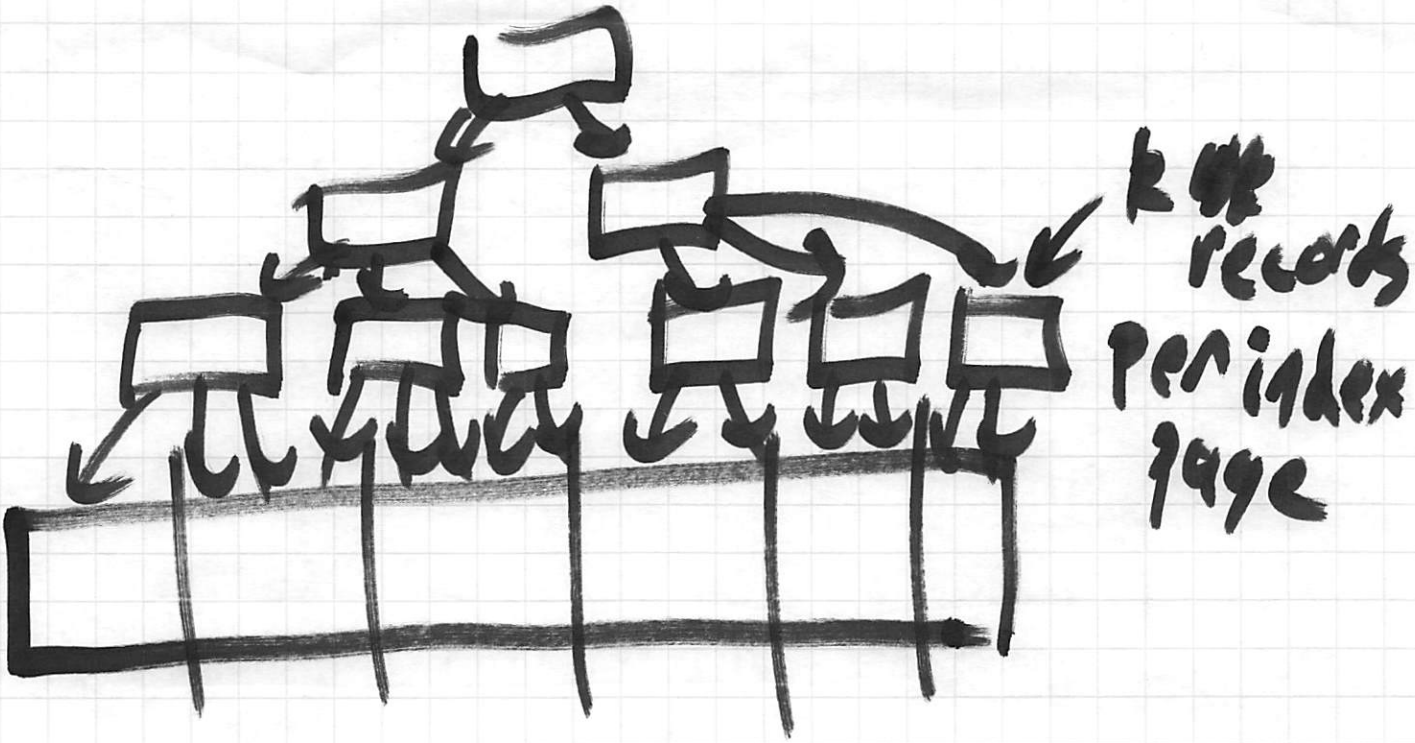
$$\log_k(N) = \frac{\log_2(N)}{\log_2(k)}$$

Index improves by const multiple



N ~~records~~ pages

Naive Binary Search

$O(\log_2 N)$

w/ Index Pages

$O(\log_k N)$

Idea 1: 2 (or more) separate copies of the index

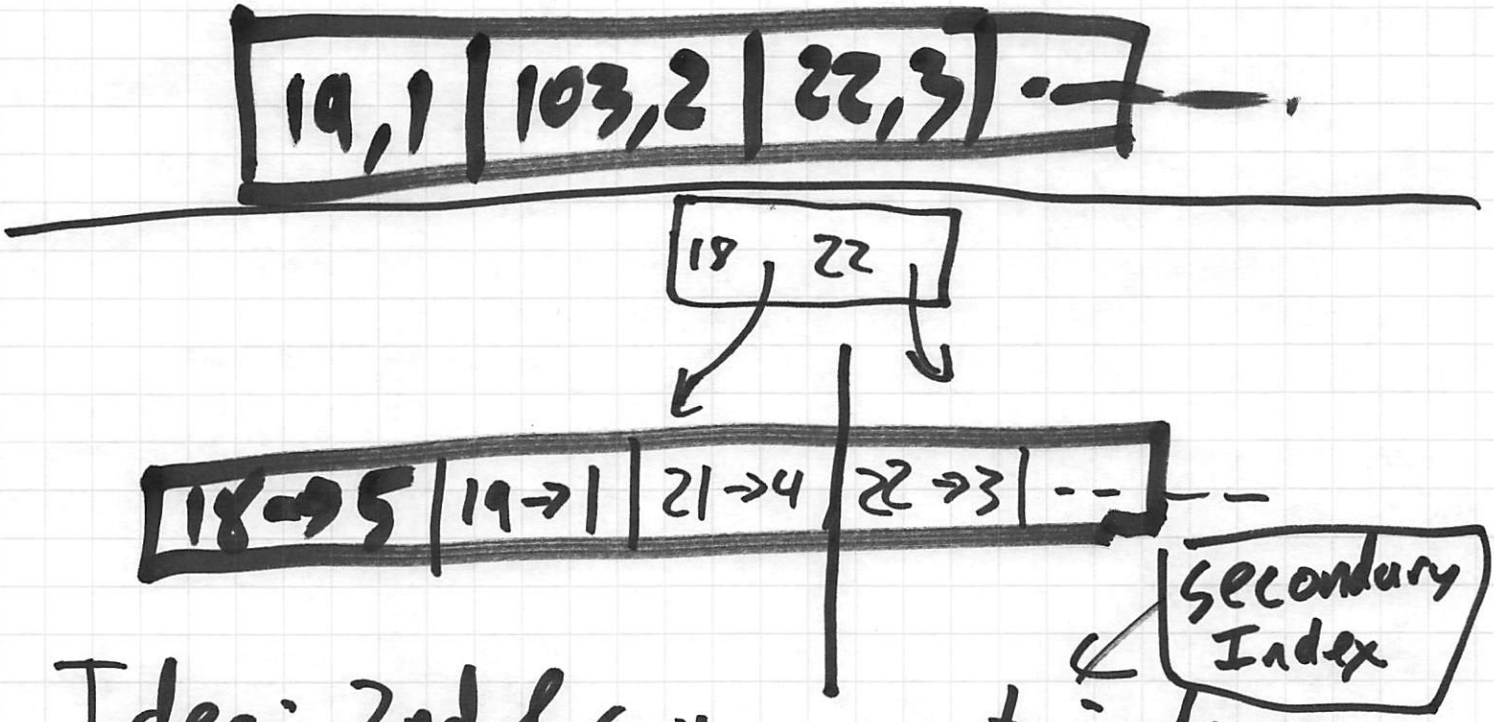
pro: easy

con: 2 copies of all data

Idea 2: Hierarchical sort

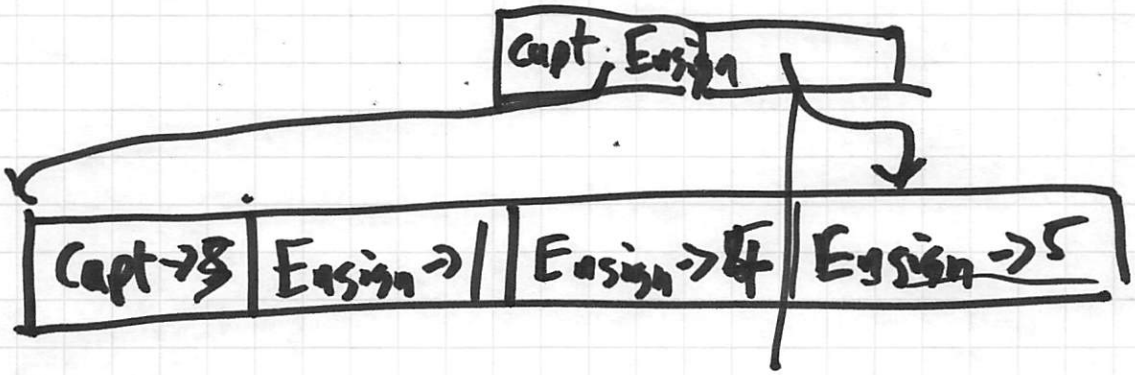
pro: Saves space

con: limited query support



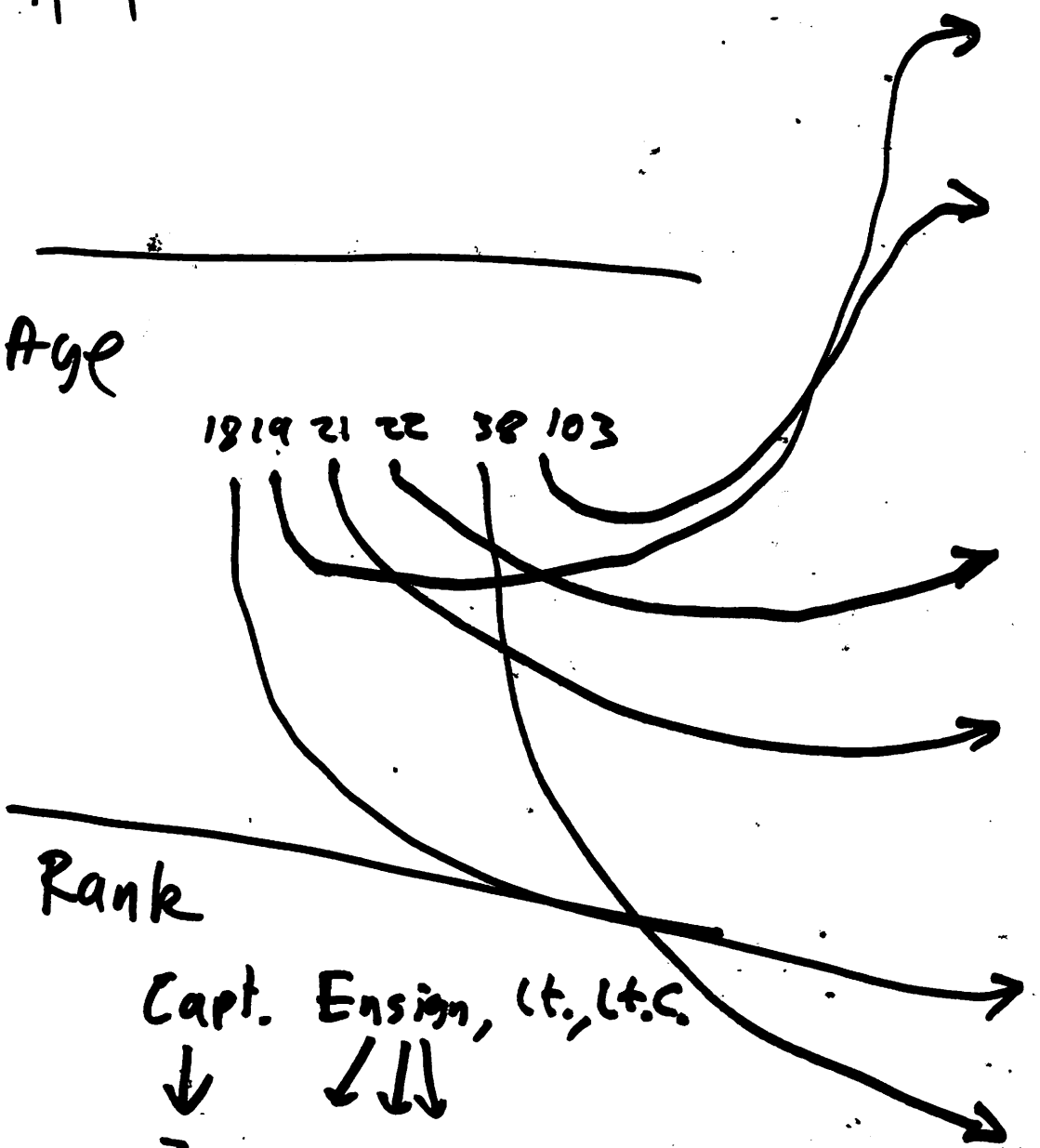
Idea: 2nd & subsequent indexes Store only unique Id of records

Pro: Less Space (so may be faster)



Con:

#4



One Index for each attr

Easy

Space Hog

Hierarchical Indexes

Saves space

Still Easy

Limited Query Support

Secondary Indexes

~~Saves~~ Saves space

~~can be~~ can be faster

Often Need to load more pages

Given a query for rank = _ ∧ age = __

pro con

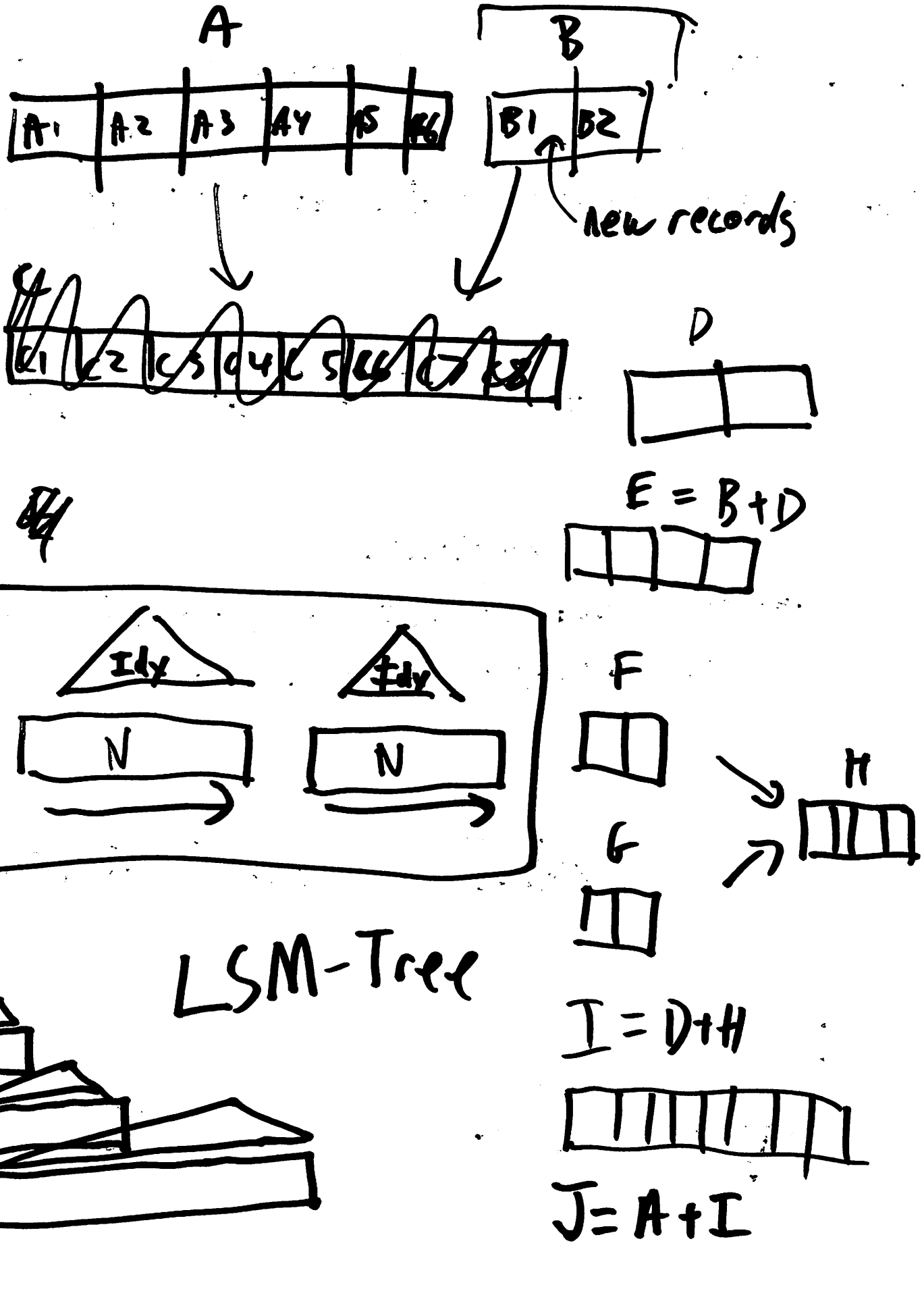
Option 1: Pick an index (either rank or age)

Option 2: Scan the full data

Option 3: Find all identifiers for rank, for age
+ compute intersection

Option 4: Hierarchical Index

Question for Piazza


A
B
A1
A2
A3
A4
A5
A6
B1
B2
new records
C
C1
C2
C3
C4
C5
D
E = B+D
F
G
H
I = D+H
J = A+I
Idx
Idx
N
N
LSM-Tree