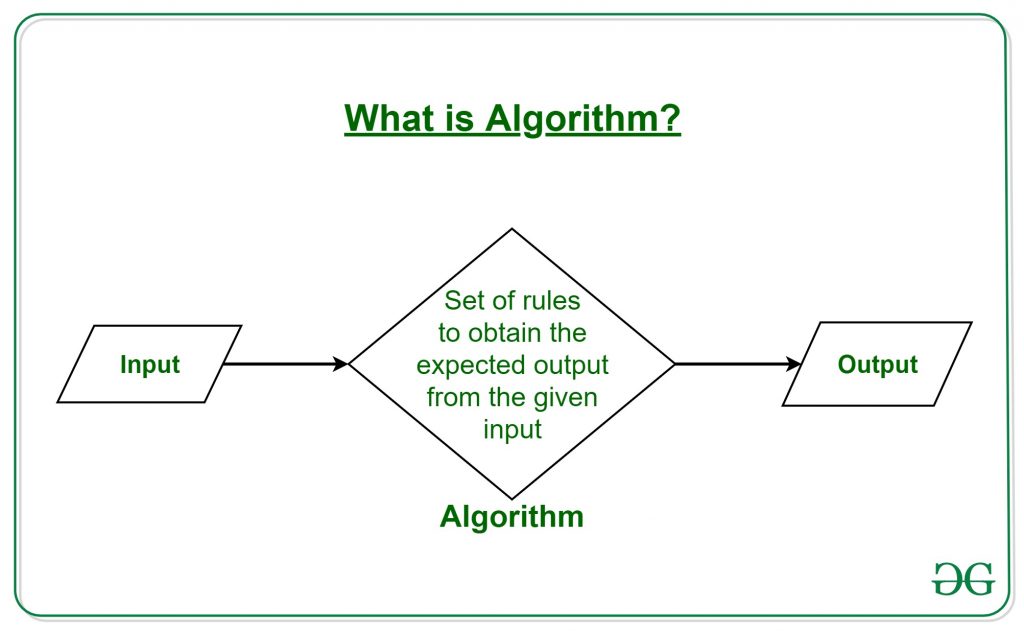
Algorithms

**What is an Algorithm?**

The word [**Algorithm**](https://www.geeksforgeeks.org/fundamentals-of-algorithms/) means ” *A  set of rules to be followed in calculations or other problem-solving operations*” Or ” *A procedure for solving a mathematical problem in a finite number of steps that frequently by recursive operations*“.

Therefore Algorithm refers to a sequence of finite steps to solve a particular problem.

**Algorithms can be simple and complex depending on what you want to achieve.**



**Types of Algorithms:**

There are several types of algorithms available. Some important algorithms are:

**1.**[**Brute Force Algorithm:**](https://www.geeksforgeeks.org/brute-force-approach-and-its-pros-and-cons/) It is the simplest approach for a problem. A brute force algorithm is the first approach that comes to finding when we see a problem.

**2.**[**Recursive Algorithm**](https://www.geeksforgeeks.org/recursion/)**:** A recursive algorithm is based on [recursion](http://www.geeksforgeeks.org/recursion/). In this case, a problem is broken into several sub-parts and called the same function again and again.

**3.**[**Backtracking Algorithm**](https://www.geeksforgeeks.org/backtracking-algorithms/)**:** The backtracking algorithm basically builds the solution by searching among all possible solutions. Using this algorithm, we keep on building the solution following criteria. Whenever a solution fails we trace back to the failure point and build on the next solution and continue this process till we find the solution or all possible solutions are looked after.

**4.**[**Searching Algorithm**](https://www.geeksforgeeks.org/searching-algorithms/)**:** Searching algorithms are the ones that are used for searching elements or groups of elements from a particular data structure. They can be of different types based on their approach or the data structure in which the element should be found.

**5.**[**Sorting Algorithm**](https://www.geeksforgeeks.org/sorting-algorithms/)**:** Sorting is arranging a group of data in a particular manner according to the requirement. The algorithms which help in performing this function are called sorting algorithms. Generally sorting algorithms are used to sort groups of data in an increasing or decreasing manner.

Advertisement

**6.**[**Hashing Algorithm**](https://www.geeksforgeeks.org/hashing-set-1-introduction/)**:** Hashing algorithms work similarly to the searching algorithm. But they contain an index with a key ID. In hashing, a key is assigned to specific data.

**7.**[**Divide and Conquer Algorithm**](http://www.geeksforgeeks.org/divide-and-conquer-introduction/)**:** This algorithm breaks a problem into sub-problems, solves a single sub-problem and merges the solutions together to get the final solution. It consists of the following three steps:

* Divide
* Solve
* Combine

**8.**[**Greedy Algorithm**](http://www.geeksforgeeks.org/greedy-algorithms/)**:** In this type of algorithm the solution is built part by part. The solution of the next part is built based on the immediate benefit of the next part. The one solution giving the most benefit will be chosen as the solution for the next part.

**9.**[**Dynamic Programming Algorithm**](https://www.geeksforgeeks.org/dynamic-programming/)**:** This algorithm uses the concept of using the already found solution to avoid repetitive calculation of the same part of the problem. It divides the problem into smaller overlapping subproblems and solves them.

**10.**[**Randomized Algorithm**](https://www.geeksforgeeks.org/randomized-algorithms/)**:** In the randomized algorithm we use a random number so it gives immediate benefit. The random number helps in deciding the expected outcome.

**Topics:**

* [Analysis of Algorithms](https://www.geeksforgeeks.org/fundamentals-of-algorithms/?ref=shm#analysis)
* [Searching and Sorting](https://www.geeksforgeeks.org/fundamentals-of-algorithms/?ref=shm#searchandsort)
* [Greedy Algorithms](https://www.geeksforgeeks.org/fundamentals-of-algorithms/?ref=shm#greedy)
* [Dynamic Programming](https://www.geeksforgeeks.org/fundamentals-of-algorithms/?ref=shm#dp)
* [Pattern Searching](https://www.geeksforgeeks.org/fundamentals-of-algorithms/?ref=shm#pattern)
* [Backtracking](https://www.geeksforgeeks.org/fundamentals-of-algorithms/?ref=shm#backtracking)
* [Divide and Conquer](https://www.geeksforgeeks.org/fundamentals-of-algorithms/?ref=shm#divideandconquer)
* [Geometric Algorithms](https://www.geeksforgeeks.org/fundamentals-of-algorithms/?ref=shm#geometric)
* [Mathematical Algorithms](https://www.geeksforgeeks.org/fundamentals-of-algorithms/?ref=shm#mathematical)
* [Bit Algorithms](https://www.geeksforgeeks.org/fundamentals-of-algorithms/?ref=shm#bitalgo)
* [Graph Algorithms](https://www.geeksforgeeks.org/fundamentals-of-algorithms/?ref=shm#graph)
* [Randomized Algorithms](https://www.geeksforgeeks.org/fundamentals-of-algorithms/?ref=shm#randomized)
* [Branch and Bound](https://www.geeksforgeeks.org/fundamentals-of-algorithms/?ref=shm#branchandbound)
* [Quizzes](https://www.geeksforgeeks.org/fundamentals-of-algorithms/?ref=shm#quiz)

**Analysis of Algorithms:**

1. [Asymptotic Analysis](https://www.geeksforgeeks.org/analysis-of-algorithms-set-1-asymptotic-analysis/)
2. [Worst, Average and Best Cases](https://www.geeksforgeeks.org/analysis-of-algorithms-set-2-asymptotic-analysis/)
3. [Asymptotic Notations](https://www.geeksforgeeks.org/analysis-of-algorithms-set-3asymptotic-notations/)
4. [Lower and Upper Bound Theory](https://www.geeksforgeeks.org/lower-and-upper-bound-theory/)
5. [Introduction to Amortized Analysis](https://www.geeksforgeeks.org/analysis-algorithm-set-5-amortized-analysis-introduction/)
6. [What does ‘Space Complexity’ mean?](https://www.geeksforgeeks.org/g-fact-86/)
7. [Polynomial Time Approximation Scheme](https://www.geeksforgeeks.org/polynomial-time-approximation-scheme/)
8. [Accounting Method | Amortized Analysis](https://www.geeksforgeeks.org/accounting-method-amortized-analysis/)
9. [Potential Method in Amortized Analysis](https://www.geeksforgeeks.org/potential-method-in-amortized-analysis/)

**Searching and Sorting:**

1. [Introduction to Searching Algorithms](https://www.geeksforgeeks.org/searching-algorithms/)
2. [Introduction to Sorting Algorithm](https://www.geeksforgeeks.org/sorting-algorithms/)
3. [Stable and Unstable Sorting Algorithms](https://www.geeksforgeeks.org/stable-and-unstable-sorting-algorithms/)
4. [Lower bound for comparison based sorting algorithms](https://www.geeksforgeeks.org/lower-bound-on-comparison-based-sorting-algorithms/)
5. [Can Run Time Complexity of a comparison-based sorting algorithm be less than N logN?](https://www.geeksforgeeks.org/can-run-time-complexity-of-a-comparison-based-sorting-algorithm-be-less-than-n-logn/)
6. [Which sorting algorithm makes minimum number of memory writes?](https://www.geeksforgeeks.org/which-sorting-algorithm-makes-minimum-number-of-writes/)

**Greedy Algorithms:**

1. [Introduction to Greedy Algorithms](https://www.geeksforgeeks.org/greedy-algorithms/)
2. [Activity Selection Problem](https://www.geeksforgeeks.org/greedy-algorithms-set-1-activity-selection-problem/)
3. [Huffman Coding](https://www.geeksforgeeks.org/greedy-algorithms-set-3-huffman-coding/)
4. [Job Sequencing Problem](https://www.geeksforgeeks.org/job-sequencing-problem-set-1-greedy-algorithm/)
5. [Quiz on Greedy Algorithms](http://geeksquiz.com/algorithms/greedy-algorithms/)
6. [Minimum Number of Platforms Required for a Railway/Bus Station](https://www.geeksforgeeks.org/minimum-number-platforms-required-railwaybus-station/)

**Dynamic Programming:**

1. [Introduction to Dynamic Programming](https://www.geeksforgeeks.org/dynamic-programming/)
2. [Overlapping Subproblems Property](https://www.geeksforgeeks.org/dynamic-programming-set-1/)
3. [Optimal Substructure Property](https://www.geeksforgeeks.org/dynamic-programming-set-2-optimal-substructure-property/)
4. [Longest Increasing Subsequence](https://www.geeksforgeeks.org/dynamic-programming-set-3-longest-increasing-subsequence/)
5. [Longest Common Subsequence](https://www.geeksforgeeks.org/dynamic-programming-set-4-longest-common-subsequence/)
6. [Min Cost Path](https://www.geeksforgeeks.org/dynamic-programming-set-6-min-cost-path/)
7. [Coin Change](https://www.geeksforgeeks.org/dynamic-programming-set-7-coin-change/)
8. [Matrix Chain Multiplication](https://www.geeksforgeeks.org/dynamic-programming-set-8-matrix-chain-multiplication/)
9. [0-1 Knapsack Problem](https://www.geeksforgeeks.org/dynamic-programming-set-10-0-1-knapsack-problem/)
10. [Longest Palindromic Subsequence](https://www.geeksforgeeks.org/dynamic-programming-set-12-longest-palindromic-subsequence/)
11. [Palindrome Partitioning](https://www.geeksforgeeks.org/dynamic-programming-set-17-palindrome-partitioning/)

**Pattern Searching:**

1. [Introduction to Pattern Searching](https://www.geeksforgeeks.org/algorithms-gq/pattern-searching/)
2. [Naive Pattern Searching](https://www.geeksforgeeks.org/searching-for-patterns-set-1-naive-pattern-searching/)
3. [KMP Algorithm](https://www.geeksforgeeks.org/searching-for-patterns-set-2-kmp-algorithm/)
4. [Rabin-Karp Algorithm](https://www.geeksforgeeks.org/searching-for-patterns-set-3-rabin-karp-algorithm/)
5. [Pattern Searching using a Trie of all Suffixes](https://www.geeksforgeeks.org/pattern-searching-using-trie-suffixes/)
6. [Aho-Corasick Algorithm for Pattern Searching](https://www.geeksforgeeks.org/aho-corasick-algorithm-pattern-searching/)
7. [Z algorithm (Linear time pattern searching Algorithm)](https://www.geeksforgeeks.org/z-algorithm-linear-time-pattern-searching-algorithm/)

**Backtracking:**

1. [Introduction to Backtracking](https://www.geeksforgeeks.org/backtracking-algorithms/)
2. [Print all permutations of a given string](https://www.geeksforgeeks.org/write-a-c-program-to-print-all-permutations-of-a-given-string/)
3. [The Knight’s tour problem](https://www.geeksforgeeks.org/backtracking-set-1-the-knights-tour-problem/)
4. [Rat in a Maze](https://www.geeksforgeeks.org/backttracking-set-2-rat-in-a-maze/)
5. [N Queen Problem](https://www.geeksforgeeks.org/backtracking-set-3-n-queen-problem/)
6. [Subset Sum](https://www.geeksforgeeks.org/backttracking-set-4-subset-sum/)
7. [m Coloring Problem](https://www.geeksforgeeks.org/backttracking-set-5-m-coloring-problem/)
8. [Hamiltonian Cycle](https://www.geeksforgeeks.org/backtracking-set-7-hamiltonian-cycle/)
9. [Sudoku](https://www.geeksforgeeks.org/backtracking-set-7-suduku/)

**Divide and Conquer:**

1. [Introduction to Divide and Conquer](https://www.geeksforgeeks.org/divide-and-conquer/)
2. [Merge Sort](https://www.geeksforgeeks.org/merge-sort/)
3. [Write your own pow(x, n) to calculate x\*n](https://www.geeksforgeeks.org/write-a-c-program-to-calculate-powxn/)
4. [Count Inversions](https://www.geeksforgeeks.org/counting-inversions/)
5. [Closest Pair of Points](https://www.geeksforgeeks.org/closest-pair-of-points/)
6. [Strassen’s Matrix Multiplication](https://www.geeksforgeeks.org/strassens-matrix-multiplication/)

**Geometric Algorithm:**

1. [Introduction to Geometric Algorithms](https://www.geeksforgeeks.org/geometric-algorithms/)
2. [Closest Pair of Points | O(nlogn) Implementation](https://www.geeksforgeeks.org/closest-pair-of-points-onlogn-implementation/)
3. [How to check if a given point lies inside or outside a polygon?](https://www.geeksforgeeks.org/how-to-check-if-a-given-point-lies-inside-a-polygon/)
4. [How to check if two given line segments intersect?](https://www.geeksforgeeks.org/check-if-two-given-line-segments-intersect/)
5. [Given n line segments, find if any two segments intersect](https://www.geeksforgeeks.org/given-a-set-of-line-segments-find-if-any-two-segments-intersect/)
6. [How to check if given four points form a square](https://www.geeksforgeeks.org/check-given-four-points-form-square/)
7. [Convex Hull using Jarvis’ Algorithm or Wrapping](https://www.geeksforgeeks.org/convex-hull-using-jarvis-algorithm-or-wrapping/)

**Mathematical Algorithms:**

1. [Introduction to Mathematical Algorithms](https://www.geeksforgeeks.org/mathematical-algorithms/)
2. [Write an Efficient Method to Check if a Number is Multiple of 3](https://www.geeksforgeeks.org/write-an-efficient-method-to-check-if-a-number-is-multiple-of-3/)
3. [Write a program to add two numbers in base 14](https://www.geeksforgeeks.org/write-a-program-to-add-two-numbers-in-base-14/)
4. [Program for Fibonacci numbers](https://www.geeksforgeeks.org/program-for-nth-fibonacci-number/)
5. [Average of a stream of numbers](https://www.geeksforgeeks.org/average-of-a-stream-of-numbers/)
6. [Multiply two integers without using multiplication, division and bitwise operators, and no loops](https://www.geeksforgeeks.org/multiply-two-numbers-without-using-multiply-division-bitwise-operators-and-no-loops/)
7. [Babylonian method for square root](https://www.geeksforgeeks.org/square-root-of-a-perfect-square/)
8. [Sieve of Eratosthenes](https://www.geeksforgeeks.org/sieve-of-eratosthenes/)
9. [Pascal’s Triangle](https://www.geeksforgeeks.org/pascal-triangle/)
10. [Given a number, find the next smallest palindrome](https://www.geeksforgeeks.org/given-a-number-find-next-smallest-palindrome-larger-than-this-number/)
11. [Program to add two polynomials](https://www.geeksforgeeks.org/program-add-two-polynomials/)
12. [Multiply two polynomials](https://www.geeksforgeeks.org/multiply-two-polynomials-2/)
13. [Count trailing zeroes in factorial of a number](https://www.geeksforgeeks.org/count-trailing-zeroes-factorial-number/)

**Bitwise Algorithms:**

1. [Introduction to Bitwise Algorithms](https://www.geeksforgeeks.org/bitwise-algorithms/)
2. [Little and Big Endian](https://www.geeksforgeeks.org/little-and-big-endian-mystery/)
3. [Detect opposite signs](https://www.geeksforgeeks.org/detect-if-two-integers-have-opposite-signs/)
4. [Swap bits](https://www.geeksforgeeks.org/swap-bits-in-a-given-number/)
5. [Turn off the rightmost set bit](https://www.geeksforgeeks.org/turn-off-the-rightmost-set-bit/)
6. [Rotate bits](https://www.geeksforgeeks.org/rotate-bits-of-an-integer/)
7. [Next higher number with same number of set bits](https://www.geeksforgeeks.org/next-higher-number-with-same-number-of-set-bits/)
8. [Swap two nibbles in a byte](https://www.geeksforgeeks.org/swap-two-nibbles-byte/)

**Graph Algorithms:**

1. [Introduction to Graph Algorithms](https://www.geeksforgeeks.org/graph-data-structure-and-algorithms/)
2. [BFS, DFS](https://www.geeksforgeeks.org/graph-data-structure-and-algorithms/#bfsndfs)
3. [Cycles in Graph](https://www.geeksforgeeks.org/graph-data-structure-and-algorithms/#cycle)
4. [Shortest paths](https://www.geeksforgeeks.org/graph-data-structure-and-algorithms/#shortest)
5. [MST](https://www.geeksforgeeks.org/graph-data-structure-and-algorithms/#MST)
6. [Topological Sorting](https://www.geeksforgeeks.org/graph-data-structure-and-algorithms/#topo)
7. [Connectivity](https://www.geeksforgeeks.org/graph-data-structure-and-algorithms/#connectivity)
8. [Max Flow](https://www.geeksforgeeks.org/graph-data-structure-and-algorithms/#maxflow)

**Randomized Algorithms:**

1. [Introduction to Randomized Algorithms](https://www.geeksforgeeks.org/randomized-algorithms/)
2. [Linearity of Expectation](https://www.geeksforgeeks.org/linearity-of-expectation/)
3. [Expected Number of Trials until Success](https://www.geeksforgeeks.org/expected-number-of-trials-before-success/)
4. [Randomized Algorithms | Set 0 (Mathematical Background)](https://www.geeksforgeeks.org/randomized-algorithms-set-0-mathematical-background/)
5. [Randomized Algorithms | Set 1 (Introduction and Analysis)](https://www.geeksforgeeks.org/randomized-algorithms-set-1-introduction-and-analysis/)
6. [Randomized Algorithms | Set 2 (Classification and Applications)](https://www.geeksforgeeks.org/randomized-algorithms-set-2-classification-and-applications/)
7. [Randomized Algorithms | Set 3 (1/2 Approximate Median)](https://www.geeksforgeeks.org/randomized-algorithms-set-3-12-approximate-median/)
8. [Reservoir Sampling](https://www.geeksforgeeks.org/reservoir-sampling/)

**Branch and Bound:**

1. [Branch and Bound | Set 1 (Introduction with 0/1 Knapsack)](https://www.geeksforgeeks.org/branch-and-bound-set-1-introduction-with-01-knapsack/)
2. [Branch and Bound | Set 2 (Implementation of 0/1 Knapsack)](https://www.geeksforgeeks.org/branch-and-bound-set-2-implementation-of-01-knapsack/)
3. [Branch and Bound | Set 3 (8 puzzle Problem)](https://www.geeksforgeeks.org/branch-bound-set-3-8-puzzle-problem/)
4. [Branch And Bound | Set 4 (Job Assignment Problem)](https://www.geeksforgeeks.org/branch-bound-set-4-job-assignment-problem/)
5. [Branch and Bound | Set 5 (N Queen Problem)](https://www.geeksforgeeks.org/branch-and-bound-set-4-n-queen-problem/)
6. [Branch And Bound | Set 6 (Traveling Salesman Problem)](https://www.geeksforgeeks.org/branch-bound-set-5-traveling-salesman-problem/)

**Quizzes:**

* [Analysis of Algorithms](http://geeksquiz.com/algorithms/analysis-of-algorithms/)
* [Sorting](http://geeksquiz.com/algorithms/searching-and-sorting/)
* [Divide and Conquer](http://geeksquiz.com/algorithms/divide-and-conquer/)
* [Greedy Algorithms](http://geeksquiz.com/algorithms/greedy-algorithms/)
* [Dynamic Programming](http://geeksquiz.com/algorithms/dynamic-programming/)
* [Backtracking](http://geeksquiz.com/algorithms/backtracking/)
* [Misc](http://geeksquiz.com/algorithms/misc-2/)
* [NP Complete](http://geeksquiz.com/algorithms/np-complete/)
* [Searching](http://geeksquiz.com/algorithms/searching/)
* [Analysis of Algorithms (Recurrences)](http://geeksquiz.com/algorithms/analysis-of-algorithms-recurrences/)
* [Recursion](http://geeksquiz.com/algorithms/recursion/)
* [Bit Algorithms](http://geeksquiz.com/algorithms/bit-algorithms/)
* [Graph Traversals](http://geeksquiz.com/algorithms/graph-traversals/)
* [Graph Shortest Paths](http://geeksquiz.com/algorithms/graph-shortest-paths/)
* [Graph Minimum Spanning Tree](http://geeksquiz.com/algorithms/graph-minimum-spanning-tree/)