COL362/632 Introduction to Database Management Systems

Database Systems – External Sorting

Kaustubh Beedkar

Department of Computer Science and Engineering
Indian Institute of Technology Delhi

Course Outlook

Part-I: Database Design

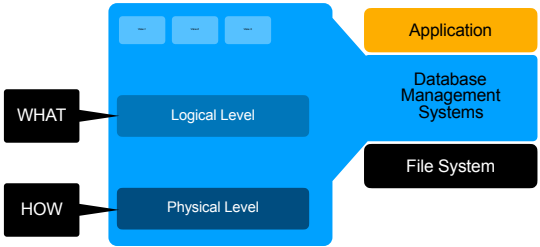
1. Data Model – E/R, Relation Model
2. Relational Algebra
3. SQL
4. Schema design

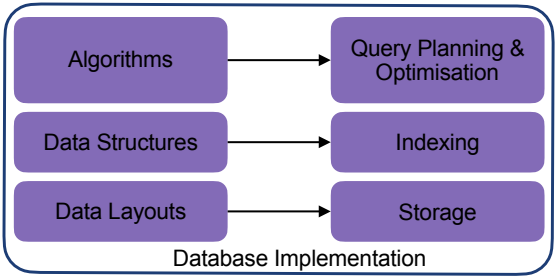
Part-II: Database Implementation

1. Storage
2. Indexing
3. Query planning & optimization ⇐

Part-III: Transactions

Part-IV: Big Data/Misc. Topics

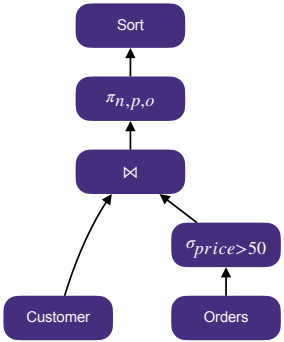




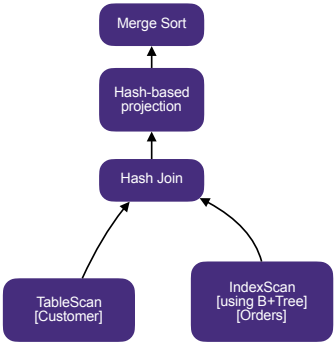
Logical and Physical Operators

- SQL : What?
- Physical plan: How?
- Query planning and optimization "compiles" a logical plan in to a physical plan

```
SELECT c.name, o.price, o.order_date
FROM customer c, orders o
WHERE c.customer_id = o.customer_id
AND o.price > 50
ORDER BY o.order_date DESC;
```



Logical Plan



Physical Plan

Logical and Physical Operators

- SQL : What?
- Physical plan: How?
- Query planning and optimization “compiles” a logical plan in to a physical plan
- Physical operators
  1. Sort
  2. Scans (access methods)
  3. Selection
  4. Projection
  5. Joins
  6. Groupby and Aggregation

Sorting in Databases

Sorting is a very fundamental operation

- `order by...`
- B+Tree bulk loading
- `select distinct ...`
- sort-merge join (more on this later)

But, what about the sorting algorithms we studied in COL106?

- quick sort
- merge sort
- heap sort
- bubble sort
- ... https://visualgo.net/en/sorting; https://www.toptal.com/developers/sorting-algorithms

Yes, but how to sort records of a file organized as blocks on disk?

and, how to sort 1TB of data with 8GB of RAM?

External Sorting Problem

- Internal sorting (Quick Sort, Heap Sort) works in memory
- **External Sorting** minimize disk IO operations, which are much slower than memory operations

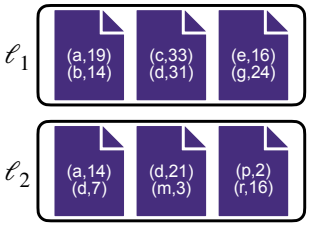
Sorting cards in hand

Sorting a million books in a library with a small table

External Merge

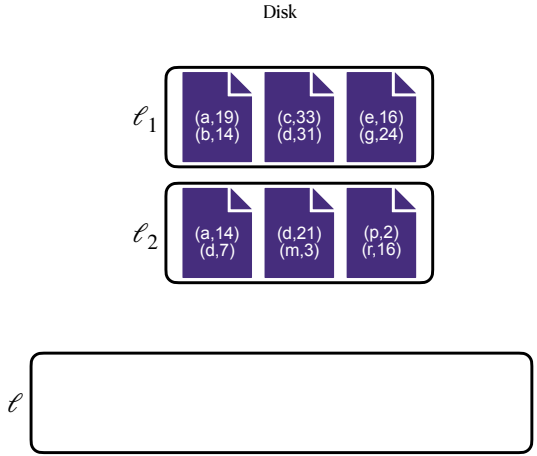
- Sorting primitive
- **Given:** two sorted list:
  - ℓ_1 with m pages
  - ℓ_2 with n pages

  and, buffer pool that can hold 3 pages
- **Goal:** Output one sorted list
  - ℓ with $m + n$ pages



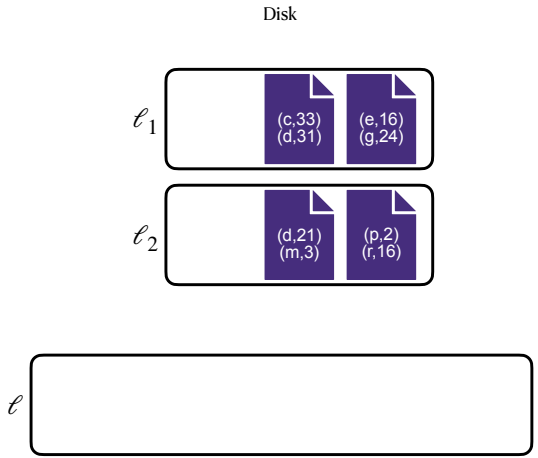
- Can be done using only $2 \times (m + n)$ I/Os

External Merge (Example)



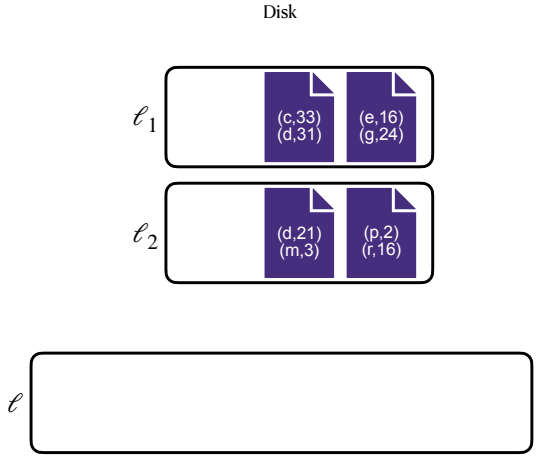
External Merge (Example)

- **Read first page from each file**



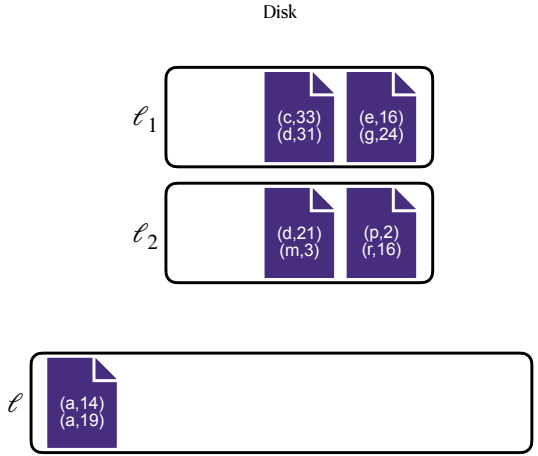
External Merge (Example)

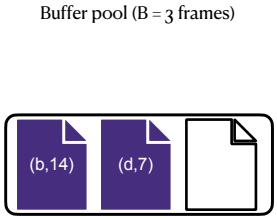
- **Merge into a new page until filled**



External Merge (Example)

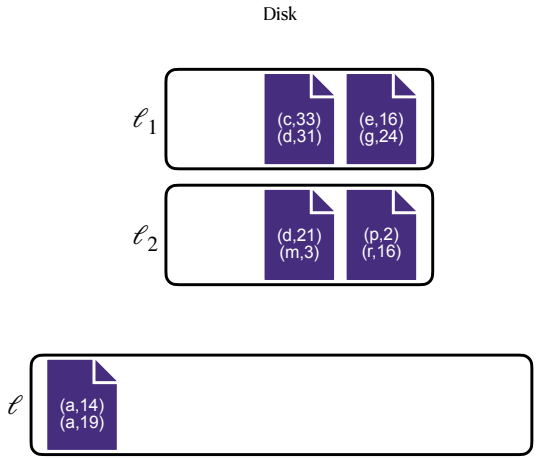
- **Write page to disk**

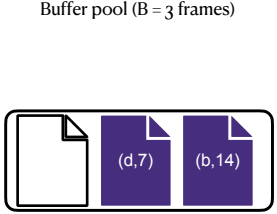




External Merge (Example)

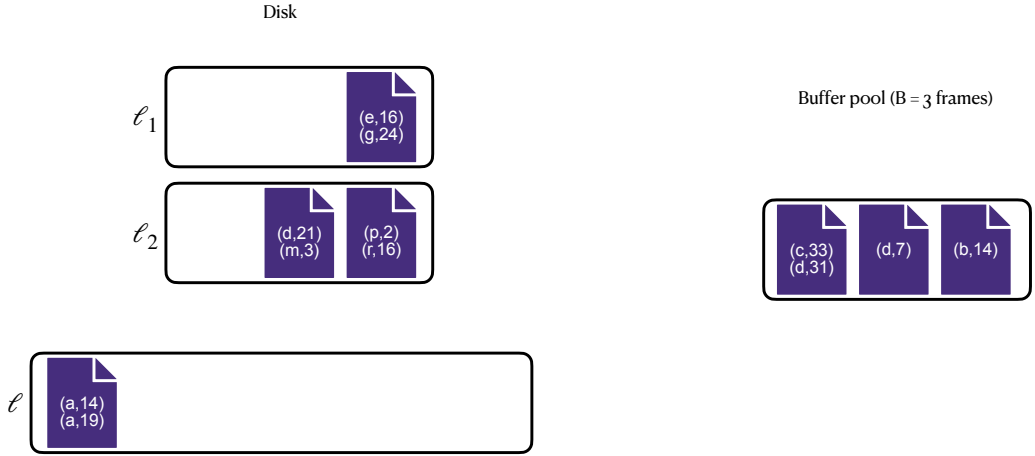
- **Continue merging until a frame is empty**





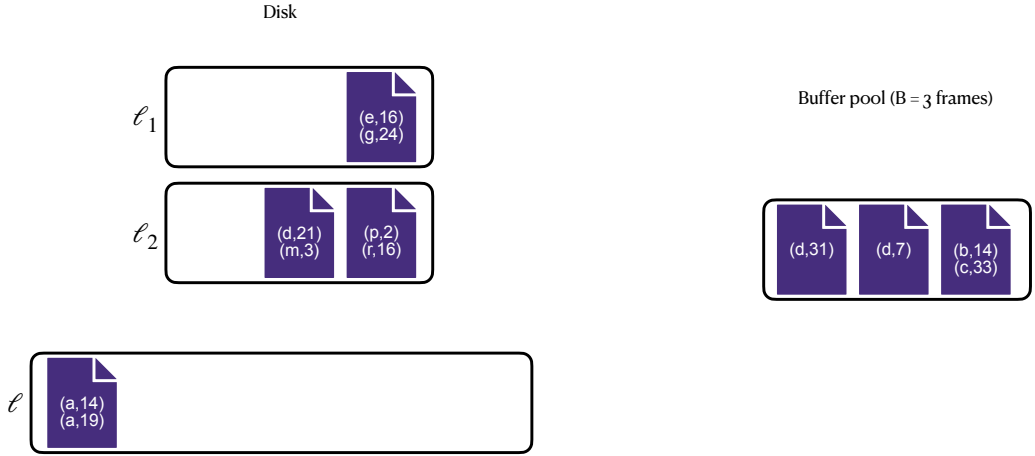
External Merge (Example)

- **Read page from ℓ_1 as $b < d$**



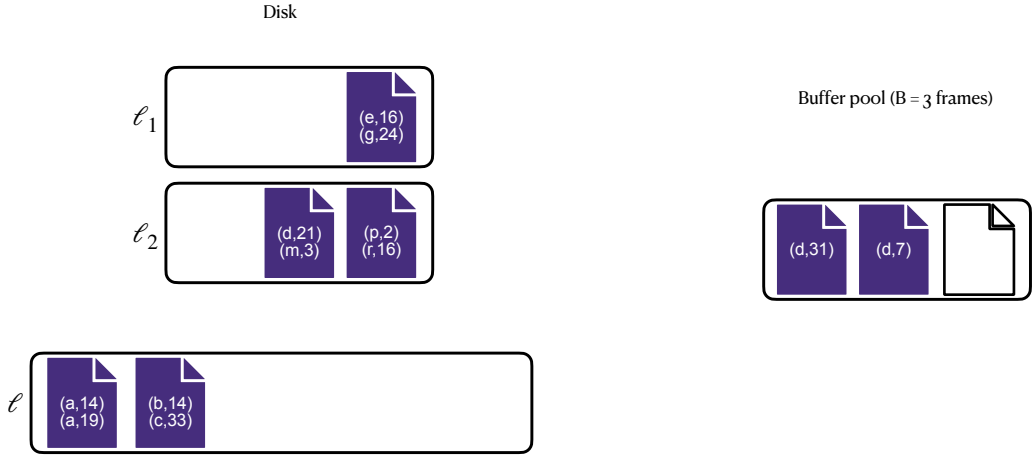
External Merge (Example)

- **Continue merging into new page until filled**



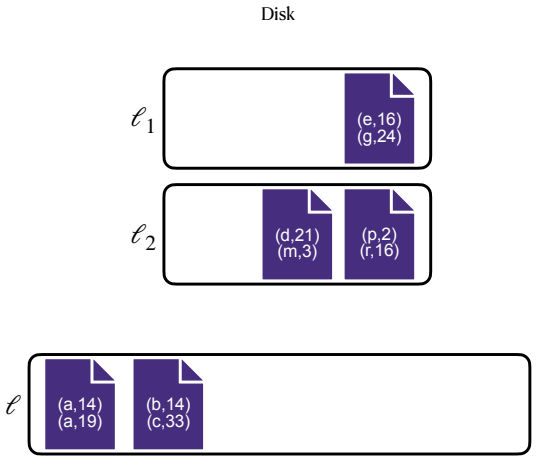
External Merge (Example)

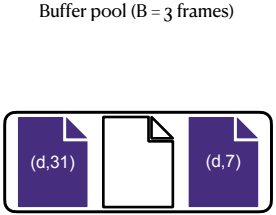
- **Write page to disk**



External Merge (Example)

- ...





Cost of External Merge

- with 3 buffer pool pages, we can merge two sorted files with an I/O cost $= 2(m + n)$
- with B buffer pool pages, we can merge $B - 1$ sorted files with the same I/O cost

External Merge Sort

Given

- Buffer pool with B frames
- Relation (file) with N pages and $N > B$

Goal

- Sort the relation on a given attribute

External Merge Sort

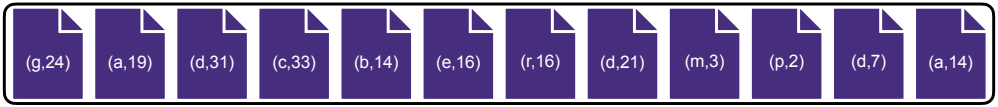
Divide-and-conquer approach

- Split the relation into separate **runs**
- Sort each run individually
- Merge runs into longer sorted runs – **passes**

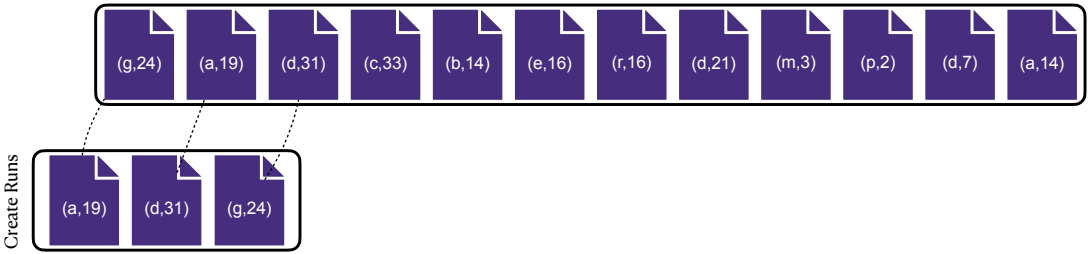
External Merge Sort comprises

1. Sorting
   - Sort chunks of data that fit in memory and write back the sorted chunks to file on disk
2. Merge
   - Merge sorted runs into larger chunks

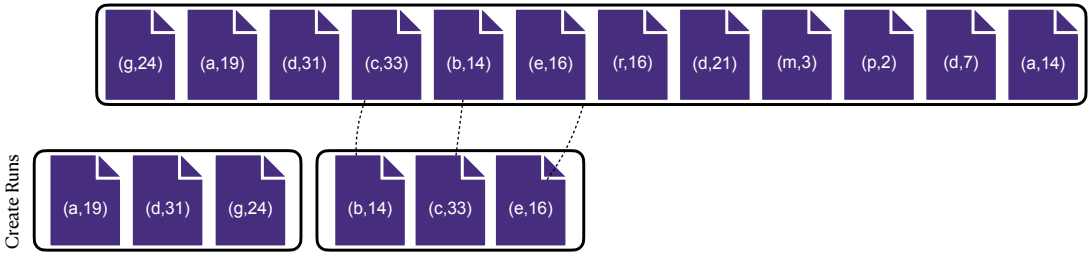
2-way External Merge Sort Example



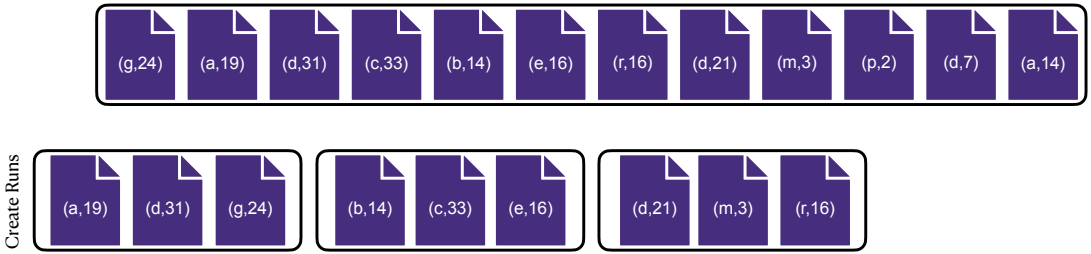
2-way External Merge Sort Example



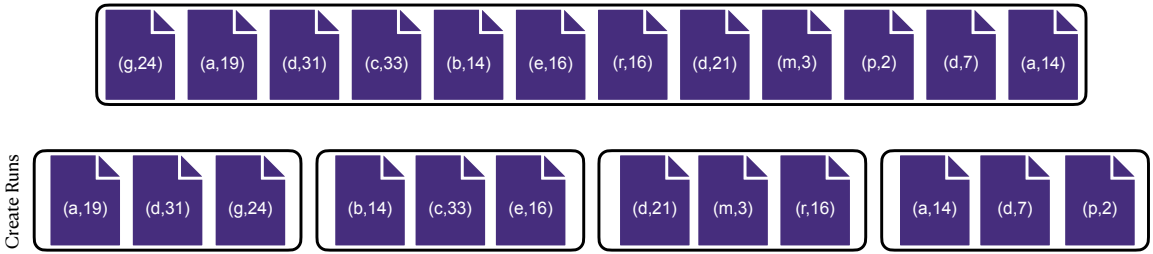
2-way External Merge Sort Example



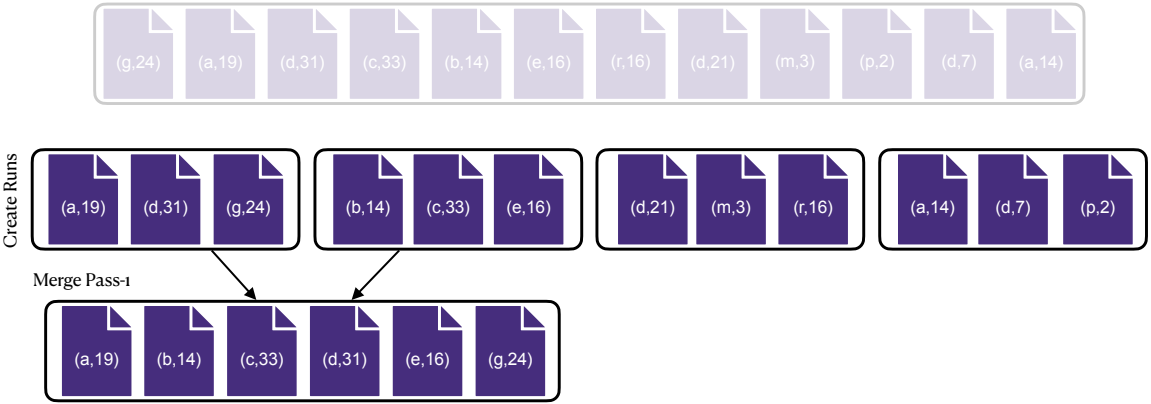
2-way External Merge Sort Example



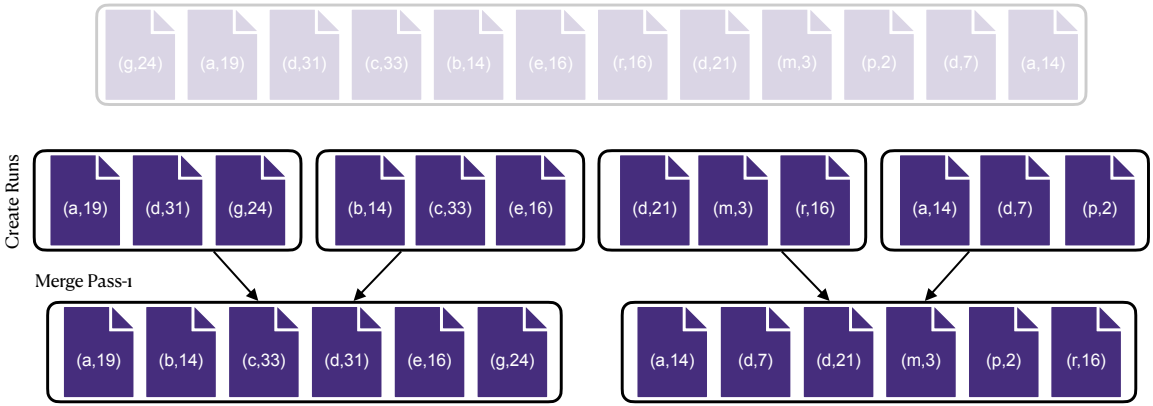
2-way External Merge Sort Example



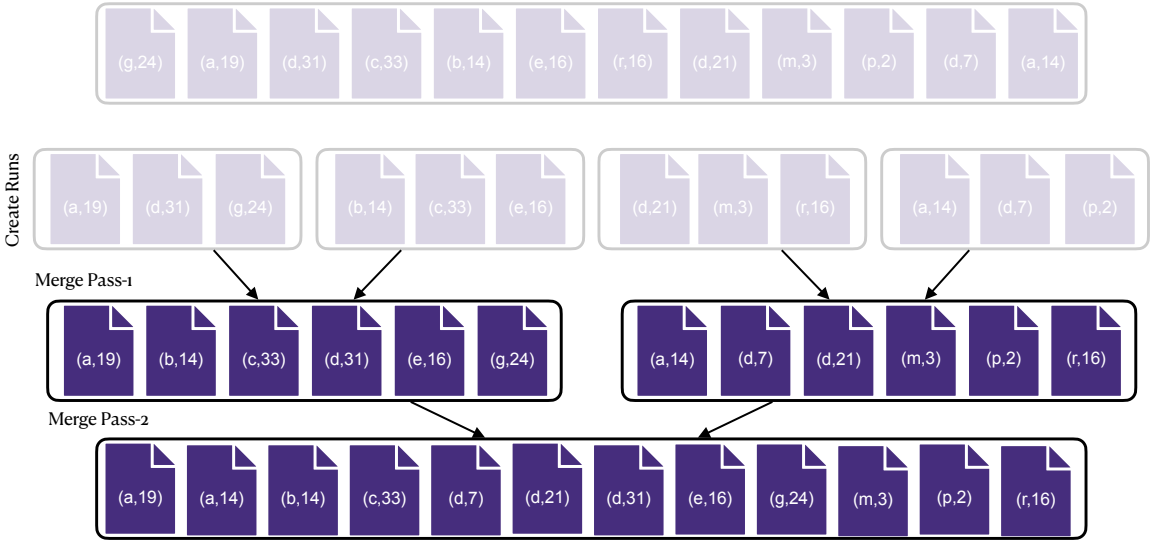
2-way External Merge Sort Example



2-way External Merge Sort Example



2-way External Merge Sort Example



2-way External Merge Sort Example Cost

I/O cost

- Each pass requires reading and writing every page in file
- Pass $0 = 12 \times (1 + 1) = 24$ I/Os
- Pass $1 = 2 \times (2 \times (3 + 3)) = 24$ I/Os
- Pass $2 = 2 \times (6 + 6) = 24$ I/Os

2-way External Merge Sort (Simplified)

- Number of passes to sort the whole file $= \lceil \log_2 n \rceil + 1$
- Number I/Os in each pass $= 2 \times n$
- Total I/O cost $= 2n(\lceil \log_2 n \rceil + 1)$

General External Merge Sort

- 2-way merge-sort only uses 3 buffer pages
- usually we have $B > 3$

General External Merge Sort

Pass 0: Use B buffer pages and produce $\lceil n/B \rceil$ sorted runs of size B

Pass 1,2,...: Merge $B - 1$ runs

General cost

- Number of passes to sort the whole file $= \lceil \log_{B-1} \lceil \frac{n}{B} \rceil \rceil + 1$
- Number I/Os in each pass $= 2 \times n$
- Total I/O cost $= 2n(\lceil \log_{B-1} \lceil \frac{n}{B} \rceil \rceil + 1)$

Homework

Fill the following table. How many passes will the external merge sort require for a file with n pages and a buffer pool with B frames?

n	B		
	4	16	256
100			
10000			
1000000			
100000000			
10000000000			

Summary

- Sorting in database requires I/O efficient algorithms
- External merge sort
  1. Sorting
  2. Merging
- Optimizations
  - Replacement sort (idea: create runs of average size $2B$)
  - Double buffering (idea: pre-fetch next run into another buffer)
  - Comparison optimizations (e.g., Suffix truncation)
  - Leverage B+Trees if available
    - Note: Unclustered index does not pay off (Why?)
  - Late materialization
    - Sorting key + Record pointers