The ICPC Danang Regional 2019
Team Notebook – PTIT.Apo11o

Mục lục

Last edited: December 2, 2019

0.1.Team template

```
#pragma GCC optimize("Ofast")

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

#define y0 Apo11o
#define y1 Apo11o
#define yn Apo11o
#define j1 Apo11o

#define endl '\n'
#define i64 long long
#define ld long double
const long long Mod = 1000000007LL, INF = 1e9, LINF = 1e18;
const long double Pi = 3.141592653589793116L;
const long double EPS = 0.000000001L, Gold = ((1.0L+sqrt(5.0L))/2.0L);
mt19937 rng32(chrono::steady_clock::now().time_since_epoch().count());
mt19937_64 rng64(chrono::steady_clock::now().time_since_epoch().count());

int MultiTest = 0;



void Input() {

}

void Solve() {

}

int main(int argc, char* argv[]) {
      ios_base::sync_with_stdio(false); cin.tie(NULL);
      int T = 1; if (MultiTest) {cin >> T; cin.ignore();}
      while(T--) {Input(); Solve();}
      return 0;
}
```

0.2.Debug

```
template <class T1, class T2>
std::ostream &operator<<(ostream &os, const pair<T1, T2> &a) {
    return os << '(' << a.first << ", " << a.second << ')';
}
```

```
template <class T>
std::ostream &operator<<(ostream &os, const vector<T> &a) {
    os << '[';
    for (unsigned int i = 0; i < a.size(); i++)
        os << a[i] << (i < a.size() - 1? ", " : "");
    os << ']';
    return os;
}

template <class T>
std::ostream &operator<<(ostream &os, const set<T> &a) {
    os << '{';
    for(typename set<T>::iterator it = a.begin(); it != a.end(); it++) {
        typename set<T>::iterator jt = it;
        os << *it << (++jt != a.end()? ", " : "");
    }
    os << '}';
    return os;
}

template <class T1, class T2>
std::ostream &operator<<(ostream &os, map<T1, T2> &a) {
    os << "{\n";
    for(typename map<T1, T2>::iterator it = a.begin(); it != a.end(); it++) {
        typename map<T1, T2>::iterator jt = it;
        os << it->first << ": " << it->second << (++jt != a.end()? ", " : "");
    }
    os << '}';
    return os;
}
```

1.Toán

Phi hàm Euler

$\phi(n)$ là số các số nguyên dương ≤ n và nguyên tố cùng nhau với n
$\phi(1) = 1$
$\phi(n) = (p - 1)p^{k-1}$ với n là luỹ thừa bậc k của số nguyên tố p
$\phi(mn) = \phi(m) \times \phi(n)$ với m và n nguyên tố cùng nhau
$n = p1^{k1}…pr^{kr}$ với pj là các số nguyên tố phân biệt thì $\phi(n) = (p1 - 1)p1^{k1-1}…(pr - 1)pr^{kr-1}$

```
int phi(int n) {
    int res = n;
    for (int i = 2; (long long)i * i <= n; i++)
        if (n % i == 0) {
            while (n % i == 0) n /= i;
            res -= res / i;
        }
    if (n > 1) res -= res / n;
    return res;
}
```

Modulo trick

$(A / B) \% MOD = (A \% (MOD \times B)) / B$ Điều kiện: không có
$(A / B) \% MOD = ((A \% MOD) \times (B^{\phi(MOD)-1} \% MOD)) \% MOD$ Điều kiện: B và MOD nguyên tố cùng nhau
$(A / B) \% MOD = ((A \% MOD) \times (B^{MOD-2} \% MOD)) \% MOD$ Điều kiện: B và MOD nguyên tố cùng nhau, MOD nguyên tố
$A^N \% MOD = A^{N \% \phi(MOD)} \% MOD$ Điều kiện: A và MOD nguyên tố cùng nhau
$A^{B^C} \% MOD = A^{[B^C \% \phi(MOD)]} \% MOD$ Điều kiện: A và MOD nguyên tố cùng nhau
$(A^{\phi(n)} - 1) \% n = 0$
Số tự nhiên $n > 1$ là số nguyên tố khi và chỉ khi $(n-1)! \equiv n - 1 (\mathrm{mod}\ n)$

Lehman

```
#include <bits/stdc++.h>
using namespace std;
typedef unsigned long long ull;
ull lehman_simple(ull n) {
    ull n_1_3 = (ull ) ceil(pow(n, 1.0/3.0));
    double n_1_6 = pow(n, 1.0/6.0);
    ull ub_d = max(n_1_3, (ull ) 19);
    for(ull d=2; d<=ub_d; d++)
        if(n % d == 0) return d;
    for(ull k=1; k<=n_1_3; k++) {
        ull lb = ceil(2*sqrt(k)*sqrt(n));
        ull ub = floor(2*sqrt(k)*sqrt(n) + n_1_6/(4*sqrt(k)));
        for(ull a=lb; a<=ub; a++) {
            ull delta = a*a - 4*k*n;
            ull b = floor(sqrt(delta));
            if(b*b == delta) {
                return __gcd(a+b, n);
            }
        }
    }
    return n;
}
void lehman(ull n, ull & p, ull & k, ull & m) {
    m = n;
    do {
        p = m;
        m = lehman_simple(p);
    } while(m != p);
    k = 0;
    while(n % p == 0) {
        n /= p;
        ++k;
    }
    m = n;
}
vector<ull> factory_prime(ull n) {
    vector <ull> vt;
    ull p, k, m;
    lehman(n, p, k, m);
    for (int i = 1; i <= k; ++i) {
        vt.push_back(p);
    }
    while(m != 1) {
        lehman(m, p, k, m);
        for (int i = 1; i <= k; ++i) {
            vt.push_back(p);
```

```
        }
    }
    return vt;
}
int main() {
    ull n;
    cin >> n;
    vector<ull> vt = factory_prime(n);
    for (int i = 0; i < vt.size(); i++) cout << vt[i] << " ";
}// input: 12 output: 2 2 3
```

Miller Rabin

```
#include <bits/stdc++.h>
typedef long long ll;
using namespace std;
ll mulmod(ll a, ll b, ll mod) {
    ll x = 0,y = a % mod;
    while (b > 0) {
        if (b % 2 == 1) {
            x = (x + y) % mod;
        }
        y = (y * 2) % mod;
        b /= 2;
    }
    return x % mod;
}
ll modulo(ll base, ll exponent, ll mod) {
    ll x = 1;
    ll y = base;
    while (exponent > 0) {
        if (exponent % 2 == 1)
            x = (x * y) % mod;
        y = (y * y) % mod;
        exponent = exponent / 2;
    }
    return x % mod;
}
bool Miller(ll p,int iteration) {
    if (p < 2) {
        return false;
    }
    if (p != 2 && p % 2==0) {
        return false;
    }
    ll s = p - 1;
    while (s % 2 == 0) {
        s /= 2;
    }
    for (int i = 0; i < iteration; i++) {
        ll a = rand() % (p - 1) + 1, temp = s;
        ll mod = modulo(a, temp, p);
        while (temp != p - 1 && mod != 1 && mod != p - 1) {
            mod = mulmod(mod, mod, p);
            temp *= 2;
        }
        if (mod != p - 1 && temp % 2 == 0) {
            return false;
        }
    }
    return true;
}
int main() {
    int iteration = 5;
    ll num;
    cout<<"Enter integer to test primality: ";
    cin>>num;
    if (Miller(num, iteration))
        cout<<num<<" is prime"<<endl;
    else
        cout<<num<<" is not prime"<<endl;
    return 0;
}
```

Chinese Remainder Theorem

```
typedef int num_t;
namespace CRT {
    num_t res = 0;
    num_t prd = 1;

    void clear() {
        res = 0, prd = 1;
    }
    num_t mul(num_t a, num_t b, num_t p) {
        a %= p, b %= p;
        num_t q = (num_t) ((long double) a * b / p);
        num_t r = a * b - q * p;
        while (r < 0) r += p;
        while (r >= p) r -= p;
        return r;
    }
    template<typename num_t>
    pair<num_t, num_t> euclid(num_t a, num_t b) {
```

```
        if (!b) return make_pair(1, 0);
        pair<num_t, num_t> r = euclid(b, a % b);
        return make_pair(r.second, r.first - a / b * r.second);
    }
    void add(num_t p, num_t r) {
        res += mul(r - res % p + p, euclid(prd, p).first + p, p) * prd;
        prd *= p;
        if (res >= prd) res -= prd;
    }
}
```

Đếm số các số ≤ n có k bit 1

```
#include <bits/stdc++.h>
using namespace std;
#define LL long long
LL getBit(LL x){
      LL ans = -1;
      while(x) {
            ans++;
            x >>= 1;
      }
      return ans;
}
LL c[65][65], a, b;
LL calC(LL m, LL k){
      if(c[m][k] != 0) return c[m][k];
      if(k == 0 || k == m) return c[m][k] = 1;
      return c[m][k] = calC(m - 1, k) + calC(m-1, k-1);
}
LL f(LL a, LL k){
      if(k < 0) return 0LL;
      LL m = getBit(a);
      if(m < k) return 0LL;
      return calC(m, k) +  f(a & ((1ll<<m)-1ll), k-1ll);
}
int main() {
      LL n, k;
      int t;
      cin >> t;
      while (t--) {
            cin >> n >> k;
            cout << f(n, k) << endl;
      }
}
```

Bất đẳng thức Bunyakovsky cho bộ 2 số

Với 2 bộ số $(a_1; a_2; \dots; a_n)$ và $(b_1; b_2; \dots; b_n)$ ta có:

$$(a_1^2 + a_2^2 + \cdots + a_n^2)(b_1^2 + b_2^2 + \cdots + b_n^2) \geq (a_1b_1 + a_2b_2 + \cdots + a_nb_n)$$

Dấu "=" xảy ra khi và chỉ khi $\frac{a_1}{b_1} = \frac{a_2}{b_2} = \cdots = \frac{a_n}{b_n}$

Hệ quả: $(a^2 + b^2)(c^2 + d^2) \geq 4abcd$

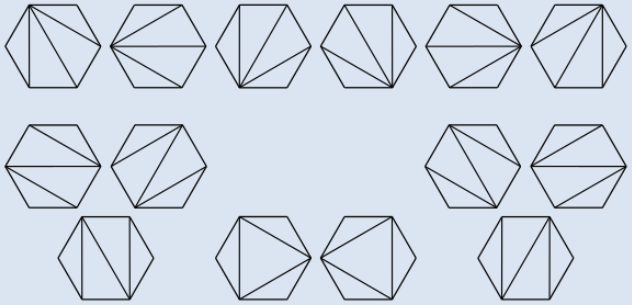
Số Catalan

$$C_n = \frac{(2n)!}{(n+1)!\,n!} = \prod_{k=2}^{n} \frac{n+k}{k} \; (n \geq 0)$$

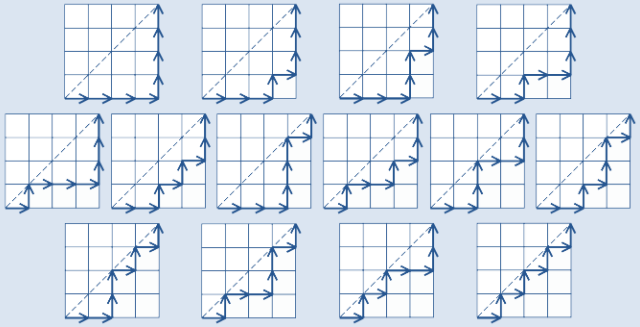
$$C_0 = 1; C_{n+1} = \sum_{i=0}^{n} C_i C_{n-i} \; (n \geq 0)$$

Ứng dụng của C_n:

1. Số cây nhị phân có n đỉnh
2. Số xâu ngoặc đúng có n cặp dấu đóng mở ngoặc tương ứng
3. Số cách chia 1 đa giác lồi có n+2 cạnh thành các tam giác bằng cách nối các đỉnh với nhau mà không cắt nhau



4. Số cây nhị phân đầy đủ (mỗi đỉnh có 2 đỉnh con hoặc không có đỉnh con nào) có n+1 lá
5. Số lượng đường đi với 2n bước trên lưới hình chữ nhật từ điểm trái dưới (0,0) đến điểm phải trên (n,n) mà không vượt qua đường chéo chính



6. Số cách thêm n cặp dấu ngoặc (hoặc n-1) vào 1 xâu n+1 kí tự mà vẫn thoả mãn tính đúng của các dấu ngoặc

7. Số hoán vị độ dài n mà không có 3 phần tử liên tiếp nào tạo thành dãy tăng dần

8. Các số từ 1 đến n xếp lần lượt theo chiều kim đồng hồ thành vòng tròn. C_n là số cách chia tập hợp các số từ 1 đến n thành các tập hợp con khác rỗng sao cho không có 2 tập con nào tạo thành các đa giác giao nhau

Discrete Logarithm

Tìm số nguyên x thỏa mãn $a^x \equiv b \pmod m$, trong đó a và m nguyên tố cùng nhau

```
int solve (int a, int b, int m) {
    int n = (int) sqrt (m + .0) + 1;
    int an = 1;
    for (int i=0; i<n; ++i)
        an = (an * a) % m;
    map<int,int> vals;
    for (int i=1, cur=an; i<=n; ++i) {
        if (!vals.count(cur))
            vals[cur] = i;
        cur = (cur * an) % m;
    }
    for (int i=0, cur=b; i<=n; ++i) {
        if (vals.count(cur)) {
            int ans = vals[cur] * n - i;
            if (ans < m)
                return ans;
        }
        cur = (cur * a) % m;
    }
    return -1;
}
```

Discrete Root

Cho số nguyên tố n và 2 số nguyên a, k, tìm mọi x thỏa mãn $x^k \equiv a \pmod n$

```
vector<int> discrete_root(int n, int k, int a) {
	function<int(int, int, int)> powmod = [&](int a, int b, int p) -> int {
		int res = 1;
		while (b)
			if (b & 1) res = int (res * 1LL * a % p), --b;
			else a = int (a * 1LL * a % p),  b >>= 1;
		return res;
	};

	function<int(int)> generator = [&](int p) -> int {
		vector<int> fact;
		int phi = p-1, n = phi;
		for (int i=2; i<=sqrt(n); i++) {
			if (n % i == 0) {
				fact.push_back(i);
				while (n % i == 0) n /= i;
			}
		}
		if (n > 1) fact.push_back (n);

		for (int res=2; res<=p; ++res) {
			bool ok = true;
			for (size_t i=0; i<fact.size() && ok; ++i)
				ok &= powmod (res, phi / fact[i], p) != 1;
			if (ok) return res;
		}
		return -1;
	};

	if (a == 0) return vector<int>(1, 0);
	if (a == 1 && k == 0) return vector<int>(1, -4);
	// This means everything within [1, n-1]

	int g = generator(n), sq = sqrt(n) + 1;
	vector < pair<int, int> > dec(sq);
	for (int i=1; i<=sq; ++i)
		dec[i-1] = {powmod (g, int (i * sq * 1ll * k % (n - 1)), n), i};
	sort(dec.begin(), dec.end()); int any_ans = -1;
	for (int i=0; i<sq; ++i) {
		int my = powmod(g, int (i * 1ll * k % (n - 1)), n) * 1ll * a % n;
		auto it = lower_bound(dec.begin(), dec.end(), make_pair(my, 0));
		if (it != dec.end() && it->first == my) {
			any_ans = it->second * sq - i; break;
		}
	}
	if (any_ans == -1) return vector<int>(0);
	int delta = (n-1) / __gcd(k, n-1); vector<int> ans;
	for (int cur=any_ans%delta; cur<n-1; cur+=delta)
		ans.push_back (powmod(g, cur, n));
	sort(ans.begin(), ans.end()); return ans;
}
```

k-th term of a linear recurrence (O(n^3xlog(k))

Tìm phần tử thứ k của dãy S(i) = S(i-1)*tr(0) + … + S(i-n)*tr(n-1) (i ≥ n)

```
int linearRec(vector<int> S, vector<int> tr, int k, int Mod) {
	#define Matrix vector<vector<int>>
	function<Matrix(Matrix, Matrix)> MatMul = [&](Matrix a, Matrix b) -> Matrix
{
		int n = a.size(), k = a[0].size(), m = b[0].size();
		Matrix res(n, vector<int>(m, 0));
		for (int z=0; z<k; z++) {
			for (int i=0; i<n; i++) {
				for (int j=0; j<m; j++) {
					res[i][j] += (1LL * a[i][z] * b[z][j]) % Mod;
					res[i][j] %= Mod;
				}
			}
		}
		return res;
	};

	function<Matrix(int)> UnitMatrix = [&](int n) -> Matrix {
		Matrix res(n, vector<int>(n, 0));
		for (int i=0; i<n; i++) res[i][i] = 1;
		return res;
	};

	function<Matrix(Matrix, int)> MatPow = [&](Matrix a, int b) -> Matrix {
		// a is guaranteed to be a square matrix
		Matrix res = UnitMatrix(a.size());
		while (b > 0) {
			if (b % 2 == 1) {res = MatMul(res, a); b--;}
			else {a = MatMul(a, a); b /= 2;}
		}
		return res;
	};

	int n = S.size();

	Matrix TransformationMatrix(n, vector<int>(n, 0));
	for (int i=1; i<n; i++) TransformationMatrix[i-1][i] = 1;
	for (int i=0; i<n; i++) TransformationMatrix[n-1][i] = tr[n-1-i];

	Matrix BaseMatrix(n, vector<int>(1, 0));
	for (int i=0; i<n; i++) BaseMatrix[i][0] = S[i];

	Matrix PostTransform = MatMul(MatPow(TransformationMatrix, k), BaseMatrix);
	#undef Matrix
	return PostTransform[0][0];
}
```

k-th term of a linear recurrence (O(n^2xlog(k))

Tìm phần tử thứ k của dãy S(i) = S(i-1)*tr(0) + … + S(i-n)*tr(n-1) (i ≥ n)

```
int linearRec(vector<int> S, vector<int> tr, int k, int Mod) {
	int n = S.size();

	auto combine = [&](vector<int> a, vector<int> b) {
		vector<int> res(n * 2 + 1);
		for (int i=0; i<n+1; i++) for (int j=0; j<n+1; j++)
			res[i + j] = (res[i + j] + (1LL * a[i] * b[j]) % Mod) % Mod;
		for (int i = 2 * n; i > n; --i) for (int j=0; j<n; j++) {
			int toAdd = (1LL * res[i] * tr[j]) % Mod;
			res[i - 1 - j] = (res[i - 1 - j] + toAdd) % Mod;
		}
		res.resize(n + 1);
		return res;
	};

	vector<int> pol(n + 1), e(pol);
	pol[0] = e[1] = 1;

	for (++k; k; k /= 2) {
		if (k % 2) pol = combine(pol, e);
		e = combine(e, e);
	}

	int res = 0;
	for (int i=0; i<n; i++) res = (res + (1LL * pol[i + 1] * S[i]) % Mod) % Mod;
	return res;
}
```

2. Đồ thị

Tarjan: Tìm thành phần liên thông mạnh

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

#define LL long long
typedef vector<LL> vi;

LL n, m, SCCcnt = 0;
LL Time = 0;
vector<vi> adj; vi Lowest, Enum;
stack<LL> S;

void traverse(LL z) {
	Lowest[z] = ++Time; Enum[z] = Time; S.push(z);
	for (LL i=0; i<adj[z].size(); i++) {
```

```
            if (Enum[adj[z][i]] == 0) traverse(adj[z][i]);
            Lowest[z] = min(Lowest[z], Lowest[adj[z][i]]);
      }
      if (Enum[z] == Lowest[z]) {
            SCCcnt++; LL t;
            do {
                  t = S.top(); S.pop();
                  Lowest[t] = 1e18; Enum[t] = 1e18;
            }
            while (z != t);
      }
}

int main() {
      ios_base::sync_with_stdio(0);
      cin.tie(0); cout.tie(0);
      cin >> n >> m; adj.resize(n+1, vi(0));
      while (m--) {
            LL u, v; cin >> u >> v;
            adj[u].push_back(v);
      }
      Lowest.resize(n+1, 0); Enum = Lowest;
      for (LL i=1; i<=n; i++) {
            if (Lowest[i] == 0) traverse(i);
      }
      cout << SCCcnt;
      return 0;
}
```

Sắp xếp topo

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 8;
vector<int> G[N];
bool trv[N], done[N], DAG = true;
int topo[N], n, m, cnt;
void dfs(int u) {
    if (trv[u]) {
        DAG = false;
        return;
    }
    if (done[u]) return;
    trv[u] = true;
    for (int v: G[u]) dfs(v);
    trv[u] = false;
    done[u] = true;
    topo[cnt--] = u;
}
void toposort() {
    for (int i = 1; i <= n; i++) {
        sort(G[i].begin(), G[i].end());
        reverse(G[i].begin(), G[i].end());
    }
    cnt = n;
    for (int i = n; i >= 1; i--) if (!done[i]) dfs(i);
}
int main()
{
    cin >> n >> m;
    for (int i = 1; i <= m; i++) {int u, v; scanf("%d %d", &u, &v);
G[u].push_back(v);}
    toposort();
    if (DAG) for (int i = 1; i <= n; i++) printf("%d ", topo[i]);
    else cout << "NOT DAG";
}
```

Tìm chu trình Euler

```
#include <bits/stdc++.h>
using namespace std;
const int MAXN = 100000;
vector<int> euler_cycle_directed(vector<int> adj[], int u) {
    vector<int> stack, res, cur_edge(MAXN);
    stack.push_back(u);
    while (!stack.empty()) {
        u = stack.back();
        stack.pop_back();
        while (cur_edge[u] < (int)adj[u].size()) {
            stack.push_back(u);
            u = adj[u][cur_edge[u]++];
        }
        res.push_back(u);
    }
    reverse(res.begin(), res.end());
    return res;
}
vector<int> euler_cycle_undirected(vector<int> adj[], int u) {
    vector<vector<bool> > used(MAXN, vector<bool>(MAXN, false));
    vector<int> stack, res, cur_edge(MAXN);
    stack.push_back(u);
    while (!stack.empty()) {
        u = stack.back();
        stack.pop_back();
```

```
        while (cur_edge[u] < (int)adj[u].size()) {
            int v = adj[u][cur_edge[u]++];
            if (!used[min(u, v)][max(u, v)]) {
                used[min(u, v)][max(u, v)] = 1;
                stack.push_back(u);
                u = v;
            }
        }
        res.push_back(u);
    }
    reverse(res.begin(), res.end());
    return res;
}
int main() {
    int nodes, edges, u, v;
    vector<int> g1[5], g2[5], cycle;
    cin >> nodes >> edges;
    for (int i = 0; i < edges; i++) {
        cin >> u >> v;
        g1[u].push_back(v);
        g2[u].push_back(v);
        g2[v].push_back(u);
    }
    cycle = euler_cycle_directed(g1, 0);
    cout << "Eulerian cycle from 0 (directed): ";
    for (int i = 0; i < (int)cycle.size(); i++)
        cout << " " << cycle[i];
    cout <<"\n";
    cycle = euler_cycle_undirected(g2, 2);
    cout << "Eulerian cycle from 2 (undirected): ";
    for (int i = 0; i < (int)cycle.size(); i++)
        cout << " " << cycle[i];
    cout << "\n";
    return 0;
}
```

Cặp ghép cực đại trên đồ thị 2 phía

```
#include <bits/stdc++.h>
using namespace std;
const int N = 102;
int n, m, Assigned[N];
int Visited[N], t=0;
vector<int> a[N];
bool visit(int u) {
    if (Visited[u]!=t)
        Visited[u]=t;
    else
        return false;
    for (int i=0; int v=a[u][i]; i++)
    if (!Assigned[v] || visit(Assigned[v])) {
        Assigned[v]=u;
        return true;
    }
    return false;
}
main() {
    scanf("%d%d", &m, &n);
    int x, y;
    while (scanf("%d%d", &x, &y) > 0)
        a[x].push_back(y);
    for (int i=1; i<=m; i++)
        a[i].push_back(0);
    int Count = 0;
    for (int i=1; i<=m; i++) {
        t++;
        Count += visit(i);
    }
    printf("%d\n", Count);
    for (int i=1; i<=n; i++)
    if (int j=Assigned[i])
    printf("%d %d\n", j, i);
}
```

Khớp & cầu

```
#include <bits/stdc++.h>
using namespace std;
const int N = 100005;
int n, m;
vector<int> a[N];
int CriticalEdge=0;
bool CriticalNode[N];
int Num[N], Low[N], Time=0;
void visit(int u, int p) {
    int NumChild = 0;
    Low[u] = Num[u] = ++Time;
    for (int i=0; int v=a[u][i]; i++)
    if (v!=p) {
        if (Num[v]!=0)
```

```
            Low[u] = min(Low[u], Num[v]);
        else {
            visit(v, u);
            NumChild++;
            Low[u] = min(Low[u], Low[v]);
            if (Low[v] >= Num[v])
                CriticalEdge++;
            if (u==p) {
                if (NumChild >= 2)
                CriticalNode[u] = true;
            } else {
                if (Low[v] >= Num[u])
                CriticalNode[u] = true;
            }
        }
    }
}
main() {
    scanf("%d%d", &n, &m);
    for (int i=1; i<=m; i++) {
        int x, y;
        scanf("%d%d", &x, &y);
        a[x].push_back(y);
        a[y].push_back(x);
    }
    for (int i=1; i<=n; i++)
        a[i].push_back(0);
    for (int i=1; i<=n; i++)
        if (!Num[i]) visit(i, i);
    int Count = 0;
    for (int i=1; i<=n; i++)
        if (CriticalNode[i]) Count++;
    printf("%d %d\n", Count, CriticalEdge);
}
```

Mincost-Maxflow

Add cạnh với trọng số 0 nếu bài toán chỉ là maxflow

```
struct Edge {int v, rev, cap, f, cost;};

class MaxFlow {
private:
      int n;
      vector<vector<Edge>> g;
      vector<int> pNode, pEdge;

      bool spfa(int s, int t) {
            queue<int> q; pNode.assign(n, -1); pEdge.assign(n, -1);
            vector<bool> inQueue(n, false); vector<int> dist(n, INF);

            q.push(s); inQueue[s] = true; dist[s] = 0;

            while (!q.empty()) {
                  int u = q.front(); q.pop();
                  inQueue[u] = false;

                  for(int i = 0; i < g[u].size(); ++i) {
                        Edge e = g[u][i];
                        if (e.cap - e.f == 0)
                              continue;
                        if (dist[u] + e.cost < dist[e.v]) {
                              dist[e.v] = dist[u] + e.cost;
                              pNode[e.v] = u;
                              pEdge[e.v] = i;
                              if (!inQueue[e.v]) {
                                    q.push(e.v);
                                    inQueue[e.v] = true;
                              }
                        }
                  }
            }

            return (dist[t] != INF);
      }

      pair<int, int> increaseFlow(int s, int t) {
            int df = INF;
            for(int v = t; v != s; v = pNode[v]) {
                  int u = pNode[v], i = pEdge[v];
                  df = min(df, g[u][i].cap - g[u][i].f);
            }

            int dcost = 0;
            for(int v = t; v != s; v = pNode[v]) {
                  int u = pNode[v], i = pEdge[v];
                  g[u][i].f += df;
                  g[v][g[u][i].rev].f -= df;
                  dcost += g[u][i].cost * df;
            }

            return {df, dcost};
      }
```

```
public:
      MaxFlow(int n): n(n) {
            g.assign(n, vector<Edge>());
      }

      void addEdge(int u, int v, int cap, int cost) {
            g[u].push_back({v, (int)g[v].size(), cap, 0, cost});
            g[v].push_back({u, (int)g[u].size() - 1, 0, 0, -cost});
      }

      pair<int, int> getMaxFlow(int s, int t) {
            int flow = 0, cost = 0;
            while (spfa(s, t)) {
                  pair<int, int> res = increaseFlow(s, t);
                  flow += res.first; cost += res.second;
            }
            return {flow, cost};
      }
};
```

Edmonds – Karp và lát cắt hẹp nhất trong mạng

Độ phức tạp: $O(nm^2)$

```
#include <bits/stdc++.h>
using namespace std;
void minimize(int &a, int b){
    if (a>b) a=b;
}
int n, m;
vector<int> a[12309];
int start, target;
int c[123][123];
int f[123][123];
int d[12309];
bool bfs(int start, int target){
    queue<int> qu;
    int u, i, v;

    for (i=1; i<=n; i++) d[i]=0;
    d[start] = -1;
    qu.push(start);

    while (qu.size()){
        u=qu.front(); qu.pop();
        if (u==target) return true;
        for (i=0; v=a[u][i]; i++)
        if (d[v]==0 && f[u][v]<c[u][v]){
            d[v]=u;
            qu.push(v);
        }
    }
    return false;
}
int mincut(bool tracing=false){
    int u, i, v, r=0;
    for (u=1; u<=n; u++)
    for (i=0; v=a[u][i]; i++)
    if (d[u] && !d[v]) {
        r += c[u][v];
        if (tracing) printf("%d %d\n", u, v);
    }
    return r;
}
void enlarge(){
    int i;
    int delta=1000111000;
    for (i=target; i!=start; i=d[i])
    minimize(delta, c[d[i]][i]-f[d[i]][i]);
    for (i=target; i!=start; i=d[i]){
        f[d[i]][i] += delta;
        f[i][d[i]] -= delta;
    }
}
main(){
    int i, p, q, w;
    for (;;){
        scanf("%d%d", &n, &m);
        if (n==0) return 0;
        for (i=1; i<=n; i++) a[i].clear();
        for (p=1; p<=n; p++)
        for (q=1; q<=n; q++)
        c[p][q]=f[p][q]=0;
        start=1, target=2;
        for (i=1; i<=m; i++){
            scanf("%d%d%d", &p, &q, &w);
            a[p].push_back(q);
            a[q].push_back(p);
            c[p][q]=c[q][p]=w;
        }
```

```
        for (i=1; i<=n; i++) a[i].push_back(0);
        while (bfs(start, target)) enlarge();
        mincut(true); printf("\n");
    }
}
```

3. Hình học

Hình học cơ bản

```
#define EPS 1e-9
struct point_t {
    double x, y;
    point_t() : x(0), y(0) {}
    point_t(double x, double y) : x(x), y(y) {}
    point_t(const point_t& p) : x(p.x), y(p.y) {}
    int operator < (const point_t& rhs) const {return make_pair(y, x) <
make_pair(rhs.y, rhs.x);}
    int operator == (const point_t& rhs) const {return make_pair(y, x) ==
make_pair(rhs.y, rhs.x);}
    point_t operator + (const point_t& p) const {return point_t(x + p.x, y
+ p.y);}
    point_t operator - (const point_t& p) const {return point_t(x - p.x, y
- p.y);}
    point_t operator * (double c) const {return point_t(x * c, y * c);}
    point_t operator / (double c) const {return point_t(x / c, y / c);}
};
double cross(point_t p, point_t q) {return p.x * q.y - p.y * q.x;}
double area(point_t a, point_t b, point_t c) {return fabs(cross(a, b) +
cross(b, c) + cross(c, a)) / 2;}
double area2(point_t a, point_t b, point_t c) {return cross(a, b) +
cross(b, c) + cross(c, a);}
double dot(point_t p, point_t q) {return p.x * q.x + p.y * q.y;}
double dist(point_t p, point_t q) {return sqrt(dot(p - q, p - q));}
double dist2(point_t p, point_t q) {return dot(p - q, p - q);}
point_t RotateCCW90(point_t p) {return point_t(-p.y, p.x);}
point_t RotateCW90(point_t p) {return point_t(p.y, -p.x);}
point_t RotateCCW(point_t p, double t) {return point_t(p.x * cos(t) - p.y
* sin(t), p.x * sin(t) + p.y * cos(t));}
int sign(double x) {return x < -EPS ? -1 : x > EPS;}
int sign(double x, double y) {return sign(x - y);}
ostream& operator << (ostream& os, const point_t& p) {
    os << "(" << p.x << "," << p.y << ")";
    return os;
}

//Project c on Line(a, b)
point_t ProjectPointLine(point_t a, point_t b, point_t c) {
    return a + (b - a) * dot(c - a, b - a) / dot(b - a, b - a);
}
point_t ProjectPointSegment(point_t a, point_t b, point_t c) {
    double r = dot(b - a, b - a);
    if (fabs(r) < EPS) return a;
    r = dot(c - a, b - a) / r;
    if (r < 0) return a;
    if (r > 1) return b;
    return a + (b - a) * r;
}
double DistancePointSegment(point_t a, point_t b, point_t c) {
    return dist(c, ProjectPointSegment(a, b, c));
}
//Compute distance between point_t (x, y, z) and plane ax + by + cz = d
double DistancePointPlane(double x, double y, double z, double a, double
b, double c, double d) {
    return fabs(a * x + b * y + c * z - d) / sqrt(a * a + b * b + c * c);
}
//Determine if lines from a to b and c to d are parallel or collinear
int LinesParallel(point_t a, point_t b, point_t c, point_t d) {
    return fabs(cross(b - a, c - d)) < EPS;
}
int LinesCollinear(point_t a, point_t b, point_t c, point_t d) {
    return LinesParallel(a, b, c, d) && fabs(cross(a - b, a - c)) < EPS &&
fabs(cross(c - d, c - a)) < EPS;
}
//Determine if line segment from a to b intersects with line segment from
c to d
int SegmentsIntersect(point_t a, point_t b, point_t c, point_t d) {
    if (LinesCollinear(a, b, c, d)) {
        if (dist2(a, c) < EPS || dist2(a, d) < EPS || dist2(b, c) < EPS ||
dist2(b, d) < EPS) return 1;
        if (dot(c - a, c - b) > 0 && dot(d - a, d - b) > 0 && dot(c - b, d
- b) > 0) return 0;
        return 1;
    }
    if (cross(d - a, b - a) * cross(c - a, b - a) > 0) return 0;
    if (cross(a - c, d - c) * cross(b - c, d - c) > 0) return 0;
    return 1;
}
//Compute intersection of line passing through a and b
//with line passing through c and d, assuming that unique
//intersection exists; for segment intersection, check if
//segments intersect first
point_t ComputeLineIntersection(point_t a, point_t b, point_t c, point_t
```

```
d) {
    b = b - a; d = c - d; c = c - a;
    return a + b * cross(c, d) / cross(b, d);
}
//Compute center of circle given three points
point_t ComputeCircleCenter(point_t a, point_t b, point_t c) {
    b = (a + b) / 2;
    c = (a + c) / 2;
    return ComputeLineIntersection(b, b + RotateCW90(a - b), c, c +
RotateCW90(a - c));
}
//Determine if point is in a possibly non-convex polygon
//returns 1 for strictly interior points, 0 for
//strictly exterior points, and 0 or 1 for the remaining points.
int PointInPolygonSlow(const vector<point_t>& p, point_t q) {
    int c = 0;
    for (int i = 0; i < p.size(); i++) {
        int j = (i + 1) % p.size();
        if ((p[i].y <= q.y && q.y < p[j].y || p[j].y <= q.y && q.y <
p[i].y) && q.x < p[i].x + (p[j].x - p[i].x) * (q.y - p[i].y) / (p[j].y -
p[i].y)) c = !c;
    }
    return c;
}
//Strictly inside convex Polygon
#define Det(a, b, c) ((b.x - a.x) * (c.y - a.y) - (b.y - a.y) * (c.x -
a.x))
int PointInPolygon(vector<point_t>& p, point_t q) {
    int a = 1, b = p.size() - 1, c;
    if (Det(p[0], p[a], p[b]) > 0) swap(a, b);
    //Allow on edge --> if (Det... > 0 || Det ... < 0)
    if (Det(p[0], p[a], q) >= 0 || Det(p[0], p[b], q) <= 0) return 0;
    while(abs(a - b) > 1) {
        c = (a + b) / 2;
        if (Det(p[0], p[c], q) > 0) b = c; else a = c;
    }
    //Alow on edge --> return Det... <= 0
    return Det(p[a], p[b], q) < 0;
}
//Determine if point is on the boundary of a polygon
int PointOnPolygon(const vector<point_t>& p, point_t q) {
    for (int i = 0; i < p.size(); i++) if (dist2(ProjectPointSegment(p[i],
p[(i + 1) % p.size()], q), q) < EPS) return 1;
    return 0;
}
```

```
}
//Compute intersection of line through points a and b with circle centered
at c with radius r > 0
vector<point_t> CircleLineIntersection(point_t a, point_t b, point_t c,
double r) {
    vector<point_t> res;
    b = b - a; a = a - c;
    double A = dot(b, b);
    double B = dot(a, b);
    double C = dot(a, a) - r * r;
    double D = B * B - A * C;
    if (D < -EPS) return res;
    res.push_back(c + a + b * (-B + sqrt(D + EPS)) / A);
    if (D > EPS) res.push_back(c + a + b * (-B - sqrt(D)) / A);
    return res;
}
//Compute intersection of circle centered at a with radius r with circle
centered at b with radius R
vector<point_t> CircleCircleIntersection(point_t a, point_t b, double r,
double R) {
    vector<point_t> res;
    double d = sqrt(dist2(a, b));
    if (d > r + R || d + min(r, R) < max(r, R)) return res;
    double x = (d * d - R * R + r * r) / (2 * d);
    double y = sqrt(r * r - x * x);
    point_t v = (b - a) / d;
    res.push_back(a + v * x + RotateCCW90(v) * y);
    if (y > 0) res.push_back(a + v * x - RotateCCW90(v) * y);
    return res;
}
```

Monotone chain

```
void ConvexHull(vector<point_t>& pts, vector<point_t>& up,
vector<point_t>& dn) {
    sort(pts.begin(), pts.end());
    pts.erase(unique(pts.begin(), pts.end()), pts.end());
    for (int i = 0; i < pts.size(); i++) {
        while (up.size() > 1 && area(up[up.size() - 2], up.back(), pts[i])
>= 0) up.pop_back();
        while (dn.size() > 1 && area(dn[dn.size() - 2], dn.back(), pts[i])
<= 0) dn.pop_back();
        up.push_back(pts[i]); dn.push_back(pts[i]);
    }
    pts = dn;
    for (int i = up.size() - 2; i >= 1; i--) pts.push_back(up[i]);}
```

Một số công thức trong tam giác

Độ dài trung tuyến: $m_a = \sqrt{\frac{2b^2+2c^2-a^2}{4}}$

Độ dài đường phân giác: $l_a = \frac{2bc\cos\frac{A}{2}}{b+c}$

Bán kính đường tròn nội tiếp: $r = \frac{2S}{a+b+c} = \frac{S}{p} = (p-a)\tan\frac{A}{2}$

Bán kính đường tròn ngoại tiếp: $R = \frac{a}{\sin A} = \frac{b}{\sin B} = \frac{c}{\sin C} = \frac{abc}{4S}$

Emo Welzl: Đường tròn nhỏ nhất chứa mọi điểm cho trước

```
#include <bits/stdc++.h>
using namespace std;
typedef pair<double, double> point;
typedef pair<point, double> circle;
#define X first
#define Y second
point operator + (point a, point b) { return point(a.X+b.X, a.Y+b.Y); }
point operator - (point a, point b) { return point(a.X-b.X, a.Y-b.Y); }
point operator / (point a, double x) { return point(a.X/x, a.Y/x); }
double abs(point a) { return sqrt(a.X*a.X+a.Y*a.Y); }
point center_from(double bx, double by, double cx, double cy) {
    double B=bx*bx+by*by, C=cx*cx+cy*cy, D=bx*cy-by*cx;
    return point((cy*B-by*C)/(2*D), (bx*C-cx*B)/(2*D));
}
circle circle_from(point A, point B, point C) {
    point I = center_from(B.X-A.X, B.Y-A.Y, C.X-A.X, C.Y-A.Y);
    return circle(I+A, abs(I));
}
const int N = 100005;
int n, x[N], y[N];
point a[N];
circle f(int n, vector<point> T) {
    if (T.size()==3 || n==0) {
        if (T.size()==0) return circle(point(0, 0), -1);
        if (T.size()==1) return circle(T[0], 0);
        if (T.size()==2) return circle((T[0]+T[1])/2, abs(T[0]-T[1])/2);
        return circle_from(T[0], T[1], T[2]);
    }
    random_shuffle(a+1, a+n+1);
    circle Result = f(0, T);
    for (int i=1; i<=n; i++)
    if (abs(Result.X - a[i]) > Result.Y+1e-9) {
        T.push_back(a[i]);
        Result = f(i-1, T);
        T.pop_back();
    }
    return Result;
}
int main() {
    scanf("%d", &n);
    for (int i=1; i<=n; i++) {
        scanf("%d%d", &x[i], &y[i]);
        a[i] = point(x[i], y[i]);
    }
    circle C = f(n, vector<point>());
    (cout << fixed).precision(2);
    cout << 2*C.Y << endl;
}
```

Đường tròn đi qua nhiều điểm nhất

```
#include <bits/stdc++.h>
using namespace std;
const double eps = 1e-9;
struct point {
    double x, y;
};
struct line {
    double a, b, c;
};
point P[101];
double dist(point A, point B) {
    return sqrt((A.x - B.x)*(A.x - B.x) + (A.y - B.y)*(A.y - B.y));
}
bool eq(double A, double B) {
    return fabs(A - B) < eps;
}
line extract(point p1, point p2) {
    line res;
    res.a = p1.y - p2.y;
    res.b = p2.x - p1.x;
    res.c = -res.a * p1.x - res.b * p1.y;
    return res;
}
line create(point p, double A, double B) {
    line res;
    if (eq(A, 0)) {
        res.a = 1;
```

```
        res.b = 0;
        res.c = -p.x;
    }
    else if (eq(B, 0)) {
        res.a = 0;
        res.b = 1;
        res.c = -p.y;
    }
    else {
        res.a = -1/(A/B);
        res.b = 1;
        res.c = -res.a * p.x - res.b * p.y;
    }
    return res;
}
line midper(point p1, point p2) {
    line tmp = extract(p1, p2);
    point ct;
    ct.x = (p1.x + p2.x) / 2;
    ct.y = (p1.y + p2.y) / 2;
    tmp = create(ct, tmp.a, tmp.b);
    return tmp;
}
bool intersect(line l1, line l2, point &p) {
    double det = l1.a * l2.b - l1.b * l2.a;
    if (eq(det, 0)) return false;
    p.x = -(l1.c * l2.b - l2.c * l1.b) / det;
    p.y = -(l2.c * l1.a - l1.c * l2.a) / det;
    return true;
}

int calc(int n) {
    if (n <= 2) return n;
    int res = 2;
    for (int i = 1; i < n - 1; i++) {
        for (int j = i + 1; j < n; j++) {
            for (int k = j + 1; k <= n; k++) {
                line l1 = midper(P[i], P[j]);
                line l2 = midper(P[j], P[k]);
                point its;
                bool ok = intersect(l1, l2, its);
                if (ok) {
                    int sum = 3;
                    double r = dist(its, P[i]);
                    for (int l = k + 1; l <= n; l++) {
                        sum += eq(r, dist(its, P[l]));
                    }
                    res = max(res, sum);
                }
            }
        }
    }
    return res;
}
int main() {
    int n;
    while (scanf("%d", &n) and n) {
        for (int i = 1; i <= n; i++) scanf("%lf %lf", &P[i].x, &P[i].y);
        printf("%d\n", calc(n));
    }
}
```

4. Xử lý xâu

Z algorithm

Z[i] là độ dài chuỗi con lớn nhất bắt đầu tại S[i] và là tiền tố của S

```
vector<int> Z_Algo(string S) {
      vector<int> z(S.size()); int x = 0, y = 0;
      for (int i=1; i<S.size(); i++) {
            z[i] = max(0, min(z[i-x], y-i+1));
            while (i+z[i] < S.size() && S[z[i]] == S[i+z[i]]) {
                  x = i; y = i + z[i]; z[i]++;
            }
      }
      return z;
}
```

Manacher: Xâu palindrome

```
int ManacherProcess(string s) {
      int n = s.size(), res = 0;
      vector<int> odd(n, 0), even(n, 0);
      for (int i=0, l=0, r=-1; i<n; i++) {
            int x = 0; if (i <= r) x = min(odd[l+r-i], r-i);
            while (0 <= i-x-1 && i+x+1 < n && s[i-x-1] == s[i+x+1]) x++;
            odd[i] = x; res += (x + 1);
            if (i + x > r) {l = i - x; r = i + x;}
      }
      for (int i=1, l=0, r=0; i<n; i++) {
            if (s[i-1] != s[i]) continue;
            int x = 0; if (i <= r) x = min(even[l+r-i+1], r-i);
```

```
            while (0 <= i-x-2 && i+x+1 < n && s[i-x-2] == s[i+x+1]) x++;
            even[i] = x; res += (x + 1);
            if (i + x > r) {l = i-1 - x; r = i + x;}
      }
      return res;
}
```

odd[x] cho biết độ dài tối đa của palindrome độ dài lẻ tâm ở x.
even[x] cho biết độ dài tối đa của palindrome độ dài chẵn tâm phải ở x.

Hàm ở trên dùng để đếm số xâu con là palindrome. Nếu muốn tìm xâu con palindrome dài nhất, ta tìm res là max của các odd[x] và even[x].

KMP: So khớp chuỗi

```
void buildPi(string& p, vector<int>& pi) {
    pi = vector<int> (p.length());
    int k = -2;
    for (int i = 0; i < p.length(); i++) {
        while(k >= -1 && p[k+1] != p[i]) k = (k == -1) ? -2 : pi[k]; pi[i]
= ++k;
    }
}
int KMP(string& t, string& p) {
    vector<int> pi;
    buildPi(p, pi);
    int k = -1;
    for (int i = 0; i < t.length(); i++) {
        while(k >= -1 && p[k+1] != t[i]) k = (k == -1) ? -2 : pi[k];
        k++;
        if(k == p.length() - 1) { // p matches t[i-m+1, ..., i]
            cout << "matched at index " << i-k << ": ";
            cout << t.substr(i-k, p.length()) << endl;
            k = (k == -1) ? -2 : pi[k];
        }
    }
    return 0;
}
```

Suffix array và Longest common prefix

sa[i] là vị trí của hậu tố có thứ tự từ điển i
lcp[i] là độ dài tiền tố chung dài nhất của hậu tố sa[i] và sa[i-1]

```
const int MAXN = 1e5;
int N, gap;
int sa[MAXN], pos[MAXN], tmp[MAXN], lcp[MAXN];
string S;
bool sufCmp(int i, int j) {
    if (pos[i] != pos[j])
        return pos[i] < pos[j];
    i += gap;
    j += gap;
    return (i < N && j < N) ? pos[i] < pos[j] : i > j;
}
void buildSA() {
    N = S.length();
    for (int i = 0; i < N; i++) sa[i] = i, pos[i] = S[i];
    for (gap = 1;; gap *= 2) {
        sort(sa, sa + N, sufCmp);
        for (int i = 0; i < N - 1; i++) tmp[i + 1] = tmp[i] +
sufCmp(sa[i], sa[i + 1]);
        for (int i = 0; i < N; i++) pos[sa[i]] = tmp[i];
        if (tmp[N - 1] == N - 1) break;
    }
}
void buildLCP() {
    for (int i = 0, k = 0; i < N; ++i) if (pos[i] != N - 1) {
        for (int j = sa[pos[i] + 1]; S[i + k] == S[j + k];)
        ++k;
        lcp[pos[i]] = k;
        if (k)--k;
    }
}
```

// Số xâu con phân biệt: $\frac{n(n+1)}{2} - \sum_{i=0}^{n-1} lcp_i$

// Điều kiện để 1 xâu độ dài q xuất hiện k lần trong xâu T là trong mảng lcp(T) tồn tại k-1 số liên tiếp ≥ q

5. Miscellaneous

QTREE

```
//1 cây n đỉnh
//CHANGE u v: đổi trọng số cạnh thứ u thành v
//QUERY u v: tìm trọng số lớn nhất của các cạnh trên đường đi từ u đến v
#include <bits/stdc++.h>
using namespace std;
const int N = 1e4 + 8;
int pos[N], d[N], p[N], t[N << 2], cn[N], cr[N], pre[N], Next[N], cid, tid, n,
eu[N], ev[N], cost[N];
char s[8];
struct data {
    int v, w;
};
vector<data> a[N];
void clear_data() {
    for (int i = 1; i <= n; i++) a[i].clear();
    for (int i = 1; i <= cid; i++) cr[i] = 0;
    memset(t, 0, sizeof(t));
```

```
    cid = 1;
    tid = -1;
    p[1] = 1;
}
void dfs(int u, int pre_) {
    d[u] = 1;
    for (int i = 0; i < a[u].size(); i++) {
        int v = a[u][i].v;
        if (v != pre_) {
            pre[v] = u;
            p[v] = p[u] + 1;
            dfs(v, u);
        }
        d[u] += d[v];
    }
}
void hld(int u) {
    if (cr[cid] == 0) cr[cid] = u;
    cn[u] = cid;
    pos[u] = ++tid;
    int id = 0, Max = 0;
    for (int i = 0; i < a[u].size(); i++) {
        if (a[u][i].v != pre[u] and d[a[u][i].v] > Max) {
            Max = d[a[u][i].v];
            id = a[u][i].v;
        }
    }
    if (id > 0) {
        Next[u] = id;
        hld(id);
    }
    for (int i = 0; i < a[u].size(); i++) {
        if (a[u][i].v == pre[u] or a[u][i].v == id) continue;
        cid++;
        hld(a[u][i].v);
    }
}
int lca(int u, int v) {
    while (cn[u] != cn[v]) {
        if (p[cr[cn[u]]] > p[cr[cn[v]]]) u = pre[cr[cn[u]]];
        else v = pre[cr[cn[v]]];
    }
    if (p[u] < p[v]) return u;
    return v;
}
void update(int k, int l, int r, int x, int v) {
    if (l == x and r == x) {
        t[k] = v;
        return;
    }
    if (l > x or r < x) return;
    int m = (l + r) >> 1;
    update(k << 1, l, m, x, v);
    update((k << 1) + 1, m + 1, r, x, v);
    t[k] = max(t[k << 1], t[(k << 1) + 1]);
}
int get(int k, int l, int r, int x, int y) {
    if (l > y or r < x) return 0;
    if (l >= x and r <= y) return t[k];
    int m = (l + r) >> 1;
    return max(get(k << 1, l, m, x, y), get((k << 1) + 1, m + 1, r, x, y));
}
int getpoint(int x) {
    if (p[eu[x]] > p[ev[x]]) return eu[x];
    return ev[x];
}
void query1(int x, int v) {
    int id = getpoint(x - 1);
    update(1, 0, n - 1, pos[id], v);
}
int calc(int u, int w) {
    if (p[u] < p[w]) return 0;
    int res = 0;
    while (cn[u] != cn[w]) {
        res = max(res, get(1, 0, n - 1, pos[cr[cn[u]]], pos[u]));
        u = pre[cr[cn[u]]];
    }
    res = max(res, get(1, 0, n - 1, pos[w], pos[u]));
    return max(res, get(1, 0, n - 1, pos[w], pos[u]));
}
int query2(int u, int v) {
    if (u == v) return 0;
    int w = lca(u, v);
    w = Next[w];
    return max(calc(u, w), calc(v, w));
}
int main() {
    int test;
    scanf("%d", &test);
    while (test--) {
        clear_data();
        scanf("\n%d", &n);
        for (int i = 1; i < n; i++) {
            int u, v, w;
            scanf("%d %d %d", &u, &v, &w);
            data tmp;
            tmp.v = v;
            tmp.w = w;
            a[u].push_back(tmp);
            tmp.v = u;
```

```
            a[v].push_back(tmp);
            eu[i - 1] = u;
            ev[i - 1] = v;
            cost[i - 1] = w;
        }
        dfs(1, 0);
        hld(1);
        for (int i = 0; i < n - 1; i++) {
            int id = getpoint(i);
            update(1, 0, n - 1, pos[id], cost[i]);
        }
        while (scanf("%s", &s)) {
            if (s[0] == 'D') break;
            if (s[0] == 'C') {
                int x, v;
                scanf("%d %d\n", &x, &v);
                query1(x, v);
            }
            else {
                int u, v;
                scanf("%d %d\n", &u, &v);
                printf("%d\n", query2(u, v));
            }
        }
    }
}
```

Persitent Segment Tree

```
struct node {
    int sum;
    node *lc, *rc;
    node() {
        sum = 0;
        lc = NULL;
        rc = NULL;
    }
};
node* init(int l, int r) {
    node *t = new node();
    if (l == r) return t;
    int m = (l + r) >> 1;
    t->lc = init(l, m);
    t->rc = init(m + 1, r);
    return t;
}
node* update(node* k, int l, int r, int x) {
    if (l > x or r < x) return k;
    node* t = new node();
    if (l == x and r == x) {
        t->sum = 1;
        return t;
    }
    int m = (l + r) >> 1;
    t->lc = update(k->lc, l, m, x);
    t->rc = update(k->rc, m + 1, r, x);
    t->sum = t->lc->sum + t->rc->sum;
    return t;
}
int get(node* k, int l, int r, int x, int y) {
    if (l > y or r < x) return 0;
    if (l >= x and r <= y) return k->sum;
    int m = (l + r) >> 1;
    return get(k->lc, l, m, x, y) + get(k->rc, m + 1, r, x, y);
}
```

Diện tích n hình chữ nhật (Sweep-line technique)

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 8;
const int M = 3e5 + 8;
struct data {
    int x, l, r, t;
};
vector<data> a;
int t[M<<2], cnt[M<<2], f[M<<2];
bool cmp(const data &A, const data &B) {
    return A.x < B.x;
}
void init(int k, int l, int r) {
    cnt[k] = r - l + 1;
    if (l == r) return;
    int m = (l + r) >> 1;
    init(k << 1, l, m);
    init(k << 1 ^ 1, m + 1, r);
}
void update(int k, int l, int r, int x, int y, int v) {
    int k1 = k<<1;
    int k2 = k<<1^1;
    if (f[k] != 0) {
        t[k] += f[k];
        if (l != r) {
            f[k1] += f[k];
            f[k2] += f[k];
        }
        f[k] = 0;
    }
    if (l > y or r < x) return;
    if (l >= x and r <= y) {
```

```
        t[k] += v;
        if (l != r) {
            f[k1] += v;
            f[k2] += v;
        }
        return;
    }
    int m = (l + r) >> 1;
    update(k1, l, m, x, y, v);
    update(k2, m + 1, r, x, y, v);
    t[k] = min(t[k1], t[k2]);
    if (t[k1] < t[k2]) cnt[k] = cnt[k1];
    else if (t[k1] > t[k2]) cnt[k] = cnt[k2];
    else cnt[k] = cnt[k1] + cnt[k2];
}
void get(int k, int l, int r) {
    if (f[k] != 0) {
        t[k] += f[k];
        if (l != r) {
            f[k<<1] += f[k];
            f[k<<1^1] += f[k];
        }
        f[k] = 0;
    }
}
int main() {
    int n; scanf("%d", &n);
    for (int i = 1; i <= n; i++) {
        int x1, y1, x2, y2;
        scanf("%d %d %d %d", &x1, &y1, &x2, &y2);
        data tmp;
        tmp.l = y1;
        tmp.r = y2;
        tmp.x = x1;
        tmp.t = 1;
        a.push_back(tmp);
        tmp.x = x2;
        tmp.t = -1;
        a.push_back(tmp);
    }
    sort(a.begin(), a.end(), cmp);
    int px = 0, res = 0;
    init(1, 0, M - 1);
    for (int i = 0; i < n*2; i++) {
        int py = M;
        get(1, 0, M - 1);
        if (t[1] == 0) py -= cnt[1];
        res += (a[i].x - px) * py;
        data tmp = a[i];
        update(1, 0, M - 1, a[i].l, a[i].r - 1, a[i].t);
        px = a[i].x;
    }
    cout << res;
}
```

SCPC3 – 2017

```
#include <bits/stdc++.h>
using namespace std;
const int N = 200;
const int M = N * N;
int n, m;
vector<pair<int, int> > row[M], col[M];
int a[N][N], r[N][N], c[N][N], rowTrv[M], colTrv[M];
vector<int> rowList, colList;
bool dfs(int odd, int u, int type) {
    if (!odd) {
        rowTrv[u] = type;
        for (int i = 0; i < row[u].size(); i++) {
            int v = row[u][i].first;
            int w = row[u][i].second;
            if (colTrv[v] != -1 and w == 0) return false;
            if (colTrv[v] == -1) {
                if (!dfs(odd ^ 1, v, w ^ 1 ^ type)) return false;
            }
        }
        return true;
    }
    else {
        colTrv[u] = type;
        for (int i = 0; i < col[u].size(); i++) {
            int v = col[u][i].first;
            int w = col[u][i].second;
            if (rowTrv[v] != -1 and w == 0) return false;
            if (rowTrv[v] == -1) {
                if (!dfs(odd ^ 1, v, w ^ 1 ^ type)) return false;
            }
        }
        return true;
    }
}
bool calc() {
    for (int i = 0; i < rowList.size(); i++) {
        if (rowTrv[rowList[i]] == -1) {
            vector<int> rowBackup;
            vector<int> colBackup;
            for (int i = 0; i < rowList.size(); i++)
rowBackup.push_back(rowTrv[rowList[i]]);
            for (int i = 0; i < colList.size(); i++)
colBackup.push_back(colTrv[colList[i]]);
            rowTrv[rowList[i]] = 0;
```

```
            if (!dfs(0, rowList[i], 0)) {
                for (int i = 0; i < rowList.size(); i++) rowTrv[rowList[i]] =
rowBackup[i];
                for (int i = 0; i < colList.size(); i++) colTrv[colList[i]] =
colBackup[i];
                rowTrv[rowList[i]] = 1;
                if (!dfs(0, rowList[i], 1)) return false;
            }
        }
    }
    return true;
}
void printRes(vector<int> &A, int B[], char C) {
    for (int i = 0; i < A.size(); i++) {
        if (B[A[i]] == 1) {
            int x = A[i] / 100;
            int y = A[i] % 100;
            cout << C;
            if (x < 10) cout << 0;
            cout << x;
            if (y < 10) cout << 0;
            cout << y << ' ';
        }
    }
    cout << endl;
}
void reset() {
    memset(rowTrv, 0, sizeof(rowTrv));
    memset(colTrv, 0, sizeof(colTrv));
    rowList.clear();
    colList.clear();
}
int main() {
    int test; cin >> test;
    for (int I = 1; I <= test; I++) {
        reset();
        cin >> n >> m;
        for (int i = 1; i <= n; i++) for (int j = 1; j <= m; j++) {
            cin >> a[i][j] >> r[i][j] >> c[i][j];
            row[i * 100 + r[i][j]].push_back(make_pair(j * 100 + c[i][j],
a[i][j]));
            rowList.push_back(i * 100 + r[i][j]);
            col[j * 100 + c[i][j]].push_back(make_pair(i * 100 + r[i][j],
a[i][j]));
            colList.push_back(j * 100 + c[i][j]);
        }
        cout << "Case #" << I << '\n';
        if (calc()) {
            printRes(rowList, rowTrv, 'R');
            printRes(colList, colTrv, 'C');
        }
        else cout << "Impossible\n";
    }
}
```

F – ACM Vietnam National 2017

Đề bài: Cho A, B, d, đếm số dãy tăng chặt độ dài k thoả mãn các phần tử nằm trong đoạn [A,B] và có đúng d chữ số khác nhau từ mọi số trong dãy.
$1 \le A \le B \le 10^{18}$; $2 \le k \le 10$; $0 \le d \le 10$.

Bước 1

Tính số lượng số trong khoảng [A, B] mà gồm các chữ số nằm đúng trong tập S (S là tập con của tập {0, 1, 2, ..., 9})

- Số số nằm trong khoảng [A, B] = (số số nằm trong [0, B]) - (số số trong [0, A]). Do đó khi quy hoạch động ta chỉ cần quan tâm đến cận trên của các số.
- Giống với các bài toán quy hoạch động chữ số, xuất phát từ số 0, ta lần lượt thêm các chữ số vào, và tính f(len, mask, lower, positive) với:
    - len là độ dài của số ta đang xây dựng.
    - mask là tập hợp các chữ số của số ta đang xây dựng.
    - lower = 1 nếu số ta đang xây dựng đã nhỏ hơn cận trên B, = 0 trong trường hợp ngược lại.
    - positive = 1 nếu số ta đang xây dựng đã lớn hơn 0.

Cài đặt:

```
f[0][0][0][0] = 1;
// Xuất phát từ số 0
for (int len = 0; len < độ dài số B; len++) {
    for (int mask = 0; mask < 1023; mask++) { // dùng bitmask lưu S.
        for (int lower = 0; lower < 1; lower++) {
            for (int positive = 0; positive = 1; positive++) {
                // Thêm 1 chữ số
                for (int new_digit = 0; new_digit < 10; new_digit++) {
                    // Đảm bảo <= cận trên
                    if (lower == 0 && new_digit > chữ số (len+1) của B)
                        continue;
                    // Tính mask2, lower2, positive2 là các giá trị của số
                    // mới sau khi thêm chữ số new_digit
                    int positive2 = positive || (new_digit > 0);
                    int lower2 = lower || (new_digit < chữ số (len+1) của
B).
                    int mask2 = mask;
                    if (positive2) mask2 |= 1<<new_digit;
                    f[len+1][mask2][lower2][positive2] +=
f[len][mask][lower][positive];
                }
            }
```

```
        }
    }
}
```

Bước 2

Với mỗi tập S, tính xem có bao nhiêu số trong [A, B] có S là tập con của tập các chữ số của nó. Bạn có thể giải phần này trong O(3^10) bằng cách duyệt mọi tập con.

Bước 3

Dùng tổ hợp để đếm số bộ k.

LCA Miscellaneous

```
int n, m, a, b; vector<vector<int>> adj;
vector<vector<int>> Table; vector<int> d, subtree;

void DFS(int z, int last) {
	if (z != 0) d[z] = d[last] + 1;
	for (auto t: adj[z]) {
		if (t == last) continue;
		DFS(t, z); Table[t].pub(z);
		subtree[z] += subtree[t];
	}
}

void Preprocess() {
	subtree.resize(n, 1); Table.resize(n);
	d.resize(n, 0); DFS(0, -1);
	for (int j=1; j<17; j++) {
		for (int i=0; i<n; i++) {
			if (Table[i].size() < j) continue;
			if (Table[Table[i][j-1]].size() < j) continue;
			Table[i].push_back(Table[Table[i][j-1]][j-1]);
		}
	}
}

int ancestor(int node, int dist) {
	if (dist == 0) return node;
	for (int i=16; i>=0; i--) {
		if (dist >= (1LL << i)) {
			return ancestor(Table[node][i], dist - (1LL << i));
		}
	}
}

int LCA(int x, int y) {
	if (x == y) return x;
	if (d[x] == d[y]) {
		int id = 0, Init = min(Table[x].size(), Table[y].size()-1);
		for (int i=Init; i>=0; i--) {
			if (Table[x][i] != Table[y][i]) {id = i; break;}
		}
		return LCA(Table[x][id], Table[y][id]);
	}
	if (d[x] < d[y]) {
		int mul = 1, id = 0;
		while (d[x] < d[y] - mul * 2) {
			mul *= 2; id++;
		}
		return LCA(x, Table[y][id]);
	}
	if (d[x] > d[y]) {
		int mul = 1, id = 0;
		while (d[y] < d[x] - mul * 2) {
			mul *= 2; id++;
		}
		return LCA(Table[x][id], y);
	}
}

int Dist(int x, int y) {
	if (x == y) return 0;
	if (d[x] == d[y]) {
		int id = 0, Init = min(Table[x].size(), Table[y].size()-1);
		for (int i=Init; i>=0; i--) {
			if (Table[x][i] != Table[y][i]) {id = i; break;}
		}
		return ((1LL << id)*2 + Dist(Table[x][id], Table[y][id]));
	}
	if (d[x] < d[y]) {
		int mul = 1, id = 0;
		while (d[x] < d[y] - mul * 2) {
			mul *= 2; id++;
		}
		return ((1LL << id) + Dist(x, Table[y][id]));
	}
	if (d[x] > d[y]) {
		int mul = 1, id = 0;
		while (d[y] < d[x] - mul * 2) {
			mul *= 2; id++;
		}
		return ((1LL << id) + Dist(Table[x][id], y));
	}
}
```

6. Python Tricks

Array fastinput

```
def readArray():
	line = input() + ' '
```

```
	a = []
	num = 0
	for c in line:
		if c == ' ':
			a.append(num)
			num = 0
		else:
			num = num * 10 + (ord(c) - 48)
	return a
```

7. Java Fast IO

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.Scanner;
import java.util.StringTokenizer;

public class Main
{
	static class FastReader
	{
		BufferedReader br;
		StringTokenizer st;

		public FastReader()
		{
			br = new BufferedReader(new
					InputStreamReader(System.in));
		}

		String next()
		{
			while (st == null || !st.hasMoreElements())
			{
				try
				{
					st = new StringTokenizer(br.readLine());
				}
				catch (IOException e)
				{
					e.printStackTrace();
				}
			}
			return st.nextToken();
		}

		int nextInt()
		{
			return Integer.parseInt(next());
		}

		long nextLong()
		{
			return Long.parseLong(next());
		}

		double nextDouble()
		{
			return Double.parseDouble(next());
		}

		String nextLine()
		{
			String str = "";
			try
			{
				str = br.readLine();
			}
			catch (IOException e)
			{
				e.printStackTrace();
			}
			return str;
		}
	}

	public static void main(String[] args)
	{
		FastReader s=new FastReader();
		int n = s.nextInt();
		int k = s.nextInt();
		int count = 0;
		while (n-- > 0)
		{
			int x = s.nextInt();
			if (x%k == 0)
			count++;
		}
		System.out.println(count);
	}
}
```

8. Ubuntu commands

```
g++ -o fileName -p -std=c++11 fileName.cpp
```

```
./fileName
```