PRECODE

```
##### Techniques #####
1. Contribution Technique
2. Binary Search on ans
3. Binary Search on other thing
4. Ternary Search
5. Number Theory
6. DP
7. Segment Tree
8. PBDS
9. Set/map
10. Sieve or Backward Sieve
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
using namespace std;
using namespace __gnu_pbds;
#define TIMER class Timer { private: chrono::time_point <chrono::steady_clock> Begin, End; public: Timer () : Begin(), End (){ Begin = chrono::steady_clock::now(); } ~Timer () { End = chrono::steady_clock::now();cerr << "\nDuration: " << ((chrono::duration <double>)(End - Begin)).count() << "s\n"; } } T;
#define int long long
#define ll unsigned long long
#define uset tree<int, null_type, less_equal<int>, rb_tree_tag, tree_order_statistics_node_update >
cout<<*os.find_by_order(val)<<endl; // k-th element it
less_equal = multiset, less = set, greater_equal = multiset decreasing, greater = set decreaseing
cout<<os.order_of_key(val)<<endl; // strictly smaller or greater
#define fo(i,n) for(int i=0;i<n;i++)
#define Fo(i,k,n) for(int i=k;k<n?i<n:i>n;k<n?i+=1:i-=1)
#define vi vector<int>
#define vii vector<pair<int,int>>
#define pii pair<int,int>
#define pb push_back
#define pf push_front
#define F first
#define S second
#define clr(x,y) memset(x, y, sizeof(x))
#define deb(x) cout << #x << "=" << x << endl
#define deb2(x, y) cout << #x << "=" << x << "," << #y << "=" << y << endl
#define s(x)   x.size()
#define all(x) x.begin(),x.end()
#define allg(x) x.begin(),x.end(),greater<int>()
#define BOOST ios_base::sync_with_stdio(false);cin.tie(NULL);cout.tie(NULL);
#define endl "\n"
#define bitOne(x) __builtin_popcount(x)
#define read freopen("input.txt","r",stdin)
#define write freopen("output.txt","w",stdout)
const int MOD=1000000007;
inline void normal(int &a) { a %= MOD; (a < 0) && (a += MOD); }
inline int modMul(int a, int b) { a %= MOD, b %= MOD; normal(a), normal(b); return (a*b)%MOD; }
inline int modAdd(int a, int b) { a %= MOD, b %= MOD; normal(a), normal(b); return (a+b)%MOD; }
inline int modSub(int a, int b) { a %= MOD, b %= MOD; normal(a), normal(b); a -= b; normal(a); return a; }
inline int modPow(int b, int p) { int r = 1; while(p) { if(p&1) r = modMul(r, b); b = modMul(b, b); p >>= 1; } return r; }
inline int modInverse(int a) { return modPow(a, MOD-2); }
inline int modDiv(int a, int b) { return modMul(a, modInverse(b)); }
mt19937_64 rang(chrono::high_resolution_clock::now().time_since_epoch().count());
int rng(int lim) {
uniform_int_distribution<int> uid(-1000,-1);
return uid(rang);
}
Kth bit on or off
bool checkBit(int n, int k){ if (n & (1 << k)) return true; else return false; }
GCD
int gcd(int a, int b) // O(logN)
{
    if(!b) return a;
    return gcd(b,a%b);
}
Directional Array
int dx[] = {-1, 1, 0, 0,-1,-1, 1,1};
int dy[] = { 0, 0,-1, 1,-1, 1,-1,1};
precalculate factorial
int fact[N];
void preFact(){
fact[0] = 1;
for(int i = 1; i < N; i++){
fact[i] = (1LL*fact[i-1]*i)%mod;
if(fact[i] < 0) fact[i] += mod;
}}
ncr mod
int ncr(int n,int r){
int denom = (fact[n-r] * fact[r] * 1LL)%mod;
int res = (1LL * fact[n] * inverse(denom))%mod;
if(res < 0) res += mod;
return res%mod;
}
```

NUMBER THEORY

```
# SIEVE OF ERATOSTHENES
// TC: O(n*log(log(n)))
const int MX = 1e7+123;
bitset<MX> is_prime;
vector<int> prime;
void primeGen ( int n ){
n += 100;
for ( int i = 3; i <= n; i += 2 )
is_prime[i] = 1;
int sq = sqrt ( n );
for ( int i = 3; i <= sq; i += 2 ) {
if ( is_prime[i] == 1 ) {
for ( int j = i*i; j <= n; j += ( i + i ) ) {
is_prime[j] = 0;
}}}
is_prime[2] = 1;
prime.push_back (2);
for ( int i = 3; i <= n; i += 2 ) {
if ( is_prime[i] == 1 )
prime.push_back ( i );
}}
# SMALLEST PRIME FACTOR
// TC: O(n*log(n))
const int N = 1e6;
vector<int> spf(N);
void smallestPrimeFactor(int n)
{
for(int i = 1; i <= n; i++) spf[i] = i;

for(int i = 2; i*i <= n; i++)
```

```
{
if(spf[i] == i)
{
for(int j = i*i; j <= n; j+=i)
{
if(spf[j] == j)
{
    spf[j] = i;
}}}}}
```

NUMBER OF DIVISORS

```
// pre-requisite: primeGen(n)
// TC : O(sqrt(n))
int NOD(int n){
int ans = 1;
for(auto p : prime){
if(p*p > n) break;
int cnt = 0;
while(n%p == 0){
n/=p;
cnt++;
}
ans *= (cnt+1);
}
if(n > 1) ans *= 2;
return ans;
}
```

SUM OF DIVISORS

```
// ** primeGen(n)
// TC: sqrt(n)
int SOD(int n)
{
int sum = 1;
for(auto p : prime)
{
if(p*p > n) break;
if(n%p == 0)
{
int pn = p;
while(n%p == 0)
{
n/=p;
pn *= p;
}
pn -= 1;
pn/=(p-1);
sum *= pn;
}}
if(n > 1)
{
int pn = n*n;
pn -= 1;
pn /= (n-1);
sum *= pn;
}
return sum;
}
```

SUM OF DIVISORS FROM 1 TO N

```
// TC: O(n)
int SODALL(int n)
{
int ans = 0;
for(int i = 1; i <= n; i++)
{
ans += (n/i)*i;
}
return ans;
}
```

PRIME FACTORIZATION

```
// ** primeGen(n)
// TC : O(sqrt(n))
vector<int> primeFactorization(int n)
{
vector<int> pf;
for(auto x : prime)
{
if(x*x > n) break;

while(n%x == 0)
{
n/=x;
pf.push_back(x);
}
}
if(n > 1) pf.push_back(n);
return pf;
}
```

PHI USING DIV FORMULA

```
const int N = 1e6;
vector<int> phi(N);
// O(n*log(n))
void phiDiv(int n){
phi[0] = phi[1] = 1;
for(int i = 2; i <= n; i++) phi[i] =
i-1;
for(int i = 2; i <= n; i++){
for(int j = i+i; j <= n; j+=i){
phi[j] -= phi[i];
}}}
```

PHI USING SIEVE

```
const int N = 1e6;
vector<int> phi(N);
// O(n*loglog(n))
void phiSieve(int n){
phi[0] = phi[1] = 1;
for(int i = 2; i <= n; i++) phi[i] =
i;
for(int i = 2; i <= n; i++){
if(phi[i] == i){
phi[i]-=1;
for(int j = i+i; j <= n; j+=i){
phi[j] = (phi[j] * (i-1));
phi[j] /= i;
}}}}
```

Linear Sieve

```
const int N = 1e7;
vector<int> lp(N);
vector<int> primes;
// TC: (O(n)) // finds primes up to
1e7
void sieveLinear(int n){
for(int i = 2; i <= n; i++){
if(lp[i] == 0){
lp[i] = i;
primes.push_back(i);
}
for(int j = 0; i * primes[j] <= n;
j++){
lp[i*primes[j]] = primes[j];
if(primes[j] == lp[i]) break;
}}}
```

Divs of N

```
vector<int> divs(int n){
vector<int> v;
for(int i = 1; i*i <= n; i++){
if(n%i == 0){
v.push_back(i);
if(n/i != i){
v.push_back(n/i);
}}}
return v;
}
/// Extended GCD
// O(logN), egcd(a,b).x = (1/a)%b , a
& b must be coprime
struct triplet
{
int x;
int y;
int gcd;
};
triplet egcd(int a, int b)
{
if(b == 0)
{
triplet ans;
ans.x = 1; // must be 1 in base case
ans.y = 1; // y can be anything since
y becomes 0 in gcd(x,y)
ans.gcd = a;
return ans;
```

```
}
triplet ans1 = egcd(b,a%b);
triplet ans;
ans.x = ans1.y; // X is the multiplicative inverse of A under B
ans.y = ans1.x-(a/b)*ans1.y;
ans.gcd = ans1.gcd;
return ans;
}

int ModInv(int a, int m)
{
auto ans = egcd(a,m);
return (ans.x%m + m)%m; // to avoid neg value
// ans.gcd must be 1
}
```

Segmented Sieve

```
vector<char> segmentedSieve(long long L,long long R){
long long lim = sqrt(R);
vector<char> mark(lim + 1, false);
vector<long long> primes;
for (long long i = 2; i <= lim; ++i) {
if (!mark[i]) {
primes.emplace_back(i);
for (long long j=i*i;j<=lim;j+=i)
mark[j]=true;}}
vector<char> isPrime(R - L + 1, true);
for (long long i:primes)
for (long long j=max(i*i, (L+i 1)/i*i);j<=R;j+=i)
isPrime[j-L]=false;
if (L==1) isPrime[0] = false;
return isPrime;}
```

Legendre's Formula:

(N! / p^x) max value of x (p must be prime)

```
int legendre(int n, int p){
int ex = 0;
while(n) {
ex += (n / p);
n /= p;}
return ex;}
```

GEOEMTRIC SUM

```
//**Give a,n will give a^1+a^2+a^3+...+a^n**
const ll MOD=1e9+7;
ll GeoSum(ll a, ll n){
ll sz = 0;ll ret = 0;ll mul = 1;
int MSB = 63 - __builtin_clzll(n);
while(MSB >= 0){
ret = ret * (1 + mul); mul = (mul *mul) % MOD; sz <<= 1;
if( (n >> MSB) & 1) {
mul = (mul *a) % MOD; ret += mul;
sz++;}
ret %= MOD; MSB--;}
return ret;}
```

NCR USING RECURRENCE

```
**ncr using recurrence{nCr = (n-1)Cr + (n-1)C(r-1)}**
void fncr(){
for(int i = 0; i < N; ++i) {
for(int j = 0; j < N; ++j) {
if (j == 0 || j == i) ncr[i][j] = 1;
else ncr[i][j] = ncr[i-1][j-1] +ncr[i-1][j];
}}}
```

NCR%M USING MODULAR MULTIPLICATIVEINVERSE

```
void precal(){
fact[0]=invFact[0]=1;
for(int i=1; i<=N; i++){
fact[i]=((fact[i-1]%MOD)*(i%MOD))%MOD;}
invFact[N]=bigmod(fact[N],MOD-2);
for(int i=N-1; i>=1; i--){
invFact[i]=((invFact[i+1]%MOD)*((i+1)%MOD))%MOD;}}
cout<<((fact[n]*invFact[r])%MOD*invFact[n-r])%MOD<<endl;
//**ncr if it stays in long long**
ll n,r,ans=1;
cin>>n>>r;
for(int i=1; i<=r; i++){
ans=ans*(n-i+1);
ans/=i;}
```

Digit Count of a number:

log10(n) + 1 (log10 for 10 base number)

How many Trailing zeros of n? –

Max power of base which divides n.

Logarithm:

log(ab) = log(a) + log(b)
log(a^x) = xlog(a)

First K digit of n^k:

```
int firstk(int n, int k) {
double a = k * log10(n);
double b = a - floor(a);
double c = pow(10, b);
return floor(c * 100);}
```

Series:

Arithmetic progression:
S(n): (n/2)*(a+p) p is last element
Or, S(n) = (n/2) * (2a + (n-1) d) a is starting element, n is number of elements ,d is common difference
Geometric Progression:
S(n) = (a * (1 - r^n)) / (1-r) r < 1
S(n) = (a * (r^n - 1)) / (r-1) r > 1
* 1^2 + 2^2 +..+ n^2 = (n * (n+1) * (2n+1)) / 6
* 1^3 + 2^3 + … + n^3 = (n^2 * (n+1)^2) / 4

$$* \quad a+ar+ar^2+ar^3+\cdots+ar = \frac{a(r^{k+1}-1)}{r-1}$$

DATA STRUCTURE

Segment Tree

```
const int MX = 1e5+10;
int arr[MX];
int Tree[MX*4];

void init(int node, int b, int e) // O(n) --> 2n nodes
{
if(b==e)
{
Tree[node] = arr[b];
return;
}
int Left = node*2;
int Right = (node*2)+1;
int mid = (b+e)/2;
init(Left,b,mid);
init(Right,mid+1,e);
Tree[node] = Tree[Left] + Tree[Right];
}

int query(int node, int b, int e, int l, int r) // O(4*Log(n))
{
if(l > e || r < b) return 0;
if(l<=b && e<=r)
{
return Tree[node];
}

int Left = node*2;
int Right = (node*2)+1;
int mid = (b+e)/2;
int leftTreeVal = query(Left,b,mid,l,r);
int rightTreeVal = query(Right,mid+1,e,l,r);
return leftTreeVal+rightTreeVal;
}

void update(int node, int b, int e, int i, int val) // O(LogN)
{
if(i > e || i < b) return;
if(b>=i && e<=i)
{
Tree[node] = val;
return;
}

int Left = node*2;
int Right = (node*2)+1;
int mid = (b+e)/2;
update(Left,b,mid,i,val);
update(Right,mid+1,e,i,val);
Tree[node] = Tree[Left] + Tree[Right];
}
```

Segment Tree with Lazy Propagation

```
const int mx = 1e5+10;
```

```cpp
int arr[mx];
struct
{
int sum,prop;
}Tree[mx*4];
void init(int node, int b, int e) //
O(NlogN){
if(b==e){
Tree[node].sum = arr[b];
return;
}
int mid = (b+e)/2;
int left = node*2;
int right = (node*2)+1;
init(left,b,mid);
init(right,mid+1,e);
Tree[node].sum =
Tree[left].sum+Tree[right].sum;
}
void push(int node, int b, int e){
if(b != e){
int mid = (b+e)/2;
int left = node*2;
int right = left+1;
Tree[left].sum += (mid-
b+1)*Tree[node].prop;
Tree[right].sum += (e-
mid)*Tree[node].prop;
Tree[left].prop += Tree[node].prop;
Tree[right].prop += Tree[node].prop;
}
Tree[node].prop = 0;
}
void update(int node,int b, int e,
int l, int r, int val) // O(4*logN){
if(Tree[node].prop != 0){
push(node,b,e);
}
if(l > e || r < b) return;
if(l <= b && r >= e)
{
Tree[node].sum += (val*(e-b+1));
Tree[node].prop += val;
if(Tree[node].prop != 0)
{
push(node,b,e);
}
return;
}
int mid = (b+e)/2;
int left = node*2;
int right = (node*2)+1;
update(left,b,mid,l,r,val);
update(right,mid+1,e,l,r,val);
Tree[node].sum =
Tree[left].sum+Tree[right].sum;
}
int query(int node,int b,int e,int
l,int r) // O(4*logN)
{
if(Tree[node].prop != 0)
{
push(node,b,e);
}
if(l > e || r < b) return 0;
if(l <= b && r >= e)
{
return Tree[node].sum;
}
int mid = (b+e)/2;
int left = node*2;
int right = (node*2)+1;
int val1 = query(left,b,mid,l,r);
int val2 = query(right,mid+1,e,l,r);
return val1 + val2;
}
// Fenwick Tree
vector<int> Bit1; // assign O(n)
space for n elements
vector<int> Bit2; // assign O(n)
space for n elements
void Update(vector<int>& Bit, int
idx, int val) // O(logN) -> single
time update korte logN time
lage...taile N ta items er jonne
O(NlogN) time lagbe
{
int N = Bit.size();
for(idx; idx<N; idx+=(idx&-idx))
{
Bit[idx]+=val;
}
}
int Sum(vector<int>& Bit, int idx) //
O(logN)
{
int sum = 0;
for(idx; idx>0; idx-=(idx&-idx))
{
sum += Bit[idx];
}
return sum;
}
void RangeUpdate(int l, int r, int
val) // O(4*logN)
{
Update(Bit1,l,val);
Update(Bit1,r+1,-val);
Update(Bit2,l,val*(l-1));
Update(Bit2,r+1,-val*r);
}
int PrefixSum(int idx)
{
return Sum(Bit1,idx)*idx -
Sum(Bit2,idx);
}
int RangeSum(int l,int r)
{
return PrefixSum(r)-PrefixSum(l-1);
}
```

GRAPH

```cpp
//DFS cycle detection
bool dfsCycle(int vertex,int parent){
bool a = false;
vis[vertex] = true;
for(auto child : adj[vertex]){
if(child != parent && vis[child]){
return true;
}else if(vis[child] == false){
a = dfsCycle(child,vertex);
}}
return a;
}
```

Topological Sort:

```cpp
const int N = 1e5 + 9;
vector<int> g[N];
vector<int> indeg(N, 0);

int32_t main() {
ios_base::sync_with_stdio(0);
cin.tie(0);

int n, m; cin >> n >> m;
while(m--) {
int u, v; cin >> u >> v;
g[u].push_back(v);
indeg[v]++;
}

queue<int> q;
for(int i = 1; i <= n; i++) {
if(indeg[i] == 0) {
q.push(i);
}}

vector<int> ans;
while(!q.empty()) {
int top = q.front();
q.pop();
ans.push_back(top);
for(auto v: g[top]) {
indeg[v]--;
if(indeg[v] == 0) {
q.push(v);}}}

if(ans.size() == n) {
for(auto i: ans) {
cout << i << ' ';}
cout << '\n';}
else {
```

```
cout << "IMPOSSIBLE\n";}

return 0;
}
//Kruskals Algirithm
#define MX 100005
int parent[MX], R[MX];
struct kruskalStruct{
int u,v,w;
};
static bool cmp(kruskalStruct &a,
kruskalStruct &b){
return a.w < b.w;
}
void init(int v){
for(int i = 0; i <= v; i++){
parent[i] = i;
R[i] = 1;
}}
int Find(int p){
if(p == parent[p]) return p;
return parent[p] = Find(parent[p]);
}
bool Union(int u,int v)
{
int p = Find(u);
int q = Find(v);
if(p != q) {
if(R[p] >= R[q]){
parent[q] = p;
R[p] += R[q];
}
else{
parent[p] = q;
R[q] += R[p];
}
return true;
}
return false;
}
vector<kruskalStruct> store;
void kruskalsMST(){
int vertex,edge;
cin >> vertex >> edge;
init(vertex);
for(int i = 0; i < edge; i++)
{
int u,v,w;
cin >> u >> v >> w;
kruskalStruct ks;
ks.u = u;
ks.v = v;
ks.w = w;
store.push_back(ks);
}
sort(store.begin(),store.end(),cmp);

int totalWeight = 0;
for(int i = 0; i < store.size(); i++)
{
if(Union(store[i].u,store[i].v))
totalWeight += store[i].w;
}
cout << "Kruskal's MST : " <<
totalWeight << endl;
}
```

Tree Diameter: max cost(distance) between 2 nodes. Dfs from any node and get 1 of the 2 nodes. Then dfs again from this node and get another 1. From every node, 1 of these two nodes is the max cost.

```
// Dijkstra
const int N = 1e5+100;
vector<pair<int,int>> adj[N];
int wt[N];
void dijkstra(int source, int nodes)
// TC : O(E + Vlog2(V))
{
for(int i = 0; i < nodes; i++) wt[i]
= 1e18;
wt[source] = 0;
priority_queue<pii,
vector<pair<int,int>>, greater<pii>>
pq;
pq.push({0,source});

while(!pq.empty())
{
int curV = pq.top().S;
int curVW = pq.top().F;
pq.pop();

if(curVW > wt[curV]) continue;

for(auto child : adj[curV])
{
int childV = child.F;
int childVW = child.S;
if(curVW + childVW < wt[childV])
{
wt[childV] = curVW + childVW;
pq.push({wt[childV],childV});
}}}}
// BFS
const int N = 1e3;
bool vis[N];
vector<int> adj[N];
void bfs(int source)
{
vis[source] = true;
queue<int> q;
q.push(source);
while(q.empty() == false)
{
int curVertex = q.front();
q.pop();
for(auto child : adj[curVertex])
{
if(vis[child]) continue;
q.push(child);
vis[child] = true;
}}}
```

STRING

****String Multiply****

```
string multiply(string num1, string
num2){
int len1 = num1.size();
int len2 = num2.size();
if (len1 == 0 || len2 == 0)return
"0";
vector<int> result(len1 + len2, 0);
int i_n1 = 0;
int i_n2 = 0;
for (int i=len1-1; i>=0; i--){
int carry = 0;
int n1 = num1[i] - '0';
i_n2 = 0;
for (int j=len2-1; j>=0; j--){
int n2 = num2[j] - '0';
int sum = n1*n2 + result[i_n1 +i_n2]
+ carry;
carry = sum/10;
result[i_n1 + i_n2] = sum % 10;
i_n2++;
}
if (carry > 0)result[i_n1 + i_n2]+=
carry;
i_n1++;
}
int i = result.size() - 1;
while (i>=0 && result[i] == 0)i--;
if (i == -1)return "0";
string s = "";
while (i >= 0)s
+=std::to_string(result[i--]);
return s;}
```

****String division****

```
string longDivision(string number,
int divisor){
string ans;
int idx = 0;
int temp = number[idx] - '0';
while (temp < divisor)
temp = temp * 10 + (number[++idx] -
'0');
while (number.size() > idx){
ans += (temp / divisor) + '0';
temp = (temp % divisor) * 10 +
number[++idx] - '0';
}
if (ans.length() == 0)return "0";
return ans;
}


// Hashing
const int MAXN=1000006;
```

```
namespace DoubleHash{
int P[2][MAXN];
int H[2][MAXN];
int R[2][MAXN];
int base[2];
int mod[2];
void gen(){
base[0] = 1949313259ll;
base[1] = 1997293877ll;
mod[0]  = 2091573227ll;
mod[1]  = 2117566807ll;
for(int j=0;j<2;j++){
for(int i=0;i<MAXN;i++){
H[j][i]=R[j][i] = 0ll;
P[j][i] = 1ll;
}
}
for(int j=0;j<2;j++){
for(int i=1;i<MAXN;i++){
P[j][i] = (P[j][i-1] *
base[j])%mod[j];
}
}
}
void make_hash(string arr){
int len = arr.size();
for(int j=0;j<2;j++){
for (int i = 1; i <= len; i++)H[j][i]
= (H[j][i - 1] * base[j] + arr[i - 1]
+ 1007) % mod[j];
for (int i = len; i >= 1; i--)R[j][i]
= (R[j][i + 1] * base[j] + arr[i - 1]
+ 1007) % mod[j];
}
}
inline int range_hash(int l,int r,int
idx){
int hashval = H[idx][r + 1] - ((long
long)P[idx][r - l + 1] * H[idx][l] %
mod[idx]);
return (hashval < 0 ? hashval +
mod[idx] : hashval);
}
inline int reverse_hash(int l,int
r,int idx){
int hashval = R[idx][l + 1] - ((long
long)P[idx][r - l + 1] * R[idx][r +
2] % mod[idx]);
return (hashval < 0 ? hashval +
mod[idx] : hashval);
}
inline int range_dhash(int l,int r){
int x = range_hash(l,r,0);
return (x<<32)^range_hash(l,r,1);
}
inline int reverse_dhash(int l,int
r){
int x = reverse_hash(l,r,0);
return (x<<32)^reverse_hash(l,r,1);
}
}
char str1[MAXN];
using namespace DoubleHash;
```

TECHNIQUE

```
// Next Greater Prev Greater
vector<pair<int,int>> ng(n,{-1,-1});
stack<int> stk;
// leff
for(int i = n-1; i >= 0; i--)
{
if(stk.empty())
{
stk.push(i);
}else
{
while(stk.size() && v[i] >
v[stk.top()])
{
ng[stk.top()].first = i;
stk.pop();
}
stk.push(i);
}
}
while(stk.empty() == false)
stk.pop();
// right
for(int i = 0; i < n; i++)
{
if(stk.empty())
{
stk.push(i);
}else
{
while(stk.size() && v[i] >
v[stk.top()])
{
ng[stk.top()].second = i;
stk.pop();
}
stk.push(i);
}
}
// Ternary Search
double ternary_search(double l,
double r) {
double eps = 1e-9;
while (r - l > eps) {
double m1 = l + (r - l) / 3;
double m2 = r - (r - l) / 3;
double f1 = f(m1);
double f2 = f(m2);
if (f1 < f2)
l = m1;
else
r = m2;
}
return f(l);
}
```

2D Prefix Sum:

```
int n, m; cin >> n >> m;
for (int i = 1; i <= n; i++) {
for (int j = 1; j <= m; j++) {
cin >> a[i][j];
}}
for (int i = 1; i <= n; i++) {
for (int j = 1; j <= m; j++) {
prefix[i][j] = prefix[i - 1][j] +
prefix[i][j - 1] - prefix[i - 1][j -
1] + a[i][j];
}}
int q; cin >> q;
while (q--) {
int x1, y1, x2, y2; cin >> x1 >> y1
>> x2 >> y2;
cout << prefix[x2][y2] - prefix[x1 -
1][y2] - prefix[x2][y1 - 1] +
prefix[x1 - 1][y1 - 1] << '\n';
}
```

2D Difference Array:

```
int n, m; cin >> n >> m;
for (int i = 1; i <= n; i++) {
for (int j = 1; j <= m; j++) {
char c; cin >> c;
a[i][j] = c - '0';
}}

int q; cin >> q;
while (q--) {
int x1, y1, x2, y2, x; cin >> x1 >>
y1 >> x2 >> y2;
x = 1;
d[x1][y1] += x;
d[x1][y2 + 1] -= x;
d[x2 + 1][y1] -= x;
d[x2 + 1][y2 + 1] += x;
}

for (int i = 1; i <= n; i++) {
for (int j = 1; j <= m; j++) {
d[i][j] += d[i - 1][j] + d[i][j - 1]
- d[i - 1][j - 1];
}}

// new updated array
for (int i = 1; i <= n; i++) {
for (int j = 1; j <= m; j++) {
cout << (d[i][j] + a[i][j]) % 2;
}
cout << '\n';
}
```

Pigeonhole Principle:

-At least 1 subarray of an array of length N must be divisible by N.
-Build all possible sequences of length 10 whose value is between 1 to 100. At least any two sequences will be same.

* Given an array of length N (N <= 10^6) and M (M <= 10^3) check if there is any subsequence of the array whose sum is divisible by k?
According to the pigeonhole principle if N >= M then it must be "YES". Else we can do DP. where N < M <= 1000.

Contribution Technique (Calculate the contribution of each element separately):

* Sum of pair sums (i=1 to n Σ j= 1 to n Σ(ai+a):
=> Every element will be added 2n times.

$$\sum_{i=1}^{n}(2\times n\times a_i) = 2\times n\times\sum_{i=1}^{n} a_i.$$

* Sum of subarray sums:

$$\sum_{i=1}^{n}(a_i\times i\times(n-i+1)).$$

* Sum of subset sums:

$$\sum_{i=1}^{n}(2^{n-1}\times a_i).$$

* Product of pair product:

$$\prod_{i=1}^{n}(a_i^{2\times n}).$$

* XOR of subarray XORS:
=> How many subarrays does an element have? (i* (n-i+1) times. If subarray length is odd then this element can contribute in total XORs.
* Sum of max-min over all subset:
=> Sort the array. Min = 2^(n-i), Max = 2^(i-1)
i=1 to n Σ(ai * 2^(i-1))-(ai*2^(n-i))
* Sum using Bit:

$$\sum_{k=0}^{30}(cnt_k[1]\times 2^k).$$

* Sum of Pair XORs:
=> XOR = 1 if two bits are different

$$\sum_{k=0}^{30}(cnt_k[0]\times cnt_k[1]\times 2^k).$$

* Sum of pair ANDs:

$$\sum_{k=0}^{30}(cnt_k[1]^2\times 2^k).$$

* Sum of pair ORs:

$$\sum_{k=0}^{30}((cnt_k[1]^2+2\times cnt_k[1]\times cnt_k[0])\times 2^k).$$

* Sum of Subset XORs:

$$\sum_{k=0}^{30}(2^{cnt_k[1]+cnt_k[0]-1}\times 2^k).$$

[where cnt0 != 0)
* Sum of Subset ANDs:

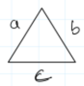
$$\sum_{k=0}^{30}((2^{cnt_k[1]}-1)\times 2^k).$$

* Sub of Subset ORs:

$$\sum_{k=0}^{30}((2^n-2^{cnt_k[0]})\times 2^k).$$

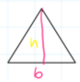
*** Sum of subarray XORs:**
=> Convert to prefix xor, then solve for pairs.

MICELLANEOUS

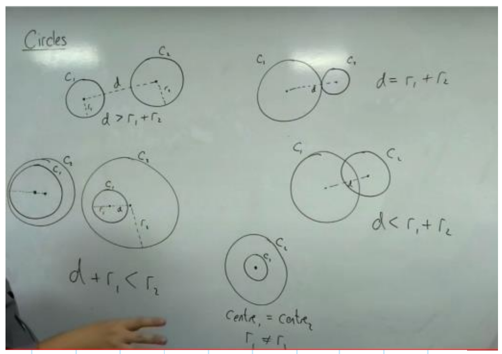


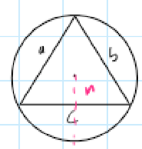
Perimeter $P = a+b+c$
$P = P/2$
Heron's Formula area

$$\boxed{A = \sqrt{P(P-a)\times(P-b)\times(P-c)}}$$



$$\boxed{A = \frac{1}{2}b\times h}$$





circumcircle

Radius $R = \dfrac{a\times b\times c}{4\,\Delta abc}$

area

Formulas:

Sum of squares: 1^2 + 2^2 + 3^2 + … + n^2 = n(n+1) (n+2)/6
Sum of cubes: 1^3 + 2^3 + 3^3+ ... + n^3 = (n^2 * (n+1)^2)/4

Geometric Series: 1+x + x^2+x^3...+x^n = (x^(n+1)-1)/(x-1)
when |x|<1 then the sum = 1 / (1-x)
Harmonic Series: 1 + ½ + 1/3 + ¼ + … + 1/n = ln (n) + O(1)

$$\sum_{k=0}^{n-1}(a_k - a_{k+1}) = a_0 - a_n .$$

$$\sum_{k=1}^{n}(a_k - a_{k-1}) = a_n - a_0 ,$$

$$\sum_{k=1}^{n-1}\frac{1}{k(k+1)} = \sum_{k=1}^{n-1}\left(\frac{1}{k}-\frac{1}{k+1}\right) = 1-\frac{1}{n}.$$

$$\ln n = \log_e n \quad \text{(natural logarithm)} ,$$

For all real $a > 0$, $b > 0$, $c > 0$, and n,

$$a = b^{\log_b a}.$$
$$\log_c(ab) = \log_c a + \log_c b ,$$
$$\log_b a = \frac{\log_c a}{\log_c b},$$
$$\log_b a = \frac{1}{\log_a b},$$
$$\log_b(1/a) = -\log_b a ,$$
$$a^{\log_b c} = c^{\log_b a} ,$$

$$(a+b)^2 = a^2+2ab+b^2$$
$$(a-b)^2 = a^2-2ab+b^2$$
$$a^2+b^2 = (a+b)^2-2ab$$
$$a^2+b^2 = (a-b)^2+2ab$$

$$(a+b)^2 = (a-b)^2+4ab$$
$$(a-b)^2 = (a+b)^2-4ab$$
$$a^2+b^2 = \frac{(a+b)^2+(a-b)^2}{2}$$
$$ab = \left(\frac{a+b}{2}\right)^2-\left(\frac{a-b}{2}\right)^2$$
$$(x+a)(x+b) = x^2+(a+b)x+ab$$
$$2(ab+bc+ac) = (a+b+c)^2-(a^2+b^2+c^2)$$
$$(a+b)^3 = a^3+3a^2b+3ab^2+b^3 = a^3+b^3+3ab(a+b)$$
$$a^3+b^3 = (a+b)^3-3ab(a+b)$$
$$(a-b)^3 = a^3-3a^2b+3ab^2-b^3 = a^3-b^3-3ab(a-b)$$
$$a^3-b^3 = (a-b)^3+3ab(a-b)$$

$$a^3+b^3 = (a+b)(a^2-ab+b^2)$$
$$a^3-b^3 = (a-b)(a^2+ab+b^2)$$
$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2},$$
$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$
$$\sum_{i=1}^{n} i^3 = \frac{n^2(n+1)^2}{4}$$

Geometric series:

$$\sum_{i=0}^{n} c^i = \frac{c^{n+1}-1}{c-1}, \quad c\neq 1$$
$$\sum_{i=0}^{\infty} c^i = \frac{1}{1-c},$$
$$\sum_{i=1}^{\infty} c^i = \frac{c}{1-c}, \quad |c|<1,$$

$$\binom{n}{k} = \frac{n!}{(n-k)!k!},$$
$$\sum_{k=0}^{n}\binom{n}{k} = 2^n.$$
$$\binom{n}{k} = \binom{n}{n-k}$$
$$\binom{n}{k} = \frac{n}{k}\binom{n-1}{k-1}$$
$$\binom{n}{m}\binom{m}{k} = \binom{n}{k}\binom{n-k}{m-k}$$
$$\sum_{k=0}^{n}\binom{r+k}{k} = \binom{r+n+1}{n}.$$
$$\sum_{k=0}^{n}\binom{k}{m} = \binom{n+1}{m+1}$$
$$\sum_{k=0}^{n}\binom{r}{k}\binom{s}{n-k} = \binom{r+s}{n}$$
$$\binom{n}{k} = (-1)^k\binom{k-n-1}{k}$$
$$\left\{{n \atop 1}\right\} = \left\{{n \atop n}\right\} = 1,$$
$$\left\{{n \atop 2}\right\} = 2^{n-1}-1,$$
$$\left\{{n \atop n-1}\right\} = \left[{n \atop n-1}\right] = \binom{n}{2}$$
$$\sum_{k=0}^{n}\left[{n \atop k}\right] = n!,$$
$$C_n = \frac{1}{n+1}\binom{2n}{n}$$
$$\log_b x = \frac{\log_a x}{\log_a b}, \qquad \frac{-b\pm\sqrt{b^2-4ac}}{2a}$$

Euler's number e:

$$e = 1 + \tfrac{1}{2} + \tfrac{1}{6} + \tfrac{1}{24} + \tfrac{1}{120} + \cdots$$

$$\lim_{n\to\infty}\left(1+\frac{x}{n}\right)^n = e^x.$$

$$\left(1+\tfrac{1}{n}\right)^n < e < \left(1+\tfrac{1}{n}\right)^{n+1}.$$

$$\left(1+\tfrac{1}{n}\right)^n = e - \frac{e}{2n} + \frac{11e}{24n^2} - O\left(\frac{1}{n^3}\right).$$

Pascal's Triangle

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
1 8 28 56 70 56 28 8 1
1 9 36 84 126 126 84 36 9 1
1 10 45 120 210 252 210 120 45 10 1

Binomial distribution:

$$\Pr[X=k] = \binom{n}{k} p^k q^{n-k}, \qquad q = 1-p,$$

$$E[X] = \sum_{k=1}^{n} k\binom{n}{k} p^k q^{n-k} = np.$$

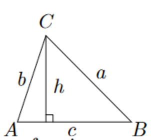
Euler's equation:

$$e^{ix} = \cos x + i\sin x, \qquad e^{i\pi} = -1.$$

Area:

$$A = \tfrac{1}{2}hc, \quad = \tfrac{1}{2}ab\sin C, \quad = \frac{c^2\sin A\sin B}{2\sin C}.$$

Heron's formula:



$$A = \sqrt{s\cdot s_a\cdot s_b\cdot s_c},$$
$$s = \tfrac{1}{2}(a+b+c),$$
$$s_a = s-a,\quad s_b = s-b,\quad s_c = s-c.$$

Law of cosines:

$$c^2 = a^2+b^2-2ab\cos C.$$

$$\sin\tfrac{x}{2} = \sqrt{\frac{1-\cos x}{2}},$$

$$\cos\tfrac{x}{2} = \sqrt{\frac{1+\cos x}{2}},$$

$$\tan\tfrac{x}{2} = \sqrt{\frac{1-\cos x}{1+\cos x}}, \quad = \frac{1-\cos x}{\sin x}, \quad = \frac{\sin x}{1+\cos x},$$

$$\cot\tfrac{x}{2} = \sqrt{\frac{1+\cos x}{1-\cos x}}, \quad = \frac{1+\cos x}{\sin x}, \quad = \frac{\sin x}{1-\cos x},$$

Euler's function: $\phi(x)$ is the number of positive integers less than x relatively prime to x. If $\prod_{i=1}^{n} p_i^{e_i}$ is the prime factorization of x then

$$\phi(x) = \prod_{i=1}^{n} p_i^{e_i-1}(p_i-1).$$

Euler's theorem: If a and b are relatively prime then

$$1 \equiv a^{\phi(b)} \bmod b.$$

Fermat's theorem:

$$1 \equiv a^{p-1} \bmod p.$$

The Euclidean algorithm: if $a > b$ are integers then

$$\gcd(a,b) = \gcd(a \bmod b, b).$$

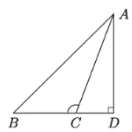
If $\prod_{i=1}^{n} p_i^{e_i}$ is the prime factorization of x then

$$S(x) = \sum_{d|x} d = \prod_{i=1}^{n} \frac{p_i^{e_i+1}-1}{p_i-1}.$$

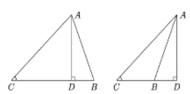
Perfect Numbers: x is an even perfect number iff $x = 2^{n-1}(2^n-1)$ and 2^n-1 is prime.

Wilson's theorem: n is a prime iff

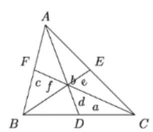
$$(n-1)! \equiv -1 \bmod n.$$



$$AB^2 = AC^2 + BC^2 + 2\cdot BC\cdot CD$$



$$AB^2 = AC^2 + BC^2 - 2\cdot BC\cdot CD$$



$$AB^2 + AC^2 = 2(AD^2 + BD^2)$$

$$d^2 = \frac{2(b^2+c^2)-a^2}{4}$$

$$e^2 = \frac{2(c^2+a^2)-b^2}{4}$$

$$f^2 = \frac{2(a^2+b^2)-c^2}{4}$$

$$x = \frac{-b\pm\sqrt{b^2-4ac}}{2a}$$

$$\log_a M = \log_b M \times \log_a b$$

$$(x+y)^n = x^n + nx^{n-1}y + \frac{n(n-1)}{1\cdot 2}x^{n-2}y^2 + \frac{n(n-1)(n-2)}{1\cdot 2\cdot 3}x^{n-3}y^3 + \cdots + y^n$$

$$n! = n(n-1)(n-2)\cdots 3\cdot 2\cdot 1$$

$$\binom{n}{r} = {}^nC_r, \quad {}^nC_n = 1$$

$$\binom{n}{r} = {}^nC_r = \frac{n!}{r!(n-r)!}, \quad \binom{n}{0} = {}^nC_0 = 1$$

$$\binom{n}{n} = {}^nC_n = 1, \quad 0! = 1$$

$$\binom{n}{k} = \frac{n!}{(n-k)!k!},$$

$$\sum_{k=0}^{n}\binom{n}{k} = 2^n$$

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2},$$

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$

$$\sum_{i=1}^{n} i^3 = \frac{n^2(n+1)^2}{4}$$

Triangle:

area = $\sqrt{s(s-a)(s-b)(s-c)}$

area = $\frac{1}{2}(x_1y_2 + x_2y_3 + x_3y_1 - x_2y_1 - x_3y_2 - x_1y_3)$

Cube:

area = abc

diag = $\sqrt{a^2+b^2+c^2}$

Circle:

circumference : $2\pi r$

area : πr^2

eqn : $(x-h)^2 + (y-k)^2 = r^2$ ... (i)

$(x-h)^2 + (y-k)^2 = r^2$ বা, $x^2 + y^2 - 2hx - 2ky + (h^2+k^2-r^2) = 0$

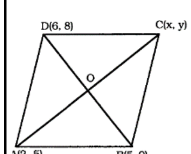
$h^2 + k^2 - r^2 = c$

বা, $g^2 + f^2 - c = r^2$

অতএব ব্যাসার্ধ, $r = \sqrt{g^2+f^2-c}.$

Cone:

volume: $\frac{1}{3}\pi r^2 h$



area = $= \frac{1}{2}(BD\times AC)$

$\therefore P(x_1,y_1)$ বিন্দু হতে $ax+by+c=0$ রেখার লম্ব দৈর্ঘ্য $= \left|\frac{ax_1+by_1+c}{\sqrt{a^2+b^2}}\right|$

$\therefore$ সমান্তরাল রেখা দুইটির মধ্যবর্তী দূরত্ব $MN = ON - OM = \left|\frac{c_1-c_2}{\sqrt{a^2+b^2}}\right|$

$x^2 + y^2 = r^2$ এবং $y = mx + c$

tangent condition: $c = \pm r\sqrt{1+m^2}$

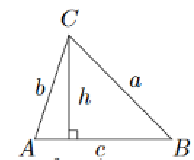
বহিঃস্থ কোন বিন্দু (x_1, y_1) থেকে $x^2+y^2+2gx+2fy+c=0$ বৃত্তের অঙ্কিত স্পর্শক দুইটির সমীকরণ

$$(x^2+y^2+2gx+2fy+c)(x_1^2+y_1^2+2gx_1+2fy_1+c)$$
$$= \{xx_1 + yy_1 + g(x+x_1) + f(y+y_1) + c\}^2$$

Area:

$$A = \tfrac{1}{2}hc, \quad = \tfrac{1}{2}ab\sin C, \quad = \frac{c^2\sin A\sin B}{2\sin C}.$$

Heron's formula:



$$A = \sqrt{s\cdot s_a\cdot s_b\cdot s_c},$$
$$s = \tfrac{1}{2}(a+b+c),$$
$$s_a = s-a,\quad s_b = s-b,\quad s_c = s-c.$$

Law of cosines:

$$c^2 = a^2+b^2-2ab\cos C.$$

Euler's function: $\phi(x)$ is the number of positive integers less than x relatively prime to x. If $\prod_{i=1}^{n} p_i^{e_i}$ is the prime factorization of x then

$$\phi(x) = \prod_{i=1}^{n} p_i^{e_i-1}(p_i-1).$$

Euler's theorem: If a and b are relatively prime then

$$1 \equiv a^{\phi(b)} \bmod b.$$

Fermat's theorem:

$$1 \equiv a^{p-1} \bmod p.$$