Getting Started in the Sport of Programming

This document is to guide those people who want to get started with competitive programming. The only prerequisite is that you know basics of at least one programming language. This document was initially made for freshers at IIT Kanpur. So it may contain phrases like “discuss” , “discussed” etc kindly ignore them.

Hope this document will help you.

If you have any queries / suggestions please contact us.

Abhilash Kumar
abhilak@iitk.ac.in
https://www.facebook.com/abhilash.276

Triveni Mahatha
triveni@iitk.ac.in
https://www.facebook.com/triveni.mahatha

Co ordinators [2014-15] @ Programming club IIT Kanpur
https://www.facebook.com/groups/pclubiitk/

These are from my experience not something written on stone.

- You will need to show motivation.
- This doc is to introduce you to the world of algorithms and competitive programming.(exclusively for beginners)
- Languages
    - C/C++/Java (its your choice)
    - We will focus on C++ , Java is slow(One big advantage of java is Big Ints,will see later).
    - C++ is in a way superset of C , With few additional tools.So basically if you know C you know C++ and you are ready to get started else go back and learn how to write codes in C/C++.
    - Sometimes Knowledge of python is helpful when you need really big Integers.

PARTICIPATE PARTICIPATE PARTICIPATE(the only mantra)

- SPOJ , Its a problems archive ,(Recommended For ALL Beginners)
    - Start with problems having maximum number of submissions.Solve first 20 problems then start skipping . When you develop little confidence start following some good coders (check their initial submissions) . Then start solving problem topic wise.
    - Never get stuck for too long in initial period . Try to google your doubts or contact someone (BUT ONLY IN BEGINNING).
    - Before getting into live contests like codechef/codeforces solve at least 50 problems on SPOJ.
- CODECHEF ,Only 3 Contests per month(MUST for all)
    - Even if you are not able to solve a problem do always look at its editorials and then code and submit it(because its the only way you will learn).
    - And even if you able to solve then also check the editorials and look at codes of top coders . **See how they implemented their code.**
    - Same points for topcoder and codeforces.
- Codeforces ,4-5 2 Hrs long contest(Once you develop some confidence)
- TOPCODER (Once you have proper experience and can write codes very fast)

Online Programming Contests
You write codes and submit codes online . The judge runs your code and checks the output of your program for several inputs and gives the result based on your program's outputs.You must follow exact I/O formats. Do not print statements like : "please enter a number", etc :P

Each problem has constraints :
Properly analyse the constraints .

- Time Limit in seconds (gives you an insight of what order of solution it expects) -> **order analysis**(discussed later)**.**
- The constraints on input (very imp): Most of the time you can correctly guess the order of the solution by analysing the input constraints and time limit .
- Memory Limit (You need not bother unless you are using insanely large amount of memory).

Types of errors you will encounter apart from wrong answer :

- Run Time Error (Most Encountered)
    - Segmentation fault (accessing an illegal memory address)
        - You declared array of smaller size than required or you are trying to access negative indices .
    - Declaration of an array of HUGE HUGE(more than 10^8 ints) size -_- .
    - Dividing by Zero / Taking modulo with zero :O .
    - USE gdb (will learn in coming lectures)
- Compilation error
    - You need to learn how to code in C++.
    - USE GNU G++ compiler or IDEONE(be careful to make codes private).
- Time Limit Exceeded
    - You program failed to generate all output within given time limit.
    - Input Files are not randomly generated , they are made such that wrong code does not pass.
    - Always think of worst cases before you start coding .Always try to avoid TLE.
    - Sometimes a little optimizations are required and sometimes you really need a totally new and efficient algorithm (this you will learn with time).
    - So whenever you are in doubt that your code will pass or not .Most of the time it won't pass :P .
    - Again do proper order analysis of your solution .

Sometimes when you are stuck . Check the running time of other accepted codes to take an insight like what Order of solution other people are writing / what amount of memory they are using.

4 MB ~ array of size 10^6 . Or 2-d array of size 10^3*10^3
Standard Memory limits are of Order of 256MB.

Order analysis :
Order of a program is a function dependent on the algorithm you code. We wont go in theoretical details just think Order of program as the total number of steps that program will take to generate output generally a function based on input like O(n^2) O(n) O(log n) .

Suppose you write a program to add N numbers .See the following code.

```
int cur,sum=0;
for(int i=0;i<n;i++)
{
        scanf("%d",&curr);
        sum = sum+curr;
}
```

Total number of computations = n*(1+1+1+1)
n times checking i<n.
n times i++
n times scanf
n times + operating

So total of 4*N.
We remove the constant and call it O(N)
This is the simplest I can explain.You will get further understanding with practice and learning.

You must know running time of these algorithms **(MUST)**
Binary Search -> ?
Merge / Quick sort -> ?
Searching an element in **sorted/unsorted** array -> ?
HCF / LCM / Factorization / Prime CHeck ?

We all know the computation power of a processor is also limited.
Assume 1 sec ~ 10^8 operations per second . (for spoj old server it is 4*10^6).
Keep this in mind while solving any problem.

If your program takes O(n^2) steps and problems has T test cases . Then total order is T*N^2.

For T < 100 and N < 1000 . It will pass .
But for T < 1000 and N < 1000 it wont .
Neither for T < 10 and N < 10000 .

INT OVERFLOW :

Sum three numbers.

Constraints :

0 < a,b,c < 10^9

```
 int main()
{
        int a , b,c;
        scanf(“%d %d”,&a,&b,&c);
        int ans = a + b + c;
        printf(“%d”,ans);
        return 0;
}
```

This program won't give correct output for all cases as 3*10^9 cannot be stored in INTS you need long long int or unsigned int (4*10^9).

what if 0<a,b,c < 10^1000 ?

Comparing Doubles :

```
int main()
{
        float a ;
        scanf("%f",&a);
        if(a == 10 ) printf("YES");
        return 0;
}
```

float / double don't have infinite precision . BEWARE (6/15 digit precision for them respectively)
Lets do the following problem.
http://www.spoj.com/problems/GAMES/ **(discussed)**

STL's :)
They already have lots of standard functions and data structures implemented within itself which we can use directly.
Data Structures (To be discussed in later lectures)

- Vectors
- Stack
- Queue
- Map
- Set

Functions

- Sort
- Reverse
- GCD
- Swap
- permutation
- binary search (left + right)
- max , min
- pow , powl
- memset

Now imagine writing codes using these . It would be much more simpler now.
No the previously written code can be written in < 10 lines .

What Headers to include ??
Basically the above functions / DS are in different libraries . But you can include everything using just one header.

```
#include <bits/stdc++.h>
```

http://www.codechef.com/COOK46/problems/ANUUND/ **(discussed)**

What if we have to sort an Array of structure ?
You can either make a struct and write compare function for it.(Read more at cplusplus.com)
Or you can use an vector of pair

Now you are ready to start competitive programming .
You can continue reading this doc or get started on your own . Good luck :)

You first must learn the basic and well known algorithms . Not only algorithm but you must also understand why that works , proof , code it and analyze it . To know what basic algorithms you must know you can read here .
http://www.quora.com/Algorithms/What-is-needed-to-become-good-algorithmist-like-top-rankers-in-Topcoder-Spoj-GCJ
http://www.quora.com/Algorithms/What-are-the-10-algorithms-one-must-know-in-order-to-solve-most-algorithm-challenges-puzzles
http://www.quora.com/Computer-Science/What-are-the-10-must-know-algorithms-and-data-structures-for-a-software-engineer

Also read these answers on how to start competitive programming and get good at it.
http://www.quora.com/ACM-ICPC-1/For-an-ACM-beginner-how-should-I-start
http://www.quora.com/Can-I-crack-the-ACM-ICPC-in-1-5-years-if-I-have-to-start-from-scratch
http://www.quora.com/Competitive-Programming/What-was-Anudeep-Nekkantis-Competitive-Programming-strategy-to-become-35th-in-Global-ranking-in-just-6-7-months

Topcoder has very nice tutorials on some topics here
http://community.topcoder.com/tc?module=Static&d1=tutorials&d2=alg_index.
You must also read this book topic wise to understand an algorithm in more broader way
http://ldc.usb.ve/~xiomara/ci2525/ALG_3rd.pdf.

To get good at writing fast codes and improving your implementation follow this:

My personal advice is to start practicing on topcoder . Start with Div2 250 master it then start with Div2 500 master it then move to Div1 250 .Also read the editorials of problem you solve and the codes of fastest submissions to learn how to implement codes in simple and elegant way.Meanwhile keep learning algorithms and keep practicing them on SPOJ or Codechef or Codeforces . And do read the tutorials, after a time you will realize that the tricks and methods to solve are repeating themselves . We learn from practice only . If you read same thing 5 times in different tutorials then it will not be stored in your short term memory only right .

Below are few topics to start with and problems related to those topic.
They are very basic stuffs and you can learn all you need to know by just googling.
Do all if you are a beginner .

PRIMES

Prime Check (O(log n) also possible read about miller-rabbin)
Factorisation …
Number of factors ..
Sum of factors ..
Generation Primes in a upto N (or in a range)..
Number of primes till N …
Phi (N) ...
Practice Problems :
http://www.spoj.com/problems/NDIV/ (**discussed**)

HW

http://codeforces.com/problemset/problem/431/B
http://www.spoj.com/problems/GAMES/
http://www.spoj.com/problems/GCJ101BB/
http://www.spoj.com/problems/GCJ1C09A/
http://www.spoj.com/problems/MAIN72/
http://www.spoj.com/problems/WINDVANE/
http://www.spoj.com/problems/NDIV/
http://www.spoj.com/problems/PTIME/
http://www.spoj.com/problems/NDIVPHI/
http://www.spoj.com/problems/NOSQ/
http://www.spoj.com/problems/AFS/
http://www.codechef.com/MAY13/problems/WITMATH/
http://www.spoj.com/problems/CUBEFR/
Try as many as you can . If all are done then notify me .
Other things that you can read meanwhile
1)Euler Totient function and Euler's theorem [[READ]]
2)Modulo function and its properties
3)Miller-Rabin Algorithm [[READ]]
4)Extended Euclid's Algorithm [[READ]]
5)Keep exploring STL

Prove running time of HCF is O(log n) , Tried sorting of structures , Practice few problems on ONJ's ,Try to do + - * operations on large numbers(<1000 digits) using char / string , number of factors and sum of factors in sqrt(n) time ,Number of primes till N ,HW problem

Basic Number Theory
Modulo operations
BIG Mod
a^b % p (demo if asked) , fib_n%p
n! % p (what if we have lots of test cases)
ETF (calculation / calculation using sieve)
Euler theorem , Fermat's little theorem , Wilson theorem [[READ]]
nCr % p (inverse modulo) (read about extended euclid algorithm)
(p-1)! % p , Use of fermat theorem in Miller-Rabin (Probabilistic) (**miller-rabin.appspot.com**)
64 C 32 < 10^19 (Pre compute 2D array)
Number of ways to traverse in 2D matrix (what if some places are blocked)
a^b % c . Given Hcf(a,c) = 1 . What if Hcf(a,c) ! = 1. [[READ CRT probably not required]]
matrix exponentiation
solving linear recurrence using matrix exponentiation(like fibonacci)

Practice problems:
http://www.spoj.com/problems/DCEPC11B/ **(discuss)**
http://www.codechef.com/MAY13/problems/FTRIP/
http://www.spoj.com/problems/FIBOSUM/
http://www.spoj.com/problems/POWPOW/

Power of BITS :::
numbers in memory are stored as bits ..
so bits manipulation are always faster..
| -> or
& -> and
^ -> xor (odd number of ones)
<< -> left shift
>> -> right shit
EX- memset() is ~ 5 times faster

Generating Subsets
Checking is the number power of two or not !(x&(x-1))
Max power of 2 that divides the number (x & -x)
Counting number 1's in bit representation of any number , check nth bit (x & (1 << n))

Q) Check whether there exist a subset whose sum of element is equal to a required sum.
Also tell how many elements make up that sum.(What if this condition is not there ?)
http://www.spoj.com/problems/SPCO/
http://codeforces.com/problemset/problem/114/B [[**this**]]

Binary Search :
left / right binary search (understand thier use)

http://www.spoj.com/problems/AGGRCOW/ [discuss]
http://codeforces.com/problemset/problem/431/D [very intersting give this problem before starting topic]

http://www.spoj.com/problems/PIE/
http://www.spoj.com/problems/TETRA/
http://www.spoj.com/problems/KOPC12A/

Practice Problems

http://www.spoj.com/problems/DCEPC11B/ discussed

http://www.spoj.com/problems/AGGRCOW/ discussed

http://www.codechef.com/problems/CHEFBM discussed

http://www.codechef.com/JUNE13/problems/PERMUTE

http://www.spoj.com/problems/KOPC12A/ (recommended)

http://www.codechef.com/MAY13/problems/WITMATH/ (recommended)

http://codeforces.com/problemset/problem/431/D (recommended)

http://www.spoj.com/problems/SPCO/

http://www.spoj.com/problems/FIBOSUM/

http://www.spoj.com/problems/POWPOW/ (recommended)

http://www.codechef.com/AUG13/problems/CNTSOLS/

http://www.spoj.com/problems/IOPC_14F/

http://www.spoj.com/problems/NDIVPHI/ (recommended)

http://www.spoj.com/problems/AU12/ (Tutorial type question)

http://www.spoj.com/problems/ETF/ (Tutorial type question)

http://codeforces.com/problemset/problem/114/B (Tutorial type question)

C++ STL's

Vectors: 1d /2d

http://www.codechef.com/MAY14/problems/CHEFBM

Stacks [[question 1st]]

http://www.codechef.com/MAY14/problems/COMPILER (can solve without stack also)
http://codeforces.com/problemset/problem/344/D

Queue

http://www.spoj.com/problems/DONALDO/ [[other very simple way]]

Priority Queue [[question first]]

http://codeforces.com/gym/100247/problem/I

Set [[question first]]

http://www.spoj.com/problems/FACEFRND/
What if i tell you apart from scanning the input this problem can be done in 2 lines :O .

Map [[if time permits]]
http://www.codechef.com/MARCH13/problems/TOTR/
http://codeforces.com/gym/100247/problem/C

PRACTICE
http://www.spoj.com/problems/HISTOGRA/ (using stack .. i dont remember that method :P)
http://www.spoj.com/problems/HOMO/
http://www.spoj.com/problems/NGM2/

GRAPHS (start with prime path problem)

Think graphs as a relation between node , related nodes are connected via edge.

How to store a graph ? (space complexity) (check if two nodes are directly connected or not)
a) Adjacency Matrix (useful in dense graph)
b) Adjacency list (useful in sparse graph) O(min(deg(v),deg(u)))

Definitions :
neighbours , node , edge ,degree
directed , undirected graph ,
tree , leaves , children , rooted tree , parent
binary tree , k-nary tree
cycle , path , walk

Tree:
n-1 edges,connected graph with n-1 nodes , acyclic graph with n-1 nodes

DAG , Bipartite Graph

BFS
Shortest path in unweighted graphs

DFS

Topological sort

Graph in 2D matrix

matrix with empty and blocked cells , reachability problem

Discussion :
relation <-> graphs (reduction into a graph problems)
Ex-prime path , cam5 , paradox , onezero
find the two most farthest nodes in a tree(n*BFS , 2*BFS)
Number of connected components
Problems related to graphs:

http://www.codechef.com/JUNE14/problems/DIGJUMP
http://www.spoj.com/problems/PRATA/
http://www.spoj.com/problems/ONEZERO/
http://www.spoj.com/problems/PPATH/
http://www.spoj.com/problems/PARADOX/
http://www.spoj.com/problems/HERDING/
http://www.spoj.com/problems/PT07Z/
http://www.spoj.com/problems/NICEBTRE/
http://www.spoj.com/problems/CERC07K/
http://www.spoj.com/problems/BUGLIFE/
http://www.spoj.com/problems/COMCB/
http://www.spoj.com/problems/NAKANJ/
http://www.codechef.com/IOPC2013/problems/IOPC13N/
http://www.codechef.com/IOPC2013/problems/IOPC13G/
http://www.codechef.com/IOPC2013/problems/IOPC13C/