## 随机数rng()

```
std::mt19937
rng(std::chrono::steady_clock::now().time_since_epoch().count());
```

# 基础算法

## 二维前缀和

```
int n, m, q; std::cin >> n >> m >> q;
std::vector<std::vector<int>> s(n + 1, std::vector<int>(m + 1));
for(int i = 1; i <= n; i ++)
    for(int j = 1; j <= m; j ++)
        std::cin >> s[i][j], s[i][j] += s[i - 1][j] + s[i][j - 1] - s[i -
1][j - 1];
while(q --)
{
    int x1, x2, y1, y2;
    std::cin >> x1 >> y1 >> x2 >> y2;
    std::cout << s[x2][y2] - s[x1 - 1][y2] - s[x2][y1 - 1] + s[x1 - 1][y1 -
1] << "\n";
}
```

## 二维差分

```
cin >> n >> m >> q;
for(int i = 1; i <= n; i ++)
    for(int j = 1; j <= m; j ++)
        cin >> a[i][j];
while(cin >> u >> v >> x >> y >> c)
{
    x ++;
    y ++;
    b[u][v] +=c;
    b[x][v] -=c;
    b[u][y] -=c;
    b[x][y] +=c;
}
for(int i = 1; i <= n; i++)
    for(int j = 1; j <= m; j ++)
        b[i][j] += b[i - 1][j] + b[i][j - 1] - b[i - 1][j - 1], cout <<
b[i][j] + a[i][j] << " \n"[j == m];
```

## 归并求逆序对

```
auto merge_sort = [&](auto merge_sort, int l, int r) -> LL{
    if(l >= r) return 0;
    int mid = l + r >> 1;
    LL res = merge_sort(merge_sort, l, mid) + merge_sort(merge_sort, mid +
1, r);
    int i = l, j = mid + 1;
    vector<int> t;
    while(i <= mid && j <= r)
    {
        if(v[i] <= v[j])
            t.push_back(v[i ++ ]);
        else
        {
            res += mid - i + 1;
            t.push_back(v[j ++]);
        }

    }
    while(i <= mid)
        t.push_back(v[i ++]);
    while(j <= r)
        t.push_back(v[j ++]);
    i = l;
    for(auto& ite : t)
        v[i ++] = ite;
    return res;
};
LL res = merge_sort(merge_sort, 0, n - 1);
```

## 取模

```
constexpr int P = 1000000007;
typedef long long LL;
// assume -P <= x < 2P
int norm(int x) {
    if (x < 0) {
        x += P;
    }
    if (x >= P) {
        x -= P;
    }
    return x;
}
template<class T>
T power(T a, LL b) {
    T res = 1;
    for (; b; b /= 2, a *= a) {
        if (b % 2) {
            res *= a;
        }
    }
    return res;
}
struct Z {
    int x;
    Z(int x = 0) : x(norm(x)) {}
    Z(LL x) : x(norm(x % P)) {}
    int val() const {
        return x;
    }
    Z operator-() const {
        return Z(norm(P - x));
    }
    Z inv() const {
        assert(x != 0);
        return power(*this, P - 2);
    }
    Z &operator*=(const Z &rhs) {
        x = LL(x) * rhs.x % P;
        return *this;
    }
    Z &operator+=(const Z &rhs) {
        x = norm(x + rhs.x);
        return *this;
    }
    Z &operator-=(const Z &rhs) {
        x = norm(x - rhs.x);
        return *this;
    }
    Z &operator/=(const Z &rhs) {
        return *this *= rhs.inv();
    }
    friend Z operator*(const Z &lhs, const Z &rhs) {
        Z res = lhs;
        res *= rhs;
        return res;
    }
    friend Z operator+(const Z &lhs, const Z &rhs) {
        Z res = lhs;
        res += rhs;
        return res;
    }
    friend Z operator-(const Z &lhs, const Z &rhs) {
        Z res = lhs;
        res -= rhs;
        return res;
    }
    friend Z operator/(const Z &lhs, const Z &rhs) {
        Z res = lhs;
        res /= rhs;
        return res;
    }
    friend std::istream &operator>>(std::istream &is, Z &a) {
        LL v;
        is >> v;
        a = Z(v);
        return is;
    }
    friend std::ostream &operator<<(std::ostream &os, const Z &a) {
        return os << a.val();
    }
```

```
};
```

# 数据结构

## 单调栈

```
int n; cin >> n;
vector<int> v(n);
for(int& x : v) cin >> x;
stack<int> s;
for(int& x : v)
    if(!s.size()) s.push(x), cout << -1 << " ";
else
{
    while(s.size() && s.top() >= x) s.pop();
    if(!s.size()) cout << -1 << " ";
    else
        cout << s.top() << " ";
    s.push(x);
}
```

## 单调队列

```
int n, k; cin >> n >> k;
 b        r<int> a(n);
for(int& x : a)cin >> x;
deque<int> q;
for(int i = 0; i < n; i ++) // find the k size window minv
{
    if(q.size() && i - q.front() >= k) q.pop_front();
    while(q.size() && a[i] <= a[q.back()]) q.pop_back();
    q.push_back(i);
    if(i - k + 1 >= 0)
    {
        cout << a[q.front()] << " ";
    }
}
q.clear`
cout << "\n";
for(int i = 0; i < n; i ++) // find the k size window maxv
{
    if(q.size() && i - q.front() >= k) q.pop_front();
    while(q.size() && a[i] >= a[q.back()]) q.pop_back();
    q.push_back(i);
    if(i - k + 1 >= 0)
    {
        cout << a[q.front()] << " ";
    }
}
```

## 树状数组

```
struct BIT{
    vector<int> tr;
    int n;
    BIT(int n) : tr(n + 1){this->n = n;}
    int lowbit(int x){
        return x & -x;
    }
    void insert(int x, int val){
        for(; x <= n; x += lowbit(x))
            tr[x] += val;
    }
    int sum(int x){
        int res = 0;
        for(; x; x -= lowbit(x))
            res += tr[x];
        return res;
    }
}bit;
```

## 并查集

```
struct DSU {
    std::vector<int> p, siz;
    DSU(int n) : p(n+1), siz(n+1, 1) { std::iota(p.begin(), p.end(), 0); }
    int find(int x) {
        return p[x] == x ? x : p[x] = find(p[x]);
    }
    bool same(int x, int y) { return find(x) == find(y); }
    bool merge(int x, int y) {
        x = find(x);
        y = find(y);
        if (x == y) return false;
        siz[x] += siz[y];
        p[y] = x;
        return true;
    }
    int size(int x) { return siz[find(x)]; }
};
```

## ST表

```
struct RMQ{ // 超时的话记得改写为数组
    int n, m;
    vector<vector<int>> f, g;
    vector<int> q;
    RMQ(int N, int M) : n(N), m(M), f(n + 1, vector<int>(m + 1)), g(n + 1,
vector<int>(m + 1)), q(n + 1){}

    void init(){
        for(int j = 0; j < m; j ++)
            for(int i = 1; i + (1 << j) <= n + 1; i ++)
            {
                if(!j) f[i][j] = q[i], g[i][j] = q[i];
                else
                {
                    f[i][j] = max(f[i][j - 1], f[i + (1 << j - 1)][j - 1]);
                    g[i][j] = min(g[i][j - 1], g[i + (1 << j - 1)][j - 1]);
                }

            }
    }
    int ask_max(int l, int r){
        int k = log2(r - l + 1);
        return max(f[l][k], f[r - (1 << k) + 1][k]);
    }
    int ask_min(int l, int r){
        int k = log2(r - l + 1);
        return min(g[l][k], g[r - (1 << k) + 1][k]);
    }
};
```

# 线段树

## 线段树二分

$ask$的作用

找到一个序列中[l, r] 这段中大于等于$d$的第一个数的下标, 不存在输出$-1$

```

#define ls u << 1
#define rs u << 1 | 1
struct SegTree{

    struct Node{
        // to do

    };

    vector<int> w;
    vector<Node> tr;
    SegTree(int n) :tr(n * 4 + 1), w(n + 1){}
```

```
void pushup(Node& u, Node& l, Node& r){
    // to do

}
void pushup(int u){
    pushup(tr[u], tr[ls], tr[rs]);
}
void build(int u, int l, int r){
    tr[u] = {l, r};
    if(l == r)
    {
        // to do

        return;
    }
    int mid = l + r >> 1;
    build(ls, l, mid);
    build(rs, mid + 1, r);
    pushup(u);
}
void modify(int u, int idx, int val){
    if(tr[u].l == tr[u].r && tr[u].l == idx)
    {
        // to do

        return;
    }
    int mid = tr[u].l + tr[u].r >> 1;
    if(idx <= mid) modify(ls, idx, val);
    else
        modify(rs, idx, val);
    pushup(u);
}

Node query(int u, int L, int R){
    if(tr[u].l >= L && tr[u].r <= R)
        return tr[u];
    int mid = tr[u].l + tr[u].r >> 1;
    if(R <= mid) return query(ls, L, R);
    else if (L > mid) return query(rs, L, R);
    else
    {
        // to do
    }
}
int ask(int u, int L, int R, int d){
    if(tr[u].l == L && tr[u].r == R)
    {
        if(tr[u].maxv < d) return -1;
        if(tr[u].l == tr[u].r) return tr[u].l;
```

```cpp
            int mid = tr[u].l + tr[u].r >> 1;
            if(tr[ls].maxv >= d) return ask(ls, L, mid, d);
            return ask(rs, mid + 1, R, d);
        }
        int mid = tr[u].l + tr[u].r >> 1;
        if(R <= mid) return ask(ls, L, R, d);
        else if(L > mid) return ask(rs, L, R, d);
        else
        {
            int pos = ask(ls, L, mid, d);
            if(pos == -1) return ask(rs, mid + 1, R, d);
            else
                return pos;
        }
    }
};

void solve() {
    int n, q; cin >> n >> q;
    SegTree se(n);
    for(int i = 1; i <= n; i ++) cin >> se.w[i];
    se.build(1, 1, n);
    while(q --)
    {
        int op; cin >> op;
        if(op == 1)
        {
            int x, d; cin >> x >> d;
            se.modify(1, x, d);
        }
        else
        {
            int l, r, d; cin >> l >> r >> d;
            cout << se.ask(1, l, r, d) << "\n";
        }
    }
}
```

## 线段树(单点修改)

```cpp
#define ls u << 1
#define rs u << 1 | 1
struct SegTree{

    struct Node{
        // to do
    };
    vector<int> w;
    vector<Node> tr;
    SegTree(int n) :tr(n * 4 + 1), w(n + 1){}

    void pushup(Node& u, Node& l, Node& r){
        // to do
    }
    void pushup(int u){
        pushup(tr[u], tr[ls], tr[rs]);
    }
    void build(int u, int l, int r){
        tr[u] = {l, r};
        if(l == r)
        {
            // to do
            return;
        }
        int mid = l + r >> 1;
        build(ls, l, mid);
        build(rs, mid + 1, r);
        pushup(u);
    }
    void modify(int u, int idx, int val){
        if(tr[u].l == tr[u].r && tr[u].l == idx)
        {
            // to do
            return;
        }
        int mid = tr[u].l + tr[u].r >> 1;
        if(idx <= mid) modify(ls, idx, val);
        else
            modify(rs, idx, val);
        pushup(u);
    }

    Node query(int u, int L, int R){
        if(tr[u].l >= L && tr[u].r <= R)
            return tr[u];
        int mid = tr[u].l + tr[u].r >> 1;
        if(R <= mid) return query(ls, L, R);
        else if (L > mid) return query(rs, L, R);
        else
        {
            // to do
        }
    }
};
```

## 线段树(区间修改)

```
#define int long long
#define ls u << 1
#define rs u << 1 | 1
struct SegTree{

    struct Node{
        // to do

    };

    vector<int> w;
    vector<Node> tr;
    SegTree(int n) :tr(n * 4 + 1), w(n + 1){}
    void pushdown(Node& u, Node& l, Node& r){
        // to do
        if(u.add)
        {
            eval(l, add);
            eval(r, add);
            u.add = 0;
        }
    }
    void pushup(Node& u, Node& l, Node& r){
        // to do

    }
    void pushup(int u){
        pushup(tr[u], tr[ls], tr[rs]);
    }
    void pushdown(int u){
        pushdown(tr[u], tr[ls], tr[rs]);
    }
    void build(int u, int l, int r){
        tr[u] = {l, r};
        if(l == r)
        {
            // to do

            return;
        }
        int mid = l + r >> 1;
        build(ls, l, mid);
        build(rs, mid + 1, r);
        pushup(u);
    }

    void eval(Node& t, int add, int mul){
        // to do
        // t.sum = (t.sum * mul + (t.r - t.l + 1) * add) % p;
        // t.mul = t.mul * mul % p;
        // t.add = (t.add * mul + add) % p;
    }

    void modify(int u, int idx, int val){ // 单点修改
        if(tr[u].l == tr[u].r && tr[u].l == idx)
        {
            // to do

            return;
        }
        int mid = tr[u].l + tr[u].r >> 1;
        if(idx <= mid) modify(ls, idx, val);
        else
            modify(rs, idx, val);
        pushup(u);
    }
    void modify(int u, int L, int R, int val) // 区间修改
    {
        if(tr[u].l >= L && tr[u].r <= R)
        {
            // to do
            // eval(tr[u], val);
            return;
        }
        pushdown(u);
        int mid = tr[u].l + tr[u].r >> 1;
        if(L <= mid) modify(ls, L, R, val);
        if(R > mid) modify(rs, L, R, val);
        pushup(u);

    }
    Node query(int u, int L, int R){
        if(tr[u].l >= L && tr[u].r <= R)
            return tr[u];
        pushdown(u);
        int mid = tr[u].l + tr[u].r >> 1;
        if(R <= mid) return query(ls, L, R);
        else if (L > mid) return query(rs, L, R);
        else
        {
            // to do

        }
    }

};
```

# 字符串

# 字典树

## 模板

```
    auto insert = [&](string& str) -> void{
        int p = 0;
        for(auto& ite : str)
        {
            int u = ite - 'a';
            if(!s[p][u]) s[p][u] = ++ idx;
            p = s[p][u];
        }
        cnt[p] ++ ;
    };
    auto query = [&](string& str) -> int{
        int p = 0;
        for(auto& ite : str)
        {
            int u = ite - 'a';
            if(!s[p][u]) return 0;
            p = s[p][u];
        }
        return cnt[p];
    };
```

## 最大异或对

```
int n; cin >> n;
idx = 1;
auto insert = [&](int x){
    int p = 0;
    for(int i = 30; ~i; i --)
    {
        int u = x >> i & 1;
        if(!s[p][u]) s[p][u] = idx ++ ;
        p = s[p][u];
    }
};
auto get_max = [&](int x) -> int{
    int p = 0, res = 0;
    for(int i = 30; ~i; i --)
    {
        int u = x >> i & 1;
        if(s[p][!u])
            res = res << 1 | !u, p = s[p][!u];
        else
            res = res << 1 | u, p = s[p][u];
    }
    return res ^ x;
};
int res = 0;
for(int i = 0; i < n; i ++)
{
    int x; cin >> x;
    insert(x);
    res = max(res, get_max(x));

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

## 能否成为字典序最小

给定$n$个字符串，互不相等，你可以任意指定字符之间的大小关系（即重定义字典序），求有多少个串可能成为字典序最小的串，并输出它们

思路: 字典树 + topsort

```
#include <bits/stdc++.h>

typedef long long LL;

using namespace std;
const int N = 300010;
int son[N][30];
int cnt[N], idx;

inline void insert(string & str){
    int p = 0;
    for(auto& ite : str)
    {
        int u = ite - 'a';
        if(!son[p][u]) son[p][u] = ++ idx;
        p = son[p][u];
    }
    cnt[p] ++ ;
}

bool check(string& str){
    int p = 0;
    vector<int> d(26);
    vector<vector<int>> g(26);
    for(auto& ite : str)
    {
        int u = ite - 'a';
        if(cnt[p]) return false; // 已经有某个字符串作为前缀了
        for(int i = 0; i < 26; i ++)
```

```
            if(i != u && son[p][i]) d[i] ++ , g[u].push_back(i); // 为topsort建图
        p = son[p][u];
    }
    queue<int> q;
    for(int i = 0; i < 26; i ++)
        if(!d[i]) q.push(i);
    while(q.size())
    {
        int t = q.front(); q.pop();
        for(auto& val : g[t])
            if(--d[val] == 0) q.push(val);
    }
    return count(d.begin(), d.end(), 0) == 26;
}

void solve() {
    int n; cin >> n;
    vector<string> v(n);
    for(int i = 0; i < n; i ++)
        cin >> v[i], insert(v[i]);
    vector<string> res;
    for(auto& ite : v)
        if(check(ite))
            res.push_back(ite);
    cout << res.size() << "\n";
    for(auto& ite : res)
        cout << ite << "\n";
}

int main() {
    std::ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int T = 1;
    // cin >> T;

    while (T--) {
        solve();
    }

    return 0;
}
```

## KMP

## KMP求匹配位置

```
struct kizk_kmp{
    string p, s;
    int n, m;
    vector<int> ne, idx;
    kizk_kmp(string& _p) : p(_p){this->n = _p.size();}
    kizk_kmp(string& _p, string& _s) : p(_p), s(_s){this->n = _p.size(),
this->m = _s.size();}
    kizk_kmp(){}
    void set_p(string& a){
        if(a == this->p) return;
        this->p = a, this->n = a.size();
        this->ne.clear();
    }
    void set_s(string& a){
        if(this->s == a) return;
        this->s = a, this->m = a.size();
        this->idx.clear();
    }
    vector<int> kmp_ne(){
        if(this->ne.size()) return this->ne;
        int n = this->n;
        string p = this->p;
        vector<int> ne(n + 1);
        for(int i = 1, j = 0; i < n; i ++)
        {
            while(j && p[i] != p[j]) j = ne[j];
            j += (p[i] == p[j]);
            ne[i + 1] = j;
        }
        return this->ne = ne;
    }
    vector<int> kmp_idx(){
        if(!this->ne.size()) this->kmp_ne();
        if(this->idx.size()) return this->idx;
        int n = this->n, m = this->m;
        string p = this->p, s = this->s;
        vector<int>v;
        for(int i = 0, j = 0; i < m; i ++)
        {
            while(j && s[i] != p[j]) j = ne[j];
            j += (s[i] == p[j]);
            if(j == n)
                v.push_back(i - j + 1), j = ne[j];
            // left idx (i - j + 1) from zero  right idx (i) idx from zero
        }
        return this->idx = v;
    }
    vector<int> kmp_idx(string& s){
        if(!this->ne.size()) this->kmp_ne();
        int n = this->n;
        int m = s.size();
        vector<int>v;
        for(int i = 0, j = 0; i < m; i ++)
        {
            while(j && s[i] != p[j]) j = ne[j];
            j += (s[i] == p[j]);
            if(j == n)
                v.push_back(i - j + 1), j = ne[j];
        }
        return v;
    }
    int kmp_cnt(){
        return this->kmp_idx().size();
    }
    int kmp_cnt(string& s){
        return this->kmp_idx(s).size();
    }

};
```

### KMP求最小循环节

```
std::string str; std::cin >> str; str = ' ' + str;
std::vector<int> ne(n + 1);
for(int i = 2, j = 0; i <= n; i ++)
{
    while(j && str[i] != str[j + 1]) j = ne[j];
    j += str[i] == str[j + 1];
    ne[i] = j;
}

for(int i = 2; i <= n; i ++)
{
    int temp = i - ne[i];
    if(i % temp == 0 && ne[i])
        std::cout << i << ' ' << i / temp << "\n";
}
```

## AC自动机

### 板子

给定$n$个长度不超过50的由小写英文字母组成的单词，以及一篇长为$m$的文章。

请问，其中有多少个单词在文章中出现了。

**注意：每个单词不论在文章中出现多少次，仅累计 $1$ 次。**

```
#include <bits/stdc++.h>
```

```cpp
using namespace std;
struct node
{
    int fail;
    int ch[26];
    int ans;
};
struct node NIL = {0};
struct trie
{
    vector<node> tr;
    vector<int> in;
    map<int, int> ref;
    int siz = 0;
    void insert(string s,int x)
    {
        if(!tr.size())
            tr.push_back(NIL);
        int place = 0;
        for (int i = 0; i < s.length();i++)
        {
            if (tr[place].ch[s[i] - 97] == 0)
            {
                tr[place].ch[s[i] - 97] = ++siz;
                tr.push_back(NIL);
                in.push_back(0);
            }
            place = tr[place].ch[s[i] - 97];
        }
        ref[x] = place;
    }
    void build()
    {
        queue<int> q;
        for (int i = 0; i < 26;i++)
            if(tr[0].ch[i])
                q.push(tr[0].ch[i]);
        while(!q.empty())
        {
            int tp = q.front();
            q.pop();
            for (int i = 0; i < 26;i++)
                if(tr[tp].ch[i])
                {
                    tr[tr[tp].ch[i]].fail = tr[tr[tp].fail].ch[i];
                    in[tr[tr[tp].fail].ch[i]]++;
                    q.push(tr[tp].ch[i]);
                }
                else
                    tr[tp].ch[i] = tr[tr[tp].fail].ch[i];
        }
    }
    void query(string t)
    {
        int place = 0;
        for (int i = 0; i < t.length();i++)
        {
            place = tr[place].ch[t[i] - 97];
            tr[place].ans++;
        }
    }
    void topo()
    {
        queue<int> q;
        for (int i = 1; i <= siz;i++)
            if(!in[i])
                q.push(i);
        while(!q.empty())
        {
            int tp = q.front();
            q.pop();
            tr[tr[tp].fail].ans += tr[tp].ans;
            in[tr[tp].fail]--;
            if(!in[tr[tp].fail])
                q.push(tr[tp].fail);
        }
    }
};
void solve(){
    int n;
    string b = "";
    cin >> n;
    vector<string> a(n + 1);
    for(int i = 1; i <= n; i ++)
        cin >> a[i];
    trie tr;
    for (int i = 1; i <= n;i++)
    {
        tr.insert(a[i], i);
    }
    tr.build();
    cin >> b;
    tr.query(b);
    tr.topo();
    int res = 0;
    for (int i = 1; i <= n;i++)
        res += tr.tr[tr.ref[i]].ans >= 1; // tr.tr[tr.ref[i]].ans即为每个单词
在文章中出现多少次
    cout << res << "\n";
}
int main()
{
    ios::sync_with_stdio(false);
    std::cin.tie(0);
    int T; cin >> T;
    while(T --) solve();
    return 0;
}
```

某人读论文，一篇论文是由许多单词组成的。

但他发现一个单词会在论文中出现很多次，现在他想知道每个单词分别在论文中出现多少次。

```
#include <bits/stdc++.h>

using namespace std;
struct node
{
    int fail;
    int ch[27]; // 开大一个
    int ans;
};
.... // 这边都是一样的
void solve(){
    int n;
    string b = "";
    cin >> n;
    vector<string> a(n + 1);
    for(int i = 1; i <= n; i ++)
        cin >> a[i];
    trie tr;
    for (int i = 1; i <= n;i++)
    {
        tr.insert(a[i], i);
        b += a[i] + char('z' + 1); // 分隔一下
    }
    tr.build();
    tr.query(b);
    tr.topo();
    int res = 0;
    for (int i = 1; i <= n;i++)
        cout << tr.tr[tr.ref[i]].ans << "\n";;
}
int main()
{
    ios::sync_with_stdio(false);
    std::cin.tie(0);
    int T = 1;
    while(T --) solve();
    return 0;
}
```

# border树

## 性质

1. 每个前缀$prefix[i]$的所有$Border$:节点$i$到根的链
2. 哪些前缀有长度为$x$的$Border$: $x$的子树
3. 求两个前缀的公共$Border$等价于求$LCA$

## 求至少出现K次

给一个长度为$n$的仅包含小写字母的字符串$S$，一个正整数$k$，求一个最长的字符串$T$，满足:
1. $T$为$S$的前缀
2. $T$为$S$的后缀
3. $T$在$S$中至少出现$k$次

```
int n, k; cin >> n >> k;
string str; cin >> str;
str = ' ' + str;
vector<int> ne(n + 1);
for(int i = 2, j = 0; i <= n; i ++)
{
    while(j && str[j + 1] != str[i]) j = ne[j];
    if(str[j + 1] == str[i]) j ++ ;
    ne[i] = j;
}
vector<int> cnt(n + 1);
for(int i = n; i >= 1; i --)
    cnt[ne[i]] += ++ cnt[i];
vector<int> ans;
ans.push_back(n);
int kk = ne[n];
while(kk)
{
    ans.push_back(kk);
    kk = ne[kk];
    if(!kk) break;
}
for(auto& ite : ans)
{
    int i = ite;
    if(cnt[i] >= k)
    {
        cout << str.substr(1, i) << "\n";
        return 0;
    }
}
cout << -1 << "\n";
```

# 字符串哈希

## 单哈希(ULL)

```
const int p = 13331;
typedef unsigned long long ULL;

void solve() {
    int n, m; cin >> n >> m;
    string str; cin >> str;
    str = " " + str;
    vector<ULL> c(n + 1), h(n + 1);
    c[0] = 1;
    for(int i = 1; i <= n; i ++)
    {
        h[i] = h[i - 1] * p + str[i];
        c[i] = c[i - 1] * p;
    }
    auto get = [&](int l, int r) -> int{
        return h[r] - h[l - 1] * c[r - l + 1];
    };
    for(int i = 0; i < m; i ++)
    {
        int l1, r1, l2, r2; cin >> l1 >> r1 >> l2 >> r2;
        cout << (get(l1, r1) == get(l2, r2) ? "Yes\n" : "No\n");
    }
}
```

## 单哈希(LL)

```
const int mod = 1e9 + 7, p = 13331;

void solve() {
    int n, m; cin >> n >> m;
    string str; cin >> str;
    str = " " + str;
    vector<LL> c(n + 1), h(n + 1);
    c[0] = 1;
    for(int i = 1; i <= n; i ++)
    {
        h[i] = (h[i - 1] * p % mod + str[i] - 'a' + 1) % mod;
        c[i] = c[i - 1] * p % mod;
    }
    auto get = [&](int l, int r) -> int{
        return (h[r] - h[l - 1] * c[r - l + 1] % mod + mod) % mod;
    };
    for(int i = 0; i < m; i ++)
    {
        int l1, r1, l2, r2; cin >> l1 >> r1 >> l2 >> r2;
        cout << (get(l1, r1) == get(l2, r2) ? "Yes\n" : "No\n");
    }
}
```

## 双哈希

```
const int p1 = 13331, p2 = 1333331;
typedef unsigned long long ULL;

void solve() {
    int n, m; cin >> n >> m;
    string str; cin >> str;
    str = " " + str;
    vector<ULL> c1(n + 1), c2(n + 1), h1(n + 1), h2(n + 1);
    c1[0] = c2[0] = 1;
    for(int i = 1; i <= n; i ++)
    {
        h1[i] = h1[i - 1] * p1 + str[i];
        c1[i] = c1[i - 1] * p1;
        h2[i] = h2[i - 1] * p2 + str[i];
        c2[i] = c2[i - 1] * p2;
    }
    auto get = [&](int l, int r) -> pair<ULL, ULL>{
        return {h1[r] - h1[l - 1] * c1[r - l + 1], h2[r] - h2[l - 1] * c2[r
- l + 1]};
    };
    for(int i = 0; i < m; i ++)
    {
        int l1, r1, l2, r2; cin >> l1 >> r1 >> l2 >> r2;
        cout << (get(l1, r1) == get(l2, r2) ? "Yes\n" : "No\n");
    }
}
```

## 线段树版本

```

#include <bits/stdc++.h>

typedef long long LL;

using namespace std;

#define ls u << 1
#define rs u << 1 | 1
typedef unsigned long long ULL;
struct SegTree{
    struct Node{
        int l, r;
        ULL val;
    };
    const int mod = 1e9 + 7, p = 13331;
    vector<int> w;
    vector<Node> tr;
    vector<ULL> base;
    string str;
    SegTree(int n, string str) :tr(n * 4 + 1), w(n + 1), base(n + 1)
    {
        this->str = str;
        base[0] = 1;
        for(int i = 1; i <= n; i ++)
        {
            base[i] = base[i - 1] * p;
        }
    }
    void pushup(Node& u, Node& l, Node& r){
        u.val = l.val * base[r.r - r.l + 1] + r.val;
    }
    void build(int u, int l, int r){
        tr[u] = {l, r};
        if(l == r)
        {
            tr[u] = {l, r, str[l] - 'A' + 1};
            return;
        }
        int mid = l + r >> 1;
        build(ls, l, mid);
        build(rs, mid + 1, r);
        pushup(tr[u], tr[ls], tr[rs]);
    }

    ULL query(int u, int L, int R){
        if(tr[u].l >= L && tr[u].r <= R)
            return tr[u].val;
        int mid = tr[u].l + tr[u].r >> 1;
        if(R <= mid) return query(ls, L, R);
        else if(L > mid) return query(rs, L, R) ;
        else
        {
             int len = R - mid;
             return query(ls, L, mid) * base[len] + query(rs, mid + 1, R);
        }

    }
};


void solve() {
    int n, q; cin >> n >> q;
    string str; cin >> str;
    str = ' ' + str;
    SegTree t(n, str);
    t.build(1, 1, n);
    while(q --)
    {
        int a, b, c, d; cin >> a >> b >> c >> d;
        if(t.query(1, a, b) == t.query(1, c, d))
            cout << "Yes\n";
        else
            cout << "No\n";
    }

}

int main() {
    std::ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int T = 1;
    // cin >> T;

    while (T--) {
        solve();
    }

    return 0;
}
```

## manacher

```
vector<int> manacher(string a) // 最长回文子串为*max_element(hw.begin(),
hw.end()) - 1;
{
    string b = "$|";
    for (auto i : a)
    {
        b += i;
        b += '|';
    }
    int len = b.length();
    vector<int> hw(b.length());
    int maxright = 1, mid = 1;
    for (int i = 1; i < len; i++)
    {
        if (i < maxright)
            hw[i] = min(hw[mid * 2 - i], hw[mid] + mid - i);
        else
            hw[i] = 1;
        while (b[i - hw[i]] == b[i + hw[i]])
            hw[i]++;
        if (i + hw[i] > maxright)
        {
            maxright = i + hw[i];
            mid = i;
        }
    }
    return hw;
}
```

## 最小表示法

```
auto minimalRepresentation = [&](string& s) -> string{
        int n = s.size();
        int i = 0, j = 1, k = 0;
        while(i < n && j < n && k < n)
        {
            char a = s[(i + k) % n], b = s[(j + k) % n];
            if(a == b) k ++ ;
            else
            {
                if(a > b) i += k + 1;
                else j += k + 1;
                if(i == j) i ++ ;
                k = 0;
            }
        }
        i = min(i, j);
        return s.substr(i) + s.substr(0, i);
    };
```

# 图论

## 树上问题

### 树的直径

```
// DFS
int n; cin >> n;
vector<vector<array<int, 2>>> g(n);
for(int i = 1, a, b, c; i < n; i ++)
cin >> a >> b >> c, g[--a].push_back({--b, c}), g[b].push_back({a, c});
int r = -1, maxv = 0;
auto dfs = [&](auto _, int u, int pre, int dist) -> void{
    if(dist > maxv) r = u, maxv = dist;
    for(auto& [i, w] : g[u])
```

```
        if(i ^ pre)
            _(_, i, u, dist + w);
};
dfs(dfs, 0, -1, 0);
dfs(dfs, r, -1, 0);

// DP 两个数组记录
int n; cin >> n;
vector<vector<array<int, 2>>> g(n + 1);
vector<int> d1(n + 1), d2(n + 1);
for(int i = 1, a, b, c; i < n; i ++)
cin >> a >> b >> c, g[a].push_back({b, c}), g[b].push_back({a, c});
int d = 0;
auto dfs = [&](auto _, int u, int pre) -> void{
    for(auto& [i, w] : g[u])
        if(i ^ pre)
        {
            _(_, i, u);
            int t = d1[i] + w;
            if(t > d1[u])
                d2[u] = d1[u], d1[u] = t;
            else if(t > d2[u])
                d2[u] = t;
        }
    d = max(d, d1[u] + d2[u]);
};
dfs(dfs, 1, -1);

// DP 一个数组记录
auto dp = [&](auto _, int u, int fa) -> void{
    for(auto& i : g[u])
        if(i != fa)
        {
            _(_, i, u);
            // w -> 边权
            d = max(d, d1[u] + d1[i] + w);
            d1[u] = max(d1[u], d1[i] + w);
        }
};
dp(dp, 0, -1);
```

# 简单算法

## bellman-ford

```
int n, m, k; cin >> n >> m >> k;
    vector<tuple<int, int, int>> v(m);
    for(int i = 0; i < m; i ++)
    {
        int a, b, c; cin >> a >> b >> c;
        v[i] = tuple<int, int, int>(a, b, c);
    }
    vector<int> d(n + 1, 1e9 + 10);
    d[1] = 0;
    while(k --)
    {
        vector<int> back = d;
        for(auto& ite : v)
        {
            int a = get<0>(ite), b = get<1>(ite), c = get<2>(ite);
            d[b] = min(d[b], back[a] + c);
        }
    }
    if(d[n] > 1e7) cout << "impossible\n";
    else
        cout << d[n] << "\n";
```

## 二分图(匈牙利)

```
// check
#include <bits/stdc++.h>

using namespace std;

int main(){
    ios::sync_with_stdio(false);
    std::cin.tie(0);
    int n, m; cin >> n >> m;
    vector<vector<int>> g(n + 1);
    for(int a, b; cin >> a >> b; ) g[a].push_back(b), g[b].push_back(a);
    vector<int> color(n + 1, 0);
    auto dfs = [&](auto dfs, int u, int c) -> bool{
        color[u] = c;
        for(auto& ite : g[u])
        {
            if(!color[ite]) if(!dfs(dfs, ite, 3 - c)) return false;
            if(color[ite] == c)
                return false;
        }
        return true;
    };
    for(int i = 1; i <= n; i ++)
        if(!color[i])
```

```cpp
        {
            if(!dfs(dfs, i, 1))
            {
                cout << "No\n";
                return 0;
            }
        }
    cout << "Yes\n";
}


// 最大匹配
#include <bits/stdc++.h>

using namespace std;

int main(){
    ios::sync_with_stdio(false);
    std::cin.tie(0);
    int n1, n2, m; cin >> n1 >> n2 >> m;
    vector<vector<int>> g(n1 + 1);
    for(int a, b; cin >> a >> b; ) g[a].push_back(b);
    int res = 0;
    vector<bool> st(n2 + 1, false);
    vector<int> ma(n2 + 1, 0);
    auto find = [&](auto find, int u) -> bool{
        for(auto& ite : g[u])
            if(!st[ite])
            {
                st[ite] = true;
                if(!ma[ite] || find(find, ma[ite]))
                {
                    ma[ite] = u;
                    return true;
                }
            }
        return false;
    };
    for(int i = 1; i <= n1; i ++)
    {
        st = vector<bool>(n2 + 1, false);
        if(find(find, i))
            res ++ ;
    }
    cout << res << "\n";
    return 0;
}
```

## dijkstra

```
auto dijkstra = [&](int root) -> int{
        vector<int> d(n + 1, 1e9 + 10);
        vector<bool> st(n + 1, false);
        d[1] = 0;
        for(int i = 1; i < n; i ++)
        {
            int t = -1;
            for(int j = 1; j <= n; j ++)
                if(!st[j] && (t == -1 || d[j] < d[t]))
                    t = j;
            for(int j = 1; j <= n; j ++)
                d[j] = min(d[j], d[t] + g[t][j]);
            st[t] = true;
        }
        return d[n] == 1e9 + 10 ? -1 : d[n];
    };


  auto dijkstra = [&](int root) -> int{
        vector<int> d(n + 1, 1e9 + 10);
        d[root] = 0;
        priority_queue<PII, vector<PII>, greater<PII> > q;
        q.push({0, root});
        while(q.size())
        {
            auto [dist, id] = q.top(); q.pop();
            if(d[id] != dist) continue;
            for(auto& [b, w] : g[id])
            {
                if(d[b] > dist + w)
                {
                    d[b] = dist + w;
                    q.push({d[b], b});
                }
            }
        }
        return d[n] == 1e9 + 10 ? -1 : d[n];
    };
```

## Floyd

```
 for(int i = 1; i <= n; i ++) g[i][i] = 0;
    for(int k = 1; k <= n; k ++)
        for(int i = 1; i <= n; i ++)
            for(int j = 1; j <= n; j ++)
                g[i][j] = min(g[i][j], g[i][k] + g[k][j]);
```

## kruskal

```
int n, m; cin >> n >> m;
    vector<array<int, 3>> v;
    for(int a, b, c; cin >> a >> b >> c; )v.push_back({a, b, c});
    vector<int> p(n + 1);
    iota(p.begin(), p.end(), 0);
    auto find = [&](auto find, int x) -> int{
        return p[x] ^ x ? p[x] = find(find, p[x]) : x;
    };
    sort(v.begin(), v.end(), [&](array<int, 3>& a, array<int, 3>& b){
        return a[2] < b[2];
    });
    int res = 0;
    int cnt = n;
    for(auto&[a, b, c] : v)
    {
        a = find(find, a), b = find(find, b);
        if(a ^ b)
        {
            p[a] = b;
            res += c;
            cnt -- ;
        }
    }
    if(cnt ^ 1)
        cout << "impossible\n";
    else
        cout << res << "\n";
```

## prime

```
auto prime = [&]() -> void{
        int res = 0;
        bool flg = true;
        vector<int> d(n + 1, 1e9 + 10);
        vector<bool> st(n + 1, false);
        d[1] = 0;
        for(int i = 0; i < n; i ++)
        {
            int t = -1;
            for(int j = 1; j <= n; j ++)
                if(!st[j] && (t == -1 or d[j] < d[t]))
                    t = j;
            if(i && d[t] == 1e9 + 10)
            {
                flg = false;
                break;
            }
            if(i)
                res += d[t];
            for(int j = 1; j <= n; j ++)
                d[j] = min(d[j], g[t][j]);
            st[t] = true;
        }
        if(flg)
            cout << res << '\n';
        else
            cout << "impossible\n";
    };
```

## spfa

```
// 判断负环
auto spfa = [&](int root) -> bool{
        memset(d, 0x3f, sizeof d);
        d[1] = 0;
        queue<int> q;
        for(int i = 1; i <= n; i ++)
            st[i] = true, q.push(i);
        while(q.size())
        {
            int t = q.front();
            q.pop();
            st[t] = false;
            for(auto& [j, w] : g[t])
            {
                if(d[j] > d[t] + w)
                {
                    d[j] = d[t] + w;
                    cnt[j] = cnt[t] + 1;
                    if(cnt[j] == n)
                        return true;
                    if(!st[j])
                    {
                        st[j] =  true;
                        q.push(j);
                    }
                }
            }
        }
        return false;
    };
    if(spfa(1))
        cout << "Yes\n";
    else
        cout << "No\n";

//
```

```
auto spfa = [&](int root) -> void{
        vector<bool> st(n + 1, false);
        vector<int> d(n + 1, 1e9 + 10);
        d[root] = 0;
        queue<int> q;
        q.push(root);
        while(q.size())
        {
            int t = q.front(); q.pop();
            st[t] = false;
            for(auto& [b, w] : g[t])
            {
                if(d[b] > d[t] + w)
                {
                    d[b] = d[t] + w;
                    if(!st[b])
                        st[b] = true, q.push(b);
                }

            }
        }
        if(d[n] > 1e7 + 10) cout << "impossible\n";
        else
            cout << d[n] << "\n";
    };
```

## LCA

```
#include <bits/stdc++.h>

using namespace std;

int main(){
    int n, k; cin >> n >> k;
    int m = __lg(n << 2);
    vector<int> d(n + 1, -1);
    vector<int> dist(n + 1, 1e9);
    vector<vector<array<int, 2>>> g(n + 1);
    vector<vector<int>> fa(n + 1, vector<int>(m + 1));
    for(int i = 0; i < n - 1; i ++)
    {
        int a, b, c; cin >> a >> b >> c;
        g[a].push_back({b, c});
        g[b].push_back({a, c});
    }
    queue<int> q;
    q.push(1);
    d[0] = 0, d[1] = 1;
    dist[1] = 0;
    while(q.size())
    {
        int t = q.front(); q.pop();
        for(auto& ite : g[t])
        {
            int id = ite[0], w = ite[1];
            if(d[id] == -1)
            {
                d[id] = d[t] + 1;
                dist[id] = dist[t] + w;
                q.push(id);
                fa[id][0] = t;
                for(int i = 1; i <= m; i ++)
                    fa[id][i] = fa[fa[id][i - 1]][i - 1];
            }
        }
    }
    auto get_lca = [&](int x, int y) -> int{
        if(d[x] < d[y]) swap(x, y);
        for(int i = m; i >= 0; i --)
            if(d[fa[x][i]] >= d[y])
                x = fa[x][i];
        if(x == y)
            return x;
        for(int i = m; i >= 0; i --)
            if(fa[x][i] != fa[y][i])
                x = fa[x][i], y = fa[y][i];
        return fa[x][0];
    };
    while(k --)
    {
        int a, b; cin >> a >> b;
        int c = get_lca(a, b);
        cout << dist[a] + dist[b] - 2 * dist[c] << "\n";
    }
}
```

# 动态规划

## 数位DP(py)

如果一个正整数每一个数位都是 **互不相同** 的，我们称它是 **特殊整数** 。

给你一个 **正** 整数 $n$ ，请你返回区间 $[1, n]$之间特殊整数的数目。

```
class Solution:
    def countSpecialNumbers(self, n: int) -> int:
        s = str(n)
        @lru_cache(None)
        def dfs(i: int, mask: int, is_limit: bool, is_number: bool) -> int:
            if i == len(s): # 到达最后一个数 判断一下之前是否填了数
                return int(is_number)
            res = 0
            if not is_number: # 如果上一个数没填 那我们该数也已不填
                res = dfs(i + 1,mask, False, False)
            up = int(s[i]) if is_limit else 9 # 上界
            for d in range(1 - int(is_number), up + 1):
                if mask >> d & 1 == 0:
                    res += dfs(i + 1, mask|(1 << d), is_limit and d == up,
True)
            return res
        return dfs(0, 0, True, False)
```

# 数学

## 数论

### 筛质数 && 求质因子 && 快速幂

```
LL mul(LL a, LL b, LL m) {
    return static_cast<__int128>(a) * b % m;
}
LL power(LL a, LL b, LL m) {
    LL res = 1 % m;
    for (; b; b >>= 1, a = mul(a, a, m))
        if (b & 1)
            res = mul(res, a, m);
    return res;
}
bool isprime(LL n) {
    if (n < 2)
        return false;
    static constexpr int A[] = {2, 3, 5, 7, 11, 13, 17, 19, 23};
    int s = __builtin_ctzll(n - 1);
    LL d = (n - 1) >> s;
    for (auto a : A) {
        if (a == n)
            return true;
        LL x = power(a, d, n);
        if (x == 1 || x == n - 1)
            continue;
        bool ok = false;
        for (int i = 0; i < s - 1; ++i) {
```

```cpp
            x = mul(x, x, n);
            if (x == n - 1) {
                ok = true;
                break;
            }
        }
        if (!ok)
            return false;
    }
    return true;
}
std::vector<LL> factorize(LL n) {
    std::vector<LL> p;
    std::function<void(LL)> f = [&](LL n) {
        if (n <= 10000) {
            for (int i = 2; i * i <= n; ++i)
                for (; n % i == 0; n /= i)
                    p.push_back(i);
            if (n > 1)
                p.push_back(n);
            return;
        }
        if (isprime(n)) {
            p.push_back(n);
            return;
        }
        auto g = [&](LL x) {
            return (mul(x, x, n) + 1) % n;
        };
        LL x0 = 2;
        while (true) {
            LL x = x0;
            LL y = x0;
            LL d = 1;
            LL power = 1, lam = 0;
            LL v = 1;
            while (d == 1) {
                y = g(y);
                ++lam;
                v = mul(v, std::abs(x - y), n);
                if (lam % 127 == 0) {
                    d = std::gcd(v, n);
                    v = 1;
                }
                if (power == lam) {
                    x = y;
                    power *= 2;
                    lam = 0;
                    d = std::gcd(v, n);
                    v = 1;
                }
            }
            if (d != n) {
                f(d);
                f(n / d);
                return;
            }
            ++x0;
        }
    };
    f(n);
    std::sort(p.begin(), p.end());
    return p;
}
```

## EXGCD

```
LL exgcd(LL a, LL b, LL& x, LL& y) {
    if(!b)
        return (x = 1, y = 0, a);
    LL d = exgcd(b, a % b, y, x);
    return (y -= x * (a / b), d);
}
```

## EXCTR

```
// find x
void solve() {
    int n; cin >> n;
    int a = 0, b = 1; // x mod b = a
    auto merge = [&](int& a, int& b, int c, int d) -> void{
        if(a == -1 || b == -1)
            return;
        LL x, y;
        LL g = exgcd(b, d, x, y);
        if((c - a) % g != 0)
        {
            a = b = -1;
            return;
        }
        d /= g; // d'
        LL t = (c - a) / g % d;
        LL t0 = x * t % d;
        if(t0 < 0) t0 += d;
        a = a + b * t0;
        b = b * d;
    };
    for(int i = 0; i < n; i ++)
    {
        int c, d; cin >> d >> c;
        merge(a, b, c, d);
        if(a == -1 && b == -1)
        {
            cout << "-1\n";
            return;
        }
    }
    cout << a << "\n";
}
// check x is exist
bool solve() {
    int n; cin >> n;
    map<int, vector<pair<int, int>>> mp;
    for(int i = 0; i < n; i ++)
    {
        int x, a; cin >> x >> a;
        for(int j = 2; j <= x / j; j ++)
        {
            if(x % j == 0)
            {
                int p = j, q = j;
                while(x % j == 0) x /= j, q *= j;

                mp[p].push_back({q, a % q});
            }
        }
        if(x > 1) mp[x].push_back({x, a % x});
    }
    for(auto& ite : mp)
    {
        auto v = ite.second;
        int t = max_element(v.begin(), v.end()) -> second;
        for(auto& val : v)
            if(t % val.first != val.second) return false;
    }
    return true;
}
```

## 求逆元

```
// 快速幂
LL qmi = [&](LL a, LL b, LL p) -> LL{
    LL t = 1;
    while(b){
        if(b & 1) t = t * a % p;
        a = a * a % p;
        b >>= 1;
    }
    return t;
};
```

```
// exgcd
LL exgcd(LL a, LL b, LL&x, LL& y){
    if(!b)
        return (x = 1, y = 0, a);
    LL d = exgcd(b, a % b, y, x);
    return (y -= x * (a / b), d);
}
void solve(){
    LL a, b; cin >> a >> b;
    LL x, y;
    LL d = exgcd(a, b, x, y);
    if(d != 1)
        cout << "impossible\n";
    else
        cout << (x % b + b) % b << "\n";
}
// 递推
vector<int> inv(n + 1);
inv[1] = 1;
int p;
for(int i = 2; i <= n; i ++) inv[i] = (p - p / i) * inv[p % i] % p;
```

## 欧拉函数

欧拉函数是积性函数: $\phi(ab) = \phi(a)\phi(b)$

$if \rightarrow n\%2 == 1 : \phi(2n) = \phi(n)$

$n = \sum_{d|n} \phi(d)$

```
// 1 ~ n 欧拉函数的和
void get(int n){
    phi[1] = 1;
    for(int i = 2; i <= n; i ++)
    {
        if(!st[i])
        {
            primes[cnt ++ ] = i;
            phi[i] = i - 1;
        }
        for(int j = 0; primes[j] <= n / i; j ++)
        {
            st[primes[j] * i] = true;
            if(i % primes[j] == 0)
            {
                phi[primes[j] * i] = phi[i] * primes[j];
                break;
            }
            phi[primes[j] * i] = phi[i] * (primes[j] - 1);
        }
    }
}
// phi(x)
int x; cin >> x;
        int res = x;
        for(int i = 2; i <= x / i; i ++)
            if(x % i == 0)
            {
                res = res / i * (i - 1);
                while(x % i == 0)
                    x /= i;
            }
        if(x > 1)
            res = res / x * (x - 1);
        cout << res << endl;
```

## 线性同余方程

```
// ax = b(mod m)
// ax - my = b;
LL exgcd(LL a, LL b, LL&x, LL& y){
    if(!b)
    {
        x = 1, y = 0;
        return a;
    }
    LL d = exgcd(b, a % b, y, x);
    y -= x * (a / b);
    return d;
}

void solve() {
    LL a, b, m; cin >> a >> b >> m;
    LL x, y;
    LL d = exgcd(a, m, x, y);
    if(b % d != 0)
    {
        cout << "impossible\n";
        return;
    }
    LL t = b / d;
    cout << x * t % m << "\n";
}
```

## 数论分块

计算 $\sum_{i=1}^{n} i \bmod k = n * k - \sum_{i=1}^{n} \lfloor \frac{k}{i} \rfloor * i$

对于每一个 $i$的 的右边界$j$为 $\left\lfloor \frac{k}{\frac{k}{i}} \right\rfloor$

```
void solve() {

    int n, k; cin >> n >> k;
    LL res = n * k;
    LL ans = 0;
    for(int l = 1, r; l <= n; l = r + 1)
    {
        r = k / l ? min(n, k / (k / l)) : n;
        ans += (r - l + 1) * (l + r) * (k / l) / 2;
    }
    cout << res - ans << "\n";
}
```

## 组合数

```
LL fpow(LL x,LL r)
{
   LL result = 1;
    while (r)
    {
        if (r & 1)result = result * x % mod;
        r >>= 1;
        x = x * x % mod;
    }
    return result;
}
namespace binom {
   LL fac[N], ifac[N];
    int __ = []
    {
        fac[0] = 1;
        for (int i = 1; i <= N - 5; i++)
            fac[i] = fac[i - 1] * i % mod;
        ifac[N - 5] = fpow(fac[N - 5], mod - 2);
        for (int i = N - 5; i; i--)
            ifac[i - 1] = ifac[i] * i % mod;
        return 0;
    }();

    inline LL C(int n, int m)
    {
        if (n < m || m < 0)return 0;
        return fac[n] * ifac[m] % mod * ifac[n - m] % mod;
    }

    inline LL A(int n, int m)
    {
        if (n < m || m < 0)return 0;
        return fac[n] * ifac[n - m] % mod;
    }
}
using namespace binom;
```

## 类欧几里德算法

计算形如 $f(a,b,c,n)=\sum_{i=0}^{n}\lfloor\frac{ai+b}{c}\rfloor$

```
#include <bits/stdc++.h>
#define int long long
using namespace std;
const int P = 998244353;
int i2 = 499122177, i6 = 166374059;

struct data {
  data() { f = g = h = 0; }

  int f, g, h;
};  // 三个函数打包

data calc(int n, int a, int b, int c) {
  int ac = a / c, bc = b / c, m = (a * n + b) / c, n1 = n + 1, n21 = n * 2
+ 1;
  data d;
  if (a == 0) {  // 迭代到最底层
    d.f = bc * n1 % P;
    d.g = bc * n % P * n1 % P * i2 % P;
    d.h = bc * bc % P * n1 % P;
    return d;
  }
  if (a >= c || b >= c) {  // 取模
    d.f = n * n1 % P * i2 % P * ac % P + bc * n1 % P;
    d.g = ac * n % P * n1 % P * n21 % P * i6 % P + bc * n % P * n1 % P * i2
% P;
    d.h = ac * ac % P * n % P * n1 % P * n21 % P * i6 % P +
          bc * bc % P * n1 % P + ac * bc % P * n % P * n1 % P;
    d.f %= P, d.g %= P, d.h %= P;

    data e = calc(n, a % c, b % c, c);  // 迭代

    d.h += e.h + 2 * bc % P * e.f % P + 2 * ac % P * e.g % P;
    d.g += e.g, d.f += e.f;
    d.f %= P, d.g %= P, d.h %= P;
    return d;
  }
  data e = calc(m - 1, c, c - b - 1, a);
  d.f = n * m % P - e.f, d.f = (d.f % P + P) % P;
  d.g = m * n % P * n1 % P - e.h - e.f, d.g = (d.g * i2 % P + P) % P;
  d.h = n * m % P * (m + 1) % P - 2 * e.g - 2 * e.f - d.f;
  d.h = (d.h % P + P) % P;
  return d;
}

int T, n, a, b, c;

signed main() {
  scanf("%lld", &T);
  while (T--) {
    scanf("%lld%lld%lld%lld", &n, &a, &b, &c);
    data ans = calc(n, a, b, c);
    printf("%lld %lld %lld\n", ans.f, ans.h, ans.g);
  }
  return 0;
}
```

## 欧拉定理 & 费马小定理

**费马小定理** 若$p$为素数, $gcd(a,p)=1$, 则 $a^{p-1}\equiv 1(mod\, p)$
**欧拉定理** 若 $gcd(a,m)=1$, 则 $a^{\varphi(m)}\equiv 1(mod\, m)$
**扩展欧拉定理**
https://oi-wiki.org/math/number-theory/fermat/

## 中国剩余定理

$$\begin{cases} x\equiv a_1(\text{mod} \quad m_1) \\ x\equiv a_2(\text{mod} \quad m_2) \\ \vdots \\ x\equiv a_n(\text{mod}) \quad m_n) \end{cases}$$

对于
1)$x=t1\cdot m_1+a_1$
2)$x=t2\cdot m_2+a_2$
联立得
3)$t_1\cdot m_1+a_1=t_2\cdot m_2+a_2$
移项
4)$t_1\cdot m_1+t_2\cdot(-m_2)=a_2-a_1$
至此可以直接 exgcd求$t_1,t_2$

---

若要找最小的$x\rightarrow$ 要找到最小的 $t_1,t_2$使得等式成立(要求$x$最小)

```
// find min x
void solve() {
    int n; cin >> n;
    int a = 0, b = 1; // x mod b = a
    auto merge = [&](int& a, int& b, int c, int d) -> void{
        if(a == -1 || b == -1)
            return;
        LL x, y;
        LL g = exgcd(b, d, x, y);
        if((c - a) % g != 0)
        {
            a = b = -1;
            return;
        }
        d /= g; // d'
        LL t = (c - a) / g % d;
        LL t0 = x * t % d; // ka = ka'(mod m) -> a = a'(mod m / k) 其中 k |
m
        if(t0 < 0) t0 += d;
        a = a + b * t0;
        b = b * d; // lcm
    };
    for(int i = 0; i < n; i ++)
    {
        int c, d; cin >> d >> c;
        merge(a, b, c, d);
        if(a == -1 && b == -1)
        {
            cout << "-1\n";
            return;
        }
    }
    cout << a << "\n";
}
// check x is exist
bool solve() {
    int n; cin >> n;
    map<int, vector<pair<int, int>>> mp;
    for(int i = 0; i < n; i ++)
    {
        int x, a; cin >> x >> a;
        for(int j = 2; j <= x / j; j ++)
        {
            if(x % j == 0)
            {
                int p = j, q = j;
                while(x % j == 0) x /= j, q *= j;
                mp[p].push_back({q, a % q});
            }
        }
        if(x > 1) mp[x].push_back({x, a % x});
    }
    for(auto& ite : mp)
    {
        auto v = ite.second;
        int t = max_element(v.begin(), v.end()) -> second;
        for(auto& val : v)
            if(t % val.first != val.second) return false;
    }
    return true;
}
```

## 威尔逊定理

定理: 对于素数 $p$ 有 $(p-1)! \equiv -1 \, (mod \, p)$

计算 $(n!)_p \rightarrow n! \, mod \, p$

```
int factmod(int n, int p) {
  vector<int> f(p);
  f[0] = 1;
  for (int i = 1; i < p; i++) f[i] = f[i - 1] * i % p;
  int res = 1;
  while (n > 1) {
    if ((n / p) % 2) res = p - res;
    res = res * f[n % p] % p;
    n /= p;
  }
  return res;
}
```

### 阶乘中素数的个数:

$O(\log_p n)$时间内计算出 $n!$含有素数$p$的幂次为:

$$v_p(n!) = \sum_{i=1}^{\infty} \lfloor \frac{n}{p^i} \rfloor$$

```
int multiplicity_factorial(int n, int p) {
  int count = 0;
  do {
    n /= p;
    count += n;
  } while (n);
  return count;
}
```

另一种比较常用的是: $v_p(n!) = \frac{n-S_p(n)}{p-1}$ 其中 $S_p(n)$ 是 $n$的$p$进制表示下的各位数的和

特别的2的幂次是: $v_2(n!) = n - S_2(n)$

```
int cal_s(int n, int p){
    int t = n;
    int res = 0;
    while(t)
    {
        res += t % p;
        t /= p;
    }
    return (n - res) / (p - 1);
}
```

### Kummer定理

组合数对一个数取模的结果，往往构成分形结构,例如谢尔宾斯基三角形就可以通过组合数模 2 得到。

$$v_p\left[\binom{m}{n}\right]=\frac{S_p(n)+S_p(m-n)-S_p(m)}{p-1}$$

特别的当 $p=2$ 时

$$v_2\left[\binom{m}{n}\right]=S_2(n)+S_2(m-n)-S_p(m)$$

### 威尔逊定理的推广

对于素数 $p$ 和正整数 $q$ 有 $(p^q!)_p=\pm1(mod\ p^q)$

$$(p^q!)_p=\begin{cases}1 & if\ p=2\ and\ q\geq 3\\ -1 & otherwise.\end{cases}$$

### 例题:

给定$n$, 计算 $\sum\limits_{k=1}^{n}\lfloor\frac{(3k+6)!+1}{3k+7}-\lfloor\frac{(3k+6)!}{3k+7}\rfloor\rfloor$

只需要看$(3k+7)$是不是质数即可

## 升幂定理

### 模为奇素数

$v_p(a^n-b^n)=v_p(a-b)+v_p(n)$

### 模为2

- 如果$n$为奇数, 定理为等式
  $v_2(a^n-b^n)=v_2(a-b)$
- 如过$n$为偶数, 定理为等式
  $v_2(a^n-b^n)=v_2(a-b)+v_2(a+b)+v_2(n)-1$

## Lucas

### Lucas($p$为素数)

Lucas 定理用于求解大组合数取模的问题，其中模数必须为素数。
定义: 对于质数$p$:

$$\binom{n}{m}\ mod\ p = \binom{\lfloor \frac{n}{p} \rfloor}{\lfloor \frac{m}{p} \rfloor} \cdot \binom{n\ mod\ p}{m\ mod\ p}\ mod\ p$$

时间复杂度: $f(p) + g(n)log(n)$, $f(n)$为预处理组合数的时间, $g(n)$为单次处理组合数的时间

```
long long Lucas(long long n, long long m, long long p) {
  if (m == 0) return 1;
  return (C(n % p, m % p, p) * Lucas(n / p, m / p, p)) % p;
}
```

### exLucas($p$为任意)

时间复杂度: $O(p + T\log p)$

```
LL calc(LL n, LL x, LL P) {
  if (!n) return 1;
  LL s = 1;
  for (LL i = 1; i <= P; i++)
    if (i % x) s = s * i % P;
  s = Pow(s, n / P, P);
  for (LL i = n / P * P + 1; i <= n; i++)
    if (i % x) s = i % P * s % P;
  return s * calc(n / x, x, P) % P;
}

LL multilucas(LL m, LL n, LL x, LL P) {
  int cnt = 0;
  for (LL i = m; i; i /= x) cnt += i / x;
  for (LL i = n; i; i /= x) cnt -= i / x;
  for (LL i = m - n; i; i /= x) cnt -= i / x;
  return Pow(x, cnt, P) % P * calc(m, x, P) % P * inverse(calc(n, x, P), P)
%
           P * inverse(calc(m - n, x, P), P) % P;
}

LL exlucas(LL m, LL n, LL P) {
  int cnt = 0;
  LL p[20], a[20];
  for (LL i = 2; i * i <= P; i++) {
    if (P % i == 0) {
      p[++cnt] = 1;
      while (P % i == 0) p[cnt] = p[cnt] * i, P /= i;
      a[cnt] = multilucas(m, n, i, p[cnt]);
    }
  }
  if (P > 1) p[++cnt] = P, a[cnt] = multilucas(m, n, P, P);
  return CRT(cnt, a, p);
}
```

## 组合数学

### 容斥

$$\left|\bigcup_{i=1}^{n} S_i\right| = \sum_{m=1}^{n}(-1)^{m-1}\sum_{a_i<a_{i+1}}\left|\bigcap_{i=1}^{m} S_{a_i}\right|$$

## 杂项

### 异或相关

# 贪心

## 区间合并

给定 $n$ 个区间 $[l_i, r_i]$，要求合并所有有交集的区间。

注意如果在端点处相交，也算有交集。

输出合并完成后的区间个数。

```
int n; cin >> n;
vector<pair<int, int>> v(n);
for(auto&[l, r] : v) cin >> l >> r;
sort(v.begin(), v.end());
int cnt = 0;
int last = -2e9 + 10;
for(auto&[l, r] : v)
{
    if(l > last) last = r, cnt ++ ;
    last = max(last, r);
}
cout << cnt << "\n";
```

# 计算几何

```

typedef double db;
const db eps = 1e-9;
inline int sign(db a) {
    return a < -eps ? -1 : a > eps;
}
```

```
inline int cmp(db a, db b){
    return sign(a - b);
}
struct P{
    db x, y;
    P(){}
    P(db _x, db _y) : x(_x), y(_y){}
    P operator+ (P p) { return {x + p.x, y + p.y}; }
    P operator- (P p) { return {x - p.x, y - p.y}; }
    P operator* (db d) { return {x * d, y * d}; }
    P operator/ (db d) { return {x / d, y / d}; }

    bool operator< (P p) const{
        int c = cmp(x, p.x);
        if(c) return c == -1;
        return cmp(y, p.y) == -1;
    }
    bool operator ==(P o) const{
        return cmp(x, o.x) == 0 && cmp(y, o.y) == 0;
    }
    db dot(P p) { return x * p.x + y * p.y; } // 求点积
    db det(P p) { return x * p.y - y * p.x; } // 求叉积
    db distTo(P p){ return (*this - p).abs(); } // 求两点之间的距离
    double alpha() { return atan2(y, x); } // 求角度
    long double alphal() {return atan2l(y, x); }
    void read() { cin >> x >> y; }
    void wirte() { cout << "(" << x << "," << y << ")" << "\n"; }
    db abs() { return sqrt(abs2()); }
    db abs2() { return x * x + y * y; }
    P rot90() { return P(-y, x); }
    P unit() { return *this/abs(); }
    int quad() const {} // to do
    P rot(db an) { return {x * cos(an) - y * sin(an), x * sin(an) + y *
cos(an)}; }

};
#define cross(p1, p2, p3) ((p2.x - p1.x) * (p3.y - p1.y) - (p3.x - p1.x) *
(p2.y - p1.y))
// p1p1 x p1p3
#define crossOp(p1, p2, p3) sign(cross(p1, p2, p3))

// 直线p1p2 q1q2 是否恰有一个交点
bool chkLL (P p1, P p2, P q1, P q2){ // 判断两条直线是否是相交关系
    db a1 = cross(q1, q2, p1), a2 = -cross(q1, q2, p2);
    return sign(a1 + a2) != 0;
}

P isLL(P p1, P p2, P q1, P q2){
    db a1 = cross(q1, q2, p1), a2 = -cross(q1, q2, p2);
    return (p1 * a2 + p2 * a1) / (a1 + a2);
}
```

```
// 判断区间 [l1, r1], [l2, r2] 是否相交
bool intersect(db l1, db r1, db l2, db r2){
    if (l1 > r1) swap(l1, r1); if(l2 > r2) swap(l2, r2);
    return !(cmp(r1, l2) == -1 || cmp(r2, l1) == -1);
}
// 线段相交
bool isSS(P p1, P p2, P q1, P q2) {
    return intersect(p1.x, p2.x, q1.x, q2.x) && intersect(p1.y, p2.y, q1.y,
q2.y)
    && crossOp(p1, p2, q1) * crossOp(p1, p2, q2) <= 0 && crossOp(q1, q2,
p1) * crossOp(q1, q2, p2) <= 0;

}
// 线段严格相交
bool isSS_strict(P p1, P p2, P q1, P q2){
    return crossOp(p1, p2, q1) * crossOp(p1, p2, q2) < 0 && crossOp(q1, q2,
p1) * crossOp(q1, q2, p2) < 0;
}
// 看m在不在[a,b]之间
bool isMiddle(db a, db m, db b){
    return sign(a - m) == 0 || sign(b - m) == 0 || (a < m != b < m);
}
// 看点m在不在 点 a, b之间
bool isMiddle(P a, P m, P b){
    return isMiddle(a.x, m.x, b.x) && isMiddle(a.y, m.y, b.y);
}
// 点q是否在线段 p1p2 上
bool onSeg(P p1, P p2, P q){
    return crossOp(p1, p2, q) == 0 && isMiddle(p1, q, p2);
}
bool onSeg_strict(P p1, P p2, P q){
    return crossOp(p1, p2, q) == 0 && sign((q-p1).dot(p1-p2)) * sign((q -
p2).dot(p1 - p2)) < 0;
}
// 求 q 到 直线p1p2的投影
P proj( P p1, P p2, P q){
    P dir = p2 - p1;
    return p1 + dir * (dir.dot(q - p1) / dir.abs2());
}
// 求 以 直线 p1p2为轴的反射
P reflect(P p1, P p2, P q){
    return proj(p1, p2, q) * 2 - q;
}
// 求 q 到线段 p1p2的最小距离
db nearest(P p1, P p2, P q){
    if(p1 == p2) return p1.distTo(q);
    P h = proj(p1, p2, q);
    if(isMiddle(p1, h, p2))
        return q.distTo(h);
    return min(p1.distTo(q), p2.distTo(q));
}
```

```
// 求线段 p1p2 q1q2 的距离
db disSS(P p1, P p2, P q1, P q2){
    if(isSS(p1, p2, q1, q2)) return 0;
    return min(min(nearest(p1, p2, q1), nearest(p1, p2, q2)),
min(nearest(q1, q2, p1), nearest(q1, q2, p2)));
}
```

# 常用函数

## python 头文件

```python
import random
import sys
import os
import math
from collections import Counter, defaultdict, deque
from functools import lru_cache, reduce
from itertools import accumulate, combinations, permutations
from heapq import nsmallest, nlargest, heapify, heappop, heappush
from io import BytesIO, IOBase
from copy import deepcopy
import operator
import threading
import bisect
BUFSIZE = 4096
Inf = float('inf')
eps = 1e-6
class FastIO(IOBase):
    newlines = 0

    def __init__(self, file):
        self._fd = file.fileno()
        self.buffer = BytesIO()
        self.writable = "x" in file.mode or "r" not in file.mode
        self.write = self.buffer.write if self.writable else None

    def read(self):
        while True:
            b = os.read(self._fd, max(os.fstat(self._fd).st_size, BUFSIZE))
            if not b:
                break
            ptr = self.buffer.tell()
            self.buffer.seek(0, 2), self.buffer.write(b),
self.buffer.seek(ptr)
        self.newlines = 0
        return self.buffer.read()

    def readline(self):
        while self.newlines == 0:
            b = os.read(self._fd, max(os.fstat(self._fd).st_size, BUFSIZE))
            self.newlines = b.count(b"\n") + (not b)
            ptr = self.buffer.tell()
            self.buffer.seek(0, 2), self.buffer.write(b),
self.buffer.seek(ptr)
        self.newlines -= 1
        return self.buffer.readline()

    def flush(self):
        if self.writable:
            os.write(self._fd, self.buffer.getvalue())
            self.buffer.truncate(0), self.buffer.seek(0)

class IOWrapper(IOBase):
    def __init__(self, file):
        self.buffer = FastIO(file)
        self.flush = self.buffer.flush
        self.writable = self.buffer.writable
        self.write = lambda s: self.buffer.write(s.encode("ascii"))
        self.read = lambda: self.buffer.read().decode("ascii")
        self.readline = lambda: self.buffer.readline().decode("ascii")

sys.stdin = IOWrapper(sys.stdin)
sys.stdout = IOWrapper(sys.stdout)
input = lambda: sys.stdin.readline().rstrip("\r\n")
def I():
   return input()
def II():
   return int(input())
def MI():
   return map(int, input().split())
def LI():
   return list(input().split())
def LII():
   return list(map(int, input().split()))
def GMI():
   return map(lambda x: int(x) - 1, input().split())
def LGMI():
   return list(map(lambda x: int(x) - 1, input().split()))
```

## bitset

```
#include <bits/stdc++.h>

using namespace std;

int main(){
    bitset<100010> s;
    int pos = 1;
    s.reset(); // set all 0
    s.reset(pos); // set pos to 0
    s.set(); // set all 1
    s.set(pos); // set pot to 1
    s.set(pos, false); // set pow false/true
    s.count(); // return count 1
    s.size(); // return size
    s.any(); // if 1 in s return true
    s.none(); // if all 0 return true
    s.all(); // if all 1 return true
    s.flip(); // ^ bitset<N> s; s.set();
    s.flip(pos); // filp pos
    s.to_string(); // to_string
    s.to_ullong(); // return unsigned long long
    s.to_ulong(); // return unsigned long
    s._Find_first(); // return first idx 1 if 1 in s else return size of s
    s._Find_next(); // return >= pos first idx 1 if 1 is s[pos:] else
return size of s
    return 0;
}
```

## algorithm

```
```