External Sorting

Why Sort?

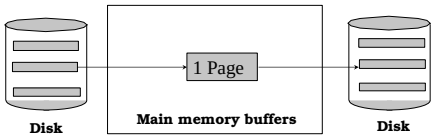
- Used for eliminating duplicates
  - Select DISTINCT …
- Bulk loading B+ trees
  - Need to first sort leaf level pages
- Data requested in sorted order
  - SELECT S.name
    FROM Sailor S
    ORDER BY S.age
- Some join algorithms use sorting
  - Sort-merge join

Sorting: Main Challenge

- Sort 1 TB of data with 1 GB of RAM
- Why not just use QuickSort? (i.e., simply map disk pages to virtual memory)

2-Way External Merge Sort

- Phase 1: Read a page at a time, sort it, write it
  - Only one buffer page used

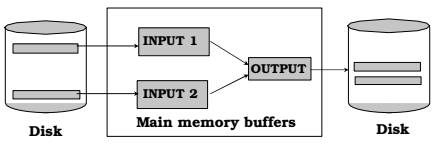


Two-Way External Merge Sort: Phase 1

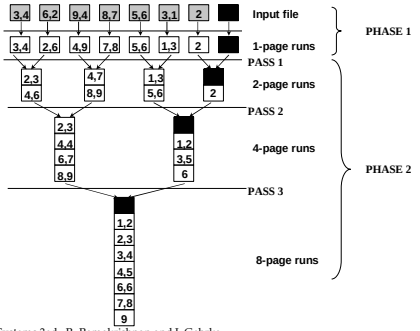
- Assume input file with N data pages
- What is the cost of Phase 1 (in terms of # I/Os)?

2-Way External Merge Sort

- Phase 2: Make multiple passes to merge runs
  - Pass 1: Merge two runs of length 1 (page)
  - Pass 2: Merge two runs of length 2 (pages)
  - … until 1 run of length N
  - Three buffer pages used



2-Way External Merge Sort



2-Way External Merge Sort: Analysis

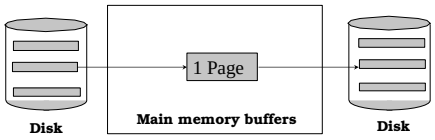
- Total I/O cost for sorting file with N pages
- Cost of Phase 1 = $2N$
- Number of passes in Phase 2 = $\lceil \log_2 N \rceil$
- Cost of each pass in Phase 2 = $2N$
- Cost of Phase 2 = $2N \times \lceil \log_2 N \rceil$
- Total cost = $2N\left(\lceil \log_2 N \rceil + 1\right)$

General External Merge Sort: Motivation

- 2-Way merge sort uses at most 3 buffer pages
- What if more buffer pages were available?
- Can we use these extra buffer pages to reduce sorting cost?
  - Specifically, how would Phases 1 and 2 change?

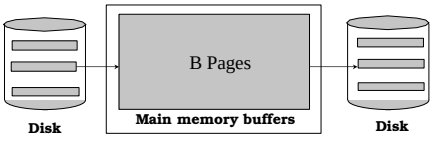
2-Way External Merge Sort

- Phase 1: Read a page at a time, sort it, write it
  - Only one buffer page used
- How can this be modified if B buffer pages are available?



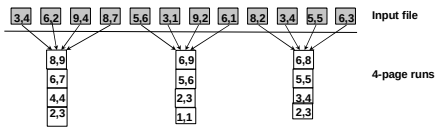
General External Merge Sort

- Phase 1: Read B pages at a time, sort B pages in main memory, and write out B pages
- Length of each run = B pages
- Assuming N input pages, number of runs = N/B
- Cost of Phase 1 = $2N$



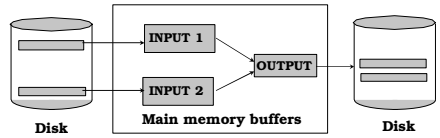
General External Merge Sort: Phase 1

- # buffer pages B = 4



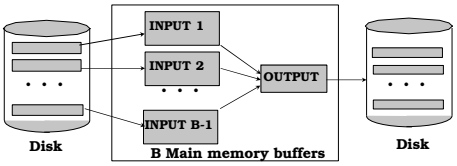
2-Way External Merge Sort

- Phase 2: Make multiple passes to merge runs
  - Pass 1: Merge two runs of length 1 (page)
  - Pass 2: Merge two runs of length 2 (pages)
  - … until 1 run of length N
  - Three buffer pages used
- How can this be modified if B buffer pages available?



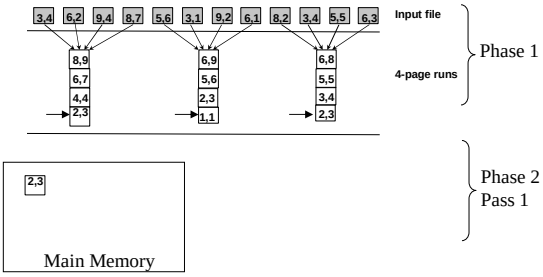
General External Merge Sort

- Phase 2: Make multiple passes to merge runs
  - Pass 1: Produce runs of length B(B-1) pages
  - Pass 2: Produce runs of length $B(B-1)^2$ pages
  - …
  - Pass P: Produce runs of length $B(B-1)^P$ pages



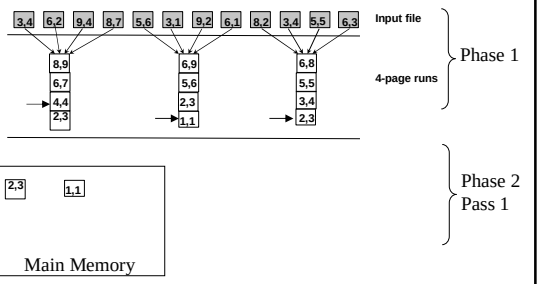
General External Merge Sort: Phase 2

- # buffer pages B = 4



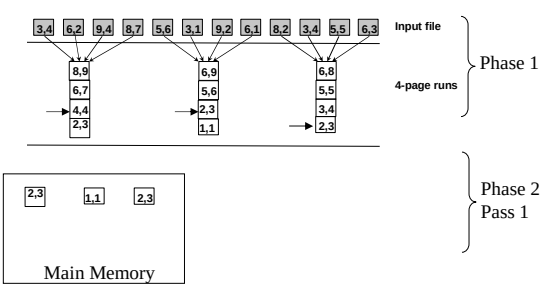
General External Merge Sort: Phase 2

- # buffer pages B = 4



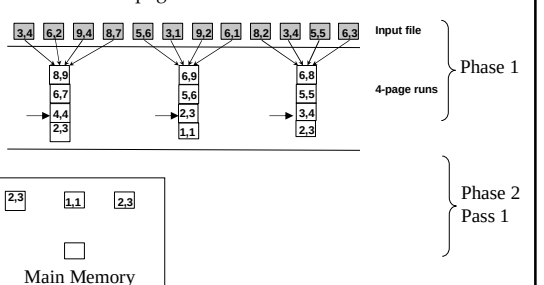
General External Merge Sort: Phase 2

- # buffer pages B = 4



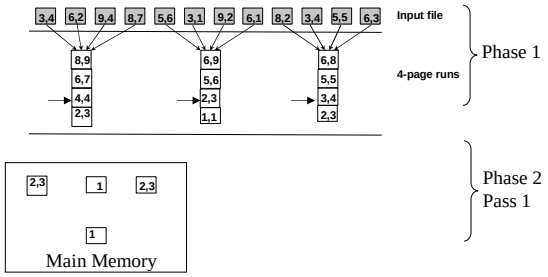
General External Merge Sort: Phase 2

- # buffer pages B = 4



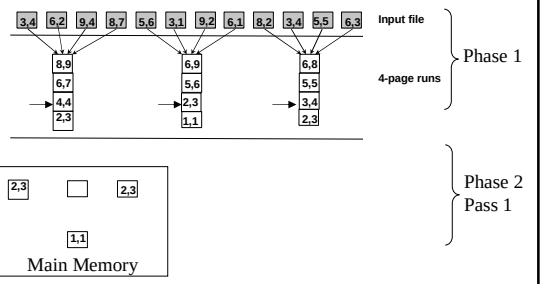
General External Merge Sort: Phase 2

- # buffer pages B = 4



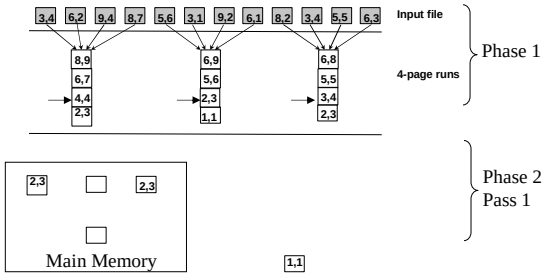
General External Merge Sort: Phase 2

- # buffer pages B = 4



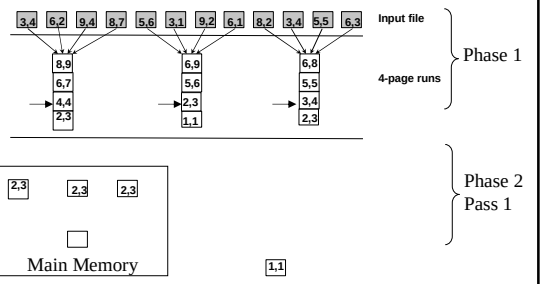
General External Merge Sort: Phase 2

- # buffer pages B = 4



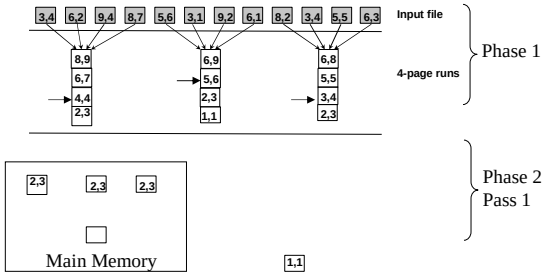
General External Merge Sort: Phase 2

- # buffer pages B = 4



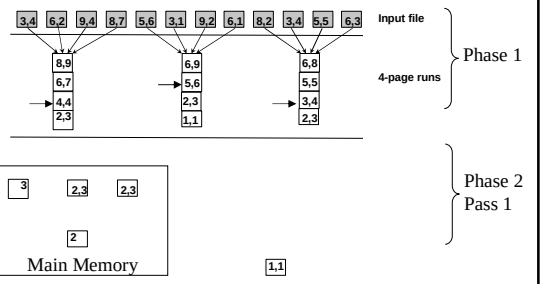
General External Merge Sort: Phase 2

- # buffer pages B = 4



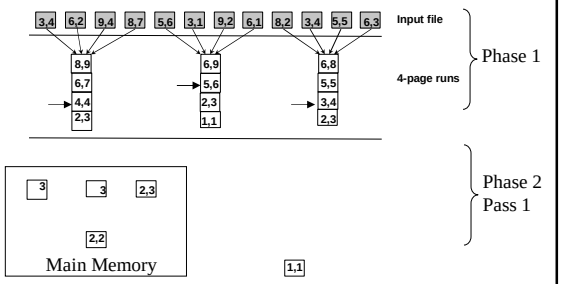
General External Merge Sort: Phase 2

- # buffer pages B = 4



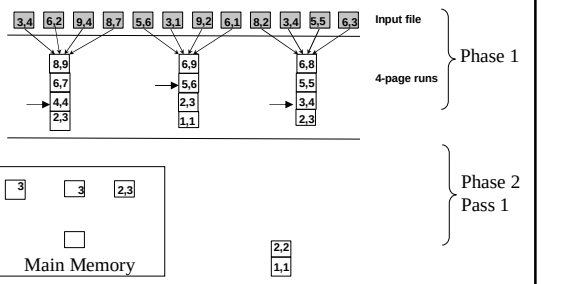
General External Merge Sort: Phase 2

- # buffer pages B = 4



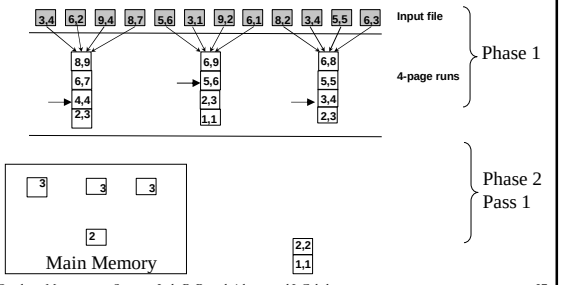
General External Merge Sort: Phase 2

- # buffer pages B = 4



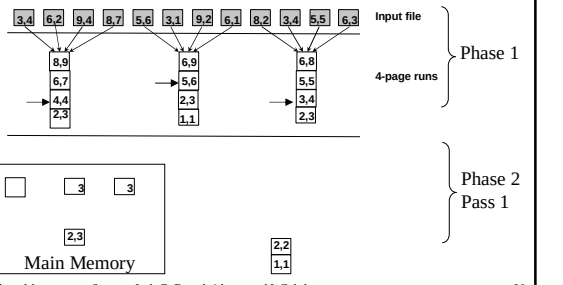
General External Merge Sort: Phase 2

- # buffer pages B = 4



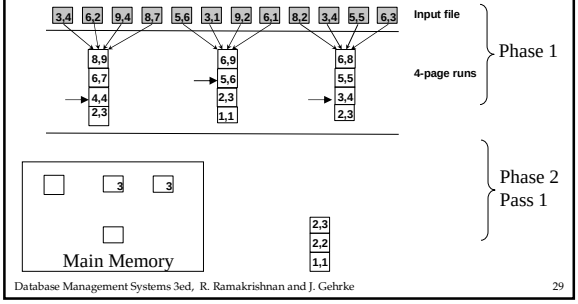
General External Merge Sort: Phase 2

- # buffer pages B = 4



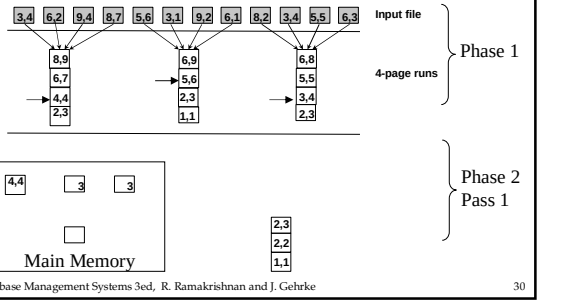
General External Merge Sort: Phase 2

- # buffer pages B = 4



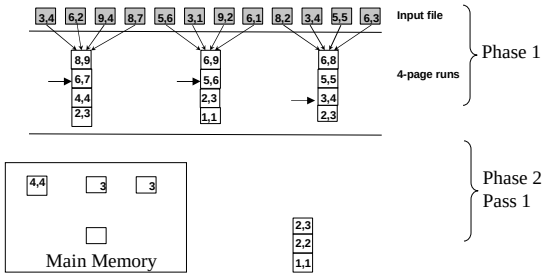
General External Merge Sort: Phase 2

- # buffer pages B = 4



General External Merge Sort: Phase 2

- # buffer pages B = 4



General External Merge Sort: Analysis

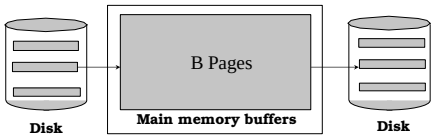
- Total I/O cost for sorting file with N pages
- Cost of Phase 1 = $2N$
- If # passes in Phase 2 is P then: $B(B-1)^P = N$
- Therefore P = $\lceil \log_{B-1} \lceil N/B \rceil \rceil$
- Cost of each pass in Phase 2 = $2N$
- Cost of Phase 2 = $2N \times \lceil \log_{B-1} \lceil N/B \rceil \rceil$
- Total cost = $2N(\lceil \log_{B-1} \lceil N/B \rceil \rceil + 1)$

Number of Passes of External Sort

N	B=3	B=5	B=9	B=17	B=129	B=257
100	7	4	3	2	1	1
1,000	10	5	4	3	2	2
10,000	13	7	5	4	2	2
100,000	17	9	6	5	3	3
1,000,000	20	10	7	5	3	3
10,000,000	23	12	8	6	4	3
100,000,000	26	14	9	7	4	4
1,000,000,000	30	15	10	8	5	4

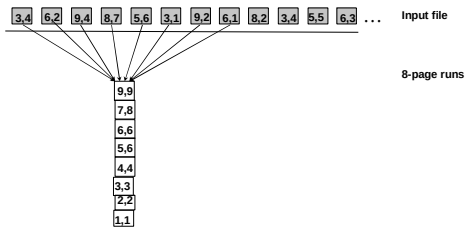
External Merge Sort: Optimizations

- Phase 1: Can produce runs of length 2B using only B buffer pages (in expected sense)!
  - Use variant of Heapsort instead of Quicksort
- Total sorting cost = $2N(\lceil \log_{B-1} \lceil N/2B \rceil \rceil + 1)$



General External Merge Sort: Phase 1

- # buffer pages B = 4

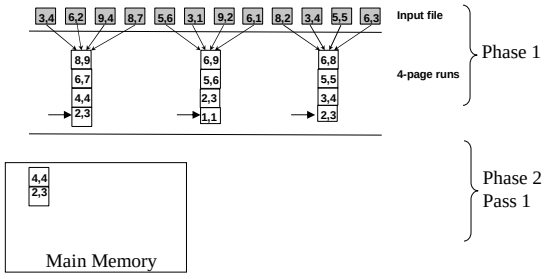


External Merge Sort: Optimizations

- Currently, do one page I/O at a time
- But can read/write a *block* of pages sequentially!
  - Make each buffer input/output a block of pages
  - Better read performance
- Possible negative impact?

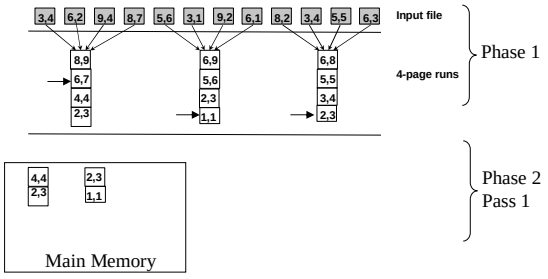
General External Merge Sort: Phase 2

- # buffer pages B = 8



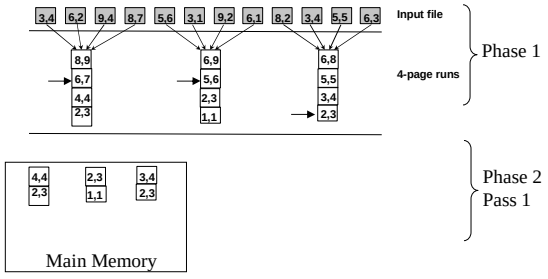
General External Merge Sort: Phase 2

- # buffer pages B = 8



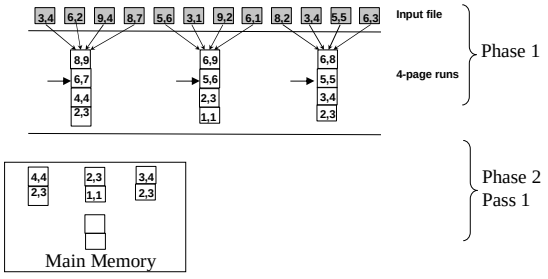
General External Merge Sort: Phase 2

- # buffer pages B = 8



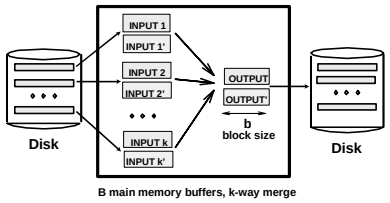
General External Merge Sort: Phase 2

- # buffer pages B = 8



General Merge Sort: Optimizations

- Double buffering: to reduce I/O wait time, *prefetch* into `shadow block'.
  - Again, potentially more passes; in practice, most files *still* sorted in 2-3 passes.



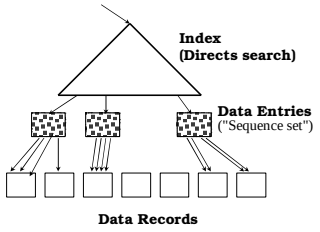
Using B+ Trees for Sorting

- Scenario: Table to be sorted has B+ tree index on sorting column(s).
- Idea: Can retrieve records in order by traversing leaf pages.
- ***Is this a good idea?***

- Cases to consider:
  - B+ tree is clustered — ***Good idea!***
  - B+ tree is not clustered — ***Could be a very bad idea!***

Clustered B+ Tree Used for Sorting

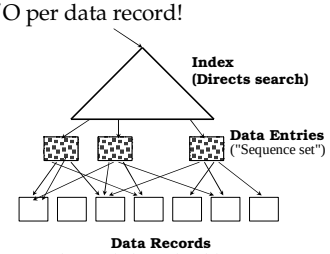
- Cost: root to the left-most leaf, then retrieve all leaf pages (Alternative 1)
- If Alternative 2 is used? Additional cost of retrieving data records: each page fetched just once.



✉ ***Always better than external sorting!***

Unclustered B+ Tree Used for Sorting

- Alternative (2) for data entries; each data entry contains *rid* of a data record. In general, one I/O per data record!



External Sorting vs. Unclustered Index

N	Sorting	p=1	p=10	p=100
100	200	100	1,000	10,000
1,000	2,000	1,000	10,000	100,000
10,000	40,000	10,000	100,000	1,000,000
100,000	600,000	100,000	1,000,000	10,000,000
1,000,000	8,000,000	1,000,000	10,000,000	100,000,000
10,000,000	80,000,000	10,000,000	100,000,000	1,000,000,000

✉ ***p*: # of records per page**
✉ ***B=1,000 and block size=32 for sorting***
✉ ***p=100 is the more realistic value.***

Sorting Records!

- Sorting has become a blood sport!
  - Parallel sorting is the name of the game ...
- Datamation: Sort 1M records of size 100 bytes
  - Typical DBMS: 15 minutes
  - World record: 3.5 ***seconds***
    - 12-CPU SGI machine, 96 disks, 2GB of RAM
- New benchmarks proposed:
  - Minute Sort: How many can you sort in 1 minute?
  - Dollar Sort: How many can you sort for $1.00?
    - PennySort!!!