Sorting Algorithms

Paolo Camurati

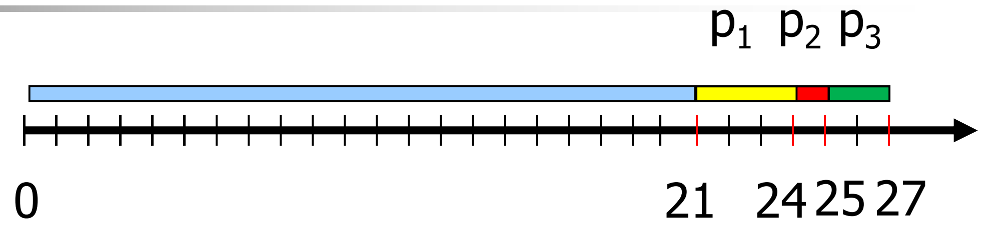
Dip. Automatica e Informatica
Politecnico di Torino

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

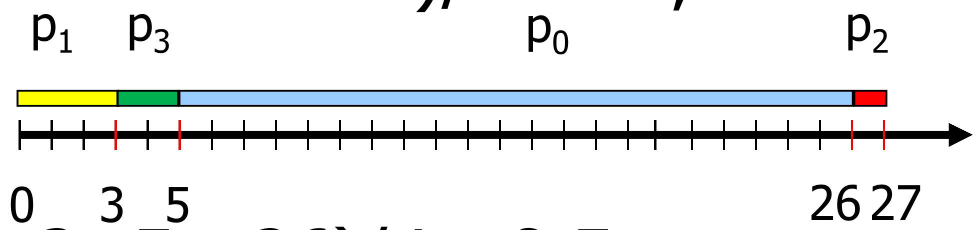
On the importance of sorting

- p_0-p_1-p_2-p_3



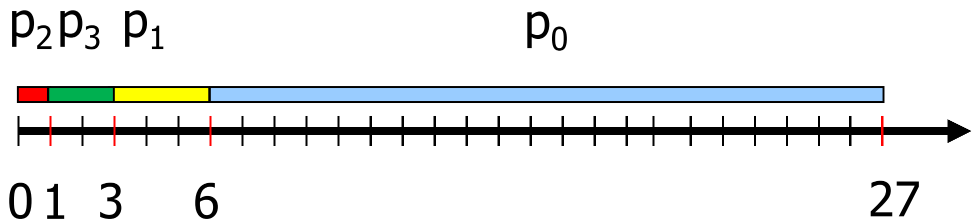
average wait time (0+21+24 +25)/4 =17,5

- 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 (eg. convex hull), computational biology, etc.

Classification

- Internal sorting
  - Data are in main memory
  - Direct access to elements
- External sorting
  - Data are on mass memory
  - Sequential access to elements

Classification

- In place sorting
  - 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)

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

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

Classification: complexity

A more detailed analysis is possible, distinguishing between

- Comparison and
- Exchange operations

When date are large, exchanging them may be expensive

Asymptotic complexity however doesn't change

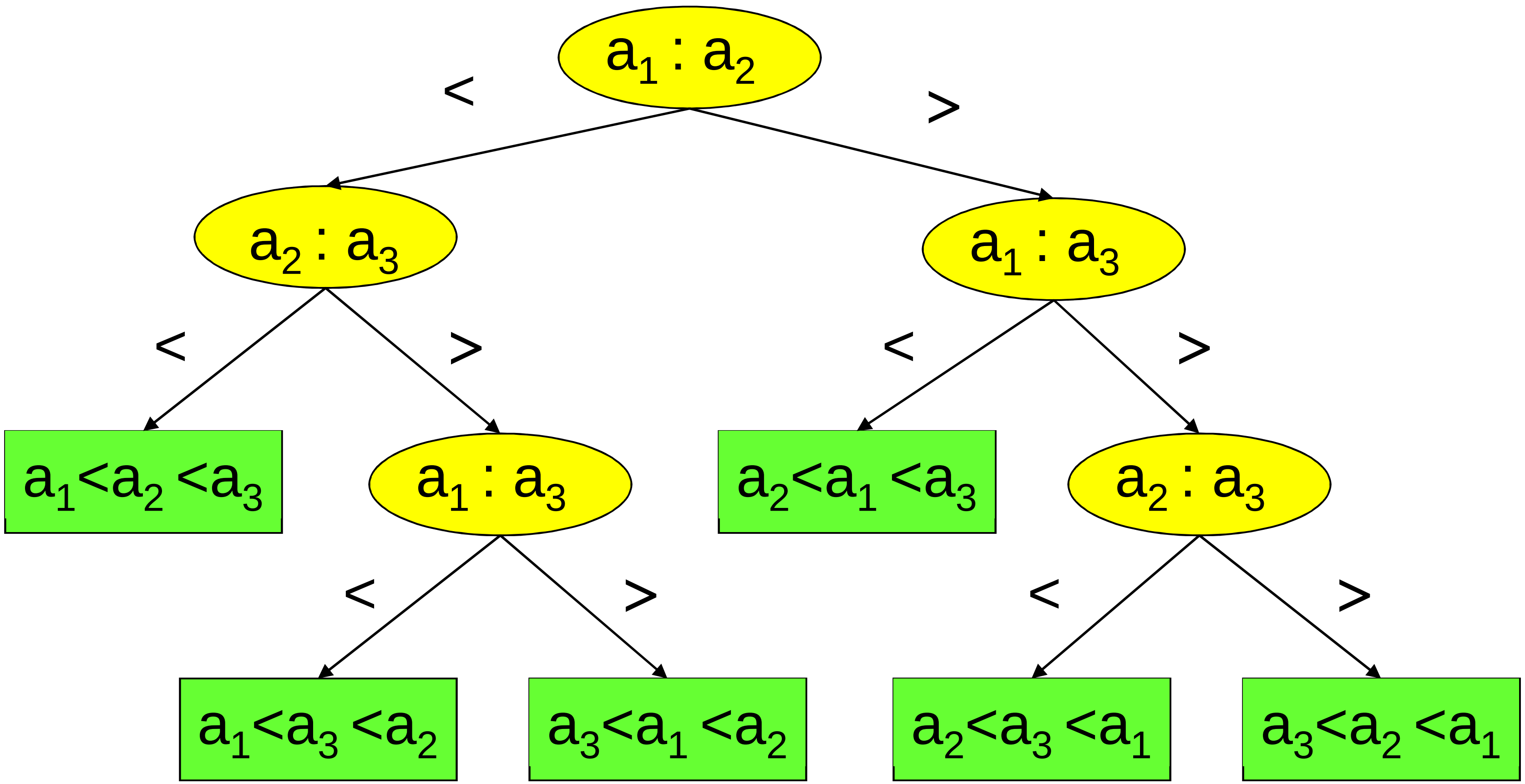
Lower bound

Algorithms based on comparison

- Elementary operation
  - Comparison $a_i : a_j$
- Outcome
  - Decision ($a_i > a_j$ or $a_i \leq a_j$), 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!
- Complexity
  - Number h of comparisons (tree height)
- Each solution = tree leaf
- Number of leaves = 2^h
- Stirling's approximation: $n! > (n/e)^n$

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

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