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++).  
   b. Some Combinatorics.  
   c. Number Theory (all what you can learn about it).  
   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.

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

* [Euler's function and its calculation [TeX]](http://e-maxx.ru/algo/euler_function)
* [Binary exponentiation in O (Log N) [TeX]](http://e-maxx.ru/algo/binary_pow)
* [Euclid's algorithm of finding the GCD (greatest common divisor) [TeX]](http://e-maxx.ru/algo/euclid_algorithm)
* [Sieve of Eratosthenes [TeX]](http://e-maxx.ru/algo/eratosthenes_sieve)
* [Advanced Euclidean algorithm [TeX]](http://e-maxx.ru/algo/extended_euclid_algorithm)
* [Fibonacci numbers and their rapid calculation [TeX]](http://e-maxx.ru/algo/fibonacci_numbers)
* [Inverse element in the ring modulo [TeX]](http://e-maxx.ru/algo/reverse_element)
* [Gray code [TeX]](http://e-maxx.ru/algo/gray_code)
* [Long arithmetic [TeX]](http://e-maxx.ru/algo/big_integer)
* [Discrete logarithm modulo M algorithm baby-step-giant-step Shanks for O (sqrt (M) Log M) [TeX]](http://e-maxx.ru/algo/discrete_log)
* [Diophantine equations with two unknowns: AX + BY = C [TeX]](http://e-maxx.ru/algo/diofant_2_equation)
* [Modular linear first-order equation: AX = B [TeX]](http://e-maxx.ru/algo/diofant_1_equation)
* [Chinese remainder theorem. Garner's algorithm [TeX]](http://e-maxx.ru/algo/chinese_theorem)
* [Finding factorial power divider [TeX]](http://e-maxx.ru/algo/factorial_divisors)
* [Ternary balanced system value [TeX]](http://e-maxx.ru/algo/balanced_ternary)
* [Factorial N! modulo P for O (N Log P) [TeX]](http://e-maxx.ru/algo/modular_factorial)
* [Through all this mask subpatterns. Grade 3 N for the total number of all subpatterns masks [TeX]](http://e-maxx.ru/algo/all_submasks)
* [Primitive root. Algorithm for finding [TeX]](http://e-maxx.ru/algo/primitive_root)
* [Discrete root extract [TeX]](http://e-maxx.ru/algo/discrete_root)
* [Sieve of Eratosthenes with linear running time [TeX]](http://e-maxx.ru/algo/prime_sieve_linear)