CAMP Summer Training

Week 1 - Warm Up

Hello and welcome to the first week of CAMP's Summer Training curriculum! This document will help you take your first steps in the world of competitive programming and gear up for the next season of ACM ICPC.

This document will contain weekly problem sets and tutorials that aim to bridge the gap between your DSA theory knowledge and your programming implementation skills.

We strongly recommend **ALL** students to at least attempt the problems from the first 3 weeks so as to brush up their problem solving and implementation skills to acceptable levels. A basic command over logic and implementation is a bare necessity in both CS and ECE branches. If you had trouble with the either the DSA or Programming courses, now is the time to cover up :)

The enthusiastic lot that finishes the entire curriculum will develop problem solving skills that are above and beyond those of an average first year Computer Science student in India, and will be in a solid position to prepare for the ICPC Regionals.

The entire document will contain 120 - 150 delicious challenges to satisfy your appetite. These are the targets you should aim for based on the kind of benefits you want to receive -

1. "I am not interested in competitive programming but I want to improve my programming skills to a bare minimum" :- Solve 30 - 40 problems
2. "I want to explore competitive programming along with other things as well" :- Solve 35 - 60 problems
3. "I want to learn enough to get good grades in subsequent programming and algorithm courses - C++, Java, Design and Analysis of Algorithms, Discrete Math, Probability" :- Solve 60 - 80 problems
4. "I want to give a shot at qualifying for the ACM ICPC Regionals" :- Solve 100+ problems
5. "You're a wizard Harry" - Solve all problems in this document

First things first, you must register on the following websites -

1. Codeforces - The most popular website for competitive programming
2. SPOJ - One of the oldest websites for competitive programming. Famous for its classic problem set.

If you face issues on Codeforces, refer this link or feel free to ask someone in Facebook group. You can use whatever programming language you like. However, if you want to prepare for the ICPC, we suggest you write code in C (or C++ if you are already familiar with it. If not, don't worry we'll teach the basics soon).

Given below are the problems for the first week. Good luck and have fun!

Expected time to solve : ~20 minutes per problem

1. codeforces.com/contest/469/problem/A
2. codeforces.com/contest/460/problem/A
3. codeforces.com/contest/244/problem/A
4. codeforces.com/contest/318/problem/A
5. codeforces.com/contest/25/problem/A
6. codeforces.com/contest/23/problem/A

7. codeforces.com/contest/519/problem/A
8. codeforces.com/contest/805/problem/B
9. codeforces.com/contest/610/problem/A
10. codeforces.com/contest/387/problem/A
11. codeforces.com/contest/46/problem/B
12. codeforces.com/contest/4/problem/A
13. codeforces.com/contest/579/problem/A
14. codeforces.com/contest/859/problem/B
15. codeforces.com/contest/195/problem/A
16. codeforces.com/contest/515/problem/A
17. codeforces.com/contest/777/problem/A

Week 2 - The essentials - arrays, stacks, queues, etc.

Congratulations on finishing Week 1! Your perseverance is commendable! However, don't let your brain muscles get lax now, we haven't even scratched the surface :D

This week we will be revisiting some tricks that we learned in DSA Lab for arrays, stacks and queues such as *prefix sums, next greater element, sliding window maximums* etc.

The concepts involved in these tricks lay down the first layer of concepts required to understand a series of increasingly sophisticated data structures and algorithms. Furthermore, many of these tricks are staple questions that are asked in programming interviews.

One last tip before you dive in - Python is usually 5 - 10x slower than C/C++, so it is possible that for some of these questions, it is impossible to pass the problem with Python. In fact, if your aim is ICPC, would **strongly** advise you against using Python.

Problems are given below. Have fun :D

(The last few problems are slightly trickier than usual and may be skipped if you find them hard right now. However make sure to come back to them later, they are really interesting!)

Expected time to solve : ~30 minutes per problem for easier ones. The harder problems may tak 60+ minutes

1. http://www.spoj.com/problems/STACKEZ/
2. http://www.spoj.com/problems/STPAR/
3. http://www.spoj.com/problems/ONP/
4. http://www.spoj.com/problems/QUEUEEZ/
5. http://www.spoj.com/problems/ADAQUEUE/
6. http://www.spoj.com/problems/JNEXT/
7. http://www.spoj.com/problems/MMASS/
8. https://www.codechef.com/problems/COMPILER
9. http://www.spoj.com/problems/HISTOGRA/
10. http://www.spoj.com/problems/ANARC09A/
11. http://www.spoj.com/problems/CSUMQ/
12. http://www.spoj.com/problems/INVCNT/
13. http://www.spoj.com/problems/RANGESUM/
14. https://www.codechef.com/problems/XXOR
15. https://www.codechef.com/problems/RECTQUER

[Hint for 14 and 15, CTRL + A to view :
]

16. http://www.spoj.com/problems/ARRAYSUB/
17. https://www.codechef.com/problems/SWAPMATR (MED)
18. https://www.codechef.com/problems/GCDMAX1 (HARD)
19. http://codeforces.com/contest/91/problem/B (HARD)

Week 2.5 - Interlude - C++ makes CP ez

Knowing C++ is essential to becoming better at Competitive Programming. In fact, knowing how to use some features of C++ itself is enough to improve your rating on Codeforces. Furthermore, there are certain problems in CP that are a nightmare to code in C, and Python is just too slow to get the job done.

This section is a short introduction to some essential tricks in C++ that will help you to solve some of the problems from the later weeks. Please note that this section is by no means exhaustive, and there is a lot to be learned in your C++ course.

Let's begin.

1. A bad but effective way of understanding C++ is that is essentially almost the same as C but with a lot of sugar sprinkled on top to make it nicer. To see this in action, let's try compiling a program written in C but with the C++ compiler instead!

Save this program in a file and name it "hello_world.cpp". To compile it, fire up your terminal, go to the directory where you have saved your file and run the command **g++ hello_world.cpp**. Pretty similar to C, ain't it? Now, you would have created a binary executable called *a.out*. **./a.out** will run the file and print the familiar "Hello World!" to your terminal!

Every program that you have ever written in C, will also compile in g++. So in some sense you are already familiar with a good chunk of C++!

2. The Standard Template Library aka STL is one of the more powerful features offered by C++. It allows you to rapidly create complex data structures with ease.

Refer to the sample programs for the following elements in C++ on this repository - click. If you want to contribute to this repo and add some more useful STL features, you are more than welcome!

Read the programs in the order given in the list below -

- vector
- iterator
- sorting
- lower_bound and upper_bound
- pair
- set and map
- auto

Week 3 - Greedy Algorithms

Greedy algorithms are some of the most fundamental algorithms and simple algorithms in Computer Science. A good grasp on greedy algorithms will form the basis for understanding harder concepts like dynamic programming. Additionally, it is very easy to come up with the solutions for greedy algorithms but it is usually notoriously hard to prove the correctness of the algorithm.

We won't go too deep with greedy algorithms and hence this will be a relatively short week.

Expected time to solve : ~30-45 minutes per problem

1. http://codeforces.com/problemset/problem/276/B
2. http://codeforces.com/problemset/problem/432/A
3. http://codeforces.com/contest/557/problem/A
4. http://codeforces.com/problemset/problem/749/A
5. http://codeforces.com/problemset/problem/401/C
6. http://codeforces.com/problemset/problem/525/A
7. http://codeforces.com/problemset/problem/160/A
8. http://codeforces.com/problemset/problem/507/A
9. http://codeforces.com/problemset/problem/349/A
10. http://www.spoj.com/problems/BUSYMAN/ [MED]

Hint for 10 [Ctrl + A to view]:

11. http://www.spoj.com/problems/CHOCOLA/ [MED]
12. http://www.spoj.com/problems/DIEHARD/ [MED]

Week 4 - Graph Theory Essentials

Arguably one of the most important topics in all of Computer Science, Graph Theory is the bread and butter that makes the Googles and Facebooks of the world tick. It is a key skill that any computer science major should possess in their bag of tricks.

You are able to book an Ola Share or an Uber Pool thanks to Graph Theory. Google Maps is able to get you from point A to point B because of extremely efficient and scalable engineering which is built on the principles of graph theory. The fields of Machine Learning and AI have picked up speed thanks to results from graph theory that allow us to train models efficiently. Even your college admission process works on the principles of graph theory! (Search *Stable Marriage Problem*)

It is needless to say that mastery of the basics concepts of this domain are a must and this will be one of the most important sections that you will attempt this summer.

Also as a general advice, just avoid the comments section on SPOJ. People lie about how "easy" the problem is for them and it will ruin your experience solving the problem.

This is a good write-up on graph algorithms. Some of the topics in this blog were not part of your DSA syllabus and they can be ignored for the time being - http://codeforces.com/blog/entry/16221

Expected time to solve : 30 minutes per problem for easier problems. Hard problems might take 60-180 minutes each. Tricky problems may require 60+ minutes of debugging so write good code.

1. http://codeforces.com/problemset/problem/500/A
2. http://www.spoj.com/problems/PT07Y/
3. http://www.spoj.com/problems/MAKEMAZE/
4. http://www.spoj.com/problems/CATM/
5. http://codeforces.com/contest/755/problem/C
6. http://www.spoj.com/problems/ELEVTRBL/
7. http://www.spoj.com/problems/PPATH/
8. http://www.spoj.com/problems/HERDING/
9. http://www.spoj.com/problems/ONEZERO/ [HARD]
10. http://www.spoj.com/problems/PT07Z/ [TRICKY]
11. http://www.spoj.com/problems/BUGLIFE/
12. http://www.spoj.com/problems/ALLIZWEL/

13. http://www.spoj.com/problems/TOPOSORT/
14. http://codeforces.com/problemset/problem/510/C [TRICKY]
15. http://www.spoj.com/problems/CAPCITY/
16. http://codeforces.com/problemset/problem/427/C [TRICKY]
17. http://www.spoj.com/problems/MST/
18. https://www.hackerrank.com/challenges/even-tree/problem
19. http://www.spoj.com/problems/TFRIENDS/
20. http://codeforces.com/contest/574/problem/B
21. http://codeforces.com/problemset/problem/330/B
22. http://codeforces.com/problemset/problem/505/B
23. http://codeforces.com/problemset/problem/639/B
24. http://codeforces.com/contest/580/problem/C
25. http://codeforces.com/contest/246/problem/D

Week 5 - Number Theory

Number Theory is the branch of mathematics that deals with the study of integers and the properties of integers. It is one of the most interesting fields in mathematics and some of the hardest open problems in mathematics are from this field. It has a wide number of applications in Computer Science and is vital to cryptography. Secure communication exists on the internet thanks to results from Number Theory.

Number Theory Tutorials -
1. https://artofproblemsolving.com/community/c90633h1291397
2. Pages 94 to 101 in this book

Problems in this section will be from UVA Online Judge. This is big section due to the large number of important concepts in this domain. Also, a word of advice on UVA - be careful of how you print the answer and use UVA Debug only when everything you try has failed.

Expected time to solve : 10 minutes per problem for the easy ones. Hard problems might take 60-180 minutes each. Tricky problems may require 60+ minutes of debugging so write good code.

Problems -

1. UVa 294 - Divisors
2. UVa 406 - Prime Cuts
3. UVa 516 - Prime Land
4. UVa 524 - Prime Ring Problem (requires backtracking)
5. UVa 543 - Goldbach's Conjecture
6. UVa 583 - Prime Factors
7. UVa 686 - Goldbach's Conjecture (II)
8. UVa 897 - Annagramatic Primes
9. UVa 914 - Jumping Champion
10. UVa 993 - Product of digits
11. UVa 10006 - Carmichael Numbers
12. UVa 11408 - Count DePrimes
13. UVa 11466 - Largest Prime Divisor
14. http://www.spoj.com/problems/PRIME1/
15. UVa 332 - Rational Numbers from Repeating Fractions
16. UVa 412 - Pi
17. UVa 530 - Binomial Showdown
18. UVa 10193 - All You Need Is Love
19. UVa 10407 - Simple Division
20. UVa 11388 - GCD LCM
21. UVa 718 - Skycraper Floors
22. UVa 10090 - Marbles
23. UVa 10104 - Euclid Problem
24. UVa 374 - Big Mod

25. UVa 602 - What Day Is It?
26. UVa 10174 - Couple-Bachelor-Spinster Numbers
27. UVa 10176 - Ocean Deep! Make it shallow!!
28. UVa 10212 - The Last Non-zero Digit
29. UVa 10489 - Boxes of Chocolates
30. UVa 763 - Fibinary Numbers (Zeckendorf representation)
31. UVa 900 - Brick Wall Patterns (Combinatorics, the pattern is similar to Fibonacci)
32. UVa 10183 - How many Fibs?
33. UVa 10229 - Modular Fibonacci
34. UVa 160 - Factors and Factorials
35. UVa 324 - Factorial Frequencies
36. UVa 568 - Just the Facts
37. UVa 623 - 500!

Week 6 - Dynamic Programming Level 1

Those who don't remember the past are condemned to repeat it, and dynamic programming is all about this saying. Hands down, this is the most important topic in competitive programming and probably the hardest to master. Roughly 30% of all problems in competitive programming are based on dynamic programming, in one way or another.

Tutorials -

1. https://www.topcoder.com/community/data-science/data-science-tutorials/dynamic-programming-from-novice-to-advanced/
2. https://www.hackerearth.com/practice/algorithms/dynamic-programming/introduction-to-dynamic-programming-1/tutorial/
3. https://www.codechef.com/wiki/tutorial-dynamic-programming#

4. https://medium.freecodecamp.org/demystifying-dynamic-programming-3efafb8d4296
5. https://www.quora.com/Are-there-any-good-resources-or-tutorials-for-dynamic-programming-DP-besides-the-TopCoder-tutorial

Problems -

1. https://www.spoj.com/problems/FIBEZ/
2. https://www.hackerrank.com/challenges/coin-change/problem
3. https://www.spoj.com/problems/BYTESM2/
4. https://www.hackerrank.com/challenges/stockmax/problem
5. https://www.hackerrank.com/challenges/play-game/problem
6. https://www.hackerrank.com/challenges/maxsubarray/problem
7. https://www.hackerrank.com/challenges/longest-increasing-subsequent/problem
8. https://www.hackerrank.com/challenges/dynamic-programming-classics-the-longest-common-subsequence/problem
9. https://www.hackerrank.com/challenges/sherlock-and-cost/problem (NEW PERSPECTIVE!)
10. https://www.spoj.com/problems/KNAPSACK/
11. https://www.spoj.com/problems/EDIST/
12. https://www.hackerrank.com/challenges/red-john-is-back/problem
13. http://codeforces.com/problemset/problem/466/C
14. http://codeforces.com/problemset/problem/189/A
15. https://www.spoj.com/problems/MIXTURES/ [Hint :]
16. https://www.spoj.com/problems/SAMER08D/
17. http://codeforces.com/problemset/problem/676/C
18. http://codeforces.com/problemset/problem/431/C

Week 7 - Segment Trees and Binary Indexed Trees

Trees are one of the most versatile data structures in Computer Science. One could say that they are omnipresent. The range query problem has been studied extensively and is extremely important in higher dimensions.

Consider this example - you are running a programming club in college and must shortlist students who will join the club in the next academic year (don't worry, we are not trying to foreshadow anything :P). You come up with the criteria that all students who have scored 90% or above in Math and have joined between 2016 and 2017 are eligible to join the club. If we represent each student as a point (x,y) in the 2D plane, where x is their score in Math and Y is the year in which they have joined, the task of shortlisting students is essentially to pick all points that are inside a rectangle in the 2D plane - a 2D range query problem. Now, imagine the same problem but on the scale of Google or Facebook when you try to search for people filtered on certain attributes.

A thorough understanding of Segment Trees and Binary Indexed trees can form the foundations that can inspire you to design your own tree-based data structures in the future.

A small tip - Anything that can be done with a BIT can also be done with a Segment Tree, but the opposite is not always true. Thus, Segment Trees are strictly stronger than BITs. However, BITs are significantly faster than Segment Trees and are also faster to code.

Tutorial -

1. http://codeforces.com/blog/entry/15729
2. http://codeforces.com/blog/entry/15890

3. https://kartikkukreja.wordpress.com/2014/11/09/a-simple-approach-to-segment-trees/
4. https://kartikkukreja.wordpress.com/2015/01/10/a-simple-approach-to-segment-trees-part-2/
5. https://www.topcoder.com/community/data-science/data-science-tutorials/binary-indexed-trees/

Expected time to solve : Anywhere between 30 and 120 minutes for standard problems. Hard problems might take multiple hours of thinking and debugging! Codeforces problems are on the harder side.

Problems -

1. http://www.spoj.com/problems/GSS1/
2. http://www.spoj.com/problems/GSS3/
3. http://www.spoj.com/problems/GSS4/
4. http://www.spoj.com/problems/BRCKTS/
5. http://www.spoj.com/problems/FREQUENT/
6. http://www.spoj.com/problems/YODANESS/
7. http://www.spoj.com/problems/HORRIBLE/
8. http://www.spoj.com/problems/GSS2/
9. http://www.spoj.com/problems/NICEDAY/
10. http://www.spoj.com/problems/KGSS/
11. http://codeforces.com/contest/339/problem/D
12. http://codeforces.com/contest/356/problem/A
13. http://codeforces.com/contest/459/problem/D
14. http://codeforces.com/contest/61/problem/E
15. http://codeforces.com/contest/380/problem/C
16. http://codeforces.com/contest/474/problem/F
17. http://codeforces.com/contest/292/problem/E
18. http://codeforces.com/contest/501/problem/D
19. http://codeforces.com/contest/220/problem/E

Segment Tree Mastery. Hard.

We advise you to come back to this section later.

If you wish to Master Segment Trees, solve all problems given here
http://codeforces.com/blog/entry/22616

And learn Persistent Segment Tree from this wonderful blog -
https://blog.anudeep2011.com/persistent-segment-trees-explained-with-spoj-problems/

Week 8 - Geometry

Week 9 - Dynamic Programming Level 2

Problems -

1. http://codeforces.com/problemset/problem/474/D
2. https://www.hackerrank.com/challenges/matrix-land/problem
3. https://www.spoj.com/problems/PT07X/
4. https://www.spoj.com/submit/BACKPACK
5. https://uva.onlinejudge.org/index.php?option=onlinejudge&page=show_problem&problem=2445
6. https://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&page=show_problem&problem=944
7. https://uva.onlinejudge.org/index.php?option=onlinejudge&page=show_problem&problem=1852

Week 10 - Dynamic Programming Level 3

Problems -

1. https://www.spoj.com/problems/COURIER/
2. http://codeforces.com/problemsets/acmsguru/problem/99999/269

Week 10 - Dynamic Programming Level 4

---

1. http://www.spoj.com/problems/TRSTAGE/
2. http://www.spoj.com/problems/HIST2/
3. http://www.spoj.com/problems/LAZYCOWS/
4. http://www.spoj.com/problems/AGGRCOW/
5. http://www.spoj.com/problems/PPATH/
6. http://www.spoj.com/problems/TRT/
7. http://www.spoj.com/problems/BITMAP/ O(nm) solution
8. http://www.spoj.com/problems/GSS3/
9. http://www.spoj.com/problems/FIBOSUM/
10. http://www.spoj.com/problems/ASSIGN/
11. Now, read this tutorial - https://threads-iiith.quora.com/Solving-Dynamic-Programming-with-Matrix-Exponentiation and solve all problems given in the tutorial
12. http://codeforces.com/contest/118/problem/D
13. http://codeforces.com/contest/2/problem/B
14. http://codeforces.com/contest/4/problem/D
15. http://codeforces.com/contest/33/problem/C
16. http://codeforces.com/contest/5/problem/C
17. http://codeforces.com/contest/82/problem/D
18. http://codeforces.com/contest/73/problem/C
19. http://codeforces.com/contest/38/problem/E

20. http://codeforces.com/contest/30/problem/C
21. http://codeforces.com/contest/119/problem/C
22. http://codeforces.com/contest/101/problem/B
23. http://codeforces.com/contest/113/problem/B
24. http://codeforces.com/contest/19/problem/B
25. http://codeforces.com/contest/8/problem/C
26. http://codeforces.com/contest/111/problem/C
27. http://codeforces.com/contest/128/problem/C
28. http://codeforces.com/contest/13/problem/C
29. http://codeforces.com/contest/46/problem/E
30. http://codeforces.com/contest/77/problem/C
31. http://codeforces.com/contest/75/problem/D
32. http://codeforces.com/contest/54/problem/C
33. http://codeforces.com/contest/41/problem/D
34. http://codeforces.com/contest/69/problem/D
35. http://codeforces.com/contest/11/problem/D
36. http://codeforces.com/contest/21/problem/C
37. http://codeforces.com/contest/27/problem/E
38. http://codeforces.com/contest/95/problem/E
39. http://codeforces.com/contest/123/problem/C
40. http://codeforces.com/contest/126/problem/D
41. http://codeforces.com/contest/49/problem/E

Blackmagic and Wizardry

Bit masking
Sparse Tables and Binary Ascent
Euler Tours
Meet-in-the-middle
Dynamic Programming Optimizations
DP over broken profile
Lagrange Multipliers
Square Root Decomposition
Heavy Light Decomposition
Centroid Decomposition
Persistent Data Structures
Palindromic Tree
Aho-Corasick Algorithm
Suffix Arrays,
Suffix Automaton
Matching and Network Flows
2-SAT
FFT, NTT, FWHT
Cut Space and Cycle Space
Matroids and Matroid Intersections