Traps and Loopholes

1. **WRITE NEAT AND CONCISE CODE. READ EDITORIALS AND OTHERs’ CODE. PRACTICE PAPER-AND-PEN CODING.**
2. **A solution must be concise and comprehensible. Avoid use of ‘if’ statements as much as possible. Try writing a generic code. See example codes in the file section which have my as well as the editorial solution, showing a contrast between what we write and what is expected.**
3. In the prob "PT07Z- longest path in a tree", I wrote the correct algo. But I was using 'vector <vector> adjacency' to solve the prob which was giving me a TLE. I used vector adjacency[n] instead (an array of size n which has vector elements), and the problem got accepted. Take care of the type of container you're using.
4. ull division is faster than ll division.
5. REMEMBER INVERSION COUNT.... GENERALLY ITS HARD TO FIGURE OUT THAT THE QUESTION IS A MERE INVERSION COUNT PROBLEM.
6. (A^B)%mod = (A^ (B%mod))%mod.
7. When nothing else works in case of TLE, try scanf,prinf.
8. When working on strings, pass the string by reference and use push\_back and pop\_back in case you need to edit the string instead of creating a new string using call by value. That will cause TLE.
9. Type carefully. 3 times have I done the mistake of writing dp[i][j]==something instead of assignment, and such a mistake while coding is very hard to track. The statement is perfectly fine and is easily missed for an assignment statement.
10. When skeptical about time complexity of dynamic hash maps, reserve the memory beforehand. unordered\_map<int,int>m1; m1.reserve(1000010);
11. **Take special care of loop terminating conditions and checking boundaries in matrices** (try[**https://leetcode.com/problems/minimum-path-sum/**](https://www.facebook.com/l.php?u=https%3A%2F%2Fleetcode.com%2Fproblems%2Fminimum-path-sum%2F&h=-AQGXQy1-&s=1)) . eg. If you want a loop to terminate when all of a, b and c are 0 simultaneously, the condition is ‘if(a || b || c)’ and not ‘if(a && b && c)’.
12. Catalan numbers follow a linear recursion (C(1)=1) C(n+1) = ((4n+2)\*C(n))/(n+2)
13. Circular arrangement of n items numbered 1 to n. Start from a. Move b steps in either direction(+ve or -ve b). The position where you’ll land is given by: ((*a* - 1 + *b*)% *n* + *n*)% *n* + 1
14. Iterative segment tree and lazy propagation: [http://codeforces.com/contest/242/s...](http://codeforces.com/contest/242/submission/9271404)

Never perform (x/y)%M. Always first compute modulo inverse of y then multiply by x.  
z=(x/y)%M;

Instead we should perform

y2=findMMI(y,M);

z=(x\*y2)%M;

1. How to find modulo inverse?  
    This can be done using Fermat’s little theorem:  
   
2. Take care in case of modular subtraction:



1. Nth catalan number

ull n\_choose\_r(ull n, ull r) {

if (n < r)

return 0;

if (r > n/2) {

r = n - r;

}

ull result = 1;

ull common\_divisor;

for (int i = 1; i <= r; ++i) {

common\_divisor = gcd(result, i);

result /= common\_divisor;

result \*= (n - i + 1) / (i / common\_divisor);

}

return result;

}

1. void generate\_ CATALAN() {

catalan[1] = 1;

for (int i = 2; i <= 4000 ; i++) {

for (int j = 1; j <= i - 1; j++) {

catalan[i] = (catalan[i] + ((catalan[j]) \* catalan[i - j]) % mod) % mod;

}

catalan[i] = catalan[i] % mod;

}

for(int i = 1; i <= 4000; i++) {

catalan[i] += catalan[i-1];

catalan[i] %= mod;

}

}