

Team notebook

Capybara

September 15, 2023

Contents

1	Attention	1
2	BruteForce	1
2.1	permmtuation	1
2.2	queen	2
3	DP	2
3.1	ActivityWeight	2
3.2	CountingTilings	2
3.3	ElevatorRides	3
3.4	LCS	3
3.5	LIS	4
3.6	MaxIntervalSum	4
3.7	coin	4
3.8	pack	4
4	Funtions	5
4.1	BIT	5
4.2	BitMask	5
4.3	MatrixQuickPower	5
4.4	MosAlgorithm	6
4.5	PYInput	6
4.6	SegmentTree	6
4.7	Treap	7
4.8	discrete	8
4.9	qmul	8
4.10	qpow	8

5	Geometry	8
5.1	ConvexHull	8
5.2	CrossProduct	9
6	Graph	9
6.1	BellmanFord	9
6.2	BipartieMaching	9
6.3	DFSandBFS	10
6.4	Dijkstra	10
6.5	Dinic	11
6.6	FloydWarshall	11
6.7	MCMF	12
6.8	Maxflow	12
6.9	PlanetsCycles	13
6.10	TopologicalSort	13
6.11	success	14
7	Mathematics	14
7.1	BabyStepGiantStep	14
7.2	CatalanNumbers	14
7.3	ChineseRemainder	15
7.4	Choose	15
7.5	CountingNecklaces	15
7.6	Derangements	16
7.7	Fermat'sLittleTheorem	16
7.8	Inverse	16
7.9	JosephusProblem	16
7.10	NimGame	17
7.11	PrimeFactor	17
7.12	PrimeSieve	18
7.13	PythagoreanTriplets	18

7.14	Wilson'sTheorem	18
8	Sorting	18
8.1	BinarySearch	18
8.2	MergeSort	19
8.3	TrafficLights	19
9	Tree	19
9.1	DFStree	19
9.2	DistanceTree	19
9.3	IndptTreeDFS	20
9.4	LCAbi	20
9.5	MST	20
10	string	21
10.1	LongestSubstringWithoutRep	21

1 Attention

```
Codeblock set
-std=c++14 -Wall -Wshadow -O2
Linux -Wextra

#include <bits/stdc++.h>
using namespace std;
ios::sync_with_stdio(false);
cin.tie(0);

vim .vimrc
cd -> vim .vimrc -> cd Desktop -> mkdir -> AC
print
```

```

syntax on
set clipboard=unnamed
set nu
set bs=2
set cin si
set cf
set sta sw=4 sts=4 ts=4
set mouse=a
set bg=dark
set cul
imap {<CR> {<CR>}<esc>ko
imap [ []<esc>i
imap ( ()<esc>i

vim command
yy copy
dd cut
d~ form top to here cut
p paste
u undo
ctrl+r unundo

warnning
g++ -O2
remainder,float error(change
int),overflow,dont eat too much

```

2 BruteForce

2.1 permmutation

```

int n;
vi subset;
void search(int k){
    if(k == n + 1){
        for(auto x:subset){
            cout << x << ' ';
        }
        cout << '\n';
    }
    else{
        subset.push_back(k);

```

```

        search(k + 1);
        subset.pop_back();
        search(k + 1);
    }
}

vi permutation;
bool chosen[10];
void search2(){
    if(permutation.size() == n){
        for(auto x:permutation){
            cout << x << ' ';
        }
        cout << '\n';
    }
    else{
        for(int i = 0; i < n; i++){
            if(chosen[i]) continue;
            chosen[i] = true;
            permutation.push_back(i);
            search2();
            chosen[i] = false;
            permutation.pop_back();
        }
    }
}

int main(){
    cin >> n;
    for(int i = 0; i < n; i++){
        permutation.push_back(i);
    }
    do{
        for(auto x:permutation){
            cout << x << ' ';
        }
        cout << '\n';
    }while(next_permutation(permutation.begin(),
        permutation.end()));
    // search2();
    return 0;
}

```

2.2 queen

```

int n = 4;
int cnt;
bool col[5],diag1[5],diag2[5];
void search(int y){
    if(y == n){
        cnt ++;
    }
    else{
        REP(x,0,n-1){
            if(col[x] || diag1[y+x] ||
                diag2[x-y+n-1])continue;
            col[x] = diag1[y+x] =
                diag2[x-y+n-1] = 1;
            search(y+1);
            col[x] = diag1[y+x] =
                diag2[x-y+n-1] = 0;
        }
    }
}

int main(){
    cin >> n;
    search(0);
    cout << cnt << '\n';
}

```

3 DP

3.1 ActivityWeight

```

#define start first.second
#define endd first.first
#define weight second
vector<pair<pair<ll,ll>,ll> > v1;
int bi_se(ll s, ll t,ll goal){
    if(s==t){//need rest
        if(goal <= v1[s].endd) return s-1; //
        or <
        else return s;
    }
}

```

```

    ll mid = (s+t)/2;
    if(goal <= v1[mid].endd) return
        bi_se(s,mid,goal);
    else return bi_se(mid + 1,t,goal);
}
int main(){
    ios::sync_with_stdio(false);
    cin.tie(0);
    ll n;
    cin >> n; //activities
    ll s,t,w;
    v1.push_back({0,0,0});
    for(ll i = 0; i < n; i++){
        cin >> s >> t >> w;
        v1.push_back(make_pair(make_pair(t,s),w));
    }
    vector<ll> dp;
    dp.push_back(0);
    sort(v1.begin(),v1.end());
    for(ll i = 1; i <= n; i++){
        ll tmp = bi_se(0,i - 1,v1[i].start);
        ll ch = dp[tmp] + v1[i].weight;
        dp.push_back(max(ch,dp[i-1]));
    }
    cout << dp[n] << "\n";
    return 0;
}

```

3.2 Counting Tilings

```

// n*m put 1*2 2*1
map<int, vi> mp;
ll n;
ll m;
ll M = 1e9+7;
void Generate(ll ind ,ll n, ll prev, ll next
) {
    if(ind == n){
        mp[next].push_back(prev);
        return;
    }
    if((prev & (1 << ind)) != 0){
        Generate(ind + 1, n, prev, next);
    }
}

```

```

        return;
    }
    if(ind < n -1 && (prev & (1 << ind+1)) ==
        0)
        Generate(ind + 2, n, prev, next);
    next |= (1 << ind);
    Generate(ind + 1, n, prev, next);
}
ll solution(){
    ll limit = (1 << n);
    for(ll i = 0; i < limit; i++){
        Generate((1<<0), n, i, (1<<0));
    }
    vi preRow(limit);
    vi DP(limit);
    preRow[0] = 1;
    for(ll i = 0; i < m; i++){
        for(ll j = 0; j < limit; j++){
            for(auto x : mp[j])
                DP[j] = (DP[j] + preRow[x]) % M;
        }
        swap(preRow,DP);
        fill(all(DP),0);
    }
    return preRow[0] % M;
}

```

```

int main(){
    // ios::sync_with_stdio(0);
    // cin.tie(0);
    n = nxt();
    m = nxt();
    cout << solution() << '\n';
}

```

3.3 Elevator Rides

```

int main(){ // once two people
    ios::sync_with_stdio(0);
    cin.tie(0);
    ll n = nxt();
    ll x = nxt();
    vi v(n);
}

```

```

for(auto &x :v) cin >> x;
pair<ll, ll> p[1<<n];
p[0] = {1, 0};
for(int s = 1; s < (1 << n); s++){
    p[s] = {n+1, 0};
    for(int j = 0; j < n; j++){
        if(s & (1<<j)){
            pair<ll, ll> best = p[s ^
                (1<<j)];
            if(best.S + v[j] <= x){
                best.S += v[j];
            }
            else{
                best.F++;
                best.S = v[j];
            }
            p[s] = min(best,p[s]);
        }
    }
}
cout << p[(1 << n)-1].F << '\n';
}

```

3.4 LCS

```

int N1, N2; //
int length[N1+1][N2+1]; // DP
int prev[N1+1][N2+1]; //
int lcs[min(N1,N2)];

void LCS()
{
    for (int i=0; i<=N1; i++) length[i][0]
        = 0;
    for (int j=0; j<=N2; j++) length[0][j]
        = 0;

    for (int i=1; i<=N1; i++)
        for (int j=1; j<=N2; j++)
            if (s1[i] == s2[j])
            {

```

```

        length[i][j] =
            length[i-1][j-1]
            + 1;
        prev[i][j] = 0;
        //
    }
    else
    {
        if
            (length[i-1][j]
            <
            length[i][j-1])
        {
            length[i][j]
            =
            length[i][j-1];
            prev[i][j]
            = 1;
            //
        }
        else
        {
            length[i][j]
            =
            length[i-1][j];
            prev[i][j]
            = 2;
            //
        }
    }

    print_LCS(N1, N2);
}
void print_LCS(int i, int j)
{
    int l = length[i][j];
    //
    // LCS
    while (l > 0)
        if (prev[i][j] == 0)
            //
            l--, lcs[l] = s1[i];
        else if (prev[i][j] == 1)
            //
            j--;
        else if (prev[i][j] == 2)
            //
            i--;
    }
}

```

```

        else if (prev[i][j] == 2)
            //
            i--;
    }
    l = length[i][j];
    for (int i=0; i<l; ++i)
        cout << lcs[i];
}

int LCSLength(string X, string Y){
    int m = X.length(), n = Y.length();
    int curr[n + 1], prev;
    for (int i = 0; i <= m; i++)
    {
        prev = curr[0];
        for (int j = 0; j <= n; j++)
        {
            int backup = curr[j];

            if (i == 0 || j == 0) {
                curr[j] = 0;
            }
            else {
                if (X[i - 1] == Y[j - 1]) {
                    curr[j] = prev + 1;
                }
                else {
                    curr[j] = max(curr[j],
                                   curr[j - 1]);
                }
            }
            prev = backup;
        }
    }
    return curr[n];
}

```

3.5 LIS

```

typedef long long ll;
typedef vector<ll> vi;
typedef pair<ll, ll> pi;

```

```

int main(){
    int num;
    vector<int> a;
    int N;
    cin >> N;
    for(int i = 0; i < N; i++){
        ll tmp = nxt();
        a.push_back(tmp);
    }
    int dp[N+1];
    vector<int> v;
    dp[0] = 1;
    v.push_back(a[0]);
    int L = 1; //LIS length
    for (int i=1; i<N; i++){
        if (a[i] > v.back()){
            v.push_back(a[i]);
            L++;
            dp[i] = L;
        } else {
            auto it = lower_bound(v.begin(),
                                   v.end(), a[i]);
            *it = a[i];
            dp[i] = (int) (it - v.begin() + 1);
        }
    }
    cout << L << '\n';
    vector<int> ans;
    for (int i=N-1; i>=0; i--){
        if (dp[i] == L){
            ans.push_back(a[i]);
            L--;
        }
    }
    reverse(ans.begin(), ans.end());
    for (auto i: ans){
        cout << i << ' ';
    }
    cout << '\n';
    return 0;
}

```

3.6 MaxIntervalSum

```

#include <bits/stdc++.h>
using namespace std;
using ll = long long int;

int main(){
    vector<ll>v1;
    ll t; // test
    cin >> t;
    while (t--){
        v1.clear();
        ll n,last,ans;
        cin >> n; // number
        for(int i = 0;i < n;i++){
            ll tmp;
            cin >> tmp;
            v1.push_back(tmp);
        }
        ll l = 0; // answer_left
        ll r = 0; // right
        ll tmp1 = 0; // temp_left
        ans = v1[0];
        last = v1[0];
        for(int i = 1;i < n;i++){
            if(0 > last){
                tmp1 = i;
            }
            last = max((ll)0,last) + v1[i];
            if(ans < last){
                l = tmp1;
                r = i;
            }
            ans = max(ans,last);
        }
        cout << ans << "\n";
        cout << l + 1 << " " << r + 1 << "\n";
    }
    return 0;
}

```

3.7 coin

```

int main(){

```

```

ios::sync_with_stdio(false);
cin.tie(0);
ll n,m;
cin >> n >> m;
vector<ll> coin(n);
for(auto &x:coin)cin >> x;
vector<ll> DP(m + 1);
for(int i = 1;i < m+1;i++) DP[i] = 1e9;
sort(coin.begin(),coin.end());
reverse(coin.begin(),coin.end());
for(int i = 0;i <= m;i++){
    for(auto x:coin){
        if(i+x<=m){
            DP[i+x] = min(DP[i+x],DP[i] +
                           1);
        }
    }
}
if(DP[m]==1e9)cout << "-1\n";
else cout << DP[m] << '\n';
return 0;
}

```

3.8 pack

```

int main(){
    int n,m;
    cin >> n >> m;
    vector <ll> DP(m+1);
    for(int i = 0;i < n;i++){
        int w,v;
        cin >> w >> v;
        for(int j = m;j >= w;j--){
            DP[j] = max(DP[j],DP[j - w] + v);
        }
    }
    cout << DP[m] << '\n';

    return 0;
}

```

4 Funtions

4.1 BIT

```

#include <bits/stdc++.h>
using namespace std;
using ll = long long int;

typedef struct {
    int set_val, add, sum, val;
} node;
node tree[100];
int n, q, nums[100], _1D_BIT[100],
    _2D_BIT[100][100];

// 1D-BIT
void modify(int x, int mod){
    for(; x <= n; x += (x&-x)){
        _1D_BIT[x] += mod;
    }
}

ll query(int x, int y){
    ll ans = 0;
    for(; x; x -= (x&-x)){
        ans += _1D_BIT[x];
    }
    return ans;
}

// 2D-BIT // Forest Queries ( )
void modify(int x, int y, int mod){
    for(; x <= n; x += (x&-x)){
        for(int tmp = y; tmp <= n; tmp +=
            (tmp&-tmp)){
            _2D_BIT[x][tmp] += mod;
        }
    }
}

ll query(int x, int y){
    ll ans = 0;
    for(; x; x -= (x&-x)){
        for(int tmp = y; tmp; tmp -=
            (tmp&-tmp)){
            ans += _2D_BIT[x][tmp];
        }
    }
}

```

```

    }
    return ans;
}

int main(){
    ios::sync_with_stdio(false);
    cin.tie(0);
    int n;
    memset(bit,0,sizeof(bit));
    for(int i = 1;i <= n;i++){
        cin >> num[i];
        add(i,num[i]);
    }
    return 0;
}

```

4.2 BitMask

```

int main(){
    ios::sync_with_stdio(false);
    cin.tie(0);
    int x = 5328; //
    000000000000000000001010011010000
    cout << __builtin_clz(x) << '\n'; //19 0
    infront
    cout << __builtin_ctz(x) << '\n'; // 4 0
    behind
    cout << __builtin_popcount(x) << '\n'; //
    how many 1
    x |= (1 << k) // k to 1
    x &= ~(1 << k); //k to 0

    for(int i = 31;i >= 0;i--){
        if(x & (1<<i))cout << '1';
        else cout << '0';
    }
    return 0;
}

```

4.3 MatrixQuickPower

```

int x1, x2, a, b, n, mod = 1e9+7;
struct mat{
    long long a[2][2];
    mat(){
        memset(a, 0, sizeof(a));
    }
    mat operator * (const mat &b)const{
        mat ret;
        for (int i = 0; i < 2; i++){
            for (int j = 0; j < 2; j++){
                for (int k = 0; k < 2; k++){
                    ret.a[i][j] = (ret.a[i][j]
                        + a[i][k] * b.a[k][j])
                        % mod;
                }
            }
        }
        return ret;
    }
};

int main(){
    while (cin >> x1 >> x2 >> a >> b >> n){
        mat ret;
        ret.a[0][0] = x1;
        ret.a[1][0] = x2;
        mat p;
        p.a[0][0] = 0; // 0
        p.a[0][1] = 1; // 1
        p.a[1][0] = a; // a * f(n-1)
        p.a[1][1] = b; // b * f(n-2)
        n--;
        while (n){
            if (n & 1){
                ret = p * ret;
            }
            p = p * p;
            n >>= 1;
        }
        cout << ret.a[0][0] << "\n";
    }
    return 0;
}

```

4.4 MosAlgorithm

```

#include <bits/stdc++.h>
#define rep(i, j, k) for(int i = j; i <= k; i++)
#define lrep(i, j, k) for(int i = j; i < k; i++)
#define all(x) x.begin(), x.end()
using namespace std;
typedef struct {
    int l, r, ind;
} query;
query queries[100];
int n, block, nums[100];
bool cmp(query a, query b){
    int block_a = a.l / block;
    int block_b = b.l / block;
    if(block_a != block_b) return block_a <
        block_b;
    return a.r < b.r;
}

void Mo(){
    // sort
    int cl = 1, cr = 0;
    for(auto i : queries){
        while(cl < i.l){} // remove
        while(cr > i.r){} // remove
        while(cl > i.l){} // add
        while(cr < i.r){} // add
    }
}

//
void compress(){
    vector<pair<int, int>> compress(n);
    rep(i, 1, n){
        cin >> nums[i];
        compress[i-1] = {nums[i], i};
    }
    sort(all(compress));
    int pre = compress[0].first, new_num = 0;
    nums[compress[0].second] = 0;
    for(auto it = compress.begin() + 1, end =
        compress.end(); it != end; it++){
        if((*it).first != pre){
            pre = (*it).first;

```

```

        new_num++;
    }
    nums[(it).second] = new_num;
}
}

```

4.5 PYInput

```
print(int(eval(input().replace(' ',''))))
```

```

t = int(input())
for _ in range(0,t):
    tmp = input().split(' ',1)
    n = int(tmp[0])
    arr = list(map(int,tmp[1].split(' ')))
    print(n)
    print(arr)
# 2
# 5 1 2 3 4 5
# 4 1 2 3 4
import sys # EOF
for line in sys.stdin:
    a = int(line)
    if a != 0:
        print(a)
# 1
# 2
# 3

```

4.6 SegmentTree

```

#include <bits/stdc++.h>
using namespace std;
#define ll long long
typedef struct {
    int set_val, add, sum, val;
} node;
int n, q; node tree[100]; int nums[100]; int
    BIT[100];
#define lp 2*now
#define rp 2*now+1

```

```

#define mid (L+R)/2
// Pull
void pull(int now){ // update now with 2
    children
    // lrpnow
    // tree[now].sum = tree[lp].sum +
        tree[rp].sum;
    // tree[now].prefix =
        max(tree[lp].sum+tree[rp].prefix,
            tree[lp].prefix);
    // tree[now].suffix =
        max(tree[lp].suffix+tree[rp].sum,
            tree[rp].suffix);
    // tree[now].middle_max =
        max(max(tree[lp].middle_max,
            tree[rp].middle_max),
            tree[lp].suffix+tree[rp].prefix);
    // tree[now].middle_max =
        max(max(tree[now].middle_max,
            tree[now].prefix), tree[now].suffix);
}
//
void push(int now, int child){
    if(tree[now].set_val){
        tree[child].set_val = 1;
        tree[child].val = tree[now].val;
        tree[child].add = tree[now].add;
    }
    else {
        tree[child].add += tree[now].add;
    }
}
void apply_tag(int now, int L, int R){
    if(tree[now].set_val)
        tree[now].sum = (R-L+1)*tree[now].val;
    tree[now].sum += (R-L+1)*tree[now].add;
    if(L != R){ //
        push(now, lp);
        push(now, rp);
    }
    tree[now].add = tree[now].set_val = 0; //
        Reset
}
// Build
void build(int L, int R, int now){

```

```

    if(L == R){
        // init tree[now];
        return;
    }
    int M = mid;
    build(L, M, lp);
    build(M + 1, R, rp);
    pull(now);
}
// modify
void modify(int l, int r, int L, int R, int
    now){
    if(R < l || r < L || L > n) //
        return;
    if(l <= L && R <= r){
        // modify tree[now];
        // tree[now].add += add; // modify_add
        // tree[now].set_val = 1; // modify_mod
            // tree[now].val = mod;
            // tree[now].add = 0; // Set
        return;
    }
    int M = mid;
    apply_tag(now, L, R);
    modify(l, r, L, M, lp);
    modify(l, r, M+1, R, rp);
    apply_tag(lp, L, M); //
        (Reset)
    apply_tag(rp, M+1, R); //
        (Reset)
    pull(now); // update now with 2 children
}
// query
ll query(int l, int r, int L, int R, int now){
    int M = mid;
    if(R < l || r < L || L > n){
        return 0;
    }
    // apply_tag(now, L, R); //
    if(l <= L && R <= r){
        return tree[now].sum;
    }
    return query(l, r, L, M, lp) +
        query(l, r, M+1, R, rp);
}

```

```
// pizza_queries
// (s < t): dis_l = (pizza[s] - s) + t;
// (t < s): dis_r = (pizza[s] + s) - t;

// List Removals
//
//                      ansmodans
//      - mod ==
//      posnums [ans]      modify
//      tree[pos]

// polynomial queries
//      seg
```

4.7 Treap

```
#include <bits/stdc++.h>
using namespace std;
#define rep(i, j, k) for(int i = j; i <= k; i++)
struct Treap {
    Treap *l, *r;
    int pri, subsize; char val; bool rev_valid;
    Treap(int _val){
        val = _val;
        pri = rand();
        l = r = nullptr;
        subsize = 1; rev_valid = 0;
    }
    void pull(){ // update subsize or other
        information
        subsize = 1;
        for(auto i: {l,r}){
            if(i) subsize += i->subsize;
        }
    };
    int size(Treap *treap) {
        if (treap == NULL) return 0;
        return treap->subsize;
    }
    // lazy
    void push(Treap *t){
```

```
    if(!t) return;
    if(t->rev_valid){
        swap(t->l, t->r);
        if(t->l) t->l->rev_valid ^= 1;
        if(t->r) t->r->rev_valid ^= 1;
    }
    t->rev_valid = false;
}
Treap *merge(Treap *a, Treap *b){
    if(!a || !b) return a ? a : b;
    // push(a); push(b); // lazy
    if(a->pri > b->pri){
        a->r = merge(a->r, b);
        a->pull();
        return a;
    }
    else {
        b->l = merge(a, b->l);
        b->pull();
        return b;
    }
}
pair<Treap*, Treap*> split(Treap *root, int
    k) { // k
    if (root == nullptr) return {nullptr,
        nullptr};
    // push(root); // lazy
    if (size(root->l) < k) {
        auto [a, b] = split(root->r, k
            - size(root->l) - 1);
        root->r = a;
        root->pull();
        return {root, b};
    }
    else {
        auto [a, b] = split(root->l, k);
        root->l = b;
        root->pull();
        return {a, root};
    }
}
void Print(Treap *t){
    if(t){
        Print(t->l);
```

```
        cout << t->val;
        Print(t->r);
    }
}
void substring_rev(){
    int n, m; cin >> n >> m;
    Treap *root = nullptr;
    string str; cin >> str;
    for(auto c : str){
        root = merge(root, new Treap(c));
    }
    rep(i, 1, m){
        int x, y; cin >> x >> y;
        auto [a, b] = split(root, x-1); // a:
            1~x-1, b: x~n
        auto [c, d] = split(b, y-x+1); //
            b
        // c->rev_valid ^= true;
        // push(c);
        b = merge(a, d); //
        root = merge(b, c);
    }
    Print(root);
}
```

4.8 discrete

```
int main(){
    ios::sync_with_stdio(false);
    cin.tie(0);
    int n;
    cin >> n;
    int a[n]; // original number
    vector<int> v; // rank
    for (int i=0; i<n; i++) {
        cin >> a[i];
        v.push_back(a[i]);
    }
    sort(v.begin(), v.end());
    v.resize(unique(v.begin(), v.end()) -
        v.begin()); // if data repeat
    for (int i=0; i<n; i++) {
```



```

        a[i]=lower_bound(v.begin(),
            v.end(), a[i]) - v.begin()
            + 1;
    }
    for (int i=0; i<n; i++) {
        cout << a[i] << ' ';
    }
    cout << endl;
    return 0;
}

```

4.9 qmul

```

ll qmul(ll x,ll y,ll m){
    ll res = 0;
    for(;y > 0;y >= 1,x = (x+x) % m){
        if(y & 1)res = (res+x) % m;
    }
    return res;
}

```

4.10 qpow

```

ll qpow(ll a,ll n,ll m){
    ll res = 1;
    while(n > 0){
        if(n & 1){
            res = res * a % m;
        }
        a = a * a % m;
        n >>= 1;
    }
    return res % m;
}

```

5 Geometry

5.1 ConvexHull

```

vector<pii> P,L,U;
ll cross(pii o, pii a,pii b){ // OA OB >0
    counterclock
    return (a.F - o.F) * (b.S - o.S) -
        (a.S-o.S) * (b.F-o.F);
}
ll Andrew_monotone_chain(ll n){
    sort(P.begin(), P.end());
    ll l = 0, u = 0; // upper and lower hull
    for (ll i=0; i<n; ++i){
        while (l >= 2 && cross(L[l-2], L[l-1],
            P[i]) <= 0){
            l--;
            L.pop_back();
        }
        while (u >= 2 && cross(U[u-2], U[u-1],
            P[i]) >= 0){
            u--;
            U.pop_back();
        }
        l++;
        u++;
        L.push_back(P[i]);
        U.push_back(P[i]);
    }
    cout << l << ' ' << u << '\n';
    return l + u;
}
int main(){
    ll n,x,y;
    cin >> n;
    for(ll i = 0;i < n;i++){
        cin >> x >> y;
        P.push_back({x,y});
    }
    ll ans = Andrew_monotone_chain(n) - 2;
    cout << ans << "\n";
    return 0;
}

```

5.2 CrossProduct

```

const double EPS = 1e-9;
struct point{
    double x, y;
    point operator * (ll a){return {a * x, a *
        y};}
    point operator + (point b){return {x +
        b.x, y + b.y};}
    point operator - (point b){return {x -
        b.x, y - b.y};}
    double operator * (point b){return x * b.x
        + y * b.y;}
    double operator ^ (point b){return x * b.y
        - y * b.x;}
    bool operator < (point b){return x == b.x
        ? y < b.y : x < b.x;}
};
// len
double abs(point a){return sqrt(a.x * a.x +
    a.y * a.y);}
int sign(double a){
    if(abs(a) < EPS)
        return 0;
    else
        return (a > 0 ? 1 : -1);
}
//cross product
int ori(point a,point b,point c){
    return sign((b - a) ^ (c - a));
}
bool colinear(point a,point b,point c){
    return sign((b - a) ^ (c - a)) == 0;
}
bool between(point a,point b,point c){ // c
    between a and b
    if(!colinear(a,b,c))
        return false;
    return sign((a - c) * (b - c)) <= 0;
}
bool intersect(point a,point b,point c,point
    d){ // line(a,b) line(c,d)
    int abc = ori(a,b,c);
    int abd = ori(a,b,d);
    int cda = ori(c,d,a);
    int cdb = ori(c,d,b);
    if(abc == 0 || abd == 0)

```

```

        return between(a,b,c) ||
               between(a,b,d) || between(c,d,a)
               || between(c,d,b);
    return abc * abd <= 0 && cda * cdb <= 0;
}
int main(){
    int n;
    cin >> n;
    point p[1010];
    cin >> p[0].x >> p[0].y;
    ll ans = 0;
    for(int i = 1; i < n; i++){
        cin >> p[i].x >> p[i].y;
        ans += (p[i] ^ p[i - 1]);
    }
    ans += (p[0] ^ p[n - 1]);
    cout << abs(ans) << '\n';

    return 0;
}

```

6 Graph

6.1 BellmanFord

```

int main(){
    ll n = nxt();
    ll m = nxt();
    vector<tuple<ll,ll,ll> > edges;
    for(int i = 0; i < m; i++){
        ll a = nxt();
        ll b = nxt();
        ll w = nxt();
        edges.push_back({a,b,w});
    }
    vi dis(n+1);
    for(int i = 1; i <= n; i++){
        dis[i] = 1e10;
    }
    dis[1] = 0;
    vi par(n+1);
    ll f; //log(nm);

```

```

    for(int i = 0; i <= n; i++){ // n-1
        f = -1;
        for(auto e:edges){
            ll a, b, w;
            tie(a,b,w) = e;
            if(dis[b] > dis[a]+w){
                dis[b] = dis[a]+w;
                par[b] = a;
                f = b;
            }
        }
    }
    if(f != -1){
        queue<ll> q;
        cout << "YES\n";
        for(int i = 0 ; i < n+1; i++) f = par[f];
        vi cycle;
        for(ll v = f; v = par[v]){
            cycle.push_back(v);
            if(v == f && cycle.size() > 1){
                break;
            }
        }
        reverse(all(cycle));
        for(auto x:cycle){
            cout << x << ' ';
        }
    }
    else cout << "NO\n";

    return 0;
}

```

6.2 Bipartite Matching

```

#include <bits/stdc++.h>
using namespace std;
int n = 510;
int m = 510;
int mx[510], my[510]; // match x match y
bool vy[510]; // Graph Traversal visit
bool adj[510][510]; // adjacency matrix
bool DFS(int x){

```

```

    for (int y = 1; y <= m; y++){
        if (adj[x][y] && !vy[y]){
            vy[y] = true;
            if (my[y] == -1 || DFS(my[y])){
                mx[x] = y;
                my[y] = x;
                return true;
            }
        }
    }
    return false;
}

int main(){
    int k,a,b;
    cin >> n >> m; // boy girl
    cin >> k; // edges
    for(int i = 1; i <= n; i++){
        for(int j = 1; j <= m; j++){
            adj[i][j] = 0;
        }
    }
    for(int i = 0; i < k; i++){
        cin >> a >> b;
        adj[a][b] = 1;
    }
    for(int i = 1; i <= n; i++){
        mx[i] = -1;
    }
    for(int i = 1; i <= m; i++){
        my[i] = -1;
    }
    int c = 0;
    for (int x = 1; x <= n; x++){
        if (mx[x] == -1){
            for(int i = 1; i <= m; i++){
                vy[i] = 0;
            }
            if (DFS(x)) c++;
        }
    }
    cout << c << "\n"; //pairs
    for(int i = 1; i <= n; i++){ // boy to girl
        if(mx[i] != -1) cout << i << " " <<
            mx[i] << "\n";
    }
}

```

```
    return 0;
}
```

6.3 DFSandBFS

```
ll N = 1e6;
vi adj[N];
bool vis[N];
void DFS(ll s){
    if(vis[s])return;
    vis[s] = true;
    for(auto u:adj[s]){
        DFS(u);
    }
}
ll timer;
void dfs(ll now, ll pa) {
    pos[now] = ++timer;
    add(timer,v[now]);
    sz[now] = 1;
    for (ll v : g[now]) {
        if (v == pa) continue;
        dfs(v, now);
        sz[now] += sz[v];
    }
}
queue<ll>q;
ll dis[N];
void BFS(ll x){
    vis[x] = true;
    dis[x] = 0;
    q.push(x);
    while(!q.empty()){
        ll s = q.front();q.pop();
        for(auto u:adj[s]){
            if(vis[u])continue;
            vis[u] = true;
            dis[u] = dis[s]+1;
            q.push(u);
        }
    }
}
```

6.4 Dijkstra

```
int main(){// 0(n+mlogm) no negative edge
    ll n = nxt();
    ll m = nxt();
    vector<vector<pi> > adj(n+1);
    for(int i = 0 ;i < m;i++){
        ll a = nxt();
        ll b = nxt();
        ll w = nxt();
        adj[a].pb({b,w});
        // adj[b].pb({a,w});
    }
    ll x = 1;
    ll dis[n+1];
    priority_queue<pi> pq;
    bool vis[n+1] = {0};
    for(int i = 1;i <= n;i++)dis[i] = 1e17;
    dis[x] = 0;
    pq.push({0,x});
    while(!pq.empty()){
        ll a = pq.top().second;pq.pop();
        if(vis[a])continue;
        vis[a] = true;
        for(auto u:adj[a]){
            ll b = u.first,w = u.second;
            if(dis[a] + w < dis[b]){
                dis[b] = dis[a]+w;
                pq.push({-dis[b],b});
            }
        }
    }
    for(int i = 1;i <= n;i++){
        cout << dis[i] << ' ';
    }
    return 0;
}
```

6.5 Dinic

```
#include <bits/stdc++.h>
using namespace std;
bool vis[505];
```

```
int lev[505], n, m, ans;
typedef struct {
    int to, w, rev_ind;
} edge;
vector<edge> adj[505];
bool label_level(){ // return false
    memset(lev, -1, sizeof(lev));
    lev[1] = 0;
    queue<int> q; q.push(1);
    while(!q.empty()){
        int u = q.front(); q.pop();
        for(auto i : adj[u]){
            if(i.w > 0 && lev[i.to] == -1){
                q.push(i.to);
                lev[i.to] = lev[u] + 1;
            }
        }
    }
    return (lev[n] == -1 ? false : true);
}
int dfs(int u, int flow){
    if(u == n) return flow;
    for(auto &i : adj[u]){
        if(lev[i.to] == lev[u] + 1 &&
           !vis[i.to] && i.w > 0) {
            vis[i.to] = true;
            int ret = dfs(i.to, min(flow, i.w));
            if(ret > 0) {
                i.w -= ret;
                adj[i.to][i.rev_ind].w += ret;
                return ret;
            }
        }
    }
    return 0; // return 0
}
void dinic(){
    while(label_level()){
        while(1){
            init(vis, 0);
            int tmp = dfs(1, inf);
            if(tmp == 0) break;
            ans += tmp;
        }
    }
}
```

```

    }
}
void build(){
    rep(i, 1, m){
        int u, v, w; cin >> u >> v >> w;
        adj[u].push_back({v, w,
            (int)adj[v].sz}); // ind
        adj[v].push_back({u, 0,
            (int)adj[u].sz-1}); //
            push -1
    }
}
// Police
// Chaseadjori
// Dinicdfs2reachu
// && w == 0 && vreach
void dfs2(int now, unordered_set<int> &reach){
    if(!vis[now]){
        vis[now] = 1;
        reach.insert(now);
        for(auto i : adj[now]){
            if(i.w > 0){
                dfs2(i.to, reach);
            }
        }
    }
}
// Dinicw == 0
// Distinct Route
//
dfs ans
bool get_road(int now, vector<int> &ans,
    vector<bool> &vis){
    if(now == 1) return true;
    for(auto &v : adj[now]){
        if(v.arg_valid && !vis[v.to]){
            ans.push_back(v.to);
            vis[v.to] = true;
            bool flag = get_road(v.to, ans,
                vis);
            if(flag){

```

```

                v.arg_valid = false;
                return true;
            }
            ans.pop_back();
        }
    }
    return false;
}

```

6.6 FloydWarshall

```

int main(){//O(n^3)
    ll n = nxt();
    ll m = nxt();
    ll q = nxt();
    ll adj[n+1][n+1] = {0};
    ll dis[n+1][n+1];
    for(int i = 0; i < m; i++){
        ll a = nxt();
        ll b = nxt();
        ll w = nxt();
        if(adj[a][b]w = min(adj[a][b], w);
        adj[a][b] = w;
        adj[b][a] = w;
    }
    for(int i = 1; i <= n; i++){
        for(int j = 1; j <= n; j++){
            if(i == j)dis[i][j] = 0;
            else if(adj[i][j]) dis[i][j] =
                adj[i][j];
            else dis[i][j] = 1e17;
        }
    }
    for(int k = 1; k <= n; k++){
        for(int i = 1; i <= n; i++){
            for(int j = 1; j <= n; j++){
                dis[i][j] = min(dis[i][j],
                    dis[i][k]+dis[k][j]);
            }
        }
    }
    for(int i = 0; i < q; i++){
        ll a = nxt();

```

```

        ll b = nxt();
        if(dis[a][b]==1e17)cout << "-1\n";
        else cout <<dis[a][b] << '\n';
    }
    return 0;
}

```

6.7 MCMF

```

#include <bits/stdc++.h>
using namespace std;
// MCMFreturn
typedef struct {
    int from, to, w, cost;
} edge;
int n, m, parcel;
vector<edge> adj; // edge
vector<int> p[505]; // p[u] edge
int now_edge = 0;
void add_edge(int u, int v, int w, int cost){
    adj.push_back({u, v, w, cost});
    p[u].push_back(now_edge);
    now_edge++;
    adj.push_back({v, u, 0, -cost}); //
    p[v].push_back(now_edge);
    now_edge++;
}
ll Bellman_Ford(){
    vector<ll> dis(n+1, inf); dis[1] = 0;
    vector<int> par(m);
    vector<int> flow_rec(n+1, 0); flow_rec[1]
        = 1e9;
    lrep(i, 1, n){
        bool flag = 1;
        int size = adj.sz;
        lrep(i, 0, size){
            auto &[from, to, w, cost] = adj[i];
            if(w > 0 && dis[to] > dis[from] +
                cost){
                flag = 0;
                dis[to] = dis[from] + cost;

```

```

        par[to] = i; //
        flow_rec[to] =
            min(flow_rec[from], w);
    }
    if(flag) break;
}
if(dis[n] == 1e9) return 0;
int mn_flow = flow_rec[n];
int v = n;
while(v != 1){
    int u = adj[par[v]].from;
    adj[par[v]].w -= mn_flow;
    adj[par[v] ^ 1].w += mn_flow;
    v = u;
}
mn_flow = min(mn_flow, parcel);
parcel -= mn_flow;
return mn_flow * dis[n];
}
void solve(){
    cin >> n >> m >> parcel;
    ll ans = 0;
    rep(i, 1, m){
        int u, v, w, cost; cin >> u >> v >> w
        >> cost;
        add_edge(u, v, w, cost);
    }
    while(parcel > 0){
        int tmp = Bellman_Ford();
        if(tmp == 0) break;
        ans += tmp;
    }
    cout << (parcel > 0 ? -1 : ans);
}

```

6.8 Maxflow

```

lli adj[510][510]; // adjacency matrix
lli q[510], *qf, *qb; // BFS queue
lli p[510]; // BFS tree
lli n,m,a,b,c;
lli Edmonds_Karp(lli s, lli t)

```

```

{
    lli f = 0; // max flow
    while(true){ // BFS
        for(int i = 0; i <= n; i++){
            p[i] = -1;
        }
        qf = qb = q;
        p[*qb++ = s] = s;
        while (qf < qb && p[t] == -1)
            for (lli i = *qf++, j = 1; j <= n;
                ++j)
                if (p[j] == -1 && adj[i][j])
                    p[*qb++ = j] = i;
        if (p[t] == -1) break;

        lli df = 1e18;
        for (lli i = p[t], j = t; i != j; i =
            p[j = i]){
            df = min(df, adj[i][j]);
        }
        for (lli i = p[t], j = t; i != j; i =
            p[j = i]){
            adj[i][j] -= df;
            adj[j][i] += df;
        }
        f += df;
    }
    return f;
}
int main(){
    cin >> n >> m; // nodes edges
    for(int i = 1; i <= n; i++){
        for(int j = 1; j <= n; j++){
            adj[i][j] = 0;
        }
    }
    for(int i = 0; i < m; i++){
        cin >> a >> b >> c; // from to capacity
        if(a == b) continue;
        adj[a][b] += c;
    }
    cout << Edmonds_Karp(1,n) << "\n";
    return 0;
}

```

6.9 PlanetsCycles

```

#include <bits/stdc++.h>
#define F first
#define S second
#define PB push_back
#define MP make_pair
#define all(x) (x).begin(), (x).end()
#define FOR(s,a,b) for (int s = a; s <= b;
    s++)
using namespace std;
typedef long long ll;
typedef vector<ll> vi;
typedef pair<ll, ll> pi;

ll nxt() {
    ll x;
    cin >> x;
    return x;
}
vi dis;
vi v;
vector<bool> vis;
ll step;
ll one;
queue<ll> path;
void dfs(ll x){
    path.push(x);
    if(vis[x]){
        step += dis[x];
        return;
    }
    vis[x] = true;
    step++;
    dfs(v[x]);
}
// count pathdis to rep
int main(){
    ios::sync_with_stdio(0);
    cin.tie(0);
    ll n = nxt();
    v.assign(n+1,0);
    dis.assign(n+1,0);
    vis.assign(n+1,false);
    for(int i = 1; i <= n; i++){

```

```

    cin >> v[i];
}
for(int i = 1; i <= n; i++){
    step = 0;
    one = 1;
    dfs(i);
    while(!path.empty()){
        if(path.front() == path.back()){
            one = 0;
        }
        dis[path.front()] = step;
        step -= one;
        path.pop();
    }
}
for(int i = 1; i <= n; i++){
    cout << dis[i] << ' ';
}
cout << '\n';
}

```

6.10 TopologicalSort

```

#include <bits/stdc++.h>
#define F first
#define S second
#define PB push_back
#define MP make_pair
#define all(x) (x).begin(), (x).end()
#define FOR(s,a,b) for (int s = a; s <= b; s++)
using namespace std;
typedef long long ll;
typedef vector<ll> vi;
typedef pair<ll, ll> pi;

ll nxt() {
    ll x;
    cin >> x;
    return x;
}

vector<vi> edge;

```

```

vi vis;
stack<ll> order;
bool dfs(ll u){
    if(vis[u]==2){
        return false;
    }
    else if(vis[u]==1) return true;
    bool cycle = false;
    vis[u] = 1;
    for(auto x:edge[u]){
        cycle = max(cycle, dfs(x));
    }
    order.push(u);
    vis[u] = 2;
    return cycle;
}

int main(){
    ios::sync_with_stdio(0);
    cin.tie(0);
    ll n = nxt();
    vi a(0);
    edge.assign(n+1, a);
    vis.assign(n+1, 0);
    ll m = nxt();
    for(ll i = 0; i < m; i++){
        ll a = nxt();
        ll b = nxt();
        edge[a].push_back(b);
    }
    bool cycle = false;
    for(int i = 1; i <= n; i++){
        cycle = max(cycle, dfs(i));
    }
    if(cycle){
        cout << "IMPOSSIBLE\n";
    }
    else{
        while(order.size()){
            cout << order.top() << ' ';
            order.pop();
        }
    }
}

```

6.11 success

```

ll succ(ll n, ll k){
    if(k == 1) return succ(n);
    return succ(succ(x, k/2), k/2);
}

ll a = v[i];
ll b = v[v[i]];
while(a != b){
    a = v[a];
    b = v[b];
}

a = i;
while(a != b){
    a = v[a];
    b = v[b];
}
ll first = a; // cycle first

b = v[a];
ll length = 1; // cycle
while(a != b){
    b = v[b];
    length++;
}

```

7 Mathematics

7.1 BabyStepGiantStep

```

ll qpow(ll a, ll n, ll m){
    ll res = 1;
    while (n > 0)
    {
        if(n & 1){
            res = res * a % m;
        }
        a = a * a % m;
        n >>= 1;
    }
}

```

```

    return res % m;
}

int main(){
    // a ^ x = b (mod n)
    ll a,b,n,ans;
    map<ll, ll>value;
    while(cin >> a >> b >> n){
        ll minn = __LONG_LONG_MAX__;
        ll m = (ll)sqrt(n)+1;
        value.clear();
        if(b == 1){
            cout << "0\n";
            continue;
        }
        for(int i = 1; i < m; i++){
            value[qpow(a,i*m,n)] = i;
        }
        bool done = false;
        for(int j = 0; j < m; j++){
            ll cur = (qpow(a,j,n) * b) % n;
            if(value[cur]){
                ans = value[cur] * m - j;
                if(ans < n && ans >= 0){
                    done = true;
                    minn = min(minn,ans);
                }
            }
        }
        if(done) cout << minn << "\n";
        else cout << "NOT FOUND\n";
    }
    return 0;
}

```

7.2 CatalanNumbers

```

// Function to print the number
// 2n! / (n + 1)! / n!
ll qpow(ll a, ll n, ll m){
    ll res = 1;
    while(n > 0){
        if(n & 1){
            res = res * a % m;

```

```

        }
        a = a * a % m;
        n >>= 1;
    }
    return res % m;
}

const ll m = 1e9+7;
const ll maxn = 1e6+10;
ll fac[maxn];
ll inv[maxn];
void factioal(){
    fac[0] = 1;
    for(int i = 1; i < maxn; i++){
        fac[i] = fac[i - 1] * i % m;
    }
}

void inverse(){
    inv[0] = 1;
    for(int i = 1; i < maxn; i++){
        inv[i] = qpow(fac[i], m - 2, m);
    }
}

ll catalan(ll n)
{
    // ((()))
    ll res;
    res = fac[2 * n] * inv[n + 1] % m;
    res = res * inv[n] % m;
    return res;
}

int main()
{
    int n;
    cin >> n;
    // Function call
    factioal();
    inverse();
    if(n & 1) cout << '0';
    else cout << catalan(n / 2) << '\n';
    return 0;
}

// there are Cn binary trees of n nodes
// there are Cn1 rooted trees of n nodes

```

7.3 ChineseRemainder

```

ll M = 1;
struct gcdstruct{ // ax + by = d
    ll d;
    ll x;
    ll y;
};

gcdstruct exgcd(ll a, ll b){
    gcdstruct aa, bb;
    if(b == 0){
        aa.d = a;
        aa.x = 1;
        aa.y = 0;
        return aa;
    }
    else{
        bb = exgcd(b, a % b);
        aa.d = bb.d;
        aa.x = bb.y;
        aa.y = bb.x - bb.y * (a / b);
    }
    return aa;
}

ll inverse(ll a, ll b){
    gcdstruct aa;
    aa = exgcd(a, b);
    return aa.x;
}

int main(){
    ll n, t1, t2;
    cin >> n; // equations
    vector<ll> v1[2];
    for(int i = 0; i < n; i++){
        cin >> t1 >> t2; // ans % m = a
        v1[0].push_back(t1);
        v1[1].push_back(t2);
        M *= t1;
    }
    ll x = 0;
    for(int i = 0; i < n; i++){
        ll m = v1[0][i];
        ll Mi = (M / m);
        x += (v1[1][i] * ((inverse(Mi, m) + m)
            % m) * Mi) % M;
    }
}

```

```

    //a * t * Mi
}
cout << x % M << "\n";
return 0;
}

```

7.4 Choose

```

ll qpow(ll a,ll n,ll m){
    ll res = 1;
    while(n > 0){
        if(n & 1){
            res = res * a % m;
        }
        a = a * a % m;
        n >>= 1;
    }
    return res % m;
}

const ll m = 1e9+7;
const ll maxn = 1e6+10;
ll fac[maxn];
ll inv[maxn];
void factioal(){
    fac[0] = 1;
    for(int i = 1; i < maxn; i++){
        fac[i] = fac[i - 1] * i % m;
    }
}

void inverse(){
    inv[0] = 1;
    for(int i = 1; i < maxn; i++){
        inv[i] = qpow(fac[i], m - 2, m);
    }
}

ll choose(ll a, ll b){
    return fac[a] * inv[b] % m * inv[a-b] % m;
}

// C(n,k)*C(k,r) = C(n,r) * C(n-r,k-r)
int main(){
    ll n = nxt();
    factioal();
    inverse();

```

```

while(n--){
    ll a = nxt();
    ll b = nxt();
    ll res = choose(a,b);
    cout << res << '\n';
}
return 0;
}

```

7.5 Counting Necklaces

```

ll M = 1e9+7;

ll qpow(ll a,ll n,ll m){
    ll res = 1;
    while(n > 0){
        if(n & 1){
            res = res * a % m;
        }
        a = a * a % m;
        n >>= 1;
    }
    return res % m;
}

// Function to find result using
// Orbit counting theorem
// or Burnside's Lemma
void countDistinctWays(ll n, ll m)
{
    ll ans = 0;
    // According to Burnside's Lemma
    // calculate distinct ways for each
    // rotation
    for (ll i = 0; i < n; i++) {
        // Find GCD
        ll K = __gcd(i, n);
        ans += qpow(m, K, M);
        ans %= M;
    }
    // Divide By N
    ans *= qpow(n, M - 2, M);
    ans %= M;
    // Print the distinct ways

```

```

    cout << ans << endl;
}

// Driver Code
int main()
{
    // N stones and M colors
    ll n,m;
    cin >> n >> m;
    // Function call
    countDistinctWays(n, m);
    return 0;
}

```

7.6 Derangements

```

// Permutation such that no element appears
// in its original position
ll countDer(ll n)
{
    // base case
    if (n == 1 or n == 2) return n - 1;

    // Variable for just storing
    // previous values
    ll a = 0;
    ll b = 1;

    // using above recursive formula
    for (ll i = 3; i <= n; ++i) {
        ll cur = (i - 1) * (a + b);
        a = b;
        b = cur;
    }

    // Return result for n
    return b;
}

// Driver Code
int main()
{
    cout << "Count of Derangements is " <<
        countDer(4);

```



```
    return 0;
}
```

7.7 Fermat's Little Theorem

```
if(p is prime)
a ^ (p-1) = 1 (mod p)
```

7.8 Inverse

```
struct gcdstruct{ // ax + by = d
    ll d;
    ll x;
    ll y;
};
ll gcd(ll a, ll b){
    return b ? gcd(b, a % b) : a;
}
gcdstruct exgcd(ll a, ll b){
    gcdstruct aa, bb;
    if(b == 0){
        aa.d = a;
        aa.x = 1;
        aa.y = 0;
        return aa;
    }
    else{
        bb = exgcd(b, a % b);
        aa.d = bb.d;
        aa.x = bb.x;
        aa.y = bb.y * (a / b);
    }
    return aa;
}
ll inverse(ll a, ll b){
    gcdstruct aa;
    aa = exgcd(a, b);
    return (aa.x % b + b) % b;
}
int main(){
    ll a, n;
```

```
while(cin >> a >> n){
    a %= n;
    if(gcd(a, n) > 1){
        cout << "No Inverse\n";
        continue;
    }
    ll ans = inverse(a, n);
    if(!ans) cout << "No Inverse\n";
    else cout << ans << "\n";
}
return 0;
}
```

7.9 Josephus Problem

```
int josephus2(int n)
{
    int p = 1;
    while (p <= n)
        p *= 2;
    return (2 * n) - p + 1;
}
int josephus(int n, int k) { // from 0 index
    if (n == 1)
        return 0;
    if (k == 1)
        return n-1;
    if (k > n)
        return (josephus(n-1, k) + k) % n;
    int cnt = n / k;
    int res = josephus(n - cnt, k);
    res -= n % k;
    if (res < 0)
        res += n;
    else
        res += res / (k - 1);
    return res;
}
ll Josephus2(ll n, ll k){
    if(n==1) return 1;
    if(k<=(n+1)/2)
    {
        if(2*k>n) return (2*k)%n;
```

```
        else return 2*k;
    }
    ll temp=f(n/2,k-(n+1)/2);
    if(n%2==1) return 2*temp+1;
    return 2*temp-1;
}
```

7.10 Nim Game

```
void nimGame(){ // removes the last stick wins the game
    int n;
    cin >> n;
    ll x = 0;
    for(int i = 0; i < n; i++){
        ll tmp;
        cin >> tmp;
        x ^= tmp;
    }
    if(x) cout << "first\n";
    else cout << "second\n";
}

void nimGame2(){ // removes 1, 2, or 3 sticks
    int n;
    cin >> n;
    ll x = 0;
    for(int i = 0; i < n; i++){
        ll tmp;
        cin >> tmp;
        x ^= (tmp % 4);
    }
    if(x) cout << "first\n";
    else cout << "second\n";
}

int main(){
    int t;
    cin >> t;
    while(t--){
        nimGame2();
    }
    return 0;
}
```

```
}

```

7.11 PrimeFactor

```
#include <bits/stdc++.h>
using namespace std;
typedef long long int ll;
vector<ll> ans;
ll qmul(ll x,ll y,ll m){
    ll res = 0;
    for(;y > 0;y >>= 1,x = (x+x) % m){
        if(y & 1)res = (res+x) % m;
    }
    return res;
}
ll GCD(ll a, ll b){
    return b ? GCD(b, a % b) : a;
}
ll qpow(ll a,ll n,ll m){
    ll res = 1;
    while(n > 0){
        if(n & 1){
            res = qmul(res,a,m);
        }
        a = qmul(a,a,m);
        n >>= 1;
    }
    return res % m;
}
bool Isprime(ll n){ // O(k log N)
    if(n==2) return true;
    if((!(n & 1))|| n==1) return false;
    ll d = n - 1;
    ll s = 0;
    while(!(d & 1)){
        s++;
        d/=2;
    }
    for(int i = 0;i < 10;i++){
        ll x = rand() % (n-1) + 1;
        ll tmp = d;
        if(qpow(x,d,n) == 1){
            continue;

```

```

    }
    else{
        bool done = false;
        for(int j = 0;j < s;j++){
            if(qpow(x,tmp,n) == n-1){
                done = true;
                break;
            }
            tmp *= 2;
        }
        if(!done) return false;
    }
    return true;
}
ll f(ll x,ll c,ll n){
    return (qmul(x,x,n) + c) %n;
}
void factor(ll n){ // O(N1/4)
    if(n == 1)return;
    if(n == 4){
        ans.push_back(2);
        ans.push_back(2);
        return;
    }
    if(Isprime(n)){
        ans.push_back(n);
        return;
    }
    else{
        again:;
        ll c = rand() % (n-1) + 1;
        ll x = rand() % (n-1) + 1;
        ll y = x;
        ll d;
        bool done = false;
        do{
            x = f(x,c,n);
            y = f(f(y,c,n),c,n);
            d = GCD(abs(x-y),n);
            if(d > 1 && d < n){
                done = true;
                break;
            }
        }while(x!=y);
    }
}

```

```

        if(done){
            factor(d);
            factor(n/d);
        }
        else goto again;
    }
    return ;
}
int main(){
    ll n;
    while(cin >> n){
        ans.clear();
        factor(n);
        sort(ans.begin(),ans.end());
        ll tmp = ans[0],cnt = 0;
        for(ll i = 0;i < (ll)ans.size();i++){
            if(ans[i]==tmp){
                cnt++;
            }
            else{
                cout << tmp << " " << cnt << "
";
                tmp = ans[i];
                cnt = 1;
            }
        }
        cout << tmp << " " << cnt << "\n";
    }
    return 0;
}

```

7.12 PrimeSieve

```
const int N = 100000010;
bool not_prime[N];
vector<int>prime;
void linear_sieve(){
    int i;
    prime.push_back(2);
    for (i = 3; i*i<=N; i+=2){
        if (!not_prime[i]) prime.push_back(i);
        for(int j = i*i,k=i+i;j <= N;j += k){
            not_prime[j] = true;

```

```

    }
}
for(;i <= N;i+=2){
    if(!not_prime[i]){
        prime.push_back(i);
    }
}
}

void Divisors(){
    vi p(1000010);
    for(int i = 2;i < N;i++){
        if(!p[i]){
            for(int j = i;j < N;j+= i){
                p[j] = i;
            }
        }
    }
}

// sum of factors
// pi^(ai+1)-1/(pi-1)

// product of factors
// n ^ (numbers of factors)
// pi = pi-1^(ki+1) * xi ^ (ki*(ki+1)/2) ^
    Ci-1
int main(){
    linear_sieve();
    for(auto x:prime) cout << x << ' ';
    return 0;
}

```

7.13 PythagoreanTriplets

```

#include <bits/stdc++.h>
void pythagoreanTriplets(int limit)
{
    // triplet: a^2 + b^2 = c^2
    ll a, b, c = 0;
    // loop from 2 to max_limit
    ll m = 2;
    // Limiting c would limit
    // all a, b and c

```

```

while (c < limit) {
    // now loop on j from 1 to i-1
    for (ll n = 1; n < m; ++n) {
        // Evaluate and print triplets using
        // the relation between a, b and c
        a = m * m - n * n;
        b = 2 * m * n;
        c = m * m + n * n;
        if (c > limit)
            break;
        printf("%d %d %d\n", a, b, c);
    }
    m++;
}
}

```

7.14 Wilson's Theorem

```

if(n is prime)
    (n - 1)! % n = (n - 1)

```

8 Sorting

8.1 BinarySearch

```

bool ans[100100];
ll bi_se(ll a,ll b){
    while (b > a)
        { //0 0 0 0...1 1 1 1
            ll mid = (a + b) / 2;
            if(!ans[mid]) a = mid + 1;
            else b = mid;
        }
    return a;
}

bool valid(int x){
    if (x > 10) return true;
    else return false;
}

int main(){

```

```

ll k = -1,z = 20;
for(ll i = z;i >= 1;i /= 2){
    while(!valid(k+i))k+=i;
}
ll ans = k + 1;
cout << ans << '\n';
sort(all(v));
v.erase(unique(all(v),v.end())); // left
    unique value
return 0;
}

```

8.2 MergeSort

```

vector<int>a,tmp;
int ans;//ans = 0 change time
void msort(int s,int t) { //start end
    if(s==t) return ;
    int mid=(s+t)>>1;
    msort(s,mid),msort(mid+1,t);

    int i=s,j=mid+1,k=s;
    while(i<=mid && j<=t) {
        if(a[i]<=a[j])
            tmp[k]=a[i],k++,i++;
        else
            tmp[k]=a[j],k++,j++,ans+=mid-i+1;
    }
    while(i<=mid) tmp[k]=a[i],k++,i++;
    while(j<=t) tmp[k]=a[j],k++,j++;
    for(int i=s;i<=t;i++) a[i]=tmp[i];
    return ;
}

int main(){
    int t;
    cin >> t;
    while (t-->0)
    {
        ans = 0;
        int n,tmp2;
        cin >> n;
        a.clear();
        tmp.clear();

```

```

    for(int i = 0; i < n; i++){
        cin >> tmp2;
        a.push_back(tmp2);
        tmp.push_back(tmp2);
    }
    msort(0, n-1);
    cout << ans << '\n';
}
return 0;
}

```

8.3 TrafficLights

```

// after each add light longest distance
set<ll> st = {0, x};
multiset<ll> mst = {x};
for(int i = 0; i < n; i++){
    ll k = nxt();
    auto it1 = st.upper_bound(k);
    auto it2 = it1;
    it2--;
    mst.erase(mst.find(*it1 - *it2));
    mst.insert(*it1 - k);
    mst.insert(k - *it2);
    st.insert(k);
    auto it = mst.end();
    it--;
    cout << *it << ' ';
}

```

9 Tree

9.1 DFStree

```

int timer;
const int maxn = 200200;
int pos[maxn], sz[maxn];
vector<int> g[maxn];
void dfs(int now, int pa) {
    pos[now] = ++timer;

```

```

    sz[now] = 1;
    for (int v : g[now]) {
        if (v == pa) continue;
        dfs(v, now);
        sz[now] += sz[v];
    }
}
cout << query(pos[a] + sz[a] - 1) -
    query(pos[a] - 1) << "\n";

```

9.2 DistanceTree

```

int main(){
    int t;
    cin >> t;
    queue<ll> q1;
    ll arr[5010][4]; // 0 parent 1 length 2 ch
    3 ch n
    while(t--){
        ll x, y;
        ll n;
        ll ans = 0;
        cin >> n;
        for(int i = 1; i <= n; i++){
            arr[i][2] = 0;
            arr[i][3] = 1;
        }
        for(int i = 2; i <= n; i++){
            cin >> arr[i][0];
            arr[arr[i][0]][2] ++;
        }
        for(int i = 2; i <= n; i++){
            cin >> arr[i][1];
        }
        for(int i = 2; i <= n; i++){
            if(arr[i][2] == 0) q1.push(i);
        }
        while(!q1.empty()){
            x = q1.front();
            q1.pop();
            y = arr[x][0];
            ans += (n - arr[x][3]) * arr[x][3] *
                2 * arr[x][1];

```

```

            arr[y][3] += arr[x][3];
            arr[y][2] --;
            if(y == 1) continue;
            if(arr[y][2] == 0) q1.push(y);
        }
        cout << ans << "\n";
    }
    return 0;
}

```

9.3 IndptTreeDFS

```

int yes[1002];
int no[1002];
vector<int> child[1002];
void ini(){
    memset(yes, 0, sizeof(yes));
    memset(no, 0, sizeof(no));
    for (int i = 0; i < 1002; i++)
        child[i].clear();
}
void DFS(int a){
    for (auto i : child[a]){
        DFS(i);
        yes[a] += no[i];
        no[a] += max(yes[i], no[i]);
    }
}
int main(){
    int T, a, b, c;
    cin >> T;
    while(T--){
        ini();
        cin >> a >> b;
        yes[1] = b;
        for (int i = 2; i <= a; i++){
            cin >> c >> b;
            yes[i] = b;
            child[c].push_back(i);
        }
        DFS(1);
        cout << max(yes[1], no[1]) << '\n';
    }
}

```

}

9.4 LCAbi

```
int parent[20][300100]; // n < 2^20
int depth[300100];
int LCA(int u, int v){
    if(depth[u] > depth[v]) swap(u,v);
    int diff = depth[v] - depth[u];
    for(int i = 19; i >= 0; i--){
        if(diff & (1 << i)){
            v = parent[i][v];
        }
    }
    if(u == v) return u;
    for(int i = 19; i >= 0; i--){
        if(parent[i][u] != parent[i][v]){
            u = parent[i][u];
            v = parent[i][v];
        }
    }
    return parent[0][u];
}

int main(){
    int n, m;
    cin >> n >> m; // nodes tests
    parent[0][1] = 1; // root
    depth[1] = 1; // root
    for(int i = 2; i <= n; i++){
        int tmp;
        cin >> tmp; // parent
        parent[0][i] = tmp;
        depth[i] = depth[tmp] + 1;
    }
    for(int i = 1; i < 20; i++){
        for(int j = n; j >= 1; j--){
            parent[i][j] =
                parent[i-1][parent[i-1][j]];
        }
    }
    int u, v;
```

```
for(int i = 0; i < m; i++){
    cin >> u >> v;
    int ans = LCA(u, v);
    cout << ans << "\n";
    //-(depth[ans] - depth[u] + depth[ans]
    - depth[v]) level
}
return 0;
}
```

9.5 MST

```
struct Union_find{
    ll link[100100];
    ll size[100100];
    void init(){
        for(int i = 0; i < 100100; i++){
            link[i] = i;
            size[i] = 1;
        }
    }
    ll find(ll x){
        if(x == link[x]) return x;
        return link[x] = find(link[x]);
    }
    bool same(ll x, ll y){
        return find(x) == find(y);
    }
    void unite(ll x, ll y){
        x = find(x);
        y = find(y);
        if(size[x] < size[y]) swap(x, y);
        size[x] += size[y];
        link[y] = x;
    }
} uf;

int main(){
    ll n = nxt();
    ll m = nxt();
    vector<tuple<ll, ll, ll> > v;
```

```
uf.init();
for(int i = 0; i < m; i++){
    ll a = nxt();
    ll b = nxt();
    ll w = nxt();
    v.push_back({w, a, b});
}
sort(all(v));
ll ans = 0;
for(auto x: v){
    ll a, b, w;
    tie(w, a, b) = x;
    if(!uf.same(a, b)){
        uf.unite(a, b);
        ans += w;
    }
}
if(uf.size[uf.find(1)] == n) cout << ans <<
    '\n';
else cout << "IMPOSSIBLE\n";
return 0;
}
```

10 string

10.1 LongestSubstringWithoutRep

```
ll ans = 0;
ll i = 1;
for(ll i = 1; i <= n; i++) {
    cin >> v[i];
}
for(ll j = 1; j <= n; j++){
    i = max(i, mp[v[j]]+1);
    ans = max(ans, j - i + 1);
    mp[v[j]] = j;
}
```
