Future Topics

Ruhan Habib

May 15, 2025

1 Useful Resources

1.1 Online Judges

- **CodeForces**
  It is the most popular website for competitive programmning. The website regularly holds individual programming contests: Div. 1, Div. 2, Div. 3, Div. 4, Education Round, and so on. Division 4 rounds are supposed to be the easiest, while the Division 1 rounds are supposed to be the hardest contests. I suggest going to the problemset page, turning off tag visibility, and start solving problems after sorting them in ascending order of difficulty. If you are stuck on a problem for too long, then you can read the editorial (aka solution) for the problem. You can also virtually participate in past contests. Regular participation in CodeForces' contests can give a wonderful boost to your programming and thinking capabilities. CodeForces is the best site for direct IUPC/ICPC training: its gym contains ICPC contests all over the world (not including Bangladesh, unfortunately). You can select an ICPC contest with your desired difficulty and then virtually participate in them. This is mostly how we train.

- **AtCoder**
  Another great competitive programming website. It has a much more regular schedule when compared to CodeForces: 6pm every week, Saturday or Sunday. There are mainly four types of contests: AtCoder Beginner Contest, AtCoder Regular Contest, AtCoder Grand Contest, and AtCoder Heuristic Contest. AtCoder Beginner Contests are great for beginners, while Regular Contests have something in it for most participants. Grand Contests are quite difficult, and are for advanced contestants. Heuristic Contests are not exactly competitive programming contests: they focus on problems that require heuristic solution. I recommend participating in the Beginner Contests. Of course, editorials are available for all of the problems.

- **CSES**
  This website contains a small but great collection of classic problems. I strongly recommend going through the collection (at least the introductory problems).

- **LightOJ**
  A great Bangladeshi online judge. While it does not seem to host regular contests, it has an organized collection of problems. It also has problems from ICPC Dhaka 2021, 2022, and 2023.

- **Toph**
  A lot of Bangladeshi IUPCs are held on this platform, and you can "virtually" participate in them.

- **CodeChef**
  It was a nice online judge once. Each month, there was a Lunchtime, Cook-Off, and a Long Challenge. Long Challenges were especially great. The editorials for the old contests should be available.

1.2 Books

- **Competitive Programmer's Handbook by Antti Laaksonen**
  As the name implies, this book contains much of what may be needed in a competitive programmer's arsenal. However, the breadth of this book necesssarily sacrifices some depth, so it might be better to use other resources along this book.

- **Programming Contest and Data Structure by Md. Mahbubul Hasan**
  This book should be of similar essence to the preceding book. However, it is written in Bengali and seems to include a lot of discussion on specific problems. It also has an exercise section, which should be great.

- **Introduction to Algorithms by Cormen, Leiserson, Rivest, and Stein**
  The classic book for algorithms. This book covers a lot of material (including stuff outside the scope of competitive programming as well). It may be harder to read than the previous two books and many other resources for competitive programming, but it is well worth it to go through the chapters on topics that you are already acquainted with. The book contains a lot of great exercises and rigorous proofs for the presented algorithms.

1.3 Other Online Resources

- **CodeForces Catalog**
  A lot of people have contributed tutorials covering various topics via blogs in CodeForces. There are blogs on most topics in competitive programming here.

- **Algorithms for Competitive Programming**
  This website contains a well-organized collection of tutorials on various topics in competitive programming. Most tutorials also contain links to practice problems as well.

- **USACO Guide**
  Should contain tutorials for a lot of the stuff needed for competitive programming. The tutorials also contain links to practice problems and other learning resources. You can start at the Bronze section and slowly move up.

- **Shafaets Planet**
  A good Bengali blog containing a lot of what is needed. Great blog.

- **The Real BCS**
  This page has a list of IUPCs dating back to 2017. This list also contains links to the problemsets and editorials. Should be very useful.

2 Topics

For each of these topics, check the resources mentioned in the previous sections.

- **Mathematics**
  The content in the Appendix (Mathematical Background) of the book by CLRS (Cormen, Rivest, Leiserson, and Stein) covers enough material for most programming contests. Essentially, you need a background in Discrete Mathematics: Σ-notation manipulation (that is, how to manipulate sums), sets, graphs, counting, probability, etc. Stars and Bars is super useful, while Inclusion-Exclusion and Pigeonhole Principle are also useful sometimes.

- **Greedy**
  Many CodeForces problems require greedy strategies. Solving a lot of CF problems (with the help of editorials if the problems feel intractable) should be enough to improve this. If you are interested in abstract stuff, you can look up matroids.

- **Constructive Algorithms**
  Many CodeForces problem also are of the constructive variant. The problems ask you to construct something (for example, construct a sudoku puzzle with no solution). The best way to get good at these is to simply solve a lot of them.

- **Implementation** Being able to implement code fast with little bugs is what we (programmers) all aspire to. The only way to be good at implementation is by coding a lot.

- **Dynamic Programming**
  This is a beautiful technique that will almost always appear in every IUPC and in a good chunk of CodeForces contests. This is a super useful technique and as such is absolutely necessary to go beyond specialist in CF. You should know the following: Knapsack DP, Bitmask DP, Digit DP, Longest Increasing Subsequence, etc. DP optimizations are quite important as well; any rated over 1900 should know them.

- **Graphs**
  Graph Theory is an incredibly beautiful branch of mathematics and computer science. Naturally, it is quite pervasive in competitive programming as well. Anyone who wants to be an expert in CF must at least know the basic properties of trees and graphs, as well as the basic algorithms. Basic algorithms include DFS, BFS, Cycle-Detection, Bellman-Ford, Djikstra, Floyd-Warshall, and Kruskal. If you find these interesting, then you can try learning about topics such as Articulation Points, Bridges, Strongly Connected Components, Topological Sorting, Bipartite Matching, and Max Flows as well.

- **Data Structures**
  Segment Trees are also a must (not as much as Dynamic Programming, but they are still important). They are quite common in IUPCs as well. Disjoint Set Union, Longest Common Ancestor, and Square Room Decomposition are also useful (though not as much as Segment Trees).

- **Strings**
  Strings are also prevalent in Competitive Programming, though specific string algorithms are usually not required (for specialists) in CodeForces. For IUPC, it is recommended to learn the following: string hashing, tries, and suffix arrays. Learning KMP is also recommended, even if it may not be directly useful in most contests.

- **Randomized Algorithms**
  Randomized Algorithms are often simple yet very beautiful. Sometimes, choosing randomly is enough to give us what we want. For example, randomly shuffling the input array turns deterministic quick sort from an $O(n^2)$-time algorithm to an $O(n \log n)$-time algorithm. It is well worth it to at least understand what randomized algorithms are.

- **Computational Geometry**
  Geometry is incredibly rare in CodeForces. It is even rarer to find them in problems A to D of a CF Division 2 contest. When they do appear, school geometry should be enough. They do, however, appear in IUPC/ICPCs. To be able to solve these, you should be able to handle 2D or 3D vectors and use them to compute stuff. So you should know about cross products, dot products, and how to use them to calculate point-line distances, angles between intersecting lines, etc. Sweep lines and convex hull are two important techniques useful in solving many geometry problems.