Sorting Algorithms

Paolo Camurati

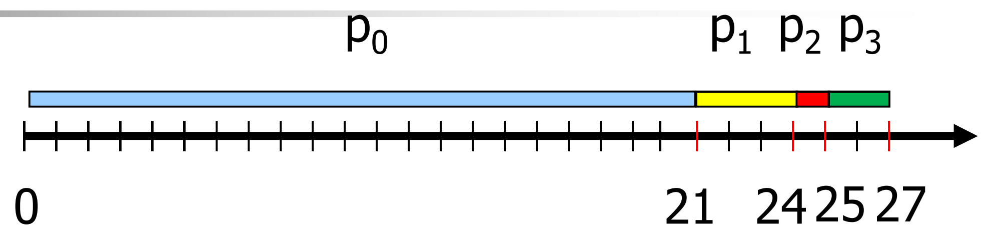
Dip. Automatica e Informatica
Politecnico di Torino

On the importance of sorting

- On an average application 30% of CPU time is spent on sorting data
- Example
  - CPU scheduling
    - Processes p_i with duration
      - p_0 21, p_1 3, p_2 1, p_3 2
    - Impact of sorting on average wait time

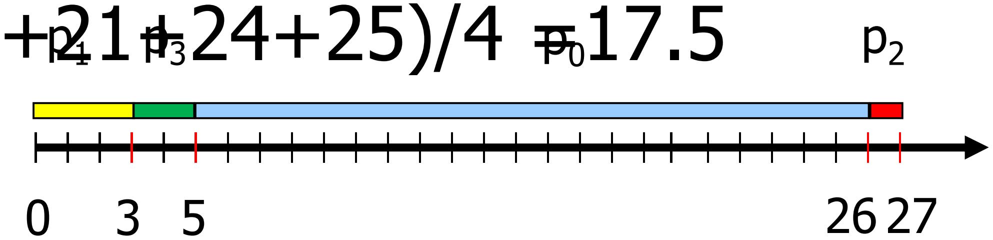
On the importance of sorting

- p_0-p_1-p_2-p_3



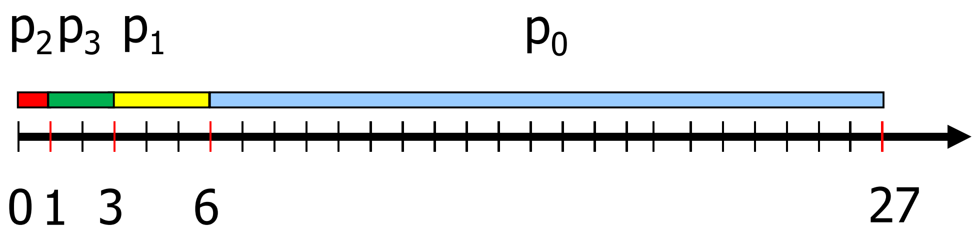
average wait time (0+21+24+25)/4 =17.5

- p_1-p_3-p_0-p_2



average wait time (0+3+5+26)/4 =8.5

- (sorted) p_2-p_3-p_1-p_0



average wait time (0+1+3+6)/4 =2.5

Sorting applications

- Trivial applications
  - Sorting a list of names, organizing an MP3 library, displaying Google PageRank results, etc.
- Simple problems if data are sorted
  - Find the median, binary search in a database, find duplicates in a mailing list, etc.
- Non trivial applications
  - Data compression, computer graphics (e.g., convex hull), computational biology, etc.

Definitions

Sorting

- Input
  - Symbols <a_1, a_2, …, a_n>
  - Symbols belong to a set having an order relation
- Output
  - Permutation <a'_1, a'_2, …, a'_n> of the input
  - Such that the order relation $a'_1 \leq a'_2 \leq \ldots \leq a'_n$ holds

Definitions

stringcmp

Order relation $\leq$

- Binary relation between elements of a set A satisfying the following properties
  - Reflexivity $\forall\, x \in A \rightarrow x \leq x$
  - Antisymmetry $\forall\, x, y \in A \rightarrow x \leq y \wedge y \leq x \Rightarrow x = y$
  - Transitivity $\forall\, x, y, z \in A \rightarrow x \leq y \wedge y \leq z \Rightarrow x \leq z$

A is a partially ordered set (poset)

If relation $\leq$ holds $\forall\, x, y \in A$, A is totally ordered set

Classification

- Internal sorting
  - Data are in main memory
  - Direct access to elements

this class

- External sorting
  - Data are on mass memory
  - Sequential access to elements

workinh few of the data
at the same time

Classification

- In place sorting use single array not extra spaces
  - n data in array + constant number of auxiliary memory locations

- Stable sorting
  - For data with duplicated keys the relative ordering is unchanged

Example

- Record with 2 keys
  - Name (key is first letter)
  - Group (key is an integer)

Unsorted data

Chiara	3
Barbara	4
Andrea	3
Roberto	2
Giada	4
Franco	1
Lucia	3
Fabio	3

Example

First sorting according to first letter

Andrea	3
Barbara	4
Chiara	3
Fabio	3
Franco	1
Giada	4
Lucia	3
Roberto	2

Second sorting according to group NON stable algorithm

Franco	1
Roberto	2
Chiara	3
Fabio	3
Andrea	3
Lucia	3
Giada	4
Barbara	4

Second sorting according to group Stable algorithm

Franco	1
Roberto	2
Andrea	3
Chiara	3
Fabio	3
Lucia	3
Barbara	4
Giada	4

not stable, change the element with the same key

Classification: complexity

- $O(n^2)$
  - Simple, iterative, based on comparison
  - Insertion sort, Selection sort, Exchange/Bubble sort
- $O(n^{3/2})$
  - Shellsort (with certain sequences)
- $O(n \log n)$
  - More complex, recursive, based on comparison
  - Merge sort, Quicksort, Heapsort
- $O(n)$
  - Applicable with restrictions on data, based on computation
  - Counting sort, Radix sort, Bin/Bucket sort

asymtotic

Classification: complexity

A more detailed analysis is possible, distinguishing between

- Comparison and
- Exchange operations

When date are large, exchanging them may be expensive data

Asymptotic complexity however doesn't change

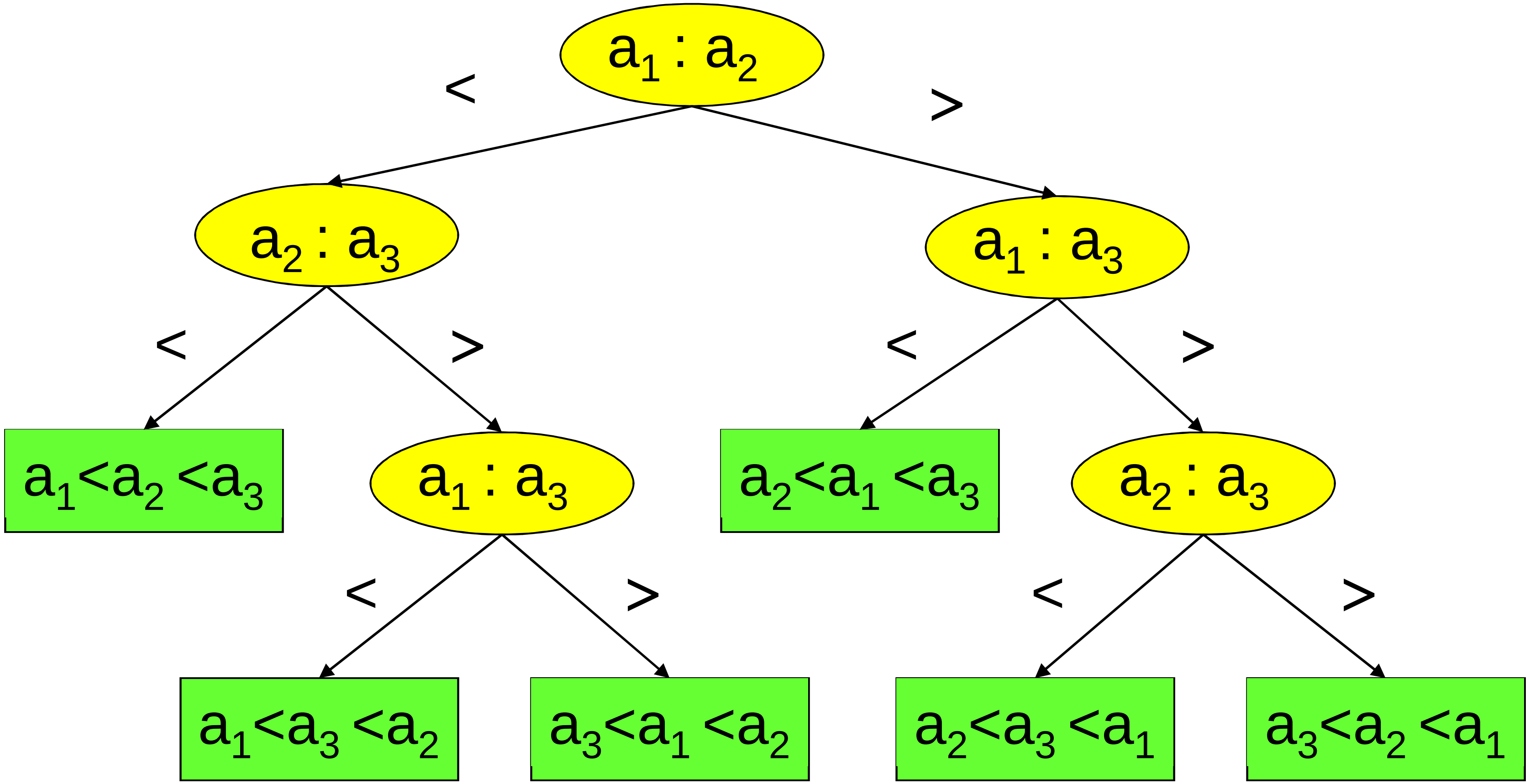
Lower bound

Algorithms based on comparison

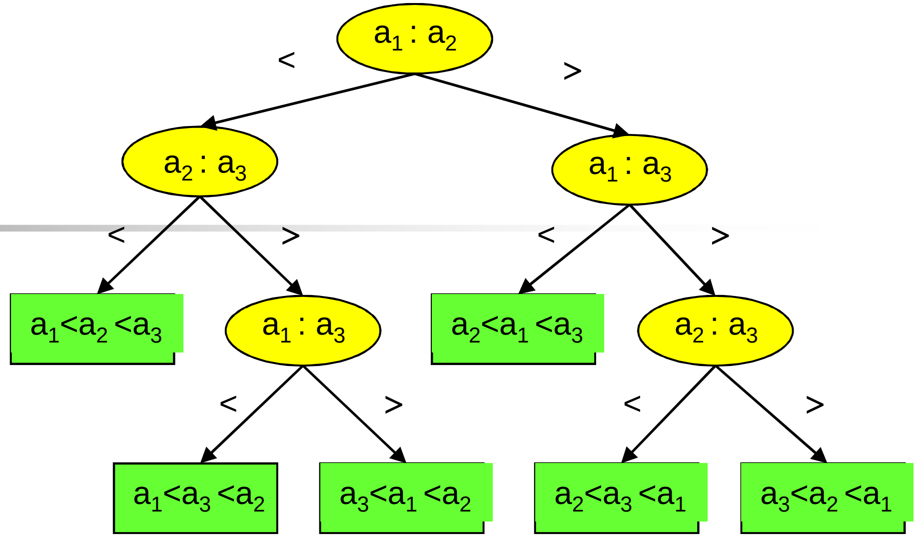
- Elementary operation
  - Comparison $a_i : a_j$
- Outcome
  - Decision ($a_i > a_j$ or $a_i \leq a_j$)
  - Decisions organized as a decision tree

Lower bound

Sort array of 3 distinct elements a_1, a_2 , a_3

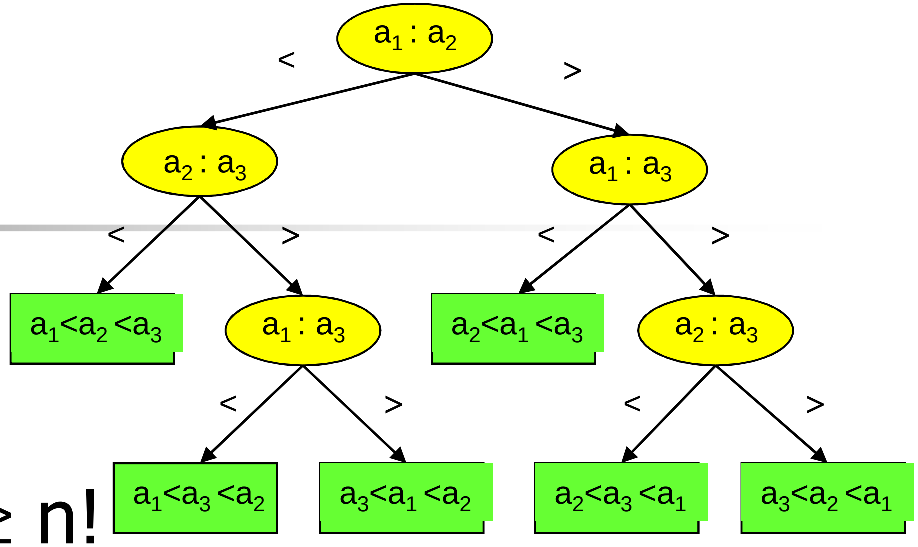


Lower bound



- For n distinct integers
  - Number of possible sortings = number of permutations = n!
- Each solution
  - Sits on a tree leaf
- Complexity
  - Number h of comparisons, that is, the tree height h
- For a complete tree
  - Number of leaves = 2^h

Lower bound



- Then we must have

$$2^h \geq n!$$

- Stirling's approximation
  - $n! > (n/e)^n$
- Then we have

$$2^h \geq n! > (n/e)^n$$

$$2^h > (n/e)^n$$

$$h > \lg(n/e)^n$$

$$h > n(\lg n - \lg e) = \Omega(n \lg n)$$