at **First,** start learning a programming language clearly like you can start with C or C++. Get comfortable writing code in either of one of these languages C, C++ or Java. Why only C, C++ or Java? Because these are the standard languages allowed in any programming competition.

If you are already good at C, it is suggested to learn C++. It is the most popular language among competitive programmers because of its speed and an excellent library in the form of STL (Standard Template Library).

Pick an online judge. Recommended ones are Topcoder and Codeforces. These sites have high quality of problems and also allow you to see other’s code post contest completion. These also categorize problems based on the topic. Some other popular judges include SPOJ, CodeChef (powered by SPOJ) andHackerEarth.

To begin with, start with simple problems that typically require transforming English to code and does not require any knowledge on algorithms.

At the early stages of programming one tends to write long pieces of code, which is actually not required. Try to keep codes short and simple.

Practice these problems until you become comfortable that you can submit it for 240 odd points on any day.

Start implementing basic(or standard) algorithms. It is suggested to read them from Topcoder tutorials or Introduction to algorithms.

* Graph algorithms: Breadth first search(BFS), Depth first search(DFS), Strongly connected components(SCC), Dijkstra, Floyd-Warshall, Minimum spanning tree(MST), Topological sort.
* Dynamic programming: Standard dynamic programming problems such as Rod Cutting, Knapsack, Matrix chain multiplication etc.
* Number theory: Modular arithmetic, Fermat’s theorem, Chinese remainder theorem(CRT), Euclidian method for GCD, Logarithmic Exponentiation, Sieve of Eratosthenes, Euler’s totient function.
* Greedy: Standard problems such as Activity selection.
* Search techniques: Binary search, Ternary search and Meet in the middle.
* Data structures (Basic): Stacks, Queues, Trees and Heaps.
* Data structures (Advanced): Trie, Segment trees, Fenwick tree or Binary indexed tree(BIT), Disjoint data structures.
* Strings: Knuth Morris Pratt(KMP), Z algorithm, Suffix arrays/Suffix trees. These are bit advanced algorithms.
* Computational geometry: Graham-Scan for convex hull, Line sweep.
* Game theory: Basic principles of Nim game, Grundy numbers, Sprague-Grundy theorem.

The list is not complete but these are the ones that you encounter very frequently in the contests. There are other algorithms but are required very rarely in the contests.

And to practice problems you can you can refer following sites 1. [http://codechef.com](http://codechef.com/) 2. [http://spoj.com](http://spoj.com/) 3. [http://hackerrank.com](http://hackerrank.com/) 4. [http://codecademy.com](http://codecademy.com/) 5. [http://topcoder.com](http://topcoder.com/) 6. [http://codeforces.com](http://codeforces.com/)

**To be noted :**as a problem solver you should not loss patience, never give up. It’s a long time process. Everything think easily and read others code never do copy from others but learn from others code.