MATH IMPOTANCE IN CODING

It depends on your programming level and the way you are going to programming, however for the basic computation and programming concepts you must have knowledge about the discrete mathematics and algebra foundations like :

logic and arithmetics operations ,

matrices (mostly is used in graphical works) ,

Sets and the way it is used in computing,

logical AND and OR (for conditional statements),

Number bases (special in 2) and the way it differs from decimal base, because all the operations in computers execute in just two number 1 and 0.

Some of the topics which u should nicely brush up are:

1.permutations and combination

2.Probability

3.Basic maths

4.Reasoning

5.Functions

If u have went through these topics well,u will be comfortable with the programming.

Here is the basic math you need to get started with competitive programming :

- **Number Theory** - Sieve, primality test, fibonacci etc
- **Linear Algebra** - Basic Matrices, multiplication, etc
- **Basic Algebra** - Solving linear equations, finding unknown, etc

- **Statistics** - Mean, Median, Variance, expected value

  Basic Arithmetic concepts like mixtures, progressions, ratios, rational numbers, hcf / lcm etc.

- **Probability** - Bayes' theorem, Probability distribution, etc

- **Co-ordinate Geometry** - 2D, 3D space, distance between points, line equation, etc
- **Discrete Mathematics and Combinatorics** - Arrangements, sorting, permutations
- **Principles like Pigeon Hole, Inclusion-Exclusion, Induction etc**

That is just for starting. Once you are comfortable with competitive programming you can learn following advance topics :

- **Graph Theory** - Path, Node, Colouring, Matching, Components, Flow, etc

- **Calculus** - Variable Minimisation, Maximisation

- **Error Analysis and Estimation**. This is important for only few tricky probs

- **Game Theory** - Nim Game, Grundy Number, etc.

- **Fast Fourier Tranformation**

- **Modular Arithmetic**

Here are some mathematical topics commonlyencountered in programming competitions:

- Modular arithmetic, extended Euclid, Euler's totient function, Moebius function, Binomical coefficients, Bernoulli numbers, linear diophantine equations
- Matrices, Gauss algorithm
- Discrete probabilities, linearity of expectation, probability of conjunction of independent variables, distribution of order statistics.
- Combinatorics, rule of sum/product, inclusion/exclusion principle, Burnside lemma, Gray code, Catalan numbers
- Simple analitycal geometry (line intersection, line/circle intersection, inscribed circle), convex hull.
- Recurrence relations, linear recurrences
- Polynomials, fast Fourier transform
- Hall's marriage theorem, Menger's theorem (graph problems in general tend to be math-heavy)
- perfect number , Goldbach's conjecture ,weak goldbach conjecture

I might not be able to give you all of the topics but let me try to give you what I can.

1. **Proof techniques:** Extremely important topic of mathematics to learn algorithms. Whenever you are studying an algorithm from a legit source like CLRS or MITOCW, you would see a lot proofs to prove the correctness and efficiency of algorithms. Mostly, two proof techniques are used. Induction and contradiction. Proof is not a thing which can be just learnt, it's a skill that has to be mastered.
2. **Probability:** I haven't seen any advanced probability up till now in my journey of learning data structures and algorithms but basic probability is used mostly in the analysis of some algorithms. One most popular example would be hashing. You have to use some probability to understand how hashing works. Probability is used more extensively in the analysis of some randomized data structures and algorithms like skip lists and randomized quicksort or randomized median finding.
3. **Recurrence relations:** Also a useful tool in analyzing the complexity of recursive algorithms like merge sort. It's good that you know how to solve recurrence relations. In the courses on MITOCW, they only used substitution method, drawing the recursion tree method and the master's theorem. They taught all of it and I think most sources to learn data structures and algorithms should also teach it.
4. **Number theory:** Widely used in cryptography although I don't know if you would want to learn that or not. I can't think of any example where number theory is directly used in any algorithm right now but there are few must know topics in number theory which you must learn to solve problems using data structures and algorithms. Some of them are-
   a. Prime numbers and finding prime numbers using sieve's algorithm
   b. Greatest common divisor and Euclid's algorithm to find gcd
   c. A little modular arithmetic might also help

I've been practicing at Codechef for a while and now I'm gradually moving toward medium/hard problems. However many algorithms at these levels are very difficult to predict, and I was always stuck because I'm not aware of them. So I open this topic, my hope is to have a wish-list of most used algorithm for online programming contest that I can look up for reference. Here is my short-list up to now:

1. Segment tree (with lazy propagation)
2. Interval Tree
3. Binary Indexed Tree
4. Fast Modulo Multiplication (Exponential Squaring) 205
5. Heuristic Algorithms
6. KMP string searching

7. Manacher's Algorithm
8. Union Find/Disjoint Set
9. Trie
10. Prime Miller Rabin
11. Matrix Recurrence + Fast Modulo Multiplication for counting
12. Stable Marriage Problem
13. Extended Euclid's algorithm 48
14. Ternary Search
15. Fast Fourier Transform for fast polynomial multiplication
16. Djikstra's algorithm, Bellman-ford algorithm, Floyd-Warshall Algorithm
17. Prim's Algorithm, Kruskal's Algorithm
18. RMQ, LCA
19. Flow related algorithms, assignment problem, Hungarian algorithm
20. Bipartite matching algorithms
21. Heavy-light decomposition
22. Sweep line algorithm
23. Z algorithm
24. Convex Hull
25. Suffix Arrays 60
26. LCP
27. Suffix Tree
28. Gaussian Elimination
29. Numerical Integration/Differentiation
30. Line Clipping
31. Advanced Maths Ad-Hoc problems
32. Aho–Corasick string matching algorithm;
33. Calculate nCr % M Lucas's Theorem 55
34. Heavy Light decomposition in trees
35. Inverse Modulo operations
36. Pollard Rho Integer Factorization
37. Catalan Numbers
38. Euclid's GCD Algorithm
39. Extended Euclid's algorithm
40. Binary Search, Ternary Search
41. Sieve of Eratosthenes for finding primes
42. Fast Fourier Transformation for fast polynomial multiplication
43. Graph algorithms - BFS, DFS, finding connected components
44. Djikstra's algorithm, Bellman-ford algorithm, Floyd-Warshall Algorithm
45. Prim's Algorithm, Kruskal's Algorithm
46. RMQ, LCA
47. Flow related algorithms, assignment problem, Hungarian algorithm
48. Bipartite matching algorithms
49. Heavy-light decomposition
50. Sweep line algorithm
51. Z algorithm
52. Suffix Arrays;
53. LCP;
54. Heuristic Algorithms;
55. Gaussian Elimination;
56. Numerical Integration/Differentiation;
57. Line Clipping;
58. Advanced Maths Ad-Hoc problems;
59. Aho–Corasick string matching algorithm;

60. Knuth–Morris–Pratt algorithm;

TABLE OF CONTENTS