External Sorting

Why Sort?

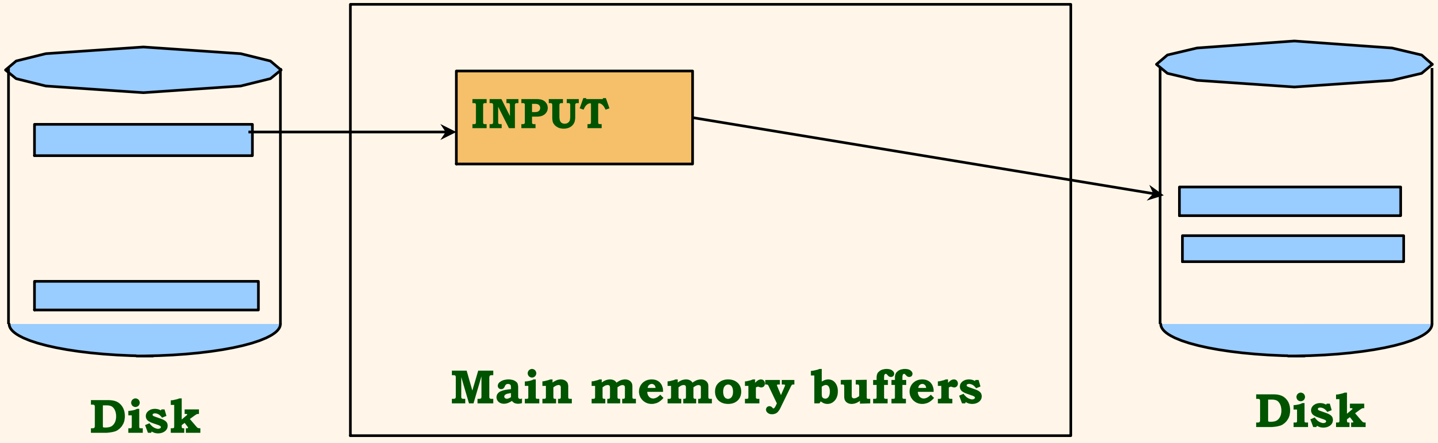
- A classic problem in computer science!
- Data requested in sorted order
  - e.g., find students in increasing *gpa* order
- Sorting is first step in *bulk loading* B+ tree index.
- Sorting useful for eliminating *duplicate copies* in a collection of records
- *Sort-merge* join algorithm involves sorting.

Using secondary storage effectively

- General Wisdom :
  - I/O costs dominate
  - Design algorithms to reduce I/O

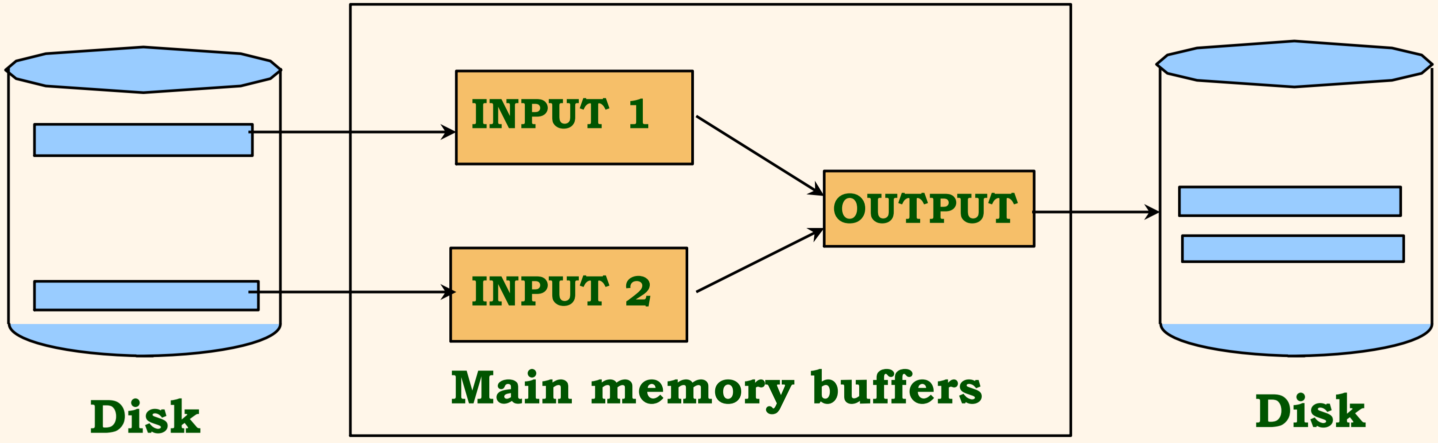
2-Way Sort: Requires 3 Buffers

- Phase 1: PREPARE.
  - Read a page, sort it, write it.
  - only one buffer page is used



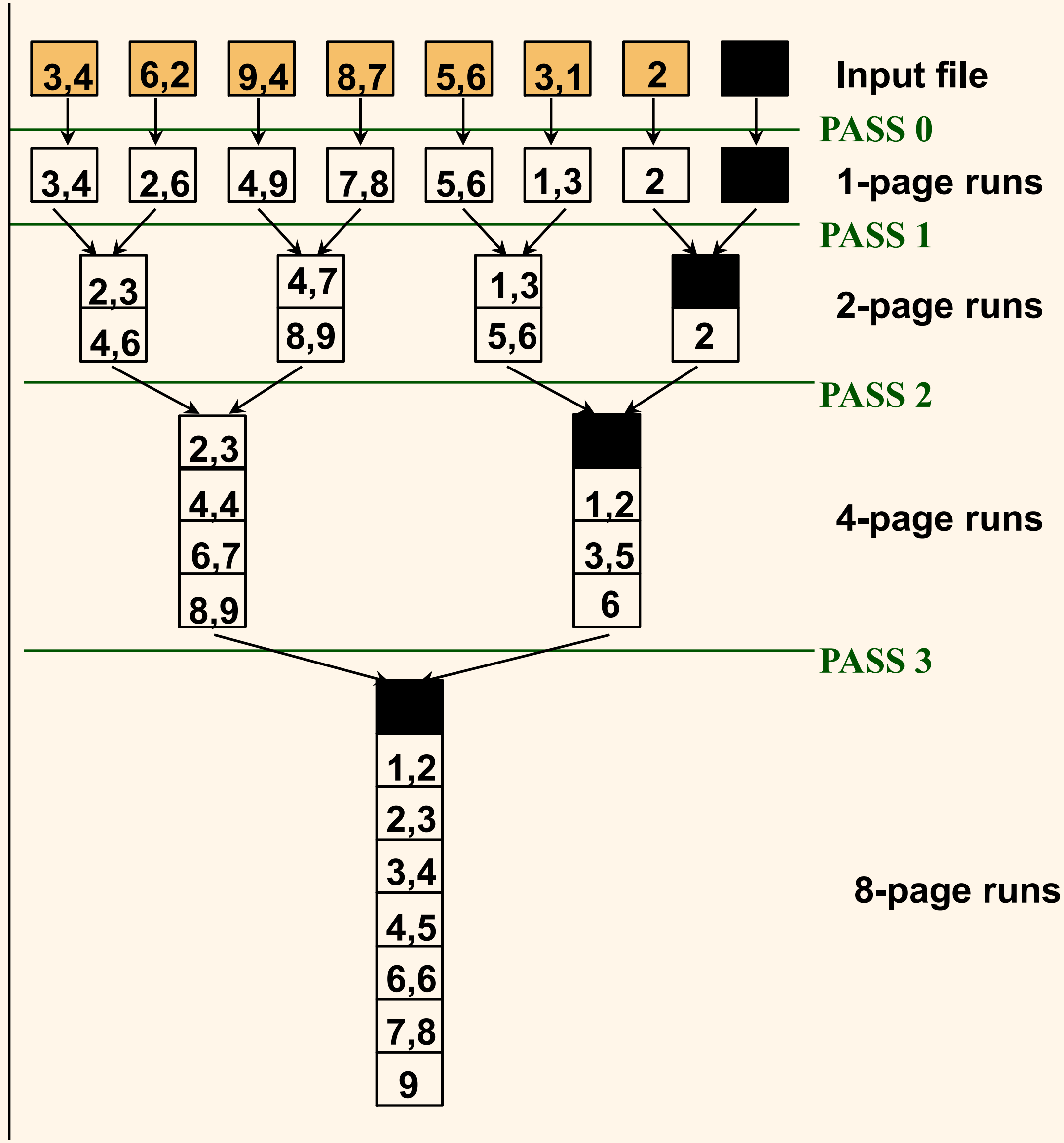
2-Way Sort: Requires 3 Buffers

- Phase 1: PREPARE.
  - Read a page, sort it, write it.
  - only one buffer page is used
- Phase 2, 3, …, etc.: MERGE:
  - three buffer pages used.



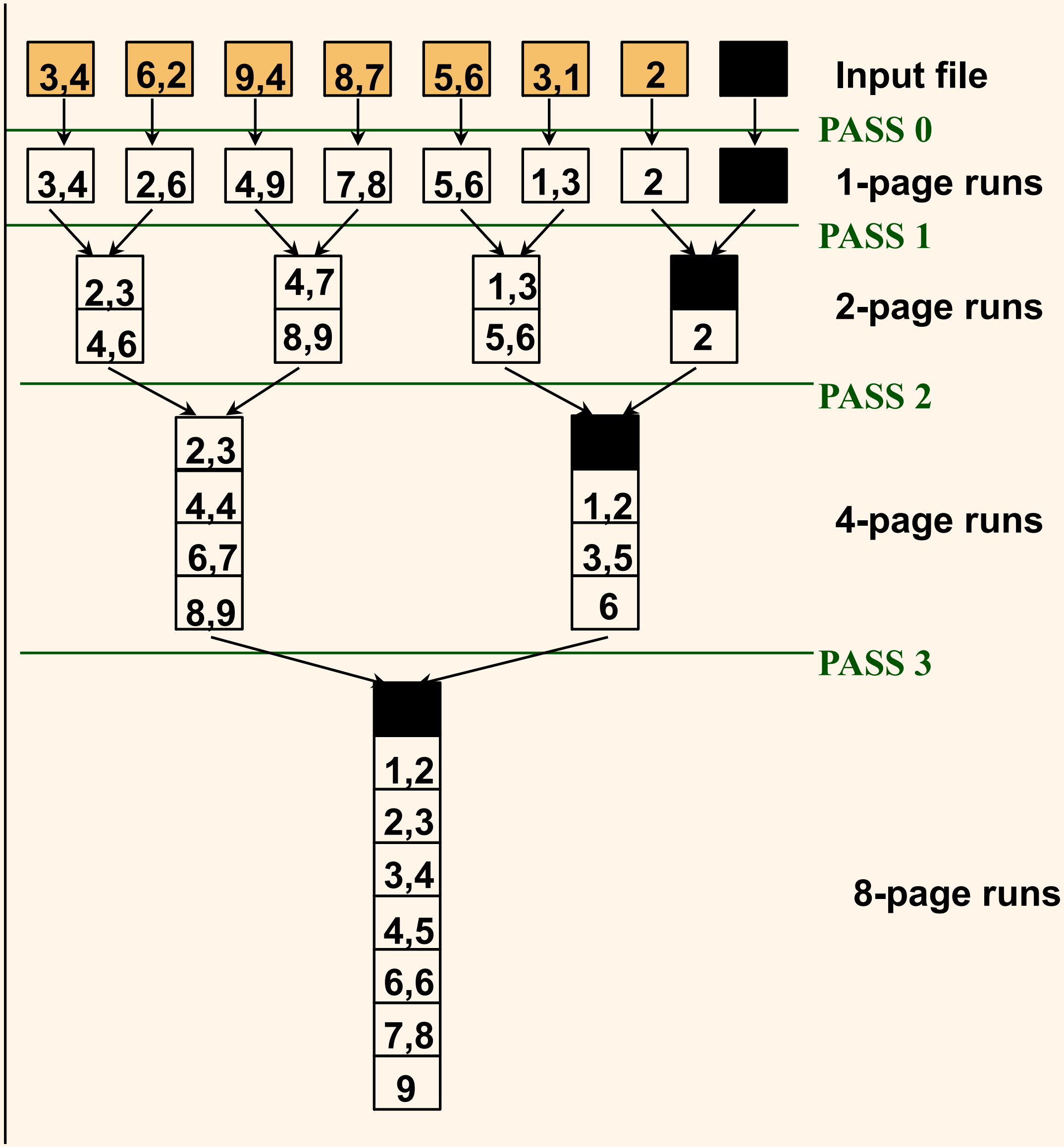
Two-Way External Merge Sort

- *Idea:* ***Divide and conquer:*** sort subfiles and merge into larger sorts



Two-Way External Merge Sort

- Costs for pass : all pages
- # of passes : height of tree
- Total cost : product of above



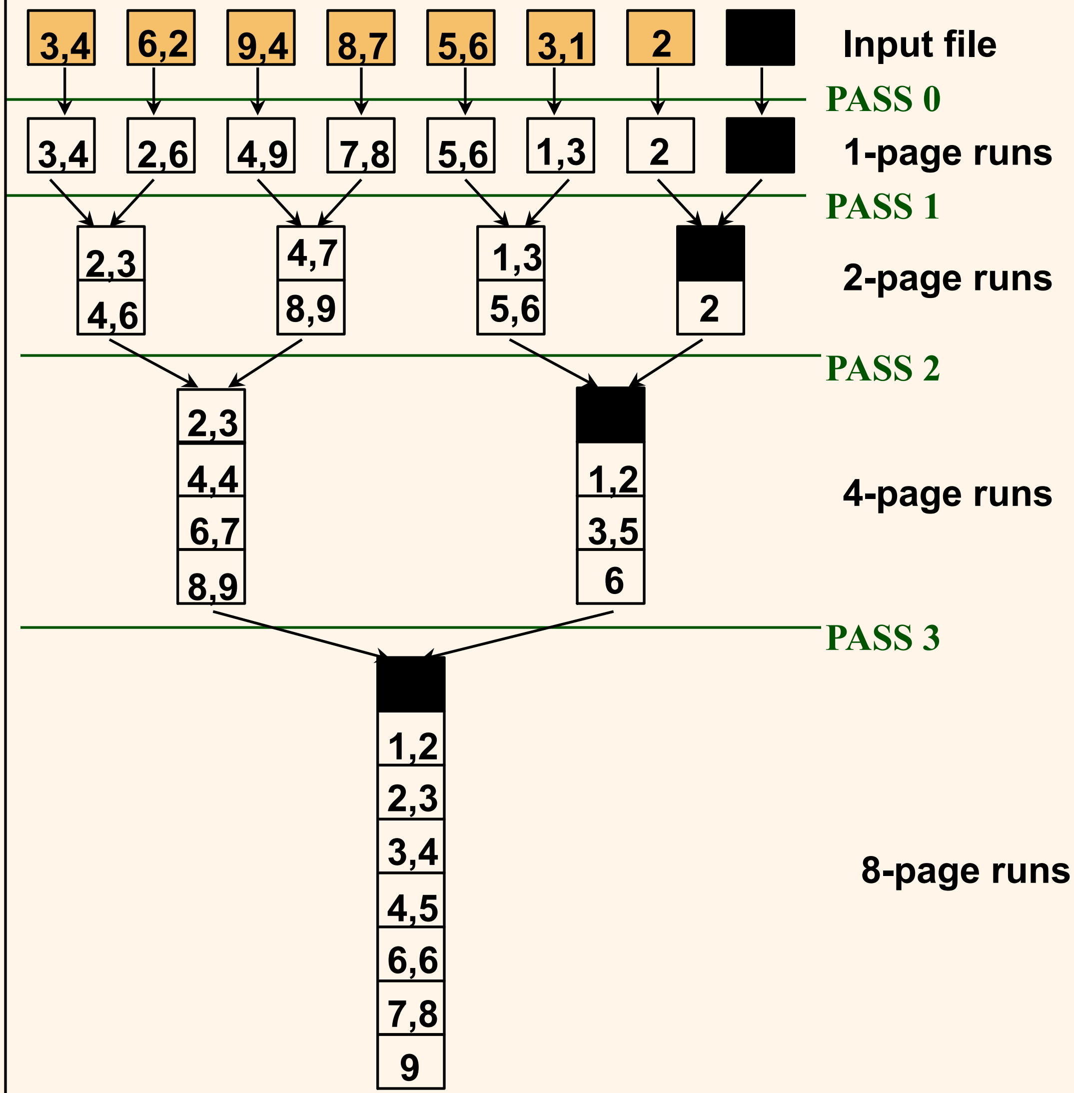
Two-Way External Merge Sort

- Each pass we read + write each page in file.
- N pages in file => 2N
- Number of passes

$$=\lceil \log_2 N \rceil + 1$$

- So total cost is:

$$2N\left(\lceil \log_2 N \rceil + 1\right)$$

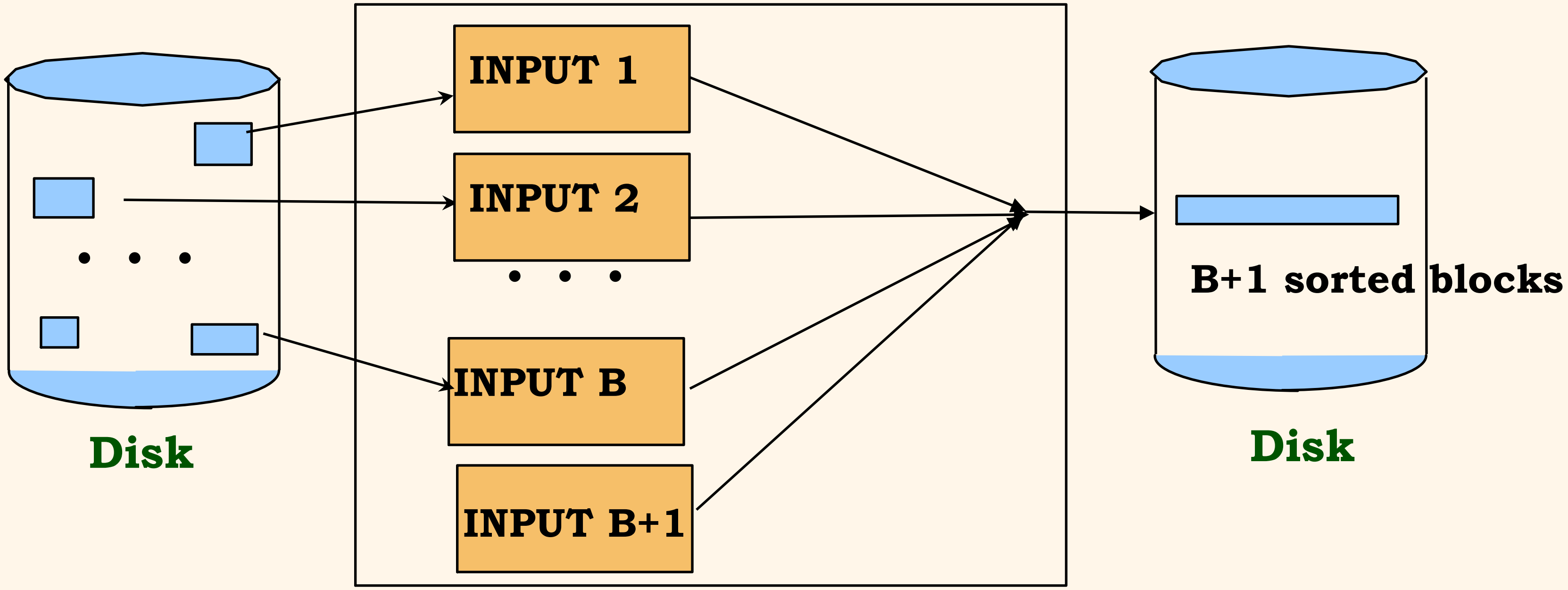


External Merge Sort

- What if we had more buffer pages?
- How do we utilize them wisely ?

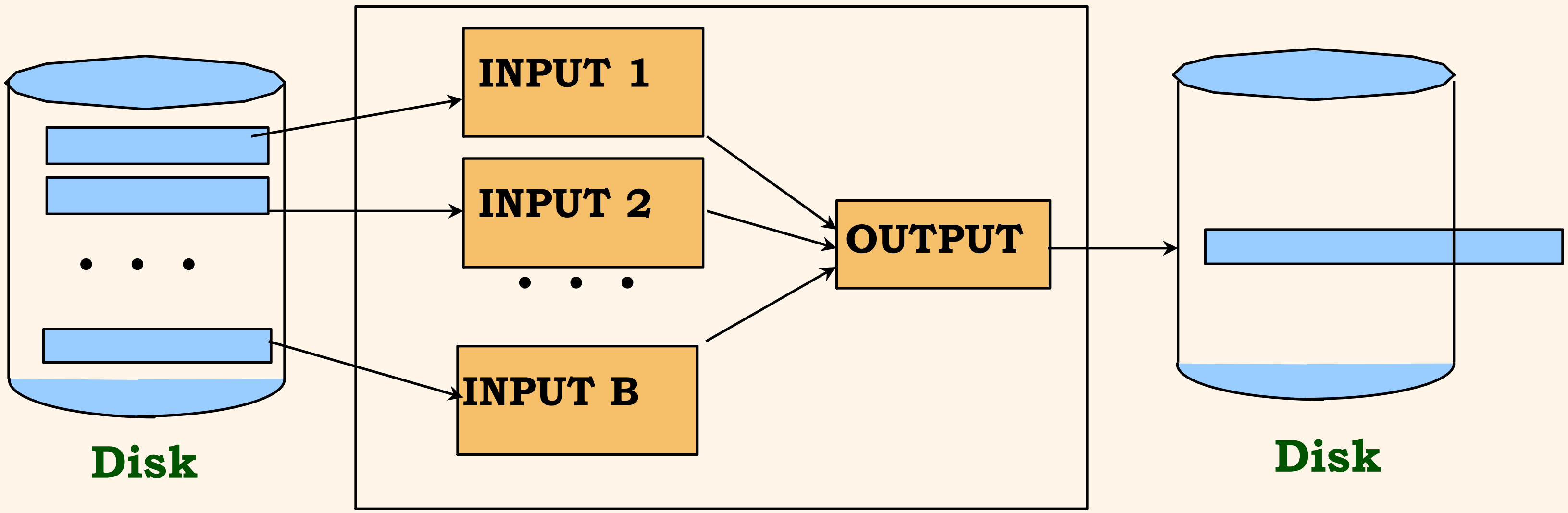
-→ Two main ideas !

Phase 1 : Prepare



- The B+1 blocks are sorted in memory as a whole
- The output block is not needed in this phase and can be used to hold and sort $(B+1)^{th}$ block
- Each run (output file) consists of B+1 blocks

Phase 2 : Merge

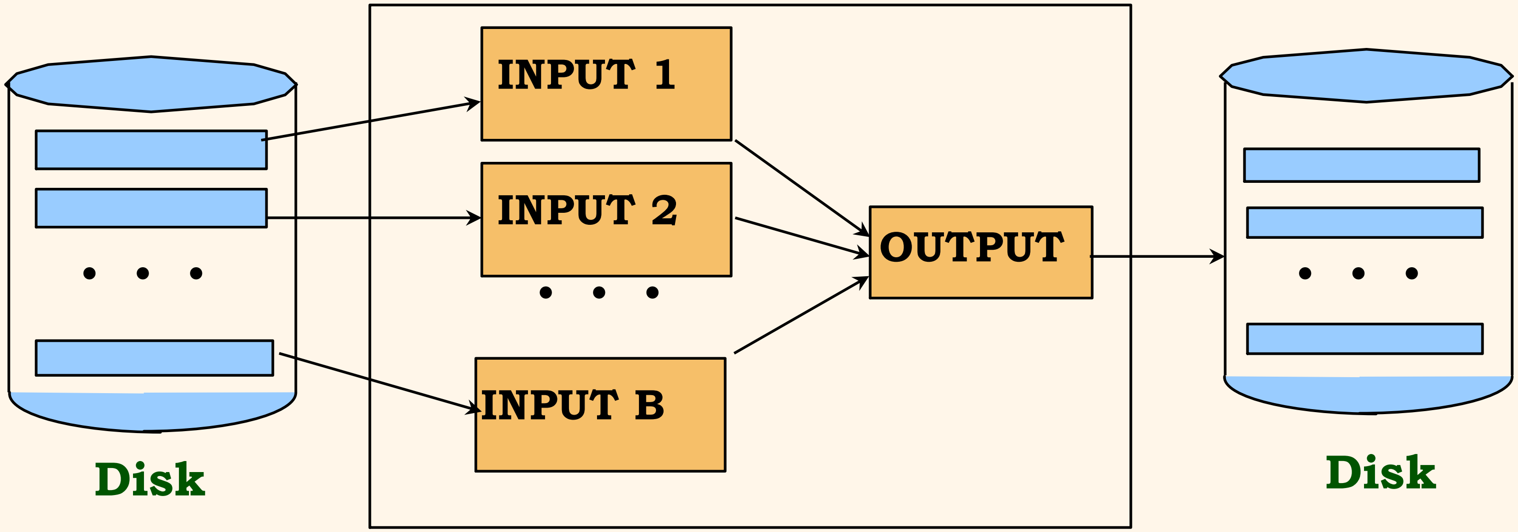


Compose as many sorted sublists into one long sorted list.

General External Merge Sort

☛ ***How can we utilize more than 3 buffer pages?***

- To sort a file with N pages using B+*1* buffer pages:
  - Pass 0: use B buffer pages.
    Produce $\lceil N / B \rceil$ sorted runs of B pages each.
  - Pass 1, 2, …, etc.: merge B runs.



Cost of External Merge Sort

- Number of passes: $1 + \lceil \log_{B-1} \lceil N / B \rceil \rceil$
- Cost = 2N * (# of passes)

Example

- Buffer : with 5 buffer pages
- File to sort : 108 pages

  - Pass 0:
    - Size of each run?
    - Number of runs?

  - Pass 1:
    - Size of each run?
    - Number of runs?

  - Pass 2: ???

Example

- Buffer : with 5 buffer pages
- File to sort : 108 pages
  - Pass 0: $\lceil 108 / 5 \rceil$ = 22 sorted runs of 5 pages each (last run is only 3 pages)
  - Pass 1: $\lceil 22 / 4 \rceil$ = 6 sorted runs of 20 pages each (last run is only 8 pages)
  - Pass 2: 2 sorted runs, 80 pages and 28 pages
  - Pass 3: Sorted file of 108 pages

- Total I/O costs: ?

Example

- Buffer : with 5 buffer pages
- File to sort : 108 pages

  - Pass 0: $\lceil 108 / 5 \rceil$ = 22 sorted runs of 5 pages each (last run is only 3 pages)
  - Pass 1: $\lceil 22 / 4 \rceil$ = 6 sorted runs of 20 pages each (last run is only 8 pages)
  - Pass 2: 2 sorted runs, 80 pages and 28 pages
  - Pass 3: Sorted file of 108 pages

- Total I/O costs: 2*N * (4 passes)

Number of Passes of External Sort

- gain of utilizing all available buffers
- importance of a high fan-in during merging

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