
COMPUTER ALGORITHMS

Sartaj Sahni *University of Minnesota*

I. Computers and Algorithms	358
II. Algorithm Design	360
III. Performance Analysis and Measurement	363
IV. Lower Bounds	368
V. NP-Hard and NP-Complete Problems	369
VI. Coping with Complexity	372
VII. Summary	375

GLOSSARY

Algorithm: Sequence of well-defined instructions the execution of which results in the solution of a specific problem. The instructions are unambiguous, and each can be performed in a finite amount of time. Furthermore, the execution of all the instructions together takes only a finite amount of time.

Approximation algorithm: Algorithm that is guaranteed to produce solutions whose value is within some prespecified amount of the value of an optimal solution.

Asymptotic analysis: Analysis of the performance of an algorithm for large problem instances. Typically, the time and space requirements are analyzed and provided as a function of parameters that reflect properties of the problem instance to be solved. Asymptotic notation (e.g., big "oh," theta, omega) is used.

Deterministic algorithm: Algorithm in which the outcome of each step is well defined and determined by the values of the variables (if any) involved in the step. For example, the value of $x + y$ is determined by the values of x and y .

Heuristic: Rule of thumb employed in an algorithm to improve its performance (time and space requirements or quality of solution produced). This rule may be very effective in certain instances and ineffective in others.

Lower bound: Defined with respect to a problem. A lower bound on the resources (time or space) needed to solve a specified problem has the property that the problem cannot be solved by any algorithm that uses less resource than the lower bound.

Nondeterministic algorithm: Algorithm that may contain some steps whose outcome is determined by selecting from a set of permissible outcomes. There are no rules determining how the selection is to be made. Rather, such an algorithm terminates in one of two modes: success and failure. It is required that, whenever possible, the selection of the outcomes of individual steps be done in such a way that the algorithm terminates successfully.

NP-Complete problem: Decision problem (one for which the solution is "yes" or "no") that has the following property. The decision problem can be solved in polynomial deterministic time iff all decision problems that can be solved in nondeterministic polynomial time are also solvable in deterministic polynomial time.

NP-Hard problem: Problem for which the following is true. If this problem can be solved in polynomial deterministic time, then all decision problems that can be solved in nondeterministic polynomial time are also solvable in deterministic polynomial time.

Performance: Amount of resources (i.e., amount of computer time and memory) required by an algorithm. If the algorithm does not guarantee optimal solutions, the term *performance* is also used to include some measure of the quality of the solutions produced.

Probabilistically good algorithm: Algorithm that does not guarantee optimal solutions but generally does provide them.

Simulated annealing: Combinatorial optimization technique adapted from statistical mechanics. The technique attempts to find solutions that have value close to optimal. It does so by simulating the physical process of annealing a metal.

Stepwise refinement: Program development method in which the final computer program is arrived at in a sequence of steps. The first step begins close to the problem specification. Each step is a refinement of the preceding one and gets one closer to the final program. This technique simplifies both the programming task and the task of proving the final program correct.

Usually good algorithm: Algorithm that generally provides optimal solutions using a small amount of computing resources. At other times, the resources required may be prohibitively large.

In order to get a computer to solve a problem, it is necessary to provide it with a sequence of instructions that if followed faithfully will result in the desired solution. This sequence of instructions is called a computer algorithm. When a computer algorithm is specified in a language the computer "understands" (i.e., a programming language), it is called a program. The topic of computer algorithms deals with methods of developing algorithms as well as methods of analyzing algorithms to determine the amount of computer resources (time and memory) required by them to solve a problem and methods of deriving lower bounds on the resources required by any algorithm to solve a specific problem. Finally, for certain problems that are difficult to solve (e.g., when the computer resources required are impractically large), heuristic methods are used. [See COMPUTER LOGIC; PROGRAMMING LANGUAGES.]

I. Computers and Algorithms

We begin by discussing the two words—*computer* and *algorithm*—that comprise the title of this article. The word *computer* is defined in *Webster's Third New International Dictionary* as "one that computes." By this definition, almost every human is a computer. Certainly each of us has at one time or other computed the number of days before school ends. More serious computing has been done by humans for

thousands of years. In fact, the Babylonians used formulas to compute the areas of surfaces and the volumes of solids.

Generally, however, the word *computer* refers to a nonhuman device that computes. Early devices in this category include the sun dial and the Chinese abacus. Contemporary computing devices include microcomputers, such as the Apple® and the IBM-PC®, and awesome supercomputers, such as the CRAY-XMP® and NASA's MPP®. These contemporary computing devices have had a very significant impact on our lives—far more significant than that of any other computing device created by humankind. Why is this so? [See COMPUTER ARCHITECTURE.]

There are essentially two reasons. The first is that the modern computer is significantly faster than any of its predecessors. Some home computers are capable of performing several hundred thousand computations per second, and the fastest supercomputers exceed a billion computations per second. This awesome speed, in itself, is not enough. Equally important is the capacity of modern computers to store programs and data. (A *program* is simply a sequence of instructions to the computer.)

To understand the significance of the latter reason, consider the following problem:

Mary intends to open a bank account with an initial deposit of \$100. She intends to deposit an additional \$100 into this account on the first day of each of the next 19 months for a total of 20 deposits (including the initial deposit). The account pays interest at a rate of 5% per annum compounded monthly. Her initial deposit is also on the first of the month. Mary would like to know what the balance in her account will be at the end of each of the 20 months in which she will be making a deposit.

In order to solve this problem, we need to know how much interest is earned each month. Since the annual interest rate is 5%, the monthly interest rate is $\frac{5}{12}\%$. Consequently, the balance at the end of a month is

$$\begin{aligned} & (\text{initial balance} + \text{interest}) \\ &= (\text{initial balance}) * (1 + \frac{5}{1200}) \\ &= \frac{241}{240} (\text{initial balance}) \end{aligned}$$

Having performed this analysis, we can proceed to compute the balance at the end of each month using the following steps:

