|  |  |  |
| --- | --- | --- |
|  |  |  |
| [**Dynamic programming**](https://www.geeksforgeeks.org/top-algorithms-and-data-structures-for-competitive-programming/#algo2) | 1. [Longest Common Subsequence](https://www.geeksforgeeks.org/dynamic-programming-set-4-longest-common-subsequence/) 2. [Longest Increasing Subsequence](https://www.geeksforgeeks.org/dynamic-programming-set-3-longest-increasing-subsequence/) 3. [Edit Distance](https://www.geeksforgeeks.org/dynamic-programming-set-5-edit-distance/) 4. [Minimum Partition](https://www.geeksforgeeks.org/partition-a-set-into-two-subsets-such-that-the-difference-of-subset-sums-is-minimum/) 5. [Ways to Cover a Distance](https://www.geeksforgeeks.org/count-number-of-ways-to-cover-a-distance/) 6. [Longest Path In Matrix](https://www.geeksforgeeks.org/find-the-longest-path-in-a-matrix-with-given-constraints/) 7. [Subset Sum Problem](https://www.geeksforgeeks.org/dynamic-programming-subset-sum-problem/) 8. [Optimal Strategy for a Game](https://www.geeksforgeeks.org/dynamic-programming-set-31-optimal-strategy-for-a-game/) 9. [0-1 Knapsack Problem](https://www.geeksforgeeks.org/dynamic-programming-set-10-0-1-knapsack-problem/) 10. [Assembly Line Scheduling](https://www.geeksforgeeks.org/dynamic-programming-set-34-assembly-line-scheduling/) |  |
| [**Graph algorithms**](https://www.geeksforgeeks.org/top-algorithms-and-data-structures-for-competitive-programming/#algo1) | 1. [Breadth First Search (BFS)](https://www.geeksforgeeks.org/breadth-first-traversal-for-a-graph/) 2. [Depth First Search (DFS)](https://www.geeksforgeeks.org/depth-first-traversal-for-a-graph/) 3. [Shortest Path from source to all vertices \*\*Dijkstra\*\*](https://www.geeksforgeeks.org/greedy-algorithms-set-6-dijkstras-shortest-path-algorithm/) 4. [Shortest Path from every vertex to every other vertex \*\*Floyd Warshall\*\*](https://www.geeksforgeeks.org/dynamic-programming-set-16-floyd-warshall-algorithm/) 5. [Minimum Spanning tree \*\*Prim\*\*](https://www.geeksforgeeks.org/greedy-algorithms-set-5-prims-minimum-spanning-tree-mst-2/) 6. [Minimum Spanning tree \*\*Kruskal\*\*](https://www.geeksforgeeks.org/greedy-algorithms-set-2-kruskals-minimum-spanning-tree-mst/) 7. [Topological Sort](https://www.geeksforgeeks.org/topological-sorting/) 8. [Johnson’s algorithm](https://www.geeksforgeeks.org/johnsons-algorithm/) 9. [Articulation Points (or Cut Vertices) in a Graph](https://www.geeksforgeeks.org/articulation-points-or-cut-vertices-in-a-graph/) 10. [Bridges in a graph](https://www.geeksforgeeks.org/bridge-in-a-graph/) |  |
| [**Searching and Sorting:**](https://www.geeksforgeeks.org/top-algorithms-and-data-structures-for-competitive-programming/#algo3) | 1. [Binary Search](http://geeksquiz.com/binary-search/) 2. [Quick Sort](http://geeksquiz.com/quick-sort/) 3. [Merge Sort](http://geeksquiz.com/merge-sort/) 4. [Order Statistics](https://www.geeksforgeeks.org/kth-smallestlargest-element-unsorted-array-set-2-expected-linear-time/) 5. [KMP algorithm](https://www.geeksforgeeks.org/searching-for-patterns-set-2-kmp-algorithm/) 6. [Rabin karp](https://www.geeksforgeeks.org/searching-for-patterns-set-3-rabin-karp-algorithm/) 7. [Z’s algorithm](https://www.geeksforgeeks.org/z-algorithm-linear-time-pattern-searching-algorithm/) 8. [Aho Corasick String Matching](https://www.geeksforgeeks.org/aho-corasick-algorithm-pattern-searching/) 9. [Counting Sort](https://www.geeksforgeeks.org/counting-sort/) 10. Manacher’s algorithm: [Part 1](https://www.geeksforgeeks.org/manachers-algorithm-linear-time-longest-palindromic-substring-part-1/), [Part 2](https://www.geeksforgeeks.org/manachers-algorithm-linear-time-longest-palindromic-substring-part-2/) and [Part 3](https://www.geeksforgeeks.org/manachers-algorithm-linear-time-longest-palindromic-substring-part-3-2/) |  |
| [**Number theory and Other Mathematical**](https://www.geeksforgeeks.org/top-algorithms-and-data-structures-for-competitive-programming/#algo4) | 1. [Primality Test | Set 1 (Introduction and School Method)](https://www.geeksforgeeks.org/primality-test-set-1-introduction-and-school-method/) 2. [Primality Test | Set 2 (Fermat Method)](https://www.geeksforgeeks.org/primality-test-set-2-fermet-method/) 3. [Primality Test | Set 3 (Miller–Rabin)](https://www.geeksforgeeks.org/primality-test-set-3-miller-rabin/) 4. [Sieve of Eratosthenes](https://www.geeksforgeeks.org/sieve-of-eratosthenes/) 5. [Segmented Sieve](https://www.geeksforgeeks.org/segmented-sieve/) 6. [Wilson’s Theorem](https://www.geeksforgeeks.org/wilsons-theorem/) 7. [Prime Factorisation](https://www.geeksforgeeks.org/print-all-prime-factors-of-a-given-number/) 8. [Pollard’s rho algorithm](https://www.geeksforgeeks.org/pollards-rho-algorithm-prime-factorization/) |  |
| [**Geometrical and Network Flow Algorithms**](https://www.geeksforgeeks.org/top-algorithms-and-data-structures-for-competitive-programming/#algo6) | 1. [Convex Hull](https://www.geeksforgeeks.org/convex-hull-set-1-jarviss-algorithm-or-wrapping/) 2. [Graham Scan](https://www.geeksforgeeks.org/convex-hull-set-2-graham-scan/) 3. [Line Intersection](https://www.geeksforgeeks.org/check-if-two-given-line-segments-intersect/) 4. [Interval Tree](https://www.geeksforgeeks.org/interval-tree/) 5. [Matrix Exponentiation](https://www.geeksforgeeks.org/matrix-exponentiation/) and [this](http://zobayer.blogspot.in/2010/11/matrix-exponentiation.html) 6. [Maxflow Ford Furkerson Algo and Edmond Karp Implementation](https://www.geeksforgeeks.org/ford-fulkerson-algorithm-for-maximum-flow-problem/) 7. [Min cut](https://www.geeksforgeeks.org/minimum-cut-in-a-directed-graph/) 8. [Stable Marriage Problem](https://www.geeksforgeeks.org/stable-marriage-problem/) 9. [Hopcroft–Karp Algorithm for Maximum Matching](https://www.geeksforgeeks.org/hopcroft-karp-algorithm-for-maximum-matching-set-1-introduction/) 10. [Dinic’s algo](https://www.geeksforgeeks.org/dinics-algorithm-maximum-flow/) and [e-maxx](http://e-maxx.ru/algo/dinic) |  |
| [**Data Structures**](https://www.geeksforgeeks.org/top-algorithms-and-data-structures-for-competitive-programming/#algo7) | 1. [Binary Indexed Tree or Fenwick tree](https://www.geeksforgeeks.org/binary-indexed-tree-or-fenwick-tree-2/) 2. [Segment Tree](https://www.geeksforgeeks.org/segment-tree-set-1-range-minimum-query/) ([RMQ](https://www.geeksforgeeks.org/segment-tree-set-1-range-minimum-query/), [Range Sum](https://www.geeksforgeeks.org/segment-tree-set-1-sum-of-given-range/) and [Lazy Propagation](https://www.geeksforgeeks.org/lazy-propagation-in-segment-tree/)) 3. [K-D tree](https://www.geeksforgeeks.org/k-dimensional-tree/) (See [insert](https://www.geeksforgeeks.org/k-dimensional-tree/), [minimum](https://www.geeksforgeeks.org/k-dimensional-tree-set-2-find-minimum/) and [delete](https://www.geeksforgeeks.org/k-dimensional-tree-set-3-delete/)) 4. [Union Find Disjoint Set](https://www.geeksforgeeks.org/union-find/) ([Cycle Detection](https://www.geeksforgeeks.org/union-find-algorithm-set-2-union-by-rank/) and [By Rank and Path Compression](https://www.geeksforgeeks.org/union-find-algorithm-set-2-union-by-rank/)) 5. [Tries](https://www.geeksforgeeks.org/trie-insert-and-search/) 6. Suffix array ([this](http://web.stanford.edu/class/cs97si/suffix-array.pdf), [this](https://www.geeksforgeeks.org/suffix-array-set-1-introduction/) and [this](https://www.geeksforgeeks.org/suffix-array-set-2-a-nlognlogn-algorithm/)) 7. [Sparse table](https://www.geeksforgeeks.org/range-minimum-query-for-static-array/) 8. [Suffix automata](https://www.geeksforgeeks.org/searching-for-patterns-set-5-finite-automata/) 9. [Suffix automata II](https://www.geeksforgeeks.org/pattern-searching-set-5-efficient-constructtion-of-finite-automata/) 10. [LCA and RMQ](https://www.geeksforgeeks.org/find-lca-in-binary-tree-using-rmq/) |  |

AGGCTATCACCTGACCTCCAGGCCGATGCCC TAGCTATCACGACCGCGGTCGATTTGCCCGAC

-**AG**G**CTATCAC**CT**GACC**T**C**CA**GG**C**CGA**--**TGCCC**---

T**AG**-**CTATCAC**--**GACC**G**C**--**GG**T**CGA**TT**TGCCC**GAC