**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
11. Do Bruteforce
12. Meet in the middle
#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;
}

bool checkYear(int year) {
if (year % 4 == 0) {
if (year % 100 == 0) {
return year % 400 == 0;
}
return true;
}
return false;
}
```

**MD. TANZIB HOSSAIN**

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
//o(nlog2(n))
vector < vector < int > > factor(1e6 + 5);
void find_factor(int number)
{
for(int factor_index = 1; factor_index <= number; factor_index++)
for(int index = factor_index; index <= number; index += factor_index)
factor[index].push_back(factor_index);
}
//vector < bool > root_flag(1e6 + 5);
vector < int > lowest_root(1e7 + 10);
vector< int > root;
void build_root_table(int limit)
{
```

```
for (int index = 2; index <= limit;
index++)
{
//if(!root_flag[index])
//root.push_back(index);
if (!lowest_root[index])
{
lowest_root[index] = index;
root.push_back(index);
}
for (int root_index = 0; index *
root[root_index] <= limit;
root_index++)
{
lowest_root[index * root[root_index]] =
root[root_index];
//root_flag[index * root[root_index]] =
true;
if(lowest_root[index] ==
root[root_index])
break;
//if(index % root[root_index] == 0)
//break;
}
}
}
bool is_not(uint64_t number, uint64_t
base, uint64_t exponent, int
largest_iteration)
{
uint64_t operated_number =
modular_power(base, exponent, number);
if (operated_number == 1 ||
operated_number == number - 1)
return false;
for (int iteration = 1; iteration <
largest_iteration; iteration++) {
operated_number =
(__uint128_t)operated_number *
operated_number % number;
if (operated_number == number - 1)
return false;
}
return true;
}
bool is_root(uint64_t number)
{
if (number < 2)
return false;
int iteration = 0;
uint64_t exponent = number - 1;
while (~exponent & 1)
{
exponent >>= 1;
iteration++;
}
for (int base : { 2, 3, 5, 7, 11, 13,
17, 19, 23, 29, 31, 37 })
{
if (number == base)
return true;
if (is_not(number, base, exponent,
iteration))
return false;
if(number <= INT_MAX && base == 7)
break;
}
return true;
}

long long
minimized_overflow_modular_multiplicati
on(uint64_t first_multiplier, uint64_t
second_multiplier, uint64_t modulo)
{
long long summand = 0, multiplier =
first_multiplier % modulo;
while (second_multiplier)
{
if (second_multiplier & 1)
summand = (summand + multiplier) %
modulo;
multiplier = multiplier * 2 % modulo;
second_multiplier >>= 1;
}
return summand % modulo;
}
long long rho_factorizer(long long
number)
{
int left_iteration = 0;
int32_t right_iteration = 2;
long long hare = 3, tortoise = 3; //
random seed = 3, other values possible
while (true)
{
left_iteration++;
hare =
(minimized_overflow_modular_multiplicat
ion(hare, hare, number) + number - 1) %
number; // generating function
long long factor = __gcd(abs(tortoise -
hare), number); // the key insight
if (factor != 1 && factor != number)
return factor; // found one non-trivial
factor
if (left_iteration == right_iteration)
{
tortoise = hare;
right_iteration <<= 1;
}
}
}
vector < long long >
root_factorize(long long number)
{
if(number < 2)
return vector < long long > ();
if(is_root(number))
return vector < long long > {number};
vector < long long > first_root_factor,
second_root_factor;
while(number > 1 && number <=
9999991LL)
{
first_root_factor.push_back(lowest_root
[number]);
number /= lowest_root[number];
}
if(number > 9999991LL)
{
long long factorizer =
rho_factorizer(number);
first_root_factor =
root_factorize(factorizer);
second_root_factor =
root_factorize(number / factorizer);
first_root_factor.insert(first_root_fac
tor.end(), second_root_factor.begin(),
second_root_factor.end());
}
return first_root_factor;
}

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
//segment tree

//indexing 0 based, rooting 1 based

struct tree_node{

ll val;

};

struct segment_node

{

//instead change

ll sum;

};

segment_node
merge_segment_node(segment_node
left_segment_index, segment_node
right_segment_index)

{

segment_node merged_segment_node;
```

```
//instead change

merged_segment_node.sum =
left_segment_index.sum +
right_segment_index.sum;

return merged_segment_node;

}

//instead change

segment_node
initialize_segment_node(tree_node
delta){

segment_node initialized_segment_node;

//instead change

initialized_segment_node.sum =
delta.val;

return initialized_segment_node;

}

vector < tree_node > linier_tree;

vector < segment_node > segment_tree;

vector < pair < int, long long > >
lazy_tree;

void build_segment_tree(int root_index,
int left_segment_index, int
right_segment_index)

{

if (left_segment_index ==
right_segment_index)

{

segment_tree[root_index] =
initialize_segment_node(linier_tree[lef
t_segment_index]);

return;

}

int middle_segment_index =
(left_segment_index +
right_segment_index) >> 1;

build_segment_tree(root_index << 1,
left_segment_index,
middle_segment_index);

build_segment_tree(root_index << 1 | 1,
middle_segment_index + 1,
right_segment_index);

segment_tree[root_index] =
merge_segment_node(segment_tree[root_in
dex << 1], segment_tree[root_index << 1
| 1]);

}



void update_point_segment_tree(int
root_index, int left_segment_index, int
right_segment_index, int index,
tree_node delta)

{

if (left_segment_index ==
right_segment_index)

segment_tree[root_index] =
initialize_segment_node(delta);

else

{

int middle_segment_index =
(left_segment_index +
right_segment_index) >> 1;

if (index <= middle_segment_index)

update_point_segment_tree(root_index <<
1, left_segment_index,
middle_segment_index, index, delta);

else

update_point_segment_tree(root_index <<
1 | 1, middle_segment_index + 1,
right_segment_index, index, delta);

segment_tree[root_index] =
merge_segment_node(segment_tree[root_in
dex << 1], segment_tree[root_index << 1
| 1]);

}

}

void propagate_segment_tree(int
root_index, int left_segment_index, int
right_segment_index)

{

if(lazy_tree[root_index].first)

{

auto delta = lazy_tree[root_index];

lazy_tree[root_index] = {0, 0};

if(delta.first == 1)//range increment

segment_tree[root_index].sum +=
delta.second * (right_segment_index -
left_segment_index + 1);//instead
change

else//range replace

segment_tree[root_index].sum =
delta.second * (right_segment_index -
left_segment_index + 1);//instead
change

if(left_segment_index ==
right_segment_index)

return;

if(delta.first == 1)//range increment

{

if(lazy_tree[root_index << 1].first)

lazy_tree[root_index << 1] =
{lazy_tree[root_index << 1].first,
lazy_tree[root_index << 1].second +
delta.second};//instead change

else

lazy_tree[root_index << 1] = delta;

if(lazy_tree[root_index << 1 |
1].first)

lazy_tree[(root_index << 1)+ 1] =
{lazy_tree[root_index << 1 | 1].first,
lazy_tree[root_index << 1 | 1].second +
delta.second};//instead change

else

lazy_tree[root_index << 1 | 1] = delta;
```

```cpp
}

else//range replace

{

lazy_tree[root_index << 1] = delta;

lazy_tree[root_index << 1 | 1] = delta;

}

}

}

void update_range_segment_tree(int root_index, int left_segment_index, int right_segment_index, int left_index, int right_index, int type, long long delta)

{

propagate_segment_tree(root_index, left_segment_index, right_segment_index);

if (left_segment_index > right_index || right_segment_index < left_index)

return;

if(left_segment_index >= left_index && right_segment_index <= right_index)

{

lazy_tree[root_index] = {type, delta};

propagate_segment_tree(root_index, left_segment_index, right_segment_index);

return;

}

int middle_segment_index = (left_segment_index + right_segment_index) >> 1;

update_range_segment_tree(root_index << 1, left_segment_index, middle_segment_index, left_index, middle_segment_index, type, delta);

update_range_segment_tree(root_index << 1 | 1, middle_segment_index + 1, right_segment_index, middle_segment_index + 1, right_index, type, delta);

segment_tree[root_index] = merge_segment_node(segment_tree[root_index << 1], segment_tree[root_index << 1 | 1]);

}

segment_node query_segment_tree(int root_index, int left_segment_index, int right_segment_index, int left_index, int right_index)

{

//instead range update

//propagate_segment_tree(root_index, left_segment_index, right_segment_index);

if (left_segment_index > right_index || right_segment_index < left_index)

return { 0 };//instead change

if(left_segment_index >= left_index && right_segment_index <= right_index)

return segment_tree[root_index];

int middle_segment_index = (left_segment_index + right_segment_index) >> 1;

return merge_segment_node(query_segment_tree(root_index << 1, left_segment_index, middle_segment_index, left_index, right_index), query_segment_tree(root_index << 1 | 1, middle_segment_index + 1, right_segment_index, left_index, right_index));

}
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

////MOBIOUS INVERSION
int mu[sz];

void mu_1_to_n(int n){

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

if(!mr[i]){

for(int j = i; j <= n; j += i){

if(j > i)

mr[j] = true;

if(j % (i * i) == 0)

mu[j] = 0;

else

mu[j] = -mu[j];

}

}

}

}

fill(mu, mu + n + 1, 1);

mu_1_to_n(n);

int c = 0;

for(int i = 1; i <= n; i++)

c += mu[i] * (n / (i * i)) * (n / (i * i));

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

////2D BIT

void upd(int n, int m, vector<vector<int>> &bit, int x, int y, int d) {

for (int i = x; i <= n; i += i & -i)

for (int j = y; j <= m; j += j & -j)

bit[i][j] += d;

}

int tot(vector<vector<int>> &bit, int x, int y) {

int ret = 0;

for (int i = x; i > 0; i -= i & -i)
```

```
for (int j = y; j > 0; j -= j & -j)
ret += bit[i][j];
return ret;
}
void solve(){
int n,q;
cin>>n>>q;
int a[n+3][n+3];
for(int i=1;i<=n;i++)
for(int j=1;j<=n;j++){
char c;
cin>>c;
c=='.'?a[i][j]=0:a[i][j]=1;
}
vector<vector<int>>FT(n+3,vector<int>(n+3));
for(int i=1;i<=n;i++)
for(int j=1;j<=n;j++)
upd(n,n,FT,i,j,a[i][j]-0);
while(q--){
int t;
cin>>t;
if(t==1){
int x,y;
cin>>x>>y;
upd(n,n,FT,x,y,1-a[x][y]-a[x][y]);
a[x][y]=1-a[x][y];
}
else{
int x1,y1,x2,y2;
cin>>x1>>y1>>x2>>y2;
cout<<tot(FT,x2,y2)-tot(FT,x1-1,y2)-tot(FT,x2,y1-1)+tot(FT,x1-1,y1-1)<<endl;
}
}
}
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
///HLD
void dfs(int s, int p) {
dp[s] = 1;
int mx = 0;
for (auto i: adj[s]) if (i != p) {
par[i] = s;
depth[i] = depth[s] + 1;
dfs(i, s);
dp[s] += dp[i];
if (dp[i] > mx)
mx = dp[i], heavy[s] = i;
}
}

int cnt = 0;
void hld(int s, int h) {
head[s] = h;
id[s] = ++cnt;
update(id[s]-1, val[s]);
if (heavy[s])
hld(heavy[s], h);
for (auto i: adj[s]) {
if (i != par[s] && i != heavy[s])
hld(i, i);
}
}
int path(int x, int y){
int ans = 0;
while (head[x] != head[y]) {
if (depth[head[x]] > depth[head[y]])
swap(x, y);
ans = max(ans, query(id[head[y]]-1, id[y]-1));
y = par[head[y]];
}
if(depth[x] > depth[y])
swap(x, y);
ans = max(ans, query(id[x]-1, id[y]-1));
return ans;
}
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
const int N = 2E6;
long long first_base = 137, second_base = 277, MOD1 = 127657753, MOD2 = 987654319;
pair < long long, long long > base_power[N], inversed_base_power[N];
void precalculate_hash_power()
{
base_power[0] = {1, 1};
for (int i = 1; i < N; i++)
{
base_power[i].first = modular_multiplication(base_power[i - 1].first, first_base, MOD1);
base_power[i].second = modular_multiplication(base_power[i - 1].second, second_base, MOD2);
}
```

```
long long inversed_first_base =
inverse_modular_power(first_base,
MOD1), inversed_second_base =
inverse_modular_power(second_base,
MOD2);
inversed_base_power[0] = {1, 1};
for (int i = 1; i < N; i++)
{
inversed_base_power[i].first =
modular_multiplication(inversed_base_po
wer[i - 1].first, inversed_first_base,
MOD1);
inversed_base_power[i].second =
modular_multiplication(inversed_base_po
wer[i - 1].second,
inversed_second_base, MOD2);
}
}

struct rolling_hash
{
int n;
string str;
vector < pair < long long, long long >
> forward_hash;
vector < pair < long long, long long >
> backward_hash;
rolling_hash() {}
rolling_hash(string &s)
{
str = s;
n = str.size();
forward_hash.emplace_back(0, 0);
for (int forward_index = 0;
forward_index < n; forward_index++)
{
pair < long long, long long >
hashed_value;
hashed_value.first =
modular_addition(forward_hash[forward_i
ndex].first, modular_multiplication(1ll
* str[forward_index],
base_power[forward_index].first, MOD1),
MOD1);
hashed_value.second =
modular_addition(forward_hash[forward_i
ndex].second,
modular_multiplication(1ll *
str[forward_index],
base_power[forward_index].second,
MOD2), MOD2);
forward_hash.push_back(hashed_value);
}

backward_hash.emplace_back(0, 0);
for (int forward_index = 0,
backward_index = n - 1; forward_index <
n; forward_index++, backward_index--)
{
pair < long long, long long >
hashed_value;
hashed_value.first =
modular_addition(backward_hash[forward_
index].first,
modular_multiplication(1ll *
str[forward_index],
base_power[backward_index].first,
MOD1), MOD1);
hashed_value.second =
modular_addition(backward_hash[forward_
index].second,
modular_multiplication(1ll *
str[forward_index],
base_power[backward_index].second,
MOD2), MOD2);
backward_hash.push_back(hashed_value);
}
}

pair < long long, long long >
forward_substring_hash(int left_index,
int right_index)// 1 indexed
{
pair < long long, long long >
hashed_value;
hashed_value.first =
modular_multiplication(modular_subtract
ion(forward_hash[right_index].first,
forward_hash[left_index - 1].first,
MOD1), inversed_base_power[left_index -
1].first, MOD1);
hashed_value.second =
modular_multiplication(modular_subtract
ion(forward_hash[right_index].second,
forward_hash[left_index - 1].second,
MOD2), inversed_base_power[left_index -
1].second, MOD2);
return hashed_value;
}
pair<long long, long long>
backward_substring_hash(int left_index,
int right_index)// 1 indexed
{
pair < long long, long long >
hashed_value;
hashed_value.first =
modular_multiplication(modular_subtract
ion(backward_hash[right_index].first,
backward_hash[left_index - 1].first,
MOD1), inversed_base_power[n -
right_index].first, MOD1);
hashed_value.second =
modular_multiplication(modular_subtract
ion(backward_hash[right_index].second,
backward_hash[left_index - 1].second,
MOD2), inversed_base_power[n -
right_index].second, MOD2);
return hashed_value;
}
bool is_palindrome(int left_index, int
right_index)
{
return
forward_substring_hash(left_index,
right_index) ==
backward_substring_hash(left_index,
right_index);
}
};

struct manachar
{
int n;
string str = "";
vector < int > longest_palindrome;
manachar() {}
manachar(string &s)
{
for(auto chracter: s)
str += string("#") + chracter;
str += string("#");

n = str.size();
longest_palindrome.clear();
longest_palindrome.resize(n, 1);
for(int index = 1, left_index = 1,
right_index = 1; index < n; index++)
{
longest_palindrome[index] = max(0,
min(right_index - index,
longest_palindrome[left_index +
(right_index - index)]));
while(index - longest_palindrome[index]
>= 0  && index +
longest_palindrome[index] < n &&
str[index - longest_palindrome[index]]
== str[index +
longest_palindrome[index]])
longest_palindrome[index]++;
if(index + longest_palindrome[index] >
right_index)
left_index = index -
longest_palindrome[index], right_index
= index + longest_palindrome[index];
longest_palindrome[index]--;
}
}
int retrive_longest_palindrome(int
center, bool parity)
{
return longest_palindrome[2 * center +
1 + parity];
}
```

```cpp
bool is_palindrome(int left_index, int right_index)
{
return right_index - left_index + 1 <= retrive_longest_palindrome((left_index + right_index) / 2, (right_index - left_index + 1) & 1);
}
};
```

## NUMBER THEORY

### # SIEVE OF ERATOSTHENES

```cpp
// 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

```cpp
// 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

```cpp
// 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

```cpp
// ** 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

```cpp
// 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

```cpp
// ** 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

```cpp
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

```cpp
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;
}}}}
```

### # Sum of all numbers that are co-prime with N

$$\frac{\phi(n)}{2} n.$$

```
# 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+i1)/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
vector<int> fact(1e5+10);
vector<int> invFact(1e5+10);
vector<int> invNum(1e5+10);
void gen()
{
fact[0] = 1;
for(int i = 1; i <= 1e5; i++ )
{
fact[i] = modMul(i,fact[i-1]);
invNum[i] = modInverse(i);
}
invFact[1] = invNum[1];
for (int i = 2; i <= 1e5; i++)
{
invFact[i] = modMul(invFact[i-
1],invNum[i]);
}}

//**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
```

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

```
DATA STRUCTURE
Sparse Table
const int N = 1e5+100;
int t[100][N];
```

```
int it[100][N];
int Log2[N];
int n;
int p;
void init()
{
for(int i = 2; i <= n; i++)  // storing log values
{
Log2[i] = Log2[i/2]+1;
}
p = Log2[n];
for(int i = 0; i < n; i++) it[0][i] = i; // init idx
for(int i = 1; i <= p; i++)  // for max
{
for(int j = 0; j+(1<<i) <= n; j++)
{
int left = t[i-1][j];
int right = t[i-1][j+(1<<(i-1))];
t[i][j] = max(left,right);
if(left >= right)
{
it[i][j] = it[i-1][j];
}else
{
it[i][j] = it[i-1][j+(1<<i-1)];
}}}}
int idx = -1;
int query(int l, int r){ // TC: O(1)
int len = r-l+1;
int p = Log2[len];
int left = t[p][l];
int right = t[p][r-(1<<p)+1];
if(left >= right){
idx = it[p][l];
}else{
idx = it[p][r-(1<<p)+1];
}
return max(left,right);
}
int overlapQuery(int l, int r)
{
int mx = INT_MIN;
for(int p = Log2[r-l+1]; l <= r; p = Log2[r-l+1])
{
mx = max(mx,t[p][l]);
l += (1<<p);
}
return mx;
}
int main()
{
cin >> n;
for(int i = 0; i < n; i++)
{
cin >> t[0][i];
}
int q;
cin >> q;
init();
while(q--)
{
int l, r;
cin >> l >> r;
int val = query(l,r);
cout << "idx : " << idx << endl;
cout << "val : " << val << endl;
cout << "overlap : " << overlapQuery(l,r) << endl;
}}
/*
13
4 2 3 7 1 5 3 3 9 6 7 -1 4
100
*/
```

*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;
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**

```
//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:**

```
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 Pair
const int N = 2e5 + 9;
int power(long long n, long long k,
const int mod) {
int ans = 1 % mod;
n %= mod;
if (n < 0) n += mod;
while (k) {
if (k & 1) ans = (long long) ans * n %
mod;
n = (long long) n * n % mod;
k >>= 1;
}
return ans;
}
const int MOD1 = 127657753, MOD2 =
987654319;
const int p1 = 137, p2 = 277; // change
here
int ip1, ip2;
pair<int, int> pw[N], ipw[N];
void prec() {
pw[0] = {1, 1};
for (int i = 1; i < N; i++) {
pw[i].first = 1ll * pw[i - 1].first *
p1 % MOD1;
pw[i].second = 1ll * pw[i - 1].second *
p2 % MOD2;
}
ip1 = power(p1, MOD1 - 2, MOD1);
```

```
ip2 = power(p2, MOD2 - 2, MOD2);
ipw[0] = {1, 1};
for (int i = 1; i < N; i++) {
ipw[i].first = 1ll * ipw[i - 1].first *
ip1 % MOD1;
ipw[i].second = 1ll * ipw[i - 1].second
* ip2 % MOD2;
}
}
struct Hashing {
int n;
string s;
vector<pair<int, int>> hash_val;
Hashing() {}
Hashing(string _s) {
s = _s;
n = s.size();
hash_val.emplace_back(0, 0);
for (int i = 0; i < n; i++) {
pair<int, int> p;
p.first = (hash_val[i].first + 1ll *
s[i] * pw[i].first % MOD1) % MOD1;
p.second = (hash_val[i].second + 1ll *
s[i] * pw[i].second % MOD2) % MOD2;
hash_val.push_back(p);
}
}

pair<int, int> get_hash(int l, int
r) { // 1 indexed
pair<int, int> ans;
ans.first = (hash_val[r].first -
hash_val[l - 1].first + MOD1) * 1ll
* ipw[l - 1].first % MOD1;
ans.second = (hash_val[r].second -
hash_val[l - 1].second + MOD2) *
1ll * ipw[l - 1].second % MOD2;
return ans;
}
pair<int, int> get_hash() { // 1
indexed
return get_hash(1, n);
}
};
```

```
// 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;
```

```
// Trie Max Subarray XOR
const int N = 2;
struct node{
node* arr[N];
};
node* getNode()
{
node* root = new node();
root->arr[0] = NULL;
root->arr[1] = NULL;
return root;
}

void insert(node* root, int n)
{
node *tempRoot = root;
for(int i = 31; i >= 0; i--)
{
int index = ((n >> i) & 1);
if(tempRoot->arr[index] == NULL)
{
tempRoot->arr[index] = getNode();
}
tempRoot = tempRoot->arr[index];
}
}

int search(node* root, int n)
{
node* tempRoot = root;
int res = 0;
for(int i = 31; i >= 0; i--)
{
int index = ((n>>i)&1);
if(tempRoot->arr[index^1])
{
res += (1 << i);
tempRoot = tempRoot->arr[index^1];
}else
{
tempRoot = tempRoot->arr[index];
}
}
return res;
}

void deleteTrie(node *root)
{
for(int i = 0; i < N; i++)
{
if(root->arr[i])
{
deleteTrie(root->arr[i]);
}
}
delete root;
}
void solve()
{
node* root = getNode();
insert(root,0);
int n;
cin >> n;
vector<int> v(n);
int pxor = 0;
int mxor = 0;
for(int i = 0; i < n; i++)
```

```
{
cin >> v[i];
pxor ^= v[i];
mxor = max(mxor,pxor);
mxor = max(mxor,search(root,pxor));
insert(root,pxor);
}
deleteTrie(root);
cout << mxor << endl;
}
// Max xor of 2d grid subgrid
#include <bits/stdc++.h>
using namespace std;
#define endl "\n"
int trie[10001*28][2],node;
void insert(int n)
{
int root = 1;
for(int i = 27; i >= 0; i--)
{
int idx = (1 & (n >> i));
if(!trie[root][idx])
{
trie[root][idx] = node++;
}
root = trie[root][idx];
}
}
int search(int n)
{
int root = 1;
int res = 0;
for(int i = 27; i >= 0; i--)
{
int idx = (1 & (n >> i));
if(trie[root][idx^1])
{
res += (1 << i);
root = trie[root][idx^1];
}else
{
root = trie[root][idx];
}
}
return res;
}

void solve()
{
int n, m;
cin >> n >> m;
int v[n + 5][m + 5];
for (int i = 0; i <= n; i++)
{
v[i][0] = 0;
}

for (int i = 1; i <= n; i++)
{
for (int j = 1; j <= m; j++)
{
cin >> v[i][j];
v[i][j] ^= v[i][j - 1];
}
}
int mxor = 0;
for (int l = 1; l <= m; l++){
for (int r = l; r <= m; r++){
memset(trie,0,sizeof trie);
node = 2;
int rowXor = 0;
insert(0);
for (int i = 1; i <= n; i++)
{
rowXor = (rowXor ^ v[i][r] ^ v[i][l -
1]);
int k = search(rowXor);
mxor = max(mxor, rowXor);
mxor = max(mxor, k);
insert(rowXor);
}}}
cout << mxor << endl;
}
int main(){
solve();
}
```

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:

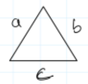
$$\sum_{k=0}^{30}((2^{cnt_k[1]}-1)\times 2^k).$$

* Sub of Subset ORs:

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

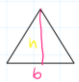
*** Sum of subarray XORs:**
=> Convert to prefix xor, then solve for pairs.

**MICELLANEOUS**

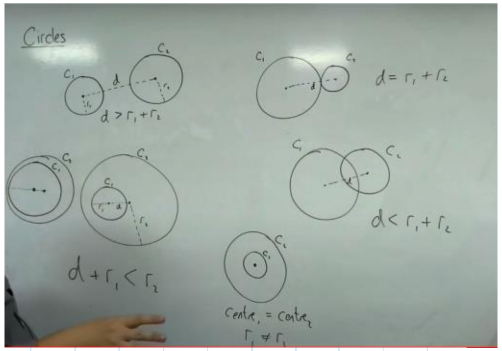


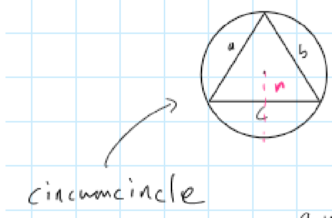

Perimeter $P = a+b+c$
$p = P/2$
Heron's Formula area

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




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




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)

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

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

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

>

$\ln n = \log_e n$ (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 |

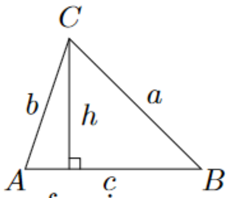
Binomial distribution:

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

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

Euler's equation:

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




Law of cosines:

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

Area:

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

Heron's formula:

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

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

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

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

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

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

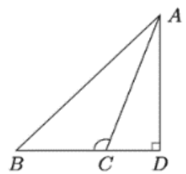
$$\gcd(a, b) = \gcd(a \bmod b, b).$$

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

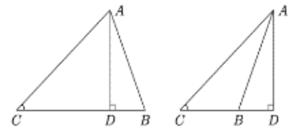
$$S(x) = \sum_{d|x} d = \prod_{i=1}^{n} \frac{p_i^{e_i+1} - 1}{p_i - 1}.$$

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

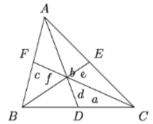
$$(n-1)! \equiv -1 \bmod n.$$




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




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




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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

**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 :** $\pi r^2$

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

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

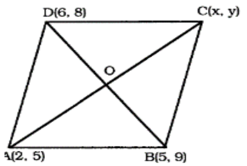
$h^2 + k^2 - r^2 = c$

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 $MN = ON - OM$

$= \left|\frac{c_1 - c_2}{\sqrt{a^2+b^2}}\right|$

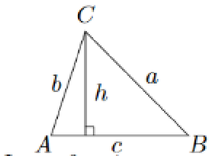
$x^2 + y^2 = r^2$ এবং $y = mx + c$

**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:

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

Heron's formula:

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

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:

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

****G_MATH & H_MATH[9-10]****

---

**Symmetric difference of two sets:** is denoted by $A \Delta B = (A - B) \cup (B - A)$

$= (A \cup B) - (A \cap B)$

**De-Morgan's laws:**

i. $(A \cup B)' = A' \cap B'$

ii. $(A \cap B)' = A' \cup B'$

iii. $A - (B \cap C) = (A - B) \cup (A - C)$

iv. $A - (B \cup C) = (A - B) \cap (A - C)$

**If A, B and C are any three sets, then**

i. $A \cap (B - C) = (A \cap B) - (A \cap C)$

ii. $A \cap (B \Delta C) = (A \cap B) \Delta (A \cap C)$

iii. $P(A) \cap P(B) = P(A \cap B)$

iv. $P(A) \cup P(B) = P(A \cup B)$

v. If $P(A) = P(B) \Rightarrow A = B$

where, $P(A)$ is the power set of A.

$A \subseteq A \cup B, B \subseteq A \cup B, A \cup B \subseteq A,$

$A \cap B \subseteq B$

$A - B = A \cap B'$, $B - A = B \cap A'$

$(A - B) \cap B = \phi$

$(A - B) \cup B = A \cup B$

$A \subseteq B \Leftrightarrow B' \subseteq A'$

$A - B = B' - A'$

$(A \cup B) \cap (A \cup B') = A$

$A \cup B = (A - B) \cup (B - A) \cup (A \cap B)$

$A - (A - B) = A \cap B$

$A - B = B - A \Leftrightarrow A = B$ and

$A \cup B = A \cap B \Rightarrow A = B$

**Results on cardinal number of some sets:**

If A, B and C are finite sets and U be the universal set, then

i. $n(A \cup B) = n(A) + n(B)$ if A and B are disjoint sets.

ii. $n(A \cup B) = n(A) + n(B) - n(A \cap B)$

iii. $n(A \cup B) = n(A - B) + n(B - A) + n(A \cap B)$

## Algebraic Formulae

$a^3 + b^3 + c^3 - 3abc = (a+b+c)(a^2 + b^2 + c^2 - ab - bc - ca)$

$a^3 + b^3 + c^3 - 3abc = \frac{1}{2}(a+b+c)\{(a-b)^2 + (b-c)^2 + (c-a)^2\}$

$(a-b)^3 + (b-c)^3 + (c-a)^3 = 3(a-b)(b-c)(c-a)$

Corollary 6. if $a + b + c = 0$, then $a^3 + b^3 + c^3 = 3abc$

Corollary 7. if $a^3 + b^3 + c^3 = 3abc$, so $a + b + c = 0$ or $a = b = c$

**Corollary 1.** $a^2 + b^2 = (a+b)^2 - 2ab$

**Corollary 2.** $a^2 + b^2 = (a-b)^2 + 2ab$

**Corollary 3.** $(a+b)^2 = (a-b)^2 + 4ab$

**Corollary 4.** $(a-b)^2 = (a+b)^2 - 4ab$

**Corollary 5.** $a^2 + b^2 = \frac{(a+b)^2 + (a-b)^2}{2}$

**Corollary 6.** $ab = \left(\frac{a+b}{2}\right)^2 - \left(\frac{a-b}{2}\right)^2$

**Formula 3.** $a^2 - b^2 = (a+b)(a-b)$

**Formula 4.** $(x+a)(x+b) = x^2 + (a+b)x + ab$

**Formula 5.** $(a+b+c)^2 = a^2 + b^2 + c^2 + 2ab + 2bc + 2ac$

**Corollary 7.** $a^2 + b^2 + c^2 = (a+b+c)^2 - 2(ab + bc + ac)$

**Corollary 8.** $2(ab + bc + ac) = (a+b+c)^2 - (a^2 + b^2 + c^2)$

**Formulae of Cubes**

**Formula 6.** $(a+b)^3 = a^3 + 3a^2b + 3ab^2 + b^3 = a^3 + b^3 + 3ab(a+b)$

**Corollary 9.** $a^3 + b^3 = (a+b)^3 - 3ab(a+b)$

**Formula 7.** $(a-b)^3 = a^3 - 3a^2b + 3ab^2 - b^3 = a^3 - b^3 - 3ab(a-b)$

**Corollary 10.** $a^3 - b^3 = (a-b)^3 + 3ab(a-b)$

**Formula 8.** $a^3 + b^3 = (a+b)(a^2 - ab + b^2)$

**Formula 9.** $a^3 - b^3 = (a-b)(a^2 + ab + b^2)$

## Binomial Expansion

| Value of $n$ | | Pascal's Triangle | Number of Terms |
|---|---|---|---|
| $n=0$ | $(1+y)^0 =$ | $1$ | 1 |
| $n=1$ | $(1+y)^1 =$ | $1+y$ | 2 |
| $n=2$ | $(1+y)^2 =$ | $1+2y+y^2$ | 3 |
| $n=3$ | $(1+y)^3 =$ | $1+3y+3y^2+y^3$ | 4 |
| $n=4$ | $(1+y)^4 =$ | $1+4y+6y^2+4y^3+y^4$ | 5 |
| $n=5$ | $(1+y)^5 =$ | $1+5y+10y^2+10y^3+5y^4+y^5$ | 6 |

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

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

## Logarithms

1) $\log_a b = x$ if and only If $a^x = b$.

2) $\log_a(a^x) = x$

3) $a^{\log_a b} = b$

(i) If $x > 0$, $y > 0$ and $a \neq 1$ then $x = y$ if and only if $\log_a x = \log_a y$

(ii) If $a > 1$ and $x > 1$ then $\log_a x > 0$

(iii) If $0 < a < 1$ and $0 < x < 1$ then $\log_a x > 0$

(iv) If $a > 1$ and $0 < x < 1$ then $\log_a x < 0$

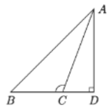
**Formula 10** (change of base). $\log_a M = \log_b M \times \log_a b$

**Corollary 1.** $\log_a b = \frac{1}{\log_b a}$ or $\log_b a = \frac{1}{\log_a b}$

## Lines, Angles and Triangles

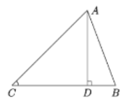
for obtuse angle C

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




For acute angle c

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



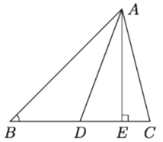
1) If $\angle ACB$ is an obtuse angle,, $AB^2 > AC^2 + BC^2$

2) If $\angle ACB$ is a right angle, $AB^2 = AC^2 + BC^2$

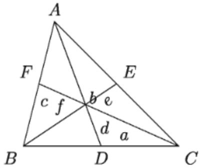
3) If $\angle ACB$ is an acute angle, $AB^2 < AC^2 + BC^2$

**Theorem 5** (Theorem of Apollonius). The sum of the areas of the squares drawn on any two sides of a triangle is equal to twice the sum of area of the squares drawn on the median of the third side and on either half of that side.

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



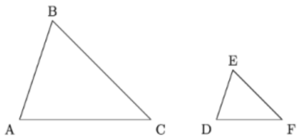
Let, length of the sides of $BC$, $CA$ and $AB$ of the $\triangle ABC$ are $a$, $b$ and $c$ respectively. $AD$, $BE$ and $CF$ are the medians drawn on sides $BC$, $CA$ and $AB$ and their lengths are $d$, $e$ and $f$ respectively.



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

Similarly we can get, $e^2 = \frac{2(c^2+a^2) - b^2}{4}$ and $f^2 = \frac{2(a^2+b^2) - c^2}{4}$

$\therefore 3(a^2 + b^2 + c^2) = 4(d^2 + e^2 + f^2)$



**Theorem 9.** The ratio of the areas of the two similar triangles is equal to the ratio of the areas of the squares drawn on their two corresponding sides.

$$\frac{\triangle ABC}{\triangle DEF} = \frac{AB^2}{DE^2} = \frac{AC^2}{DF^2}$$

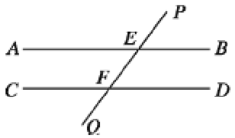
**Circumcenter of a Triangle**: The **circumcenter** of a triangle is the point of intersection of two perpendicular bisectors of that triangle. Noted that, the perpendicular bisector of the third side of the triangle would pass through the circumcenter too.

**Centroid of a Triangle**: The **centroid** of a triangle is the point of intersection of three medians of that triangle. The centroid of a triangle divides each median

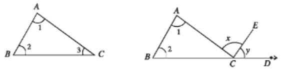
**Orthocenter of a Triangle**: The **orthocenter** of a triangle is the point of intersection of the perpendiculars drawn from each vertex to their respective opposite side.

**Theorem 3.** When a transversal cuts two parallel straight lines,

1) the pair of corresponding angles are equal.
2) the pair of alternate angles are equal.
3) that pair of interior angles on the same side of the transversal are supplementary.



**Theorem 5.** The sum of the three angles of a triangle is equal to two right angles.



$\therefore \angle ABC + \angle BAC = \angle ECD + \angle ACE = \angle ACD$

$\angle ABC + \angle BAC + \angle ACB = \angle ECD + \angle ACE + \angle ACB = \angle ACD + \angle ACB = 2$ right angles.

**Corollary 2.** If a side of a triangle is produced then exterior angle so formed is equal to the sum of the two opposite interior angles.

**Corollary 3.** If a side of a triangle is produced, the exterior angle so formed is greater than each of the two interior opposite angles.

**Corollary 4.** The acute angles of a right angled triangle are complementary to each other.

**Theorem 12.** If one side of a triangle is greater than another, the angle opposite the greater side is greater than the angle opposite the lesser sides.

Let, in triangle $\triangle ABC$, $AC > AB$. Therefore $\angle ABC > \angle ACB$

**Corollary 5.** The difference of the lengths of any two sides of a triangle is smaller than the third side.

**Theorem 15.** The line segment joining the mid-points of any two sides of a triangle is parallel to the third side and in length it is half.

# Circle

**Theorem 17.** The line segment drawn from the centre of a circle to bisect a chord other than diameter is perpendicular to the chord.

**Theorem 19.** Chords equidistant from the centre of a circle are equal.

**Theorem 20.** The angle subtended by the same arc at the centre is double of the angle subtended by it at any point on the remaining part of the circle.

**Theorem 21.** Angles in a circle standing on the same arc are equal.

**Theorem 22.** The angle inscribed in the semi- circle is a right angle.

**Corollary 4.** The circle drawn with hypotenuse of a right-angled triangle as diameter passes through the vertices of the triangle.

**Corollary 5.** The angle inscribed in the major arc of a circle is an acute angle.

**Theorem 23.** The sum of the two opposite angles of a quadrilateral inscribed in a circle is two right angles.

**Corollary 6.** If one side of a cyclic quadrilateral is extended, the exterior angle formed is equal to the opposite interior angle.

**Corollary 7.** A parallelogram inscribed in a circle is a rectangle.

**Theorem 24.** If two opposite angles of a quadrilateral are supplementary, the four vertices of the quadrilateral are concyclic.

# Trigonometric Ratio

**Proposition 5.** Any arc of length $s$ produces an angle $\theta$ in the centre of the circle of radius $r$ then $s = r\theta$.

**Proposition 6.** $1^\circ = \left(\frac{\pi}{180}\right)^c$ and $1^c = \left(\frac{180}{\pi}\right)^\circ$

$(i)$ $1^\circ = \left(\frac{\pi}{180}\right)^c$

$(ii)$ $30^\circ = \left(30 \times \frac{\pi}{180}\right)^c = \left(\frac{\pi}{6}\right)^c$

$(iii)$ $45^\circ = \left(45 \times \frac{\pi}{180}\right)^c = \left(\frac{\pi}{4}\right)^c$

$(iv)$ $60^\circ = \left(60 \times \frac{\pi}{180}\right)^c = \left(\frac{\pi}{3}\right)^c$

168

$(v)$ $90^\circ = \left(90 \times \frac{\pi}{180}\right)^c = \left(\frac{\pi}{2}\right)^c$

$(vi)$ $180^\circ = \left(180 \times \frac{\pi}{180}\right)^c = \pi^c$

$(vii)$ $360^\circ = \left(360 \times \frac{\pi}{180}\right)^c = (2\pi)^c$

$\therefore \sin(2\pi - \theta) = \sin(-\theta) = -\sin\theta$, $\cos(2\pi - \theta) = \cos(-\theta) = \cos\theta$

$\tan(2\pi - \theta) = \tan(-\theta) = -\tan\theta$, $\text{cosec}(2\pi - \theta) = \text{cosec}(-\theta) = -\text{cosec}\theta$

$\sec(2\pi - \theta) = \sec(-\theta) = \sec\theta$ and $\cot(2\pi - \theta) = \cot(-\theta) = -\cot\theta$

$\therefore \sin(2\pi + \theta) = \sin\theta$, $\cos(2\pi + \theta) = \cos\theta$

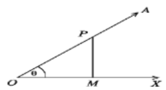
$\tan(2\pi + \theta) = \tan\theta$, $\text{cosec}(2\pi + \theta) = \text{cosec}\theta$

$\sec(2\pi + \theta) = \sec\theta$, $\cot(2\pi + \theta) = \cot\theta$.

$\sin\theta = \frac{PM}{OP} = \frac{\text{opposite side}}{\text{Hypotenuse}}$ [sine of angle $\theta$]

$\cos\theta = \frac{OM}{OP} = \frac{\text{adjacent side}}{\text{Hypotenuse}}$ [cosine of angle $\theta$]

$\tan\theta = \frac{PM}{OM} = \frac{\text{opposite side}}{\text{adjacent side}}$ [tangent of angle $\theta$]



$\text{cosec}\theta = \frac{1}{\sin\theta}$ [cosecant of angle $\theta$]

$\sec\theta = \frac{1}{\cos\theta}$ [secant of angle $\theta$]

$\cot\theta = \frac{1}{\tan\theta}$ [cotangent of angle $\theta$]

$\boxed{\cot\theta = \frac{\cos\theta}{\sin\theta}}$ $\boxed{\tan\theta = \frac{\sin\theta}{\cos\theta}}$ $\boxed{\sec^2\theta - \tan^2\theta = 1}$

$\boxed{(\sin\theta)^2 + (\cos\theta)^2 = 1}$ $\boxed{\text{cosec}^2\theta - \cot^2\theta = 1}$

# Algebraic Ratio and Proportion

3. if $a : b = c : d$ then, $\frac{a+b}{b} = \frac{c+d}{d}$ [Componendo]

if $a : b = c : d$ then, $\frac{a-b}{b} = \frac{c-d}{d}$ [Dividendo]

if $a : b = c : d$, $\frac{a+b}{a-b} = \frac{c+d}{c-d}$ [Componendo-Dividendo]

$\frac{a}{b} = \frac{c}{d} = \frac{e}{f} = \frac{g}{h}$ then each of the ratio $= \frac{a+c+e+g}{b+d+f+h}$

If $a : b = b : c$, prove that, $\left(\frac{a+b}{b+c}\right)^2 = \frac{a^2+b^2}{b^2+c^2}$

If $\frac{a}{b} = \frac{c}{d}$ show that, $\frac{a^2+b^2}{a^2-b^2} = \frac{ac+bd}{ac-bd}$

| Ratio/Angle | 0° | 30° | 45° | 60° | 90° |
|---|---|---|---|---|---|
| sine | 0 | $\frac{1}{2}$ | $\frac{1}{\sqrt{2}}$ | $\frac{\sqrt{3}}{2}$ | 1 |
| cosine | 1 | $\frac{\sqrt{3}}{2}$ | $\frac{1}{\sqrt{2}}$ | $\frac{1}{2}$ | 0 |
| tangent | 0 | $\frac{1}{\sqrt{3}}$ | 1 | $\sqrt{3}$ | undefined |
| cotangent | undefined | $\sqrt{3}$ | 1 | $\frac{1}{\sqrt{3}}$ | 0 |
| secant | 1 | $\frac{2}{\sqrt{3}}$ | $\sqrt{2}$ | 2 | undefined |
| cosecant | undefined | 2 | $\sqrt{2}$ | $\frac{2}{\sqrt{3}}$ | 1 |

$\therefore \sin(90^\circ - \theta) = \frac{OM}{OP} = \cos\angle POM = \cos\theta$

$\cos(90^\circ - \theta) = \frac{PM}{OP} = \sin\angle POM = \sin\theta$

$\tan(90^0 - \theta) = \frac{OM}{PM} = \cot\angle POM = \cot\theta$

$\cot(90^0 - \theta) = \frac{PM}{OM} = \tan\angle POM = \tan\theta$

$\sec(90^0 - \theta) = \frac{OP}{PM} = \text{cosec}\angle POM = \text{cosec}\theta$

$\text{cosec}(90^0 - \theta) = \frac{OP}{OM} = \sec\angle POM = \sec\theta$

# Finite Series

## Arithmetic Series $\therefore n$ th term $= a + (n-1)d$

**Sum of $n$ terms of an arithmetic series**

Let the first term of any arithmetic series be $a$, last term be $p$, common difference be $d$, number of terms be $n$ and sum of $n$ terms be $S_n$.

$$\therefore S_n = \frac{n}{2}(a + p) \ldots (3)$$

i.e., $S_n = \frac{n}{2}\{2a + (n-1)d\} \ldots (4)$

1. $1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}$
2. $1^2 + 2^2 + 3^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6}$
3. $1^3 + 2^3 + 3^3 + \cdots + n^3 = \left\{\frac{n(n+1)}{2}\right\}^2$

**N.B:** $1^3 + 2^3 + 3^3 + \cdots + n^3 = (1 + 2 + 3 + \cdots + n)^2$

# Coordinate Geometry

The distance of $P$ from $Q$ is, $PQ = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

$\therefore$ Area of the Triangle $ABC$

$= \frac{1}{2} \times (BE + AF) \times EF + \frac{1}{2} \times (CD + BE) \times DE - \frac{1}{2} \times (CD + AF) \times DF$

$= \frac{1}{2} \times (y_2 + y_1) \times (x_1 - x_2) + \frac{1}{2} \times (y_3 + y_2) \times (x_2 - x_3) - \frac{1}{2} \times (y_3 + y_1) \times (x_1 - x_3)$

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

$= \frac{1}{2}\begin{vmatrix} x_1 & x_2 & x_3 & x_1 \\ y_1 & y_2 & y_3 & y_1 \end{vmatrix}$ বর্গ একক

# Geometric Series

Let the first term of a geometric series be $a$ and common ratio be $r$. Then, of the series

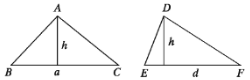
$$n\text{th term} = ar^{n-1}$$

Let the first term of the geometric series be $a$, common ratio $r$ and number of terms $n$. If $S_n$ is the sum of $n$ terms,

$$\therefore S_n = \frac{a(r^n - 1)}{r-1}, \text{ when } r > 1$$

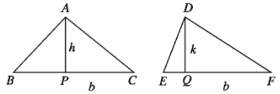
## Ratio, Similarity and Symmetry

1. If the heights of two triangles are equal, their bases and areas are proportional.



$$= \frac{1}{2} \times a \times h : \frac{1}{2} \times d \times h = a : d = BC : EF$$

2. If the bases of two triangles are equal, their heights and areas are proportional.



$$\frac{1}{2} \times b \times h : \frac{1}{2} \times b \times k = h : k = AP : DQ$$

**Theorem 28.** A straight line drawn parallel to one side of a triangle intersects the other two sides or those sides produced proportionally.

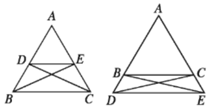


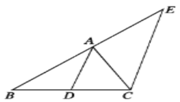
Figure 1 Figure 2

$$AD : DB = AE : EC$$

**Corollary 1.** If the line parallel to $BC$ of the triangle $ABC$ intersects the sides $AB$ and $AC$ at $D$ and $E$ respectively, then $\frac{AB}{AD} = \frac{AC}{AE}$ and $\frac{AB}{BD} = \frac{AC}{CE}$.

**Corollary 2.** The line through the mid point of a side of a triangle parallel to another side bisects the third line.

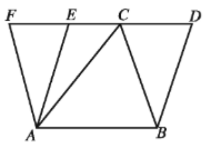
**Theorem 30.** The internal bisector of an angle of a triangle divides its opposite side in the ratio of the sides constituting to the angle.



$$\therefore \frac{BD}{DC} = \frac{BA}{AC}$$

## Area Related Theorems and Constructions

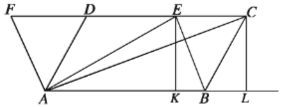
**Theorem 36.** Areas of all the triangular regions having same base and lying between the same pair of parallel lines are equal to one another.



$$\triangle \text{ region } ABC = \triangle \text{ region } DBC$$

**Corollary 1.** If a triangle and a parallelogram lie on bases with equal length and between same pair of parallel lines, the area of the triangle is equal to exactly half of the area of the parallelogram.

**Theorem 38.** Parallelograms lying on the same base and between the same pair of parallel lines arc of equal area.



$\triangle ABC = \triangle ABE$

$\Rightarrow \frac{1}{2}$ area of the parallelogram $ABCD = \frac{1}{2}$ area of the parallelogram $ABEF$.

Area of the parallelogram $ABCD$ = area of the parallelogram $ABEF$. (proved)

## Area of Triangular region

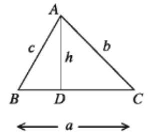
1. **Right angled triangle:** $\frac{1}{2} \times$ base $\times$ height $= \frac{1}{2}ab$
2. **Two sides of a triangular region and the angle included between them are given:**

Area of $\triangle ABC = \frac{1}{2} BC \times AD$

$= \frac{1}{2} a \times b \sin C = \frac{1}{2} ab \sin C$

Similarly, area of $\triangle ABC$

$= \frac{1}{2} bc \sin A = \frac{1}{2} ca \sin B$



3. **Three sides of a triangle are given:**

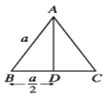
$2s = a + b + c.$

$\therefore$ Area of $\triangle ABC$

$$= \frac{1}{2} BC \cdot AD = \frac{1}{2} \cdot a \cdot \frac{2}{a} \sqrt{s(s-a)(s-b)(s-c)} = \sqrt{s(s-a)(s-b)(s-c)}$$

4. **Equilateral triangle:**

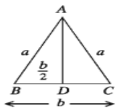
$$\triangle ABC = \frac{1}{2} \cdot BC \cdot AD = \frac{1}{2} \cdot a \cdot \frac{\sqrt{3}a}{2} = \frac{\sqrt{3}}{4} a^2$$



5. **Isosceles triangle:**

Area of isosceles $\triangle ABC = \frac{1}{2} \cdot BC \cdot AD$

$$= \frac{1}{2} \cdot b \cdot \frac{\sqrt{4a^2 - b^2}}{2} = \frac{b}{4} \sqrt{4a^2 - b^2}$$

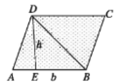


**Area of a parallelogram region:**

**a) Base and height are given:** Let, the base $AB = b$ and height $DE = h$ of parallelogram $ABCD$. The diagonal $BD$ divides the parallelogram into two equal triangular regions.

$\therefore$ The area of the parallelogram $ABCD$

$= 2\times$ area of $\triangle ABD = 2 \times \frac{1}{2} b \cdot h = bh$

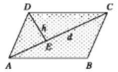


**b) The length of a diagonal and the length of a perpendicular drawn from the opposite angular point on that diagonal are given:**

Let, in a parallelogram $ABCD$, the diagonal $AC = d$ and the perpendicular from opposite angular point $D$ on $AC$ be $DE = h$. Diagonal $AC$ divides the parallelogram into two equal triangular regions.

$\therefore$ area of the parallelogram $ABCD$

$= 2\times$ area of $\triangle ACD = 2 \times \frac{1}{2} d \cdot h = dh$



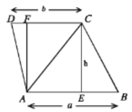
**Area of trapezium region:** Two parallel sides of trapezium region and the distance of perpendicular between them are given. Let $ABCD$ be a trapezium whose lengths of parallel sides are $AB = a$ unit, $CD = b$ unit and distance between them be $CE = AF = h$. Diagonal $AC$ divides the trapezium region $ABCD$ into $\triangle ABC$ and $\triangle ACD$.

Area of trapezium region $ABCD$

= area of $\triangle ABC$ + area of $\triangle ACD$

$= \frac{1}{2} AB \times CE + \frac{1}{2} CD \times AF$

$= \frac{1}{2} ah + \frac{1}{2} bh = \frac{h(a+b)}{2}$



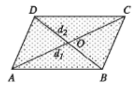
**Area of Rhombus Region:** Two diagonals of a rhombus region are given. Let the diagonals be $AC = d_1$, $BD = d_2$ of the rhombus $ABCD$ and the diagonals intersect each other at $O$.

Diagonal $AC$ divides the rhombus region into two equal triangular regions. We know that the diagonals of a rhombus bisect each other at right angles.

$\therefore$ height of $\triangle ACD = \frac{d_2}{2}$

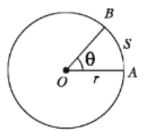
$\therefore$ area of the rhombus $ABCD$

$= 2\times$ area of $\triangle ACD = 2 \times \frac{1}{2} d_1 \cdot \frac{d_2}{2} = \frac{1}{2} d_1 d_2$



## Measurement regarding circle

$\therefore$ diameter of the circle $= 2r$ and circumference $= 2\pi r$



**Length of arc of a circle**

$$\therefore \frac{\theta}{360°} = \frac{s}{2\pi r} \text{ or, } s = \frac{\pi r \theta}{180°}$$

3. **Area of circular region and circular segment**

$$\frac{\theta}{360°} \times \pi r^2$$

# Solid Geometry

## Rectangular solid

$\therefore$ the diagonal of the rectangular solid $= \sqrt{a^2 + b^2 + c^2}$

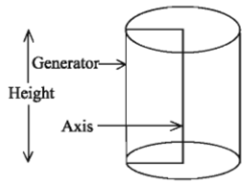
area of the whole surface: $2(ab + bc + ca)$

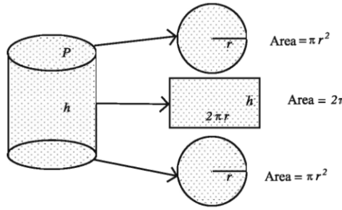
Volume of the rectangular solid = length $\times$ width $\times$ height $= abc$

## Cube

1. The length of diagonal of the cube $= \sqrt{a^2 + a^2 + a^2} = \sqrt{3a^2} = \sqrt{3}a$
2. The area of the whole surface of the cube $= 2(a \cdot a + a \cdot a + a \cdot a) = 2(a^2 + a^2 + a^2) = 6a^2$
3. The volume of the cube $= a \cdot a \cdot a = a^3$

## Cylinder



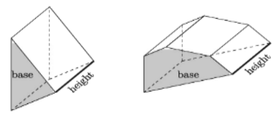




1. Area of the base $= \pi r^2$
2. Area of the curved surface = perimeter of the base $\times$ height $= 2\pi rh$
3. Area of the whole surface

$= (\pi r^2 + 2\pi rh + \pi r^2) = 2\pi r(r + h)$

4. Volume = Area of the base $\times$ height $= \pi r^2 h$

## 3. Prism



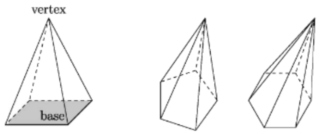

1) The area of total surfaces of a prism

= 2 (area of the base) + area of the lateral surfaces

= 2 (area of the base) + perimeter of the base $\times$ height

2) volume = area of the base $\times$ height

# 4. Pyramid




The base of a pyramid is a any polygon and its lateral surfaces are of any triangular shape. But if the base is a regular polygon and the lateral faces are congruent triangles, the pyramid is called **regular pyramid**. The regular pyramids are eye-catching. The line joining the vertex and any corner of the base is called the **edge** of the pyramid. The length of the perpendicular from the vertex to the base is called the **height** of the pyramid. Usually, a solid with a square base and four congruent triangles meeting at a point is considered as a pyramid. These pyramids are in wide use.

A solid enclosed by four equilateral triangles is known as **regular tetrahedron** which is also a pyramid. This pyramid has $3+3=6$ edges and 4 vertices. The perpendicular from the vertex falls on the centroid of the base.

1) The area of all surfaces of pyramid = Area of the base + area of the lateral surfaces

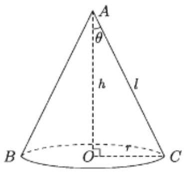
   But if the lateral surfaces are congruent triangles,

   The area of all surafcs of the pyramid = Area of the base+$\frac{1}{2}$ (perimeter of the base × slant height)

   If the height of the perimid is $h$, radius of the inscribed circle of the base is $r$ and $l$ is its slant height, then $l = \sqrt{h^2+r^2}$

2) volume $= \frac{1}{3}\times$ area of the base × height

# 5. Right circular cone




In the figure, the right circular cone $ABC$ is formed by revolving the right-angled triangle $OAC$ about $OA$. In this case, if $\theta$ is the vertical angle $\angle OAC$ of the triangle then it is called the Semi-vertical Angle of the cone.

If the circular cone has height $OA = h$, radius of the base $OC = r$ and slant height $AC = l$, then

1) Area of the curved surface $= \frac{1}{2}\times$ circumference of the base × slant height

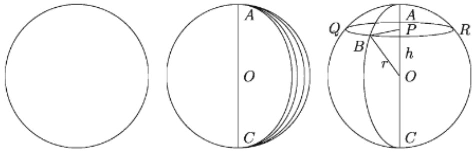
   $= \frac{1}{2}\times 2\pi r \times l = \pi r l$ square units

2) Area of the whole surface = Area of the curved surface + area of base $= \pi r l + \pi r^2 = \pi r(r+l)$ square units

3) Volume $= \frac{1}{3}\times$ area of base × height $= \frac{1}{3}\pi r^2 h$ cubic units [You will learn the method of deduction of this formula in higher classes]

# 6. Sphere

The solid formed by a complete revolution of a semi-circle about its diameter as axis is called a sphere. The centre of the semi-circle is the centre of the sphere. The surface formed by the revolution of the semi-circle about its diameter is the surface of the sphere.




The centre of the sphere $CQAR$ is the point $O$, radius $OA = OB = OC$ and a plane perpendicular to $OA$ and passing through a point at a distance $h$ from the centre cuts the sphere and form the circle $QBR$. The centre of this circle is $P$ and radius $PB$. Then $PB$ and $OP$ are perpendicular to each other.

$\therefore OB^2 = OP^2 + PB^2$

$\therefore PB^2 = OB^2 - OP^2 = r^2 - h^2$

If the radius of the sphere is $r$ then

1) Area of the surface of the sphere $= 4\pi r^2$ sq units
2) Volume $= \frac{4}{3}\pi r^3$ cubic units
3) Radius of the circle formed by the section of a plane at a distance $h$ from the centre $= \sqrt{r^2-h^2}$ unit

So, volume of the cone $= \frac{1}{3}\pi r^2 h = \frac{1}{3}\pi r^3$ cubic units

Volume of the semi-sphere $= \frac{1}{2}(\frac{4}{3}\pi r^3) = \frac{2}{3}\pi r^3$ cubic units

Volume of the cylinder $= \pi r^2 h = \pi r^3$ cubic units

**Example 8.** If the volume of a right circular cone is $V$, the area of its curved surface is $S$, radius of the base is $r$, height is $h$ and semi-vertical angle is $\alpha$. Then show that,

1) $S = \frac{\pi h^2 \tan\alpha}{\cos\alpha} = \frac{\pi r^2}{\sin\alpha}$ square units.
2) $V = \frac{1}{3}\pi h^3 \tan^2\alpha = \frac{\pi r^3}{3\tan\alpha}$ cubic units.