CSE 350

Advanced Data Structures

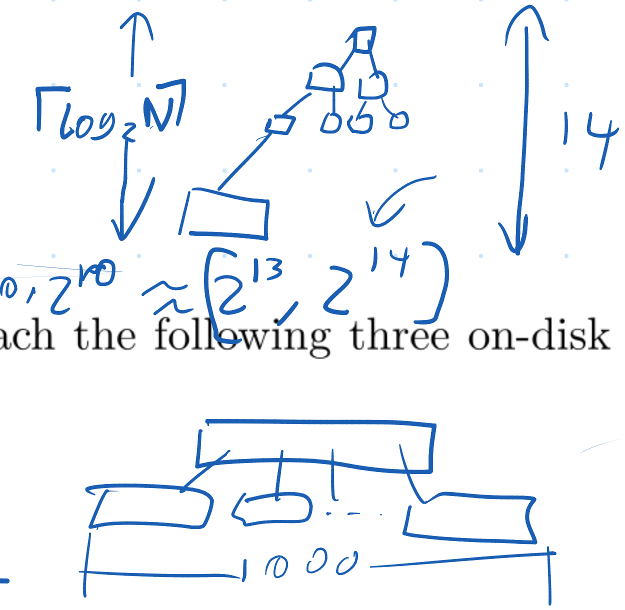
Topic 18: External Algorithms
(and midterm review)

Midterm Review

$10^3 = 1000 \approx 1024 = 2^{10}$ / K

$M\ 2^{20}$, $G\ 2^{30}$

$N = 10{,}000$ pages $\approx 10 \cdot 2^{10} \approx [2^{13}, 2^{14})$



Answer each question in this section with respect to **1,000,000** records stored in each the following three on-disk data structures:

- A sorted layout *accessed by binary search*. → up to 14 levels
- An ISAM index. → 2 levels + Data = 3 levels
- An on-disk Hash Table with one disk page per bucket. → 1 level

For all three on-disk data structures, assume that each disk page holds exactly 100 records. For the ISAM index, assume that each directory page holds 1000 pointers (and thus 999 separator keys). For the hash table, assume that each bucket is completely full with no overflow (i.e., 10,000 buckets). For simplicity, assume that record keys are assigned sequentially starting from 0.

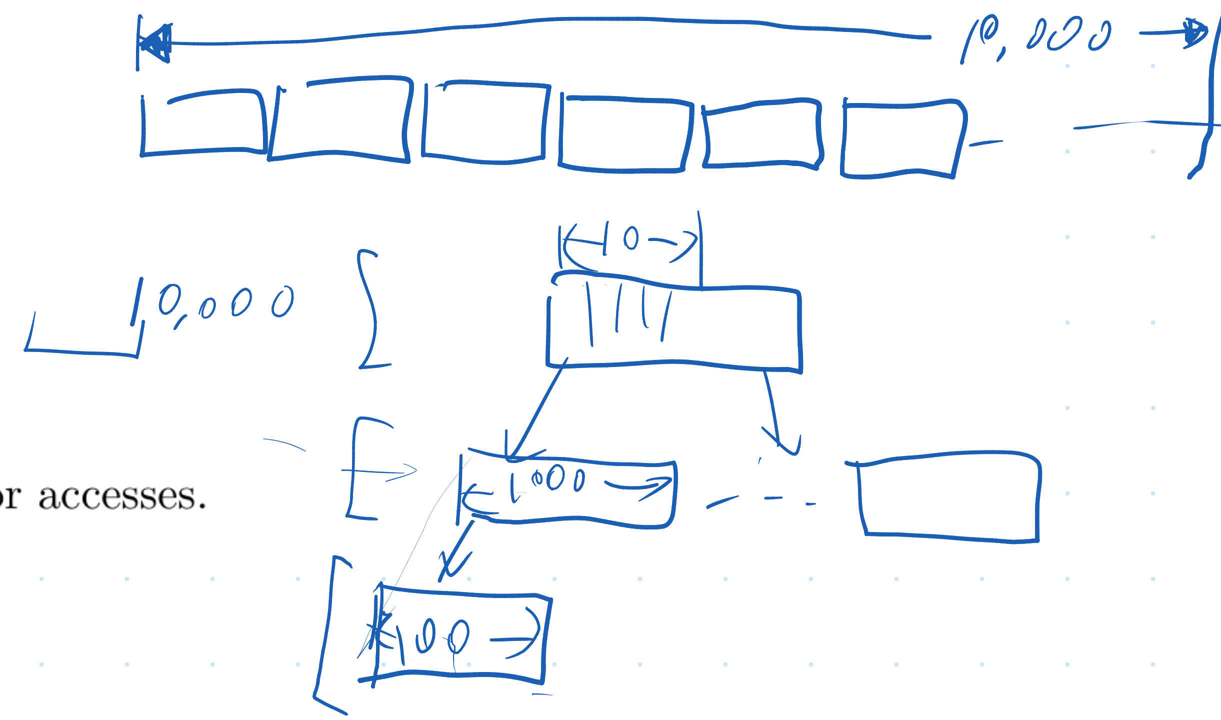
Each of the following The questions ask about the **IO complexity of** the following workload

- Access record with key 100,000 — 15
- Access record with key 100,001 — 0 — same data
- Access record with key 20 — 10 — [5, 10]
- Access record with key 21 — 0 — same data — same L1
- Access record with key 1,240 — 1 — [4, 10]
- Access record with key 1,241 — 0 — same data
- Access record with key 100,105 — 10 — 14 — same L1

Shared root

Each access is started entirely from scratch, with no memory of any prior accesses.

Hash Table



Question 1 [**10 points**]

Consider a naive implementation of disk access similar to what you implemented in PA0. That is, you have 1 page of working memory available to load on-disk data into, and so each access to a new page requires an IO. For each of the three data structures, state the exact IO complexity (i.e., exactly how many pages need to be read from disk) to perform the workload. Show your work.

Hash Table: 7

ISAM: No temporal locality benefit (1 page)

$7 \cdot 3 = 21$

Binary: $15 \cdot 7 = 105$

Consider a caching buffer manager with an **LRU** replacement policy and **3 pages** of working memory. For each of the three data structures, state the exact IO complexity. Show your work.

Hash Table: 7

ISAM: $3+0+2+0+1+0+2=8$

Binary: $15 \cdot 7 = 105$

Consider a caching buffer manager with a **FIFO** replacement policy and **3 pages** of working memory. For each of the three data structures, state the exact IO complexity. Show your work.

Hash Table : 7

ISAM : 3+0+2+3+1+3+2 = 14

Binary : 105

Consider a caching buffer manager with an **LRU** replacement policy and **20 pages** of working memory. For each of the three data structures, state the exact IO complexity. Show your work.

Hash Table : 7

ISAM : 1 root + 2 L1 + 4 data = 7

Binary : 36

15
0
10
0
1
0
10

36

Express the following query in terms of the relational algebra operators: Project (π; indicate what attributes to project), Filter (σ; indicate the condition), Join ($\bowtie$; indicate the join condition), Union ($\cup$), and/or Aggregate ($\sum$; indicate the computation).

```
SELECT a.name, m.name FROM actors a, movies m WHERE a.acted_in = m.id
```

1 2 3

$$\pi_{a.name,\, m.name}\left(a \bowtie_{acted\_in = id} m\right) = \pi_{a.name,\, m.name}\left(\sigma_{acted\_in = id}(a \times m)\right)$$

Actors

id	name	born_in	acted_in
144	Carey Elwes	London, UK	799
705	Robin Wright	Dallas, TX	799
705	Robin Wright	Dallas, TX	830

redundant

redundant

Movies

id	name	awards	director	director_birthday
799	The Princess Bride	ASFFHF, Hugo, NFPB	Rob Reiner	1947-03-06
258	This is Spinal Tap	OFTA-HoF, NFR	Rob Reiner	1947-03-06
830	Forrest Gump	Oscar, Oscar, SFFHF, ACA	Robert Zemeckis	1952-05-14

more than one value

The SQL `LIKE` condition performs simple pattern matching, where `?` matches a single character and `%` matches any number of characters. For example, one could write `name LIKE 'Oliver%'` to match any record who's name field *begins with* the string `Oliver`.
Write a SQL query to count the number of movies that have won at least one Oscar.

Select count(*)
FROM movies
WHERE awards LIKE "%Oscar%"

Identify at least two ways in which the datasets above violate our rules for "tidy" data. Propose a fix for the violations you list in the form of a new schema for the dataset. That is, list each table in your corrected dataset. For each table list the attributes in the table.

See above

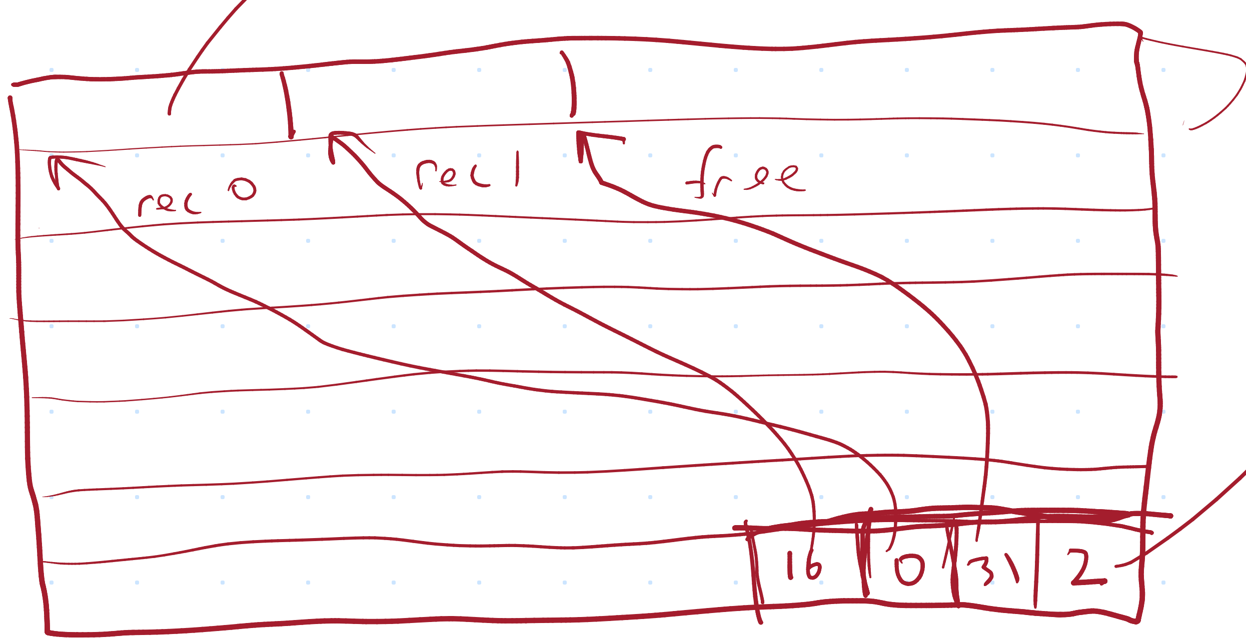
The questions in this section are about record layouts stored on 50 byte pages. Each question presents a sequence of operations, and asks you to draw the state of the page after the operations are performed. You do not need to draw each bit/byte individually; Instead follow the convention used in class (and in the PA2 function descriptions): Rows bytes with the upper-left representing the zero'th byte, and the lower-right representing the last byte. For each region of the page used to store data, make sure to identify: (i) how many bytes the region occupies, (ii) what role the data stored there serves, and (iii) what specific data is stored there (e.g., as a string or integer). Use your PA2 implementation as a reference, but *do not defragment*.

Assume that all strings are **not** '\0'-terminated, and that all integers are normal width (4 bytes). Strings should not include their enclosing quotation marks.

- Insert ["foo!", 42] at record index 0
- Insert ["bar", 9] at record index 1

num of fields
start of each field
end of record

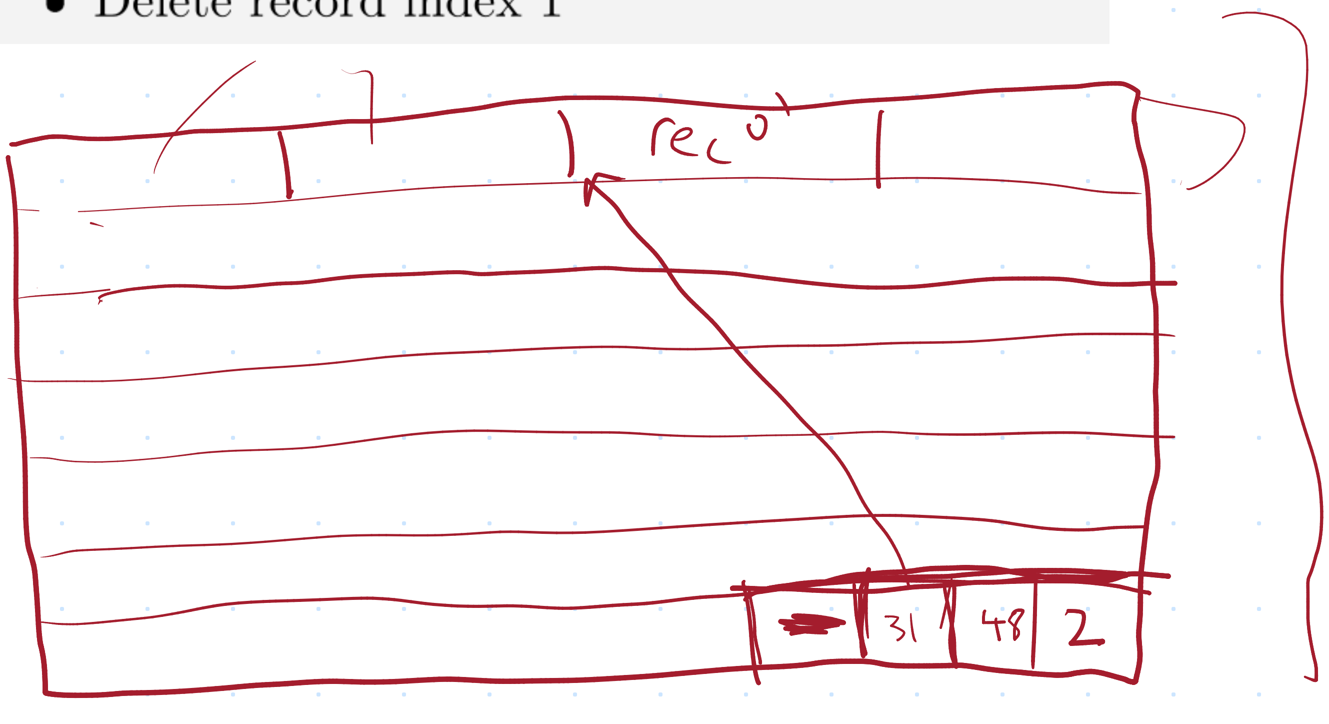
answer must have this info



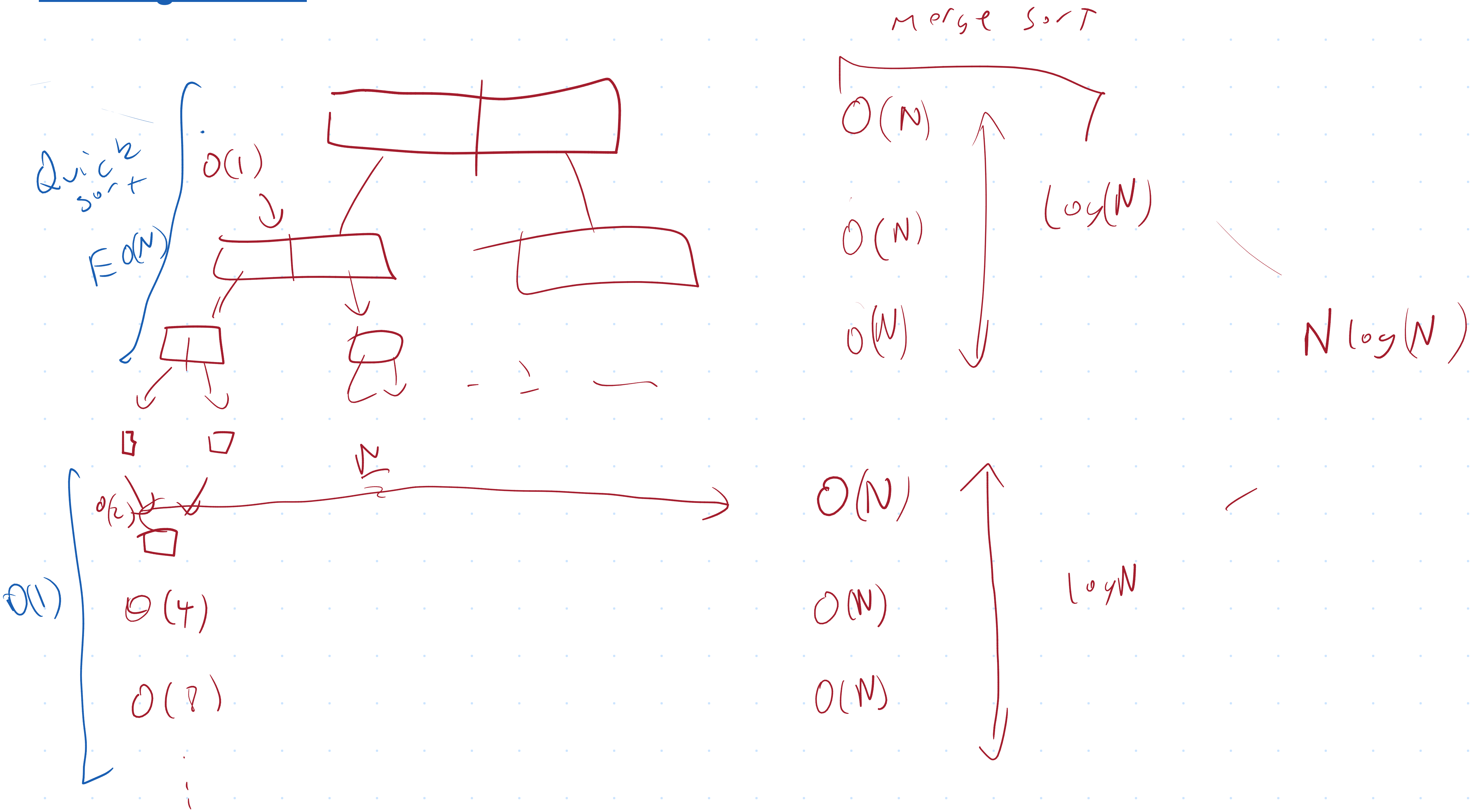
Number of records allocated
Free space ptr
Start address of each record

As before, but then also...

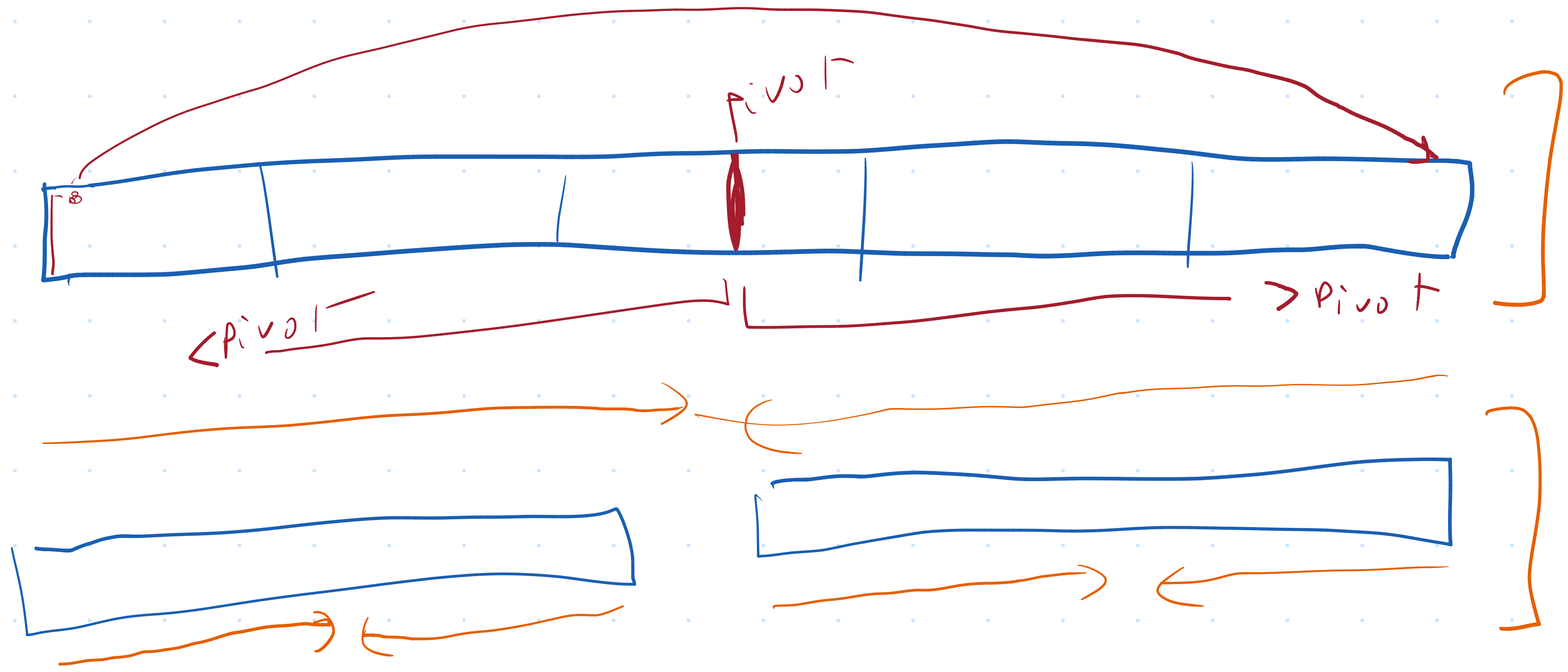
- Update record index 0 to `["moof!", 100]`
- Delete record index 1



Sort Algorithms



$\frac{N}{P}$ Pages



$\frac{N}{P}$ IOs

$\frac{N}{P}$ IOs

$E_x = \log_2 N$

5 3 9 10 2 4

3 5 9 10 2 4

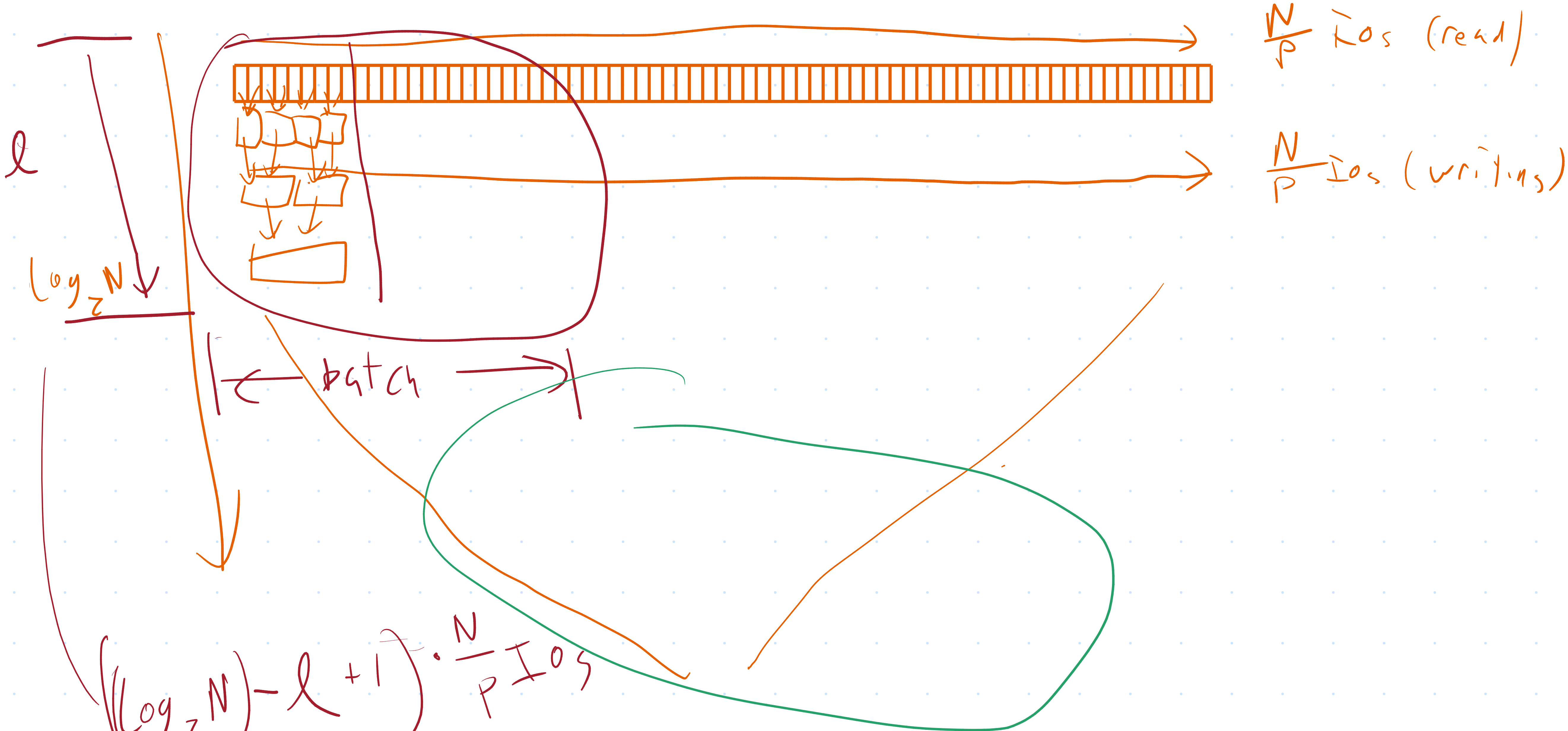
3 5 4 10 2 9

3 4 5 10 2 9

3 4 5 2 10 9

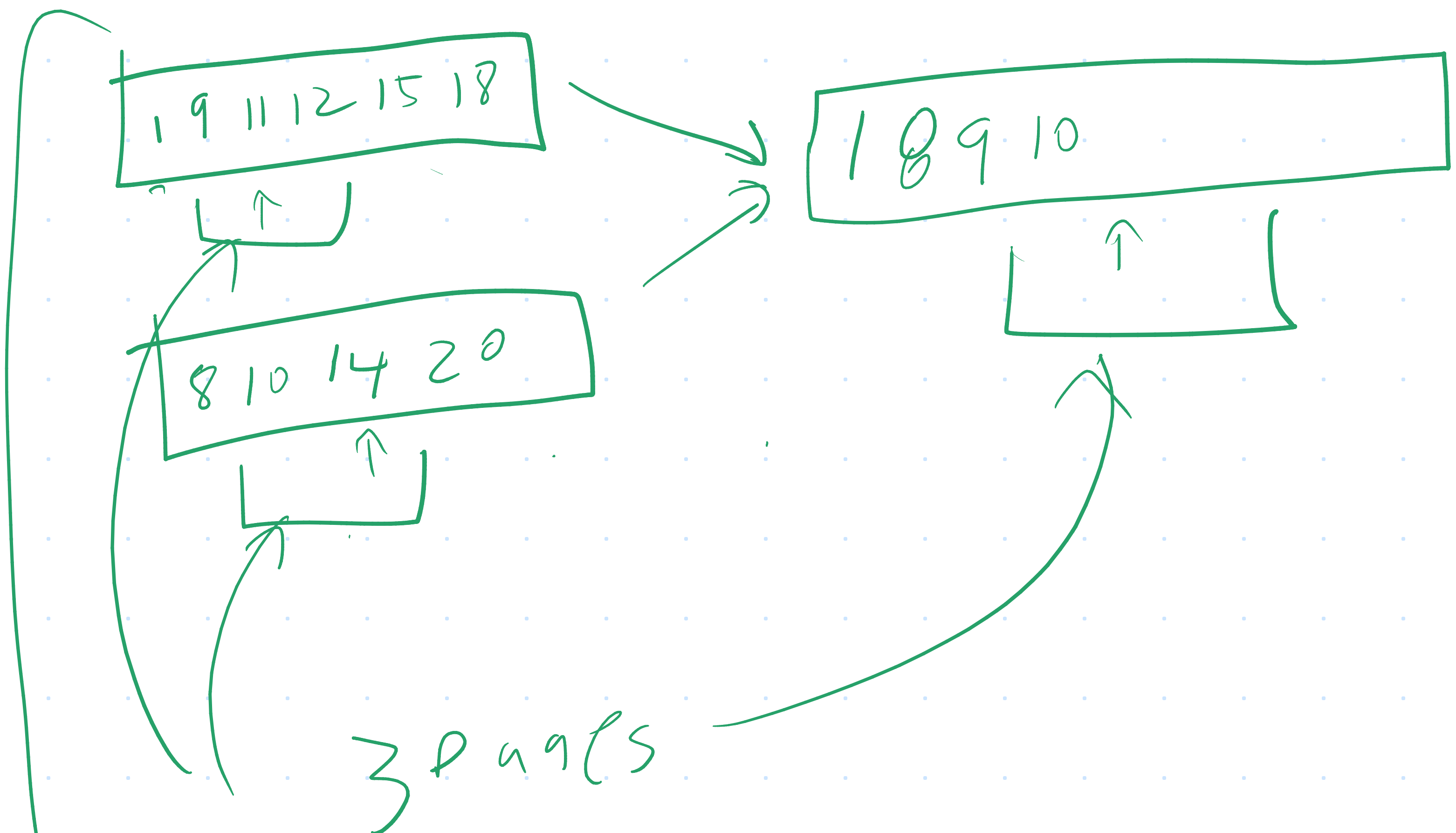
3 4 2 5 10 9

$(\log_2 N) \cdot \frac{N}{P}$ IO



$$\left((\log_2 N) - \ell + 1\right) \cdot \frac{N}{p} \text{ IOs}$$

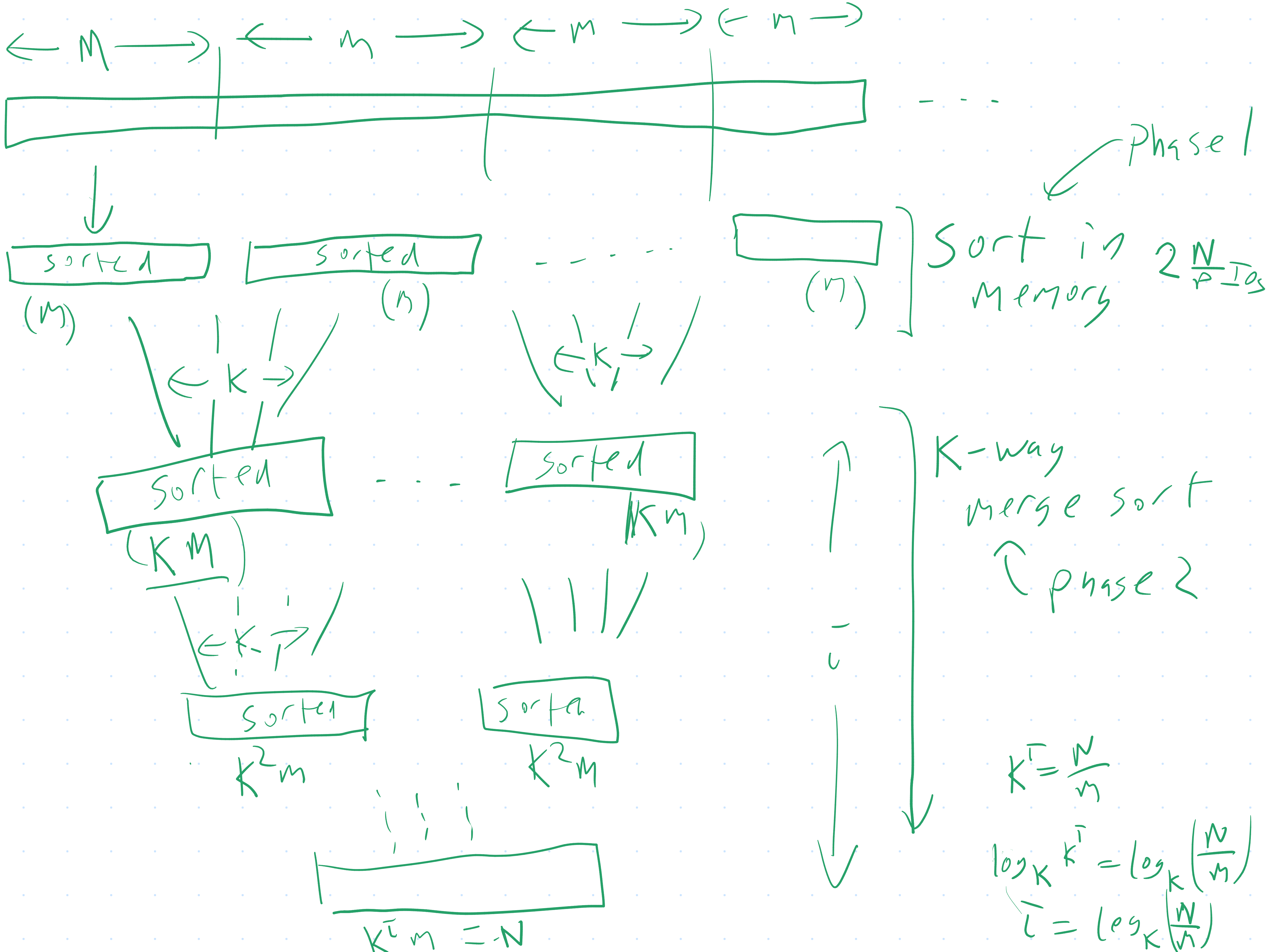
$$\underbrace{\left((\log_2 N)\right) \cdot \frac{N}{p}}_{\text{Big}}$$



Read from K ≥ 2 sorted inputs at once

2-Phase Sort

M = available memory



$$K^i = \frac{N}{M}$$

$$\log_K K^i = \log_K\left(\frac{N}{M}\right)$$

$$i = \log_K\left(\frac{N}{M}\right)$$

Phase 1 $\quad 2\frac{N}{P}$ IOs (read + write)

Phase 2 $\quad \left(\log_K\left(\frac{N}{M}\right)\right) \cdot 2 \cdot \frac{N}{P}$ IOs $= \left(\log_K N - \log_K M\right) \cdot 2 \cdot \frac{N}{P}$

rounds of merging

one read one write

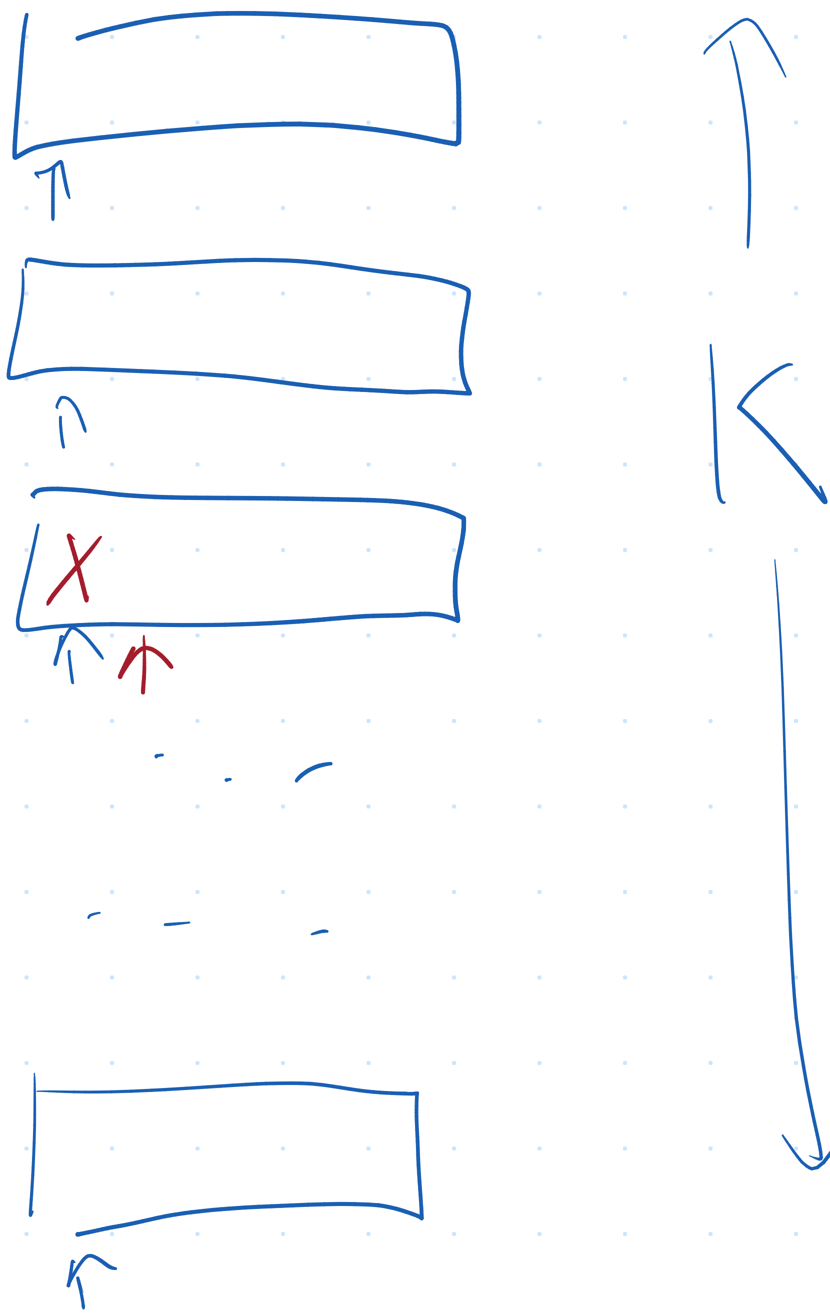
size of data

always a win to maximize amount of work in phase 1

$$\left(1 + \log_K \frac{N}{M}\right) \cdot 2 \cdot \frac{N}{P} = O\left(\frac{N}{P} \log_K\left(\frac{N}{M}\right)\right)$$

vs

$$\frac{N}{P} \log_2(N) \qquad O\left(\frac{N}{P} \log_2(N)\right)$$



$O(K)$ steps to find least element

$O(N)$ repetitions

$O(N \cdot K)$ runtime.

Idea

$O(K)$ ↳ sort "list" of inputs

$O(\log K)$ ↳ remove min element

$O(\log K)$ ↳ reinsert input

heap!

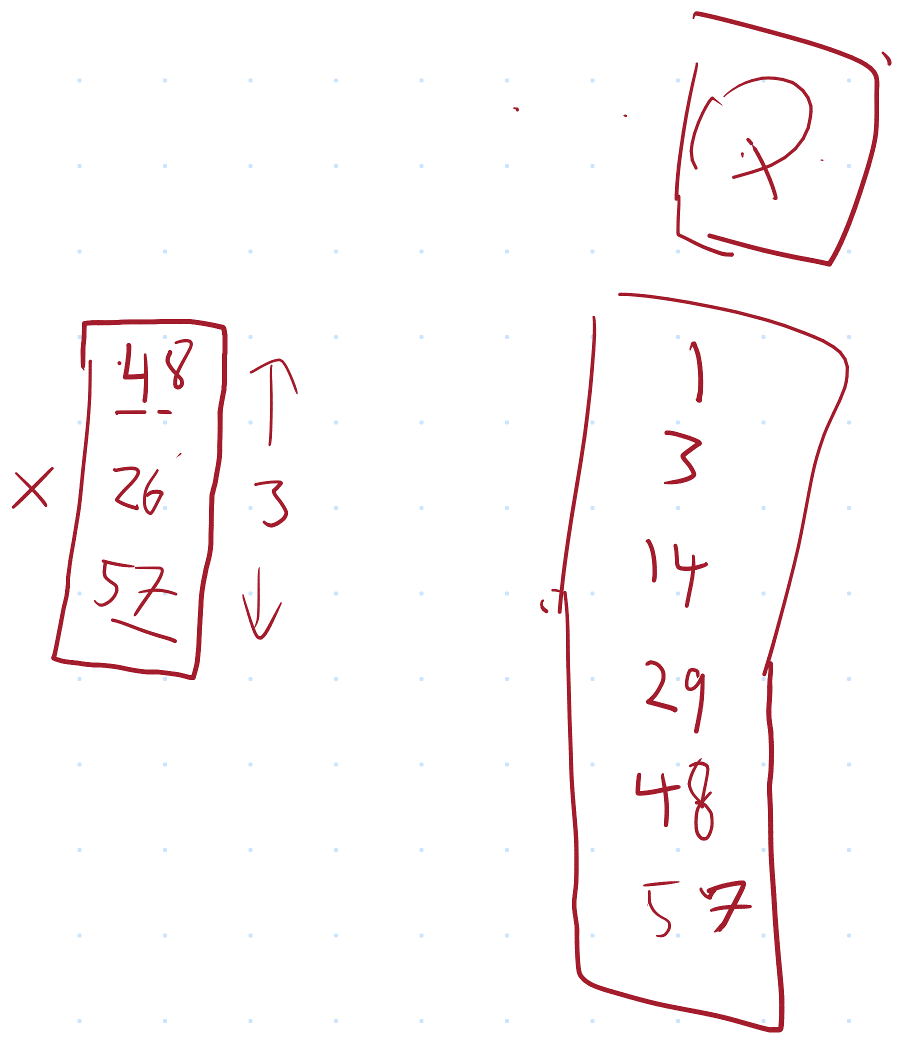
heap

↳ $O(N \cdot \log(K) + K)$ → runtime

Optimizing Phase 1

Array

1 29 3 14 57 48 26



1. fill buffer
2. find min record (that is greater than greatest record output in current batch)
3. move to output
4. replace it w/next input
5. goto 2 if possible
6. else finish batch & goto 7

Cases

Fully shuffled array: Expected sorted run size = 2M

Fully sorted array: (Unqualified) sorted run size = N

~~for~~ almost

Worst case = sorted run size = M