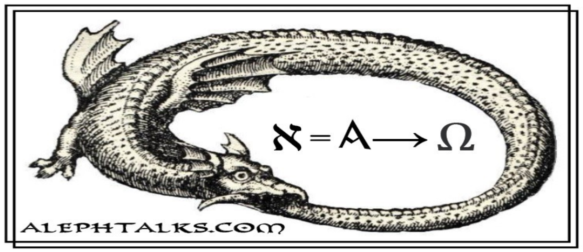
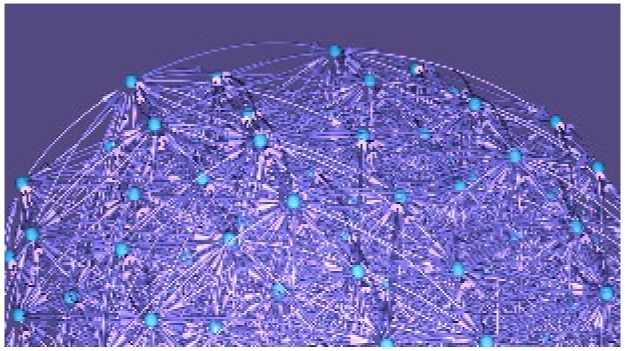


Algorithms
The Science in Computer Science

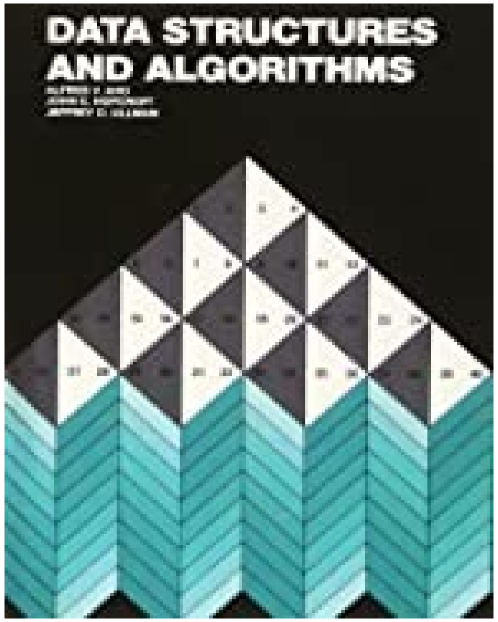
30 March 2022

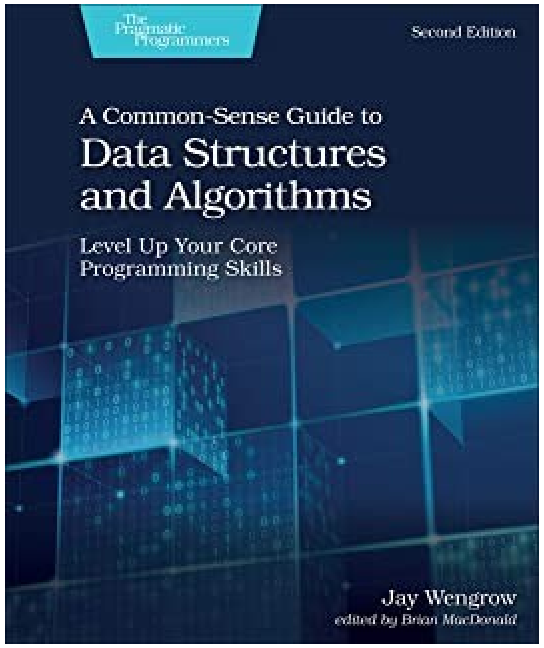
©AlephTalks, 2022



Information Sources

- Wikipedia
- Association for Computing Machinery




The Pragmatic Programmers
Second Edition
A Common-Sense Guide to
Data Structures and Algorithms
Level Up Your Core Programming Skills
Jay Wengrow
edited by Brian MacDonald

The Design and Analysis of Computer Algorithms
Alfred V. Aho | John E. Hopcroft
Jeffrey D. Ullman
PEARSON

What Is An Algorithm?

- In mathematics and computer science, an algorithm is a finite sequence of well-defined instructions, typically used to solve a class of specific problems or to perform a computation.
- Algorithms are used as specifications for performing calculations and data processing.

What Is A Heuristic Algorithm?

- A heuristic algorithm is an approach to problem solving that may not be fully specified or may not guarantee correct or optimal results, especially in problem domains where there is no well-defined correct or optimal result

What Is An Algorithm?

- As an effective method, an algorithm can be expressed within a finite amount of space and time, and in a well-defined formal language for calculating a function.
- Starting from an initial state and initial input (perhaps empty), the instructions describe a computation that, when executed, proceeds through a finite number of well-defined successive states, eventually producing "output” and terminating at a final ending state.

P vs NP Problem

- The P (polynomial time) versus NP (nondeterministic polynomial time) problem is a major unsolved problem in computer science.
- It asks whether every problem whose solution can be quickly verified can also be solved quickly.

P vs NP Problem

- The informal term quickly, used above, means the existence of an algorithm solving the task that runs in polynomial time, such that the time to complete the task varies as a polynomial function on the size of the input to the algorithm (as opposed to, say, exponential time).
- The general class of questions for which some algorithm can provide an answer in polynomial time is "P" or "class P". For some questions, there is no known way to find an answer quickly, but if one is provided with information showing what the answer is, it is possible to verify the answer quickly.
- The class of questions for which an answer can be verified in polynomial time is NP, which stands for "nondeterministic polynomial time".

NP Hard Problem

- .In computational complexity theory, NP-hardness (non-deterministic polynomial-time hardness) is the defining property of a class of problems that are informally "at least as hard as the hardest problems in NP". A simple example of an NP-hard problem is the subset sum problem.
- A more precise specification is: a problem H is NP-hard when every problem L in NP can be reduced in polynomial time to H; that is, assuming a solution for H takes 1 unit time, H's solution can be used to solve L in polynomial time. As a consequence, finding a polynomial time algorithm to solve any NP-hard problem would give polynomial time algorithms for all the problems in NP. As it is suspected that P≠NP, it is unlikely that such an algorithm exists.
-

NP Complete Problem

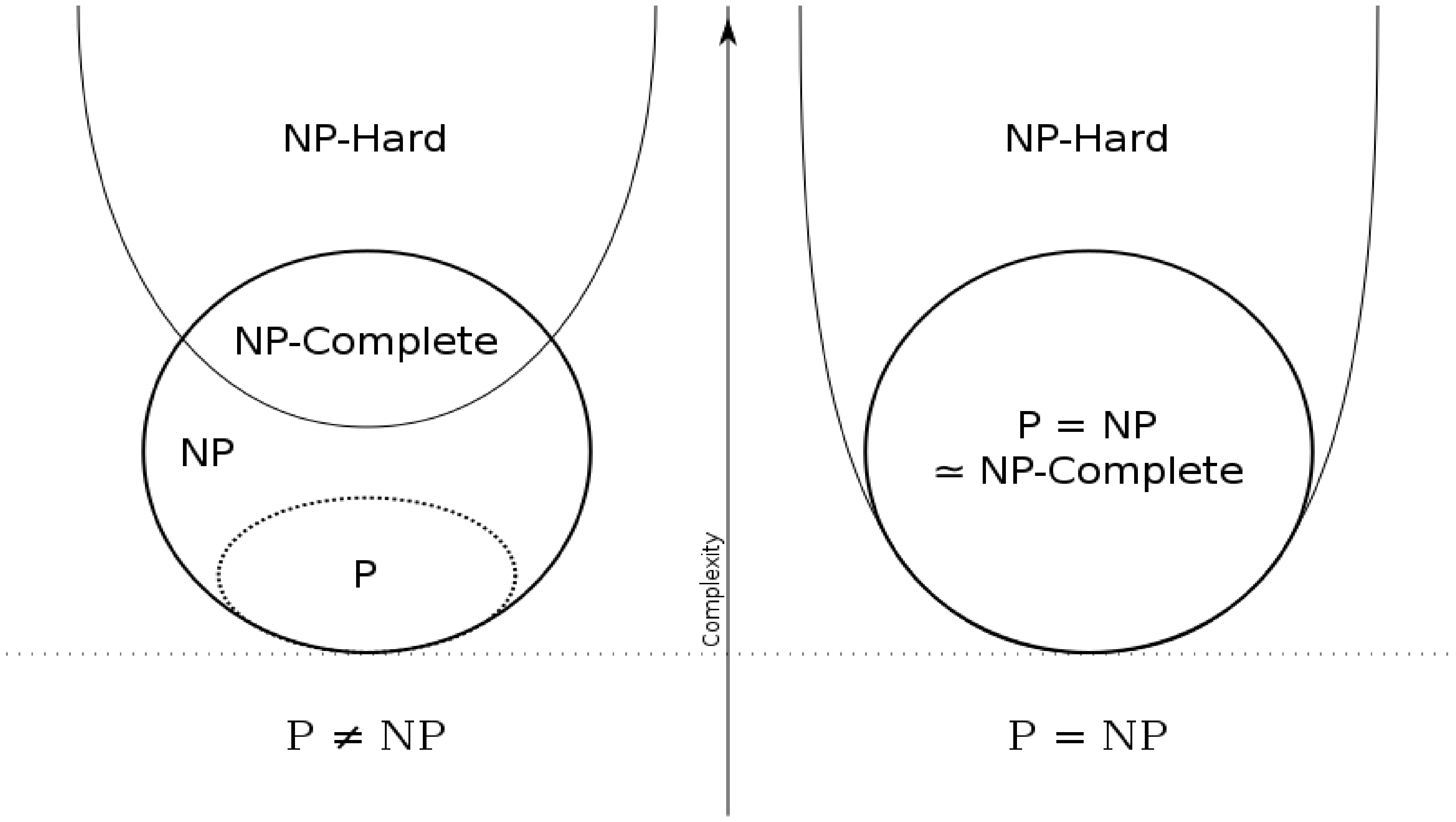
- In computational complexity theory, a problem is NP-complete when:
  - it is a problem for which the correctness of each solution can be verified quickly (namely, in polynomial time) and a brute-force search algorithm can find a solution by trying all possible solutions.
  - the problem can be used to simulate every other problem for which we can verify quickly that a solution is correct. In this sense, NP-complete problems are the hardest of the problems to which solutions can be verified quickly. If we could find solutions of some NP-complete problem quickly, we could quickly find the solutions of every other problem to which a given solution can be easily verified

NP Complete Problem

- The name "NP-complete" is short for "nondeterministic polynomial-time complete".
- In this name, "nondeterministic" refers to nondeterministic Turing machines, a way of mathematically formalizing the idea of a brute-force search algorithm.
- Polynomial time refers to an amount of time that is considered "quick" for a deterministic algorithm to check a single solution, or for a nondeterministic Turing machine to perform the whole search.
- "Complete" refers to the property of being able to simulate everything in the same complexity class.

NP Hard Problem

By Behnam Esfahbod, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=3532181



Traveling Salesman Problem

- The travelling salesman problem (also called the travelling salesperson problem or TSP) asks the following question: "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?" It is an NP-hard problem in combinatorial optimization, important in theoretical computer science and operations research.
- The travelling purchaser problem and the vehicle routing problem are both generalizations of TSP.

Traveling Salesman Problem

- In the theory of computational complexity, the decision version of the TSP (where given a length L, the task is to decide whether the graph has a tour of at most L) belongs to the class of NP-complete problems.
- Thus, it is possible that the worst-case running time for any algorithm for the TSP increases superpolynomially (but no more than exponentially) with the number of cities.

Traveling Salesman Problem Heuristic

- In combinatorial optimization, Lin–Kernighan is one of the best heuristics for solving the symmetric travelling salesman problem.
- Briefly, it involves swapping pairs of sub-tours to make a new tour. It is a generalization of 2-opt and 3-opt. 2-opt and 3-opt work by switching two or three edges to make the tour shorter. Lin–Kernighan is adaptive and at each step decides how many paths between cities need to be switched to find a shorter tour.

Traveling Salesman Problem Heuristic

- The input to the algorithm is an undirected graph G = (V, E) with vertex set V, edge set E, and (optionally) numerical weights on the edges in E. The goal of the algorithm is to partition V into two disjoint subsets A and B of equal (or nearly equal) size, in a way that minimizes the sum T of the weights of the subset of edges that cross from A to B. If the graph is unweighted, then instead the goal is to minimize the number of crossing edges; this is equivalent to assigning weight one to each edge.

Traveling Salesman Problem Heuristic

- The algorithm maintains and improves a partition, in each pass using a greedy algorithm to pair up vertices of A with vertices of B, so that moving the paired vertices from one side of the partition to the other will improve the partition. After matching the vertices, it then performs a subset of the pairs chosen to have the best overall effect on the solution quality T. Given a graph with n vertices, each pass of the algorithm runs in time O(n2 log n).

Traveling Salesman Problem Pseudo Code

```
function Kernighan-Lin(G(V, E)) is
•   determine a balanced initial partition of the nodes into sets A and B
•   do
•       compute D values for all a in A and b in B
•       let gv, av, and bv be empty lists
•       for n := 1 to |V| / 2 do
•           find a from A and b from B, such that g = D[a] + D[b] − 2×c(a, b) is maximal
•           remove a and b from further consideration in this pass
•           add g to gv, a to av, and b to bv
•           update D values for the elements of A = A \ a and B = B \ b
•       end for
•       find k which maximizes g_max, the sum of gv[1], ..., gv[k]
•       if g_max > 0 then
•           Exchange av[1], av[2], ..., av[k] with bv[1], bv[2], ..., bv[k]
•   until (g_max ≤ 0)
• return G(V, E)
```