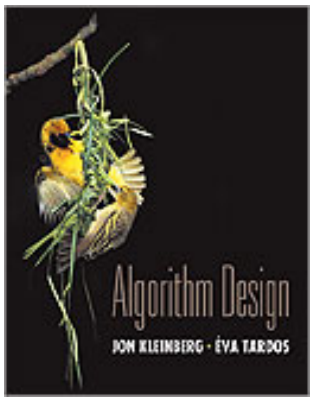
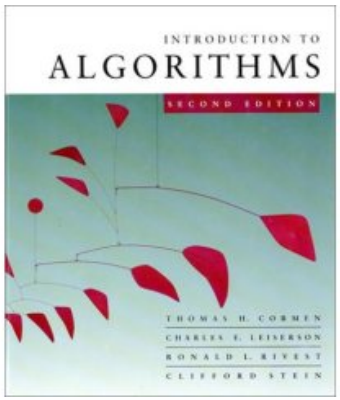
Algorithms and Complexity

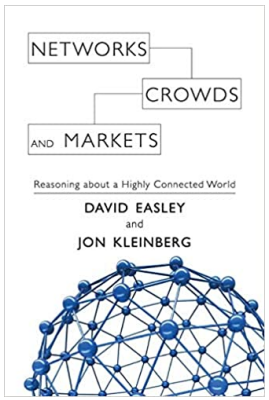
Fall 2025-2026

Algorithmics: Basic references

- Kleinberg, Tardos. Algorithm Design, Pearson Education, 2006.
- Cormen, Leisserson, Rivest and Stein. Introduction to algorithms. Second edition, MIT Press and McGraw Hill 2001.
- Easley, Kleinberg. Networks, Crowds, and Markets: Reasoning About a Highly Connected World, Cambridge University Press, 2010

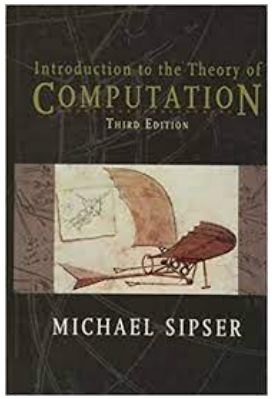


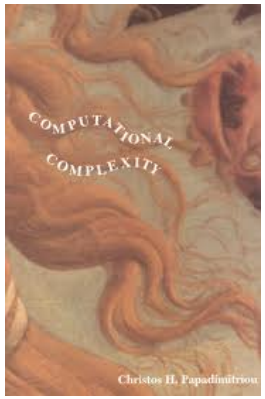


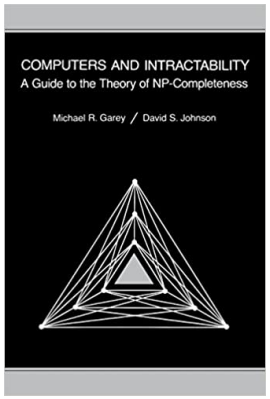


Computational Complexity: Basic references

- Sipser Introduction to the Theory of Computation 2013.
- Papadimitriou Computational Complexity 1994.
- Garey and Johnson Computers and Intractability: A Guide to the Theory of NP-Completeness 1979







Alphabets and languages

- Alphabet: a non-empty finite set.
- Symbol: an element of an alphabet.
- Word: a finite sequence of symbols.
  λ denotes the empty word, a sequence with 0 symbols.
- Language: set of words over an alphabet.

Alphabets and languages: concatenation

- For an alphabet Σ,
- Σ^* denotes the set of words (finite sequence of symbols) over Σ (finite sequence of symbols in Σ).
- The basic operation on words is the concatenation.
  - For $x, y \in \Sigma^*$, $x \cdot y$ is the word obtained placing the symbols in x followed by the symbols in y.
  - For example, if $\Sigma = \{0, 1\}$, $x = 001000$ and $y = 11101$, $x \cdot y = 00100011101$.
  - $(\Sigma^*, \cdot)$ is a non-commutative monoid.
- For $x \in \Sigma^*$, the length of x ($|x|$) is the number of symbols in x.
  - $|x \cdot y| = |x| + |y|$
  - $|\lambda| = 0$
- A language L is a subset of Σ^*.

  We can extend concatenation to languages in the usual form.

$$L_1 \cdot L_2 = \{x \cdot y \mid x \in L_1, y \in L_2\}$$

Alphabets and languages: enumerability

- Let Σ be an alphabet, Σ^* is enumerable.
- We cannot use alphabetical order $a, aa, aaa, aaaa, \ldots$
- We use lexicographic order
  - Order words by length.
  - Among words with the same length order them according to alphabetical order.
- For $\Sigma = \{0, 1\}$ we can enumerate $\{0.1\}^*$ as

$$\lambda, 0, 1, 00, 01, 10, 11, 000, \ldots, 111, 0000, \ldots$$

Turing machines

- A Turing machine (TM) M is a tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\mathtt{F}})$, where
  - Q is a finite set of states,
  - Σ is the input alphabet.
  - Γ is the tape alphabet, $\Gamma = \Sigma \cup \{\mathtt{b}, \blacktriangleright\}$, with $\mathtt{b}, \blacktriangleright \notin \Sigma$.
  - $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\mathtt{l}, \mathtt{r}, \mathtt{n}\}$ is the transition function.
  - q_0 is the initial state.
  - $q_{\mathtt{F}}$ is the final or accepting state.

Turing machines: computation

- Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_F)$ be a TM and $x \in \Sigma^*$
- The computation of M with input x goes as follows:
  - Initially: the state is q_0; the tape has ▶ x and all remaining cells in the tape hold a ␣; the head has access to the first symbol of x.
  - While there is a transition in δ for the combination state, symbol accessed by the head, the transition is applied.
- Assuming that Q and Γ are disjoint, the word ▶ $\alpha q \beta$ is a configuration in which ▶ $\alpha\beta$ are the tape contents (␣ outside), q is a state and the head is accessing the tape cell holding the first symbol in β.
- The computation of M on x is a sequence of configurations,

Turing machines: output

- The computation of a TM on input x is a sequence of configurations, starting with $\blacktriangleright q_0 x$, and continuing so far as there is a next configuration according to δ.
- If the sequence of configurations is finite, we say the M halts on input x, we note this as $M(x)\downarrow$,
  otherwise, the computation diverges or does not halt, denoted as $M(x)\uparrow$.
- When $M(x)\downarrow$,
  - the number of configurations in the computation is the computation time.
  - Let $\blacktriangleright \alpha q \beta$ be the last configuration in the computation.
    - $M(x) = \alpha\beta$ is the output,
    - we say that M halts on input x in state q.

Turing machines: Recognizing languages

- $L(M) \subseteq \Sigma^*$ is the set of words that M accepts, i.e, M on input x halts in state $q_{\mathtt{F}}$.
- A language $L \subseteq \Sigma^*$ is recognizable (or recursively enumerable) iff there is a TM M with $L = L(M)$.

Turing machines: Computing functions

- The output $M(x)$ of TM M, allows us to associate a partial function $f_M(x) = M(x)$ to a TM.
- Note that $\text{Dom} f_M = \{x \mid M(x) \downarrow\}$.

- A function $f : D \subseteq \Sigma^*$ is computable iff there is a TM M with $f = f_M$.

Deciders

- A TM M stops always on accepted inputs, but it may stop or not on rejected inputs.
- A TM M is a decider if it stops on any input $x \in \Sigma^*$.
- A language $L \subseteq \Sigma^*$ is decidable iff there is a decider M such that $L = L(M)$.

Questions

- All languages over alphabet Σ are recognizable?
- All languages over alphabet Σ are decidable?
- All functions from Σ^* to Σ^* are computable?