COL 362 & COL 632

Sorting

16 Mar 2023

Why Sort ?

- A classic problem in computer science!
- Data requested in sorted order
  - e.g., find students in increasing *gpa* order

- Sorting is first step in *bulk loading* B+ tree index.
- Sorting useful for eliminating *duplicate copies* in a collection of records (Why?)
- *Many efficient join algorithms need sorting*

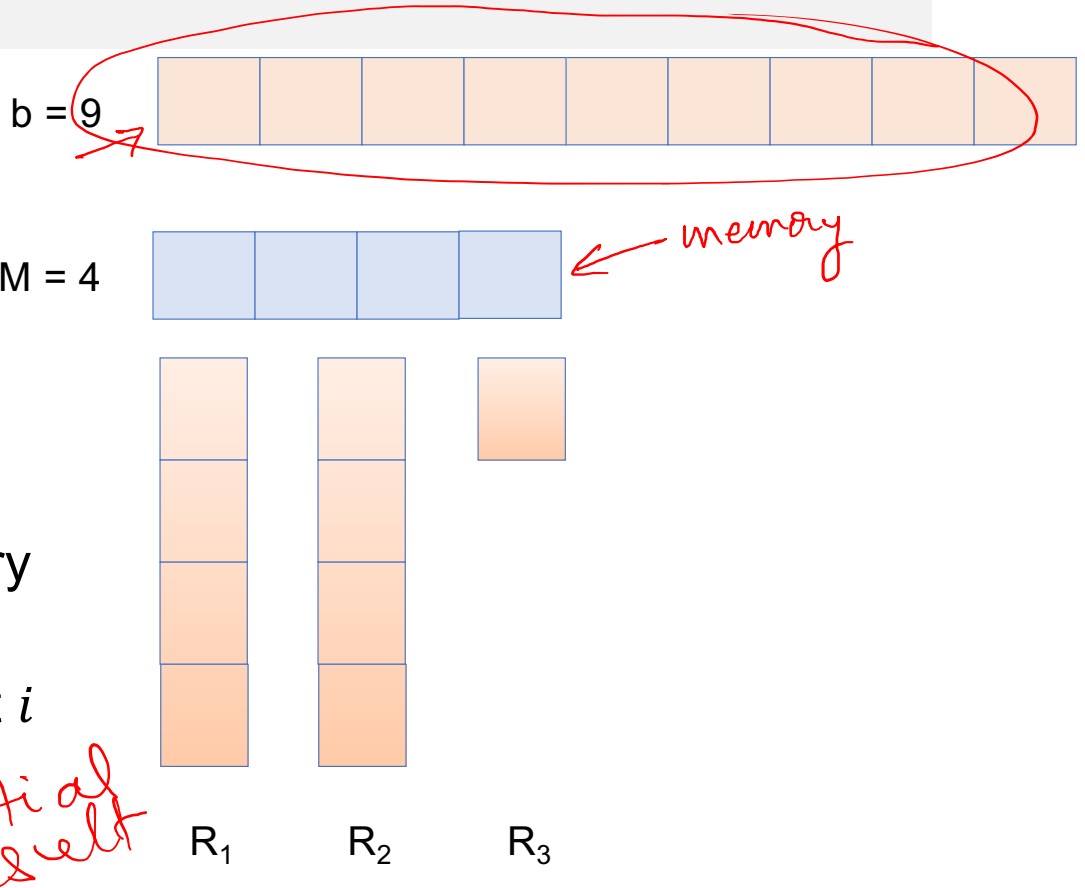
Sorting

- We can build an index on the relation, and then use the index to read the relation in sorted order.
- Scenario: Table to be sorted has B+ tree index on sort column(s).
- Idea: Can retrieve records in order by traversing leaf pages.
- **Is this a good idea?**
- B+ tree is clustered — ***Good idea!***
- B+ tree is not clustered — ***Could be a very bad idea!***

Sorting: External sort-merge (1/4)

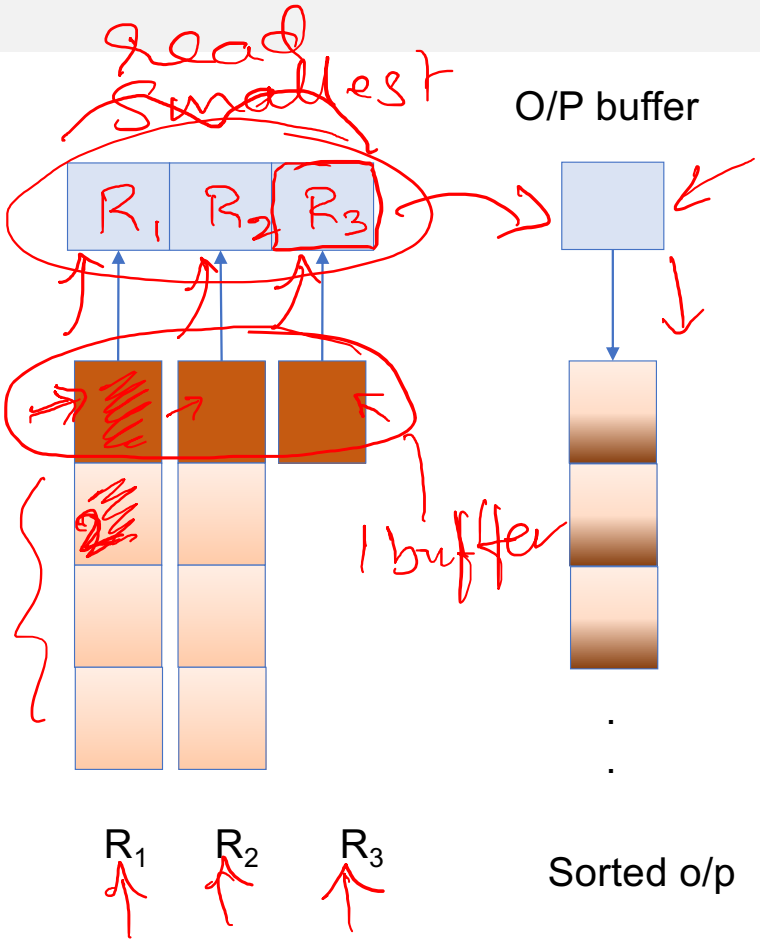
1. Let M denote memory size (in pages).
2. **Create sorted runs**. Let i be 0 initially.
   Repeatedly do the following till the end of the relation:
   1. Read M blocks of relation into memory
   2. Sort the in-memory blocks
   3. Write sorted data to run R_i; increment i

   Let the final value of i be N
3. *Merge the runs*



Sorting: External sort-merge (2/4)

2. **Merge the runs (N-way merge)**. We assume (for now) that $N < M$.
   1. Use N blocks of memory to buffer input runs
   2. 1 block to buffer output
   3. **repeat**
      1. Select the first record (in sort order) among all buffer pages
      2. Write the record to the output buffer. If the output buffer is full write it to disk.
      3. Delete the record from its input buffer page. **If** the buffer page becomes empty, **then** read the next block (if any) of the run into the buffer.
   4. **until** all input buffer pages are empty:

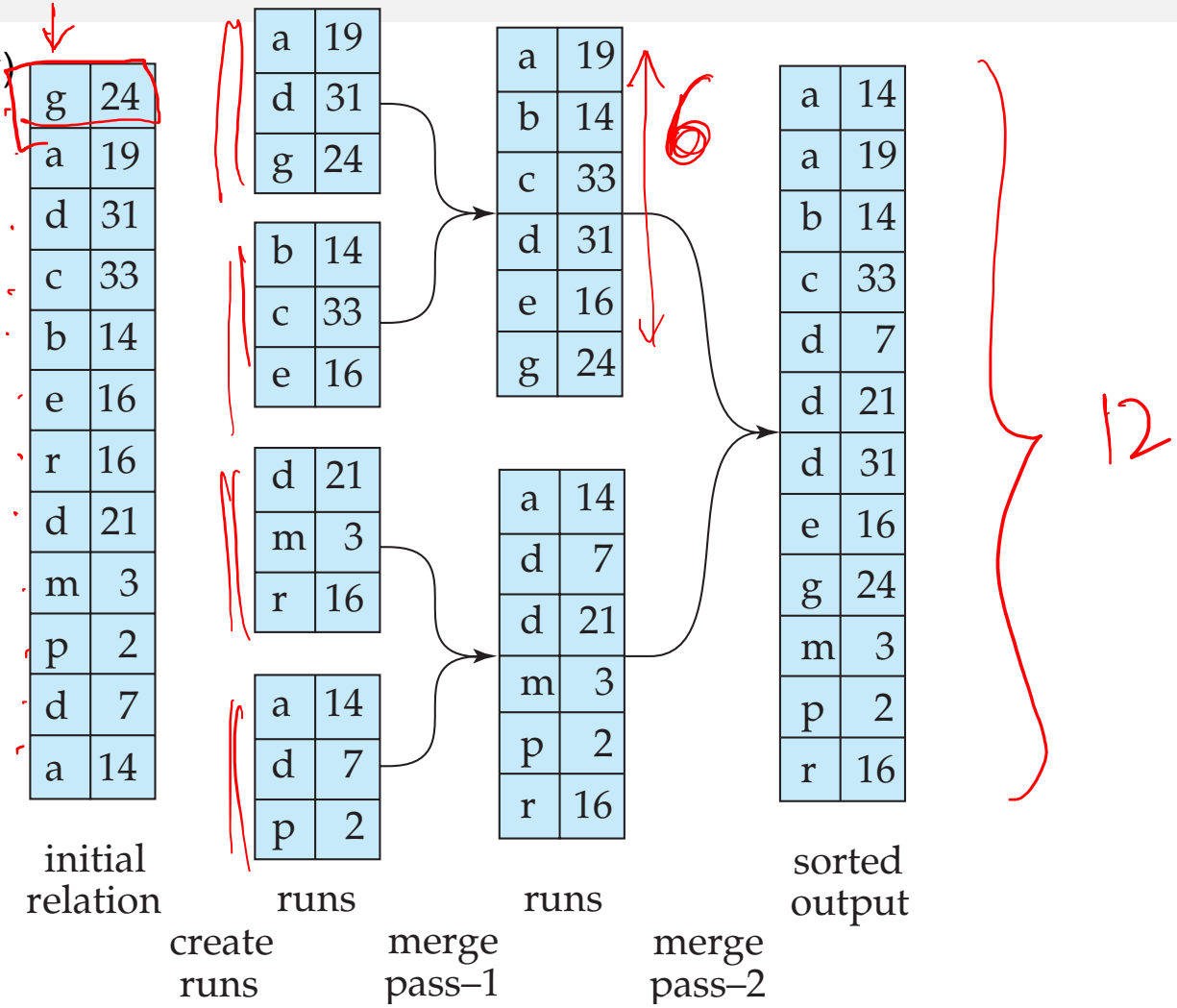


Sorting: External sort-merge (3/4)

- If $N \geq M$, several merge *passes* are required.
- In each pass, contiguous groups of M - 1 runs are merged.
- A pass reduces the number of runs by a factor of M -1 and creates runs longer by the same factor.
- Repeated passes are performed till all runs have been merged into one.

Sorting: External sort-merge (4/4)

M = 3 (no. of blocks in memory)



Reducing the Seeks

- 1 block per run leads to too many seeks during merge
- Instead use b_b buffer blocks per run
  => read/write b_b blocks at a time from each run
- Can merge $\left\lfloor \frac{M}{b_b} \right\rfloor - 1$ runs in one pass
- Total number of merge passes required:

$$\left\lceil \log_{\left\lfloor \frac{M}{b_b} \right\rfloor - 1} \frac{b_r}{M} \right\rceil$$