CSE 326 Quiz Section
Memory Use of Sorting Algorithms

April 11, 2002

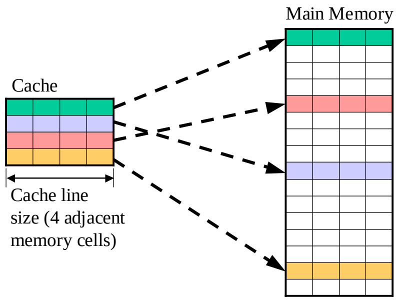
Example Memory Hierarchy Statistics

Name	Extra CPU cycles used to access	Size
L1 (on chip) cache	0	32 KB
L2 cache	8	512 KB
RAM	35	256 MB
Hard Drive	500,000	8 GB

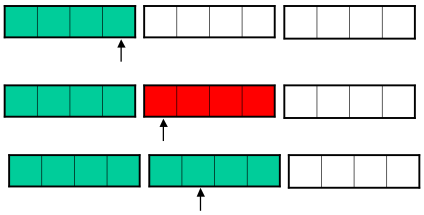
The Memory Hierarchy Exploits Locality of Reference

- Idea: *small* amount of *fast* memory
- Keep *frequently* used data in the *fast* memory
- LRU replacement policy
  - Keep recently used data in cache
  - To free space, remove Least Recently Used data

Cache Details (simplified)

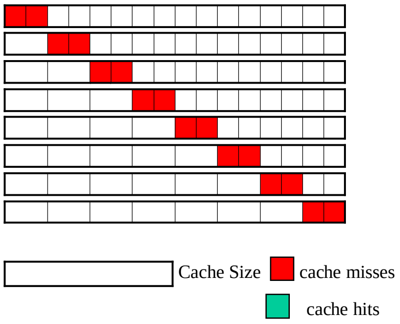


Traversing an Array

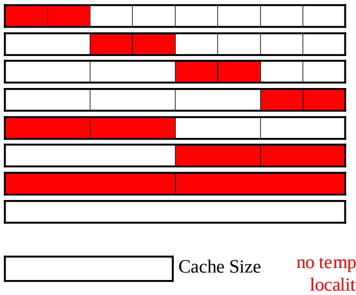


- One miss for every 4 accesses in a traversal

Iterative MergeSort

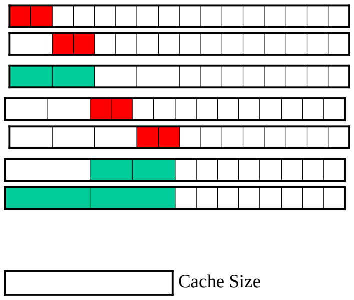


Iterative MergeSort – cont’d

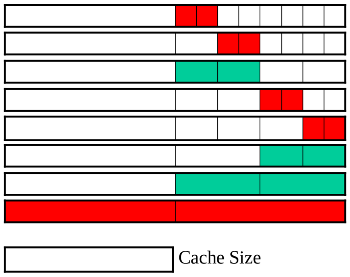


no temporal locality!

“Tiled” MergeSort – better



“Tiled” MergeSort – cont’d



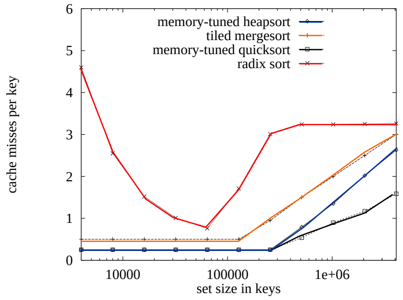
QuickSort

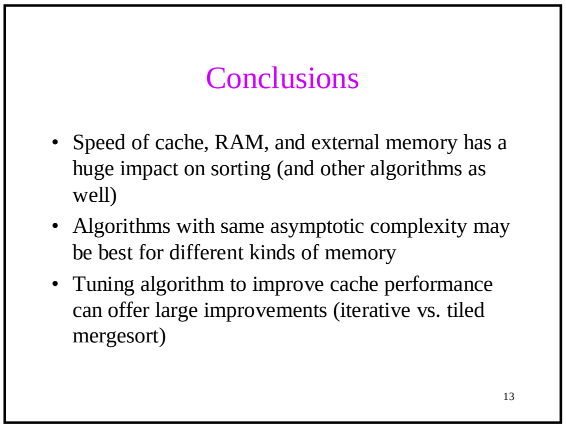
- Initial partition causes a lot of cache misses
- As subproblems become smaller, they fit into cache
- Good cache performance

Radix Sort – Very Naughty

- On each BucketSort
  - Sweep through input list – cache misses along the way (bad!)
  - Append to output list – indexed by pseudo-random digit (ouch!)

Cache Misses





Conclusions

- Speed of cache, RAM, and external memory has a huge impact on sorting (and other algorithms as well)
- Algorithms with same asymptotic complexity may be best for different kinds of memory
- Tuning algorithm to improve cache performance can offer large improvements (iterative vs. tiled mergesort)

13