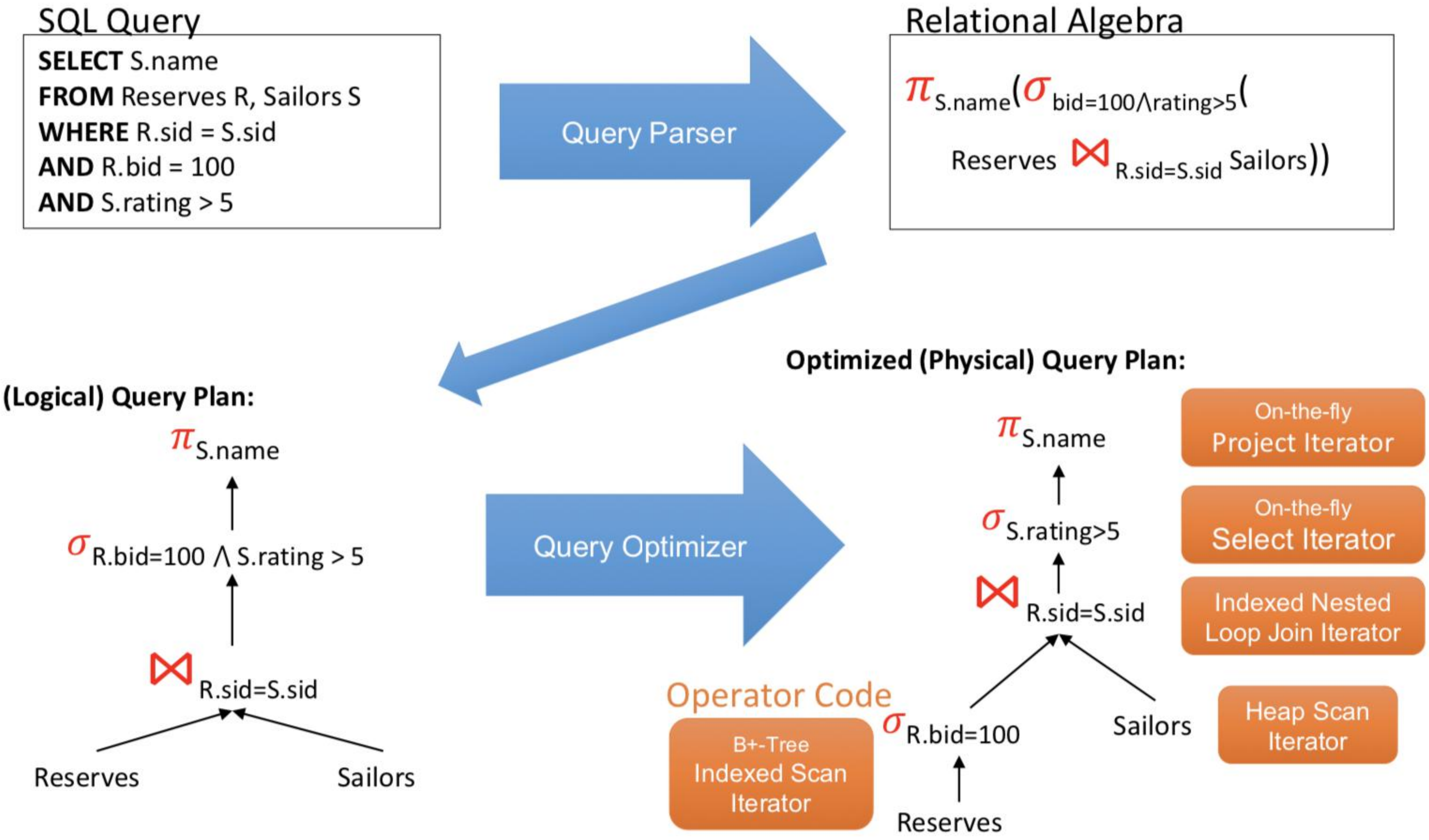
Disccusion5

Relational Algebra & External Merge Sort

Relational Algebra

- <u>Unary Operators</u>: on **single relation**
- **Projection** (π): Retains only desired columns (vertical)
- **Selection** (σ): Selects a subset of rows (horizontal)
- **Renaming** (ρ): Rename attributes and relations.
- <u>Binary Operators</u>: on **pairs of relations**
- **Union** ($\cup$): Tuples in r1 or in r2.
- **Set-difference** (—): Tuples in r1, but not in r2.
- **Cross-product** ($\times$): Allows us to combine two relations.
- <u>Compound Operators</u>: common "*macros*" for the above
- **Intersection** ($\cap$): Tuples in r1 and in r2.
- **Joins** ($\bowtie_\theta$, $\bowtie$): Combine relations that satisfy predicates

Big Picture Overview



Compound Operator: Join

- Joins are compound operators (like intersection):
  - Generally, σ_{θ}(R × S)
- Hierarchy of common kinds:
  - **Theta Join** ($\bowtie_{\theta}$): join on logical expression θ
    - **Equi-Join:** theta join with theta being a conjunction of equalities
      - **Natural Join** ($\bowtie$): equi-join on all matching column names

Note: we will need to learn a good join algorithm.

Avoid cross-product if we can!!

Another Theta Join ($\bowtie\theta$), Pt 2

- $\mathbf{R} \bowtie_\theta \mathbf{S} = \sigma_\theta(\ \mathbf{R} \times \mathbf{S})$
- **Example:** *More senior sailors for each sailor.*
- S1 $\bowtie_{age\ <\ age2}$ S1

S1:

f1	f2	f3	f4
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S1				S1			
f1	f2	f3	f4	f5	f6	f7	f8
~~22~~	~~dustin~~	~~7~~	~~45.0~~	~~22~~	~~dustin~~	~~7~~	~~45.0~~
22	dustin	7	45.0	31	lubber	8	55.5
~~22~~	~~dustin~~	~~7~~	~~45.0~~	~~58~~	~~rusty~~	~~10~~	~~35.0~~
~~31~~	~~lubber~~	~~8~~	~~55.5~~	~~22~~	~~dustin~~	~~7~~	~~45.0~~
~~31~~	~~lubber~~	~~8~~	~~55.5~~	~~31~~	~~lubber~~	~~8~~	~~55.5~~
~~31~~	~~lubber~~	~~8~~	~~55.5~~	~~58~~	~~rusty~~	~~10~~	~~35.0~~
58	rusty	10	35.0	22	dustin	7	45.0
58	rusty	10	35.0	31	lubber	8	55.5
~~58~~	~~rusty~~	~~10~~	~~35.0~~	~~58~~	~~rusty~~	~~10~~	~~35.0~~

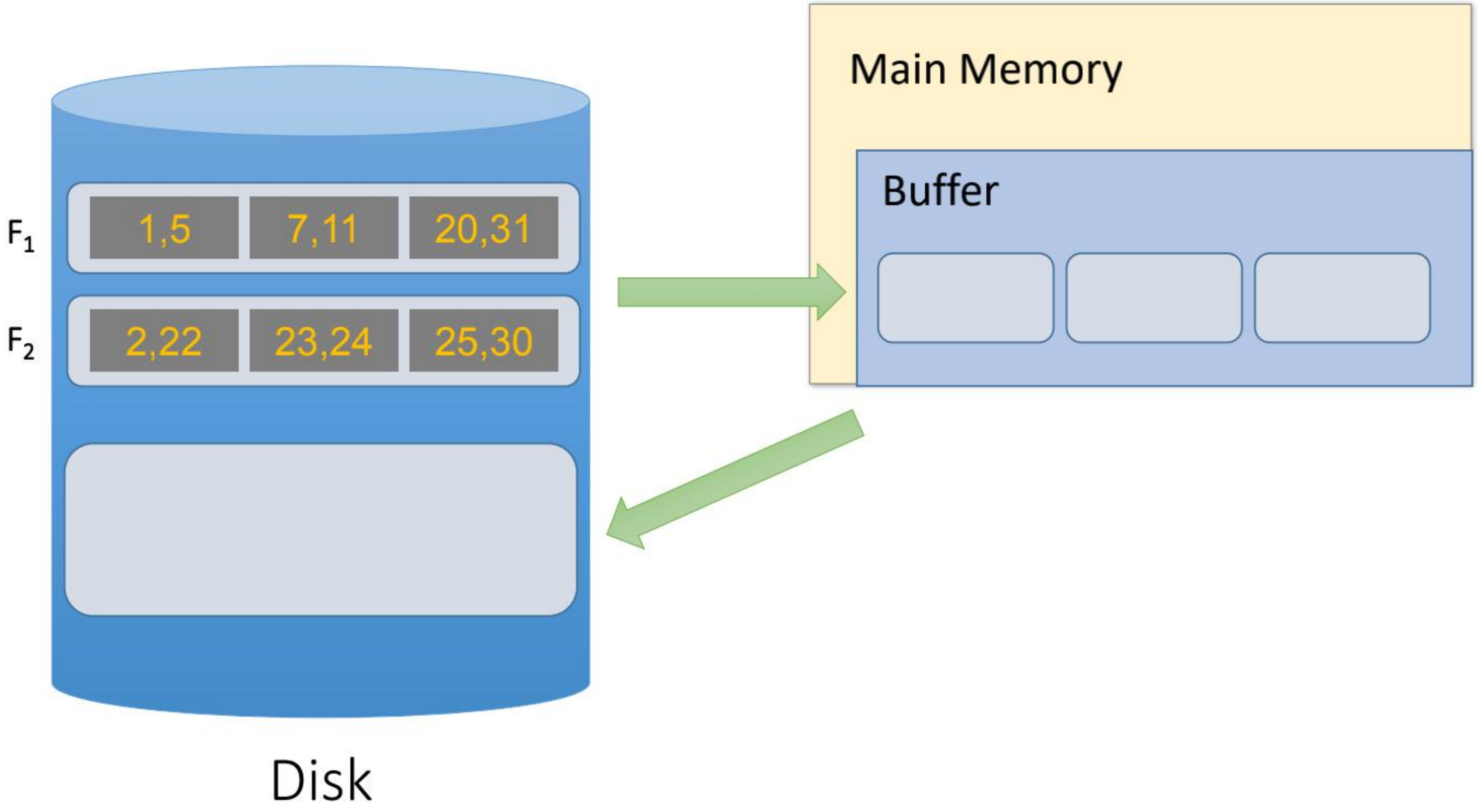
External Merge Sort

- External Merge and Sort Algorithm（simple example）
- Running External Merge Sort on Larger Files
- 3 page buffer/B+1buffer

External Merge Algorithm

Input:
Two sorted
files

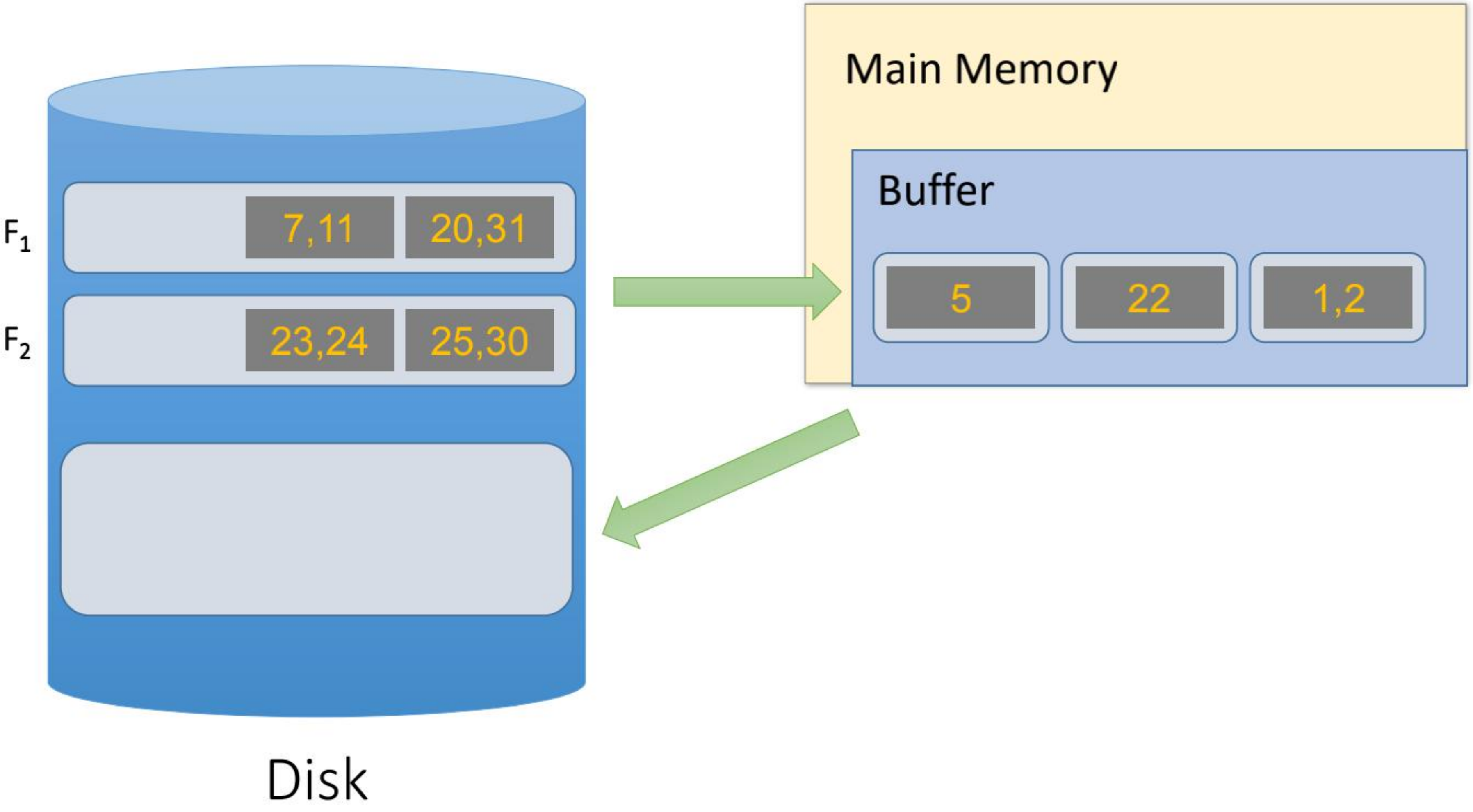
Output:
One *merged*
sorted file



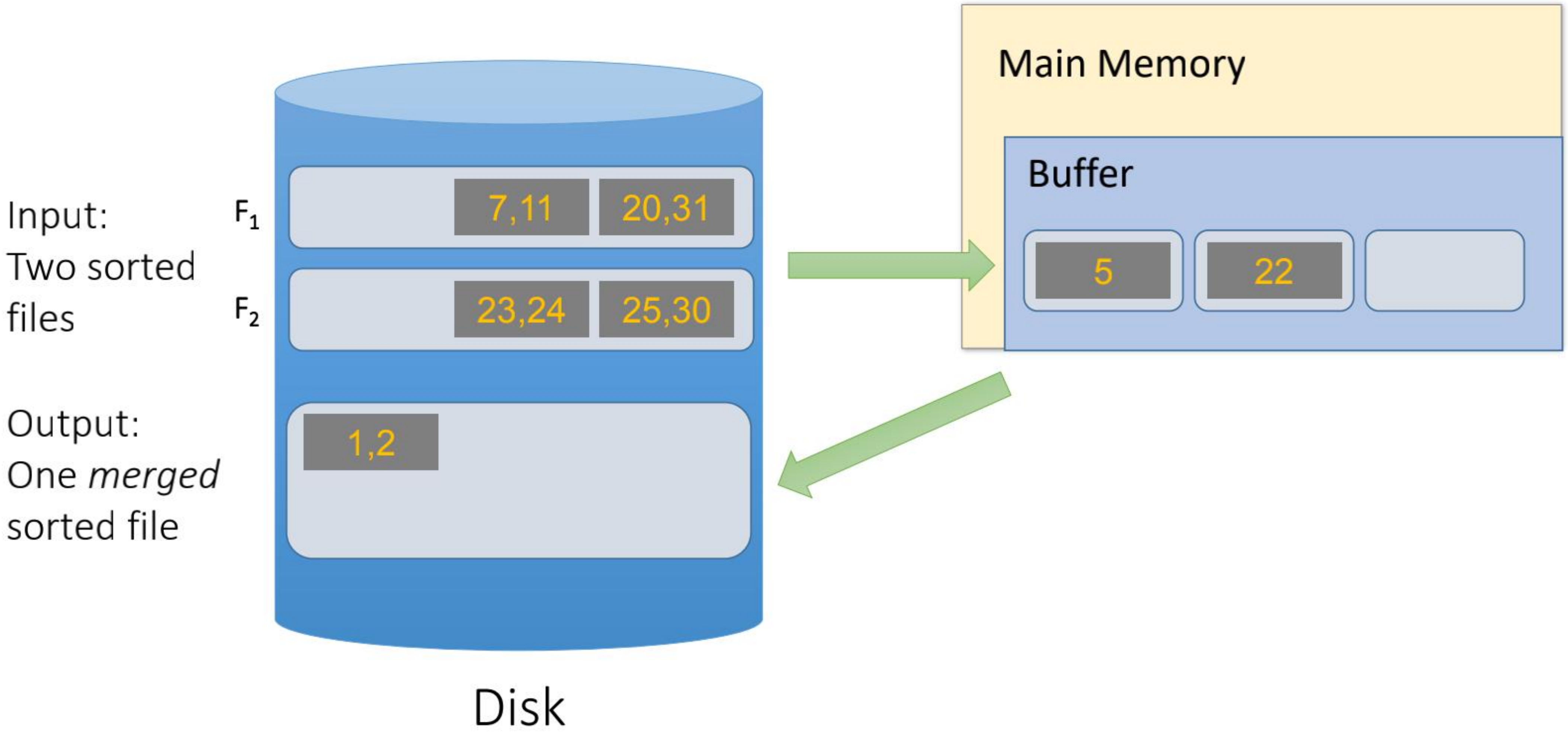
External Merge Algorithm

Input:
Two sorted
files

Output:
One *merged*
sorted file



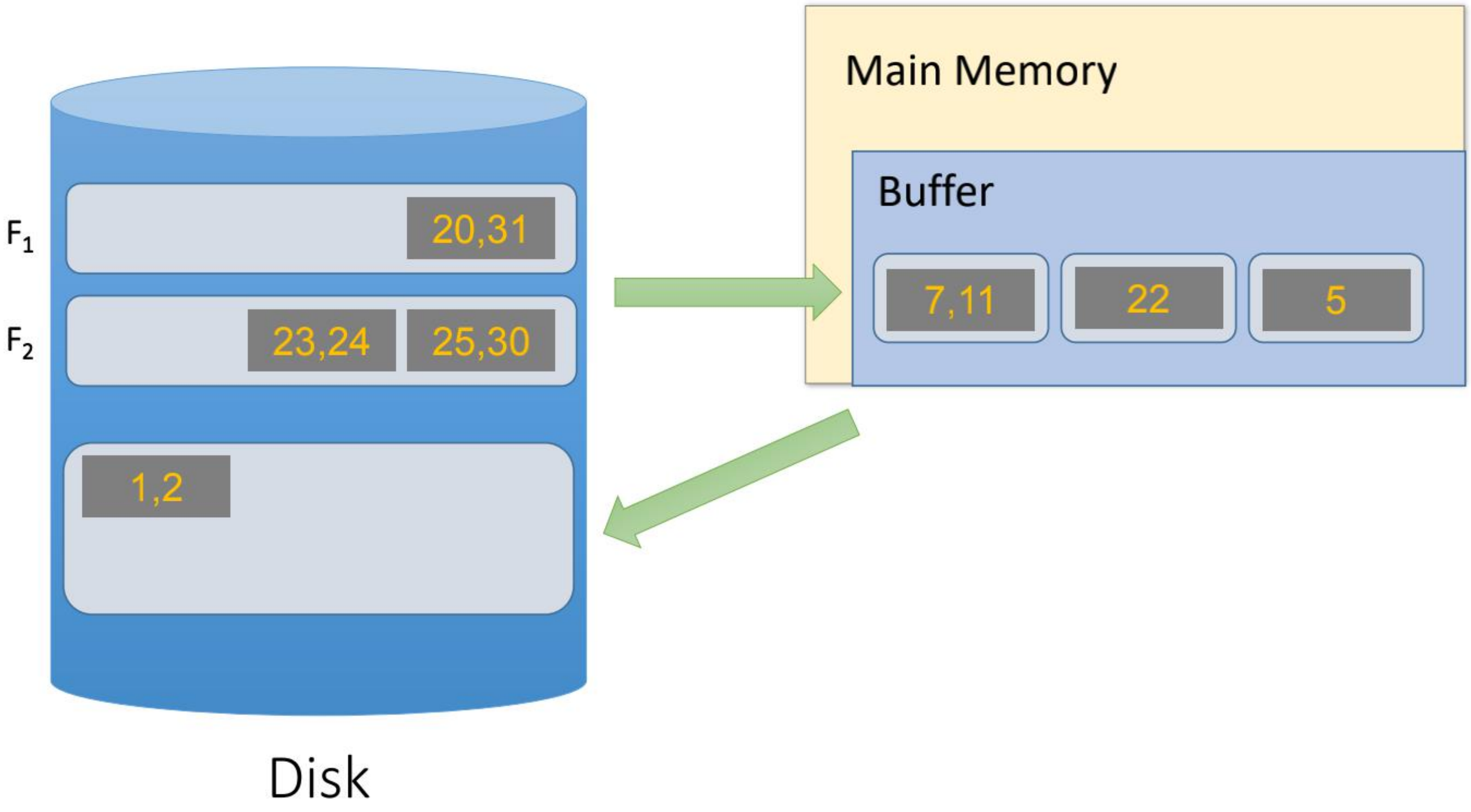
External Merge Algorithm



External Merge Algorithm

Input:
Two sorted
files

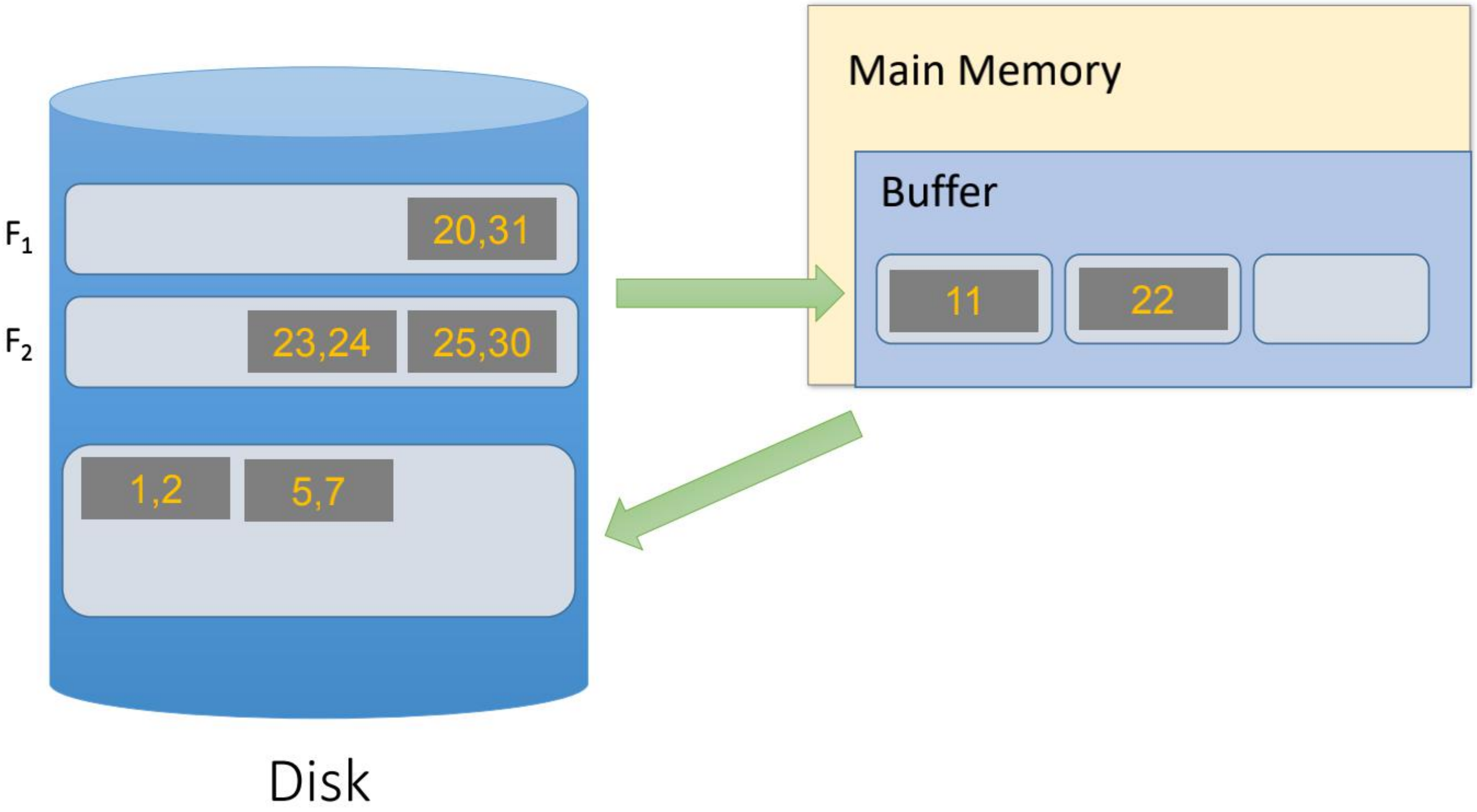
Output:
One *merged*
sorted file



External Merge Algorithm

Input:
Two sorted
files

Output:
One *merged*
sorted file



We can merge lists of **arbitrary length** with *only* 3 buffer pages.

If lists of size M and N, then

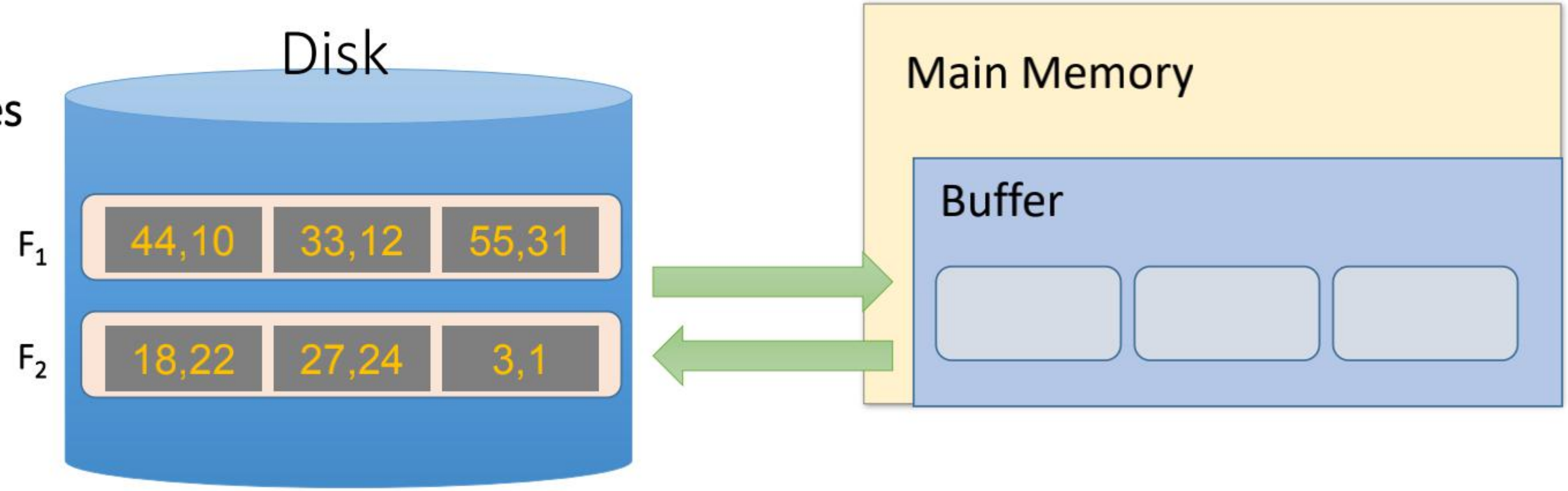
Cost: 2(M+N) IOs

Each page is read once, written once

External Merge Sort Algorithm

Example:
- 3 Buffer pages
- 6-page file

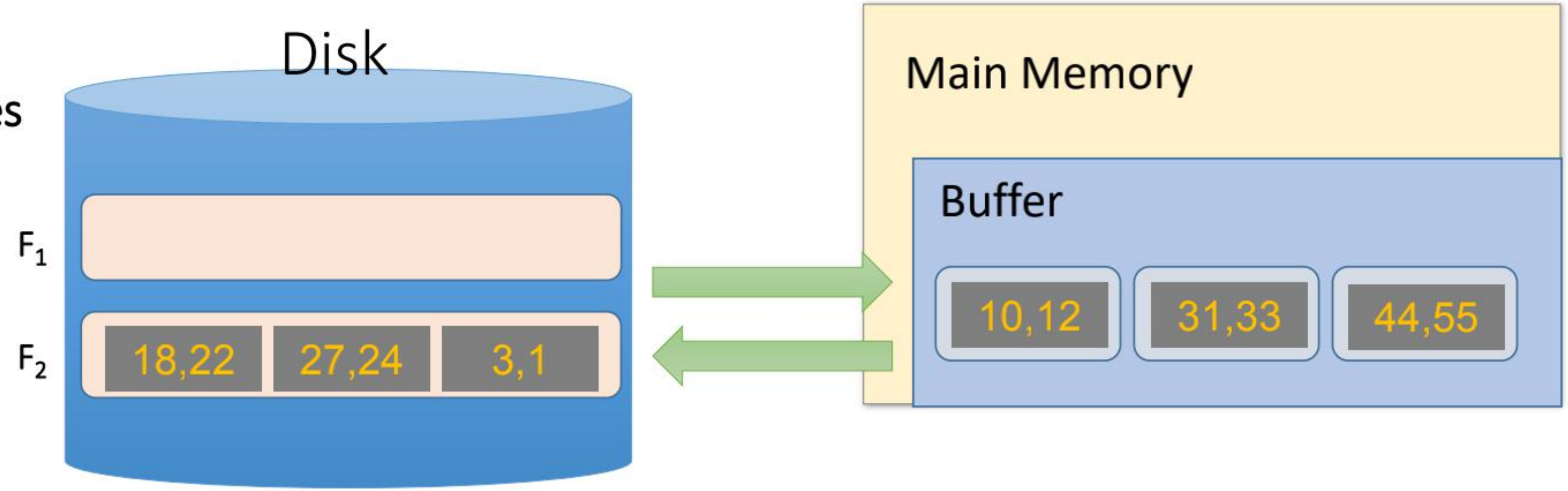
Orange file = unsorted



1. Split into chunks small enough to **sort in memory**

sorting in buffer

Example:
- 3 Buffer pages
- 6-page file

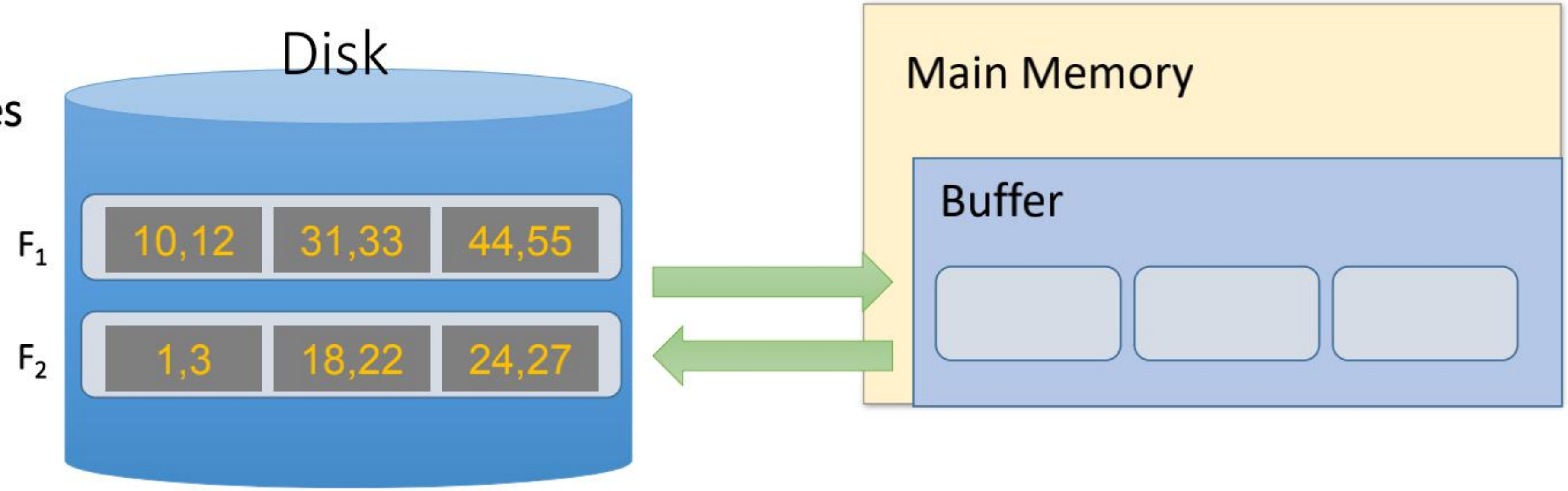


Orange file
= unsorted

1. Split into chunks small enough to **sort in memory**

External Merge Sort Algorithm

Example:
- 3 Buffer pages
- 6-page file

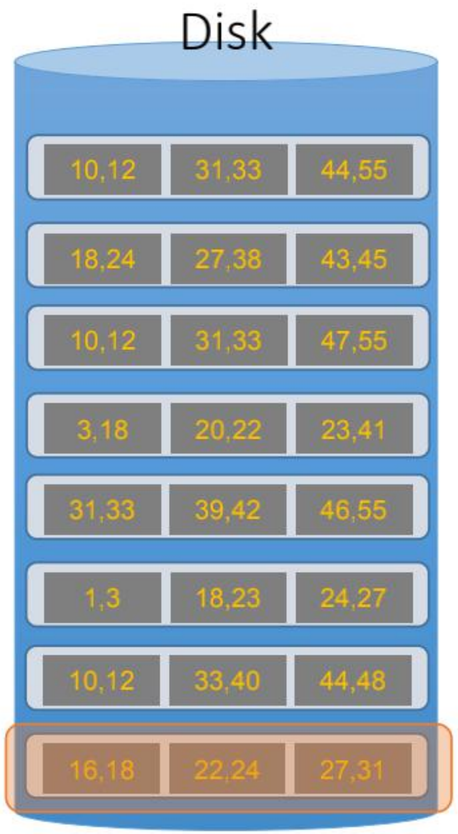


2. Now just run the **external merge** algorithm & we're done!

External Merge Sort

- External Merge Sort Algorithm （simple example）
- Running External Merge Sort on Larger Files
- 3 page buffer/B+1buffer

Running External Merge Sort on Larger Files

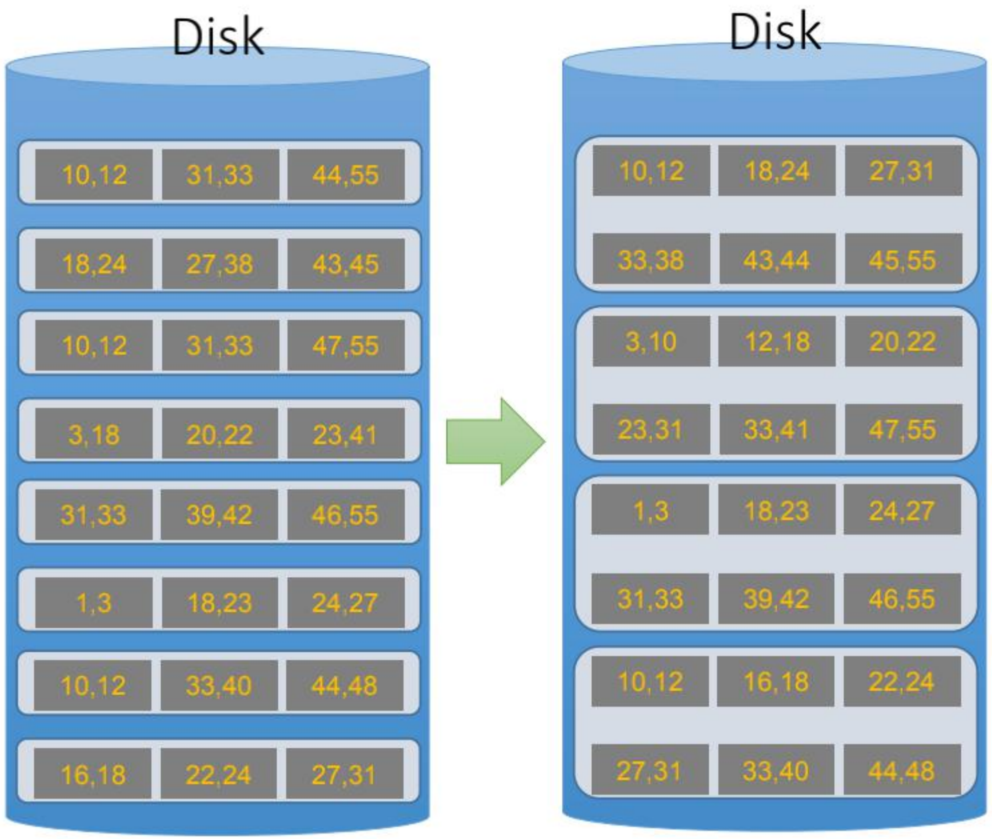


1. Split into files small enough to sort in buffer… and sort

Assume we still only have *3* buffer pages *(Buffer not pictured)*

Call each of these sorted files a ***run***

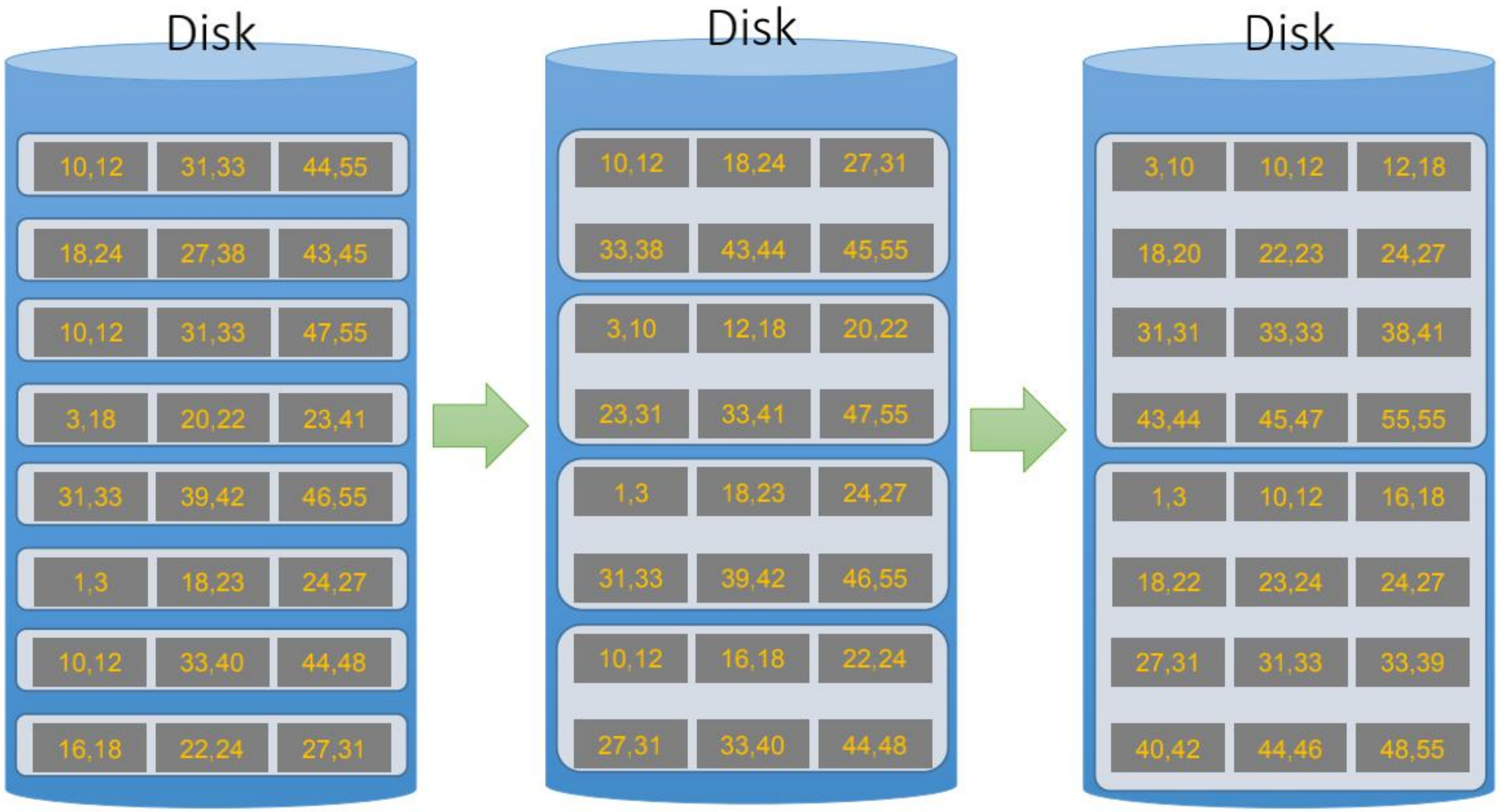
Running External Merge Sort on Larger Files



Assume we still only have *3* buffer pages *(Buffer not pictured)*

2. Now merge pairs of (sorted) files... **the resulting files will be sorted!**

Running External Merge Sort on Larger Files



Assume we still only have *3* buffer pages *(Buffer not pictured)*

3. And repeat…

Call each of these steps a ***pass***

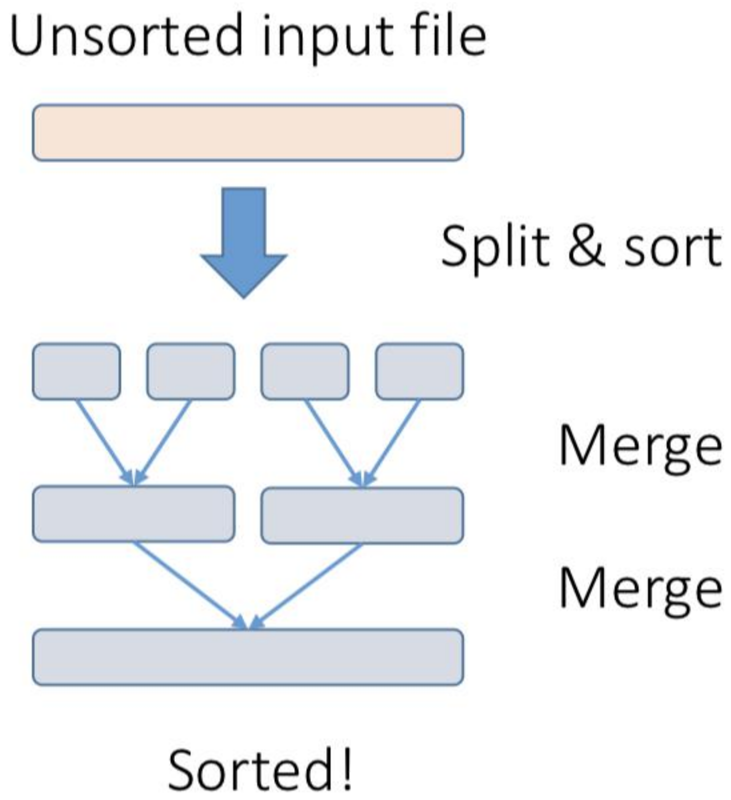
External Merge Sort

- External Merge Sort Algorithm （simple example）
- Running External Merge Sort on Larger Files
- 3 page buffer/B+1buffer

Simplified 3-page Buffer Version

Assume for simplicity that we split an N-page file into N single-page ***runs*** and sort these; then:

- First pass: Merge **N/2 *pairs* of runs** each of length **1 page**
- Second pass: Merge **N/4 *pairs* of runs** each of length **2 pages**
- In general, for **N** pages, we do $\lceil \boldsymbol{log_2 N} \rceil$ passes
  - +1 for the initial split & sort
- Each pass involves reading in & writing out all the pages = ***2N IO***



→ 2N*($\lceil \boldsymbol{log_2 N} \rceil$+1) total IO cost!

Using B+1 buffer pages to reduce # of passes

Suppose we have B+1 buffer pages now; we can:

1. **Increase length of initial runs**. Sort B+1 at a time!

At the beginning, we can split the N pages into runs of length B+1 and sort these in memory

IO Cost:

$$2N(\lceil \log_2 N \rceil + 1) \rightarrow 2N\left(\left\lceil \log_2 \frac{\boldsymbol{N}}{\boldsymbol{B+1}} \right\rceil + 1\right)$$

Starting with runs of length 1

Starting with runs of length ***B+1***

Using B+1 buffer pages to reduce # of passes

Suppose we have B+1 buffer pages now; we can:

2. Perform a B-way merge.

On each pass, we can merge groups of ***B*** runs at a time (vs. merging pairs of runs)!

IO Cost:

$$2N(\lceil \log_2 N \rceil + 1) \rightarrow 2N\left(\left\lceil \log_2 \frac{\boldsymbol{N}}{\boldsymbol{B+1}} \right\rceil + 1\right) \rightarrow 2N\left(\left\lceil \log_B \frac{\boldsymbol{N}}{\boldsymbol{B+1}} \right\rceil + 1\right)$$

Starting with runs of length 1

Starting with runs of length ***B+1***

Performing ***B***-way merges

practise

1) You are trying to sort the Students table which has 1960 pages with 8 available buffer pages.
a. How many sorted runs will be produced after each pass?
b. How many pages will be in each sorted run for each pass?
c. How many IOs does the entire sorting operation take?

2) What is the minimum number of buffer pages that we need to sort 1000 data pages in two passes?

1) pass 0: $\frac{1960}{8} = 245$ sorted runs → 8 pages / run

↓

merge 7 sorted runs at a time (one for output buffer)

$\frac{245}{7} = 35$ sorted runs → $8 \times 7 = 56$ pages

↓

$\frac{35}{7} = 5$ sorted runs → $56 \times 7 = 392$ pages

↓

1 sorted runs → 1960

$2N \cdot 4 = 15680$ IOs

↓

$\text{pass} = \lceil \log_7 \frac{N}{8} \rceil + 1$

2). $\frac{1000}{B(B-1)} \leq 1$

$B = 32.1$ ∴ 33 buffer pages