Theoretical Computer Science · Competitive Programming · Computer Programming

What is the mathematical background necessary for competitive programming?

9 Answers

Aditya Varma, Undergraduate in computer science.
Written 27 May 2015

Hope you find this useful!

1. **Math (Probability, Counting, Game Theory, Group Theory, Generating functions, Permutation Cycles, Linear Algebra)**
   a. **Probability.**

Syllabus

- Basic probability and Conditional probability

1. Suggested problems

1. SPOJ.com - Problem CT16E
2. SPOJ.com - Problem CHICAGO

- Random variables, probability generating functions
- Mathematical expectation + Linearity of expectation
  a. Suggested problems
     i. SPOJ.com - Problem FAVDICE
     ii. Problem Statement

1. Special discrete and continuous probability distributions
   a. Bernoulli, Binomial, Poisson, normal distribution
   b. Suggested Problem
      i. SSU Online Contester :: 498. Coins
2. Suggested Readings
   a. Cormen appendix C (very basic)
   b. Topcoder probabilty tutorial *http://www.topcoder.com/tc?module=Static&d1=tutorials&d2=probabilities*
   c. Random variable
   d. Expected value
   e. William Feller, An introduction to probability theory and its applications

1. **Counting**

Syllabus

- Basic principles - Pigeon hole principle, addition, multiplication rules

1. Suggested problems
   a. http://acm.timus.ru/problem.aspx...
   b. *Problem Statement*

1. Suggested readings
   a. Combinatorial principles
   b. Page on topcoder.com
   c. Page on maa.org

- Inclusion-exclusion

1. Suggested readings
   a. Inclusion–exclusion principle
2. Suggested problems
   a. Problem Statement
   b. Problem Statement

Related Questions

- For a software engineer who doesn't find competitive programming interesting but would like to join one of the big tech companies (for example...
- What are some must-know topics in discrete math and probability for competitive programming?
- Is it possible to do well in competitive programming if you don't have a strong mathematical background?
- What is the mathematical background necessary to understand a typical book on algorithms?
- Is it possible to be a good programmer without competitive programming? How is it possible?
- Is it possible to develop some math skills (or rather mathematical approach/thinking) by solving programming problems and practicing for compe...
- Is it ever too late for someone to learn how to program?
- What's the most fun kind of math?
- How do I get strong mathematical background?
- Is it necessary to have powerful mathematics for learning to programming?

Special numbers

1. Suggested reading - Stirling, eurlerian, harmonic, bernoulli, fibonnacci numbers
    a. Stirling number
    b. Eulerian number
    c. Harmonic series (mathematics)
    d. Bernoulli number
    e. Fibonacci number
    f. Concrete mathematics by Knuth
2. Suggested problems
    a. Problem Statement
    b. Problem Statement
    c. Problem Statement
    d. Problem Statement

1. Advanced counting techniques - Polya counting, burnsides lemma
    a. Suggested reading
        i. Burnside's lemma
        ii. Algorithms Weekly by Petr Mitrichev
    b. Suggested Problems
        i. Problem Statement
        ii. SPOJ.com - Problem TRANSP

c. Game theory
Syllabus

- Basic principles and Nim game

1. Sprague grundy theorem, grundy numbers
2. Suggested readings
    a. Sprague–Grundy theorem
    b. Algorithm Games - topcoder
    c. Feature Column from the AMS
    d. Page on codechef.com
3. Suggested problems
    a. Problem Statement
    b. Problem Statement

Hackenbush

1. Suggested readings
    a. Hackenbush
    b. Feature Column from the AMS
2. Suggested problems
    a. Computing + Mathematical Sciences
    b. SPOJ.com - Problem PT07A

d. Linear Algebra
Syllabus

- Matrix Operations

1. Addition and subtraction of matrices
    a. Suggested Reading
        i. Cormen 28.1
2. Multiplication (Strassen's algorithm), logarithmic exponentiation
    a. Suggested reading
        i. Cormen 28.2
        ii. Linear Algebra by Kenneth Hoffman Section 1.6
    b. Problems

        i. Contest of Newbies 2006

3. Matrix transformations [Transpose, Rotation of Matrix, Representing Linear transformations using matrix]
    a. Suggested Reading
        i. Linear Algebra By Kenneth Hoffman Section 3.1,3.2,3.4,3.7
    b. Problems
        i. Problem Statement
        ii. JPIX on Spoj
4. Determinant , Rank and Inverse of Matrix [Gaussean Elimination , Gauss Jordan Elimination]
    a. Suggested Reading
        i. 28.4 Cormen
        ii. Linear Algebra by Kenneth Chapter 1
    b. Problems
        i. Problem Statement
        ii. Problem Statement
        iii. Problem Statement
        iv. HIGH on Spoj
5. Solving system of linear equations
    a. Suggested Reading
        i. 28.3 Cormen
        ii. Linear Algebra by Kenneth Chapter 1
    b. Problems -
        i. Problem Statement
6. Using matrix exponentiation to solve recurrences
    a. Suggested Reading
        i. TopCoder Feature Articles
    b. Problems
        i. REC, RABBIT1 , PLHOP on spoj
        ii. Problem Statement , Problem Statement, Problem Statement
7. Eigen values and Eigen vectors
    a. Problems
        i. Problem Statement

Polynomials
1. Roots of a polynomial [Prime factorization of a polynomial, Integer roots of a polynomial, All real roots of a polynomial]
    a. Problems
        i. Problem Statement
        ii. POLYEQ , ROOTCIPH on Spoj
2. Lagrange Interpolation
    a. Problems
        i. Problem Statement
        ii. Problem Statement

e. Permutation cycles
- Suggested Reading

1. Art of Computer Programming by Knuth Vol. 3

Problems
1. ShuffleMethod, Permutation and WordGame on topcoder.

f. Group Theory
    - Bernside Lemma, Polias theorem

1. Suggested Reading
    a. Hernstein's topics in algebra
    b. Algorithms Weekly by Petr Mitrichev
2. Problems
    a. TRANSP on spoj
    b. Problem Statement

Generating functions

- Suggested Reading

1. Herbert Wilf's generating functionology
2. Robert Sedgewick and Flajoulet's Combinatorial analysis

> Source: Programming Camp Syllabus

Related Questions More Answers Below

For a software engineer who doesn't find competitive programming interesting but would like to join one of the big tech companies (for example...

What are some must-know topics in discrete math and probability for competitive programming?

Is it possible to do well in competitive programming if you don't have a strong mathematical background?

What is the mathematical background necessary to understand a typical book on algorithms?

Is it possible to be a good programmer without competitive programming? How is it possible?

Istasis Mishra, Have learnt Java, C, C++ and Python ;)
Written 26 Oct 2016

If you want to be a very serious competitive programmer, I will go a bit deeper into the details of what all math you might ever need for competitive programming.

- **Arithmetic and Geometry:-**
    - Integers, operations(including exponentiation), comparison
    - Basic property of integers(sign, parity, divisibility)
    - Basic modular arithmetic(addition, subtraction, multiplication, division)
    - Prime numbers(Its a vast topic)
    - Fractions and percentages
    - Line, line segment, angle, triangle, rectangle, square, circle(and specially way to represent them in your code)
    - Point, vector, coordinates in a plane
    - Polygon(vertex, side/edge, simple, convex, inside, area)
    - Euclidean distances
    - Pythogorean theorem(There are pretty diverse use of this theorem)
    - In some questions you might need Fermat’s Little Theorem, Chinese remainder theorem, or may need to use the Euler function etc. You need to get better in number theory. I have encountered some problems on codechef which need these myself.
    - Knowing some techniques like Euclidean Algorithm and fast exponentiation are considered to be a prerequisite in some contests.
    - Some computational geometry techniques like Graham Scan and convex hull.
- **Discrete Structures:-**
    - Functions, Relations and Sets
        - Functions(surjections, injections, inverses, composition)
        - Relations(reflexivity, symmetry, transitivity, equivalence relations, total/linear order relations, lexicographic order)

        - Sets(cardinality, inclusion/exclusion, complements, Cartesian products, power sets)
    - Basic Logic
        - First-order logic
        - Logical connectives(including their basic properties)
        - Truth Tables
        - Universal and Existential quantification
        - Modus ponens and Modus tollens(Not to get intimidated by the names :P)
    - Proof techniques(You don’t need them too rigorously)
    - Basics of Counting
        - Counting arguments (sum and product rule, arithmetic and geometric progressions, Fibonacci numbers)
        - Permutations and combinations (basic definitions)
        - Factorial function, binomial coefficients
        - Inclusion-exclusion principle
        - Pigeonhole principle
        - Pascal’s identity, Binomial theorem
        - Solving recurrence relations
        - Burnside Lemma
    - Graphs and trees
        - Trees and their basic properties, rooted trees
        - Undirected graphs (degree, path, cycle, connectedness, Euler/Hamilton path/cycle, handshaking lemma)
        - Directed graphs (in-degree, out-degree, directed path/cycle, Euler/Hamilton path/cycle)
        - Spanning trees
        - Traversal strategies
        - ‘Decorated’ graphs with edge/node labels, weights, colors
        - Multigraphs, graphs with self-loops
        - Bipartite graphs
        - Planar graphs
- **Probability theory(This is not in the scope of syllabus of the IOI so I won’t be going into the details. I would leave a resource here for your reference tho, ’cause I am an angel 0:-))**
    - Topcoder Tutorial: Understanding Probabilities
- **Linear Algebra(including but not limited to)**
    - Matrix multiplication, exponentiation, inversion, and Gaussian elimination
    - Fast Fourier transform

Even though, many of these topics are not to be mastered rigorously, some are prerequisites and competitive programmers should have the basic knowledge of at least most of them to be among the top tier of programmers.

I hope this helped :D

Happy Coding!

Source: IOI syllabus 2015

2.1k Views · View Upvotes · Answer requested by Aulene De

Arun Prasad, Sport programmer for over a year
Updated 26 Oct 2016

Discrete Mathematics Mainly

1.Discrete Maths and Logics
2. Number Theory
3.Graph Theory
4.Mathematical Structures

5.Probablity Theory

First Do this
Mathematics For Computer Science By MIT OpenCourse Ware

Mathematics for Computer Science

This will cover all maths needed for programming like Graph Theory, Number system etc.

Then start learning algorithms

Good Luck :)

1.6k Views · View Upvotes

Vivek Raj, I am extremely interested in TOC.
Written 6 Nov 2016

Just fundamentals things nothing else. But it will depends upon what level of competitive programming are you talking about. But in any case if you are strong with you programming skills then you will easily surpass the mathematics barrier.Try to solve programming challenge on various websites like codechef to enhance your skills, you will find your answer. My approach is always do problems then theory….

285 Views · View Upvotes · Answer requested by Anuj Jain

Aulene De, All hail DP
Written 26 Oct 2016

Most answers have already outlined whatever you needed. I'll drop this website here, About - Project Euler

I'm pretty weak when it comes to mathematical problem-solving (i.e problems which are entirely mathematical and not as algorithmic in nature). Solving Project Euler problems gives you the best insight as to what type of problems can be formed based on math, in my opinion. I've been trying to do 1–2 questions a day since a little less than a week ago, and can say that I've improved since then. You learn something after every problem.

There's also a contest on HackerRank for the same.

603 Views · View Upvotes

Related Questions

Is it possible to develop some math skills (or rather mathematical approach/thinking) by solving programming problems and practicing for compe...

Is it ever too late for someone to learn how to program?

What's the most fun kind of math?

How do I get strong mathematical background?

Is it necessary to have powerful mathematics for learning to programming?

Why is the knowledge of mathematical proofs necessary in competitive programming?

How can I do good in competitive programming and algorithms? Does it require very deep knowledge in mathematics?

Why was Big-O notation invented?

Should I quit competitive programming?

What are the most important math courses for Competitive Programming and upgrading my brain ?

Is it necessary to learn competitive programming as a electrical engineer?

Is there a problem of Project Euler which is an NP problem?

What is the best way to learn the mathematics which are needed in competitive programming?

What sucks about competitive programming?

I am final year under grad and 21 years old. Is it too late to start competitive programming?

Top Stories