Neaj Morshad's All Notes of Competitive Programming...

**c++ language tricks**

unsigned char is very faster than int / short int

you can use it for numeric input / ouput with the format specifier %hhu

sample unsigned char arr[200000];

for(int i=0; i<n; i++) scanf("%hhu", &arr[i]); printf("%hhu ", arr[i]);

/// works for 0 <=arr[i] <= 255 this code GOT TLE with short int / int declaration see-> https://codeforces.com/contest/911/submission/113563731

Array global declaration is very faster than the local declaration this code GOT TLE with local declaration see-> https://codeforces.com/contest/911/submission/113563731

**<Xor Properties>**

if a ^ b = c

then a^c = b

and b^c = a

**<NUMBER THEORY>**

Euler Totient Properties

1. phi(m\*n) = phi(m) \* phi(n)

2. d occurs as gcd(i, N) exactly phi(N/d) times for each i form 1 to N

3. phi(n) repeats periodically

ex:

numbers that are coprime to 6 ( gcd(i, 6)=1 ) is,

\*1 2 3 4 \*5 6 || \*7 8 9 10 \*11 12 || .......

4. For all d such that d|n , SUM(phi(d)) = n

5. For n > 2, phi(n) is always even.

6. sum of integers that are coprime to n equals to (phi(n) \* n)/2

Euler's Theorem : a^phi(n) % n = 1 if a and n are co-prime.

Fermat's Theorem: a^(p-1) % p = 1 if a and p are co-prime and p is prime.

Existence of Modular Inverse

Modular Inverse of A with respect to M , that is X = A^-1 (mod M) exists if and only if A and M are coprime.

A^phi(M) = 1 (mod M)

A^(phi(M)-1) = A^-1 (mod M) = X = Mod Inverse of A.

if M is prime then phi(M) = M - 1

So A^(M-2) (mod M) = A^-1 (mod M) = X (if and only if M is PRIME)

SUM = lcm(1, n) + lcm(2, n) + ……. + lcm(n, n) = ?

SUM=(n \* (∑d|n(ϕ(d)×d)+1)) / 2

proof : https://forthright48.com/spoj-lcmsum-lcm-sum/

**<COMBINATORICS>**

Catalan Numbers

Ci = {2(2i - 1) \* Ci-1} / (i + 1)

C0 = 1

*Cn*+1​=*C*0​*Cn*​+*C*1​*Cn*−1​+⋯+*Cn*​*C*0​ = *k*=0∑*n*​ *Ck*​*Cn*−*k*​.

<--GRAPH THEORY-->

BFS:

for any node u and v in bfs queue |dis[u]-dis[v]|<=1 (0, 1)

Eulers Characteristics: No of Regions in a 2D plain when connects all possible edges.

Regions, R = (E – V + 2)

E = No of edges. V = No of Vertex = (n + nC4)

for N point Polygon The formula is: E = (n \* (n – 1) + nC4 \* 4) / 2

where (n \* (n – 1)) is for N points that are on the polygon

and (nC4 \* 4) is for inside points that will be created when connects the all edges