Training for Competitive Programming +6

What basic data structures and algorithms should one learn before starting competitive programming?

https://en.wikipedia.org/wiki/SPOJ

66 Answers

Aman Goel, B.Tech Computer Science and Engineering, Indian Institute of Technology, Bombay (2017)
Answered Sep 20, 2014 · Upvoted by Krishna Mohan Shakya, M.Tech Computer Programming & Learning to Program, BITS Pilani (2019)

Originally Answered: What all basic data structures and algorithms one should learn before starting competitive programming?

I am sure this is going to help you (source : codechef) :

Binary Search : tutorial with problems , tutorial with implementation , problem
Quicksort : tutorial with implementation , tutorial with problems
Merge Sort : tutorial with implementation , tutorial with problems
Suffix Array : tutorial with implementation , tutorial with implementation , problem , problem , problem
Knuth-Morris-Pratt Algorithm (KMP) : tutorial , tutorial with implementation , problem , problem , problem
Rabin-Karp Algorithm : tutorial with implementation , tutorial , problem , problem
Tries : tutorial with problems , Tutorial : I, II , tutorial, problem , problem , problem
Depth First Traversal of a graph : tutorial with implementation , tutorial with problems , problem , problem , problem
Breadth First Traversal of a graph : tutorial with implementation , tutorial with problems , problem , problem , problem , Flood Fill
Dijkstra's Algorithm : tutorial with problems , problem , tutorial(greedy) , tutorial (with heap) , implementation , problem , problem
Binary Indexed Tree : tutorial with problems , tutorial , original paper , tutorial , tutorial , problem , problem , problem , problem ,problem , problem
Segment Tree (with lazy propagation) : tutorial with implementation , tutorial , tutorial with problems and implementation ,tutorial with implementation , Persistent Segment Tree , problems same as BIT, problem
Z algorithm : tutorial with problem , tutorial , problems same as KMP.
Floyd Warshall Algorithm : tutorial with implementation , problem , problem
Sparse Table(RMQ) : tutorial with problems , tutorial with implementation(C++) , java implementation
Heap / Priority Queue / Heapsort : implementation with explanation , tutorial , implementation , problem , reading the chapter from clrs is highly recommended.
Modular Multiplicative Inverse
nCr % M
Suffix Automaton : detailed paper , tutorial with implementation (I) , tutorial with implementation (II) , problem , problem ,problem , problem , tutorial with implementation
Lowest Common Ancestor : tutorial with problems , tutorial(binary tree) with implementation , detailed paper for LCA in DAGs , problem , problem
Counting Inversions : Divide and Conquer , Segment Tree, Fenwick Tree , problem
Euclid's Extended Algorithm
Suffix Tree : tutorial , tutorial , tutorial , tutorial , implementation , implementation , problem , problem , problem , problem
Dynamic Programming : chapter from clrs(essential), tutorial with problems , problem , problem , problem , problem , problem ,tutorial, problem , problem , problem , longest increasing subsequence , bitmask dp , bitmask

Related Questions

What is the best strategy to improve my skills in competitive programming in C++ in 2-3 months?

What should I learn first: data structures or algorithms?

How do I start learning or strengthen my knowledge of data structures and algorithms?

What are some of the best ways to learn programming?

Can anyone give me the list of basics to advance one by one algorithm and data structure learning topics for competitive programming?

What are the best books on algorithms and data structures?

Ask Question · More Related Questions

Basic Data Structures : tutorial , Stack implementation , Queue implementation and tutorial , Linked list implementation
logarithmic exponentiation
Graphs : definition, representation , definition, representation , problem , problem
Minimum Spanning Tree : tutorial , tutorial with kruskal's implementation , Prim's implementation , problem , problem , problem ,problem , problem
Efficient Prime Factorization
Combinatorics : tutorial with problems , problem
Union Find/Disjoint Set : tutorial , tutorial with problems , problem , problem , problem
Knapsack problem : solution and implementation
Aho-Corasick String Matching Algorithm : tutorial , implementation , problem , problem , problem , problem
Strongly Connected Components : tutorial with implementation , tutorial , problem , problem , problem
Bellman Ford algorithm : tutorial with implementation , tutorial with implementation , problem , problem
Heavy-light Decomposition : tutorial , tutorial , implementation , implementation , problem , problem , problem , problem , problem ,problem , problem
Convex Hull : tutorial with jarvis's algorithm implementation , tutorial with graham scan , tutorial , implementation , problem ,problem , problem , problem
Line Intersection : tutorial with imp. , tutorial with problems
Sieve of Erastothenes
Interval Tree : tutorial with implementation , problem , problem , problem , problem , problem , problem , tutorial
Counting Sort
Probabilities
Building up the recurrence matrix to compute recurrences in O(logN) time
Network flow : (Max Flow)Tutorial : I, II , Max flow(ford-fulkerson) tutorial with implementation , (Min cut) tutorial with implementation , (Min cost flow)Tutorial : I, II, III , Dinic's algorithm with implementation , Max flow by Edmonds Karp with implementation , problem , problem , problem , problem , problem , problem , problem , problem , problem , problem , problem ,problem , problem , problem , problem
K-d tree : tutorial , tutorial , implementation , problem
Deque
Binary Search Tree : tutorial with implementation , Searching and insertion , Deletion
Quick Select : implementation , implementation
Treap/Cartesian Tree : tutorial(detailed) , tutorial with implementation , problem
Game Theory : detailed paper , tutorial with problems , Grundy numbers , Tutorial with example problems - I, II, III, IV , tutorial with problems , problem , problem , problem , problem , problem , problem , problem , problem , problem , problem , problem , problem , Nim
STL (C++) : I, II , Crash Course
Maximum Bipartite Matching
Manacher's Algorithm : implementation , tutorial , tutorial with implementation , tutorial with implementation , problem , problem ,problem
Miller-Rabin Primality Test : Code
Stable Marriage Problem
Hungarian Algorithm
Sweep line Algorithm : I , II
LCP : tutorial with implementation , tutorial with implementation
Gaussian Elimination
Pollard Rho Integer Factorization , problem
Topological Sorting
Detecting Cycles in a Graph : Directed - I , II Undirected : I
Geometry : Basics
Backtracking : N queens problem , Tug of War , Sudoku
Eulerian and Hamiltonian Paths : tutorial , tutorial , (eulerian path and

Meet in the Middle : tutorial , implementation
Arbitrary Precision Integer(BigInt) , II
Radix Sort , Bucket Sort
Johnson's Algorithm : tutorial , tutorial , implementation
Maximal Matching in a General Graph : Blossom/Edmond's Algorithm with implementation , Tutte Matrix , problem
Recursion : I, II , Towers of Hanoi with explanation
Inclusion and Exclusion Principle

Unnikrishnan Menon
Bookmarked! :)

Khalid Jamal Abdulnasser, ACM contestant and coach.
Answered Apr 2, 2016 · Upvoted by Medo Nasser, ACM ICPC World Finalist, CS Student.

Originally Answered: which algorithms needed to learn for competitive programming ?

1. Basic data sturctures (arrays, queues, linked lists, etc.).
2. Bit manipulation.
3. Advanced data structures:
   a. Union-Find Disjoint Sets.
   b. Segment Tree.
   c. Binary Indexed Tree (a.k.a Fenwik Tree).
   d. Graph.
   e. Treap.
   f. Skip Lists.
   e. Some self balanced Binary Search trees (e.g. Red Black Trees).
4. Brute force and it's tricks and advanced techniques (such as, pruning, bitmasks, meet in the middle, iterative deepining etc.)
5. Binary Search (not only the basic code).
6. Greedy.
7. Dynamic programming and it's tricks and optimisations (Knuth optimisation, convex hull optimisation, bitmasks, etc.).
8. Graph algorithms:
   a. Traversal (DFS & BFS) algorithms and how to use them.
   b. Finding Connected Components.
   c. Flood Fill.
   d. Topological Sorting (the famous algorithm uses DFS but you should also know Kahn's algorithm that uses BFS as it has much applications).
   e. Bipartite Check.
   d. Finding Strongly Connected Components.
   f. Kruskal's and Prim's algorithms for finding the Minimum Spanning Tree of a graph and the variants of the problem.
   g. Dijkstra's algorithm for solving the Single Source Shortest Path (SSSP) Problem with out negaitive cycles.
   h. Bellman-Ford's algorithm for solving the SSSP problem with negative sycles.
   i. Floyd-Warshall's algorithm for solving the All Pairs Shortest Path (APSP) problem and it's variants.
   j. Network Flow problem (all it's algorithms, variants and the problems reducable to it).
9. Mathematics:
   a. You should be familiar with the BigInteger class in Java (maybe write your own if you are in love with C++).

d. Probability Theory.
e. Floyd-Cycle detection algorithm.
f. Game Theory (especially impartial games and Sprague-Grundy Theorem).

10. Strings:
    a. Basic Manipulation.
    b. Z-Algorithm for finding a pattern in a text.
    c. Knuth-Morris-Pratt Algorithm for finding a pattern in a text.
    d. Hashing and Rabin-Karp Algorithm for finding a pattern in a text.
    e. Trie data structure.
    f. Aho-Corasick Algorithm for finding multiple patterns in a text.
    g. Suffix Array data structure.
    h. Suffix Automaton data structure.
11. Computational Geometry Algorithms.

Those are most of the algorithms I have studied and used for competitive programming.

Anton Urusov, Yellow on Codeforces
Answered Jan 13, 2017

In my opinion, it'a not a very good idea to learn algorithms **before** starting competitive programming. The best way to become a good competitive programmer is to solve a lot of problems. Period.

Of course, it's not very clever to solve just any problems (you won't get any better if you solve A+B a thousand times). One approach I've read somewhere is the following:

1. Choose some online judge (personally, I like codeforces.com)
2. Go to the archive
3. Sort all the problems by difficulty
4. Start solving.
5. If you feel the problems are too easy (you solve most right away), skip a couple of pages. If you feel they are too difficult, go a couple of pages back. Repeat until you find the right level of problems. Optimally, you would be able to solve about half of the problems you try.

the editorial (codeforces is good here too, because most problems have editorials). The solution might include some algorithms you don't know yet. This is exactly where you should learn them.

7. After you have a solution (either you came up with your own or read the editorial), it's very important to implement it and get it accepted.
8. After that, you can also look at other peoples' solutions to see how it could be implemented shorter/faster/more efficient/...

Good luck!

8k views · View Upvoters · View Sharers

Upvote · 53 Share · 1

Add a comment... Recommended All

Related Questions

What is the best strategy to improve my skills in competitive programming in C++ in 2-3 months?

What should I learn first: data structures or algorithms?

How do I start learning or strengthen my knowledge of data structures and algorithms?

Ashish Mishra, Proud INDIAN || Traveller ||Full Stack Developer
Answered Aug 7, 2017

You should learn

1.Linked List

Linked list data structure provides better memory management than arrays. Because linked list is allocated memory at run time, so, there is no waste of memory. Performance wise linked list is slower than array because there is no direct access to linked list elements.

Linked list is proved to be a useful data structure when the number of elements to be stored is not known ahead of time.

There are many flavors of linked list you will see: linear, circular, doubly, and doubly circular.

2.Stack

Stack is a last-in-first-out strategy data structure; this means that the e...(more)

Upvote · 6 Share

Add a comment... Recommended All

Maly Petu, Master Information Technology, University of Technology, Sydney
Answered Aug 29, 2016

Ok the simplest way to get into software engineer is to learn the following:

- Machines: Learn and understand Hardware and Software in deep, what it is and how it works,
- Learn structured programming, don't need to use a fancy,complex language, Pascal will do well for instance,

- RDBMS read about different types, you'll find many but only Relational and new models will give you jobs for sure,
- Jump to Object Oriented Programming and learn its concepts,
- Now, tackle any Programming Language ...

(more)

Ashish Gupta, Ashishgup on CodeForces and other sites
Updated Aug 28, 2018 · Upvoted by Bhavik Dhandhalya, Hacker, Problem Setter on SPOJ & HackerEarth, ACM ICPC 2016 Participant

What algorithms/data structures do you need **before** you start competitive programming? **None.**

You learn things while practicing. That's how I and almost everyone I know has done it. The only pre-requisite is knowing how to code in any language - preferably C++ or Java and you're good to go.

The most basic algorithms you can and should learn once you start competitive programming (and you'll see the need for them very soon - when you are not able to solve a question related to them):

- **Number Theory** (Part 1 , Part 2) - This includes:
  - Modular Arithmetic Basics
  - Modular Exponentiation (computing $x^n % m$ in $\log n$
  - GCD in $\log n$
  - Modular Multiplicative Inverse,
  - Extended Euclidean Algorithm
  - Prime Factorisation - $\sqrt n$ and Sieve of Eratosthenes
- **Greedy:**
  - Tutorial . Developing intuition for greedy algorithms/proving or knowing their correctness for a question comes with practice
- **Binary Search:**
  - Tutorial . Some of my favorite questions are on Binary Search! It's a nice topic, and questions based on it can be pretty good if the monotonicity of the function isn't obvious.
- **Graph Theory:**
  - Graph Representation
  - BFS , DFS
  - Shortest Path Algorithms (Dijkstra, Floyd-Warshall, Bellman-Ford)
  - Minimum Spanning Tree (Prim, Kruskal)
  - Topological Sort
  - Strongly Connected Components
  - Articulation Points and Bridges
- **Data Structures:**

- Sparse Table
- Segment Tree

- **Dynamic Programming:**
  - Writing Top-Down DP (which will suffice 95% of the times)
  - Writing Bottom-Up DP (if time limit/memory limit is too strict)
- **Bit Manipulation:**
  - Bitwise Operators and their manipulation
  - Bitmasks (and their use in Dynamic Programming)
- **Strings:**
  - KMP Algorithm
  - Z Algorithm

You can find my implementation for some of these algorithms here

Note: I have linked the topics to where I learned them from and the resources I find the best for the specific topic. You can learn it from elsewhere if you find the linked resources lacking. If you want to learn more advanced algorithms, you can refer to my answer here.

These are the main topics that you will encounter in competitive-programming. If you master them, you should be able to solve over 95% of all questions. More advanced algorithms such as FFT, Flows, etc are very uncommon

It's been 2 years since I've been doing competitive programming and 99% of the questions I practice today are questions from these topics :)

I'll stress once again, learn these algorithms **while** doing competitive programming, **not before** doing it, because learning something without implementing it is much less useful, and frankly, a little boring.

28.9k views · View Upvoters · View Sharers

Upvote · 740 Share · 15

Rituraj Joshi
Can you refer some good resources for learning advanced topics like FFT, convex hull opti...

5 more comments from Ansuman Mohanty, Harshini Lakshmi, and more

Emily Condit, A.A. Degree in Computer Programming
Answered Aug 29, 2016

Algorithms are solutions to problems. There are lots of ways to solve all kinds of problems. When you talk about them in the context of computer science, you are speaking of the way we solve a problem programmatically. These solutions use basic logic.

If A, then do B.

If not A, then do C.

This is extremely simple logic and it's not written in a programming language (though many languages look similar to this) but an algorithm could be this simple. This is A way to solve a problem that a computer understands.

Algorithms get much more complex than this but at their core, they are assembled using ...(more)

Add a comment... Recommended All

Kanhaiya Kumar Singh, Software Development Engineer at Factset Systems India (2018-present)
Answered Aug 4, 2017

basic data structure and algorithms that one should know. Its not like that, you would be knowing most of data structures and algorithm but until u start coding them they are of none use because even if you know them.
So the question should be how to start competitive programming rather than knowing data structure and algorithm.
So lets start:-
The way we learnt math , physics same way we would go for it also.
Math and physics are what we have learnt from childhood but computer programming most people start when they go to engineering . So I will break the way of learning in 4 years.
It goe...(more)

Upvote · 42 Share

Add a comment... Recommended All

Suruchi Jain, A Competitive Programmer
Answered Dec 1, 2018

Data structures are an important part of software development and one of the most common topics for questions at developer interviews.

The good news is that they are basically just specialised formats for organising and storing data. In this article, you will learn about the 10 most common data structures.

“Bad programmers are worried about the code. Good programmers worry about data structures and their relationships. ” - Linus Torvalds, creator of Linux

Note that some of these data structures include temporal complexity in the Big O notation. This does not apply to all of them, since tempora...(more)

Upvote · 231 Share · 10

Add a comment... Recommended All

Rubén Berenguel, studied at University of Barcelona
Answered Sep 21, 2014

Originally Answered: What all basic data structures and algorithms should one learn before starting competitive programming?

I will ask you a question, though: why do you want to enter competitive programming? Of course, some people love the challenge and getting to the top (I know a few of these, and they are brilliant indeed.) But when programming for the real world, usually you don't need anything specially out-of-the-box for 90% of the cases. So, great, CP will train you for the other 10%, but usually this 10% you can pick on your own if you are smart enough, no need to keep banging your head against some relatively unnatural problem for months at a time.

As for the GSOC, I know at least that for the Plan9 p...(more)

Upvote · 3 Share

Add a comment... Recommended All

Answered Sep 29, 2018

Hello. I think you at least need to have knowledge of basic data structures like arrays, strings, linked list, stacks, queues and trees. By knowledge, I mean you should know how to implement them or some of it's applications. Secondly, how to implement basic sorting and searching algorithms like merge sort, quick sort, selection sort, bubble sort and binary search. If basics are clear then you can start with practicing problems.

I know this because I took training for both Java with data structures and competitive programming from Coding Ninjas. And, it was a very good experience. In the fir... (more)

Upvote · 6 Share

Add a comment... Recommended All

Alok Tripathy, studied at Georgia Institute of Technology

Answered Apr 28, 2013

Originally Answered: What are the things/algorithms I need to know to be successful in programming competitions?

You're in luck. There are plenty of resources.

I would suggest starting out with the USACO Training Gateway (Page on Delos). It's a compilation of hundreds of problems and tutorials that are meant to help people train for USACO. But, one could use it to train for essentially any programming competition. It might start out as pretty basic for you, but as soon as you get into the later part of Chapter 1 and into Chapter 2, you will cover some graph theory concepts as well as Dynamic Programming. It's probably one of the best resources out there to start training for competitions.

Another se...(more)

Upvote · 81 Share

Add a comment... Recommended All

Yash Sharma, I know a thing or two about programming.

Answered May 5, 2016

Thanks A2A.

But I really don't think that there's anything left for me to write. Answers here have noted down just about every algorithm you are gonna need and more. I specifically liked these.

1. Sambit Acharya's answer to What basic data structures and algorithms should one learn before starting competitive programming?
2. Duncan Smith's answer to What basic data structures and algorithms should one learn before starting competitive programming?
3. Vikesh Tiwari's answer to What basic data structures and algorithms should one learn before starting competitive programming?

These cover max of the algorit...(more)

Upvote · 3 Share

Add a comment... Recommended All

Updated Nov 22, 2014

Originally Answered: What are most important algorithms for competitive programming?

Most Basic Algorithms to read before starting Competitive Programming....

- Euler's function and its calculation [TeX]
- Binary exponentiation in O (Log N) [TeX]
- Euclid's algorithm of finding the GCD (greatest common divisor) [TeX]
- Sieve of Eratosthenes [TeX]
- Advanced Euclidean algorithm [TeX]
- Fibonacci numbers and their rapid calculation [TeX]
- Inverse element in the ring modulo [TeX]
- Gray code [TeX]
- Long arithmetic [TeX]
- Discrete logarithm modulo M algorithm baby-step-giant-step Shanks for O (sqrt (M) Log M) [TeX]
- Diophantine equations with two unknowns: AX + BY = C [TeX]
- Modular linear first-order equat...

(more)

Upvote · 420 Share · 5

Prakash Srivastava
Thank you for this answer. I cannot upvote it 500 times so I am giving you 5000 credits.

1 more comment from Pratik Mehta

Shivanshu Chauhan, Strategist at Goldman Sachs (2018-present)
Answered Feb 10, 2016

As you are second year student of Computer Science, you must be having Data Structures and Algorithms course. Do that course faithfully and try solving some exercise problems in Cormen.

For your kick start in Competitive programming, you should understand following topic in sequence:

1. **Sorting and searching algorithms:** Try to implement searching and sorting algorithms like, Binary search, insertion sort, merge sort, quick sort etc. Try to twist these algorithm till you get feel of how algorithm is working.
2. **Linked List, Stack, Queue:** These standard data structures will help you to solve most of...

(more)

Upvote · 90 Share · 2

Add a comment... Recommended All

Anonymous
Updated Dec 23, 2016

Originally Answered: What are the data structures and algorithms used in competitive programming?

Data Structures In Competitive Coding

1. Array

4. Binary Tree
5. Binary search tree
6. Red Black tree (Set in c++)
7. Optimal binary search tree (Not Recommended)
8. Binary Indexed tree
9. Segment tree
10. Persistence segment tree
11. Partial sum
12. Hash Table (MAP)
13. Trie (Keyword tree)
14. Suffix Array
15. Suffix Tree
16. Linked list (Interview purpose)
17. Graph
18. Sparse Table (RMQ)
19. Disjoint Data Structure...

(more)

Upvote · 6 Share

Add a comment... Recommended All

Ashish Chopra, passionate about teaching computers to solve problems by machine learning.
Answered Feb 24, 2016

Originally Answered: What is the best way to learn data structures and algorithms for competitive programming?

I always had sheer interest in learning data structures and algorithms, not because i need to prepare for some interviews - that is just a side effect of it - I always like to see how a human can solve any wild problem at hand, that too with Computers.

I do not want to share a comprehensive list of "ideal" resources which are classics in CS, but would like to share what i did to fulfill my desire of learning data structures and algorithms. I tried very hard on internet to find good lecture and books on the same topics. And then picked those which appealed to my understanding:

1. The most importa...

(more)

Upvote · 19 Share

Add a comment... Recommended All

Shreeprakash Agrahari, Algorithm
Answered Jul 20, 2016

definitely because ds and algo develop your logic .

njjnjnsomewhere within a method.

answer to What are the best websites a programmer should visit?

Tutorials: Log into Facebook | Facebook

Topicwise Questions: Log into Facebook | Facebook

Data Structures book(Must have): Data Structures and Algorithms Made Easy: Data Structure and Algorithmic Puzzles (English) - Buy Data Structures and Algorithms Made Easy: Data Structure and Algorithmi... (more)

Upvote · 4 Share

Add a comment... Recommended All

Mansi Gupta, The purpose has always been clear, just figuring out the right way.
Answered May 14, 2016

Originally Answered: What is the best way to learn data structures and algorithms for competitive programming?

Hey,

best way and also the most fun way, is coding on online judges like SPOJ,Codechef,Codeforces,TopCoder.

You would't even realize and with time you would start understanding algorithms and data structures so well. You don't even need a guide as online coding acted like a virtual guide.

From tutorials and online discussion forums and blogs from fellow competitors, you get to learn so much every time you find yourself stuck.

The best thing about it is, it makes you smarter. And it is super fun.Just have patience.

All the best!:)

805 views · View Upvoters

Upvote · 1 Share

Add a comment... Recommended All

Jie Feng, JIe Feng, Student@Johns Hopkins
Answered Nov 18, 2013

Originally Answered: What are some basic algorithms for competitve programming?

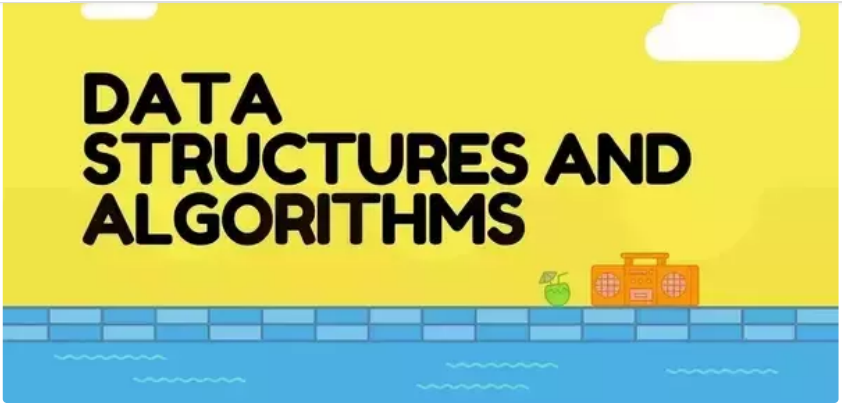
First of all, a solid understanding of searching algorithm(BFS, DFS, Searching tree pruning: A star, apha-beta) builds your first step in competitive programming. Then, graph theory, advanced data structures(Segment Tree, Trie, AVL balance tree, prefix tree, Union-Find-Set). Last but not least, network flow, number theory, computation geometry, approximation algorithm and greedy algorithm(this category of problems can become very hard, not as easy as many people imagined)

5.5k views · View Upvoters

Upvote · 7 Share

Jie Feng
I'm sorry I omitted dynamic programming, which is very very important(lies between sear...

Meitri Goutham, M.S Computer Science, University of Washington
Updated Sep 18, 2018



The first thing you'll need if you want to get better at algorithms and data structures is a solid base. This base can be learned one of several ways, either through a computer science program or from books. But my suggestion is to learn from online courses. This is the Best way. I can also suggest you the **Best Algorithms And DataStructures Online Courses:**

- Algorithms & Data Structures DP,NP Complete,Hard,Coloring
- Learning Data Structures and Algorithms
- Easy to Advanced Data Structures

Choose the first course..

From this course you may learn about:

This comprehensive course will give you enough un...(more)

Upvote · 14 Share · 1

Add a comment... Recommended All

Sai Krishna, javaScript Developer
Answered Aug 29, 2016

Sorting is the most heavily studied concept in Computer Science. Idea is to arrange the items of a list in a specific order. Though every major programming language has built-in sorting libraries, it comes in handy if you know how they work. Depending upon requirement you may want to use any of these.

- Merge Sort
- Quick Sort
- Bucket Sort
- Heap Sort
- Counting Sort

More importantly one should know to use them. Some examples where you can find direct application of sorting techniques include:

- Sorting by price, popularity etc in e-commerce websites
- leader board positions in any contests

Search Algorithms:

- **Bin...**

(more)

Add a comment... Recommended All

Siddharth Kamaria, Applications of Algos and Data Structures fascinate me!
Updated Jun 3, 2015

Originally Answered: What is the best way to learn data structures and algorithms for competitive programming?

To encompass my answer in short. I would like to enumerate a few points.

1.) Introduction to Algorithms by Cormen, Lieserson, Stein and Rivest is one of the internationally acclaimed books. It is the Bible of Algorithms. It has some thought provoking exercises alongside a good explanation of each and every aspect. It is a must read book!

2.) @http://www.geeksforgeeks.org/ has some awesome problems to scratch you brains. Do practice it!

3.) www.stackoverflow.com has a great community which would help you achieve great levels. Those programmers are beyond our imagination.

4.) Practice tho...(more)

Upvote · 13 Share

Add a comment... Recommended All

Akriti Mehrotra, Community Manager to a community of 2.5M+ developers
Answered Jul 11, 2019

You can't learn until you practice. There is no better way to practice than to solve problems and participate in competitive coding challenges.

To solve problems in Data Structure, you can visit HackerEarth. Their content is detailed and you can find tutorials along with questions of different difficulty level. The following are the topics covered in their Data Structures section:

- Arrays
- Stacks
- Queues
- Hash Tables
- Linked List
- Trees
- Advanced Data Structures
- Disjoint Data Structures

Similarly, you can solve different kind of Algorithms problem. The HackerEarth's tutorials are fairly detailed which will ...(more)

Upvote · 9 Share · 1

Add a comment... Recommended All

S Kiran, BE Computer Science, College of Engineering Guindy, Tamil Nadu, India (2015)
Answered Jul 20, 2019

Upvote · 1 Share

Ashutosh Singh, Subject Expert (Computer Science & Engg) at Chegg (2019-present)
Answered Aug 23, 2018

Following:

1. Arrays
2. Linked lists (single, double , circular)
3. Stacks (implementations using Arrays & Linked lists)
4. Queue & Dequeue (using Arrays & Linked lists, multiple queues, circular queues)
5. Trees (Binary Trees, AVL trees, B-trees)
6. Graphs (Various graph algorithms)
7. Hash tables
8. Trie

and their associated algorithms.

279 views · View Upvoters · Answer requested by Abhishek

Upvote · 2 Share

Add a comment... Recommended All

Hiresh Trivedi, AIR 961 in Codevita
Answered Nov 25, 2015

Originally Answered: What are the algorithms I should learn for competitive programming?

Its not necessary to learn all of them or a particular list of algorithms, but you should be able to design your own algorithms on the basis of other algorithms, because in competitive programming what matters is your approach towards the problem.

Thus, I recommend getting a thorough knowledge and analytical ability in applying and developing algorithms.

You could refer standard books on this and Youtube lectures from Stanford university could help too.

Thanks for A2A.

150 views · View Upvoters · Answer requested by Suyog Jain

Upvote · 1 Share

Add a comment... Recommended All

Prashant Kumar Katariya, Black Pearl is mine...
Answered Sep 20, 2014

Originally Answered: What all basic data structures and algorithms one should learn before starting competitive programming?

To enter in order to make yourself in good Competitive Programming, mostly you should aware of at least basic of each data structure and their auxiliary operation (means creating, searching, deleting, updating, adding etc).

So you need to focus on :

(3). Queue
(4). Tree
(5). Graph

go in this way only, because this flow is independent, and you need to understand previous one to jump further. Hope it will help you...

4k views · View Upvoters

Upvote · 2 Share

Add a comment... Recommended All

Hari Krishnan, person
Answered Nov 22, 2015

Originally Answered: What are the algorithms I should learn for competitive programming?

hey!, i found these answers helpful.
have a look

1. How should I practice so that I will be at a level where I can approach TopCoder's Div1-500 problems with confidence? I am a rookie with zero competitive programming experience.

2. What was Anudeep Nekkanti's Competitive Programming strategy to become 35th in Global ranking, in just 6-7 months?

3. What is the best strategy to improve my skills in competitive programming in C++ in 2-3 months?

have fun.enjoy coding ;)

1k views · View Upvoters

Upvote · 1 Share

Add a comment... Recommended All

Ratnesh Tiwari, 3+ years with programming and love is getting stronger day by day.
Answered Jan 30, 2016

Originally Answered: What all basic data structures and algorithms should one learn before starting competitive programming?

www.geeksforgeeks.org maintains a large collection of programming questions.
codemonks of hackerearth.com will help you in basic and intermediate level of datastructure learning.
On www.hackerrank.com you will get different domains to practice on.

I personally like hackerrank platform for practice.

Programming language is never a constraint, its all about solving big problems start from easy one.

Happy coding :)

1.2k views · View Upvoters

Upvote · 3 Share

Add a comment... Recommended All

Answered Dec 25, 2015

Originally Answered: What all basic data structures and algorithms should one learn before starting competitive programming?

You will find detailed content on the algorithms that you should learn to get started with Competitive coding.

Learn to Code by Competitive Programming - HackerEarth Blog

The complete reference to Competitive Programming

Also, you should start taking up the Code Monk Series, it contains detailed tutorials on all the topics you would need to learn, along with practice problems to solve.

10.3k views · View Upvoters · Answer requested by Suyog Jain

Upvote · 50 Share

Add a comment... Recommended All

Duncan Smith, Programmer by day (and night)
Answered Jun 15, 2015

Originally Answered: which data structure and algorithms for competetive programming?

Rather than worrying about learning which data structures and algorithms you need, I would recommend just solving problems. When you need an algorithms or data structure to solve a problem, look it up. That way you'll get experience solving problems, and you'll also learn the theory that you need. If you have trouble figuring out which algorithm/data structure to use for which problem, try using a categorized list like the one at uHunt :: UVa Hunting .

16.1k views · View Upvoters

Upvote · 44 Share

Add a comment... Recommended All

Related Questions

What is the best strategy to improve my skills in competitive programming in C++ in 2-3 months?

What should I learn first: data structures or algorithms?

How do I start learning or strengthen my knowledge of data structures and algorithms?

What are some of the best ways to learn programming?

Can anyone give me the list of basics to advance one by one algorithm and data structure learning topics for competitive programming?

What are the best books on algorithms and data structures?

Do I need to learn data structures before starting to learn programming?

What road map do you suggest for a beginner in programming to learn algorithms and data structures to usable level for competitive programming...

What are some good websites to learn data structures and algorithms?

Is it possible to learn Data Structures and Algorithms in a week?

How do I learn data structures and algorithms for interviews of big companies in just 2 months?

What are the requirements before I start learning algorithm and data-structures?

How should you start learning programming?

Can I learn data structures and algorithms while doing competitive programming?

Top Stories from Your Feed

Answer written · Computer Programming · Topic you might like

How can I learn competitive coding?

Raymond Deng, B.A. Computer Engineering, University of California, Santa Barbara (2021)

Answered Jul 8, 2016 · Upvoted by Abhishek Pratap, ECE undergrad ,IIT-R

Read In Feed

Answer written

How can I prepare for interviews in any big software company?

Fahim ul Haq, former Software Engineer at Facebook

Updated Dec 10

(NOTE: I've interviewed hundreds of candidates for software engineering jobs at Facebook and Microsoft.) A note on timeframe: Three months is a reasonable for preparing. 4–6 weeks is the minimum if ... (more)

100+ 20 1

Read In Feed

Answer written

If programming languages had honest titles, what would they be?

Fred Mitchell, I know Ruby, Rust, Python, Haskell, C++, Erlang, and more.

Updated Thu

1. C++ — A Force of Nature
2. Ruby —The Slow Scripting Language
3. Haskell — Academic Hardon
4. Python — 21st Century Basic
5. Erlang — The Dying Language
6. Elixir — It ain't Ruby!...

Read In Feed