

Algoritms

CodeChef

Competitive Programming


Algorithms

List Question

Computer Programming

What are some good questions on CodeChef from which I will learn more algorithms?

4 Answers



Mukesh Pathak

Programming- Only thing i never get bored of.

Updated 82w ago · Upvoted by Anup Kalbalia, CodeChef organizing team and Bhavik Dhandhalya, Hacker, Problem Setter on SPOJ & HackerEarth, ACM ICPC 2016 Participant

It will be helpful to everyone in many ways. I request everyone to contribute to this list by providing links to tutorials, problems, etc. I will keep updating this list regularly.

1. Binary Search : [Tutorial](#), [Problems](#), [Tutorial](#), [Implementation](#), [Problem](#)
2. Quicksort : [Tutorial](#), [Implementation](#), [Tutorial](#)
3. Merge Sort : [Tutorial](#), [Implementation](#), [Tutorial](#)
4. Suffix Array : [Tutorial](#), [Tutorial](#), [Implementation](#), [Tutorial](#), [Implementation](#), [Problem](#), [Problem](#)
5. Knuth-Morris-Pratt Algorithm (KMP) : [Tutorial](#), [Tutorial](#), [Implementation](#), [Tutorial](#), [Problem](#)
6. Rabin-Karp Algorithm : [Tutorial](#), [Implementation](#), [Tutorial](#), [Problem](#), [Problem](#)
7. Tries : [Tutorial](#), [Problems](#), [Tutorial](#) : I, II, [Tutorial](#), [Problem](#), [Problem](#), [Problem](#)
8. Depth First Traversal of a graph : [Tutorial](#), [Impelementation](#), [Tutorial](#), [Problems](#), [Problem](#), [Problem](#), [Problem](#)
9. Breadth First Traversal of a graph : [Tutorial](#), [Impelementation](#), [Tutorial](#), [Problems](#), [Problem](#), [Problem](#), [Problem](#), [Problem](#), [Flood Fill](#)
10. Dijkstra's Algorithm : [Tutorial](#), [Problems](#), [Problem](#), [Tutorial\(greedy\)](#), [Tutorial \(with heap\)](#), [Implementation](#), [Problem](#), [Problem](#)
11. Binary Indexed Tree : [Tutorial](#), [Problems](#), [Tutorial](#), [Original Paper](#), [Tutorial](#), [Tutorial](#), [Problem](#), [Problem](#), [Problem](#), [Problem](#), [Problem](#), [Problem](#), [Problem](#)
12. Segment Tree (with lazy propagation) : [Tutorial](#), [Implementation](#), [Tutorial](#), [Tutorial](#), [Problems](#), [Implementation](#), [Tutorial](#), [Implementation and Various Uses](#), [Persistent Segment Tree](#), [problems same as BIT](#), [Problem](#), [Problem](#)/*HLD is used as well*
13. Z algorithm : [Tutorial](#), [Problem](#), [Tutorial](#), [problems same as KMP](#).
14. Floyd Warshall Algorithm : [Tutorial](#), [Implementation](#), [Problem](#), [Problem](#)
15. Sparse Table(RMQ) : [Tutorial](#), [Problems](#), [Tutorial](#), [Implementation\(C++\)](#), [Java implementation](#)
16. Heap / Priority Queue / Heapsort : [Implementation](#), [Explanation](#), [Tutorial](#), [Implementation](#), [Problem](#), [Chapter from CLRS](#)
17. [Modular Multiplicative Inverse](#)
18. [nCr % M](#)
19. Suffix Automaton : [Detailed Paper](#), [Tutorial](#), [Implementation \(I\)](#), [Tutorial](#), [Implementation \(II\)](#), [Problem](#), [Problem](#), [Problem](#), [Problem](#), [Tutorial](#), [Implementation](#)
20. Lowest Common Ancestor : [Tutorial](#), [Problems](#), [Paper](#), [Paper](#), [Problem](#), [Problem](#), [Problem](#)
21. Counting Inversions : [Divide and Conquer](#), [Segment Tree](#), [Fenwick Tree](#), [Problem](#)
22. [Euclid's Extended Algorithm](#)
23. Suffix Tree : [Tutorial](#), [Tutorial](#), [Intro](#), [Construction](#) : I, II, [Implementation](#), [Implementation](#), [Problem](#), [Problem](#), [Problem](#), [Problem](#)
24. Dynamic Programming : [Chapter from CLRS\(essential\)](#), [Tutorial](#), [Problems](#), [Problem](#), [Problem](#), [Problem](#), [Problem](#), [Tutorial](#), [Problem](#), [Problem](#), [Problem](#), [Longest Increasing Subsequence](#), [Bitmask DP](#), [Bitmask DP](#), [Optimization](#), [Problem](#), [Problem](#), [Problem](#), [Problem](#), [Problem](#), [Problem](#), [DP on Trees](#) : I, II
25. Basic Data Structures : [Tutorial](#), [Stack Implementation](#), [Queue Implementation](#), [Tutorial](#), [Linked List Implementation](#)
26. [Logarithmic Exponentiation](#)
27. Graphs : [Definition](#), [Representation](#), [Definition](#), [Representation](#), [Problem](#), [Problem](#)
28. Minimum Spanning Tree : [Tutorial](#), [Tutorial](#), [Kruskal's Implementation](#), [Prim's Implementation](#), [Problem](#), [Problem](#), [Problem](#), [Problem](#), [Problem](#)
29. [Efficient Prime Factorization](#)
30. Combinatorics : [Tutorial](#), [Problems](#), [Problem](#), [Tutorial](#)
31. Union Find/Disjoint Set : [Tutorial](#), [Tutorial](#), [Problems](#), [Problem](#), [Problem](#), [Problem](#)
32. Knapsack problem : [Solution](#), [Implementation](#)
33. Aho-Corasick String Matching Algorithm : [Tutorial](#), [Implementation](#)

Related Questions

What are some must solve from questions (from CodeChef or TopCoder or any other) which help in learning algorithms and data structures editorials?

Which is the easiest question on CodeChef?

Is there any website where you can learn an algorithm and then there would be questions relating to it organised from novice to experienced?

CodeChef: From which sources did Rudradev Basak, Pradeep George Mathias and Nikhil G learn algorithms?

What would be a good plan for learning algorithms (for competitive programming) in months from T. Cormen?

Which is good online video tutorial to start with learning algorithms from scratch?

Should I learn algorithm from a standard book or should try to derive it myself while solving questions from SPOJ and CodeChef?

What are good ways to learn Algorithms and Structures using Python and Programming contests like Codechef, Codeforces?

What are some good blogs for learning algorithms and competitive programming techniques?

What is a good site for learning algorithms and getting good questions to code?