§1 Introduction

- "Virtues" of Computer *Science*
- Classes of asymptotic growth
- Importance of asymptotic efficiency
- Sorting: specification and optimality

Recommended Literature

- Cormen, Leiserson, Rivest (&Stein): *Introduction to Algorithms*, MIT Press.
- Dasgupta, Papadimitriou, Vazirani: *Algorithms*, McGraw-Hill
- Kleinberg, Tardos: *Algorithm Design*, Pearson.
- Vöcking, Alt, Dietzfelbinger, Reischuk, Scheideler, Vollmer, Wagner: *Algorithms Unplugged*, Springer.
- *Introduction to the Analysis of Algorithms* (Robert Sedgewick and Philippe Flajolet)
- *Online Computation and Competitive Analysis* (Allan Borodin and Ran El-Yaniv)
- *Probability and Computing*: Randomized Algorithms and Probabilistic Analysis (Michael Mitzenmacher and Eli Upfal)
- M. Sipser: *Introduction to the theory of computation*, Boston.

Computer *Science*

"***Virtues***":

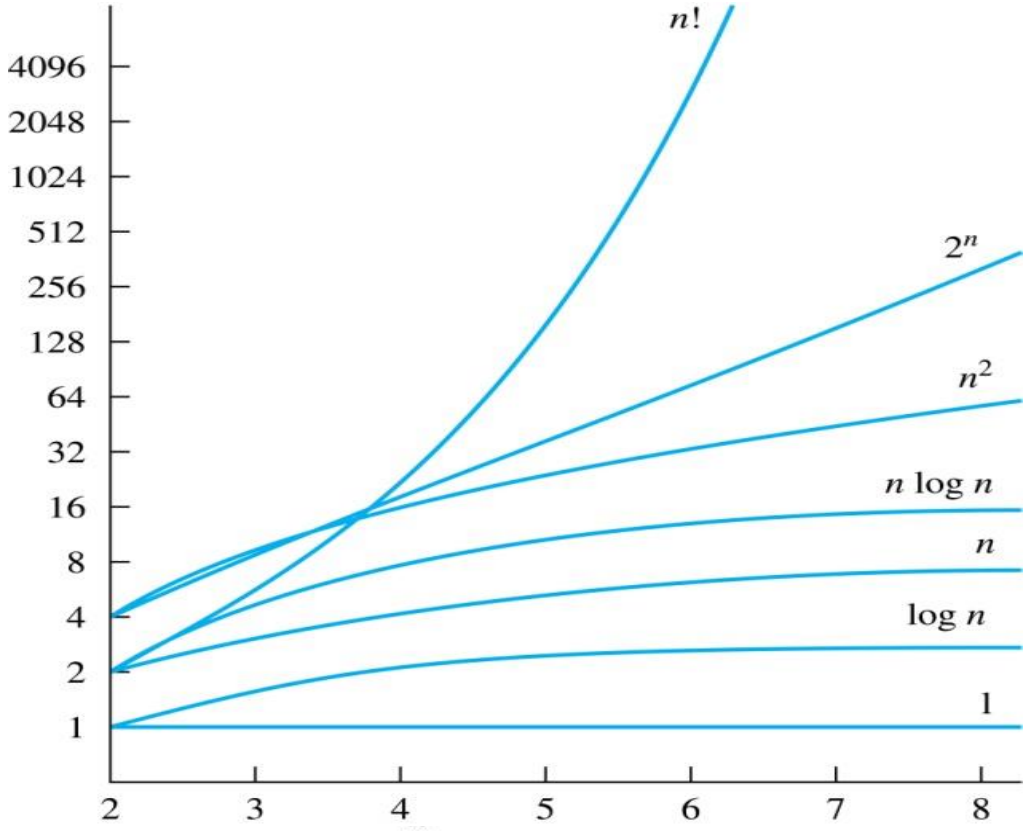
- problem specification
- formal semantics
- algorithm design
- and analysis (correctness, efficiency)
- proof of optimality (complexity theory, §6)

≠program
≠heuristic

Asymptotic Growth

- constant 17
- logarithmic $\log(n)$
- square root $\sqrt{n}$
- linear n
- quasilinear $n \cdot \log(n)$
- quadratic n^2
- cubic n^3
- polynomial $c \cdot n^c$
- superpolynomial $n^{\log(n)}$
- exponential 2^n
- doubly exponential 2^{2^n}
- tetration $2^{2^{\cdots^{2^n}}}$

Landau „big-Oh“ notation $f = O(g)$



Asymptotic Efficiency

n	$\log_2 n \cdot 10s$	$n \cdot \log n$ sec	n^2 msec	n^3 μsec	2^n nsec
10	33sec	33sec	0.1sec	1msec	1msec
100	≈1min	11min	10sec	1sec	40 bill. y
1000	≈1.5min	≈3h	17min	17min	
10 000	≈2min	1.5 days	≈1 day	11 days	
100 000	≈2.5min	19 days	4 months	32 years	

- Running times of some sorting algorithms
  - **BubbleSort**: $O(n^2)$ comparisons and copy instructions
  - **QuickSort**: typically $O(n \cdot \log n)$ steps
    but $O(n^2)$ in the worst-case
  - **HeapSort**: always at most $O(n \cdot \log n)$ operations
  - **BucketSort**: $O(n)$ operations
  - SORT primitive: $O(1)$
- **Worst-case** vs. **average**-case vs. **best** case
  w.r.t. input size =: $n \to \infty$

Sorting: Specification, Optimality

Specification: Fix set X with total order $\leq$.

Input: $N \in \mathbb{N}$ and array $x[1 \ldots N]$ with values in X

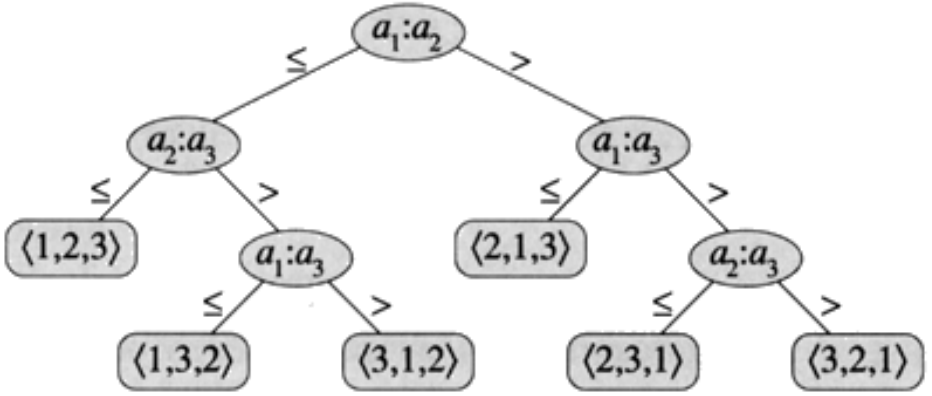
Output: *Permutation* $\pi:[1 \ldots N] \to [1 \ldots N]$ s.t. $x_{\pi(1)} \leq \ldots \leq x_{\pi(N)}$

Lemma: a) For fixed N, every *comparison-based* sorting algorithm of worst-case runtime $T(N)$ can be "unrolled" into a *decision tree* for sorting N elements of depth $\leq T(N)$.

b) Every permutation $x=\pi$ ends up in the unique leaf with label π^{-1}.

c) A binary tree with $N!$ leaves has depth $\geq \log(N!) = \Theta(N \cdot \log N)$

Definition: A *Decision Tree* *for sorting N elements* is a binary tree whose internal nodes are labeled with tests "$x[i] \leq x[j]$?" and whose leaves are labeled with permutations π such that every input x "ends up" in a leaf whose label π satisfies $x[\pi[1]] \leq \ldots \leq x[\pi[N]]$.



§1 Introduction Recap

- "Virtues" of Computer *Science*
- Classes of asymptotic growth
- Importance of asymptotic efficiency
- Sorting: specification and optimality