Recap: Where are we, structurally, in the organization of the course.

- So far: All about *access*… and what?
  – Recall what is an **access path**?

Recap: We know…

- Where/how to place data on the disk,
  1. Row store: sorted rows in blocks
  2. Parallel partitioning
  3. Column store
  4. Key, value store

- How to get the data off the disk.
  1. B__________
  2. B__________
  3. B__________

Recap: Where are we, structurally, in the organization of the course.

✓Storage Management

Next Section

- Given a [large] quantity of disk resident data
  i.e. | Data | > | Memory |

- How to implement database functions

Had a Peek: B+ trees

- Fat fanout. → shortens the tree height
- Primary index, leverages sorted data

  → Sparse Index

  → Shorten the tree

Why shorten the tree? Fewer accesses

Two Phase Multiway Merge Sort

a.k.a. External, Two Phase Sort

Objectives:

- External (of RAM) Algorithm
  - explicitly understand and incorporate movement of data into the algorithm
- The algorithm

Reading: Text 15.4.1

Reminder:

Simple I/O cost models – and how they fail

I/O models:

- Linear:
  - 1 average seek per block
    - n blocks, f(n) = cn, c = average seek time
- Affine:
  - seek first block,
  - weighted average of rotational latency + track to track seek time for each additional block
    - n block f(n) = c + c' (n-1)
- and how they fail
  - buffering (hardware and software)
  - prefetch

Text & Class Cost Model

- B(R) number of blocks of relation R
- T(R) number of tuples [rows] in relation R

Count the number of blocks read

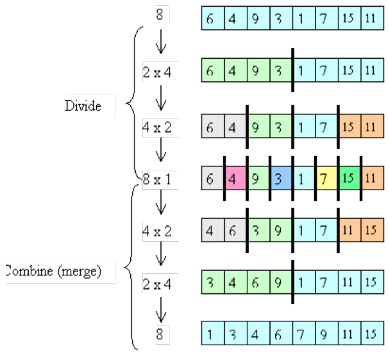
- Not yet: commonly, no cost for writing output.

A word about notation

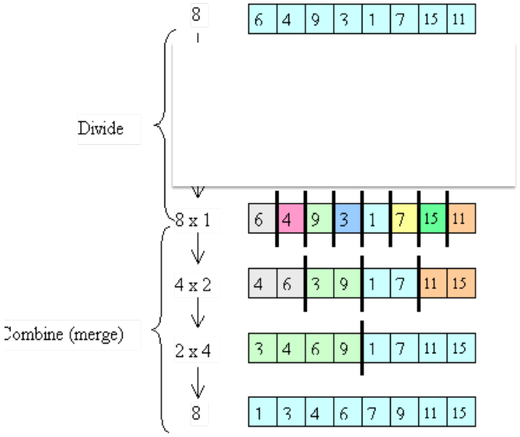
- text & class: B(R) and T(R)
- common in the literature
  - T(R) = | R | // number or rows/tuples
  - B(R) = || R || // number of blocks/pages

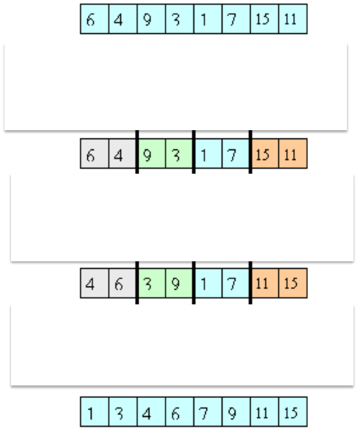
Two Phase Multiway Merge Sort

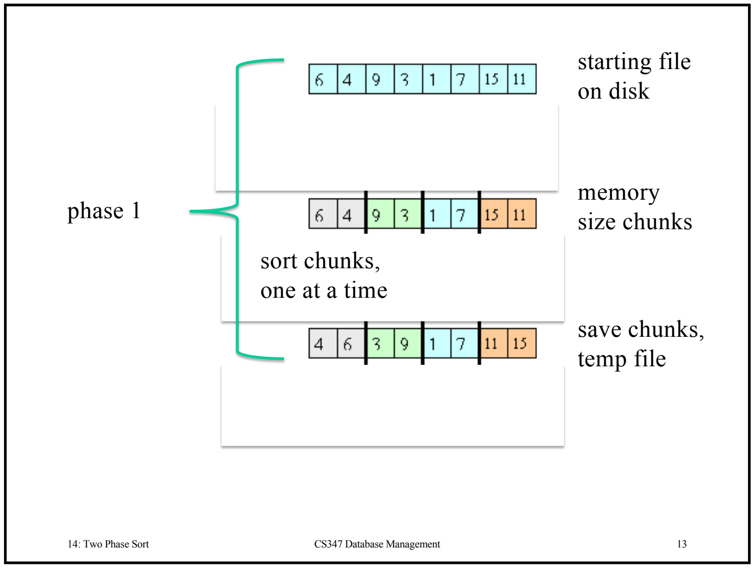
Sort-of Start With Merge Sort

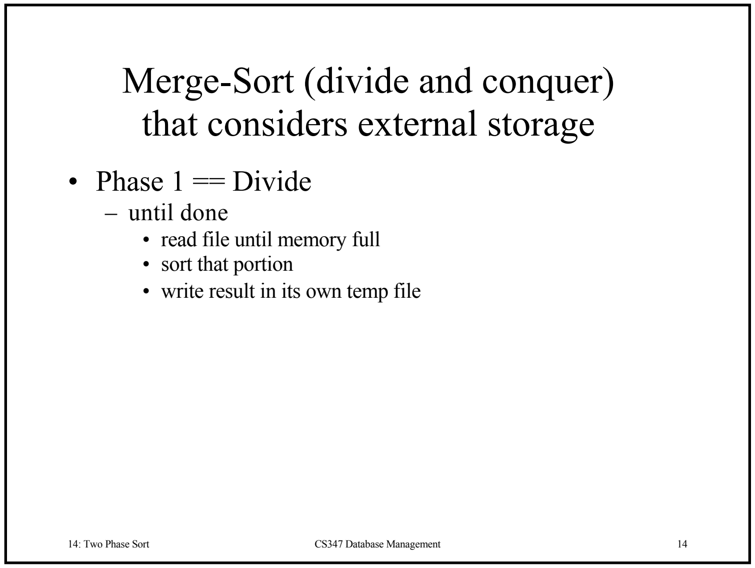


http://pages.cs.wisc.edu/~vernon/cs367



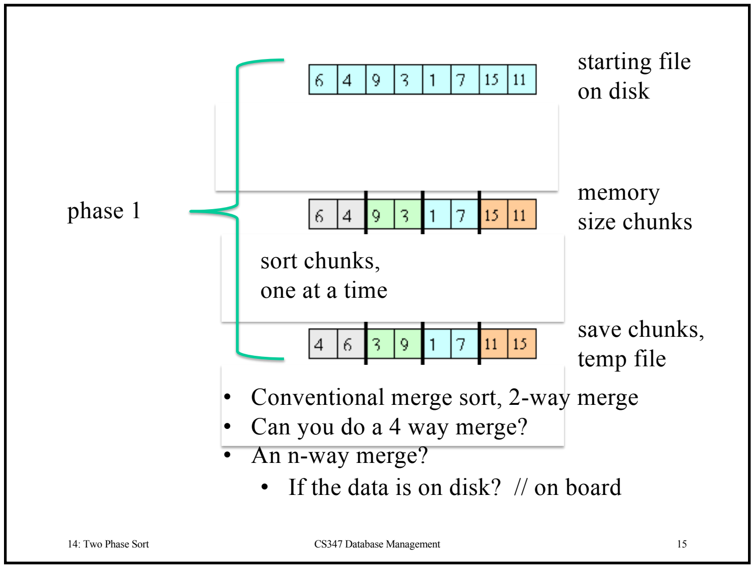




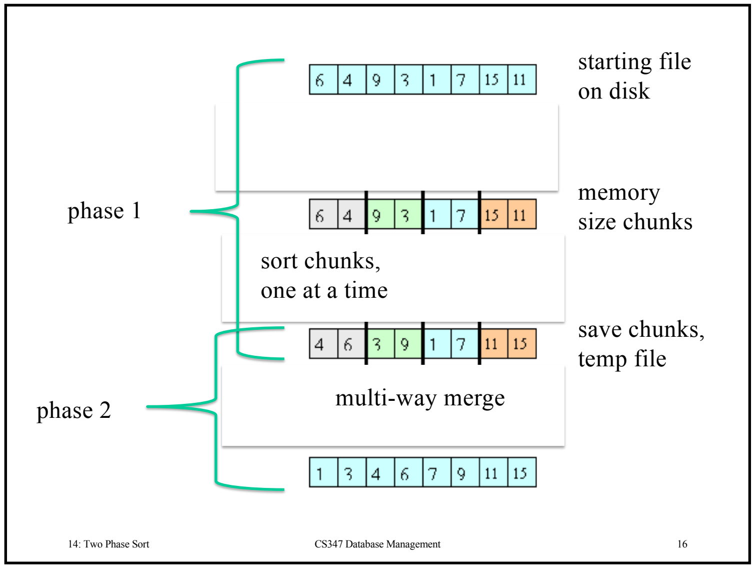


Merge-Sort (divide and conquer) that considers external storage

- Phase 1 == Divide
  - until done
    - read file until memory full
    - sort that portion
    - write result in its own temp file



- Conventional merge sort, 2-way merge
- Can you do a 4 way merge?
- An n-way merge?
  - If the data is on disk? // on board



Merge-Sort (divide and conquer) that considers external storage

- Phase 1 == Divide
  - until done
    - read file until memory full
    - sort that portion
    - write result in its own temp file
- Phase 2 == n-way Merge for n temp files
  - read 1 block of each temp file
  - until done
    - n-way merge to produce sorted block in an output buffer
      - upon fulll output buffer, write to disk
    - if a temp file's block is exhausted, read another block

In how long? How large a file can we sort?

Two Phase External Sorting

- For all intents and purposes, sorting a file, f, bigger than available memory requires
  - 2 * B(f) I/O reads + 2 * B(f) I/O writes

Bottleneck is the Merge Phase

- Given RAM, M
- Can't have more [temporary] sorted chunks than B(M) – 1, // number of page buffers + 1 for output
- How big is each chunk?

  M
- So we can sort ~ B(M) * M

Suppose 4 GBytes of RAM

- 4 Kbyte pages [buffers]
- B(M) = 4GB/4KB = __________