contents are available here as well : http://sportprogramming.blogspot.in/2014/07/getting-started-with-sport-of.html

Getting Started with 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 but now it has been fully edited to help others too.It contains all the baisic things a beginner should know !!

Hope this document will help you.

Note : Please note that this doc is not meant to explain concepts in details. The aim of this blog is to guide you about which topics you should read and practice in a systematic way. However, in many places short explanations have been included for their relevance. Relevant problems are given after each topic. Proper sources are given from where these concepts can be studied. Where sources are not mentioned, that means these are very very popular and you can get to know about them just by a single google search. Move forward and enjoy it !

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

All the following things are from our experience not something written on stone.

- You will need to show motivation.

- Languages that should be used
    - C/C++/JAVA (your choice)
    - We will focus on C++, JAVA is slow (one big advantage of JAVA is Big Integers, we will see later)
    - C++ is like superset of C with some additional tools. So, basically if you have knowledge of C, you are ready to code in C++ as well. Otherwise go back and learn how to write codes in C/C++.
    - Sometimes knowledge of PYTHON is helpful when you really need big integers.

PARTICIPATE PARTICIPATE PARTICIPATE (the only mantra)

- SPOJ: Its a problem Archive (recommended for all beginners)
    - Start with problems having maximum submissions. Solve first few problems (may be 20). Build some confidence. Then start following some good coders (check their initial submissions). Then start solving problems topic wise.
    - Never get stuck for too long in the initial period. Google out your doubts and try to sort them out or you can discuss with someone (ONLY IN THE BEGINNING).
    - Before getting into live contests like codeforces or codechef, make sure that you have solved about 50-70 problems on SPOJ.
- CODECHEF: Do all the three contests every month. Do participate in CodeChef LunchTime for sure.
    - Even if you are unable to solve a problem do always look at the editorials and then code it and get it accepted (this is the way you will learn).
    - And even if you are able to do it, do look at the codes of some good coders. **See how they have implemented**. Again you will learn.
    - Same point apply to TopCoder and Codeforces as well.
- Codeforces: 4 to 5 short contests of 2 hour in a month (Do them once you develop some confidence).
- TopCoder: Once you have proper experience and you can write codes very fast.

Online Programming Contests

You write codes and submit them 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. For example, do not print statements like : "please enter a number", etc :P

Each problem has constraints :

Properly analyse the constraints before you start coding.

- **Time Limit** in seconds (gives you an insight of what is the 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 may 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 .
    - Again do proper order analysis of your solution .

Wrong Answer [most encountered]
Wrong answer means that the output given by your program did not match the correct output for that input (or did not fulfill the conditions in case multiple solutions were possible). This is the most frequently occurring bug that you will face and getting rid of it can be a pain.

- First of all you must check that your program gives correct output for the sample test cases, exactly satisfying the output format.
- Read your code completely once before testing. This way you will be able to remove any obvious bugs.
- Check for incorrect variable initializations / uncleared memory, etc. These errors can also occur when you copy paste code.
- In case you keep getting wrong answer even after you have tried to find the bug in your program you must rethink upon you algorithms and prove it if you haven't done so.If you find bug in your algorithm start working on new algorithm.

Now its time for some serious debugging.Modulate your code, that means if I have to first generate a graph and then apply shortest path on it, I check first if the graph has been generated correctly. Some speed can be traded for accuracy, with practice you will learn to write accurate programs faster.

If you have already proven your algorithm then try to generate some test cases and check it against you program.
Case 1: **Your program fails for you own inputs but you don't understand how.**
- Try to put a lot of print/cout statements for all important variable and check at each step that they are changing as it was desired.Most probably you will find the bug.

Case 2: **You program passes for you hand made test case but still gives WA on judge.**

- Now there are two possible scenarios . Your algorithm is wrong or you program fails at tricky test cases like overflow or corner cases etcs. Assuming that you have already proven you algorithm you now need to find those corner test case.For this write a generator file which will generate a lot of test cases.And write a brute force solution which you must be 100% sure that it give correct output.Now match the output of these two codes (don't use large test cases if your brute force is too slow).As you don't have time limit condition for your machine your brute force can be slow.Check for which test case the answer differs then again go to case 1 with this test case.

Other Points:
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. You cannot allocate more than 4 MB space inside a function (it gives segmentation fault). Thus if you have to make an array of size >= 10^6 , make it global or use dynamic memory allocation.

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) or O(log n) .

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

```
int curr,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 %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). Use a margin of EPS (~ 0.0000001) in comparing. Ex -
if (abs(a - 10) < EPS) printf("YES\n");

Lets do the following problem.
http://www.spoj.com/problems/GAMES/ **(discussed)**

Standard Template Library (STL):
In your code sometimes you need some data structures and some functions which are used quite frequently. 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 / Priority_queue (heap)
- Map
- Set

Functions
- Sort
- Reverse
- GCD
- Swap
- permutation
- binary search (left + right)
- max , min
- pow , powl (find out the difference)
- memset, fill
- string stream

Now imagine writing codes using these inbuilt functions and data structures . It would be much more simpler now.

What headers/libraries should you include ?
Basically the above functions / DS are in different libraries So in some cases you may need to include many headers . But you can include everything using just one header.Ignore headers and #define's in other coders codes for now.

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

Try the following problem :
www.codechef.com/problems/ANUUND
Which of the above inbuilt function did you use ?
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
Read This:
http://community.topcoder.com/tc?module=Static&d1=tutorials&d2=sorting

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

First, you 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(Advice make a quora account if don't have one).
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. [Introduction to Algorithms By Cormen]

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.In the beginning you will feel that every problem uses a different algorithm,How can anyone learn so much but I assure you after a time the logics/algorithms will start to repeat and after some time you will be able to solve problems using those algorithms in actual contests.

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.
"When i will get some time I will try to update and give more details about the topics a newbie should cover."
Try to do all the problems stated below if you are a beginner.

PRIMES

- Prime Check (O(log n) also possible read about miller-rabbin)
- Factorization
- Number of factors
- Sum of factors
- Generating Primes using sieve of eratosthenes
- Bounds on number of primes till N
- Euler's totient function
- **Practice Problems :**
    - http://www.spoj.com/problems/NDIV/
    - 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.
- Other things that you can read meanwhile
    - Euler Totient function and Euler's theorem [[READ]]
    - Modulo function and its properties
    - Miller-Rabin Algorithm [[READ]]
    - Extended Euclid's Algorithm [[READ]]
    - Keep exploring STL
    - Prove running time of HCF is O(log n)
    - Try sorting of structures
    - Practice few problems on several Online Judges
    - Try to do + - * operations on large numbers(<1000 digits) using char array (for learning implementation)
    - Number of factors and sum of factors in sqrt(n) time ,Number of primes till N

Go through these tutorials (The listed problems might be tough but do read the tutorial)

http://community.topcoder.com/tc?module=Static&d1=tutorials&d2=primalityTesting
http://community.topcoder.com/tc?module=Static&d1=tutorials&d2=combinatorics
http://community.topcoder.com/tc?module=Static&d1=tutorials&d2=math_for_topcoders
http://community.topcoder.com/tc?module=Static&d1=tutorials&d2=primeNumbers

Basic Number Theory

- Modulo operations and Inverse modulo
- How to compute a ^ b % p in O(log b), where p is prime
- Find Nth fibonacci number modulo p [Read Matrix exponentiation]
- n! % p (what if we have lots of test cases and n<10^6 and P is fixed)
- ETF (calculation / calculation using sieve / properties) [[read Euler's_totient_function]]
- Euler theorem , Fermat's little theorem , Wilson theorem [[READ]]
- nCr % p (inverse modulo) (read about extended euclid algorithm)
- (p-1)! % p for prime p (read wilson's theorem), Use of fermat theorem in Miller-Rabin (Probabilistic) (**miller-rabin.appspot.com**)
- 64 Choose 32 < 10^19 we can precompute till herein a 2 dimensional array [Learn use of the recursive relation : (n+1)Cr = nCr + nC(r-1)]
- Number of ways to traverse in 2D matrix[Catalan Number] (what if some places are blocked ? Hint : DP)
- a^b % c . Given Hcf(a,c) = 1 .And what if Hcf(a,c) ! = 1. [[READ Chinese Remainder Theorem, not used much in competition]]
- Matrix Exponentiation
- solving linear recurrence using matrix exponentiation(like fibonacci)

- Practice problems:
  - http://www.spoj.com/problems/DCEPC11B
  - http://www.codechef.com/MAY13/problems/FTRIP/
  - http://www.spoj.com/problems/FIBOSUM/
  - http://www.spoj.com/problems/POWPOW/
  - http://www.spoj.com/problems/POWPOW2 [[CRT]]

Power of BITS

- Numbers are stored as binary bits in the memory so bits manipulation are alway faster.
- Bitwise or operator : |
- Bitwise and operator : &
- Bitwise xor operator : ^
- Bitwise left shift : <<
- Bitwise right shift : >>
- Memset and its uses using function : sizeof()
- Bitmask and use of Bitmask in Dynamic Programming [[subset DP]]
- Some cool Tricks
  - n = n * 2 :: n = n << 1
  - n = n /2 :: n = n >> 1
  - checking if n is power of 2 (1,2,4,8…) ::checking !(n & (n-1))
  - if x is max power of 2 dividing n, then x = (n & -n)
  - Total number of bits which are set in n = __builtin_popcount(n)

    - setting xth bit of n :: n |= (1<<x)
    - checking if xth bit of n is set :: checking if n&(1<<x) is non zero
- Problem : You are given N numbers and a numbers S. Check if there exist some subset of the given numbers which sums equal to S .What if you are asked to compute the number of such subsets ?
- Practice :
    - http://www.spoj.com/problems/SPCO/
    - http://codeforces.com/problemset/problem/114/B
    - http://www.spoj.com/problems/CLEANRBT/
    - More will be added later

Read this for further knowledge
http://community.topcoder.com/tc?module=Static&d1=tutorials&d2=bitManipulation

Binary Search :

- Try this : http://codeforces.com/problemset/problem/431/D
- Understand the concept of binary search. Both left_binary_search and right_binary_search. Try to implement it on your own. Look at others implementation.
- sample implementation :

```
int l = 0, r = 10000,  key_val = SOME_VALUE, m;
while (r - l > 1)
{
m = (l+r) >> 1;
int val = some_non_decreasing_function(m);
if(val < key_val) l = m;
else r = m;
}
if  (some_non_decreasing_function(l) == key_val ) return l;
else return r;
```

// this can be modified in a variety of ways, as required in the problem

- Practice Problems:
    - http://www.spoj.com/problems/AGGRCOW/
    - http://codeforces.com/problemset/problem/431/D [[Learn't something new ?]]
    - http://www.spoj.com/problems/PIE/
    - http://www.spoj.com/problems/TETRA/
    - http://www.spoj.com/problems/KOPC12A/

The Beauty of Standard Template Library of C++

- Vectors in one dimension and two dimension
    - http://www.codechef.com/MAY14/problems/CHEFBM

- solve : http://www.codechef.com/MAY14/problems/COMPILER
- Now use stacks to taste its beauty and solve the following problem too.

    - http://codeforces.com/problemset/problem/344/D

- Queue
    - http://www.spoj.com/problems/DONALDO/
- Priority Queue
    - http://codeforces.com/gym/100247/problem/I [[First try without using Priority queue]]

- Set
    - http://www.spoj.com/problems/FACEFRND/ [[First try without using set]]
        - What if I tell you that apart from scanning the input this problem can be done in 2 lines ? Interesting ? Think!

- Map
    - http://www.codechef.com/MARCH13/problems/TOTR/
    - http://codeforces.com/gym/100247/problem/C
    - http://www.spoj.com/problems/SETSTACK/ [[map, set, set_intersection / union]]

Some Practice Problems Before you proceed further

- http://www.spoj.com/problems/DCEPC11B/
- http://www.spoj.com/problems/AGGRCOW/
- http://www.codechef.com/problems/CHEFBM
- 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/ (easy)
- http://www.spoj.com/problems/ETF/ (easy)
- http://codeforces.com/problemset/problem/114/B (easy)
- http://www.spoj.com/problems/HISTOGRA/ [[Hint : use stacks]]
- http://www.spoj.com/problems/HOMO/
- http://www.spoj.com/problems/NGM2/
- http://www.spoj.com/problems/RENT/ [[recommended]]

GRAPHS

- Try the following problems :
    - Prime Path
    - Prayatna PR

  Any Ideas ?

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

- How to store a graph ? (space complexity)
  - Adjacency Matrix (useful in dense graph) using 2-D array of bool/ints.
  - Adjacency List (useful in sparse graph) O(min(deg(v),deg(u))) using vector of ints.

- You must know the following terminologies regarding Graphs :
  - Neighbours
  - Node
  - Edge
  - Degree of vertices
  - Directed Graph
  - Connected Graph
  - Undirected Graph
  - Connected components
  - Articulation Points
  - Articulation Bridges
  - Tree [[connected graph with N nodes and N-1 edges]]
    - Leaves
    - Children
    - Parent
    - Ancestor
    - Rooted Tree
    - Binary Tree
    - K-ary Tree
  - Cycle in graph
  - Path
  - Walk
  - Directed Acyclic Graph [[DAG]]
    - Topological Sorting (Not very important, in my opinion)
  - Bipartite Graph (Tree is an example of Bipartite Graph . Interesting Isn't it.)

- Breadth First Search/Traversal (**BFS**) [[very important, master it as soon as possible]]
  - Application : Shortest path in unweighted graphs

- Depth First Search/Traversal (**DFS**) [[very very important, master it as soon as possible]]
  - Infinitely many applications, just kidding :P (But Its true, Indeed !)
- Now try the problems given at the beginning !
- Practice Problems :
  - 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

Problem : You are given a Graph. Find the number of connected components in the Graph.
Hint : DFS or BFS.

Problem : You are given a grid with few cells blocked and others open. You are given a cell , call is **source**, and another cell , call it **dest**. You can move from some cell **u** to some another cell **v** if cell **v** is open and it is adjacent to cell **u.** You have to find the shortest path from **source** to **dest**.
Hint : Try to think the grid as a Graph and apply some shortest path algorithm. Which one ? You think !

Problem : You are given a Tree. You need to find two vertices **u** and **v** such that distance between them maximum. [[http://www.spoj.com/problems/PT07Z/]]
Hint : Try to do it in O(1) number of DFS or BFS !

GREEDY ALGORITHMS

Greedy Algorithms are one of the most intuitive algorithms. Whenever we see a problem we first try to apply some greedy strategy to get the answer(we humans are greedy, aren't we :P ?). Read this tutorial for further insight or you can directly attempt the problems most of the greedy approaches are quite simple and easy to understand/formulate.But many times the proving part might be difficult. But you should always try to prove your greedy approach because most the times it happens that you later realise that you solution does not give the optimal answer.

http://community.topcoder.com/tc?module=Static&d1=tutorials&d2=greedyAlg

They are generally used in optimization problems and there exists an optimal substructure to the problem and solutions are generally O(n log n) (sorting) or O(n) (single pass).

Problems List:

- http://www.spoj.com/problems/BAISED/
- http://www.spoj.com/problems/BALIFE/

- http://www.spoj.com/problems/GCJ101BB/
- http://www.codechef.com/problems/FGFS
- http://www.codechef.com/problems/KNPSK
- http://www.codechef.com/problems/LEMUSIC
- http://www.spoj.com/problems/ARRANGE/
- http://www.spoj.com/problems/FASHION/

Q)A thief breaks into a shop and finds there are **N** items weight of **i**th item is **Wi** and cost of **i**th item is **Ci** and thief has a bag of which can carry at most **W** units of weight. Obviously thief wants to have maximum profit . What strategy he should choose if :

Case 1: If he is allowed to take fractional part of items (like assume item to be a bag of rice and you can take whatever fraction of rice you want). [Hint :: greedy])

Case 2:If he cannot break the items in fractional parts. Will now greedy work ? Try to make some test cases for which greedy will fail.

Most of time when greedy fails its the problem can be solved by Dynamic Programming(DP).

DYNAMIC PROGRAMMING [[DP]]

In my view this is one ***the most important*** topic in competitive programming. The problems are simple and easy to code but hard to master. Practice as many DP problems as much possible.
You must go through this topcoder tutorial and you **must** try to solve all the problems listed below in this doc.

(These are basic problems and some with few variations that we feel one **should** know. You must practice other DP problems too)
Problems list:

- http://www.spoj.com/problems/COINS/
- Read about Maximum Sum Subarray [I dint find exact question on any online judge as its very very basic]
- http://www.codechef.com/problems/DELISH
- http://www.codechef.com/problems/KSUBSUM/
- Q)Finding NCR [Using above discussed recursion in math section and DP]
- https://projecteuler.net/problem=18
- Q)Given a matrix filled with numbers.You are initially at upper left corner , you have to reach to the lower right corner.In each step you can either go right or down.When ever you go to a cell you points increase by value of that cell.What is the maximim possible points you can gain?
- http://www.codechef.com/JUNE13/problems/LEMOUSE
- http://www.spoj.com/problems/MAXWOODS/
- http://www.spoj.com/problems/EDIST/
- http://www.spoj.com/problems/ADFRUITS/
- http://www.spoj.com/problems/IOIPALIN/
- http://www.codechef.com/problems/PPTEST/

- http://www.codechef.com/problems/MAXPR
- http://www.codechef.com/problems/LEBALONS
- http://www.codechef.com/problems/DBOY/
- http://www.codechef.com/problems/HAREJUMP

For further advanced topics you can follow topcoder tutorials.
This also might be helpful introduction to competitive programming - Stanford.

Good Luck !!!
Practice Hard !!
Regards
Abhilash Kumar
Triveni Mahatha
Co-ordinators @ Programming Club , IIT Kanpur