

# CP Snippets

[About](#)[Codeforces](#)[GitHub](#)[LinkedIn](#)

## About

A collection of CPP Snippets to aid in competitive programming.

This site was auto generated with the help of [marked](#).

The old version of site is available [here](#).

This site is also available in the form of a PDF book for your convenience, you can download it from [here](#).

---

## BIT-general

- easy BIT general with range updates by diff array too
- <https://thesobersobber.github.io/CP-Snippets/BIT-general>
- [github-snip-file](#)

```
template <class T>
class BIT
{
    static_assert(is_integral<T>::value, "Integer required");

private:
    const size_t N;
    vector<T> data;

public:
    // Binary indexed tree or fenwick tree
```

```

// 0 (log n) all operations except order
// order complexity - 0 (log n)
// 1 based indexing
BIT() : N(0) {}
BIT(const size_t _N) : N(_N), data(_N + 1) {}

size_t size()
{
    return N;
}
// sum of [1, idx]
// range sum query
T sum(size_t idx)
{
    T ans = 0;
    for (; idx > 0; idx -= (idx & -idx))
    {
        ans += data[idx];
    }
    return ans;
}
T sum(size_t l, size_t r)
{
    return sum(r) - sum(l - 1);
}

// Point update
void add(size_t idx, T val)
{
    for (; idx <= N; idx += (idx & -idx))
    {
        data[idx] += val;
    }
}

```

```

    }
}

// Range update
void range_add(size_t l, size_t r, T val)
{
    add(l, val);
    add(r + 1, -val);
}

template <class OStream>
friend OStream &operator<<(OStream &os, BIT &bit)
{
    T prv = 0;
    os << '[';
    for (int i = 1; i <= bit.N; i++)
    {
        T now = bit.sum(i);
        os << now - prv << ', ', prv = now;
    }
    return os << ']';
}
};

```

## DSU

- DSU
- <https://thesobersobber.github.io/CP-Snippets/DSU>
- [github-snip-file](#)

```

class DSU {
private:

```

```

    vector<int> parent, size;
public:
    DSU(int n) {
        parent = vector<int>(n);
        size = vector<int>(n, 1);
        iota(begin(parent), end(parent), 0);
    }

    int getParent(int x) {
        if (parent[x] == x) return x;
        return parent[x] = getParent(parent[x]);
    }

    void join(int x, int y) {
        x = getParent(x);
        y = getParent(y);
        if (size[x] > size[y])
            swap(x, y);
        if (x == y) return;
        parent[x] = y;
        size[y] += size[x];
    }

    int getSize(int x) {
        return size[x] = size[getParent(x)];
    }
};

```

## Efficient-Divisibility-Trick-7-and-13

- used here: <https://codeforces.com/contest/2043/problem/B>

- <https://thesobersobber.github.io/CP-Snippets/> Efficient Div
- [github-snip-file](#)

<https://www.mathematical-gemstones.com/gemstones/pearl/a-faster->

## ExtendedGcdDiophantine

- Diophantine any and all soln
- [https://thesobersobber.github.io/CP-Snippets/Extended GCD D](https://thesobersobber.github.io/CP-Snippets/Extended%20GCD%20D)
- [github-snip-file](#)

```
int _abs(int a) {
    if(a < 0) return -a;
    return a;
}
```

```
void shift_solution(int & x, int & y, int a, int b, int cnt) {
    x += cnt * b;
    y -= cnt * a;
}
```

```
int gcd(int a, int b, int& x, int& y) {
    if (b == 0) {
        x = 1;
        y = 0;
        return a;
    }
    int x1, y1;
    int d = gcd(b, a % b, x1, y1);
    x = y1;
    y = x1 - y1 * (a / b);
}
```

```

    return d;
}

int64_t X, Y;

bool find_any_solution(int a, int b, int c, int &x0, int &y0, int &g) {
    g = gcd(_abs(a), _abs(b), x0, y0);
    if (c % g) {
        return false;
    }

    x0 *= c / g;
    y0 *= c / g;
    if (a < 0) x0 = -x0;
    if (b < 0) y0 = -y0;
    X = (int64_t)x0;
    Y = (int64_t)y0;
    return true;
}

int find_all_solutions(int a, int b, int c, int minx, int maxx,
    int x, y, g) {
    if (!find_any_solution(a, b, c, x, y, g))
        return 0;
    a /= g;
    b /= g;

    int sign_a = a > 0 ? +1 : -1;
    int sign_b = b > 0 ? +1 : -1;

    shift_solution(x, y, a, b, (minx - x) / b);
    if (x < minx)

```

```

    shift_solution(x, y, a, b, sign_b);
if (x > maxx)
    return 0;
int lx1 = x;

shift_solution(x, y, a, b, (maxx - x) / b);
if (x > maxx)
    shift_solution(x, y, a, b, -sign_b);
int rx1 = x;

shift_solution(x, y, a, b, -(miny - y) / a);
if (y < miny)
    shift_solution(x, y, a, b, -sign_a);
if (y > maxy)
    return 0;
int lx2 = x;

shift_solution(x, y, a, b, -(maxy - y) / a);
if (y > maxy)
    shift_solution(x, y, a, b, sign_a);
int rx2 = x;

if (lx2 > rx2)
    swap(lx2, rx2);
int lx = max(lx1, lx2);
int rx = min(rx1, rx2);

if (lx > rx)
    return 0;
return (rx - lx) / _abs(b) + 1;
}

```

```

// EXAMPLE USAGE
// void solve(){
//     int64_t a, b; cin >> a >> b;

//     swap(a, b);

//     if(a == 0) {
//         if(2 % b == 0) {
//             cout << 0 << " " << 2 / b << "
// ";
//             return;
//         }
//     }

//     swap(a, b);

//     if(a == 0) {
//         if(2 % b == 0) {
//             cout << 2 / b << " " << 0 << "
// ";
//             return;
//         }
//     }

//     if(                                     find_all_soluti
//         cout << "-1";
//     }
//     else {
//         cout << X << " " << Y << "
// ";
//     },

```



```
// }  
// }
```

## General-Hash

- General Hash functions that returns two hashes, takes in 0 indexed arr or string, allows hash query on range, beware that this uses the decreasing power convention
- <https://thesobersobber.github.io/CP-Snippets/General-Hash>
- [github-snip-file](#)

```
struct PolyHash {  
    /*  
        WARNING: make sure the values in the array or string are  
        */  
    vector<long long> powers;  
    vector<long long> powers2;  
    vector<long long> hashes;  
    vector<long long> hashes2;  
    long long seed = 500002961;  
    long long seed2 = 500003263;  
    const long long mod = (long long)1e9 + 7;  
    const long long mod2 = 998244353;  
    vector<long long> arr;  
    void init(long long n){  
        powers.resize(n + 5);  
        powers[0] = 1;  
        powers2.resize(n + 5);  
        powers2[0] = 1;  
        hashes.resize(n + 5);  
        hashes[0] = arr[0];  
    }  
};
```

```

        hashes2.resize(n + 5);
        hashes2[0] = arr[0];
        for (long long i = 1; i <= n; i++){
            powers[i] = powers[i - 1] * seed;
            powers[i] %= mod;
            powers2[i] = powers2[i - 1] * seed2;
            powers2[i] %= mod2;
        }
        for (long long i = 1; i <= n; i++){
            hashes[i] = hashes[i - 1] * seed + arr[i];
            hashes[i] %= mod;
            hashes2[i] = hashes2[i - 1] * seed2 + arr[i];
            hashes2[i] %= mod2;
        }
    }
    void init(long long n, string s){ //string is 0 indexed
        arr.resize(n + 5);
        for (long long i = 1; i <= n; i++){
            arr[i] = s[i - 1];
        }
        init(n);
    }
    void init(long long n, vector<long long> a){ //a is 0 index
        arr.resize(n + 5);
        for (long long i = 1; i <= n; i++){
            arr[i] = a[i - 1];
        }
        init(n);
    }
    // returns hash like a1 a2 a3 a4 a5 a6 a7 a8 a9 a10
    // 2,5 query will yeild:  $a_2 * p^3 + a_3 * p^2 + a_4 * p^1 + a_5$  and
    // no need of power combi manually

```

```

    pair<long long, long long> subhash(long long l, long long r
        long long hsh = hashes[r] - hashes[l - 1] * powers[r -
        hsh += mod;
        hsh %= mod;
        long long hsh2 = hashes2[r] - hashes2[l - 1] * powers2[
        hsh2 += mod2;
        hsh2 %= mod2;
        return {hsh, hsh2};
    }
};

// Example Usage:
// PolyHash hsh;
// int n = word.size();
// hsh.init(n,word);
// subhash is inclusive of l and r remember that

```

## Inline-break

- Inline Break by defining break to a function that basically breaks but returns value and hence can be used as a expression with comma operator
- [https://thesobersobber.github.io/CP-Snippets/Inline Break](https://thesobersobber.github.io/CP-Snippets/Inline%20Break)
- [github-snip-file](#)

```

static inline int break_(void) {
    extern void abort(void);
    abort();
    return 0;
}
#define break break_

```

```
int main(int c, char **v) {
    int yay = 1;
    for (int i = 0; ++i < c;)
        if (v[i][0] == '1') yay = 0, break;
    return yay;
}
```

## LCA-RMQ

- LCA using Range Min Query(Sparse) on ETT ([Reference](#))
- <https://thesobersobber.github.io/CP-Snippets/LCA-RMQ>
- [github-snip-file](#)

```
template<class T>
struct RMQ{
    int n, logn;
    vector<vector<int>> b;
    vector<T> A;
    void build(const vector<T> &a) {
        A = a, n = sz(a);
        logn = 32 - __builtin_clz(n);
        b.resize(logn, vi(n));
        iota(all(b[0]), 0);
        for(int i = 1; i < logn ; i++){
            for(int j = 0; j < n ; j++){
                b[i][j] = b[i - 1][j];
                if(j + (1 << (i - 1)) < n && A[b[i - 1][j + (1
                    b[i][j] = b[i - 1][j + (1 << (i - 1))]);
            }
        }
    }
}
```

```

int rangeMin(int x, int y){
    int k = 31 - __builtin_clz(y - x + 1);
    return min(A[b[k][x]], A[b[k][y - (1 << k) + 1]]);
}
int minIndx(int x, int y){
    int k = 31 - __builtin_clz(y - x + 1);
    return A[b[k][x]] < A[b[k][y - (1 << k) + 1]] ? b[k][x]
}
};

struct LCA {
    vector<int> tour, Findx, dpth;
    RMQ<int> rmq;
    void build(const vector<vector<int>> &adj, int src) {
        vector<bool> vis(sz(adj));
        vector<int> dpth1(sz(adj));
        function<void(int, int)> dfs = [&](int i, int d) {
            tour.pb(i);
            vis[i] = 1;
            dpth1[i] = d;
            for(auto &u: adj[i]) if(!vis[u]) dfs(u, d + 1),
};
        dfs(src, 0);
        Findx.resize(sz(adj));
        dpth.resize(sz(tour));
        for(int i = sz(tour) - 1 ; i >= 0 ; i--) {
            dpth[i] = dpth1[tour[i]], Findx[tour[i]] = i;
        }
        rmq.build(dpth);
    }
    int lca(int x, int y) {
        x = Findx[x], y = Findx[y];

```

```

        if(x > y)        swap(x, y);
        return tour[rmq.minIndx(x, y)];
    }
    int dist(int x, int y) {
        x = Findx[x], y = Findx[y];
        if(x > y)        swap(x, y);
        return dpth[x] + dpth[y] - 2 * rmq.rangeMin(x, y);
    }
};

```

## PrefSum2D

- 2D Prefix Sum
- <https://thesobersobber.github.io/CP-Snippets/PrefSum2D>
- [github-snip-file](#)

```

struct PrefSum2D {
    vector<vector<ll>> pref; int n; PrefSum2D() {}
    PrefSum2D(vector<vector<ll>> &_v) : n(_v.size()), pref(_v)
        for (int j = 1; j < n; j++) { pref[0][j] += pref[0][j - 1]; }
        for (int i = 1; i < n; i++) { pref[i][0] += pref[i - 1][0]; }
        for (int i = 1; i < n; i++)      for (int j = 1; j < n; j++)
            pref[i][j] += pref[i - 1][j] + pref[i][j - 1] - pref[i - 1][j - 1];
    }
    ll query(int xTL, int yTL, int xBR, int yBR) {
        ll res = pref[xBR][yBR];
        if (xTL) { res -= pref[xTL - 1][yBR]; }
        if (yTL) { res -= pref[xBR][yTL - 1]; }
        if (xTL && yTL) { res += pref[xTL - 1][yTL - 1]; }
        return res;
    }
};

```

```
    }  
};
```

## Segtree-General

- General segtree, needs node struct (with members def and epsilon(default) for all of them) and operation lambda (merge)
- <https://thesobersobber.github.io/CP-Snippets/Segtree-General>
- [github-snip-file](#)

```
template <typename T>  
class segtree  
{  
public:  
    // 0 based indexing  
    // def= default value  
    vector<T> t;  
    int n;  
    T def;  
    function<T(T, T)> merge;  
    void build(int _n, T _def, function<T(T, T)> _fx)  
    {  
        n = _n;  
        def = _def;  
        merge = _fx;  
        t.assign(n * 2, def);  
        for (int i = n - 1; i; i--)  
            t[i] = merge(t[i * 2], t[i * 2 + 1]);  
    }  
    void build(vector<T> &a, T _def, function<T(T, T)> _fx)  
    {
```

```

    n = a.size();
    def = _def;
    merge = _fx;
    t.assign(n * 2, def);
    for (int i = 0; i < n; i++)
        t[i + n] = T(a[i]);
    for (int i = n - 1; i; i--)
        t[i] = merge(t[i * 2], t[i * 2 + 1]);
}
void update(int i, T v)
{
    for (t[i += n] = T(v); i;)
    {
        i /= 2;
        t[i] = merge(t[i * 2], t[i * 2 + 1]);
    }
}
// this query is made on [l, r]
T query(int l, int r)
{
    T lans = def, rans = def;
    for (l += n, r += n + 1; l < r; l /= 2, r /= 2)
    {
        if (l % 2)
            lans = merge(lans, t[l++]);
        if (r % 2)
            rans = merge(t[--r], rans);
    }
    return merge(lans, rans);
}
};

```



```
// demo usage
struct node
{
    int val;
    node(int x)
    {
        val = x;
    }
    // default value
    node()
    {
        val = 1e18;
    }
};

segtree<node> seg;
seg.build(n + 1, node(), [&](node x, node y){ return node(min(x
```

## Simpler-Segtree

- Init with an array simply using the build fn, customize operation and epsilon in the struct itself, supports point updates and range queries
- <https://thesobersobber.github.io/CP-Snippets/Simpler-Segtree>
- [github-snip-file](#)

```
struct segtree {
    vector<int> t;
    int emptyans = -1e18;
    int n;
    int op(int a, int b){
        return max(a, b); // custom operation
    }
};
```

```

}
int construct(int v, int l, int r, vi &a){
    if(l == r){
        t[v] = a[l];
        return t[v];
    }
    int mid = (r + l)/2;
    return t[v] = op(construct(2*v+1, l, mid, a), construct
}
void build(vi &a){
    n = a.size();
    t = vector<int> (4*n);
    construct(0, 0, n-1, a);
}
int queryans(int v, int curl, int curr, int l, int r){
    if(curl >= l && curr <= r){
        return t[v];
    }
    if(curr < l || curl > r){
        return emptyans;
    }
    int mid = (curl + curr)/2;
    return op(queryans(2*v+1, curl, mid, l, r), queryans(2*
}
int query(int l, int r){
    return queryans(0, 0, n-1, l, r);
}
int updateval(int v, int i, int x, int l, int r){
    if(r < i || l > i){
        return t[v];
    }
    if(l == r && l == i){

```

```

        return t[v] = x;
    }
    int mid = (r + l)/2;
    return t[v] = op(updateval(2*v+1, i, x, l, mid), updateval(2*v+2, i, x, mid+1, r));
}
void update(int i, int x){
    updateval(0, i, x, 0, n-1);
}
};

```

## Sparse-General

- General Implementation of Sparse table with the template structure
- <https://thesobersobber.github.io/CP-Snippets/Sparse-General>
- [github-snip-file](#)

```

template<class T>
class sparseTable
{
public:
    int n,k;
    vector<vector<T>> table;
    vector<T> logs;
    function<T(T,T)> operation;
    void init(int x,function<T(T,T)> _operation)
    {
        operation=_operation;
        n=x;
        logs.resize(n+1);
        logs[1]=0;
        for(int i=2;i<=n;i++)

```

```

        logs[i]=logs[i/2]+1;
        k=*max_element(logs.begin(),logs.end());
        table.resize(k+1,vector<T>(n));
    }

    void build(vector<T> &arr)
    {
        for(int i=0;i<n;i++)
            table[0][i]=arr[i];

        for(int j=1;j<=k;j++)
        {
            for(int i=0;i+(1<<j)<=n;i++)
                table[j][i]=operation(table[j-1][i],table[j-1][i+(1<<j)]);
        }
    }

    // 1 based indexing
    T query(int l , int r)
    {
        assert(l<=r);
        assert(l>=0 && r<n);
        int j = logs[r - l + 1];
        T answer = operation(table[j][l], table[j][r-(1<<j)+1]);
        return answer;
    }
};

```

// does not have a constructor, make an instance and then use it

## Weird\_Lazy\_Segtree

- A lazy segtree taken from a abc340 E mridulahi submission, it's supposed to be able to do range updates and point queries
- [https://thesobersobber.github.io/CP-Snippets/Lazy Segtree](https://thesobersobber.github.io/CP-Snippets/Lazy%20Segtree)
- [github-snip-file](#)

```
// I can see a merge operation but not default values where to  
#include<bits/stdc++.h>
```

```
using namespace std;
```

```
#define all(x) begin(x), end(x)  
#define sz(x) static_cast<int>((x).size())  
#define int long long
```

```
const int INF = 1e18;
```

```
struct lazy {  
    int val, lazyy;  
};
```

```
struct SegtreeLazy {  
    int size;  
    vector<lazy> val;  
    void init (int n) {  
        size = 1;  
        while (size < n) size *= 2;  
        val.resize (2 * size - 1);  
    }
```

```
    lazy merge (int x, int y) {
```

```

        return {min (val[x].val, val[y].val), 0};
    }

    void propagate (int x) {
        val[2 * x + 1].val += val[x].lazy;
        val[2 * x + 2].val += val[x].lazy;
        val[2 * x + 1].lazy += val[x].lazy;
        val[2 * x + 2].lazy += val[x].lazy;
        val[x].lazy = 0;
    }

    void build (vector<int> &a, int x, int lx, int rx) {
        if (rx - lx == 1) {
            if (lx < sz(a)) val[x] = {a[lx], 0};
            else val[x] = {INF, 0};
            return;
        }
        int m = (lx + rx) / 2;
        build (a, 2 * x + 1, lx, m);
        build (a, 2 * x + 2, m, rx);
        val[x] = merge (2 * x + 1, 2 * x + 2);
    }

    void build (vector<int> &a) {
        build (a, 0, 0, size);
    }

    void RangeUpdate (int l, int r, int x, int lx, int rx,
        if (rx - lx == 1) {
            val[x].val += v;
            return;
        }
    }

```

```

        if (lx >= l && rx <= r) {
            val[x].val += v;
            val[x].lazy += v;
            return;
        }
        int m = (lx + rx) / 2;
        propagate (x);
        if (m > l) {
            RangeUpdate (l, r, 2 * x + 1, lx, m, v)
        }
        if (m < r) {
            RangeUpdate (l, r, 2 * x + 2, m, rx, v)
        }
        val[x] = merge (2 * x + 1, 2 * x + 2);
    }

    void update (int l, int r, int v) {
        if (r <= l) return;
        RangeUpdate (l, r, 0, 0, size, v);
    }

    int get (int l, int r, int x, int lx, int rx) {
        if (rx - lx == 1) {
            return val[x].val;
        }
        if (lx >= l && rx <= r) {
            return val[x].val;
        }
        int m = (lx + rx) / 2;
        propagate (x);
        int a1 = INF, a2 = INF;

```





```

//                                     seg.init
//                                     seg.buil

//         for (auto i : b) {
//             int x = seg.get(i, i + 1);
//             int y = (i + 1) % n, z = (i + x) % n;
//             if (y <= z) {
//                                     seg.upda
//             }
//             else {
//                                     seg.upda
//                                     seg.upda
//             }
//                                     seg.upda
//                                     seg.upda
//         }

//         for (int i = 0; i < n; i++) cout <<         seg.get(
// }

```

## Xor-Of-First-N-Natural-Numbers

- based on the fact that if  $n \% 4 == 3$  then xor till n from 1 is 0
- [https://thesobersobber.github.io/CP-Snippets/Xor of first N](https://thesobersobber.github.io/CP-Snippets/Xor%20of%20first%20N)
- [github-snip-file](#)

```

auto xorOfFirstN = [&](int n) -> int {
    // n%4 = 3 => xor = 0
    int t = n, ans = 0;
    while(t%4!=3) ans^=t, --t;

```

```
        return ans;
};

// more explicit version
int computeXOR(int n) {
    if (n % 4 == 0) return n;
    if (n % 4 == 1) return 1;
    if (n % 4 == 2) return n + 1;
    return 0;
}

// source: https://www.geeksforgeeks.org/calculate-xor-1-n/
```

## arr-inp

- arr-inp
- <https://thesobersobber.github.io/CP-Snippets/arr-inp>
- [github-snip-file](#)

```
vector<int> a(n, 0);
for(int i=0;i<n;i++) cin>>a[i];
```

## arr-pref

- arr-pref
- <https://thesobersobber.github.io/CP-Snippets/arr-pref>
- [github-snip-file](#)

```
vector<int> pre(n, 0);
for(int i=1;i<n;i++) pre[i]=a[i]+pref[i-1];
```

## bfs-dist

- bfs that measures levels/dist
- <https://thesobersobber.github.io/CP-Snippets/bfs-dist>
- [github-snip-file](#)

```
queue<int> q;
vector<int> dist, visG(n+1, 0);
q.push(1); visG[1]=1;
while(!q.empty()){
    int curr = q.front();
    q.pop();
    for(auto i: g[curr]){
        if(!visG[i]) continue;
        dist[i] = dist[curr] + 1;
        q.push(i);
    }
}
```

## binpow

- binpow
- <https://thesobersobber.github.io/CP-Snippets/binpow>
- [github-snip-file](#)

```
ll binpow(ll x, ll y){
    ll res = 1;
    while (y>0){
        if (y&1) res = (ll)(res*x);
        y = y>>1;
    }
```

```

        x = (ll)(x*x);
    }
    return res;
}

```

## binsearch

- binsearch
- <https://thesobersobber.github.io/CP-Snippets/binsearch>
- [github-snip-file](#)

```

int lo = 0, hi = n-1;
while(hi-lo>1) {
    int mid = lo + ((hi-lo) >> 1);
    // if condition true toh bas right segment mai search hoga,
    auto check = [&](ll mid) {
        return (/*condition here*/);
    };
    if(check(mid)){
        // do stuff here
        lo = mid;
    }
    else {
        hi = mid;
    }
}

```

## bp-small

- bp-small
- <https://thesobersobber.github.io/CP-Snippets/bp-small>

- [github-snip-file](#)

```
#include <bits/stdc++.h>
#ifdef ONLINE_JUDGE
#include "debug.h"
#pragma GCC optimize("O3,unroll-loops")
#pragma GCC target("avx2,bmi,bmi2,lzcnt,popcnt")
#else
#define dbg(x...) "11-111"
#endif

using namespace std;

#define ll long long
#define int long long // because mai bevakoof hu

constexpr int mod = 1e9+7;
// constexpr int mod = 998244353;
constexpr int maxn = 1e6+5;

// pows
inline ll po(ll a, ll b) { ll res = 1; for (; b; b >>= 1) { if
inline ll modpow(ll a, ll b, ll mod) { ll res = 1; for (; b; b

void pre_process(){

}

int solve(){
    int n; cin>>n;
    dbg(n);
    return 2*n;
```

```

}
int32_t main(){
    ios_base::sync_with_stdio(0);
    cin.tie(0); cout.tie(0);
    pre_process();
    int t; cin>>t;
    while(t-->0) cout<<solve()<<'
';
}

```

## bp

- bp
- <https://thesobersobber.github.io/CP-Snippets/bp>
- [github-snip-file](#)

```

#include <bits/stdc++.h>
#ifdef ONLINE_JUDGE
#include "debug.h"
#pragma GCC optimize("O3,unroll-loops")
#pragma GCC target("avx2,bmi,bmi2,lzcnt,popcnt")
#else
#define dbg(x...) "11-111"
#endif

using namespace std;

#define ll long long
#define int long long // because mai bevakoof hu
#define logCont(arr,f,l) { auto start=arr.begin(), end=arr.be
"; }

```

```

#define uniq(x) x.erase(unique(all(x)),x.end());
#define tr(s, args...) transform(s.begin(), s.end(), args)
#define sz(x) (ll)x.size()

// variadic lambda
#define f(u, args...) [&](auto &&u) { return args; }
#define g(u, v, args...) [&](auto &&u, auto &&v) { return args

// precesion
#define precise(n) cout<<fixed<<setprecision((n))
// bits
#define bpc(n) std::popcount((unsigned long long)(n))
#define hsb(n) std::has_single_bit((unsigned long long)(n))
#define MSB(n) std::bit_floor((unsigned long long)(n))
#define ctz(n) ((n) ? __builtin_ctzll((unsigned long long)(n)) : 0)
#define clz(n) ((n) ? __builtin_clzll((unsigned long long)(n)) : 64)
#define LSB(n) ((n)&(-(n)))

// general amax, amin for any ds, to be able to use swap in gra
template<typename T,typename T1> inline bool amax(T &a,T1 b){ if (a<b) swap(a,b); return a; }
template<typename T,typename T1> inline bool amin(T &a,T1 b){ if (a>b) swap(a,b); return a; }

// comparison struct for maps (or use decltype)
template<typename T> struct Comp { bool operator()(const T& l, const T& r) const { return l<r; } };

constexpr ll Inf = 4e18;
constexpr int mod = 1e9+7;
// constexpr int mod = 998244353;
constexpr int maxn = 1e6+5;

// sasta mint
ll inv(ll i) {if (i == 1) return 1; return (mod - (mod / i) *

```

```

ll mod_mul(ll a, ll b) {a = a % mod; b = b % mod; return ((a *
ll mod_add(ll a, ll b) {a = a % mod; b = b % mod; return ((a +
ll gcd(ll a, ll b) { if (b == 0) return a; return gcd(b, a % b)
ll ceil_div(ll a, ll b) {return a % b == 0 ? a / b : a / b + 1;
ll pwr(ll a, ll b) {a %= mod; ll res = 1; while (b > 0) {if (b

// pows
inline ll po(ll a, ll b) { ll res = 1; for (; b; b >>= 1) { if
inline ll modpow(ll a, ll b, ll mod) { ll res = 1; for (; b; b

void pre_process(){

}

int solve(){
    int n; cin>>n;
    dbg(n);
    return 2*n;
}

int32_t main(){
    ios_base::sync_with_stdio(0);
    cin.tie(0); cout.tie(0);
    pre_process();
    int t; cin>>t;
    while(t--) cout<<solve()<<'
';
}

```

## clock\_for\_TL

- clock



- [https://thesobersobber.github.io/CP-Snippets/clock\\_for\\_TL](https://thesobersobber.github.io/CP-Snippets/clock_for_TL)
- [github-snip-file](#)

```
auto start = chrono::high_resolution_clock::now();
// code goes here
auto stop = chrono::high_resolution_clock::now();
auto duration = chrono::duration_cast<chrono::milliseconds>(stop - start);
cout << duration.count() << " ms\n";
```

## combi-mint

- combi template with mint
- <https://thesobersobber.github.io/CP-Snippets/combi-mint>
- [github-snip-file](#)

```
const int mod=1e9+7;
struct mi {
    int64_t v; explicit operator int64_t() const { return v % mod; }
    mi() { v = 0; }
    mi(int64_t _v) {
        v = (-mod < _v && _v < mod) ? _v : _v % mod;
        if (v < 0) v += mod;
    }
    friend bool operator==(const mi& a, const mi& b) {
        return a.v == b.v; }
    friend bool operator!=(const mi& a, const mi& b) {
        return !(a == b); }
    friend bool operator<(const mi& a, const mi& b) {
        return a.v < b.v; }
```

```

mi& operator+=(const mi& m) {
    if ((v += m.v) >= mod) v -= mod;
    return *this; }
mi& operator-=(const mi& m) {
    if ((v -= m.v) < 0) v += mod;
    return *this; }
mi& operator*=(const mi& m) {
    v = v*m.v%mod; return *this; }
mi& operator/=(const mi& m) { return (*this) *= inv(m); }
friend mi pow(mi a, int64_t p) {
    mi ans = 1; assert(p >= 0);
    for (; p; p /= 2, a *= a) if (p&1) ans *= a;
    return ans;
}
friend mi inv(const mi& a) { assert(a.v != 0);
    return pow(a,mod-2); }

mi operator-() const { return mi(-v); }
mi& operator++() { return *this += 1; }
mi& operator--() { return *this -= 1; }
mi operator++(int32_t) { mi temp; temp.v = v++; return temp; }
mi operator--(int32_t) { mi temp; temp.v = v--; return temp; }
friend mi operator+(mi a, const mi& b) { return a += b; }
friend mi operator-(mi a, const mi& b) { return a -= b; }
friend mi operator*(mi a, const mi& b) { return a *= b; }
friend mi operator/(mi a, const mi& b) { return a /= b; }
friend ostream& operator<<(ostream& os, const mi& m) {
    os << m.v; return os;
}
friend istream& operator>>(istream& is, mi& m) {
    int64_t x; is >> x;
    m.v = x;
}

```

```

        return is;
    }
    friend void __print(const mi &x) {
        cerr << x.v;
    }
};

const int maxn=2e5+5;
vector<mi> fct(maxn, 1), invf(maxn, 1);
void calc_fact() {
    for(int i = 1 ; i < maxn ; i++) {
        fct[i] = fct[i - 1] * i;
    }
    invf.back() = mi(1) / fct.back();
    for(int i = maxn - 1 ; i ; i--)
        invf[i - 1] = i * invf[i];
}

mi choose(int n, int r) { // choose r elements out of n element
    if(r > n)    return mi(0);
    assert(r <= n);
    return fct[n] * invf[r] * invf[n - r];
}

```

## combi-struct

- combi-struct
- <https://thesobersobber.github.io/CP-Snippets/combi-struct>
- [github-snip-file](#)

```

struct Comb {
    int n;

```

```

std::vector<int> _fac;
std::vector<int> _invfac;
std::vector<int> _inv;

Comb() : n{0}, _fac{1}, _invfac{1}, _inv{0} {}
Comb(int n) : Comb() {
    init(n);
}

void init(int m) {
    if (m <= n) return;
    _fac.resize(m + 1);
    _invfac.resize(m + 1);
    _inv.resize(m + 1);

    for (int i = n + 1; i <= m; i++) {
        _fac[i] = _fac[i - 1] * i;
    }
    _invfac[m] = _fac[m].inv();
    for (int i = m; i > n; i--) {
        _invfac[i - 1] = _invfac[i] * i;
        _inv[i] = _invfac[i] * _fac[i - 1];
    }
    n = m;
}

int fac(int m) {
    if (m > n) init(2 * m);
    return _fac[m];
}

int invfac(int m) {
    if (m > n) init(2 * m);

```

```

        return _invfac[m];
    }
    int inv(int m) {
        if (m > n) init(2 * m);
        return _inv[m];
    }
    int binom(int n, int r) {
        if (n < r || r < 0) return 0;
        return fac(n) * invfac(r) * invfac(n - r);
    }
};

```

## combination-non-mod

- combination-non-mod
- <https://thesobersobber.github.io/CP-Snippets/combination-no>
- [github-snip-file](#)

```

vector<vector<int>> dp(n+1, vector<int> (k+1));
int binomCoeff(int n, int k){
    for (int i=0; i<=n; i++){
        for (int j=0; j<=k; j++){
            if (!j || j == i) dp[i][j] = 1;
            // binomial coefficient approach
            else dp[i][j] = dp[i - 1][j - 1] + dp[i - 1][j];
        }
    }
    return dp[n][k];
}

```

## combination-small

- combination-small
- <https://thesobersobber.github.io/CP-Snippets/combination-sm>
- [github-snip-file](#)

```
int C(int n,int r){
    r = min(r,n-r);
    int ans = 1;
    for(int i=1;i<=r;i++,n--){
        ans *=n;
        ans/=i;
    }
    return ans;
}
```

## combination

- combination
- <https://thesobersobber.github.io/CP-Snippets/combination>
- [github-snip-file](#)

```
int C(int n, int r){
    int v = (fac[n] * inv[r])%mod;
    v = (v * inv[n-r])%mod;

    return v;
}
```

## crt

- crt

- <https://thesobersobber.github.io/CP-Snippets/crt>
- [github-snip-file](#)

```
/**
 * Chinese remainder theorem.
 * Find z such that  z % x[i] = a[i] for all i.
 * */
long long crt(vector<long long> &a, vector<long long> &x) {
    long long z = 0;
    long long n = 1;
    for (int i = 0; i < x.size(); ++i)
        n *= x[i];

    for (int i = 0; i < a.size(); ++i) {
        long long tmp = (a[i] * (n / x[i])) % n;
        tmp = (tmp * mod_inv(n / x[i], x[i])) % n;
        z = (z + tmp) % n;
    }

    return (z + n) % n;
}
```

## cute-lcm

- $[a,b,c]=abc(a,b,c)/(a,b)(b,c)(c,a)$ , where  $[]$ =lcm and  $()$ =gcd or  $[a,b,c]=abc/\gcd(ab,bc,ca)$
- <https://thesobersobber.github.io/CP-Snippets/cute-lcm>
- [github-snip-file](#)

"<https://math.stackexchange.com/questions/1579/n-ary-version-of>  
"N-ary versions of gcd and lcm"

"proof is heavy lattice ordered smthing based or use inclusion

## derangments

- derangments
- <https://thesobersobber.github.io/CP-Snippets/derangments>
- [github-snip-file](#)

```
int countDerangements(int n){
    int dp[n + 1];
    if (n < 3) return (dp[n]=(n % 2)?1:0);
    dp[0] = 1, dp[1] = 0, dp[2] = 1;
    for (int i=3; i< n; i++) dp[i] = (i-1)*(dp[i-1]+dp[i-2]);
    return dp[n];
}
```

## dfs-full

- dfs with lots of stuff implemented
- <https://thesobersobber.github.io/CP-Snippets/dfs-full>
- [github-snip-file](#)

```
auto dfs = [&](auto &&dfs, int curr, int parent, vector<int> &v
    for(auto i: adj[curr]){
        if(visPath[i]) cycle_directed|=1;
        if(i==parent || visG[i]) continue;
        dfs(dfs, i, curr, visG, visPath, comp, cycle_directed,
            topo.push(i);
    }
};
```



```

int cnt_comp=0;
vector<int> visG(n+1, 0), visPath(n+1, 0), comp;
vector<vector<int>> components;
stack<int> topo;
bool cycle_directed=0;
for(int i=1; i<=n; i++){
    if(visG[i]) continue;
    visG[i]=visPath[i]=1;
    comp.push_back(i);
    dfs(dfs, 1, -1, visG, visPath, comp, cycle_directed, topo,
    components.push_back(comp);
    comp.clear();
    visPath.assign(n+1, 0);
    cnt_comp++;
}

```

## dfs

- weird ass dfs
- <https://thesobersobber.github.io/CP-Snippets/dfs>
- [github-snip-file](#)

```

map<int,int> dfs(int cur,int par,vi&a){
    // stuff
    for(auto child:adj[cur]){
        if(child==par)continue;
        // stuff
        dfs(child,cur,a);
        // or return smthing and use it
        auto smthing = dfs(child,cur,a);
        // stuff
    }
}

```

```

    }
    // stuff and then return smthing or not, meh
    return cur_prime;
}

```

## diophantine

- linear diophantine
- <https://thesobersobber.github.io/CP-Snippets/diophantine>
- [github-snip-file](#)

```

long long gcd(long long a, long long b, long long &x, long long &y) {
    if (a == 0) {
        x = 0;
        y = 1;
        return b;
    }
    long long x1, y1;
    long long d = gcd(b % a, a, x1, y1);
    x = y1 - (b / a) * x1;
    y = x1;
    return d;
}

```

```

bool find_any_solution(long long a, long long b, long long c, long long &x0, long long &y0, long long &g) {
    g = gcd(abs(a), abs(b), x0, y0);
    if (c % g) {
        return false;
    }
}

```

```

    x0 *= c / g;
    y0 *= c / g;
    if (a < 0) x0 = -x0;
    if (b < 0) y0 = -y0;
    return true;
}

void shift_solution(long long &x, long long &y, long long a, long long b, long long cnt) {
    x += cnt * b;
    y -= cnt * a;
}

long long find_all_solutions(long long a, long long b, long long c, long long minx, long long maxx, long long miny, long long maxy) {
    long long x, y, g;
    if (!find_any_solution(a, b, c, x, y, g)) return 0;
    a /= g;
    b /= g;

    long long sign_a = a > 0 ? +1 : -1;
    long long sign_b = b > 0 ? +1 : -1;

    shift_solution(x, y, a, b, (minx - x) / b);
    if (x < minx) shift_solution(x, y, a, b, sign_b);
    if (x > maxx) return 0;
    long long lx1 = x;

    shift_solution(x, y, a, b, (maxx - x) / b);
    if (x > maxx) shift_solution(x, y, a, b, -sign_b);
    long long rx1 = x;

```

```

shift_solution(x, y, a, b, -(miny - y) / a);
if (y < miny) shift_solution(x, y, a, b, -sign_a);
if (y > maxy) return 0;
long long lx2 = x;

shift_solution(x, y, a, b, -(maxy - y) / a);
if (y > maxy) shift_solution(x, y, a, b, sign_a);
long long rx2 = x;

if (lx2 > rx2) swap(lx2, rx2);
long long lx = max(lx1, lx2);
long long rx = min(rx1, rx2);

if (lx > rx) return 0;
return (rx - lx) / abs(b) + 1;
}

```

## dsu-rr

- dsu-rr
- <https://thesobersobber.github.io/CP-Snippets/dsu-rr>
- [github-snip-file](#)

```

class Solution {
    struct DSU
    {
        vector<int> siz,parent;
        void init()
        {
            siz.resize(26);

```

```

        parent.resize(26);
        for(int i=0;i<26;i++)
        {
            siz[i]=1;
            parent[i]=i;
        }
    }
    int leader(int ex)
    {
        if(ex==parent[ex])
            return ex;
        return parent[ex]=leader(parent[ex]);
    }
    void merge(int a,int b)
    {
        a=leader(a);
        b=leader(b);
        if(a==b)
            return;
        if(siz[a]<siz[b])
            swap(a,b);
        siz[a]+=siz[b];
        parent[b]=parent[a];
    }
};

```

## easy\_seive

- `easy_seive`
- [https://thesobersobber.github.io/CP-Snippets/easy\\_seive](https://thesobersobber.github.io/CP-Snippets/easy_seive)
- [github-snip-file](#)

```

void ez_seive(int n){
    vector<bool> prime(n,1);
    for (int p = 2; p*p <= n; p++){
        if (prime[p]){
            for (int i = p * p; i <= n; i += p) prime[i] = false;
        }
    }
}

for (int p = 2; p <= n; p++){
    // do whatever you want with those primes
    if (prime[p]) cout << p << " ";
}

```

## euclid

- euclid
- <https://thesobersobber.github.io/CP-Snippets/euclid>
- [github-snip-file](#)

```

int euclid_gcd(int a, int b){
    if (b==0) return a;
    return gcd(b, a % b);
}

```

```

int euclid_gcdExtended(int a, int b, int *x, int *y){
    if (a == 0){
        *x = 0;
        *y = 1;
        return b;
    }
    int x1, y1;

```

```

    int gcd = gcdExtended(b % a, a, &x1, &y1);
    *x = y1 - (b / a) * x1;
    *y = x1;
    return gcd;
}

```

## explanation\_binsearch

- explanation\_binsearch
- [https://thesobersobber.github.io/CP-Snippets/explanation\\_binsearch.html](https://thesobersobber.github.io/CP-Snippets/explanation_binsearch.html)
- [github-snip-file](#)

```

int lo = 0, hi = n-1; // see constraints for lo and hi, nahi mi
while(hi-lo>1) {
    int mid = lo + ((hi-lo) >> 1); // to avoid overflows
    // lo will become the last index that satisfies X condition
    // hi is the first element that doesn't satisfy X condition
    // lower_bound = <
    // upper_bound = <=
    // upper using lower = lo, < + ek for loop to traverse the

    // essence ->
    // remember, lo ke left mai condition always true, lo last
    // hi ke right mai condition always false, hi first one jis
    // hi will probably be the answer in most cases
    // hi+1, lo, lo-1 are also potential answers (maybe, mujhe

    // always make condition such that when it's true, left seg
    // if condition true toh bas right segment mai search hoga,
    auto check = [&](ll mid) {

```

```

        // this is where majority is what you wanna write happens
        return (/*condition here*/);
    };
    if(check(mid)){
        // do stuff here
        lo = mid;
    }
    else {
        hi = mid;
    }
}

```

## fac

- fac
- <https://thesobersobber.github.io/CP-Snippets/fac>
- [github-snip-file](#)

```

int fac[maxn];
int inv[maxn];
fac[1] = inv[1] = 1;
for (int i=2; i<maxn; i++){
    fac[i] = (fac[i-1] * i)%mod;
    inv[i] = power(fac[i], mod - 2);
}

```

## factorization

- factorization
- <https://thesobersobber.github.io/CP-Snippets/factorization>



- [github-snip-file](#)

```
void printFactors(int n) {
    for (int i=1; i * i<=n; i++){
        if (n%i == 0) {
            if (n/i == i) cout << i << " ";
            else cout << i << " " << n/i << " ";
        }
    }
    cout << "
";
}
```

```
void printPrimeFactors(int n){
    set<int> f;
    for (int i = 2; i*i <= n; i++){
        while (n % i == 0){
            f.insert(i);
            n /= i;
        }
    }
    for (auto &i : f){
        cout << i << " ";
    }
    cout << "
";
}
```

## fenwick

- binary indexed tree
- <https://thesobersobber.github.io/CP-Snippets/fenwick>

- [github-snip-file](#)

```
// 0-indexed BIT (binary indexed tree / Fenwick tree) (i : [0,
template <class T>
struct BIT{
    int n;
    vector<T> data;
    BIT(int len = 0) : n(len), data(len) {}
    void reset() { fill(data.begin(), data.end(), T(0)); }
    void add(int pos, T v){
        // a[pos] += v
        pos++;
        while (pos > 0 and pos <= n)
            data[pos - 1] += v, pos += pos & -pos;
    }
    T sum(int k) const{
        // a[0] + ... + a[k - 1]
        T res = 0;
        while (k > 0)
            res += data[k - 1], k -= k & -k;
        return res;
    }

    T sum(int l, int r) const { return sum(r) - sum(l); } // a[
// dbg functions
template <class OStream>
friend OStream &operator<<(OStream &os, const BIT &bit){
    T prv = 0;
    os << '[';
    for (int i = 1; i <= bit.n; i++){
        T now = bit.sum(i);
        os << now - prv << ', ', prv = now;
```

```
    }
    return os << ']';
}
};
```

## file\_io

- for coding competetions
- [https://thesobersobber.github.io/CP-Snippets/file\\_io](https://thesobersobber.github.io/CP-Snippets/file_io)
- [github-snip-file](#)

```
void file_i_o(){
    freopen("./tests/test01.txt", "r", stdin);
    freopen("./tests/output01.txt", "w", stdout);
}
```

## freq-map

- freq-map
- <https://thesobersobber.github.io/CP-Snippets/freq-map>
- [github-snip-file](#)

```
map<int, int> m;
for(int i=0; i<n;i++){
    if(m.find(a[i])==m.end()) m[a[i]]=1;
    else m[a[i]]++;
}
```

## gr-inp-Fwt

- graph input weight

- <https://thesobersobber.github.io/CP-Snippets/gr-inp-Fwt>
- [github-snip-file](#)

```
int e=f(n);
vector<vector<pair<int,int>>> g(n+1);
for(int i=1;i<=e;i++){
    int u,v,wt; cin>>u>>v>>wt;
    g[u].push_back({v,wt});
    g[v].push_back({u,wt});
}
```

## gr-inp

- graph input
- <https://thesobersobber.github.io/CP-Snippets/gr-inp>
- [github-snip-file](#)

```
int e=f(n);
vector<vector<int>> g(n+1);
for(int i=1;i<=e;i++){
    int u,v; cin>>u>>v;
    g[u].push_back(v);
    g[v].push_back(u);
}
```

## highest\_exponent

- power\_in\_fac
- [https://thesobersobber.github.io/CP-Snippets/highest\\_expone](https://thesobersobber.github.io/CP-Snippets/highest_expone)
- [github-snip-file](#)

```

int highest_exponent(int p, const int &n){
    int ans = 0;
    int t = p;
    while(t <= n){
        ans += n/t;
        t*=p;
    }
    return ans;
}

```

## interactive

- essential measures for interactive problems
- <https://thesobersobber.github.io/CP-Snippets/interactive>
- [github-snip-file](#)

```

void solve(){
    int n; cin>>n;

    auto querySystem = [&](int l, int r) {
        // print your query
        cout<<r-l+1<<endl;
        cout<<endl;
        // receive and return reply from system
        int wt; cin>>wt;
        return wt;
    };

    // write your logic here and use querySystem to receive ans
    // do a cout<<endl after each cout
}

```

```
    cout<<endl;
}
```

## ip-overloads

- I/O Overloads that I don't use
- <https://thesobersobber.github.io/CP-Snippets/ip-overloads>
- [github-snip-file](#)

```
template<typename T1, typename T2> inline istream& operator >>
template<typename T1, typename T2> inline ostream& operator <<
template<typename T> istream& operator >> (istream& in, vector<
```

```
void read(auto&... args) { ((cin>>args), ...); }
void put(auto&&... args) { ((cout<<args<<" "), ...);}
```

```
#define get(T,args...)    T args; read(args);
#define putn(args...)    { put(args); cout<<"
"; }
#define pute(args...)    { put(args); cout<<endl; }
#define putr(args...)    { putn(args) return ;}
```

## kadane

- max subarray sum  $O(n)$
- <https://thesobersobber.github.io/CP-Snippets/kadane>
- [github-snip-file](#)

```

int maxSubArraySum(vector<int> &v, int size){
    int max_so_far=INT_MIN, max_ending_here = 0;
    for (int i=0; i<v.size(); i++){
        max_ending_here += a[i];
        if (max_so_far<max_ending_here) max_so_far=max_ending_h
        if (max_ending_here < 0) max_ending_here = 0;
    }
    return max_so_far;
}

```

## kahn's algo

- toposort using bfs (kahn's algo)
- <https://thesobersobber.github.io/CP-Snippets/topo-bfs>
- [github-snip-file](#)

```

queue<int> q;
vector<int> in(n+1, 0), topo, visG(n+1, 0);
for(int i=1; i<=n; i++) for(auto child: adj[i]) in[child]++;
for(int i=1; i<=n; i++) if(in[i]==0) q.push(i);
while(!q.empty()){
    int curr = q.front(); q.pop();
    topo.push_back(curr);
    for(auto i: g[curr]){
        if(!visG[i]) continue;
        in[i]--;
        if(in[i]==0) q.push(i);
    }
}
if(topo.size()==n) for(auto i: topo) cout<<i<<" ";
else cout<<"cycle in und graph";

```

# kosaraju

- kosaraju
- <https://thesobersobber.github.io/CP-Snippets/kosaraju>
- [github-snip-file](#)

```
class Graph {
    int V;
    vector<int> *adj;

    void fillOrder(int v, bool visited[], stack<int> &s);

    void dfsUtil(int v, bool visited[]);

public:
    Graph(int V) : V(V)
    {
        adj = new vector<int>[V];
    }
    ~Graph()
    {
        delete[] adj;
    }

    void addEdge(int v, int w);

    void printSCCs();

    Graph getTranspose();
};
```



```

void Graph::dfsUtil(int v, bool visited[]) {
    visited[v] = true;
    cout << v << " ";
    for (auto &it : adj[v])
        if (!visited[it])
            dfsUtil(it, visited);
}

Graph Graph::getTranspose() {
    Graph g(V);
    for (int i = 0; i < V; i++) {
        for (auto &it : adj[i])
            g.adj[it].push_back(i);
    }
    return g;
}

void Graph::addEdge(int v, int w) {
    adj[v].push_back(w);
}

void Graph::fillOrder(int v, bool visited[], stack<int> &s) {
    visited[v] = true;
    for (auto &it : adj[v])
        if (!visited[it])
            fillOrder(it, visited, s);
    s.push(v);
}

void Graph::printSCCs() {
    stack<int> s;
    bool visited[V] = {0};

```

```

for (int i = 0; i < V; i++)
    if (!visited[i])
        fillOrder(i, visited, s);

Graph gr = getTranspose();
for (int i = 0; i < V; i++)
    visited[i] = false;

while (!s.empty()) {
    int v = s.top();
    s.pop();
    if (!visited[v]){
        gr.dfsUtil(v, visited);
        cout << "
";
    }
}
}

```

## kruskal

- kruskal
- <https://thesobersobber.github.io/CP-Snippets/kruskal>
- [github-snip-file](#)

```

auto kruskalMST(vector<Edge> &edges, int V){
    int cost = 0;
    DSU dsu(V);
    sort(begin(edges), end(edges));
    vector<Edge> tree;
    for (const auto &[u, v, w] : edges){

```

```

        if (dsu.getParent(u) != dsu.getParent(v)) {
            cost += w;
            tree.emplace_back(u, v, w);
            dsu.join(u, v);
        }
    }
    return make_pair(tree, cost);
}

```

## lambda\_function

- lambda\_function
- [https://thesobersobber.github.io/CP-Snippets/lambda\\_function](https://thesobersobber.github.io/CP-Snippets/lambda_function)
- [github-snip-file](#)

```

auto check = [&](ll mid) {
    return mid - (mid / n) >= k;
};

```

## lca-isAncestor

- lca that uses isAncestor instead of level jumping, sets a level upper limit of 25 itself since  $2^{25}$  is bigger than any N give anyways
- <https://thesobersobber.github.io/CP-Snippets/lca-isAncestor>
- [github-snip-file](#)

```

void dfs(int node, int parent, vector<vector<pair<int, int>>>&g, vector<int>&tin, vector<int>&
    up[node][0]=parent;

```

```

for(int i=1;i<25;i++)
    up[ node ][i] = up[ up[node][i-1] ][i-1];

tin[node]=timer++;

for(auto &[child,wt] : g[node])
{
    if(child==parent)
        continue;
    depth[child]=depth[node]+1;

    dp[child]=dp[node];
    dp[child][wt]++;

    dfs(child,node,g,up,dp,tin,tout,depth);
}

tout[node]=timer++;
}

bool is_ancestor(int u,int v,vector<int>&tin,vector<int>&tout)
{
    return tin[u]<=tin[v] && tout[u]>=tout[v];
}

int LCAquery(int u,int v,vector<vector<int>>&up,vector<int>&tir
{
    if( is_ancestor(u,v,tin,tout) )
        return u;
    if( is_ancestor(v,u,tin,tout) )
        return v;
    for(int i=24;i>=0;i--)

```

```

    {
        if (!is_ancestor(up[u][i], v, tin, tout))
        {
            u = up[u][i];
        }
    }
    return up[u][0];
}

```

## lca

- LCA path satisfying some condition
- <https://thesobersobber.github.io/CP-Snippets/lca>
- [github-snip-file](#)

```

constexpr int N = 5; // No. of vertices
constexpr int L = 4; // ceil(logN / log2) + 1

// Vertices from 1 to N.
vector<int> adj[N + 1];
int up[N + 1][L];
int level[N + 1];

void dfs(int u, int prev = 0){
    up[u][0] = prev;
    for (auto &v : adj[u]){
        if (v == prev) continue;

        level[v] = level[u] + 1;
        dfs(v, u);
    }
}

```

```

}

void binaryLift(){
    dfs(1);
    for (int i = 1; i < L; i++)
        for (int j = 1; j <= N; j++)
            up[j][i] = up[up[j][i - 1]][i - 1];
}

int LCA(int a, int b){
    if (level[a] > level[b])
        swap(a, b);

    int diff = level[b] - level[a];
    for (int i = 0; i < L; i++){
        if ((diff & (1 << i)))
            b = up[b][i];
    }

    if (a == b) return a;

    for (int i = L - 1; i >= 0; i--){
        if (up[a][i] != up[b][i]){
            a = up[a][i];
            b = up[b][i];
        }
    }
    return up[a][0];
}

void addEdge(int u, int v){
    adj[u].push_back(v);
}

```

```

adj[v].push_back(u);
}

int dist(int a, int b){
    return level[a] + level[b] - 2 * level[LCA(a, b)];
}

```

## log

- log
- <https://thesobersobber.github.io/CP-Snippets/log>
- [github-snip-file](#)

// Computes  $x$  which  $a^x = b \pmod n$ .

```

long long d_log(long long a, long long b, long long n) {
    long long m = ceil(sqrt(n));
    long long aj = 1;
    map<long long, long long> M;
    for (int i = 0; i < m; ++i) {
        if (!M.count(aj))
            M[aj] = i;
        aj = (aj * a) % n;
    }

    long long coef = mod_pow(a, n - 2, n);
    coef = mod_pow(coef, m, n);
    // coef = a ^ (-m)
    long long gamma = b;
    for (int i = 0; i < m; ++i) {
        if (M.count(gamma)) {

```

```

        return i * m + M[gamma];
    } else {
        gamma = (gamma * coef) % n;
    }
}
return -1;
}

```

## matrix

- matrix
- <https://thesobersobber.github.io/CP-Snippets/matrix>
- [github-snip-file](#)

```

const int MN  = 111;
const int mod = 10000;

struct matrix {
    int r, c;
    int m[MN][MN];

    matrix (int _r, int _c) : r (_r), c (_c) {
        memset(m, 0, sizeof m);
    }

    void print() {
        for (int i = 0; i < r; ++i) {
            for (int j = 0; j < c; ++j)
                cout << m[i][j] << " ";
            cout << endl;
        }
    }
}

```



```

}

int x[MN][MN];
matrix & operator *= (const matrix &o) {
    memset(x, 0, sizeof x);
    for (int i = 0; i < r; ++i)
        for (int k = 0; k < c; ++k)
            if (m[i][k] != 0)
                for (int j = 0; j < c; ++j) {
                    x[i][j] = (x[i][j] + ((m[i][k] * o.m[k][j]) % mod))
                }
    memcpy(m, x, sizeof(m));
    return *this;
}
};

void matrix_pow(matrix b, long long e, matrix &res) {
    memset(res.m, 0, sizeof res.m);
    for (int i = 0; i < b.r; ++i)
        res.m[i][i] = 1;

    if (e == 0) return;
    while (true) {
        if (e & 1) res *= b;
        if ((e >>= 1) == 0) break;
        b *= b;
    }
}

```

## mint

- modular integer

- <https://thesobersobber.github.io/CP-Snippets/mint>
- [github-snip-file](#)

```
struct mi {
    int64_t v; explicit operator int64_t() const { return v % m; }
    mi() { v = 0; }
    mi(int64_t _v) {
        v = (-mod < _v && _v < mod) ? _v : _v % mod;
        if (v < 0) v += mod;
    }
    friend bool operator==(const mi& a, const mi& b) {
        return a.v == b.v; }
    friend bool operator!=(const mi& a, const mi& b) {
        return !(a == b); }
    friend bool operator<(const mi& a, const mi& b) {
        return a.v < b.v; }

    mi& operator+=(const mi& m) {
        if ((v += m.v) >= mod) v -= mod;
        return *this; }
    mi& operator-=(const mi& m) {
        if ((v -= m.v) < 0) v += mod;
        return *this; }
    mi& operator*=(const mi& m) {
        v = v*m.v%mod; return *this; }
    mi& operator/=(const mi& m) { return (*this) *= inv(m); }
    friend mi pow(mi a, int64_t p) {
        mi ans = 1; assert(p >= 0);
        for (; p; p /= 2, a *= a) if (p&1) ans *= a;
        return ans;
    }
}
```

```

friend mi inv(const mi& a) { assert(a.v != 0);
    return pow(a,mod-2); }

mi operator-() const { return mi(-v); }
mi& operator++() { return *this += 1; }
mi& operator--() { return *this -= 1; }
mi operator++(int32_t) { mi temp; temp.v = v++; return temp; }
mi operator--(int32_t) { mi temp; temp.v = v--; return temp; }
friend mi operator+(mi a, const mi& b) { return a += b; }
friend mi operator-(mi a, const mi& b) { return a -= b; }
friend mi operator*(mi a, const mi& b) { return a *= b; }
friend mi operator/(mi a, const mi& b) { return a /= b; }
friend ostream& operator<<(ostream& os, const mi& m) {
    os << m.v; return os;
}
friend istream& operator>>(istream& is, mi& m) {
    int64_t x; is >> x;
    m.v = x;
    return is;
}
friend void __print(const mi &x) {
    cerr << x.v;
}
};

```

## modpow

- modpow
- <https://thesobersobber.github.io/CP-Snippets/modpow>
- [github-snip-file](#)

```

11 modpow(11 a, 11 b){
    a %= m;
    11 res = 1;
    while (b > 0) {
        if (b & 1) res = res * a % m;
        a=a*a%m;
        b>>=1;
    }
    return res;
}

```

## pbds

- pbds
- <https://thesobersobber.github.io/CP-Snippets/pbds>
- [github-snip-file](#)

```

#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
// pbds = find_by_value(), order_of_key()
// find_by_order(k) returns iterator to kth element starting f
// order_of_key(k) returns count of elements strictly smaller t
template<class T> using minheap = priority_queue<T,vector<T>,gr
template<class T> using ordered_set = tree<T, null_type,less<T>
template<class key, class value, class cmp = std::less<key>> us

```

## pq

- pq
- <https://thesobersobber.github.io/CP-Snippets/pq>

- [github-snip-file](#)

```
priority_queue<int> pq;  
priority_queue<int, vector<int>, greater<>> pq;
```

## prime-related-stuff

- implements prime fac, fac list and is\_prime in both space optimized and time optimized ways
- <https://thesobersobber.github.io/CP-Snippets/prime-related->
- [github-snip-file](#)

```
vector<int> smallest_factor;  
vector<bool> prime;  
vector<int> primes;  
void sieve(int maximum)  
{  
    maximum = max(maximum, 1);  
    smallest_factor.assign(maximum + 1, 0);  
    prime.assign(maximum + 1, true);  
    prime[0] = prime[1] = false;  
    primes = {};  
    for (int p = 2; p <= maximum; p++)  
        if (prime[p])  
        {  
            smallest_factor[p] = p;  
            primes.push_back(p);  
            for (int64_t i = int64_t(p) * p; i <= maximum; i +=  
                if (prime[i])  
                {  
                    prime[i] = false;
```

```

        smallest_factor[i] = p;
    }
}

// Determines whether n is prime in worst case  $O(\sqrt{n} / \log n)$ . Requires
// If we've run `sieve` up to at least n, takes  $O(1)$  time.
bool is_prime(int64_t n)
{
    int64_t sieve_max = int64_t(smallest_factor.size()) - 1;
    assert(1 <= n && n <= sieve_max * sieve_max);
    if (n <= sieve_max)
        return prime[n];
    for (int64_t p : primes)
    {
        if (p * p > n)
            break;
        if (n % p == 0)
            return false;
    }
    return true;
}

// Prime factorizes n in worst case  $O(\sqrt{n} / \log n)$ . Requires
// If we've run `sieve` up to at least n, takes  $O(\log n)$  time.
template <typename T>
vector<pair<T, int>> prime_factorize(T n)
{

```

```

int64_t sieve_max = int64_t(smallest_factor.size()) - 1;
assert(1 <= n && n <= sieve_max * sieve_max);
vector<pair<T, int>> result;
if (n <= sieve_max)
{
    while (n != 1)
    {
        int p = smallest_factor[n];
        int exponent = 0;
        do
        {
            n /= p;
            exponent++;
        } while (n % p == 0);
        result.emplace_back(p, exponent);
    }
    return result;
}
for (int p : primes)
{
    if (int64_t(p) * p > n)
        break;
    if (n % p == 0)
    {
        result.emplace_back(p, 0);
        do
        {
            n /= p;
            result.back().second++;
        } while (n % p == 0);
    }
}

```

```

    if (n > 1)
        result.emplace_back(n, 1);
    return result;
}

template <typename T>
vector<T> generate_factors(const vector<pair<T, int>> &prime_factors)
{
    // See http://oeis.org/A066150 and http://oeis.org/A036451
    static vector<T> buffer;
    int product = 1;
    for (auto &pf : prime_factors)
        product *= pf.second + 1;
    vector<T> factors = {1};
    factors.reserve(product);
    if (sorted)
        buffer.resize(product);
    for (auto &pf : prime_factors)
    {
        T p = pf.first;
        int exponent = pf.second;
        int before_size = int(factors.size());
        for (int i = 0; i < exponent * before_size; i++)
            factors.push_back(factors[factors.size() - before_size]);
        if (sorted && factors[before_size - 1] > p)
            for (int section = before_size; section < int(factors.size()); section++)
                for (int i = 0; i + section < int(factors.size()); i++)
                {
                    int length = min(2 * section, int(factors.size() - section));
                    merge(factors.begin() + i, factors.begin() + i + section, factors.begin() + i + section, factors.begin() + i + section + length, factors.begin() + i + section + length);
                }
    }
}

```



```

                factors.begin() + i + section, factor
                buffer.begin());
            copy(buffer.begin(), buffer.begin() + length,
                factors.begin() + i + section);
        }
    }
    assert(int(factors.size()) == product);
    return factors;
}

void pre_process() {
    sieve(1e6+5);
}

// mint

struct mi {
    int64_t v; explicit operator int64_t() const { return v % mod; }
    mi() { v = 0; }
    mi(int64_t _v) {
        v = (-mod < _v && _v < mod) ? _v : _v % mod;
        if (v < 0) v += mod;
    }
    friend bool operator==(const mi& a, const mi& b) {
        return a.v == b.v; }
    friend bool operator!=(const mi& a, const mi& b) {
        return !(a == b); }
    friend bool operator<(const mi& a, const mi& b) {
        return a.v < b.v; }

    mi& operator+=(const mi& m) {

```

```

        if ((v += m.v) >= mod) v -= mod;
        return *this; }
mi& operator-=(const mi& m) {
    if ((v -= m.v) < 0) v += mod;
    return *this; }
mi& operator*=(const mi& m) {
    v = v*m.v%mod; return *this; }
mi& operator/=(const mi& m) { return (*this) *= inv(m); }
friend mi pow(mi a, int64_t p) {
    mi ans = 1; assert(p >= 0);
    for (; p; p /= 2, a *= a) if (p&1) ans *= a;
    return ans;
}
friend mi inv(const mi& a) { assert(a.v != 0);
    return pow(a,mod-2); }

mi operator-() const { return mi(-v); }
mi& operator++() { return *this += 1; }
mi& operator--() { return *this -= 1; }
mi operator++(int32_t) { mi temp; temp.v = v++; return temp; }
mi operator--(int32_t) { mi temp; temp.v = v--; return temp; }
friend mi operator+(mi a, const mi& b) { return a += b; }
friend mi operator-(mi a, const mi& b) { return a -= b; }
friend mi operator*(mi a, const mi& b) { return a *= b; }
friend mi operator/(mi a, const mi& b) { return a /= b; }
friend ostream& operator<<(ostream& os, const mi& m) {
    os << m.v; return os;
}
friend istream& operator>>(istream& is, mi& m) {
    int64_t x; is >> x;
    m.v = x;
    return is;
}

```

```

    }
    friend void __print(const mi &x) {
        cerr << x.v;
    }
};

```

## re-write

- a bunch of re and write functions based on template meta programming  
helpful in cp.
- <https://thesobersobber.github.io/CP-Snippets/read-write-fn->
- [github-snip-file](#)

```

template <class T1, class T2> void re(pair<T1, T2> &p);
template <class T> void re(vector<T> &a);
template <class T, size_t SZ> void re(array<T, SZ> &a);
template <class T> void re(T &x) { cin >> x; }
void re(double &x) { string t;re(t); x = stod(t);}
template <class Arg, class... Args> void re(Arg &first, Args &.
template <class T1, class T2> void re(pair<T1, T2> &p) { re(p.f
template <class T> void re(vector<T> &a) {for (int i = 0; i < s
template <class T, size_t SZ>void re(array<T, SZ> &a) { for (ir
template <class T>
void write(T x) { cout << x << " "; }
template <class T> void writen(T x) { cout << x << nl; }
template<class T> using minheap = priority_queue<T,vector<T>,gr
template<class T> using ordered_set = tree<T, null_type,less<T>
template<class key, class value, class cmp = std::less<key>> us

```

## recur-binsearch

- recursive binary search implementation to make intuition easier ig
- <https://thesobersobber.github.io/CP-Snippets/recur-binsearch>
- [github-snip-file](#)

```
auto check = [&](int mid) {
    // smthing here
    return bool ();
};
function<int(int,int)> recur_binsearch = [&](int lo, int hi) {
    if(hi<=lo) return lo;
    int mid=(lo+hi)/2;
    if(check(mid)) return recur_binsearch(lo, mid-1);
    return recur_binsearch(mid+1, hi);
}
```

## recur-modpow

- recur-modpow
- <https://thesobersobber.github.io/CP-Snippets/recur-modpow>
- [github-snip-file](#)

```
int power(int x, int y){
    if (y==0) return 1;

    int v = power(x, y/2);
    v *= v;
    v %= mod;
    if (y&1) return (v*x)%mod;
    else return v;
}
```

## rng

- rng
- <https://thesobersobber.github.io/CP-Snippets/rng>
- [github-snip-file](#)

```
//random generator
```

```
mt19937 rng(chrono::steady_clock::now().time_since_epoch().count());
ll rnd(ll a,ll b){if(a > b){return -1;}return a + (ll)rng() % (b-a+1);}
```

## rr-segtree

- best segtree
- <https://thesobersobber.github.io/CP-Snippets/rr-segtree>
- [github-snip-file](#)

```
int phi[N+1];
```

```
struct node
```

```
{
    long long sum,max,lca,size;
    node()
    {
        lca=-1;
        max=-1;
        sum=-1;
        size=0;
    };
};
```

```

struct Segment_Tree
{
    vector<node> segtree;
    int n;
    node identity;

    void init(int _n)
    {
        identity.lca=-1;
        identity.sum=0;
        identity.max=-1;
        identity.size=0;

        n=1;
        while(n<_n)
            n=n*2;
        segtree.resize(2*n);
    }

    node merge(node a,node b)
    {
        if(a.lca<1)
            return b;
        if(b.lca<1)
            return a;

        node ans;
        ans.max=std::max(a.max,b.max);
        ans.sum=a.sum+b.sum;
        ans.size=a.size+b.size;

        int ex=50;
    }
};

```

```

        int A=a.lca;
        int B=b.lca;

        while(true)
        {
            if(A==B)
                break;
            if(A>B)
            {
                ans.sum=ans.sum+a.size;
                A=phi[A];
            }
            else
            {
                ans.sum=ans.sum+b.size;
                B=phi[B];
            }
        }
        ans.lca=A;

        return ans;
    }

    void build(int curr,int left,int right,vector<int>&ar)
    {

        if(right-left==1)
        {
            if(left<ar.size())
            {
                segtree[curr].sum=0;
                segtree[curr].max=ar[left];
            }
        }
    }

```

```

        segtree[curr].lca=ar[left];
        segtree[curr].size=1;
    }
    else
    {
        segtree[curr].sum=0;
        segtree[curr].max=-1;
        segtree[curr].lca=-1;
        segtree[curr].size=0;
    }
    return;
}

int mid=(left+right)/2;
build(2*curr+1,left,mid,ar);
build(2*curr+2,mid,right,ar);

segtree[curr]=merge(segtree[2*curr+1],segtree[2*curr+
}

node sum(int lq,int rq,int node,int left,int right)
{

    if(lq>=right || rq<=left)
        return identity;
    if(left>=lq && rq>=right)
        return segtree[node];

    int mid=(left+right)/2;
    return merge(sum(lq,rq,2*node+1,left,mid),sum(lq,rq,2
}

```



```

void operate(int lq,int rq,int curr,int left,int right)
{
    if(lq>=right || rq<=left)
        return;

    if(right-left==1)
    {
        int val=segtree[curr].lca;
        val=phi[val];
        segtree[curr].lca=val;
        segtree[curr].max=val;
        segtree[curr].sum=0;
        segtree[curr].size=1;
        return;
    }

    if(segtree[curr].max<=1)
        return;

    int mid=(left+right)/2;
    operate(lq,rq,2*curr+1,left,mid);
    operate(lq,rq,2*curr+2,mid,right);

    segtree[curr]=merge(segtree[2*curr+1],segtree[2*curr+2]);
}

};

```

## segtree

- sextree

- <https://thesobersobber.github.io/CP-Snippets/segtree>
- [github-snip-file](#)

```
template<class T, class U>
// T -> node, U->update.
struct Lsegtree{
    vector<T>st;
    vector<U>lazy;
    ll n;
    T identity_element;
    U identity_update;

    /*
        Definition of identity_element: the element I such that
        for all x

        Definition of identity_update: the element I such that
        for all x
    */

    Lsegtree(ll n, T identity_element, U identity_update){
        this->n = n;
        this->identity_element = identity_element;
        this->identity_update = identity_update;
        st.assign(4*n, identity_element);
        lazy.assign(4*n, identity_update);
    }

    T combine(T l, T r){
        // change this function as required.
        T ans = (l + r);
    }
};
```

```

        return ans;
    }

void buildUtil(ll v, ll tl, ll tr, vector<T>&a){
    if(tl == tr){
        st[v] = a[tl];
        return;
    }
    ll tm = (tl + tr)>>1;
    buildUtil(2*v + 1, tl, tm,a);
    buildUtil(2*v + 2,tm+1,tr,a);
    st[v] = combine(st[2*v + 1], st[2*v + 2]);
}

// change the following 2 functions, and you're more or less
T apply(T curr, U upd, ll tl, ll tr){
    T ans = (tr-tl+1)*upd;
    // increment range by upd:
    // T ans = curr + (tr - tl + 1)*upd
    return ans;
}

U combineUpdate(U old_upd, U new_upd, ll tl, ll tr){
    U ans = old_upd;
    ans=new_upd;
    return ans;
}

void push_down(ll v, ll tl, ll tr){
    //for the below line to work, make sure the "==" operat
    if(lazy[v] == identity_update)return;
    st[v] = apply(st[v], lazy[v], tl, tr);
}

```

```

        if(2*v + 1 <= 4*n){
            ll tm = (tl + tr)>>1;
            lazy[2*v + 1] = combineUpdate(lazy[2*v+1], lazy[v],
            lazy[2*v + 2] = combineUpdate(lazy[2*v+2], lazy[v],
        }
        lazy[v] = identity_update;
    }

T queryUtil(ll v, ll tl, ll tr, ll l, ll r){
    push_down(v,tl,tr);
    if(l > r)return identity_element;
    if(tr < l or tl > r){
        return identity_element;
    }
    if(l <= tl and r >= tr){
        return st[v];
    }
    ll tm = (tl + tr)>>1;
    return combine(queryUtil(2*v+1,tl,tm,l,r), queryUtil(2*
}

void updateUtil(ll v, ll tl, ll tr, ll l, ll r, U upd){
    push_down(v,tl,tr);
    if(tr < l or tl > r)return;
    if(tl >=l and tr <=r){
        lazy[v] = combineUpdate(lazy[v],upd,tl,tr);
        push_down(v,tl,tr);
    } else{
        ll tm = (tl + tr)>>1;
        updateUtil(2*v+1,tl,tm,l,r,upd);
        updateUtil(2*v+2,tm+1,tr,l,r,upd);
        st[v] = combine(st[2*v + 1], st[2*v+2]);
    }
}

```

```

    }
}

void build(vector<T>a){
    assert( (ll)a.size() == n);
    buildUtil(0,0,n-1,a);
}

T query(ll l, ll r){
    return queryUtil(0,0,n-1,l,r);
}

void update(ll l,ll r, U upd){
    updateUtil(0,0,n-1,l,r,upd);
}
};

```

## seive

- seive
- <https://thesobersobber.github.io/CP-Snippets/seive>
- [github-snip-file](#)

```

vector<bool> Prime;
vector<int> spf;
void sieve(int s = maxn) {
    Prime.resize(s + 1, 1);
    spf.resize(s + 1, s + 1);
    for(int i = 2 ; i <= s ; i++)    if(Prime[i]) {
        spf[i] = min(spf[i], i);
        for(int j = i ; (ll)j * i <= s ; j++)

```

```

        Prime[j * i] = 0, spf[j * i] = min(i, spf[j * i]);
    }
}

```

## splay-tree-rr-sir

- used here by rr sir, I have no idea how to use it or what it's used in mostly, [RR Sir ABC F Submission](#)
- [https://thesobersobber.github.io/CP-Snippets/Splay Tree](https://thesobersobber.github.io/CP-Snippets/Splay%20Tree)
- [github-snip-file](#)

```

#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>

```

```

namespace allocator {

```

```

// Array allocator.

```

```

template <class T, int MAXSIZE>

```

```

struct array {

```

```

    T v[MAXSIZE], *top;

```

```

    array() : top(v) {}

```

```

    T *alloc(const T &val = T()) { return &(*top++ = val); }

```

```

    void dealloc(T *p) {}

```

```

};

```

```

// Stack-based array allocator.

```

```

template <class T, int MAXSIZE>

```

```

struct stack {

```

```

    T v[MAXSIZE];

```

```

    T *spot[MAXSIZE], **top;

```

```

stack() {
    for (int i = 0; i < MAXSIZE; ++i) spot[i] = v + i;
    top = spot + MAXSIZE;
}
T *alloc(const T &val = T()) { return &(**--top = val); }
void dealloc(T *p) { *top++ = p; }
};

} // namespace allocator

namespace splay {

// Abstract node struct.
template <class T>
struct node {
    T *f, *c[2];
    int size;
    node() {
        f = c[0] = c[1] = nullptr;
        size = 1;
    }
    void push_down() {}
    void update() {
        size = 1;
        for (int t = 0; t < 2; ++t)
            if (c[t]) size += c[t]->size;
    }
};

// Abstract reversible node struct.
template <class T>
struct reversible_node : node<T> {

```

```

int r;
reversible_node() : node<T>() { r = 0; }
void push_down() {
    node<T>::push_down();
    if (r) {
        for (int t = 0; t < 2; ++t)
            if (node<T>::c[t]) node<T>::c[t]->reverse();
        r = 0;
    }
}
void update() { node<T>::update(); }
// Reverse the range of this node.
void reverse() {
    std::swap(node<T>::c[0], node<T>::c[1]);
    r = r ^ 1;
}
};

template <class T, int MAXSIZE = 500000,
          class alloc = allocator::array<T, MAXSIZE + 2>>
struct tree {
    alloc pool;
    T *root;
    // Get a new node from the pool.
    T *new_node(const T &val = T()) { return pool.alloc(val); }
    tree() {
        root = new_node(), root->c[1] = new_node(), root->size = 2;
        root->c[1]->f = root;
    }
    // Helper function to rotate node.
    void rotate(T *n) {
        int v = n->f->c[0] == n;

```



```

    T *p = n->f, *m = n->c[v];
    if (p->f) p->f->c[p->f->c[1] == p] = n;
    n->f = p->f, n->c[v] = p;
    p->f = n, p->c[v ^ 1] = m;
    if (m) m->f = p;
    p->update(), n->update();
}
// Splay n so that it is under s (or to root if s is null).
void splay(T *n, T *s = nullptr) {
    while (n->f != s) {
        T *m = n->f, *l = m->f;
        if (l == s)
            rotate(n);
        else if ((l->c[0] == m) == (m->c[0] == n))
            rotate(m), rotate(n);
        else
            rotate(n), rotate(n);
    }
    if (!s) root = n;
}
// Get the size of the tree.
int size() { return root->size - 2; }
// Helper function to walk down the tree.
int walk(T *n, int &v, int &pos) {
    n->push_down();
    int s = n->c[0] ? n->c[0]->size : 0;
    (v = s < pos) && (pos -= s + 1);
    return s;
}
// Insert node n to position pos.
void insert(T *n, int pos) {
    T *c = root;

```

```

    int v;
    ++pos;
    while (walk(c, v, pos), c->c[v] && (c = c->c[v]))
        ;
    c->c[v] = n, n->f = c, splay(n);
}
// Find the node at position pos. If sp is true, splay it.
T *find(int pos, int sp = true) {
    T *c = root;
    int v;
    ++pos;
    while ((pos < walk(c, v, pos) || v) && (c = c->c[v]))
        ;
    if (sp) splay(c);
    return c;
}
// Find the range [posl, posr) on the splay tree.
T *find_range(int posl, int posr) {
    T *r = find(posr), *l = find(posl - 1, false);
    splay(l, r);
    if (l->c[1]) l->c[1]->push_down();
    return l->c[1];
}
// Insert nn of size nn_size to position pos.
void insert_range(T **nn, int nn_size, int pos) {
    T *r = find(pos), *l = find(pos - 1, false), *c = l;
    splay(l, r);
    for (int i = 0; i < nn_size; ++i) c->c[1] = nn[i], nn[i]->f = c;
    for (int i = nn_size - 1; i >= 0; --i) nn[i]->update();
    l->update(), r->update(), splay(nn[nn_size - 1]);
}
// Helper function to dealloc a subtree.

```

```

void dealloc(T *n) {
    if (!n) return;
    dealloc(n->c[0]);
    dealloc(n->c[1]);
    pool.dealloc(n);
}
// Remove from position [posl, posr).
void erase_range(int posl, int posr) {
    T *n = find_range(posl, posr);
    n->f->c[1] = nullptr, n->f->update(), n->f->f->update(), n-
    dealloc(n);
}
};

} // namespace splay

const int MAXSIZE = 500005;

struct node: splay::reversible_node<node> {
    long long val, val_min, label_add;
    node(long long v = 0) : splay::reversible_node<node>(), val(v)
    // Add v to the subtree.
    void add(long long v) {
        val += v;
        val_min += v;
        label_add += v;
    }
    void push_down() {
        splay::reversible_node<node>::push_down();
        for (int t = 0; t < 2; ++t) if (c[t]) c[t]->add(label_add);
        label_add = 0;
    }
}

```

```

void update() {
    splay::reversible_node<node>::update();
    val_min = val;
    for (int t = 0; t < 2; ++t) if (c[t]) val_min = std::min(va
}
};

```

```

splay::tree<node, MAXSIZE, allocator::stack<node, MAXSIZE + 2>>

```

## tokenizer

- tokenizer that has no use
- <https://thesobersobber.github.io/CP-Snippets/tokenizer>
- [github-snip-file](#)

```

vec(string) tokenizer(string str,char ch) {std::istringstream v

```

## totient-seive

- totient-seive
- <https://thesobersobber.github.io/CP-Snippets/totient-seive>
- [github-snip-file](#)

```

for (int i = 1; i < MN; i++)
    phi[i] = i;

for (int i = 1; i < MN; i++)
    if (!sieve[i]) // is prime
        for (int j = i; j < MN; j += i)
            phi[j] -= phi[j] / i;

```

## totient

- totient
- <https://thesobersobber.github.io/CP-Snippets/totient>
- [github-snip-file](#)

```
long long totient(long long n) {
    if (n == 1) return 0;
    long long ans = n;
    for (int i = 0; primes[i] * primes[i] <= n; ++i) {
        if ((n % primes[i]) == 0) {
            while ((n % primes[i]) == 0) n /= primes[i];
            ans -= ans / primes[i];
        }
    }
    if (n > 1) {
        ans -= ans / n;
    }
    return ans;
}
```

## trie

- trie
- <https://thesobersobber.github.io/CP-Snippets/trie>
- [github-snip-file](#)

```
struct Trie{
    struct node{
        node* next[10];
```

```

        node(){
            for(int i=0;i<10;i++) next[i]=NULL;
        }
};

node root;

void add(vector<int>&val){
    node* temp=&root;
    for(auto ele : val){
        if(temp->next[ele]==NULL) temp->next[ele]=new node(
            temp=temp->next[ele];
        }
    }

    int query(vector<int>&val){
        node* temp=&root;
        int ans=0;
        for(auto ele : val){
            if(temp->next[ele]==NULL) break;
            ans++;
            temp=temp->next[ele];
        }
        return ans;
    }
};

```

## troll

- troll
- <https://thesobersobber.github.io/CP-Snippets/troll>

- [github-snip-file](#)

```
// Assembly Generator: gcc -S -o temp.s fileName.cpp
// Executable: gcc -o temp.exe fileName.cpp
#define assembler(x) __asm__(R"(x)");
// real source -
```

## two-sat (kosaraju)

- two-sat (kosaraju)
- <https://thesobersobber.github.io/CP-Snippets/two-sat> (kosar
- [github-snip-file](#)

```
/**
 * Given a set of clauses (a1 v a2)^(a2 v ¬a3)....
 * this algorithm find a solution to it set of clauses.
 * test: http://lightoj.com/volume_showproblem.php?problem=125
 */

#include<bits/stdc++.h>
using namespace std;
#define MAX 100000
#define endl '
'

vector<int> G[MAX];
vector<int> GT[MAX];
vector<int> Ftime;
vector<vector<int> > SCC;
bool visited[MAX];
int n;
```

```
void dfs1(int n){
    visited[n] = 1;

    for (int i = 0; i < G[n].size(); ++i) {
        int curr = G[n][i];
        if (visited[curr]) continue;
        dfs1(curr);
    }

    Ftime.push_back(n);
}
```

```
void dfs2(int n, vector<int> &scc) {
    visited[n] = 1;
    scc.push_back(n);

    for (int i = 0; i < GT[n].size(); ++i) {
        int curr = GT[n][i];
        if (visited[curr]) continue;
        dfs2(curr, scc);
    }
}
```

```
void kosaraju() {
    memset(visited, 0, sizeof visited);

    for (int i = 0; i < 2 * n ; ++i) {
        if (!visited[i]) dfs1(i);
    }
}
```



```

memset(visited, 0, sizeof visited);
for (int i = Ftime.size() - 1; i >= 0; i--) {
    if (visited[Ftime[i]]) continue;
    vector<int> _scc;
    dfs2(Ftime[i], _scc);
    SCC.push_back(_scc);
}
}

/**
 * After having the SCC, we must traverse each scc, if in one
 * Otherwise we build a solution, making the first "node" that
 */

bool two_sat(vector<int> &val) {
    kosaraju();
    for (int i = 0; i < SCC.size(); ++i) {
        vector<bool> tmpvisited(2 * n, false);
        for (int j = 0; j < SCC[i].size(); ++j) {
            if (tmpvisited[SCC[i][j] ^ 1]) return 0;
            if (val[SCC[i][j]] != -1) continue;
            else {
                val[SCC[i][j]] = 0;
                val[SCC[i][j] ^ 1] = 1;
            }
            tmpvisited[SCC[i][j]] = 1;
        }
    }
    return 1;
}

```

```

// Example of use

int main() {

    int m, u, v, nc = 0, t; cin >> t;
    // n = "nodes" number, m = clauses number

    while (t--) {
        cin >> m >> n;
        Ftime.clear();
        SCC.clear();
        for (int i = 0; i < 2 * n; ++i) {
            G[i].clear();
            GT[i].clear();
        }

        //  $(a1 \vee a2) = (\neg a1 \rightarrow a2) = (\neg a2 \rightarrow a1)$ 
        for (int i = 0; i < m ; ++i) {
            cin >> u >> v;
            int t1 = abs(u) - 1;
            int t2 = abs(v) - 1;
            int p = t1 * 2 + ((u < 0)? 1 : 0);
            int q = t2 * 2 + ((v < 0)? 1 : 0);
            G[p ^ 1].push_back(q);
            G[q ^ 1].push_back(p);
            GT[p].push_back(q ^ 1);
            GT[q].push_back(p ^ 1);
        }

        vector<int> val(2 * n, -1);
        cout << "Case " << ++nc << ": ";
    }
}

```

```

if (two_sat(val)) {
    cout << "Yes" << endl;
    vector<int> sol;
    for (int i = 0; i < 2 * n; ++i)
        if (i % 2 == 0 and val[i] == 1)
            sol.push_back(i / 2 + 1);
    cout << sol.size() ;

    for (int i = 0; i < sol.size(); ++i) {
        cout << " " << sol[i];
    }
    cout << endl;
} else {
    cout << "No" << endl;
}
}
return 0;
}

```

## variadic

- variadic lambdas with 1 and 2 arguments
- <https://thesobersobber.github.io/CP-Snippets/variadic>
- [github-snip-file](#)

```

#define f(u, args...)    [&](auto &&u) { return args; }
#define g(u, v, args...) [&](auto &&u, auto &&v) { return args

```

## xor-basis

- xor-basis

- <https://thesobersobber.github.io/CP-Snippets/xor-basis>
- [github-snip-file](#)

```
struct XorBasis{
    private:
    vector<ll> basis;
    int lg;
    int sz = 0;

    public:
    XorBasis(int lg) : lg(lg){
        basis.resize(lg);
    }
    void add(ll x){
        if(x >= (1ll<<lg)) return;
        for(int i=0;i<lg;i++){
            if(~x&(1ll<<i)) continue;
            if(!basis[i]){
                basis[i] = x;
                ++sz;
            }
            x^=basis[i];
        }
    }
    bool contains(ll x){
        for(int i=0;i<lg;i++){
            if(~x&(1ll<<i)) continue;
            if(!basis[i]){
                return false;
            }
            x^=basis[i];
        }
    }
}
```

```
    }  
    return true;  
}  
int size(){  
    return sz;  
}  
const vector<ll>::iterator begin(){  
    return basis.begin();  
}  
const vector<ll>::iterator end(){  
    return basis.end();  
}  
};
```