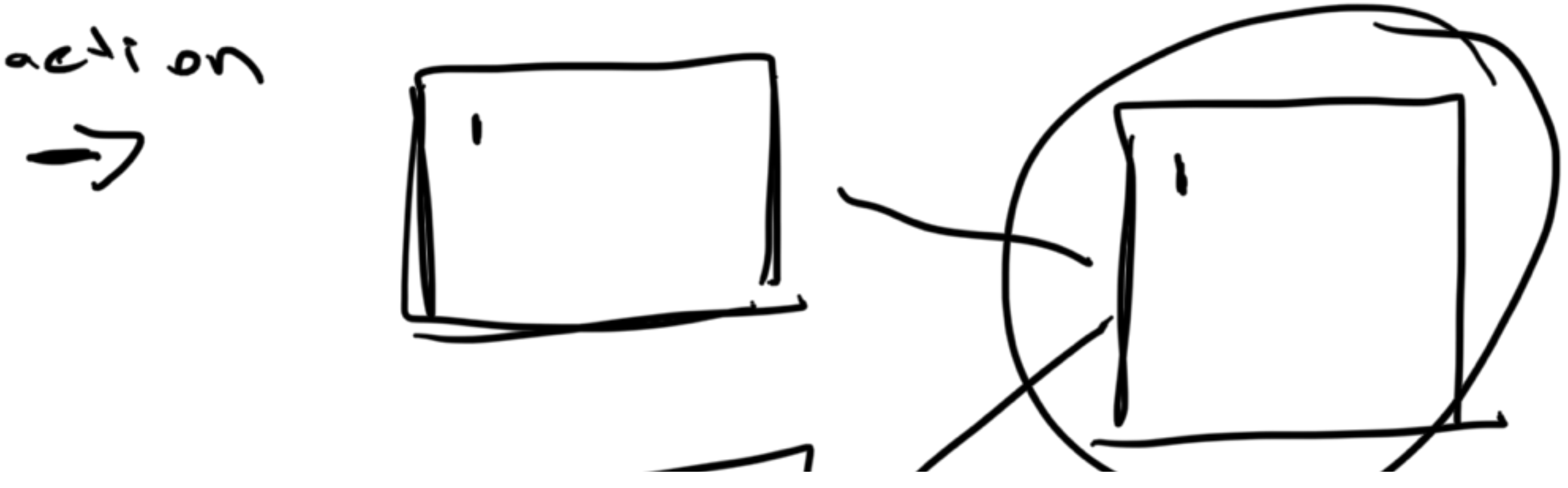


update (1)

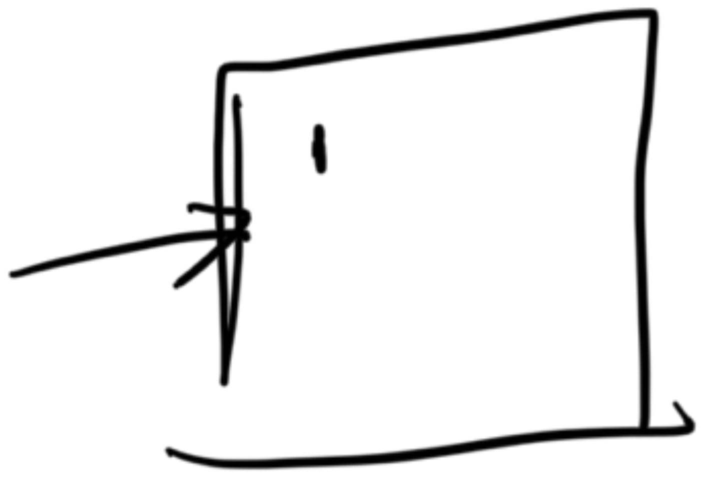
(2)

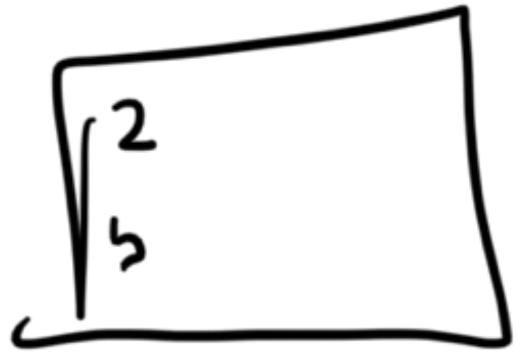
(3)

compaction



Search for key across chunks



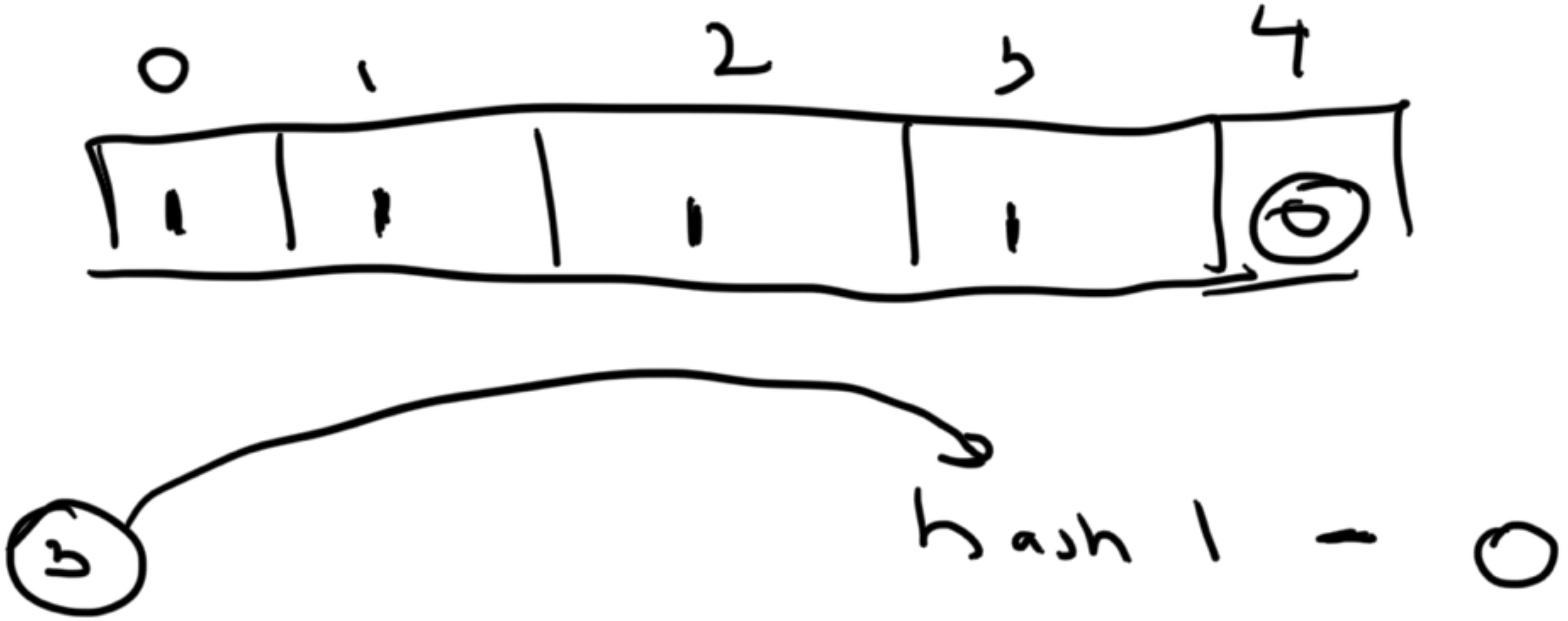


Bloom filter

does A Key Exist ()

False → Does not exist

True → May exist



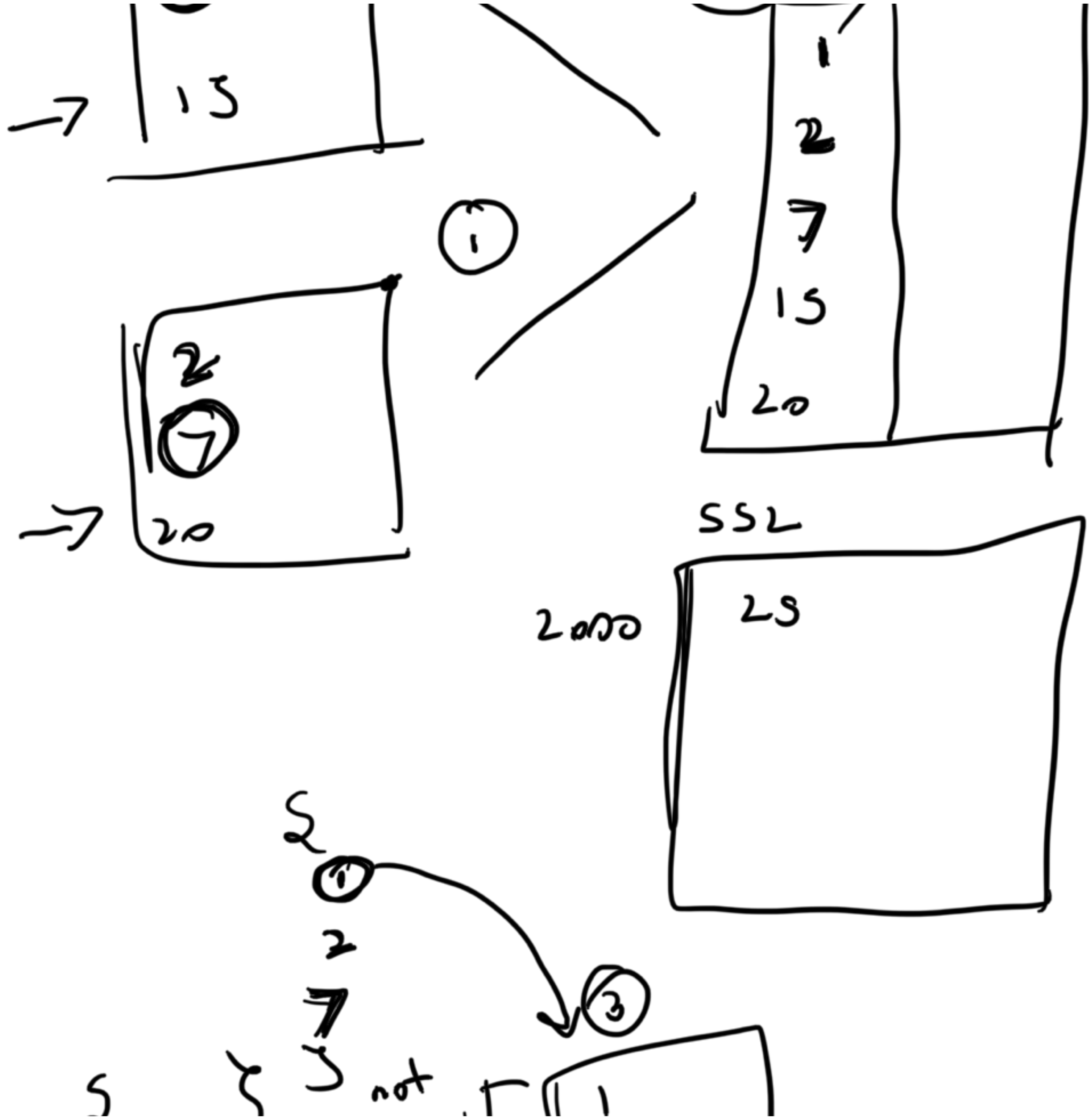
$\longrightarrow$ hash 2 - $\textcircled{2}$

$\textcircled{4}$ h1 - 1

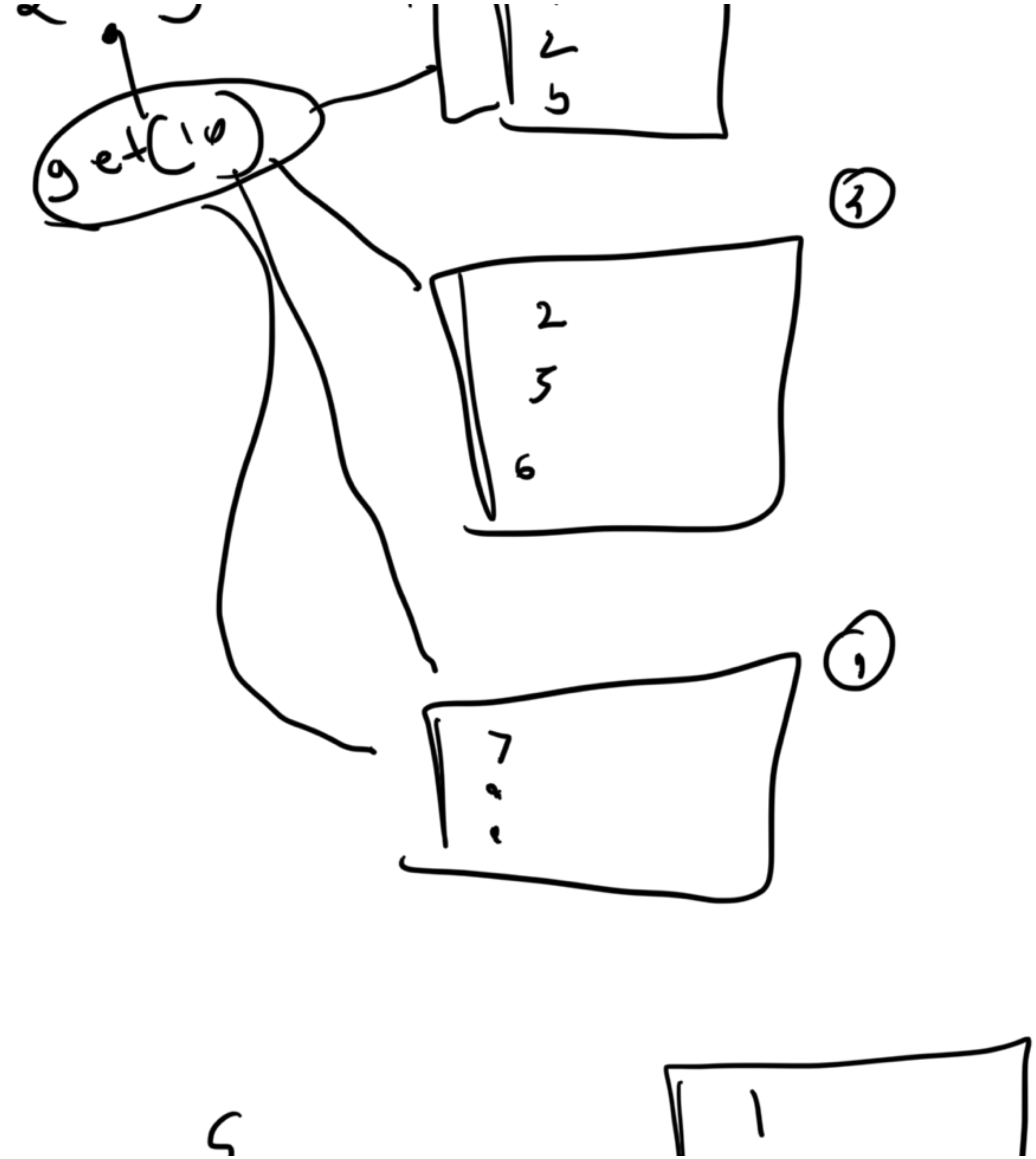
h2 - 3

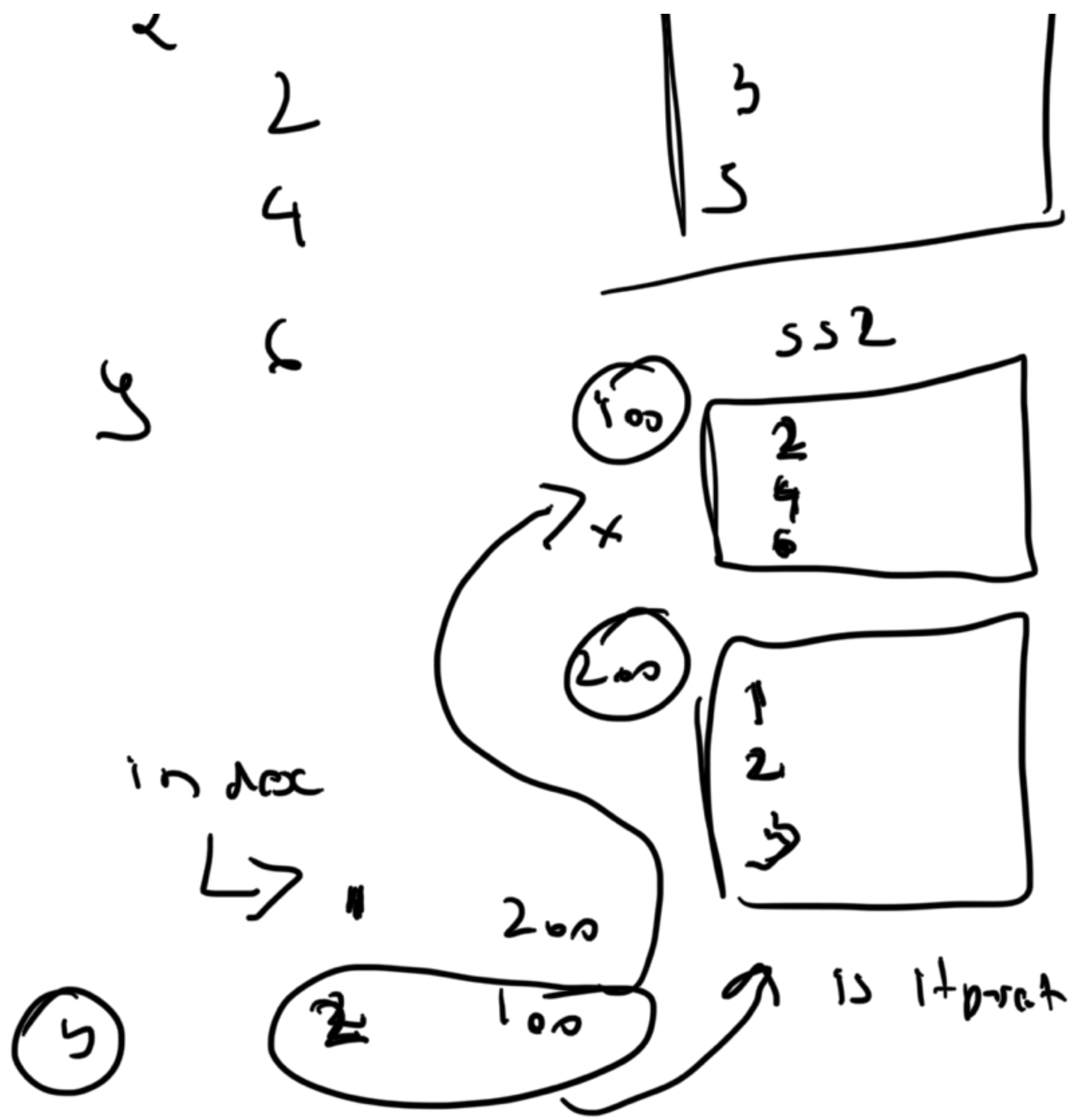
$\textcircled{5}$ h1 $\textcircled{1}$ ✓

h2 $\textcircled{4}$ 0 ✗



15
1
2
7
20
1
2
7
15
20
SSL
2000
25
S
1
2
7
3
3
not
S
1


get(10)
2
5
3
2
3
6
9
7
1
S


2
2
4
6
5
3
5
ss2
100
2
4
6
200
1
2
3
index
1
200
2
100
5
is ltgreat

1	A
2	B
3	C
1	E
2	D

$\{1 : \boxed{1000},\ 2 : 1001,\ 3 : 1002\}$

$\{\ \boxed{1 \quad 3}\ \}$