Code Book

Math

Modular arithmetic

```
(a+ b) mod m = (a mod m+ b mod m) mod m
(a- b) mod m = (a mod m- b mod m) mod m
(a· b) mod m = (a mod m· b mod m) mod m
```

---

logk (ab) = logk (a)+logk (b)
logk (x^n) = n·logk (x).
logk (a/b) = logk(a)−logk(b).
max_pow_of_k = log2(k); /// base 2 te k num tar maximum power koto hoita pare oita ber korar niom
How many different squares are there in a grid of N × N squares?
(n*(n+1)*(2*n+1)/6)
How about for N × M squares?
(n*(n+1)*(2*n+1+(3*t))/6) // t = abs(n - m)

Sum of first n positive integers: $1+2+3+\cdots+n=\frac{n(n+1)}{2}$. Why?

Sum of first n odd positive integers: $1+3+5+\cdots+(2n-1)=n^2$. Why?

Sum of first n even positive integers: $2+4+6+\cdots+2n=n(n+1)$. Why?

Sum of first n squares: $1^2+2^2+3^2+\cdots+n^2=\frac{n(n+1)(2n+1)}{6}$.

Sum of first n cubes: $1^3+2^3+3^3+\cdots+n^3=\frac{n^2(n+1)^2}{4}=\left(\frac{n(n+1)}{2}\right)^2$.

$$\sum_{x=1}^{n} x = 1+2+3+\ldots+n = \frac{n(n+1)}{2}$$

$$\sum_{x=1}^{n} x^2 = 1^2+2^2+3^2+\ldots+n^2 = \frac{n(n+1)(2n+1)}{6}.$$

$$\underbrace{a+\cdots+b}_{n \text{ numbers}} = \frac{n(a+b)}{2} \qquad 3+7+11+15 = \frac{4\cdot(3+15)}{2} = 36.$$

If: 5 8 11 14 17.....
Ex: Nth number ki? d = 3, a1 = 5.
aNth = a1 + (n - 1) * d;

Number Theory

Prime check

```
bool isPrime(int n)
{
if(n<=1)return false;
for(int i=2;i*i<=n;i++)if(n%i==0)return
false;
return true;
}
```

Sum of divisors between a to b

```
ll triangle(ll a, ll b)
{
    return (a + b + 1) * (b - a) / 2 ;
}
ll divSum(ll a, ll b)
{
ll n=sqrt(b);
ll sum=0;
for(ll i = 1; i <= n; i++)
sum+=i*(b/i-a/i)+
         triangle(max(n,a/i),max(n,b/i));
return sum;
}
int main()
{
    ll a, b; cin >> a >> b;
    ll ans = divSum(a - 1, b);
    cout << ans << endl;
    return 0;
}
```

Sum of Divisors & Divisors count 1 to N

```
/// O(nlog(log(n)))
for(int i=1; i<=n; i++)
{
    for(int j=i; j<=n; j+=i)
    {
        Div[j]++;
        //Div[j]+=i;///for sumOfDivisors
    }
}
```

---

2 ta number er common divisor's jeta 2 ta number ke e divide kore

```
int32_t main()
{
    int a, b; cin >> a >> b;
    int gcd = __gcd(a, b);
    //cout << "gcd: " << gcd << endl;
    for(int i = 1; i <= gcd; i++)
    {
       if(gcd % i == 0) cout << i << ' ';
    }
}
```

X to the power n under modulo mod:

```
ll power(ll a, ll k)
{
ll ans = 1;
while(k)
{
if(k&1) ans = 1LL * ans * a % mod;
a = 1LL * a * a % mod;
k >>= 1;
}
return ans;
}
```

Calculating x * y under modulo mod:

```
ll mulmod(ll x,ll y,ll mod)
{
  // O(1)
  return (ll)((__int128)x * y % mod);
}
```

Ceil ber korar niom: (a-1)/b + 1;

Number of values smaller than x = lower bound
Number of values greater or equal to x = n - lower bound
Number of values less than or equal to x = upper bound
Number of values greater than x = n - upper bound

Prime Factorization

```
#include<bits/stdc++.h>
using namespace std;
#define endl "\n"
int32_t main()
{
int t; cin >> t;
while(t--)
{
int n; cin >> n;
vector<int> v;
for(int i=2;(long long)i*i<=n;i++)
{
    if(n % i == 0)
    {
        while(n % i == 0)
        {
            v.push_back(i);
            n /= i;
        }
    }
}
if(n > 1) v.push_back(n);
for(auto u : v) cout << u << ' ';
cout << endl;
}
}
```

Sum of Divisor with Factorization

```
const int N = 1e6 + 128;
int spf[N];
ll SOD(ll n)
{
int sod = 1;
while(n > 1)
{
    int p = spf[n];
    int sum = 1, e = 1;
    while(n > 1 and p == spf[n])
    {
        e *= spf[n];
        sum += e;
        n /= spf[n];
    }
    sod *= sum;
}
return sod;
}
int32_t main()
{
for(int i = 2; i < N; i++)
{
    spf[i] = i;
}
for(int i = 2; i < N; i++)
{
    if(spf[i] == i)
    {
        for(int j = i; j < N; j += i)
        {
            spf[j] = min(spf[j], i);
        }
    }
}
int t; cin >> t;
while(t--)
{
    int n; cin >> n;
    cout << SOD(n) << endl;
}
}
```

BigMod

```
ll BigMod( ll a, ll b, ll m )
{
    if(!b) return 1 % m;
    ll x = BigMod(a, b/2, m);
    x = (x * x) % m;
    if(b & 1) x = (x * a) % m;
    return x;
}
```

Prime Factorization using Sieve

```
#include<bits/stdc++.h>
using namespace std;
#define endl "\n"
const int N = 1e6 + 128;
int spf[N];
int32_t main()
{
for(int i = 2; i < N; i++) spf[i] =
i;
for(int i = 2; i < N; i++)
{
for(int j = i; j < N; j += i)
{
    spf[j] = min(spf[j], i);
}
}
int q; cin >> q;
while(q--)
{
int n; cin >> n;
vector<int> ans;
while(n > 1)
{
    ans.push_back(spf[n]);
    n /= spf[n];
}
for(auto u : ans) cout << u << ' ';
    cout << endl;
}
}
```

Sieve

```
bitset<N> isPrime;
vector<int> primes;
void sieve(int n)
{
for(int i=3; i<=n; i+=2) isPrime[i] =
1;
for(int i=3; i*i<=n; i+=2)
{
    if(isPrime[i])
    {
      for(int j=i*i; j<=n;
j+=i)isPrime[j]=0;
    }
}
isPrime[2] = 1; primes.push_back(2);
for(int i=3; i<=n; i+=2)
{
    if(isPrime[i]==1)
primes.push_back(i);
}
}
```

Kth divisor find

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
#define endl "\n"
set<ll> s;
void divisor(ll n)
{
for(ll i=1; i<=sqrt(n); i++)
{
if(n % i == 0)
{
    ll x = i;
    ll y = n/i;
    s.insert(x);
    s.insert(y);
}
}
}
int main()
{
ll n, k; cin >> n >> k;
divisor(n);
vector<ll> v;
v.assign(s.begin(), s.end());
if(v.size() >= k) cout << v[k-1] << endl;
else cout << -1 << endl;
}
```

Subarrays having sum x

```
for(int i = 1; i <= n; i++)
sum[i] = sum[i - 1] + a[i];
map<ll, int> cnt;
ll ans = 0;
cnt[0] = 1;
for(int i = 1; i <= n; i++)
{
    ll t = sum[i] - x;
    ans += cnt[t];
    cnt[sum[i]]++;
}
cout << ans << endl;
```

Is a isPerfectSquare

```
bool isPerfectSquare(long long n)
{
    if(n == 0) return true;
    long long odd = 1;
    while(n > 0)
    {
        n -= odd;
        odd += 2;
        cout << odd << endl;
    }
    return n == 0;
}
```

N! Problems

int trailingZeroes(int n)

```
{
int c = 0, f = 5;
while(f <= n) { c += n/f; f *= 5; }
return c;
}
```

int factDigitCnt(int n)

```
{
if(n<=1) return n;
double digits = 0;
for(int i=2; i<=n; i++)
{
    digits += log10(i);
}
return floor(digits)+1;
}
```

How many N in the given range have K-Divisors in (L to R)

```
#include<bits/stdc++.h>
using namespace std;
using ll = long long;
namespace pcf
{
// initialize once by calling init()
#define MAXN 10000010 // initial sieve limit
#define MAX_PRIMES 1000010 // max size of the prime array for sieve
#define PHI_N 100000
#define PHI_K 100

int len = 0; // total number of primes generated by sieve
int primes[MAX_PRIMES];
int pref[MAXN]; // pref[i] --> number of primes <= i
int dp[PHI_N][PHI_K]; // precal of yo(n,k)
bitset <MAXN> f;
void sieve(int n)
{
    f[1] = true;
    for (int i = 4; i <= n; i += 2) f[i] = true;
    for (int i = 3; i * i <= n; i += 2)
    {
        if (!f[i])
        {
            for (int j = i * i; j <= n; j += i << 1) f[j] = 1;
        }
    }
    for (int i = 1; i <= n; i++)
    {
        if (!f[i]) primes[len++] = i;
        pref[i] = len;
    }
}
void init()
{
    sieve(MAXN - 1);
    // precalculation of phi upto size (PHI_N,PHI_K)
    for (int n = 0; n < PHI_N; n++) dp[n][0] = n;
    for (int k = 1; k < PHI_K; k++)
    {
        for (int n = 0; n < PHI_N; n++)
        {
            dp[n][k] = dp[n][k - 1] - dp[n / primes[k - 1]][k - 1];
        }
    }
}
// returns the number of integers less or equal n which are
// not divisible by any of the first k primes
// recurrence --> yo(n, k) = yo(n, k-1) - yo(n / p_k , k-1)
// for sum of primes yo(n, k) = yo(n, k-1) - p_k * yo(n / p_k , k-1)
long long yo(long long n, int k)
{
    if (n < PHI_N && k < PHI_K) return dp[n][k];
    if (k == 1) return ((++n) >> 1);
    if (primes[k - 1] >= n) return 1;
    return yo(n, k - 1) - yo(n / primes[k - 1], k - 1);
}
// complexity: n^(2/3).(log n^(1/3))
```

```
long long Legendre(long long n)
{
    if (n < MAXN) return pref[n];
    int lim = sqrt(n) + 1;
    int k = upper_bound(primes, primes + len, lim) - primes;
    return yo(n, k) + (k - 1);
}
// runs under 0.5s for n = 1e12
long long Lehmer(long long n)
{
    if (n < MAXN) return pref[n];
    long long w, res = 0;
    int b = sqrt(n), c = Lehmer(cbrt(n)), a = Lehmer(sqrt(b));
    b = Lehmer(b);
    res = yo(n, a) + ((1LL * (b + a - 2) * (b - a + 1)) >> 1);
    for (int i = a; i < b; i++)
    {
        w = n / primes[i];
        int lim = Lehmer(sqrt(w));
        res -= Lehmer(w);
        if (i <= c)
        {
            for (int j = i; j < lim; j++)
            {
                res += j;
                res -= Lehmer(w / primes[j]);
            }
        }
    }
    return res;
}
}
using namespace pcf;
const ll inf = 1e12 + 9; // > max n
const int M = 2020; // maximum number of divisors
const int C = 1010; // 4th root of max n
const int L = 41; // 2^L >= max n
ll pw[C][L];
int s[M];
ll power(ll n, int k)
{
    if (k >= L) return inf;
    if (n < C) return pw[n][k];
    if (k == 1) return n;
    if (k == 2) return n * n;
    if (k == 3) return n * n * n;
    return inf;
}
vector<int> d[M];
// returns the count of numbers <= n s.t. each has exactly k divisors and
isn't divisible by the first 'last' primes
// pretty fast!
ll backtrack(ll n, int k, int last = 0)
{
    if (k >= M) return 0;
    if (n <= 0) return 0;
    if (k == 1) return 1;
    if (n == 1) return 0;
    if (k == 2) return max(0LL, Lehmer(n) - last);
    if (d[k].size() == 1)
```

```
    {
        if (d[k][0] - 1 >= L) return 0;
        if (power(primes[last], d[k][0] - 1) > n) return 0;
        int ans = last - 1, l = last, r = pow(n, 1.0 / (d[k][0] - 1)) + 1;
        while (l <= r)
        {
            int mid = (l + r) >> 1;
            ll p = power(primes[mid], d[k][0] - 1);
            if (p > n) r = mid - 1;
            else l = mid + 1, ans = mid;
        }
        return ans - last + 1;
    }
    ll ans = 0;
    for (auto x: d[k])
    {
        for (int i = last; i < len; i++)
        {
            ll p = power(primes[i], x - 1);
            if (p > n) break;
            if (k / x == 1)
            {
                ++ans;
                continue;
            }
            if (power(primes[i + 1], s[k / x]) > n / p) break;
            ans += backtrack(n / p, k / x, i + 1);
        }
    }
    return ans;
}
int32_t main()
{
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    init();
    for (int i = 2; i < M; i++)
    {
        for (int j = i; j < M; j += i)
        {
            d[j].push_back(i);
        }
        reverse(d[i].begin(), d[i].end());
        for (int j = 0; j < len; j++)
        {
            if (primes[j] > i) break;
            int cur = i;
            while (cur % primes[j] == 0)
            {
                cur /= primes[j];
                s[i] += primes[j] - 1;
            }
        }
    }
    for (int i = 2; i < C; i++)
    {
        pw[i][0] = 1;
        for (int j = 1; j < L; j++)
        {
            pw[i][j] = pw[i][j - 1] * i;
```

```
            if (pw[i][j] > inf) pw[i][j] = inf;
        }
    }
    int t, cs = 0;
    cin >> t;
    while (t--)
    {
        ll l, r, k;
        cin >> l >> r >> k;
        ll ans = backtrack(r, k);
        ans -= backtrack(l - 1, k);
        cout << "Case " << ++cs << ": " << ans << '\n';
    }
    return 0;
}
```

Prime Counting 1 to N

```
#include<bits/stdc++.h>
using namespace std;
const int N = 3e5 + 9;
namespace pcf
{
// initialize once by calling init()
#define MAXN 20000010 // initial sieve limit
#define MAX_PRIMES 2000010 // max size of the prime array for sieve
#define PHI_N 100000
#define PHI_K 100

int len = 0; // total number of primes generated by sieve
int primes[MAX_PRIMES];
int pref[MAXN]; // pref[i] --> number of primes <= i
int dp[PHI_N][PHI_K]; // precal of yo(n,k)
bitset <MAXN> f;
void sieve(int n)
{
    f[1] = true;
    for (int i = 4; i <= n; i += 2) f[i] = true;
    for (int i = 3; i * i <= n; i += 2)
    {
        if (!f[i])
        {
            for (int j = i * i; j <= n; j += i << 1) f[j] = 1;
        }
    }
    for (int i = 1; i <= n; i++)
    {
        if (!f[i]) primes[len++] = i;
        pref[i] = len;
    }
}
void init()
{
    sieve(MAXN - 1);
// precalculation of phi upto size (PHI_N,PHI_K)
    for (int n = 0; n < PHI_N; n++) dp[n][0] = n;
    for (int k = 1; k < PHI_K; k++)
    {
        for (int n = 0; n < PHI_N; n++)
        {
            dp[n][k] = dp[n][k - 1] - dp[n / primes[k - 1]][k - 1];
```

```
        }
    }
}
// returns the number of integers less or equal n which are
// not divisible by any of the first k primes
// recurrence --> yo(n, k) = yo(n, k-1) - yo(n / p_k , k-1)
// for sum of primes yo(n, k) = yo(n, k-1) - p_k * yo(n / p_k , k-1)
long long yo(long long n, int k)
{
    if (n < PHI_N && k < PHI_K) return dp[n][k];
    if (k == 1) return ((++n) >> 1);
    if (primes[k - 1] >= n) return 1;
    return yo(n, k - 1) - yo(n / primes[k - 1], k - 1);
}
// complexity: n^(2/3).(log n^(1/3))
long long Legendre(long long n)
{
    if (n < MAXN) return pref[n];
    int lim = sqrt(n) + 1;
    int k = upper_bound(primes, primes + len, lim) - primes;
    return yo(n, k) + (k - 1);
}
// runs under 0.2s for n = 1e12
long long Lehmer(long long n)
{
    if (n < MAXN) return pref[n];
    long long w, res = 0;
    int b = sqrt(n), c = Lehmer(cbrt(n)), a = Lehmer(sqrt(b));
    b = Lehmer(b);
    res = yo(n, a) + ((1LL * (b + a - 2) * (b - a + 1)) >> 1);
    for (int i = a; i < b; i++)
    {
        w = n / primes[i];
        int lim = Lehmer(sqrt(w));
        res -= Lehmer(w);
        if (i <= c)
        {
            for (int j = i; j < lim; j++)
            {
                res += j;
                res -= Lehmer(w / primes[j]);
            }
        }
    }
    return res;
}
}
int32_t main()
{
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    pcf::init();
    long long n;
    cin >> n;
    cout << pcf::Lehmer(n) << '\n';
    return 0;
}
```

Sum of The Number of Divisors from 1 to N

```
/// if N = 10
/*
1 = 1
2 = 2
3 = 2
4 = 3
5 = 2
6 = 4
7 = 2
8 = 4
9 = 3
10 = 4
-------
1 to 10 == 27 Divisor
*/

#include<bits/stdc++.h>
using namespace std;

using uint32 = unsigned int;
using uint64 = unsigned long long;
using uint128 = __uint128_t;

uint128 sum_sigma0(uint64 n)
{
    auto out = [n] (uint64 x, uint32 y)
    {
        return x * y > n;
    };
    auto cut = [n] (uint64 x, uint32 dx, uint32 dy)
    {
        return uint128(x) * x * dy >= uint128(n) * dx;
    };
    const uint64 sn = sqrtl(n);
    const uint64 cn = pow(n, 0.34); //cbrtl(n);
    uint64 x = n / sn;
    uint32 y = n / x + 1;
    uint128 ret = 0;
    stack<pair<uint32, uint32>> st;
    st.emplace(1, 0);
    st.emplace(1, 1);
    while (true)
    {
        uint32 lx, ly;
        tie(lx, ly) = st.top();
        st.pop();
        while (out(x + lx, y - ly))
        {
            ret += x * ly + uint64(ly + 1) * (lx - 1) / 2;
            x += lx, y -= ly;
        }
        if (y <= cn) break;
        uint32 rx = lx, ry = ly;
        while (true)
        {
            tie(lx, ly) = st.top();
            if (out(x + lx, y - ly)) break;
            rx = lx, ry = ly;
            st.pop();
```

```
        }
        while (true)
        {
            uint32 mx = lx + rx, my = ly + ry;
            if (out(x + mx, y - my))
            {
                st.emplace(lx = mx, ly = my);
            }
            else
            {
                if (cut(x + mx, lx, ly)) break;
                rx = mx, ry = my;
            }
        }
    }
    for (--y; y > 0; --y) ret += n / y;
    return ret * 2 - sn * sn;
}

int32_t main()
{
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    int t;
    cin >> t;
    while (t--)
    {
        long long n;
        cin >> n;
        auto ans = sum_sigma0(n);
        string s = "";
        while (ans > 0)
        {
            s += char('0' + ans % 10);
            ans /= 10;
        }
        reverse(s.begin(), s.end());
        cout << s << '\n';
    }
    return 0;
}
```

Data Structure

Segment Tree

```
/// template for q query => l, r and find min,min value and sum
vector<int> a;
vector<int> seg_tree;
void build(int node, int b, int e)
{
    if(b == e)
    {
        seg_tree[node] = a[b];
        return;
    }
    int Left, Right, Mid;
    Left = node * 2;
    Right = (node * 2) + 1;
    Mid = b + ((e - b) / 2);
    build(Left, b, Mid);
    build(Right, Mid + 1, e);
    seg_tree[node] = seg_tree[Left] + seg_tree[Right];
    ///seg_tree[node] = min(seg_tree[Left], seg_tree[Right]);
                     ///max(seg_tree[Left], seg_tree[Right]);
}
int query(int node, int b, int e, int I, int J)
{
    if(e < I || J < b) return 0;///INT_MAX; // INT_MIN
    if(I <= b && e <= J) return seg_tree[node];
    int Left, Right, Mid, q1, q2;
    Left = node * 2;
    Right = (node * 2) + 1;
    Mid = b + ((e - b) / 2);
    q1 = query(Left, b, Mid, I, J);
    q2 = query(Right, Mid + 1, e, I, J);
    return q1 + q2;
    ///return min(q1, q2); //max(q1, q2);
}
void upd(int node, int b, int e, int I, int x)
{
    if(b > I || e < I) return;
    if(b == e && b == I)
    {
        seg_tree[node] = x;
        return;
    }
    int Mid, id, Left, Right;
    Left = node * 2;
    Right = (node * 2) + 1;
    Mid = b + ((e - b) / 2);
    upd(Left, b, Mid, I, x);
    upd(Right, Mid + 1, e, I, x);
    seg_tree[node] = seg_tree[Left] + seg_tree[Right];
    ///seg_tree[n] = max(seg_tree[Left], seg_tree[Right]);
}
int32_t main()
{
    int t; cin >> t;
    for(int tc = 1; tc <= t; tc++)
    {
        int n, q; cin >> n >> q;
        a.assign(n + 2, 0);
        seg_tree.assign((4 * n) + 2, 0);
```

```
        for(int i = 1; i <= n; i++) cin >> a[i];
        build(1, 1, n);
        cout << "Case " << tc << ":" << endl;
        while(q--)
        {
            int l, r;
            cin >> l >> r;
            cout << query(1, 1, n, l, r) << endl;
            ///int k, m; cin >> k >> m;
            ///upd(1, 1, n, k, m); // kth index +m value
        }
    }
}
```

Difference array

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e6 + 9;
int a[N], d[N];
int main()
{
int n = 5;
a[1]=1,a[2]=3,a[3]=5,a[4]=2,a[5]=4;
// Finding the difference array from the
original array
for (int i = 1; i <= n; ++i)
{
    d[i] = a[i] - a[i - 1];
}
for (int i = 1; i <= n; ++i)
{
    cout << d[i] << " ";
}
cout << endl;
// Finding the original array from the
difference array
for (int i = 1; i <= n; ++i)
{
    a[i] = a[i - 1] + d[i];
}
for (int i = 1; i <= n; ++i)
{
    cout << a[i] << " ";
}
cout << endl;
}
```

Two Pointer

```
#include<bits/stdc++.h>
using namespace std;
#define endl "\n"
#define ll long long
const int N = 1e5 + 128;
int a[N];
int32_t main()
{
int n; ll s; cin >> n >> s;
for(int i = 1; i <= n; i++)
{
    cin >> a[i];
}
int ans = 0, r = 1; ll sum = 0;
for(int l = 1; l <= n; l++)
{
```

2D Prefix Sum

```
#include<bits/stdc++.h>
using namespace std;
#define endl "\n"
const int N = 1e3 + 128;
int a[N][N];
int pref_sum[N][N];
int32_t main()
{
int n, q; cin >> n >> q;
for(int i = 1; i <= n; i++)
{
for(int j = 1; j <= n; j++)
{
char c; cin >> c;
if(c == '*') a[i][j] = 1;
else a[i][j] = 0;
}
}
for(int i = 1; i <= n; i++)
{
for(int j = 1; j <= n; j++)
{
pref_sum[i][j]=a[i][j]+pref_sum[i-1][j]+
pref_sum[i][j-1]-pref_sum[i-1][j-1];
}
}
while(q--)
{
int x1, y1, x2, y2; cin >> x1 >> y1 >> x2
>> y2;
int ans =
pref_sum[x2][y2]-(pref_sum[x1-1][y2] +
pref_sum[x2][y1-1])+pref_sum[x1-1][y1-1];
cout << ans << endl;
}
}
```

```
    while(r <= n and sum + a[r] <= s)
    {
        sum += a[r];
        r++;
    }
    ans = max(ans, r - l);
    sum -= a[l];0
}
cout << ans << endl;
}
```

0/1 knapsack

```
#include<bits/stdc++.h>
using namespace std;

#define endl "\n"
#define int long long

const int N = 1e5+128;
int w[N], v[N];
int n, mx_w, dp[100][N];

int solve(int i, int c_w) //current_w =
c_w
{
if(i == n + 1) return 0;
if(dp[i][c_w] != -1) return dp[i][c_w];
int ans = solve(i + 1, c_w);
if(c_w + w[i] <= mx_w) ans = max(ans,
solve(i + 1, c_w + w[i]) + v[i]);
return dp[i][c_w] = ans;
}
int32_t main()
{
ios_base::sync_with_stdio(0);
cin.tie(0);
cout.tie(0);
cin >> n >> mx_w;
for(int i = 1; i <= n; i++)
{
cin >> w[i] >> v[i];
}
memset(dp, -1, sizeof(dp));
cout << solve(1, 0) << endl;
}
```

Bit Manipulation

```
#include<bits/stdc++.h>
using namespace std;
bool check_Kth_bit_on_or_off(int x, int k)
{
    return (x >> k) & 1;
}
void print_on_bit(int x)
{
    for(int i=0; i<32; i++)
    {
        if( check_Kth_bit_on_or_off(x, i) ) cout << i << " ";
    }
    cout << endl;
}
void print_off_bit(int x)
{
    for(int i=0; i<32; i++)
    {
        if( !check_Kth_bit_on_or_off(x, i) ) cout << i << " ";
    }
    cout << endl;
}
int count_on_bit(int x)
{
    int cnt = 0;
    for(int i=0; i<32; i++)
    {
        if( check_Kth_bit_on_or_off(x, i) ) cnt++;
    }
    return cnt;
}
bool is_even(int x)
{
    if(x & 1) cout << "odd" << endl; /// sob even number er last bit 0 hoy,
                                          //odd number er 1 hoy
    else cout << "even" << endl;
}
int set_Kth_bit(int x, int k)
{
    return(x | (1 << k));
}

int unset_Kth_bit(int x, int k)
{
    return(x & (~(1 << k)));
}
int toggle_Kth_bit(int x, int k)
{
    return(x ^ ((1 << k)));
}
bool check_power_of_2(int x)
{
    return (count_on_bit(x) == 1);
}
int main()
{
    ///How to multiply a number by 2 without using  * ? Or how to divide a
number by 2 without using
    //cout << (1 << 6) << endl; /// 1*2^6 (2 er power ber kora jay)
```

```
    ///How to find the power of 2?
    //cout << (1LL*1 << 2) << endl;
    //cout << (10 >> 1) << endl; /// 5
    ///How to check if a bit is on or off in a number?
    //cout << check_Kth_bit_on_or_off(11, 2) << endl; /// 0
    //cout << check_Kth_bit_on_or_off(11, 3) << endl; /// 1
    ///How to find the on bits in a number?
    //print_on_bit(11); /// 0 1 3
    ///How to find the off bits in a number?
    //print_off_bit(11); /// 2 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
                          /// 21 22 23 24 25 26 27 28 29 30 31
    ///How to find the number of on bits in a number?
    //cout << count_on_bit(11) << endl;
    ///How to check if a number is even or odd?
    //is_even(9); // odd
    ///How to turn on a bit in a number?
    //cout << set_Kth_bit(11, 2) << endl; // 15
    ///How to turn off a bit in a number?
    //cout << unset_Kth_bit(15, 2) << endl; // 11
    ///How to toggle a bit in a number?
    //cout << toggle_Kth_bit(10, 2) << endl; // 12
    ///How to check if a number is a power of 2?
    //cout << check_power_of_2(16) << endl; // true
    //cout << __builtin_popcount(5) << endl; /// ekta number a koyta set on
                                             ///ase ta ber kore diba for int. O(1)
    //cout << __builtin_popcountll(5) << endl; /// ekta number a koyta set on
                          ///ase ta ber kore diba for long long. O(1)

}
```

Graph Theory

```
void dfs(int u)
{
    vis[u] = true;
    for(auto v : adj[u])
    {
        if(!vis[v])
        {
            dfs(v);
        }
    }
}
```

```
void bfs(int s_node)
{
    queue<int> q;
    q.push(s_node);
    vis[s_node] = true;
    while(!q.empty())
    {
        int c_node = q.front();
        q.pop();
        for(auto u : adj[c_node])
        {
            if(!vis[u])
            {
                q.push(u);
                vis[u] = true;
            }
        }
    }
}
```

Diameter of a Tree

```
#include<bits/stdc++.h>
using namespace std;
#define endl "\n"
const int N = 2e5+128;
vector<int> g[N];
int ans, mx, node;
int dep[N];
void dfs1(int u, int p)
{
    dep[u] = dep[p] + 1;
    if(dep[u] >= mx)
    {
        mx = dep[u];
        node = u;
    }
    for(auto v : g[u])
    {
        if(v != p) dfs1(v, u);
    }
}
void dfs2(int u, int p)
{
    dep[u] = dep[p] + 1;
    ans = max(ans, dep[u] - 1);
    for(auto v : g[u])
    {
        if(v != p) dfs2(v, u);
    }
}
int32_t main()
{
```

Finding Cycle

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
#define endl "\n"
const int N = 2e5+128;
vector<int> g[N];
int col[N], par[N];
bool cycle;
vector<int> cyc;
void dfs(int u)
{
col[u] = 1;
for(auto v : g[u])
{
if(col[v] == 0)
{
    par[v] = u;
    dfs(v);
}
else if(col[v] == 1)
{
    int path_last = v;
    int cycle_path = u;
    while(cycle_path != path_last)
    {
        cout << cycle_path << ' ';
        cycle_path = par[cycle_path];
    }
    cout << path_last << endl;
    cycle = true;
}
}
col[u] = 2;
}
int32_t main()
{
int n, m; cin >> n >> m;
for(int i = 0; i < m; i++)
{
    int u, v;
    cin >> u >> v;
    g[u].push_back(v);
    //g[v].push_back(u);
}
cycle = false;
for(int i = 1; i <= n; i++)
{
    if(col[i] == 0) dfs(i);
}
cout << (cycle ? "Cycle" : "No
Cycle") << endl;
return 0;
}
```

```
    int n;cin >> n;
    for(int i = 0; i < n - 1; i++)
    {
        int u, v;
        cin >> u >> v;
        g[u].push_back(v);
        g[v].push_back(u);
    }
    dfs1(1, 0);
    dfs2(node, 0);
    cout << ans << endl;
    return 0;
}
```

Bicolering

```
#include<bits/stdc++.h>
using namespace std;
#define endl "\n"
const int N = 2e5+128;
vector<int> g[N];
int col[N];
bool isOk = true;
void dfs(int u, int clr)
{
    col[u] = clr;
    for(auto v : g[u])
    {
        if(col[v] == 0)
        {
            dfs(v, (clr == 1 ? 2 : 1));
        }
        else if(col[u] == col[v]) isOk =
false;
    }
}
int32_t main()
{
    ios_base::sync_with_stdio(0);
    cin.tie(0); cout.tie(0);
    int t, tc = 0; cin >> t;
    while(t--)
    {
        for(int i = 0; i <= N; i++)
        {
            g[i].clear();
            col[i] = 0;
        }
        int n, m; cin >> n >> m;
        for(int i = 0; i < m; i++)
        {
            int u, v;
            cin >> u >> v;
            g[u].push_back(v);
            g[v].push_back(u);
        }
        isOk = true;
        for(int i = 1; i <= n; i++)
        {
            if(col[i] == 0) dfs(i, 1);
        }
cout << (isOk ? "Bicolorable" : "Not
Bicolorable") << endl;
    }
    return 0;
}
```

Odd Length Cycle Not Exist

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
#define endl "\n"
const int N = 2e5+128;
vector<int> g[N];
bool vis[N];
int col[N];
bool isBipartite = true;
void dfs(int u, int clr)
{
vis[u] = true;
col[u] = clr;
if(clr == 1) clr = 2;
else clr = 1;
for(auto v : g[u])
{
if(!vis[v])
{
dfs(v, clr);
}
else if(col[v] == col[u])
{
isBipartite = false;
}
}
}
int32_t main()
{
ios_base::sync_with_stdio(0);
cin.tie(0);cout.tie(0);
int n, m; cin >> n >> m;
for(int i = 0; i < n; i++)
{
int u, v;
cin >> u >> v;
g[u].push_back(v);
g[v].push_back(u);
}
for(int i = 1; i <= n; i++)
{
if(!vis[i]) dfs(i, 1);
}
cout << (isBipartite ? "Odd Length
Cycle Not Exist" : "Odd Length Cycle
Exist") << endl;
return 0;
}
```

Shortest Path And Grid Access with Bfs

```
#include<bits/stdc++.h>
using namespace std;
#define endl "\n"
const int N = 1e3 + 128, inf = 1e9 + 128;
string s[N];
pair<int, int> par[N][N];
int dis[N][N], n, m;
int di[] = {1, 0, -1, 0};
int dj[] = {0, 1, 0, -1};
bool is_valid(int i, int j)
{
return (i >= 0 and i < n and j >= 0 and j
< m);
}
int32_t main()
{
ios_base::sync_with_stdio(0);
cin.tie(0); cout.tie(0);
cin >> n >> m;
for(int i = 0; i < n; i++)
{
cin >> s[i];
}
pair<int, int> _start, _end;
for(int i = 0; i < n; i++)
{
for(int j = 0; j < m; j++)
{
if(s[i][j] == 'A')
{
_start = {i, j};
}
if(s[i][j] == 'B')
{
_end = {i, j};
}
}
}
for(int i = 0; i < n; i++)
{
for(int j = 0; j < m; j++)
{
dis[i][j] = inf;
}
}
queue<pair<int, int>> q;
q.push(_start);
dis[_start.first][_start.second] = 0;
while(!q.empty())
{
auto [i, j] = q.front();
q.pop();
for(int k = 0; k < 4; k++)
{
int next_i = i + di[k];
int next_j = j + dj[k];
if(is_valid(next_i, next_j) and
s[next_i][next_j] != '#' and dis[i][j] +
1 < dis[next_i][next_j])
{
dis[next_i][next_j] = dis[i][j] + 1;
q.push({next_i, next_j});
par[next_i][next_j] = {i, j};
}
```

Grid Access with Dfs

```
#include<bits/stdc++.h>
using namespace std;
#define endl "\n"
#define mem(a,b) memset(a, b, sizeof(a))
const int N = 200 + 128;
bool vis[N][N];
string s[N];
int n, m;
int di[]={0,0,+1,-1,+1,+1,-1,-1};
int dj[]={+1,-1,0,0,+1,-1,+1,-1};
void dfs(int i, int j)
{
vis[i][j] = true;
for(int k = 0; k < 8; k++)
{
int i1 = i + di[k];
int j1 = j + dj[k];
if(i1 >= 0 and i1 < n and j1 >= 0 and j1
< m and !vis[i1][j1] and s[i1][j1] ==
'@')
{
dfs(i1, j1);
}
}
}
int main()
{
ios_base::sync_with_stdio(0);
cin.tie(0); cout.tie(0);
while(cin >> n >> m and n and m)
{
for(int i = 0; i < n; i++)
{
    cin >> s[i];
}
int component = 0;
for(int i = 0; i < n; i++)
{
    for(int j = 0 ; j < m; j++)
    {
        if(s[i][j] == '@' and !vis[i][j])
        {
            dfs(i, j);
            component++;
        }
    }
}
cout << component << endl;
for(int i = 0; i < n; i++)
{
    for(int j = 0; j < m; j++)
    {
        vis[i][j] = false;
    }
}
}
return 0;
}
```

```
}
}
int ans = dis[_end.first][_end.second];
if(ans == inf)
{
cout << "NO" << endl;
}
else
{
cout << "YES\n" << ans << endl;
vector<pair<int, int>> path;
auto cur = _end;
while(cur != _start)
{
path.push_back(cur);
cur = par[cur.first][cur.second];
}
path.push_back(_start);
reverse(path.begin(), path.end());
for(int k = 0; k < path.size() - 1; k++)
{
int i = path[k].first - path[k +
1].first;
int j = path[k].second - path[k +
1].second;
if(i == 1) cout << 'U';
else if(i == -1)cout << 'D';
else if(j == 1) cout << 'L';
else cout << 'R';
}
}
return 0;
}
```

Dijkstra

```
#include<bits/stdc++.h>
using namespace std;
const int N = 3e5 + 9, mod = 998244353;
int n, m;
vector<pair<int, int>> g[N], r[N];
vector<long long> dijkstra(int s, int t,
vector<int> &cnt) {
  const long long inf = 1e18;
  priority_queue<pair<long long, int>,
vector<pair<long long, int>>,
greater<pair<long long, int>>> q;
  vector<long long> d(n + 1, inf);
  vector<bool> vis(n + 1, 0);
  q.push({0, s});
  d[s] = 0;
  cnt.resize(n + 1, 0); // number of
shortest paths
  cnt[s] = 1;
  while(!q.empty()) {
    auto x = q.top();
    q.pop();
    int u = x.second;
    if(vis[u]) continue;
    vis[u] = 1;
    for(auto y: g[u]) {
      int v = y.first;
      long long w = y.second;
      if(d[u] + w < d[v]) {
        d[v] = d[u] + w;
        q.push({d[v], v});
        cnt[v] = cnt[u];
```

Topological Sorting

```
#include <bits/stdc++.h>
using namespace std;
#define endl "\n"
const int N = 128;
vector<int> adj[N];
bool vis[N];
int indeg[N];
int main()
{
    int n, m; cin >> n >> m;
    for(int i = 1; i <= m; i++)
    {
        int u, v;
        cin >> u >> v;
        adj[u].push_back(v);
        indeg[v]++;
    }
    vector<int> zero;
    for(int i = 1; i <= n; i++)
    {
        if(indeg[i] == 0)
        {
            zero.push_back(i);
            vis[i] = true;
        }
    }
    vector<int> ans;
    while(ans.size() != n)
    {
        if(zero.empty())
        {
            cout << "Impossible" << endl;
            return 0;
        }
        int cur = zero.back();
        zero.pop_back();
        vis[cur] = true;
        ans.push_back(cur);
        for(auto v : adj[cur])
        {
            indeg[v]--;
            if(!vis[v] and indeg[v] == 0)
            {
                zero.push_back(v);
                vis[v] = true;
            }
        }
    }
    for(auto x : ans)
    {
        cout << x << ' ';
    }
    return 0;
}
```

```
      } else if(d[u] + w == d[v]) cnt[v]
= (cnt[v] + cnt[u]) % mod;
    }
  }
  return d;
}

int u[N], v[N], w[N];
int32_t main() {
  int s, t; // source, target
  cin >> n >> m >> s >> t;
  for(int i = 1; i <= m; i++) {
    cin >> u[i] >> v[i] >> w[i];
    g[u[i]].push_back({v[i], w[i]});
    r[v[i]].push_back({u[i], w[i]});
  }
  vector<int> cnt1, cnt2;
  auto d1 = dijkstra(s, t, cnt1);
  auto d2 = dijkstra(t, s, cnt2);

  long long ans = d1[t];
  for(int i = 1; i <= m; i++) {
    int x = u[i], y = v[i];
    long long nw = d1[x] + w[i] + d2[y];
    if(nw == ans && 1LL * cnt1[x] *
cnt2[y] % mod == cnt1[t]) cout <<
"YES\n";
    else if(nw - ans + 1 < w[i]) cout <<
"CAN " << nw - ans + 1 << '\n';
    else cout << "NO\n";
  }
  return 0;
}
```

Dynamic Programming

LCS

```
#include<bits/stdc++.h>
using namespace std;

#define endl "\n"
const int N = 3000+128;
string a, b;
int dp[N][N];

int lcs(int i, int j)
{
    if(i >= a.size() or j >= b.size())
return 0;
    if(dp[i][j] != -1) return dp[i][j];
    int ans = lcs(i, j + 1);
    ans = max(ans, lcs(i + 1, j));
    if(a[i] == b[j]) ans = max(ans, lcs(i
+ 1, j + 1) + 1);
    return dp[i][j] = ans;
}
void print_lcs(int i, int j)
{
    if(i >= a.size() or j >= b.size())
                                    return;
    if(a[i] == b[j])
    {
        cout << a[i];
        print_lcs(i + 1, j + 1);
        return;
    }
    int x = lcs(i + 1, j);
    int y = lcs(i, j + 1);
    if(x >= y)
    {
        print_lcs(i + 1, j);
    }
    else
    {
        print_lcs(i, j + 1);
    }
}
int32_t main()
{
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    cin >> a >> b;
    memset(dp, -1, sizeof dp);
    //print_lcs(0, 0);
    cout << lcs(0, 0) << endl;
    return 0;
}
```

Minimum Cost On Matrix

```
#include<bits/stdc++.h>
using namespace std;
#define endl "\n"
const int N = 128, inf = 1e9;
int a[N][N], n, m, dp[N][N];
vector<int> v;
int min_cost(int i, int j)
{
if(i > n or j > m) return inf;
if(i == n and j == m) return a[i][j];
if(dp[i][j] != -1) return dp[i][j];
return dp[i][j] = a[i][j] +
min(min_cost(i+1,j),min_cost(i,j+1));
}
void path(int i, int j)
{
cout<<"(" << i << ", " << j << ") ";
//v.push_back(a[i][j]);
if(i == n and j == m) return;
int right = min_cost(i, j + 1);
int down = min_cost(i + 1, j);
if(right <= down)
{
   path(i, j + 1);
}
else
{
   path(i + 1, j);
}
}
int32_t main()
{
ios_base::sync_with_stdio(0);
cin.tie(0);
cout.tie(0);
cin >> n >> m;
for(int i = 1; i <= n; i++)
{
 for(int j = 1; j <= m; j++)
 {
    cin >> a[i][j];
 }
}
memset(dp, -1, sizeof(dp));
cout << min_cost(1, 1) << endl;
path(1, 1);
//for(auto u : v) cout << u << ' ';
return 0;
}
```

Geometry formulas

Perimeter

Perimeter of a **square**: s + s + s + s
s:length of one side
Perimeter of a **rectangle**: l+w+l+w
l: length
w: width
Perimeter of a **triangle**: a + b + c
a, b, and c: lengths of the 3 sides

Volume

Volume of a **cube**: s × s × s
s: length of one side
Volume of a **box**: l × w × h
l: length
w: width
h: height
Volume of a **sphere**: (4/3) × pi × r^3
pi: 3.14
r: radius of sphere

Volume of a **triangular prism**:
area of triangle × Height = (1/2 base × height) × Height
base: length of the base of the triangle
height: height of the triangle
Height: height of the triangular prism

Volume of a **cylinder**: pi×r^2×Height
pi: 3.14
r: radius of the circle of the base
Height: height of the cylinder

Area

Area of a **square**: s × s
s: length of one side
Area of a **rectangle**: l × w
l: length
w: width
Area of a **triangle**: (b × h)/2
b: length of base
h: length of height
Area of a trapezoid: (b1 + b2) × h/2
b1 and b2 : parallel sides or the bases
h: length of height