# Sorting atomic items

Chapter 5

Lower bounds

# Sorting and permuting

In RAM model Sorting includes Permuting since we need to determine the sorted permutation and then permute the items. Sorting is $\Theta(n \log n)$ while permuting is $\Theta(n)$.

In disk model Sorting problem is equivalent to Permuting problem by the point of view of I/O complexity.

Moving elements is difficult as Sorting in this model. It is the real bottleneck: I/O bottelneck.

How to use Sort to Permute

# Use Sort to Permute

Permute Sequence S, S[1,n] according Π[1,n], i.e.
Output S[Π(1)], S[Π(2)], …S[Π(n)]

RAM model: jump on the memory to read S[Π(i)] then O(n).
Same algorithm on 2-level model: O(n) I/O's: Too much!

Use Sort and Scan to Permute;

1) Create sequence P of pairs <i , Π(i)>
2) Sort according Π component
3) Parallel scan of of S and P and change Π(i) with S[Π(i)]
4) Sort P on the first component

# Use Sort to Permute

S: a, b, c, d   Π : 2, 4, 1, 3

RESULT: b, d, a, c

1. Create P.

   P: <1, 2> <2,4> <3, 1> <4, 3>

2. Sort on Π component

   P: <3, 1> <1, 2> <4, 3> <2, 4>

3 Parallel scan of S and P to substitute in P to $\Pi[i]$ ,$S[\Pi(i)]$

   S: a, b, c, d

   P: <3, 1> <1, 2> <4, 3> <2, 4>

   P: <3, a> <1, b> <4, c> <2, d>

4 Sort on the first component

   P: <1, b> <2, d> <3, a> <4,c>

# Use Sort to Permute

Algorithm uses 2 Scan and 2 Sort. Hence:

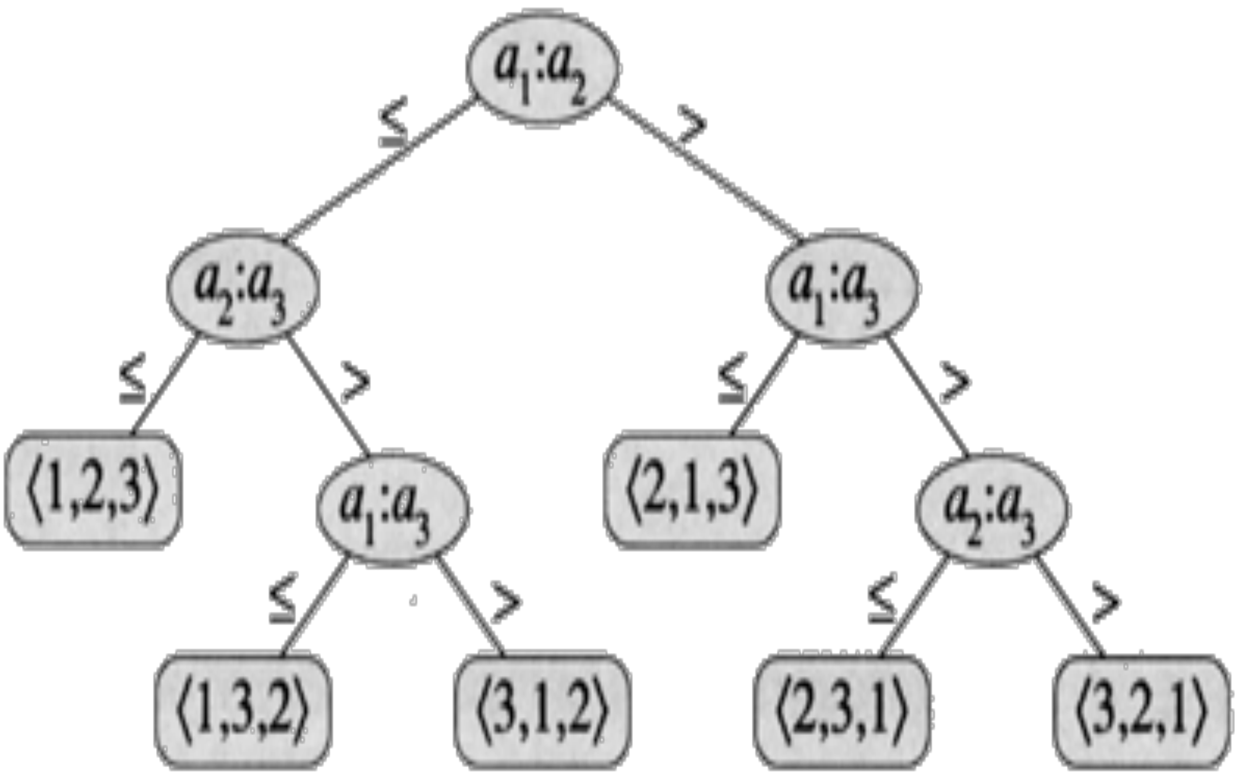
O(min{n, (n/B) log(n/M)}) I/O's

This bound and that for Sorting are optimal for I/O's.
The bounds are equal whenever n = Ω(n/B) log(n/M)

| | time complexity (RAM model) | I/O complexity (two-level memory model) |
|---|---|---|
| Permuting | $O(n)$ | $O(\min\{n, \frac{n}{B}\log_{M/B}\frac{n}{M}\})$ |
| Sorting | $O(n\log_2 n)$ | $O(\frac{n}{B}\log_{\frac{M}{B}}\frac{n}{M})$ |

# Lower bound for sorting

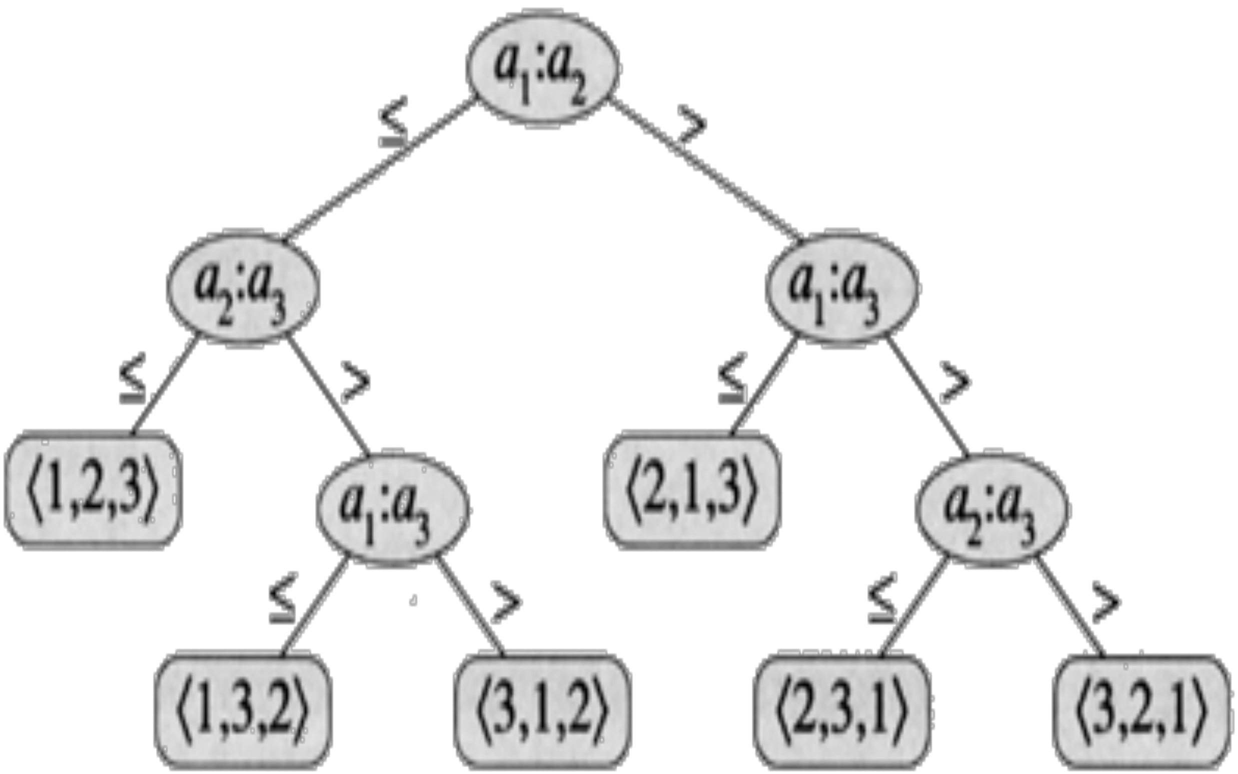
RAM model: Comparison tree to prove lower bound.
Node: comparison. Leaf: solution. Root-to-leaf path t: execution of the alg. on specific data.



# Lower bound for sorting

Sorting: binary tree. The possible solutions (n! for sorting) must be allocated on the leaves. $2^t \geq n!$

$T \geq \log(n!)$ $t = \Omega(n\log n)$



# Lower bound sorting in 2 level model

Comparison tree.

Account for I/O operations.

Operations in internal memory can be used for free.

Every node of the decision tree corresponds to one I/O.

The fan-out corresponds to the result of the comparisons after an I/O:


A block of B new elements is fetched to M. M-B elements are old , B are new.

The B elements can B occupy positions of M in $\binom{M}{B}$ possible ways.

# Lower bound sorting in 2 level model

An I/O operations can generate $\binom{M}{B}$ different results.

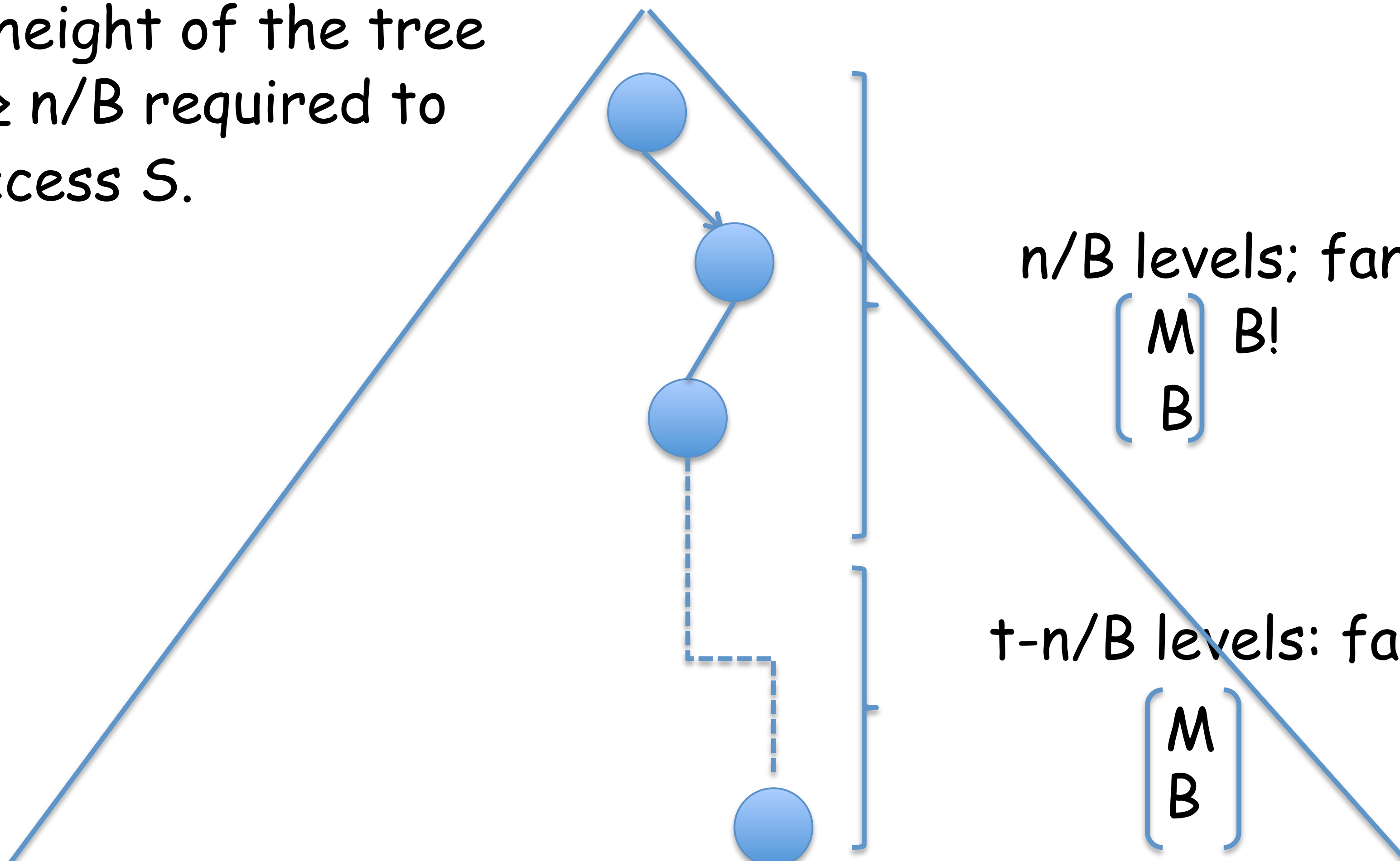
Un addition, we have to consider B! different permutations of B. (the other M-B items have already been considered in previous I/O operations).

In total $\binom{M}{B} B!$ possible orderings generated by an I/O operation and by the internal comparisons.

$\binom{M}{B} B!$ is the fan-out of each node.

# Lower bound sorting in 2 level model

t height of the tree
$t \geq n/B$ required to access S.



n/B levels; fan-out $\binom{M}{B} B!$

t-n/B levels: fan-out $\binom{M}{B}$

The number of leaves is $\binom{M}{B}^{t} (B!)^{n/B}$

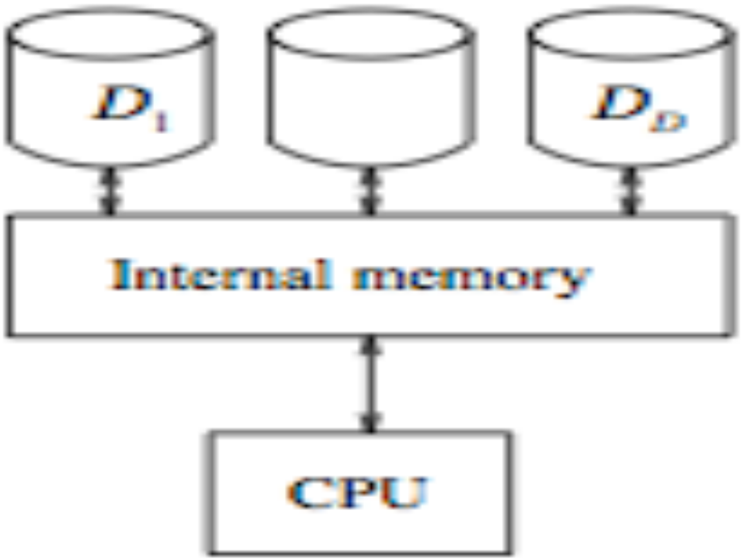
# Sorting: lower bound

$$\begin{aligned}
\left(\frac{M}{B}\right)^{t} (B!)^{\frac{N}{B}} &\geq N! \\
t\log\left(\frac{M}{B}\right) + \frac{N}{B}\log(B!) &\geq \log(N!) \\
tB\log\left(\frac{M}{B}\right) + \frac{N}{B}B\log B &\geq N\log N \\
tB\log\left(\frac{M}{B}\right) &\geq N\log\left(\frac{N}{B}\right) \\
t &\geq \frac{N}{B}\frac{\log\left(\frac{N}{B}\right)}{\log\left(\frac{M}{B}\right)}
\end{aligned}$$

$t = \Omega\,(N/B)\log_{M/B}(N/B) = \Omega\,(N/B)\log_{M/B}(N/M)$

# Lower bound for the D disks model

- Parallel D disks model : computer + D disks
- Input/output are from disks



# Lower bound for the D disks model

I/O operation: 1 block of B data is fetched to the core memory of size M from each one disk. DB data are fetched in parallel.

- Evaluate the number of parallel I/O's

The previous bound can be easily extended to D disks.

A comparison-based Sorting algorithm must execute:

$$\Omega((n/DB) \log_{M/B} (n/DB)) \text{ I/O operations}$$

# Lower bound for the D disks model

Observation: D does not appear in the base of log. If this would be the case, it will increase the bound, so penalizing the sorting algorithm which uses D disks!

MergeSort is optimal for 1 disks but it is not for D disks.

The merging should be O(n/DB) I/O's, that is at each step D pages are fetched one per disk, with an I/O.

Merging is not parallel operation: after a comparison more than B items have to be possibly fetched from the same disk.

# Sorting in the D disks model

- Disk Striping technique: data layout on disks
- Look to the D disks as a single disk B'=DB.
- The bandwith of I/O's increases but design efficient alg . is more difficult.

$$O((n/B') \log_{M/B'} (n/M)) = O((n/DB) \log_{M/DB}(n/M))$$

- Observe: the base of log. increases and disk striping is more and more inefficient as D increases.
- Merge is as before.
- Problem: the independency of disks is not exploited they are used as a monolithic system. Very difficult to exploit it!

# Sort in the D disks model

We must design a different algorithm.
In the following:

- Greed Sort: elegant and complex new algorithm achieving a close to optimal upper bound.

# Lower bound for Permuting

1 disk model:

If $B \log (M/B) \leq \log n$ then $\Omega (n)$

otherwise $\Omega ((n/B)\log_{M/B}( n/M))$ NO PROOF

The previous algorithm was optimal!

D disks model: $\Omega(\min \{n/D, (n/DB)\log_{M/B}( n/DB)\})$

The bounds for sorting and permuting are the same except for the case:

$B\log(M/B) \leq \log n$.

This inequality holds for $n = \Omega (2^B)$

(since B and M are few k bytes and few Gigabytes respectively and log (M/B) can be neglected) .

This situation is unreasonable!

In practice, Sorting = Permuting