ACM-ICPC Code Templates

太奇怪了，准备交一发暴力

Pengpenglang @ CUGB

2021 年 10 月 17 日

目　录

1 书写调试环境

1.1 常用模板

```cpp
/**
 * @author pengpenlang
 * @brief 外能头文件 + 多样例
 */
#include <bits/stdc++.h>
#pragma GCC optimize(2)  //开启 O2 编译
#pragma G++ optimize(2)
// #define ONLINE_JUDGE
#define endl "\n"
#define fi first
#define se second
#define pb push_back
#define all(x) x.begin(), x.end()
#define rep(i, x, y) for (auto i = (x); i != (y + 1); ++i)
#define dep(i, x, y) for (auto i = (x); i != (y - 1); --i)
#ifndef ONLINE_JUDGE
#define de(...) cout << '[' << #__VA_ARGS__ << "] = " << __VA_ARGS__ << endl;
#else
#define de(...)
#endif
using namespace std;
typedef long long ll;
typedef pair<int, int> pii;

void solve() {
    /* 处理每组样例 */
}

signed main() {
    ios::sync_with_stdio(false), cin.tie(0);
#ifndef ONLINE_JUDGE
    freopen("IO\\in.txt", "r", stdin);
    freopen("IO\\out.txt", "w", stdout);
    clock_t start, end;
    start = clock();
#endif
    solve();
#ifndef ONLINE_JUDGE
    end = clock();
    cout << endl
         << "Runtime: " << (double)(end - start) / CLOCKS_PER_SEC << "s\n";
#endif
    return 0;
}
```

1.2 所有宏和头文件

```cpp
/**
 * @author pengpenglang
 * @brief ACM 个人常用头文件与宏定义
 */
```

```cpp
#include <algorithm>
#include <bitset>
#include <cctype>
#include <climits>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <functional>
#include <iomanip>
#include <iostream>
#include <map>
#include <queue>
#include <set>
#include <stack>
#include <string>
#include <vector>
using namespace std;
#pragma GCC optimize(2)  //开启 o2 编译
#pragma G++ optimize(2)
#pragma GCC optimize(3, "Ofast", "inline")  //开启 o3 编译
#pragma G++ optimize(3, "Ofast", "inline")
// #define ONLINE_JUDGE
#define endl "\n"
#define fi first
#define se second
#define pb push_back
#define all(x) x.begin(), x.end()
#define fastio ios::sync_with_stdio(false), cin.tie(0);
#define rep(i, x, y) for (auto i = (x); i != (y + 1); ++i)
#define dep(i, x, y) for (auto i = (x); i != (y - 1); --i)
#ifndef ONLINE_JUDGE
#define de(...) cout << '[' << #__VA_ARGS__ << "] = " << __VA_ARGS__ << endl;
#else
#define de(...)
#endif
typedef long long ll;
typedef unsigned long long ull;
typedef pair<int, int> pii;
typedef vector<int> vi;
const int inf = 0x3f3f3f3f, mod = 1e9 + 7;
const int dir[][2]={{0,-1},{1,0},{0,1},{-1,0},{1,-1},{1,1},{-1,1},{-1,-1}};//上右下左、右上、右下、左下、左上
const double eps = 1e-8;
inline int sgn(double x) {  //和 0 比大于返 1 等于返 0 小于返-1
    return (x > eps) - (x < -eps);
}
```

1.3 程序对拍模板

1.3.1 对拍脚本

```bat
@echo off
:loop
    rand.exe %random% > data.in
    std.exe < data.in > std.out
```

```
    my.exe < data.in > my.out
    fc my.out std.out
if not errorlevel 1 goto loop
pause
goto loop
```

1.3.2 造随机数

```cpp
/**
 * @author pengpenglang
 * @brief 利用 random 造随机样例
 */
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const ll a = 0, b = 1e5;  //规定生成随机数的范围

signed main(int argc, char *argv[]) {
    stringstream ss;
    ll seed = time(NULL);
    if (argc > 1) {  //如果传入了参数
        ss.clear();
        ss << argv[1];
        ss >> seed;  //把参数转换成整数赋值给 seed
    }
    //rand_max=32767
    auto random = [] {  //加强随机数范围，生成数在 [a,b]
        return a + rand() * rand() % (b - a + 1);
    };
    srand(seed);
    //以上为随机数初始化，请勿修改
    //下面利用利用 rand() 或者自定义的 random() 生成随机数
    cout << 1 << endl;  //单组循环、测试
    int len = random();
    string s = "";
    for (int i = 0; i < len; ++i)
        s += (char)(rand() % 25 + 'a');
    cout << s << endl;
    return 0;
}
```

1.3.3 使用方法

文件位置关系:

duipai

|– check.bat $\Longrightarrow$ 利用随机数对比 std 与自己程序的 exe 文件输出的对拍脚本

|– data.in $\Longrightarrow$ 保存每次对拍的随机样例

|– my.cpp $\Longrightarrow$ 自己程序的源代码

|– my.exe $\Longrightarrow$ 自己程序编译的 exe

|– my.out $\Longrightarrow$ 保存每次对拍自己程序的输出

|– rand.cpp $\Longrightarrow$ 造随机数程序的源代码

|– rand.exe $\Longrightarrow$ 造随机数程序编译的 exe

|– std.cpp $\Longrightarrow$ std 程序的源代码

|– std.exe $\Longrightarrow$ std 程序编译的 exe

'– std.out ⟹ 保存每次对拍 std 程序的输出
造随机数源码写好一并和自己的源码和 std 源码都编译生成.exe 文件，然后终端运行对拍脚本。系统自动对拍直至发现错误样例停止，打开 deta.in 文件查看出错样例，停止对拍快捷键 crtl+c

2 STL 和 fastIO

2.1 快读

2.1.1 整数 getchar 版

```
/**
 * @brief getchar 版快读支持所有整数类型
 */
inline _Tp read(_Tp &x) {
    char ch = getchar(), sgn = 0;
    x = 0;
    while (ch ^ '-' && !isdigit(ch)) ch = getchar();
    if (ch == '-') ch = getchar(), sgn = 1;
    while (isdigit(ch)) x = x * 10 + ch - '0', ch = getchar();
    if (sgn) x = -x;
    return x;
}

read(a);
```

2.1.2 整数 fread 版

```
/**
 * @brief fread 版快读支持所有整数类型，超多数据时表现优异
 */
struct ios_in {
    inline char gc() {
        const int MAXN = 1e5 + 100;  //读入字符串的最大长度（根据情况调整）
        static char buf[MAXN], *l, *r;
        return (l == r) && (r = (l = buf) + fread(buf, 1, MAXN, stdin), l == r)
↪   ? EOF : *l++;
    }
    template <typename _Tp>
    inline ios_in &operator>>(_Tp &x) {
        static char ch, sgn;
        for (sgn = 0, ch = gc(); !isdigit(ch); ch = gc()) {
            if (!~ch) return *this;
            sgn |= ch == '-';
        }
        for (x = 0; isdigit(ch); ch = gc())
            x = (x << 1) + (x << 3) + (ch ^ '0');
        sgn && (x = -x);
        return *this;
    }
} Cin;

Cin >> a;
```

2.1.3 浮点数 getchar 版

```cpp
/**
 * @brief getchar 版快读支持浮点数
 */
inline bool read_fl(double &num) {
    char in;
    double Dec = 0.1;
    bool IsN = false, IsD = false;
    in = getchar();
    if (in == EOF) return false;
    while (in != '-' && in != '.' && (in < '0' || in > '9'))
        in = getchar();
    if (in == '-') {
        IsN = true;
        num = 0;
    } else if (in == '.') {
        IsD = true;
        num = 0;
    } else
        num = in - '0';
    if (!IsD) {
        while (in = getchar(), in >= '0' && in <= '9') {
            num *= 10;
            num += in - '0';
        }
    }
    if (in != '.') {
        if (IsN) num = -num;
        return true;
    } else {
        while (in = getchar(), in >= '0' && in <= '9') {
            num += Dec * (in - '0');
            Dec *= 0.1;
        }
    }
    if (IsN) num = -num;
    return true;
}

read_fl(a);
```

2.1.4 对比说明

windows-gcc-x86 环境下测试连续存入 $[0, 1e6]$ 的数据:
cin(1.4s)　cin 关闭同步流 (0.796s)　scanf(0.311s)
read 版快读 (0.039s)　fread 版快读 (0.03s)
在多于 $1e5$ 的数据量时使用快读可以明显加快运行速度:
cin≪cin 关闭同步流 <scanf≪getchar 版快读 ≪fread 版快读

2.2 快写

2.2.1 整数 putchar 版

```cpp
/**
 * @brief putchar 版快写，仅支持 [INT_MIN,INT_MAX] 范围
 */
void write(int x) {
    if (x < 0) putchar('-'), x = -x;
    if (x > 9) write(x / 10);
    putchar(x % 10 + '0');
}

putchar(a);//不能输出回车
```

2.2.2 整数数组版

```cpp
/**
 * @brief putchar 进一步数组优化版快写，超多数据时表现优异
 */
struct ios_out {
    template <typename _Tp>
    inline void operator<<(_Tp &x) {
        const int MAXN = 1e3 + 100;  //存储数字的数组（根据情况调整）
        char F[MAXN];
        _Tp tmp = x >= 0 ? x : (putchar('-'), -x);
        int cnt = 0;
        while (tmp) {
            F[cnt++] = tmp % 10 + '0';
            tmp /= 10;
        }
        if (!cnt) {
            putchar('0');
            return;
        }
        while (cnt) putchar(F[--cnt]);
    }
} Cout;

Cout << a;  //不能输出回车
```

2.2.3 浮点数 putchar 版

```cpp
/**
 * @brief 浮点数 putchar 快写
 */

inline void dwrite(ll x) {  //用于输出整数部分
    if (x == 0) {
        putchar(48);
        return;
    }
    int bit[20], p = 0, i;
    for (; x; x /= 10) bit[++p] = x % 10;
    for (i = p; i > 0; --i) putchar(bit[i] + 48);
}
```

```cpp
inline void write(double x, int k = 6) {  //不加位数，默认保留小数点后 6 位小数
    static int n = pow(10, k);            //和读入相反，这里我无法直接转化小数部分，先乘以 n，就可以当做整数处理
    if (x == 0) {                         //x=0，保留的 k 位不断输出 0;
        putchar('0'), putchar('.');
        for (int i = 1; i <= k; ++i) putchar('0');
        return;
    }
    if (x < 0) putchar('-'), x = -x;  //负数
    ll y = (ll)(x * n) % n;
    x = (ll)x;            //y 表小数部分,x*n 之后把小数部分截去再对 n 取余就可以得到需要保留的小数部分.
    dwrite(x), putchar('.');  //输出整数部分和小数点
    int bit[10], p = 0, i;
    for (; p < k; y /= 10) bit[++p] = y % 10;  //必须严格按照 k 位保留，否则就 gg 了
    for (i = p; i > 0; i--) putchar(bit[i] + 48);
}
```

2.2.4 对比说明

windows-gcc-x86 环境下测试连续输出 $[0, 1e6]$ 的数据：
cout(0.558)　cout 关闭同步流 (0.601)　scanf(0.312)
putchar 版快写 (0.082)　数组版快写 (0.03s)
在多于 $1e5$ 的数据量时使用快读可以明显加快运行速度：
cout≈cout 关闭同步流 ≪scanf≪getchar 版快写 ≪ 数组版快写

2.3 容器

2.3.1 vector

2.3.2 queue

2.3.3 set

2.3.4 map

2.3.5 bitset

2.3.6 stack

3 数学

3.1 素数

3.1.1 埃氏筛

```
// O(n log log n) 筛出 MAXN 内所有素数
// notprime[i] = 0/1 0 为素数 1 为非素数
const int MAXN = 1000100;
bool notprime[MAXN] = {1, 1};          // 0/1 为非素数
void GetPrime() {
        for (int i = 2; i < MAXN; i++)
                if (!notprime[i] && i <= MAXN / i)          // 筛到 √n 为止
                        for (int j = i * i; j < MAXN; j += i)
                                notprime[j] = 1;
}
```

3.1.2 欧拉筛

```
// O(n) 得到欧拉函数 phi[]、素数表 prime[]、素数个数 tot
// 传入的 n 为函数定义域上界
const int MAXN = 100010;
bool vis[MAXN];
int tot, phi[MAXN], prime[MAXN];
void CalPhi(int n) {
        set(vis, 0); phi[1] = 1; tot = 0;
        for (int i = 2; i < n; i++) {
                if (!vis[i]) {
                        prime[tot++] = i;
                        phi[i] = i - 1;
                }
                for (int j = 0; j < tot; j++) {
                        if (i * prime[j] > n) break;
                        vis[i * prime[j]] = 1;
                        if (i % prime[j] == 0) {
                                phi[i * prime[j]] = phi[i] * prime[j];
                                break;
                        }
                        else phi[i * prime[j]] = phi[i] * (prime[j] - 1);
                }
        }
}
```

3.1.3 随机素数判定

```
// O(s log n) 内判定 2^63 内的数是不是素数，s 为测定次数
bool Miller_Rabin(ll n, int s) {
        if (n == 2) return 1;
        if (n < 2 || !(n & 1)) return 0;
        int t = 0; ll  x, y, u = n - 1;
        while ((u & 1) == 0) t++, u >>= 1;
        for (int i = 0; i < s; i++) {
                ll a = rand() % (n - 1) + 1;
                ll x = Pow(a, u, n);
                for (int j = 0; j < t; j++) {
                        ll y = Mul(x, x, n);
```

```
                if (y == 1 && x != 1 && x != n - 1) return 0;
                x = y;
            }
            if (x != 1) return 0;
        }
        return 1;
}
```

3.1.4 分解质因数

```
// 函数返回素因数个数
// 数组以 fact[i][0]^{fact[i][1]} 的形式保存第 i 个素因数
ll fact[100][2];
int getFactors(ll x) {
        int cnt = 0;
        for (int i = 0; prime[i] <= x / prime[i]; i++) {
                fact[cnt][1] = 0;
                if (x % prime[i] == 0 ) {
                        fact[cnt][0] = prime[i];
                        while (x % prime[i] == 0) {
                                fact[cnt][1]++;
                                x /= prime[i];
                        }
                        cnt++;
                }
        }
        if (x != 1) {
                fact[cnt][0] = x;
                fact[cnt++][1] = 1;
        }
        return cnt;
}
```

3.2 欧拉函数

3.2.1 求一个数的欧拉函数

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

3.2.2 筛法求欧拉函数

```
const int N = 10001;
int phi[N] = {0, 1};
void CalEuler() {
        for (int i = 2; i < N; i++)
                if (!phi[i]) for (int j = i; j < N; j += i) {
                                if (!phi[j]) phi[j] = j;
```

```
                                phi[j] = phi[j] / i * (i - 1);
                        }
}
```

3.3 扩展欧几里得-乘法逆元

3.3.1 扩展欧几里得

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

3.3.2 求 ax+by=c 的解

```
// 引用返回通解：X = x + k * dx, Y = y–k * dy
// 引用返回的 x 是最小非负整数解，方程无解函数返回 0
#define Mod(a,b) (((a)%(b)+(b))%(b))
bool solve(ll a, ll b, ll c, ll &x, ll &y, ll &dx, ll &dy) {
        if (a == 0 && b == 0) return 0;
        ll d, x0, y0; exgcd(a, b, d, x0, y0);
        if (c % d != 0) return 0;
        dx = b / d; dy = a / d;
        x = Mod(x0 * c / d, dx); y = (c - a * x) / b;
// y = Mod(y0 * c / d, dy); x = (c - b * y) / a;
        return 1;
}
```

3.3.3 乘法逆元

```
// 利用 exgcd 求 a 在模 m 下的逆元，需要保证 gcd(a, m) == 1.
ll inv(ll a, ll m) {
        ll x, y, d; exgcd(a, m, d, x, y);
        return d == 1 ? (x + m) % m : -1;
}
// a < m 且 m 为素数时，有以下两种求法
ll inv(ll a, ll m) {
        return a == 1 ? 1 : inv(m % a, m) * (m - m / a) % m;
}
ll inv(ll a, ll m) {
        return Pow(a, m - 2, m);
}
```

3.4 模线性方程组

3.4.1 中国剩余定理

```
// X = r[i]%m[i]，要求 m[i] 两两互质
// 引用返回通解 X = re + k * mo
void crt(ll r[], ll m[], ll n, ll &re, ll &mo) {
        mo = 1, re = 0;
        for (int i = 0; i < n; i++) mo *= m[i];
        for (int i = 0; i < n; i++) {
                ll x, y, d, tm = mo / m[i];
                exgcd(tm, m[i], d, x, y);
                re = (re + tm * x * r[i]) % mo;
```

```
        } re = (re + mo) % mo;
}
```

3.4.2 一般模线性方程组

```
// X = r[i]%m[i], m[i] 可以不两两互质
// 引用返回通解 X = re + k * mo, 函数返回是否有解
bool excrt(ll r[], ll m[], ll n, ll &re, ll &mo) {
        ll x, y, d; mo = m[0], re = r[0];
        for (int i = 1; i < n; i++) {
                exgcd(mo, m[i], d, x, y);
                if ((r[i] - re) % d != 0) return 0;
                x = (r[i] - re) / d * x % (m[i] / d);
                re += x * mo;
                mo = mo / d * m[i];
                re %= mo;
        } re = (re + mo) % mo;
        return 1;
}
```

3.5 组合数学

3.5.1 一般组合数

```
// 0 ≤ m ≤ n ≤ 1000
const int maxn = 1010;
ll C[maxn][maxn];
void CalComb() {
        C[0][0] = 1;
        for (int i = 1; i < maxn; i++) {
                C[i][0] = 1;
                for (int j = 1; j <= i; j++)
                        C[i][j] = (C[i - 1][j - 1] + C[i - 1][j]) % mod;
        }
}

// 0 ≤ m ≤ n ≤ 10^5, 模 p 为素数
const int maxn = 100010;
ll f[maxn];
void CalFact() {
        f[0] = 1;
        for (int i = 1; i < maxn; i++)
                f[i] = (f[i - 1] * i) % mod;
}
ll C(int n, int m) {
        return f[n] * inv(f[m] * f[n - m] % mod, mod) % mod;
}
```

3.5.2 Lucas 定理

```
// 1 ≤ n,m ≤ 10^9, 1 < p < 10^5, p 是素数
const int maxp = 100010;
ll f[maxp];
void CalFact(ll p) {
        f[0] = 1;
        for (int i = 1; i <= p; i++)
```

```
		f[i] = (f[i - 1] * i) % p;
}
ll Lucas(ll n, ll m, ll p) {
	ll ret = 1;
	while (n && m) {
		ll a = n % p, b = m % p;
		if (a < b) return 0;
		ret = (ret * f[a] * Pow(f[b] * f[a - b] % p, p - 2, p)) % p;
		n /= p; m /= p;
	}
	return ret;
}
```

3.5.3 大组合数

```
// 0 ≤ n ≤ 10^9, 0 ≤ m ≤ 10^4, 1 ≤ k ≤ 10^9 + 7
vector<int> v;
int dp[110];
ll Cal(int l, int r, int k, int dis) {
	ll res = 1;
	for (int i = l; i <= r; i++) {
		int t = i;
		for (int j = 0; j < v.size(); j++) {
			int y = v[j];
			while (t % y == 0) {
				dp[j] += dis;
				t /= y;
			}
		}
		res = res * (ll)t % k;
	}
	return res;
}
ll Comb(int n, int m, int k) {
	set(dp, 0); v.clear(); int tmp = k;
	for (int i = 2; i * i <= tmp; i++) {
		if (tmp % i == 0) {
			int num = 0;
			while (tmp % i == 0) {
				tmp /= i;
				num++;
			}
			v.pb(i);
		}
	} if (tmp != 1) v.pb(tmp);
	ll ans = Cal(n - m + 1, n, k, 1);
	for (int j = 0; j < v.size(); j++) {
		ans = ans * Pow(v[j], dp[j], k) % k;
	}
	ans = ans * inv(Cal(2, m, k, -1), k) % k;
	return ans;
}
```

3.5.4 Polya 定理

```
// 推论：一共 n 个置换，第 i 个置换的循环节个数为 gcd(i,n)
// N*N 的正方形格子，c^{n^2} + 2c^{\frac{n^2+3}{4}} + c^{\frac{n^2+1}{2}} + 2c^{n\frac{n+1}{2}} + 2c^{\frac{n(n+1)}{2}}
// 正六面体，(m^8 + 17m^4 + 6m^2)/24
// 正四面体，(m^4 + 11m^2)/12
// 长度为 n 的项链串用 c 种颜色染
ll solve(int c, int n) {
        if (n == 0) return 0;
        ll ans = 0;
        for (int i = 1; i <= n; i++)
                ans += Pow(c, __gcd(i, n));
        if (n & 1)
                ans += n * Pow(c, n + 1 >> 1);
        else
                ans += n / 2 * (1 + c) * Pow(c, n >> 1);
        return ans / n / 2;
}
```

3.6 快速乘 + 快速幂

```
ll Mul(ll a, ll b, ll mod) {
        ll t = 0;
        for (; b; b >>= 1, a = (a << 1) % mod)
                if (b & 1) t = (t + a) % mod;
        return t;
}

ll Pow(ll a, ll n, ll mod) {
    ll t = 1;
    for (; n; n >>= 1, a = (a * a % mod))
        if (n & 1) t = (t * a % mod);
    return t;
}
```

3.7 莫比乌斯反演

3.7.1 莫比乌斯

```
// F(n) = \sum_{d|n} f(d) \Rightarrow f(n) = \sum_{d|n} \mu(d)F(\frac{n}{d})
// F(n) = \sum_{n|d} f(d) \Rightarrow f(n) = \sum_{n|d} \mu(\frac{d}{n})F(d)
long long ans;
const int MAXN = 1e5 + 1;
int n, x, prime[MAXN], tot, mu[MAXN];
bool check[MAXN];
void calmu() {
    mu[1] = 1;
    for (int i = 2; i < MAXN; i++) {
        if (!check[i]) {
            prime[tot++] = i;
            mu[i] = -1;
        }
        for (int j = 0; j < tot; j++) {
            if (i * prime[j] >= MAXN) break;
            check[i * prime[j]] = true;
```

```
                if (i % prime[j] == 0) {
                    mu[i * prime[j]] = 0;
                    break;
                } else {
                    mu[i * prime[j]] = -mu[i];
                }
            }
        }
}
```

3.7.2 n 个数中互质数对数

```
// 有 n 个数 (n ≤ 10^5), 问这 n 个数中互质的数的对数
#include <cstdio>
#include <cstring>
#include <cstdlib>
using namespace std;
long long ans;
const int MAXN = 1e5 + 1;
int n, x, prime[MAXN], _max, b[MAXN], tot, mu[MAXN];
bool check[MAXN];
void calmu() {
        mu[1] = 1;
        for (int i = 2; i < MAXN; i++) {
                if (!check[i]) {
                        prime[tot++] = i;
                        mu[i] = -1;
                }
                for (int j = 0; j < tot; j++) {
                        if (i * prime[j] >= MAXN) break;
                        check[i * prime[j]] = true;
                        if (i % prime[j] == 0) {
                                mu[i * prime[j]] = 0;
                                break;
                        } else {
                                mu[i * prime[j]] = -mu[i];
                        }
                }
        }
}
int main() {
        calmu();
        while (scanf("%d", &n) == 1) {
                memset(b, 0, sizeof(b));
                _max = 0; ans = 0;
                for (int i = 0; i < n; i++) {
                        scanf("%d", &x);
                        if (x > _max) _max = x;
                        b[x]++;
                }
                int cnt;
                for (int i = 1; i <= _max; i++) {
                        cnt = 0;
                        for (long long j = i; j <= _max; j += i)
                                cnt += b[j];
```

```
                ans += 1LL * mu[i] * cnt * cnt;
            }
            printf("%lld\n", (ans - b[1]) / 2);
        }
        return 0;
}
```

3.7.3 VisibleTrees

```
// gcd(x,y)==1 的对数 x ≤ n,y ≤ m
int main() {
    calmu();
    int n, m;
    scanf("%d %d", &n, &m);
    if (n < m) swap(n, m);
    ll ans = 0;
    for (int i = 1; i <= m; ++i) {
        ans += (ll)mu[i] * (n / i) * (m / i);
    }
    printf("%lld\n", ans);
    return 0;
}
```

3.8 其他

3.8.1 Josephus 问题

```
#include <iostream>
using namespace std;
int main() {
        int num, m, r
        while (cin >> num >> m) {
                r = 0;
                for (int k = 1; k <= num; ++k)
                        r = (r + m) % k;
                cout << r + 1 << endl;
        }
        return 0;
}
```

3.8.2 数位问题

```
// n^n 最左边一位数
int leftmost(int n) {
        double m = n * log10((double)n);
        double g = m - (long long)m;
        g = pow(10.0, g);
        return (int)g;
}

// n! 位数
int count(ll n) {
        return n == 1 ? 1 : (int)ceil(0.5 * log10(2 * M_PI * n) + n * log10(n)
↪   - n * log10(M_E));
}
```

3.9 相关公式

约数定理：若 $n=\prod_{i=1}^{k}p_i^{a_i}$，则

1. 约数个数 $f(n)=\prod_{i=1}^{k}(a_i+1)$
2. 约数和 $g(n)=\prod_{i=1}^{k}(\sum_{j=0}^{a_i}p_i^j)$

小于 n 且互素的数之和为 $n\varphi(n)/2$

若 $gcd(n,i)=1$，则 $gcd(n,n-i)=1(1\le i\le n)$

错排公式：$D(n)=(n-1)(D(n-2)+D(n-1))=\sum_{i=2}^{n}\frac{(-1)^k n!}{k!}=[\frac{n!}{e}+0.5]$

威尔逊定理：$p\ is\ prime\ \Rightarrow (p-1)!\equiv -1(mod\ p)$

欧拉定理：$gcd(a,n)=1\Rightarrow a^{\varphi(n)}\equiv 1(mod\ n)$

欧拉定理推广：$gcd(n,p)=1\Rightarrow a^n\equiv a^{n\%\varphi(p)}(mod\ p)$

素数定理：对于不大于 n 的素数个数 $\pi(n)$，$\lim\limits_{n\to\infty}\pi(n)=\frac{n}{\ln n}$

位数公式：正整数 x 的位数 $N=log10(n)+1$

斯特灵公式 $n!\approx\sqrt{2\pi n}(\frac{n}{e})^n$

设 $a>1,m,n>0$, 则 $gcd(a^m-1,a^n-1)=a^{gcd(m,n)}-1$

设 $a>b,gcd(a,b)=1$, 则 $gcd(a^m-b^m,a^n-b^n)=a^{gcd(m,n)}-b^{gcd(m,n)}$

$$G=gcd(C_n^1,C_n^2,...,C_n^{n-1})=\begin{cases}n, & n \text{ is prime}\\ 1, & n \text{ has multy prime factors}\\ p, & n \text{ has single prime factor } p\end{cases}$$

$gcd(Fib(m),Fib(n))=Fib(gcd(m,n))$

若 $gcd(m,n)=1$, 则:

1. 最大不能组合的数为 $m*n-m-n$
2. 不能组合数个数 $N=\frac{(m-1)(n-1)}{2}$

$(n+1)lcm(C_n^0,C_n^1,...,C_n^{n-1},C_n^n)=lcm(1,2,...,n+1)$

若 p 为素数，则 $(x+y+...+w)^p\equiv x^p+y^p+...+w^p(mod\ p)$

卡特兰数：1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012

$h(0)=h(1)=1,h(n)=\frac{(4n-2)h(n-1)}{n+1}=\frac{C_{2n}^n}{n+1}=C_{2n}^n-C_{2n}^{n-1}$

$$a_{n+m}=\sum_{i=0}^{m-1}b_ia_{n+i}\Rightarrow\begin{pmatrix}a_{n+m}\\ a_{n+m-1}\\ \vdots\\ a_{n+1}\end{pmatrix}=\begin{pmatrix}b_{m-1} & \cdots & b_1 & b_0\\ 1 & \cdots & 0 & 0\\ \vdots & \ddots & \vdots & \vdots\\ 0 & \cdots & 1 & 0\end{pmatrix}\begin{pmatrix}a_{n+m-1}\\ a_{n+m-2}\\ \vdots\\ a_n\end{pmatrix}$$

$$a_{n+m}=\sum_{i=0}^{m-1}b_ia_{n+i}+c\Rightarrow\begin{pmatrix}a_{n+m}\\ a_{n+m-1}\\ \vdots\\ a_{n+1}\\ 1\end{pmatrix}=\begin{pmatrix}b_{m-1} & \cdots & b_1 & b_0 & c\\ 1 & \cdots & 0 & 0 & 0\\ \vdots & \ddots & \vdots & \vdots & \vdots\\ 0 & \cdots & 1 & 0 & 0\\ 0 & \cdots & 0 & 0 & 1\end{pmatrix}\begin{pmatrix}a_{n+m-1}\\ a_{n+m-2}\\ \vdots\\ a_n\\ 1\end{pmatrix}$$

4 数据结构

4.1 树状数组

```
// O(log n) 查询和修改数组的前缀和
// 注意下标应从 1 开始 n 是全局变量
int bit[N], n;
int sum(int i){
        int s = 0;
        while(i){
                s += bit[i];
                i -= i&-i;
        }
        return s;
}
void add(int i, int x){
        while(i<=n){
                bit[i] += x;
                i += i&-i;
        }
}
```

4.2 线段树

4.2.1 声明

```
#define lson rt<<1                              // 左儿子
#define rson rt<<1|1                            // 右儿子
#define Lson l,m,lson                   // 左子树
#define Rson m+1,r,rson         // 右子树
void PushUp(int rt);                            // 用 lson 和 rson 更新 rt
void PushDown(int rt[, int m]);         // rt 的标记下移，m 为区间长度（若与标记有
    关）
void build(int l, int r, int rt);         // 以 rt 为根节点，对区间 [l, r] 建立线
    段树
void update([...,] int l, int r, int rt)          // rt[l, r] 内寻找目标并更新
int query(int L, int R, int l, int r, int rt) // rt-[l, r] 内查询 [L, R]
```

4.2.2 单点更新-区间查询

```
const int maxn = 50010;
int sum[maxn << 2];
void PushUp(int rt) {
        sum[rt] = sum[lson] + sum[rson];
}
void build(int l, int r, int rt) {
        if (l == r) {scanf("%d", &sum[rt]); return;}         // 建立的时候直接输
    入叶节点
        int m = (l + r) >> 1;
        build(Lson); build(Rson);
        PushUp(rt);
}
void update(int p, int add, int l, int r, int rt) {
        if (l == r) {sum[rt] += add; return;}
        int m = (l + r) >> 1;
        if (p <= m) update(p, add, Lson);
```

```
        else update(p, add, Rson);
        PushUp(rt);
}
int query(int L, int R, int l, int r, int rt) {
        if (L <= l && r <= R) {return sum[rt];}
        int m = (l + r) >> 1, s = 0;
        if (L <= m) s += query(L, R, Lson);
        if (m < R) s += query(L, R, Rson);
        return s;
}
```

4.2.3 区间更新-区间查询

```
// seg[rt] 用于存放懒惰标记，注意 PushDown 时标记的传递
const int maxn = 101010;
int seg[maxn << 2], sum[maxn << 2];
void PushUp(int rt) {
        sum[rt] = sum[lson] + sum[rson];
}
void PushDown(int rt, int m) {
        if (seg[rt] == 0) return;
        seg[lson] += seg[rt];
        seg[rson] += seg[rt];
        sum[lson] += seg[rt] * (m - (m >> 1));
        sum[rson] += seg[rt] * (m >> 1);
        seg[rt] = 0;
}
void build(int l, int r, int rt) {
        seg[rt] = 0;
        if (l == r) {scanf("%lld", &sum[rt]); return;}
        int m = (l + r) >> 1;
        build(Lson); build(Rson);
        PushUp(rt);
}
void update(int L, int R, int add, int l, int r, int rt) {
        if (L <= l && r <= R) {
                seg[rt] += add;
                sum[rt] += add * (r - l + 1);
                return;
        }
        PushDown(rt, r - l + 1);
        int m = (l + r) >> 1;
        if (L <= m) update(L, R, add, Lson);
        if (m < R) update(L, R, add, Rson);
        PushUp(rt);
}
int query(int L, int R, int l, int r, int rt) {
        if (L <= l && r <= R) return sum[rt];
        PushDown(rt, r - l + 1);
        int m = (l + r) >> 1, ret = 0;
        if (L <= m) ret += query(L, R, Lson);
        if (m < R) ret += query(L, R, Rson);
        return ret;
}
```

4.3 字典树

```
struct Node {
    char c;
    Node* next[26];
    Node(char cc) {
        c = cc;
        REP(i, 26)next[i] = NULL;
    }
    ~Node() {
        REP(i, 26) if (next[i] != NULL) {
            next[i]->~Node();
            delete next[i];
            next[i] = NULL;
        }
    }
    bool empty() {
        REP(i, 26)if (next[i])return 0;
        return 1;
    }
};

class Trie {
public:
    Node *rt;
    Trie() {
        rt = new Node('*');
    }
    ~Trie() {
        rt->~Node();
    }
    void insert(char s[]) {
        Node *p = rt;
        for (int i = 0; s[i]; i++) {
            int d = s[i] - 'A';
            if (!p->next[d])
                p->next[d] = new Node(s[i]);
            p = p->next[d];
        }
    }
    int find(char s[]) {
        Node *p = rt;
        for (int i = 0; s[i]; i++) {
            int d = s[i] - 'A';
            if (!p->next[d]) return 0;
            p = p->next[d];
        }
        return 1;
    }
    void remove(char s[]) {
        stack<Node*> st;
        Node *pp = rt;
        for (int i = 0; s[i]; i++) {
            int d = s[i] - 'A';
            if (!pp->next[d]) return;
```

```
                st.push(pp);
                pp = pp->next[d];
            }
            pp->~Node();
            while (!st.empty()) {
                Node *p = st.top(); st.pop();
                p->next[pp->c - 'A'] = NULL;
                pp = p;
                bool f = 1;
                REP(i, 26) if (p->next[i]) f = 0;
                if (f) {
                    p->~Node();
                    if (!st.empty())st.top()->next[p->c - 'A'] = NULL;
                }
                else break;
            }
            if (rt == NULL) rt = new Node('*');
        }
};
```

4.4 RMQ

```
const int MAXN = 200000 + 100;
int mmax[MAXN][30], mmin[MAXN][30];
int a[MAXN], n, k;

void init() {
    for (int i = 1; i <= n; i++) {
        mmax[i][0] = mmin[i][0] = a[i];
    }
    for (int j = 1; (1 << j) <= n; j++)
        for (int i = 1; i + (1 << j) - 1 <= n; i++) {
            mmax[i][j] = max(mmax[i][j - 1], mmax[i + (1 << (j - 1))][j - 1]);
            mmin[i][j] = min(mmin[i][j - 1], mmin[i + (1 << (j - 1))][j - 1]);
        }
}

// op=0/1 返回 [l,r] 最大/小值
int rmq(int l, int r, int op) {
    int k = 0;
    while ((1 << (k + 1)) <= r - l + 1) k++;
    if (op == 0) return max(mmax[l][k], mmax[r - (1 << k) + 1][k]);
    return min(mmin[l][k], mmin[r - (1 << k) + 1][k]);
}
```

5 图论

5.1 并查集

```
const int MAXN = 128;
int n, fa[MAXN], ra[MAXN];
void init() {
    for (int i = 0; i <= n; i++) {
        fa[i] = i; ra[i] = 0;
    }
```

```
}
int find(int x) {
    if (fa[x] != x) fa[x] = find(fa[x]);
    return fa[x];
}
void unite(int x, int y) {
    x = find(x); y = find(y); if (x == y) return;
    if (ra[x] < ra[y]) fa[x] = y;
    else {
        fa[y] = x; if (ra[x] == ra[y]) ra[x]++;
    }
}
bool same(int x, int y) {
    return find(x) == find(y);
}
```

5.2 最小生成树

5.2.1 Kruskal

```
vector<pair<int, PII> > G;
void add_edge(int u, int v, int d) {
    G.pb(mp(d, mp(u, v)));
}
int Kruskal(int n) {
    init(n);
    sort(G.begin(), G.end());
    int m = G.size();
    int num = 0, ret = 0;
    for (int i = 0; i < m; i++) {
        pair<int, PII> p = G[i];
        int x = p.Y.X;
        int y = p.Y.Y;
        int d = p.X;
        if (!same(x, y)) {
            unite(x, y);
            num++;
            ret += d;
        }
        if (num == n - 1) break;
    }
    return ret;
}
```

5.2.2 Prim

```
// 耗费矩阵 cost[][], 标号从 0 开始,0~n-1
// 返回最小生成树的权值, 返回-1 表示原图不连通
const int INF = 0x3f3f3f3f;
const int MAXN = 110;
bool vis[MAXN];
int lowc[MAXN];
int Prim(int cost[][MAXN], int n) {
        int ans = 0;
        set(vis, 0);
        vis[0] = 1;
```

```
        for (int i = 1; i < n; i++)
                lowc[i] = cost[0][i];
        for (int i = 1; i < n; i++) {
                int minc = INF;
                int p = -1;
                for (int j = 0; j < n; j++)
                        if (!vis[j] && minc > lowc[j]) {
                                minc = lowc[j];
                                p = j;
                        }
                if (minc == INF) return -1;
                vis[p] = 1;
                for (int j = 0; j < n; j++)
                        if (!vis[j] && lowc[j] > cost[p][j]) lowc[j] =
    ↪   cost[p][j];
        }
        return ans;
}
```

5.3 最短路

5.3.1 Dijkstra-邻接矩阵

```
// N 为点数最大值 求 s 到所有点的最短路
// 要求边权值为非负数 模板为有向边
// dis[x] 为起点到点 x 的最短路 inf 表示无法走到
// 记得初始化
const int N = 100;        // 点数最大值
const int INF = 0x3f3f3f3f;
int G[N][N], dis[N];
bool vis[N];
void init(int n) {
        set(G, 0x3f);
}
void add_edge(int u, int v, int w) {
        G[u][v] = min(G[u][v], w);
}
void Dijkstra(int s, int n) {
        set(vis, 0);
        set(dis, 0x3f);
        dis[s] = 0;
        for (int i = 0; i < n; i++) {
                int x, minDis = INF;
                for (int j = 0; j < n; j++) {
                        if (!vis[j] && dis[j] <= minDis) {
                                x = j;
                                minDis = dis[j];
                        }
                }
                vis[x] = 1;
                for (int j = 0; j < n; j++)
                        dis[j] = min(dis[j], dis[x] + G[x][j]);
        }
}
```

5.3.2 Dijkstra-邻接表数组

```cpp
// 点最大值: MAX_N  边最大值: MAX_E
// 求起点 s 到每个点 x 的最短路 dis[x]
const int MAX_N = "Edit";        // 点数最大值
const int MAX_E = "Edit";
const int INF = 0x3F3F3F3F;
int tot;
int Head[MAX_N], vis[MAX_N], dis[MAX_N];
int Next[MAX_E], To[MAX_E], W[MAX_E];
void init() {
        tot = 0;
        memset(Head, -1, sizeof(Head));
}
void add_edge(int u, int v, int d) {
        W[tot] = d;
        To[tot] = v;
        Next[tot] = Head[u];
        Head[u] = tot++;
}
void Dijkstra(int s, int n) {
        memset(vis, 0, sizeof(vis));
        memset(dis, 0x3F, sizeof(dis));
        dis[s] = 0;
        for (int i = 0; i < n; i++) {
                int x, min_dis = INF;
                for (int j = 0; j < n; j++) {
                        if (!vis[j] && dis[j] <= min_dis) {
                                x = j;
                                min_dis = dis[j];
                        }
                }
                vis[x] = 1;
                for (int j = Head[x]; j != -1; j = Next[j]) {
                        int y = To[j];
                        dis[y] = min(dis[y], dis[x] + W[j]);
                }
        }
}
```

5.3.3 Dijkstra-邻接表向量

```cpp
// MAXN: 点数最大值
// 求起点 s 到所有点 x 的最短路 dis[x]
// 记得初始化
const int MAXN = "Edit";
const int INF = 0x3F3F3F3F;
vector<int> G[MAXN];
vector<int> GW[MAXN];
bool vis[MAXN];
int dis[MAXN];
void init(int n) {
        for (int i = 0; i < n; i++) {
                G[i].clear();
                GW[i].clear();
        }
```

```
}
void add_edge(int u, int v, int w) {
        G[u].push_back(v);
        GW[u].push_back(w);
}
void Dijkstra(int s, int n) {
        memset(vis, false, sizeof(vis));
        memset(dis, 0x3F, sizeof(dis));
        dis[s] = 0;
        for (int i = 0; i < n; i++) {
                int x;
                int min_dis = INF;
                for (int j = 0; j < n; j++) {
                        if (!vis[j] && dis[j] <= min_dis) {
                                x = j;
                                min_dis = dis[j];
                        }
                }
                vis[x] = true;
                for (int j = 0; j < (int)G[x].size(); j++) {
                        int y = G[x][j];
                        int w = GW[x][j];
                        dis[y] = min(dis[y], dis[x] + w);
                }
        }
}
```

5.3.4 Dijkstra-优先队列

```
// pair< 权值, 点 >
// 记得初始化
const int MAXN = "Edit";
const int INF = 0x3F3F3F3F;
typedef pair<int, int> PII;
typedef vector<PII> VII;
VII G[MAXN];
int vis[MAXN], dis[MAXN];
void init(int n) {
        for (int i = 0; i < n; i++)
                G[i].clear();
}
void add_edge(int u, int v, int w) {
        G[u].push_back(make_pair(w, v));
}
void Dijkstra(int s, int n) {
        memset(vis, 0, sizeof(vis));
        memset(dis, 0x3F, sizeof(dis));
        dis[s] = 0;
        priority_queue<PII, VII, greater<PII> > q;
        q.push(make_pair(dis[s], s));
        while (!q.empty()) {
                PII t = q.top();
                int x = t.second;
                q.pop();
                if (vis[x]) continue;
```

```cpp
            vis[x] = 1;
            for (int i = 0; i < (int)G[x].size(); i++) {
                int y = G[x][i].second;
                int w = G[x][i].first;
                if (!vis[y] && dis[y] > dis[x] + w) {
                    dis[y] = dis[x] + w;
                    q.push(make_pair(dis[y], y));
                }
            }
        }
}
```

5.3.5 Bellman-Ford(可判负环)

```cpp
// 求出起点 s 到每个点 x 的最短路 dis[x]
// 存在负环返回 1 否则返回 0
// 记得初始化
const int MAX_N = "Edit";        // 点数最大值
const int MAX_E = "Edit";        // 边数最大值
const int INF = 0x3F3F3F3F;
int From[MAX_E], To[MAX_E], W[MAX_E];
int dis[MAX_N], tot;
void init() {tot = 0;}
void add_edge(int u, int v, int d) {
        From[tot] = u;
        To[tot] = v;
        W[tot++] = d;
}
bool Bellman_Ford(int s, int n) {
        memset(dis, 0x3F, sizeof(dis));
        dis[s] = 0;
        for (int k = 0; k < n - 1; k++) {
                bool relaxed = 0;
                for (int i = 0; i < tot; i++) {
                        int x = From[i], y = To[i];
                        if (dis[y] > dis[x] + W[i]) {
                                dis[y] = dis[x] + W[i];
                                relaxed = 1;
                        }
                }
                if (!relaxed) break;
        }
        for (int i = 0; i < tot; i++)
                if (dis[To[i]] > dis[From[i]] + W[i])
                        return 1;
        return 0;
}
```

5.3.6 SPFA

```cpp
// G[u] = mp(v, w)
// SPFA() 返回 0 表示存在负环
const int MAXN = "Edit";
const int INF = 0x3F3F3F3F;
vector<pair<int, int> > G[MAXN];
```

```
bool vis[MAXN];
int dis[MAXN];
int inqueue[MAXN];
void init(int n) {
        for (int i = 0; i < n; i++)
                G[i].clear();
}
void add_edge(int u, int v, int w) {
        G[u].push_back(make_pair(v, w));
}
bool SPFA(int s, int n) {
        memset(vis, 0, sizeof(vis));
        memset(dis, 0x3F, sizeof(dis));
        memset(inqueue, 0, sizeof(inqueue));
        dis[s] = 0;
        queue<int> q;        // 待优化的节点入队
        q.push(s);
        while (!q.empty()) {
                int x = q.front();
                q.pop();
                vis[x] = false;
                for (int i = 0; i < G[x].size(); i++) {
                        int y = G[x][i].first;
                        int w = G[x][i].second;
                        if (dis[y] > dis[x] + w) {
                                dis[y] = dis[x] + w;
                                if (!vis[y]) {
                                        q.push(y);
                                        vis[y] = true;
                                        if (++inqueue[y] >= n) return 0;
                                }
                        }
                }
        }
        return 1;
}
```

5.3.7 Floyd 算法

```
// O(n^3) 求出任意两点间最短路
const int MAXN = "Edit";
const int INF = 0x3F3F3F3F;
int G[MAXN][MAXN];
void init(int n) {
        memset(G, 0x3F, sizeof(G));
        for (int i = 0; i < n; i++)
                G[i][i] = 0;
}
void add_edge(int u, int v, int w) {
        G[u][v] = min(G[u][v], w);
}
void Floyd(int n) {
        for (int k = 0; k < n; k++)
                for (int i = 0; i < n; i++)
                        for (int j = 0; j < n; j++)
```

```
                        G[i][j] = min(G[i][j], G[i][k] + G[k][j]);
}
```

5.4 拓扑排序

5.4.1 邻接矩阵

```
// 存图前记得初始化
// Ans 存放拓排结果，G 为邻接矩阵，deg 为入度信息
// 排序成功返回 1，存在环返回 0
const int MAXN = "Edit";
int Ans[MAXN];                  // 存放拓扑排序结果
int G[MAXN][MAXN];          // 存放图信息
int deg[MAXN];                  // 存放点入度信息
void init() {
        memset(G, 0, sizeof(G));
        memset(deg, 0, sizeof(deg));
        memset(Ans, 0, sizeof(Ans));
}
void add_edge(int u, int v) {
        if (G[u][v]) return;
        G[u][v] = 1;
        deg[v]++;
}
bool Toposort(int n) {
        int tot = 0;
        queue<int> que;
        for (int i = 0; i < n; ++i)
                if (deg[i] == 0) que.push(i);
        while (!que.empty()) {
                int v = que.front(); que.pop();
                Ans[tot++] = v;
                for (int i = 0; i < n; ++i)
                        if (G[v][i] == 1)
                                if (--deg[t] == 0) que.push(t);
        }
        if (tot < n) return false;
        return true;
}
```

5.4.2 邻接表

```
// 存图前记得初始化
// Ans 排序结果，G 邻接表，deg 入度，map 用于判断重边
// 排序成功返回 1，存在环返回 0
const int MAXN = "Edit";
typedef pair<int, int> PII;
int Ans[MAXN];
vector<int> G[MAXN];
int deg[MAXN];
map<PII, bool> S;
void init(int n) {
        S.clear();
        for (int i = 0; i < n; i++)G[i].clear();
        memset(deg, 0, sizeof(deg));
        memset(Ans, 0, sizeof(Ans));
```

```
}
void add_edge(int u, int v) {
        if (S[make_pair(u, v)]) return;
        G[u].push_back(v);
        S[make_pair(u, v)] = 1;
        deg[v]++;
}
bool Toposort(int n) {
        int tot = 0; queue<int> que;
        for (int i = 0; i < n; ++i)
                if (deg[i] == 0) que.push(i);
        while (!que.empty()) {
                int v = que.front(); que.pop();
                Ans[tot++] = v;
                for (int i = 0; i < G[v].size(); ++i) {
                        int t = G[v][i];
                        if (--deg[t] == 0) que.push(t);
                }
        }
        if (tot < n) return false;
        return true;
}
```

5.5 欧拉回路

5.5.1 判定

定理 5.1. 无向图 G 存在欧拉通路的充要条件是：G 为连通图，并且 G 仅有两个奇度结点或无奇度结点。

推论 5.1. *(1)* 当 G 是仅有两个奇度结点的连通图时，G 的欧拉通路必以此两个结点为端点。*(2)* 当 G 时无奇度结点的连通图时，G 必有欧拉回路。*(3)* G 为欧拉图（存在欧拉回路）的充要条件是 G 为无奇度结点的连通图。

定理 5.2. 有向图 D 存在欧拉通路的充要条件是：D 为有向图，D 的基图连通，并且所有顶点的出度与入度都相等；或者除两个顶点外，其余顶点的出度与入度都相等，而这两个顶点中一个顶点的出度与入度只差为 *1*，另一个顶点的出度与入度之差为 *-1*。

推论 5.2. *(1)* 当 D 除出、入度之差为 *1*，*-1* 的两个顶点之外，其余顶点的出度与入度都相等时，D 的有向欧拉通路必以出、入度之差为 *1* 的顶点作为始点，以出、入度之差为 *-1* 的顶点作为终点。*(2)* 当 D 的所有顶点的出、入度都相等时，D 中存在有向欧拉回路。*(3)* 有向图 D 为有向欧拉图的充要条件是 D 的基图为连通图，并且所有顶点的出、入度都相等。

5.5.2 求解

```
#define MAXN 200
struct stack {
        int top, node[MAXN];
} s;

int G[MAXN][MAXN];          // 邻接矩阵
int n;          // 顶点个数

void dfs(int x) {
        int i;
```

```
        s.node[++s.top] = x;
        for (int i = 0; i < n; i++)
                if (G[i][x] > 0) {
                        G[i][x] = G[x][i] = 0;
                        dfs(i);
                        break;
                }
}
void Fleury(int x) {
        int i, b;
        s.node[s.top = 0] = x;
        while (s.top >= 0) {
                b = 0;
                for (int i = 0; i < n; i++)
                        if (G[s.node[s.top]][i] > 0) {
                                b = 1;
                                break;
                        }
                if (b == 0) {
                        printf("%d ", s.node[s.top] + 1);
                        s.top--;
                }
                else {
                        s.top--;
                        dfs(s.node[s.top + 1]);
                }
        }
        printf("\n");
}

int main() {
        int i, j;
        int m, s, t; // 边数，读入的边的起点和终点
        int degree, num, start; // 每个顶点的度、奇度顶点个数、欧拉回路的起点
        scanf("%d%d", &n, &m);
        set(G, 0);
        for (i = 0; i < m; i++) {
                scanf("%d%d", &s, &t)
                G[s - 1][t - 1] = G[t - 1][s - 1] = 1;
        }
        num = 0; start = 0;
        for (i = 0; i < n; i++) {
                degree = 0;
                for (j = 0; j < n; j++)
                        degree += G[i][j];
                if (degree & 1) {
                        start = i;
                        num++;
                }
        }
        if (num == 0 || num == 2) Fleury(start);
        else puts("No Euler path");
        return 0;
}
```

6 计算几何

6.1 定义

```cpp
#define eps 1e-8
#define pi M_PI
#define zero(x) ((fabs(x)<eps?1:0))
#define sgn(x) (fabs(x)<eps?0:((x)<0?-1:1))
#define mp make_pair
#define X first
#define Y second

struct point {
    double x, y;
    point(double a = 0, double b = 0) {x = a; y = b;}
    point operator - (const point& b) const {
        return point(x - b.x, y - b.y);
    }
    point operator + (const point &b) const {
        return point(x + b.x, y + b.y);
    }
    // 两点是否重合
    bool operator == (point& b) {
        return zero(x - b.x) && zero(y - b.y);
    }
    // 点积 (以原点为基准)
    double operator * (const point &b) const {
        return x * b.x + y * b.y;
    }
    // 叉积 (以原点为基准)
    double operator ^ (const point &b) const {
        return x * b.y - y * b.x;
    }
    // 绕 P 点逆时针旋转 a 弧度后的点
    point rotate(point b, double a) {
        double dx, dy; (*this - b).split(dx, dy);
        double tx = dx * cos(a) - dy * sin(a);
        double ty = dx * sin(a) + dy * cos(a);
        return point(tx, ty) + b;
    }
    // 点坐标分别赋值到 a 和 b
    void split(double &a, double &b) {
        a = x; b = y;
    }
};

struct line {
    point s, e;
    line() {}
    line(point ss, point ee) {s = ss; e = ee;}
};
```

6.2 位置关系

6.2.1 两点间距离

```cpp
double dist(point a, point b) {
    return sqrt((a - b) * (a - b));
}
```

6.2.2 直线与直线的交点

```cpp
// <0, *> 表示重合; <1, *> 表示平行; <2, P> 表示交点是 P;
pair<int, point> spoint(line l1, line l2) {
    point res = l1.s;
    if (sgn((l1.s - l1.e) ^ (l2.s - l2.e)) == 0)
        return mp(sgn((l1.s - l2.e) ^ (l2.s - l2.e)) != 0, res);
    double t = ((l1.s - l2.s) ^ (l2.s - l2.e)) / ((l1.s - l1.e) ^ (l2.s - l2.e));
    res.x += (l1.e.x - l1.s.x) * t;
    res.y += (l1.e.y - l1.s.y) * t;
    return mp(2, res);
}
```

6.2.3 判断线段与线段相交

```cpp
bool segxseg(line l1, line l2) {
    return
        max(l1.s.x, l1.e.x) >= min(l2.s.x, l2.e.x) &&
        max(l2.s.x, l2.e.x) >= min(l1.s.x, l1.e.x) &&
        max(l1.s.y, l1.e.y) >= min(l2.s.y, l2.e.y) &&
        max(l2.s.y, l2.e.y) >= min(l1.s.y, l1.e.y) &&
        sgn((l2.s - l1.e) ^ (l1.s - l1.e)) * sgn((l2.e-l1.e) ^ (l1.s - l1.e)) <= 0 &&
        sgn((l1.s - l2.e) ^ (l2.s - l2.e)) * sgn((l1.e-l2.e) ^ (l2.s - l2.e)) <= 0;
}
```

6.2.4 判断线段与直线相交

```cpp
bool segxline(line l1, line l2) {
        return sgn((l2.s - l1.e) ^ (l1.s - l1.e)) * sgn((l2.e-l1.e) ^ (l1.s - l1.e)) <= 0;
}
```

6.2.5 点到直线距离

```cpp
point pointtoline(point P, line L) {
        point res;
        double t = ((P - L.s) * (L.e-L.s)) / ((L.e-L.s) * (L.e-L.s));
        res.x = L.s.x + (L.e.x - L.s.x) * t;
        res.y = L.s.y + (L.e.y - L.s.y) * t;
        return dist(P, res);
}
```

6.2.6 点到线段距离

```cpp
point pointtosegment(point p, line l) {
        point res;
```

```
        double t = ((p - l.s) * (l.e-l.s)) / ((l.e-l.s) * (l.e-l.s));
        if (t >= 0 && t <= 1) {
                res.x = l.s.x + (l.e.x - l.s.x) * t;
                res.y = l.s.y + (l.e.y - l.s.y) * t;
        }
        else res = dist(p, l.s) < dist(p, l.e) ? l.s : l.e;
        return res;
}
```

6.2.7 点在线段上

```
bool PointOnSeg(point p, line l) {
        return
            sgn((l.s - p) ^ (l.e-p)) == 0 &&
            sgn((p.x - l.s.x) * (p.x - l.e.x)) <= 0 &&
            sgn((p.y - l.s.y) * (p.y - l.e.y)) <= 0;
}
```

6.3 多边形

6.3.1 多边形面积

```
double area(point p[], int n) {
    double res = 0;
    for (int i = 0; i < n; i++)
        res += (p[i] ^ p[(i + 1) % n]) / 2;
    return fabs(res);
}
```

6.3.2 点在凸多边形内

```
// 点形成一个凸包，而且按逆时针排序（如果是顺时针把里面的 <0 改为 >0)
// 点的编号 : [0,n)
// -1 : 点在凸多边形外
// 0  : 点在凸多边形边界上
// 1  : 点在凸多边形内
int PointInConvex(point a, point p[], int n) {
        for (int i = 0; i < n; i++) {
                if (sgn((p[i] - a) ^ (p[(i + 1) % n] - a)) < 0)
                        return -1;
                else if (PointOnSeg(a, line(p[i], p[(i + 1) % n])))
                        return 0;
        }
        return 1;
}
```

6.3.3 点在任意多边形内

```
// 射线法,poly[] 的顶点数要大于等于 3, 点的编号 0~n-1
// -1 : 点在凸多边形外
// 0  : 点在凸多边形边界上
// 1  : 点在凸多边形内
int PointInPoly(point p, point poly[], int n) {
        int cnt;
        line ray, side;
        cnt = 0;
```

```
        ray.s = p;
        ray.e.y = p.y;
        ray.e.x = -100000000000.0; // -INF, 注意取值防止越界
        for (int i = 0; i < n; i++) {
                side.s = poly[i];
                side.e = poly[(i + 1) % n];
                if (PointOnSeg(p, side))return 0;
                //如果平行轴则不考虑
                if (sgn(side.s.y - side.e.y) == 0)
                        continue;
                if (PointOnSeg(sid e.s, r ay)) {
                        if (sgn(side.s.y - side.e.y) > 0) cnt++;
                }
                else if (PointOnSeg(side.e, ray)) {
                        if (sgn(side.e.y - side.s.y) > 0) cnt++;
                }
                else if (segxseg(ray, side)) cnt++;
        }
        return cnt % 2 == 1 ? 1 : -1;
}
```

6.3.4 判断凸多边形

```
// 点可以是顺时针给出也可以是逆时针给出
// 点的编号 1~n-1
bool isconvex(point poly[], int n) {
        bool s[3];
        set(s, 0);
        for (int i = 0; i < n; i++) {
                s[sgn((poly[(i + 1) % n] - poly[i]) ^ (poly[(i + 2) % n] - poly[i]) ) + 1] = 1;
                if (s[0] && s[2]) return 0;
        }
        return 1;
}
```

6.3.5 小结

```
#include <stdlib.h>
#include <math.h>
#define MAXN 1000
#define offset 10000
#define eps 1e-8
#define zero(x) (((x)>0?(x):-(x))<eps)
#define _sign(x) ((x)>eps?1:((x)<-eps?2:0))
struct point{double x,y;};
struct line{point a,b;};

double xmult(point p1,point p2,point p0){
    return (p1.x-p0.x)*(p2.y-p0.y)-(p2.x-p0.x)*(p1.y-p0.y);
}

// 判定凸多边形，顶点按顺时针或逆时针给出，允许相邻边共线
int is_convex(int n,point* p){
    int i,s[3]={1,1,1};
```

```
    for (i=0;i<n&&s[1]|s[2];i++)
        s[_sign(xmult(p[(i+1)%n],p[(i+2)%n],p[i]))]=0;
    return s[1]|s[2];
}

// 判定凸多边形，顶点按顺时针或逆时针给出，不允许相邻边共线
int is_convex_v2(int n,point* p){
    int i,s[3]={1,1,1};
    for (i=0;i<n&&s[0]&&s[1]|s[2];i++)
        s[_sign(xmult(p[(i+1)%n],p[(i+2)%n],p[i]))]=0;
    return s[0]&&s[1]|s[2];
}

// 判点在凸多边形内或多边形边上，顶点按顺时针或逆时针给出
int inside_convex(point q,int n,point* p){
    int i,s[3]={1,1,1};
    for (i=0;i<n&&s[1]|s[2];i++)
        s[_sign(xmult(p[(i+1)%n],q,p[i]))]=0;
    return s[1]|s[2];
}

// 判点在凸多边形内，顶点按顺时针或逆时针给出，在多边形边上返回 0
int inside_convex_v2(point q,int n,point* p){
    int i,s[3]={1,1,1};
    for (i=0;i<n&&s[0]&&s[1]|s[2];i++)
        s[_sign(xmult(p[(i+1)%n],q,p[i]))]=0;
    return s[0]&&s[1]|s[2];
}

// 判点在任意多边形内，顶点按顺时针或逆时针给出
// on_edge 表示点在多边形边上时的返回值,offset 为多边形坐标上限
int inside_polygon(point q,int n,point* p,int on_edge=1){
    point q2;
    int i=0,count;
    while (i<n)
        for (count=i=0,q2.x=rand()+offset,q2.y=rand()+offset;i<n;i++)
            if
 ↪ (zero(xmult(q,p[i],p[(i+1)%n]))&&(p[i].x-q.x)*(p[(i+1)%n].x-q.x)<eps&&(p[i].y-q.y)*(p[(i+1)%n
                return on_edge;
            else if (zero(xmult(q,q2,p[i])))
                break;
            else if
 ↪ (xmult(q,p[i],q2)*xmult(q,p[(i+1)%n],q2)<-eps&&xmult(p[i],q,p[(i+1)%n])*xmult(p[i],q2,p[(i+1)
                count++;
    return count&1;
}

inline int opposite_side(point p1,point p2,point l1,point l2){
    return xmult(l1,p1,l2)*xmult(l1,p2,l2)<-eps;
}

inline int dot_online_in(point p,point l1,point l2){
    return
 ↪ zero(xmult(p,l1,l2))&&(l1.x-p.x)*(l2.x-p.x)<eps&&(l1.y-p.y)*(l2.y-p.y)<eps;
}
```

```
// 判线段在任意多边形内，顶点按顺时针或逆时针给出，与边界相交返回 1
int inside_polygon(point l1,point l2,int n,point* p){
    point t[MAXN],tt;
    int i,j,k=0;
    if (!inside_polygon(l1,n,p)||!inside_polygon(l2,n,p))
        return 0;
    for (i=0;i<n;i++)
        if
↪   (opposite_side(l1,l2,p[i],p[(i+1)%n])&&opposite_side(p[i],p[(i+1)%n],l1,l2))
            return 0;
        else if (dot_online_in(l1,p[i],p[(i+1)%n]))
            t[k++]=l1;
        else if (dot_online_in(l2,p[i],p[(i+1)%n]))
            t[k++]=l2;
        else if (dot_online_in(p[i],l1,l2))
            t[k++]=p[i];
    for (i=0;i<k;i++)
        for (j=i+1;j<k;j++){
            tt.x=(t[i].x+t[j].x)/2;
            tt.y=(t[i].y+t[j].y)/2;
            if (!inside_polygon(tt,n,p))
                return 0;
        }
    return 1;
}

point intersection(line u,line v){
    point ret=u.a;
    double t=((u.a.x-v.a.x)*(v.a.y-v.b.y)-(u.a.y-v.a.y)*(v.a.x-v.b.x))
            /((u.a.x-u.b.x)*(v.a.y-v.b.y)-(u.a.y-u.b.y)*(v.a.x-v.b.x));
    ret.x+=(u.b.x-u.a.x)*t;
    ret.y+=(u.b.y-u.a.y)*t;
    return ret;
}

point barycenter(point a,point b,point c){
    line u,v;
    u.a.x=(a.x+b.x)/2;
    u.a.y=(a.y+b.y)/2;
    u.b=c;
    v.a.x=(a.x+c.x)/2;
    v.a.y=(a.y+c.y)/2;
    v.b=b;
    return intersection(u,v);
}

// 多边形重心
point barycenter(int n,point* p){
    point ret,t;
    double t1=0,t2;
    int i;
    ret.x=ret.y=0;
    for (i=1;i<n-1;i++)
        if (fabs(t2=xmult(p[0],p[i],p[i+1]))>eps){
```

```
            t=barycenter(p[0],p[i],p[i+1]);
            ret.x+=t.x*t2;
            ret.y+=t.y*t2;
            t1+=t2;
        }
    if (fabs(t1)>eps)
        ret.x/=t1,ret.y/=t1;
    return ret;
}
```

6.4 整数点问题

6.4.1 线段上整点个数

```
int OnSegment(line l) {
        return __gcd(fabs(l.s.x - l.e.x), fabs(l.s.y - l.e.y)) + 1;
}
```

6.4.2 多边形边上整点个数

```
int OnEdge(point p[], int n) {
        int i, ret = 0;
        for (i = 0; i < n; i++)
                ret += __gcd(fabs(p[i].x - p[(i + 1) % n].x), fabs(p[i].y - p[(i + 1) % n].y));
        return ret;
}
```

6.4.3 多边形内整点个数

```
int InSide(point p[], int n) {
        int i, area = 0;
        for (i = 0; i < n; i++)
                area += p[(i + 1) % n].y * (p[i].x - p[(i + 2) % n].x);
        return (fabs(area) - OnEdge(n, p)) / 2 + 1;
}
```

6.5 圆

```
point waixin(point a, point b, point c) {
        double a1 = b.x - a.x, b1 = b.y - a.y, c1 = (a1 * a1 + b1 * b1) / 2;
        double a2 = c.x - a.x, b2 = c.y - a.y, c2 = (a2 * a2 + b2 * b2) / 2;
        double d = a1 * b2 - a2 * b1;
        return point(a.x + (c1 * b2 - c2 * b1) / d, a.y + (a1 * c2 - a2 * c1) / d);
}
```

6.6 经典题

```
#include <cstdio>
#include <cmath>
#include <algorithm>
using namespace std;
const int N = 100100;
struct Point {
    double x, y;
```

```
};
int n;
Point p[N], tmp[N];

bool cmp(Point a, Point b) {return a.x == b.x ? a.y < b.y : a.x < b.x;}
bool cmpy(Point a, Point b) {return a.y < b.y;}
double dis(Point a, Point b) {
    double dx = a.x - b.x;
    double dy = a.y - b.y;
    return sqrt(dx * dx + dy * dy);
}
double solve(int l, int r) {
    double d = 1e20;
    if (l == r) return d;
    if (l + 1 == r) return dis(p[l], p[r]);
    int mid = l + r >> 1;
    double d1 = solve(l, mid);
    double d2 = solve(mid + 1, r);
    d = min(d1, d2);
    int k = 0;
    for (int i = l; i <= r; i++)
        if (fabs(p[i].x - p[mid].x) <= d)
            tmp[k++] = p[i];
    sort(tmp, tmp + k, cmpy);
    for (int i = 0; i < k; i++)
        for (int j = i + 1; j < k; j++) {
            if (tmp[j].y - tmp[i].y > d) break;
            d = min(d, dis(tmp[i], tmp[j]));
        }
    return d;
}
int main() {
    while (scanf("%d", &n) && n != 0) {
        for (int i = 0; i < n; i++)
            scanf("%lf %lf", &p[i].x, &p[i].y);
        sort(p, p + n, cmp);
        printf("%.2lf\n", solve(0, n - 1) / 2);
    }
    return 0;
}
```

7 字符串

7.1 KMP

```
// 返回 y 中 x 的个数
int ne[N];
void initkmp(char x[], int m) {
    int i, j; j = ne[0] = -1; i = 0;
    while (i < m) {
        while (j != -1 && x[i] != x[j])
            j = ne[j];
        ne[++i] = ++j;
    }
}
```

```
int kmp(char x[], int m, char y[], int n) {
    int i, j, ans; i = j = ans = 0;
    initkmp (x, m);
    while (i < n) {
        while (j != -1 && y[i] != x[j]) j = ne[j];
        i++; j++;
        if (j >= m) {
            ans++; j = ne[j];
        }
    }
    return ans;
}
```

7.2 Manacher 最长回文子串

```
// O(n) 求解最长回文子串
const int N = 1000100;
char s[N], str[N << 1];
int p[N << 1];
void Manacher(char s[], int &n) {
    str[0] = '$';
    str[1] = '#';
    for (int i = 0; i < n; i++) {
        str[(i << 1) + 2] = s[i];
        str[(i << 1) + 3] = '#';
    }
    n = 2 * n + 2;
    str[n] = 0;
    int mx = 0, id;
    for (int i = 1; i < n; i++) {
        p[i] = mx > i ? min(p[2 * id - i], mx - i) : 1;
        while(str[i - p[i]] == str[i + p[i]]) p[i]++;
        if (p[i] + i > mx) {
            mx = p[i] + i;
            id = i;
        }
    }
}
int solve(char s[]) {
    int n = strlen(s);
    Manacher(s, n);
    int res = 0;
    for (int i = 0; i < n; i++)
        res = max(res, p[i]);
    return res - 1;
}
```

7.3 AC 自动机

```
#include <cstdio>
#include <cstring>
using namespace std;
#define rep(i,a,n) for (int i=a;i<n;i++)
const int AC_SIGMA = 26, AC_V = 29, AC_N = 500100;
struct AC_automaton {
```

```
    struct node {
        node *go[AC_V], *fail, *fa;
        int fg, id;
    } pool[AC_N], *cur, *root, *q[AC_N];
    node* newnode() {
        node *p = cur++;
        memset(p->go, 0, sizeof(p->go));
        p->fail = p->fa = NULL; p->fg = 0;
        return p;
    }
    void init() { cur = pool; root = newnode();}
    node* append(node *p, int w) {
        if (!p->go[w]) p->go[w] = newnode(), p->go[w]->fa = p;
        return p = p->go[w];
    }
    void build() {
        int t = 1;
        q[0] = root;
        rep(i, 0, t) rep(j, 0, AC_SIGMA) if (q[i]->go[j]) {
            node *v = q[i]->go[j], *p = v->fa->fail;
            while (p && !p->go[j]) p = p->fail;
            if (p) v->fail = p->go[j]; else v->fail = root;
            q[t++] = q[i]->go[j];
        } else {
            node *p = q[i]->fail;
            while (p && !p->go[j]) p = p->fail;
            if (p) q[i]->go[j] = p->go[j]; else q[i]->go[j] = root;
        }
    }
    int query(char s[]) {
        node *p = root;
        int res = 0;
        for (int i = 0; s[i]; i++) {
            p = p->go[s[i] - 'a'];
            node *v = p;
            while (v != root) {
                res += v->fg;
                v->fg = 0;
                v = v->fail;
            }
        }
        return res;
    }
} T;
typedef AC_automaton::node ACnode;


const int MAXN = 1000000 + 1000;
char txt[MAXN];

int main() {
#ifdef MANGOGAO
    freopen("data.in", "r", stdin);
#endif

```

```
    int t;
    scanf("%d", &t);
    while (t--) {
        int n;
        scanf("%d", &n);
        T.init();
        char s[55];
        rep(i, 0, n) {
            ACnode *p = T.root;
            scanf("%s", s);
            for (int j = 0; s[j]; j++)
                p = T.append(p, s[j] - 'a');
            p->fg++;
        }
        T.build();
        scanf("%s", txt);
        printf("%d\n", T.query(txt));
    }
    return 0;
}
```

8 动态规划

8.1 最大子序列和

```
// 传入序列 a 和长度 n，返回最大子序列和
// 限制最短长度：用 cnt 记录长度，rt 更新时判断
int MaxSeqSum(int a[], int n) {
        int rt = 0, cur = 0;
        for (int i = 0; i < n; i++) {
                cur += a[i];
                rt = rt < cur ? cur : rt;
                cur = cur < 0 ? 0 : cur;
        }
        return rt;
}
```

8.2 最长上升子序列 LIS

```
// 序列下标从 1 开始，LIS() 返回长度，序列存在 lis[] 中
#define N 100100
int n, len, a[N], b[N], f[N];
int Find(int p, int l, int r) {
    int mid;
    while (l <= r) {
        mid = (l + r) >> 1;
        if (a[p] > b[mid]) l = mid + 1;
        else r = mid - 1;
    }
    return f[p] = l;
}
int LIS(int lis[]) {
int len = 1;
f[1] = 1;
b[1] = a[1];
```

```
    for (int i = 2; i <= n; i++) {
        if (a[i] > b[len]) b[++len] = a[i], f[i] = len;
        else b[Find(i, 1, len)] = a[i];
    }
    for (int i = n, t = len; i >= 1 && t >= 1; i--)
        if (f[i] == t)
            lis[--t] = a[i];
    return len;
}
```

8.3 最长公共上升子序列 LCIS

```
// 序列下标从 1 开始
int LCIS(int a[], int b[], int n, int m) {
        set(dp, 0);
        for (int i = 1; i <= n; i++) {
                int ma = 0;
                for (int j = 1; j <= m; j++) {
                        dp[i][j] = dp[i - 1][j];
                        if (a[i] > b[j]) ma = max(ma, dp[i - 1][j]);
                        if (a[i] == b[j]) dp[i][j] = ma + 1;
                }
        }
        return *max_element(dp[n] + 1, dp[n] + 1 + m);
}
```

9 附录

```
typedef long long ll;
namespace fastIO {  //包含所有类型
#define BUF_SIZE 100000
#define OUT_SIZE 100000
#define ll long long
// fread->read
bool IOerror = 0;
inline char nc() {
    static char buf[BUF_SIZE], *p1 = buf + BUF_SIZE, *pend = buf + BUF_SIZE;
    if (p1 == pend) {
        p1 = buf;
        pend = buf + fread(buf, 1, BUF_SIZE, stdin);
        if (pend == p1) {
            IOerror = 1;
            return -1;
        }
        //{printf("IO error!\n");system("pause");for (;;);exit(0);}
    }
    return *p1++;
}
inline bool blank(char ch) {
    return ch == ' ' || ch == '\n' || ch == '\r' || ch == '\t';
}
inline void read(int &x) {
    bool sign = 0;
    char ch = nc();
```

```
    x = 0;
    for (; blank(ch); ch = nc())
        ;
    if (IOerror) return;
    if (ch == '-') sign = 1, ch = nc();
    for (; ch >= '0' && ch <= '9'; ch = nc()) x = x * 10 + ch - '0';
    if (sign) x = -x;
}
inline void read(ll &x) {
    bool sign = 0;
    char ch = nc();
    x = 0;
    for (; blank(ch); ch = nc())
        ;
    if (IOerror) return;
    if (ch == '-') sign = 1, ch = nc();
    for (; ch >= '0' && ch <= '9'; ch = nc()) x = x * 10 + ch - '0';
    if (sign) x = -x;
}
inline void read(double &x) {
    bool sign = 0;
    char ch = nc();
    x = 0;
    for (; blank(ch); ch = nc())
        ;
    if (IOerror) return;
    if (ch == '-') sign = 1, ch = nc();
    for (; ch >= '0' && ch <= '9'; ch = nc()) x = x * 10 + ch - '0';
    if (ch == '.') {
        double tmp = 1;
        ch = nc();
        for (; ch >= '0' && ch <= '9'; ch = nc()) tmp /= 10.0, x += tmp * (ch - '0');
    }
    if (sign) x = -x;
}
inline void read(char *s) {
    char ch = nc();
    for (; blank(ch); ch = nc())
        ;
    if (IOerror) return;
    for (; !blank(ch) && !IOerror; ch = nc()) *s++ = ch;
    *s = 0;
}
inline void read(char &c) {
    for (c = nc(); blank(c); c = nc())
        ;
    if (IOerror) {
        c = -1;
        return;
    }
}
// getchar->read
inline void read1(int &x) {
    char ch;
```

```
    int bo = 0;
    x = 0;
    for (ch = getchar(); ch < '0' || ch > '9'; ch = getchar())
        if (ch == '-') bo = 1;
    for (; ch >= '0' && ch <= '9'; x = x * 10 + ch - '0', ch = getchar())
        ;
    if (bo) x = -x;
}
inline void read1(ll &x) {
    char ch;
    int bo = 0;
    x = 0;
    for (ch = getchar(); ch < '0' || ch > '9'; ch = getchar())
        if (ch == '-') bo = 1;
    for (; ch >= '0' && ch <= '9'; x = x * 10 + ch - '0', ch = getchar())
        ;
    if (bo) x = -x;
}
inline void read1(double &x) {
    char ch;
    int bo = 0;
    x = 0;
    for (ch = getchar(); ch < '0' || ch > '9'; ch = getchar())
        if (ch == '-') bo = 1;
    for (; ch >= '0' && ch <= '9'; x = x * 10 + ch - '0', ch = getchar())
        ;
    if (ch == '.') {
        double tmp = 1;
        for (ch = getchar(); ch >= '0' && ch <= '9'; tmp /= 10.0, x += tmp * (ch - '0'), ch = getchar())
            ;
    }
    if (bo) x = -x;
}
inline void read1(char *s) {
    char ch = getchar();
    for (; blank(ch); ch = getchar())
        ;
    for (; !blank(ch); ch = getchar()) *s++ = ch;
    *s = 0;
}
inline void read1(char &c) {
    for (c = getchar(); blank(c); c = getchar())
        ;
}
// scanf->read
inline void read2(int &x) {
    scanf("%d", &x);
}
inline void read2(ll &x) {
#ifdef _WIN32
    scanf("%I64d", &x);
#else
#ifdef __linux
    scanf("%lld", &x);
```

```
#else
    puts("error:can't recognize the system!");
#endif
#endif
}
inline void read2(double &x) {
    scanf("%lf", &x);
}
inline void read2(char *s) {
    scanf("%s", s);
}
inline void read2(char &c) {
    scanf(" %c", &c);
}
inline void readln2(char *s) {
    gets(s);
}
// fwrite->write
struct Ostream_fwrite {
    char *buf, *p1, *pend;
    Ostream_fwrite() {
        buf = new char[BUF_SIZE];
        p1 = buf;
        pend = buf + BUF_SIZE;
    }
    void out(char ch) {
        if (p1 == pend) {
            fwrite(buf, 1, BUF_SIZE, stdout);
            p1 = buf;
        }
        *p1++ = ch;
    }
    void print(int x) {
        static char s[15], *s1;
        s1 = s;
        if (!x) *s1++ = '0';
        if (x < 0) out('-'), x = -x;
        while (x) *s1++ = x % 10 + '0', x /= 10;
        while (s1-- != s) out(*s1);
    }
    void println(int x) {
        static char s[15], *s1;
        s1 = s;
        if (!x) *s1++ = '0';
        if (x < 0) out('-'), x = -x;
        while (x) *s1++ = x % 10 + '0', x /= 10;
        while (s1-- != s) out(*s1);
        out('\n');
    }
    void print(ll x) {
        static char s[25], *s1;
        s1 = s;
        if (!x) *s1++ = '0';
        if (x < 0) out('-'), x = -x;
        while (x) *s1++ = x % 10 + '0', x /= 10;
```

```
        while (s1-- != s) out(*s1);
    }
    void println(ll x) {
        static char s[25], *s1;
        s1 = s;
        if (!x) *s1++ = '0';
        if (x < 0) out('-'), x = -x;
        while (x) *s1++ = x % 10 + '0', x /= 10;
        while (s1-- != s) out(*s1);
        out('\n');
    }
    void print(double x, int y) {
        static ll mul[] = {1, 10, 100, 1000, 10000, 100000, 1000000, 10000000, 100000000, 1000000000, 10000000000LL, 100000000000LL, 1000000000000LL, 10000000000000LL, 100000000000000LL, 1000000000000000LL, 10000000000000000LL, 100000000000000000LL};
        if (x < -1e-12) out('-'), x = -x;
        x *= mul[y];
        ll x1 = (ll)floor(x);
        if (x - floor(x) >= 0.5) ++x1;
        ll x2 = x1 / mul[y], x3 = x1 - x2 * mul[y];
        print(x2);
        if (y > 0) {
            out('.');
            for (size_t i = 1; i < y && x3 * mul[i] < mul[y]; out('0'), ++i)
                ;
            print(x3);
        }
    }
    void println(double x, int y) {
        print(x, y);
        out('\n');
    }
    void print(char *s) {
        while (*s) out(*s++);
    }
    void println(char *s) {
        while (*s) out(*s++);
        out('\n');
    }
    void flush() {
        if (p1 != buf) {
            fwrite(buf, 1, p1 - buf, stdout);
            p1 = buf;
        }
    }
    ~Ostream_fwrite() {
        flush();
    }
} Ostream;
inline void print(int x) {
    Ostream.print(x);
}
inline void println(int x) {
    Ostream.println(x);
```

```
}
inline void print(char x) {
    Ostream.out(x);
}
inline void println(char x) {
    Ostream.out(x);
    Ostream.out('\n');
}
inline void print(ll x) {
    Ostream.print(x);
}
inline void println(ll x) {
    Ostream.println(x);
}
inline void print(double x, int y) {
    Ostream.print(x, y);
}
inline void println(double x, int y) {
    Ostream.println(x, y);
}
inline void print(char *s) {
    Ostream.print(s);
}
inline void println(char *s) {
    Ostream.println(s);
}
inline void println() {
    Ostream.out('\n');
}
inline void flush() {
    Ostream.flush();
}
// puts->write
char Out[OUT_SIZE], *o = Out;
inline void print1(int x) {
    static char buf[15];
    char *p1 = buf;
    if (!x) *p1++ = '0';
    if (x < 0) *o++ = '-', x = -x;
    while (x) *p1++ = x % 10 + '0', x /= 10;
    while (p1-- != buf) *o++ = *p1;
}
inline void println1(int x) {
    print1(x);
    *o++ = '\n';
}
inline void print1(ll x) {
    static char buf[25];
    char *p1 = buf;
    if (!x) *p1++ = '0';
    if (x < 0) *o++ = '-', x = -x;
    while (x) *p1++ = x % 10 + '0', x /= 10;
    while (p1-- != buf) *o++ = *p1;
}
inline void println1(ll x) {
```

```
    print1(x);
    *o++ = '\n';
}
inline void print1(char c) {
    *o++ = c;
}
inline void println1(char c) {
    *o++ = c;
    *o++ = '\n';
}
inline void print1(char *s) {
    while (*s) *o++ = *s++;
}
inline void println1(char *s) {
    print1(s);
    *o++ = '\n';
}
inline void println1() {
    *o++ = '\n';
}
inline void flush1() {
    if (o != Out) {
        if (*(o - 1) == '\n') *--o = 0;
        puts(Out);
    }
}
struct puts_write {
    ~puts_write() {
        flush1();
    }
} _puts;
inline void print2(int x) {
    printf("%d", x);
}
inline void println2(int x) {
    printf("%d\n", x);
}
inline void print2(char x) {
    printf("%c", x);
}
inline void println2(char x) {
    printf("%c\n", x);
}
inline void print2(ll x) {
#ifdef _WIN32
    printf("%I64d", x);
#else
#ifdef __linux
    printf("%lld", x);
#else
    puts("error:can't recognize the system!");
#endif
#endif
}
inline void println2(ll x) {
```

```
    print2(x);
    printf("\n");
}
inline void println2() {
    printf("\n");
}
#undef ll
#undef OUT_SIZE
#undef BUF_SIZE
};  // namespace fastIO
using namespace fastIO;
```