



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
1.1 常用模板	1
1.2 所有宏和头文件	1
1.3 程序对拍模板	2
1.3.1 对拍脚本	2
1.3.2 造随机数	3
1.3.3 使用方法	3
2 STL 和 fastIO	4
2.1 快读	4
2.1.1 整数 getchar 版	4
2.1.2 整数 fread 版	4
2.1.3 浮点数 getchar 版	5
2.1.4 对比说明	5
2.2 快写	6
2.2.1 整数 putchar 版	6
2.2.2 整数数组版	6
2.2.3 浮点数 putchar 版	6
2.2.4 对比说明	7
2.3 容器	7
2.3.1 vector	7
2.3.2 queue	7
2.3.3 set	7
2.3.4 map	7
2.3.5 bitset	7
2.3.6 stack	7
3 数学	8
3.1 素数	8
3.1.1 埃氏筛	8
3.1.2 欧拉筛	8
3.1.3 随机素数判定	8
3.1.4 分解质因数	9
3.2 欧拉函数	9
3.2.1 求一个数的欧拉函数	9
3.2.2 筛法求欧拉函数	9
3.3 扩展欧几里得-乘法逆元	10
3.3.1 扩展欧几里得	10
3.3.2 求 $ax+by=c$ 的解	10
3.3.3 乘法逆元	10
3.4 模线性方程组	10
3.4.1 中国剩余定理	10
3.4.2 一般模线性方程组	11
3.5 组合数学	11
3.5.1 一般组合数	11
3.5.2 Lucas 定理	11

3.5.3	大组合数	12
3.5.4	Polya 定理	13
3.6	快速乘 + 快速幂	13
3.7	莫比乌斯反演	13
3.7.1	莫比乌斯	13
3.7.2	n 个数中互质数对数	14
3.7.3	VisibleTrees	15
3.8	其他	15
3.8.1	Josephus 问题	15
3.8.2	数位问题	15
3.9	相关公式	16
4	数据结构	17
4.1	树状数组	17
4.2	线段树	17
4.2.1	声明	17
4.2.2	单点更新-区间查询	17
4.2.3	区间更新-区间查询	18
4.3	字典树	19
4.4	RMQ	20
5	图论	20
5.1	并查集	20
5.2	最小生成树	21
5.2.1	Kruskal	21
5.2.2	Prim	21
5.3	最短路	22
5.3.1	Dijkstra-邻接矩阵	22
5.3.2	Dijkstra-邻接表数组	23
5.3.3	Dijkstra-邻接表向量	23
5.3.4	Dijkstra-优先队列	24
5.3.5	Bellman-Ford(可判负环)	25
5.3.6	SPFA	25
5.3.7	Floyd 算法	26
5.4	拓扑排序	27
5.4.1	邻接矩阵	27
5.4.2	邻接表	27
5.5	欧拉回路	28
5.5.1	判定	28
5.5.2	求解	28
6	计算几何	30
6.1	定义	30
6.2	位置关系	31
6.2.1	两点间距离	31
6.2.2	直线与直线的交点	31
6.2.3	判断线段与线段相交	31
6.2.4	判断线段与直线相交	31

6.2.5	点到直线距离	31
6.2.6	点到线段距离	31
6.2.7	点在线段上	32
6.3	多边形	32
6.3.1	多边形面积	32
6.3.2	点在凸多边形内	32
6.3.3	点在任意多边形内	32
6.3.4	判断凸多边形	33
6.3.5	小结	33
6.4	整数点问题	36
6.4.1	线段上整点个数	36
6.4.2	多边形边上整点个数	36
6.4.3	多边形内整点个数	36
6.5	圆	36
6.6	经典题	36
7	字符串	37
7.1	KMP	37
7.2	Manacher 最长回文子串	38
7.3	AC 自动机	38
8	动态规划	40
8.1	最大子序列和	40
8.2	最长上升子序列 LIS	40
8.3	最长公共上升子序列 LCIS	41
9	附录	41

1 书写调试环境

1.1 常用模板

```
1  /**
2   * @author pengpenlang
3   * @brief 外能头文件 + 多样例
4   */
5   #include <bits/stdc++.h>
6   #pragma GCC optimize(2) //开启 O2 编译
7   #pragma G++ optimize(2)
8   // #define ONLINE_JUDGE
9   #define endl "\n"
10  #define fi first
11  #define se second
12  #define pb push_back
13  #define all(x) x.begin(), x.end()
14  #define rep(i, x, y) for (auto i = (x); i != (y + 1); ++i)
15  #define dep(i, x, y) for (auto i = (x); i != (y - 1); --i)
16  #ifndef ONLINE_JUDGE
17  #define de(...) cout << '[' << __VA_ARGS__ << "]" = " << __VA_ARGS__ << endl;
18  #else
19  #define de(...)
20  #endif
21  using namespace std;
22  typedef long long ll;
23  typedef pair<int, int> pii;
24
25  void solve() {
26      /* 处理每组样例 */
27  }
28
29  signed main() {
30      ios::sync_with_stdio(false), cin.tie(0);
31  #ifndef ONLINE_JUDGE
32      freopen("IO\\in.txt", "r", stdin);
33      freopen("IO\\out.txt", "w", stdout);
34      clock_t start, end;
35      start = clock();
36  #endif
37      solve();
38  #ifndef ONLINE_JUDGE
39      end = clock();
40      cout << endl
41           << "Runtime: " << (double)(end - start) / CLOCKS_PER_SEC << "s\n";
42  #endif
43      return 0;
44  }
```

1.2 所有宏和头文件

```
1  /**
2   * @author pengpenglang
3   * @brief ACM 个人常用头文件与宏定义
4   */
```

```

5  #include <algorithm>
6  #include <bitset>
7  #include <cctype>
8  #include <climits>
9  #include <cmath>
10 #include <cstdio>
11 #include <cstdlib>
12 #include <cstring>
13 #include <functional>
14 #include <iomanip>
15 #include <iostream>
16 #include <map>
17 #include <queue>
18 #include <set>
19 #include <stack>
20 #include <string>
21 #include <vector>
22 using namespace std;
23 #pragma GCC optimize(2) //开启 o2 编译
24 #pragma G++ optimize(2)
25 #pragma GCC optimize(3, "Ofast", "inline") //开启 o3 编译
26 #pragma G++ optimize(3, "Ofast", "inline")
27 // #define ONLINE_JUDGE
28 #define endl "\n"
29 #define fi first
30 #define se second
31 #define pb push_back
32 #define all(x) x.begin(), x.end()
33 #define fastio ios::sync_with_stdio(false), cin.tie(0);
34 #define rep(i, x, y) for (auto i = (x); i != (y + 1); ++i)
35 #define dep(i, x, y) for (auto i = (x); i != (y - 1); --i)
36 #ifndef ONLINE_JUDGE
37 #define de(...) cout << '[' << __VA_ARGS__ << "]" = " << __VA_ARGS__ << endl;
38 #else
39 #define de(...)
40 #endif
41 typedef long long ll;
42 typedef unsigned long long ull;
43 typedef pair<int, int> pii;
44 typedef vector<int> vi;
45 const int inf = 0x3f3f3f3f, mod = 1e9 + 7;
46 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
47 const double eps = 1e-8;
48 inline int sgn(double x) { //和 0 比大于返 1 等于返 0 小于返 -1
49     return (x > eps) - (x < -eps);
50 }

```

1.3 程序对拍模板

1.3.1 对拍脚本

```

1  @echo off
2  :loop
3      rand.exe %random% > data.in
4      std.exe < data.in > std.out

```

```

5     my.exe < data.in > my.out
6     fc my.out std.out
7     if not errorlevel 1 goto loop
8     pause
9     goto loop

```

1.3.2 造随机数

```

1  /**
2   * @author pengpenglang
3   * @brief 利用 random 造随机样例
4   */
5  #include <bits/stdc++.h>
6  using namespace std;
7  typedef long long ll;
8  const ll a = 0, b = 1e5; //规定生成随机数的范围
9
10 signed main(int argc, char *argv[]) {
11     stringstream ss;
12     ll seed = time(NULL);
13     if (argc > 1) { //如果传入了参数
14         ss.clear();
15         ss << argv[1];
16         ss >> seed; //把参数转换成整数赋值给 seed
17     }
18     //rand_max=32767
19     auto random = [] { //加强随机数范围, 生成数在 [a,b]
20         return a + rand() * rand() % (b - a + 1);
21     };
22     srand(seed);
23     //以上为随机数初始化, 请勿修改
24     //下面利用利用 rand() 或者自定义的 random() 生成随机数
25     cout << 1 << endl; //单组循环、测试
26     int len = random();
27     string s = "";
28     for (int i = 0; i < len; ++i)
29         s += (char)(rand() % 25 + 'a');
30     cout << s << endl;
31     return 0;
32 }

```

1.3.3 使用方法

文件位置关系:

duipai

|- check.bat ==> 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 ==> 保存每次对拍的随机样例

|- my.cpp ==> 自己程序的源代码

|- my.exe ==> 自己程序编译的 exe

|- my.out ==> 保存每次对拍自己程序的输出

|- rand.cpp ==> 造随机数程序的源代码

|- rand.exe ==> 造随机数程序编译的 exe

|- std.cpp ==> std 程序的源代码

|- std.exe ==> std 程序编译的 exe

‘- std.out \implies 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 ctrl+c

2 STL 和 fastIO

2.1 快读

2.1.1 整数 getchar 版

```
1  /**
2   * @brief getchar 版快读支持所有整数类型
3   */
4  inline _Tp read(_Tp &x) {
5      char ch = getchar(), sgn = 0;
6      x = 0;
7      while (ch ^ '-' && !isdigit(ch)) ch = getchar();
8      if (ch == '-') ch = getchar(), sgn = 1;
9      while (isdigit(ch)) x = x * 10 + ch - '0', ch = getchar();
10     if (sgn) x = -x;
11     return x;
12 }
13
14 read(a);
```

2.1.2 整数 fread 版

```
1  /**
2   * @brief fread 版快读支持所有整数类型，超多数据时表现优异
3   */
4  struct ios_in {
5      inline char gc() {
6          const int MAXN = 1e5 + 100; //读入字符串的最大长度（根据实际情况调整）
7          static char buf[MAXN], *l, *r;
8          return (l == r) && (r = (l = buf) + fread(buf, 1, MAXN, stdin), l == r)
9              ↪ ? EOF : *l++;
10     }
11     template <typename _Tp>
12     inline ios_in &operator>>(_Tp &x) {
13         static char ch, sgn;
14         for (sgn = 0, ch = gc(); !isdigit(ch); ch = gc()) {
15             if (!~ch) return *this;
16             sgn |= ch == '-';
17         }
18         for (x = 0; isdigit(ch); ch = gc())
19             x = (x << 1) + (x << 3) + (ch ^ '0');
20         sgn && (x = -x);
21         return *this;
22     }
23 } Cin;
24
25 Cin >> a;
```


2.1.3 浮点数 getchar 版

```
1  /**
2   * @brief getchar 版快读支持浮点数
3   */
4  inline bool read_fl(double &num) {
5      char in;
6      double Dec = 0.1;
7      bool IsN = false, IsD = false;
8      in = getchar();
9      if (in == EOF) return false;
10     while (in != '-' && in != '.' && (in < '0' || in > '9'))
11         in = getchar();
12     if (in == '-') {
13         IsN = true;
14         num = 0;
15     } else if (in == '.') {
16         IsD = true;
17         num = 0;
18     } else
19         num = in - '0';
20     if (!IsD) {
21         while (in = getchar(), in >= '0' && in <= '9') {
22             num *= 10;
23             num += in - '0';
24         }
25     }
26     if (in != '.') {
27         if (IsN) num = -num;
28         return true;
29     } else {
30         while (in = getchar(), in >= '0' && in <= '9') {
31             num += Dec * (in - '0');
32             Dec *= 0.1;
33         }
34     }
35     if (IsN) num = -num;
36     return true;
37 }
38
39 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

```
1  /**
2   * @brief putchar 版快写, 仅支持  $[INT_{MIN}, INT_{MAX}]$  范围
3   */
4  void write(int x) {
5      if (x < 0) putchar('-'), x = -x;
6      if (x > 9) write(x / 10);
7      putchar(x % 10 + '0');
8  }
9
10 putchar(a); //不能输出回车
```

2.2.2 整数数组版

```
1  /**
2   * @brief putchar 进一步数组优化版快写, 超多数据时表现优异
3   */
4  struct ios_out {
5      template <typename _Tp>
6      inline void operator<<(_Tp &x) {
7          const int MAXN = 1e3 + 100; //存储数字的数组 (根据情况调整)
8          char F[MAXN];
9          _Tp tmp = x >= 0 ? x : (putchar('-'), -x);
10         int cnt = 0;
11         while (tmp) {
12             F[cnt++] = tmp % 10 + '0';
13             tmp /= 10;
14         }
15         if (!cnt) {
16             putchar('0');
17             return;
18         }
19         while (cnt) putchar(F[--cnt]);
20     }
21 } Cout;
22
23 Cout << a; //不能输出回车
```

2.2.3 浮点数 putchar 版

```
1  /**
2   * @brief 浮点数 putchar 快写
3   */
4
5  inline void dwrite(ll x) { //用于输出整数部分
6      if (x == 0) {
7          putchar(48);
8          return;
9      }
10     int bit[20], p = 0, i;
11     for (; x; x /= 10) bit[++p] = x % 10;
12     for (i = p; i > 0; --i) putchar(bit[i] + 48);
13 }
```

```

14 inline void write(double x, int k = 6) { //不加位数, 默认保留小数点后 6 位小数
15     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
16     if (x == 0) { //x=0, 保留的 k 位不断输出 0;
17         putchar('0'), putchar('.');
18         for (int i = 1; i <= k; ++i) putchar('0');
19         return;
20     }
21     if (x < 0) putchar('-'), x = -x; //负数
22     ll y = (ll)(x * n) % n;
23     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.
24     dwrite(x), putchar('.'); //输出整数部分和小数点
25     int bit[10], p = 0, i;
26     for (; p < k; y /= 10) bit[++p] = y % 10; //必须严格按照 k 位保留, 否则就
    ↪ gg 了
27     for (i = p; i > 0; i--) putchar(bit[i] + 48);
28 }

```

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

1

2.3.2 queue

1

2.3.3 set

1

2.3.4 map

1

2.3.5 bitset

1

2.3.6 stack

1

3 数学

3.1 素数

3.1.1 埃氏筛

```
1 //  $O(n \log \log n)$  筛出  $MAXN$  内所有素数
2 //  $notprime[i] = 0/1$  0 为素数 1 为非素数
3 const int MAXN = 1000100;
4 bool notprime[MAXN] = {1, 1};          // 0/1 为非素数
5 void GetPrime() {
6     for (int i = 2; i < MAXN; i++)
7         if (!notprime[i] && i <= MAXN / i)        // 筛到  $\sqrt{n}$  为止
8             for (int j = i * i; j < MAXN; j += i)
9                 notprime[j] = 1;
10 }
```

3.1.2 欧拉筛

```
1 //  $O(n)$  得到欧拉函数  $\phi[]$ 、素数表  $prime[]$ 、素数个数  $tot$ 
2 // 传入的  $n$  为函数定义域上界
3 const int MAXN = 100010;
4 bool vis[MAXN];
5 int tot, phi[MAXN], prime[MAXN];
6 void CalPhi(int n) {
7     set(vis, 0); phi[1] = 1; tot = 0;
8     for (int i = 2; i < n; i++) {
9         if (!vis[i]) {
10             prime[tot++] = i;
11             phi[i] = i - 1;
12         }
13         for (int j = 0; j < tot; j++) {
14             if (i * prime[j] > n) break;
15             vis[i * prime[j]] = 1;
16             if (i % prime[j] == 0) {
17                 phi[i * prime[j]] = phi[i] * prime[j];
18                 break;
19             }
20             else phi[i * prime[j]] = phi[i] * (prime[j] - 1);
21         }
22     }
23 }
```

3.1.3 随机素数判定

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

```

12         if (y == 1 && x != 1 && x != n - 1) return 0;
13         x = y;
14     }
15     if (x != 1) return 0;
16 }
17 return 1;
18 }

```

3.1.4 分解质因数

```

1 // 函数返回素因数个数
2 // 数组以 fact[i][0]fact[i][1] 的形式保存第 i 个素因数
3 ll fact[100][2];
4 int getFactors(ll x) {
5     int cnt = 0;
6     for (int i = 0; prime[i] <= x / prime[i]; i++) {
7         fact[cnt][1] = 0;
8         if (x % prime[i] == 0) {
9             fact[cnt][0] = prime[i];
10            while (x % prime[i] == 0) {
11                fact[cnt][1]++;
12                x /= prime[i];
13            }
14            cnt++;
15        }
16    }
17    if (x != 1) {
18        fact[cnt][0] = x;
19        fact[cnt++][1] = 1;
20    }
21    return cnt;
22 }

```

3.2 欧拉函数

3.2.1 求一个数的欧拉函数

```

1 long long Euler(long long n) {
2     long long rt = n;
3     for (int i = 2; i * i <= n; i++)
4         if (n % i == 0) {
5             rt -= rt / i;
6             while (n % i == 0) n /= i;
7         }
8     if (n > 1) rt -= rt / n;
9     return rt;
10 }

```

3.2.2 筛法求欧拉函数

```

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

```

```

7         phi[j] = phi[j] / i * (i - 1);
8     }
9 }

```

3.3 扩展欧几里得-乘法逆元

3.3.1 扩展欧几里得

```

1 void exgcd(ll a, ll b, ll &d, ll &x, ll &y) {
2     if (!b) {d = a; x = 1; y = 0;}
3     else {exgcd(b, a % b, d, y, x); y -= x * (a / b);}
4 }

```

3.3.2 求 $ax+by=c$ 的解

```

1 // 引用返回通解:  $X = x + k * dx, Y = y - k * dy$ 
2 // 引用返回的  $x$  是最小非负整数解, 方程无解函数返回 0
3 #define Mod(a,b) (((a)%(b)+(b))%(b))
4 bool solve(ll a, ll b, ll c, ll &x, ll &y, ll &dx, ll &dy) {
5     if (a == 0 && b == 0) return 0;
6     ll d, x0, y0; exgcd(a, b, d, x0, y0);
7     if (c % d != 0) return 0;
8     dx = b / d; dy = a / d;
9     x = Mod(x0 * c / d, dx); y = (c - a * x) / b;
10    //  $y = \text{Mod}(y0 * c / d, dy); x = (c - b * y) / a;$ 
11    return 1;
12 }

```

3.3.3 乘法逆元

```

1 // 利用 exgcd 求  $a$  在模  $m$  下的逆元, 需要保证  $\text{gcd}(a, m) == 1$ .
2 ll inv(ll a, ll m) {
3     ll x, y, d; exgcd(a, m, d, x, y);
4     return d == 1 ? (x + m) % m : -1;
5 }
6 //  $a < m$  且  $m$  为素数时, 有以下两种求法
7 ll inv(ll a, ll m) {
8     return a == 1 ? 1 : inv(m % a, m) * (m - m / a) % m;
9 }
10 ll inv(ll a, ll m) {
11     return Pow(a, m - 2, m);
12 }

```

3.4 模线性方程组

3.4.1 中国剩余定理

```

1 //  $X = r[i] \% m[i]$ , 要求  $m[i]$  两两互质
2 // 引用返回通解  $X = re + k * mo$ 
3 void crt(ll r[], ll m[], ll n, ll &re, ll &mo) {
4     mo = 1, re = 0;
5     for (int i = 0; i < n; i++) mo *= m[i];
6     for (int i = 0; i < n; i++) {
7         ll x, y, d, tm = mo / m[i];
8         exgcd(tm, m[i], d, x, y);
9         re = (re + tm * x * r[i]) % mo;

```

```

10     } re = (re + mo) % mo;
11 }

```

3.4.2 一般模线性方程组

```

1 //  $X = r[i] \% m[i]$ ,  $m[i]$  可以不两两互质
2 // 引用返回通解  $X = re + k * mo$ , 函数返回是否有解
3 bool exCRT(ll r[], ll m[], ll n, ll &re, ll &mo) {
4     ll x, y, d; mo = m[0], re = r[0];
5     for (int i = 1; i < n; i++) {
6         exgcd(mo, m[i], d, x, y);
7         if ((r[i] - re) % d != 0) return 0;
8         x = (r[i] - re) / d * x % (m[i] / d);
9         re += x * mo;
10        mo = mo / d * m[i];
11        re %= mo;
12    } re = (re + mo) % mo;
13    return 1;
14 }

```

3.5 组合数学

3.5.1 一般组合数

```

1 //  $0 \leq m \leq n \leq 1000$ 
2 const int maxn = 1010;
3 ll C[maxn][maxn];
4 void CalComb() {
5     C[0][0] = 1;
6     for (int i = 1; i < maxn; i++) {
7         C[i][0] = 1;
8         for (int j = 1; j <= i; j++)
9             C[i][j] = (C[i - 1][j - 1] + C[i - 1][j]) % mod;
10    }
11 }
12
13 //  $0 \leq m \leq n \leq 10^5$ , 模  $p$  为素数
14 const int maxn = 100010;
15 ll f[maxn];
16 void CalFact() {
17     f[0] = 1;
18     for (int i = 1; i < maxn; i++)
19         f[i] = (f[i - 1] * i) % mod;
20 }
21 ll C(int n, int m) {
22     return f[n] * inv(f[m] * f[n - m] % mod, mod) % mod;
23 }

```

3.5.2 Lucas 定理

```

1 //  $1 \leq n, m \leq 10^9, 1 < p < 10^5$ ,  $p$  是素数
2 const int maxp = 100010;
3 ll f[maxp];
4 void CalFact(ll p) {
5     f[0] = 1;
6     for (int i = 1; i <= p; i++)

```

```

7         f[i] = (f[i - 1] * i) % p;
8     }
9     ll Lucas(ll n, ll m, ll p) {
10         ll ret = 1;
11         while (n && m) {
12             ll a = n % p, b = m % p;
13             if (a < b) return 0;
14             ret = (ret * f[a] * Pow(f[b] * f[a - b] % p, p - 2, p)) % p;
15             n /= p; m /= p;
16         }
17         return ret;
18     }

```

3.5.3 大组合数

```

1 //  $0 \leq n \leq 10^9, 0 \leq m \leq 10^4, 1 \leq k \leq 10^9 + 7$ 
2 vector<int> v;
3 int dp[110];
4 ll Cal(int l, int r, int k, int dis) {
5     ll res = 1;
6     for (int i = l; i <= r; i++) {
7         int t = i;
8         for (int j = 0; j < v.size(); j++) {
9             int y = v[j];
10            while (t % y == 0) {
11                dp[j] += dis;
12                t /= y;
13            }
14        }
15        res = res * (ll)t % k;
16    }
17    return res;
18 }
19 ll Comb(int n, int m, int k) {
20     set(dp, 0); v.clear(); int tmp = k;
21     for (int i = 2; i * i <= tmp; i++) {
22         if (tmp % i == 0) {
23             int num = 0;
24             while (tmp % i == 0) {
25                 tmp /= i;
26                 num++;
27             }
28             v.pb(i);
29         }
30     } if (tmp != 1) v.pb(tmp);
31     ll ans = Cal(n - m + 1, n, k, 1);
32     for (int j = 0; j < v.size(); j++) {
33         ans = ans * Pow(v[j], dp[j], k) % k;
34     }
35     ans = ans * inv(Cal(2, m, k, -1), k) % k;
36     return ans;
37 }

```


3.5.4 Polya 定理

```
1 // 推论：一共  $n$  个置换，第  $i$  个置换的循环节个数为  $\gcd(i, n)$ 
2 //  $N * N$  的正方形格子， $c^{n^2} + 2c^{\frac{n^2+3}{4}} + c^{\frac{n^2+1}{2}} + 2c^{\frac{n+1}{2}} + 2c^{\frac{n(n+1)}{2}}$ 
3 // 正六面体， $(m^8 + 17m^4 + 6m^2)/24$ 
4 // 正四面体， $(m^4 + 11m^2)/12$ 
5 // 长度为  $n$  的项链串用  $c$  种颜色染
6 ll solve(int c, int n) {
7     if (n == 0) return 0;
8     ll ans = 0;
9     for (int i = 1; i <= n; i++)
10         ans += Pow(c, __gcd(i, n));
11     if (n & 1)
12         ans += n * Pow(c, n + 1 >> 1);
13     else
14         ans += n / 2 * (1 + c) * Pow(c, n >> 1);
15     return ans / n / 2;
16 }
```

3.6 快速乘 + 快速幂

```
1 ll Mul(ll a, ll b, ll mod) {
2     ll t = 0;
3     for (; b >>= 1; a = (a << 1) % mod)
4         if (b & 1) t = (t + a) % mod;
5     return t;
6 }
7
8 ll Pow(ll a, ll n, ll mod) {
9     ll t = 1;
10    for (; n >>= 1; a = (a * a % mod))
11        if (n & 1) t = (t * a % mod);
12    return t;
13 }
```

3.7 莫比乌斯反演

3.7.1 莫比乌斯

```
1 //  $F(n) = \sum_{d|n} f(d) \Rightarrow f(n) = \sum_{d|n} \mu(d) F(\frac{n}{d})$ 
2 //  $F(n) = \sum_{n|d} f(d) \Rightarrow f(n) = \sum_{n|d} \mu(\frac{d}{n}) F(d)$ 
3 long long ans;
4 const int MAXN = 1e5 + 1;
5 int n, x, prime[MAXN], tot, mu[MAXN];
6 bool check[MAXN];
7 void calmu() {
8     mu[1] = 1;
9     for (int i = 2; i < MAXN; i++) {
10         if (!check[i]) {
11             prime[tot++] = i;
12             mu[i] = -1;
13         }
14         for (int j = 0; j < tot; j++) {
15             if (i * prime[j] >= MAXN) break;
16             check[i * prime[j]] = true;
```

```

17         if (i % prime[j] == 0) {
18             mu[i * prime[j]] = 0;
19             break;
20         } else {
21             mu[i * prime[j]] = -mu[i];
22         }
23     }
24 }
25 }

```

3.7.2 n 个数中互质数对数

```

1 // 有  $n$  个数 ( $n \leq 10^5$ ), 问这  $n$  个数中互质的数的对数
2 #include <cstdio>
3 #include <cstring>
4 #include <cstdlib>
5 using namespace std;
6 long long ans;
7 const int MAXN = 1e5 + 1;
8 int n, x, prime[MAXN], _max, b[MAXN], tot, mu[MAXN];
9 bool check[MAXN];
10 void calmu() {
11     mu[1] = 1;
12     for (int i = 2; i < MAXN; i++) {
13         if (!check[i]) {
14             prime[tot++] = i;
15             mu[i] = -1;
16         }
17         for (int j = 0; j < tot; j++) {
18             if (i * prime[j] >= MAXN) break;
19             check[i * prime[j]] = true;
20             if (i % prime[j] == 0) {
21                 mu[i * prime[j]] = 0;
22                 break;
23             } else {
24                 mu[i * prime[j]] = -mu[i];
25             }
26         }
27     }
28 }
29 int main() {
30     calmu();
31     while (scanf("%d", &n) == 1) {
32         memset(b, 0, sizeof(b));
33         _max = 0; ans = 0;
34         for (int i = 0; i < n; i++) {
35             scanf("%d", &x);
36             if (x > _max) _max = x;
37             b[x]++;
38         }
39         int cnt;
40         for (int i = 1; i <= _max; i++) {
41             cnt = 0;
42             for (long long j = i; j <= _max; j += i)
43                 cnt += b[j];

```

```

44         ans += 1LL * mu[i] * cnt * cnt;
45     }
46     printf("%lld\n", (ans - b[1]) / 2);
47 }
48 return 0;
49 }

```

3.7.3 VisibleTrees

```

1 // gcd(x,y)==1 的对数  $x \leq n, y \leq m$ 
2 int main() {
3     calmu();
4     int n, m;
5     scanf("%d %d", &n, &m);
6     if (n < m) swap(n, m);
7     ll ans = 0;
8     for (int i = 1; i <= m; ++i) {
9         ans += (ll)mu[i] * (n / i) * (m / i);
10    }
11    printf("%lld\n", ans);
12    return 0;
13 }

```

3.8 其他

3.8.1 Josephus 问题

```

1 #include <iostream>
2 using namespace std;
3 int main() {
4     int num, m, r
5     while (cin >> num >> m) {
6         r = 0;
7         for (int k = 1; k <= num; ++k)
8             r = (r + m) % k;
9         cout << r + 1 << endl;
10    }
11    return 0;
12 }

```

3.8.2 数位问题

```

1 //  $n^n$  最左边一位数
2 int leftmost(int n) {
3     double m = n * log10((double)n);
4     double g = m - (long long)m;
5     g = pow(10.0, g);
6     return (int)g;
7 }
8
9 //  $n!$  位数
10 int count(ll n) {
11     return n == 1 ? 1 : (int)ceil(0.5 * log10(2 * M_PI * n) + n * log10(n)
12     ↪ - n * log10(M_E));
13 }

```

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 $\gcd(n, n-i) = 1 (1 \leq i \leq n)$

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： $p \text{ is prime} \Rightarrow (p-1)! \equiv -1 \pmod{p}$

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

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

素数定理：对于不大于 n 的素数个数 $\pi(n)$ ， $\lim_{n \rightarrow \infty} \frac{\pi(n)}{n} = \frac{1}{\ln n}$

位数公式：正整数 x 的位数 $N = \log_{10}(n) + 1$

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 $\gcd(a^m - b^m, a^n - b^n) = a^{\gcd(m, n)} - b^{\gcd(m, n)}$

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

$\gcd(\text{Fib}(m), \text{Fib}(n)) = \text{Fib}(\gcd(m, n))$

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 $N = \frac{(m-1)(n-1)}{2}$

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

若 p 为素数，则 $(x+y+\dots+w)^p \equiv x^p + y^p + \dots + w^p \pmod{p}$

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

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

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

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

4 数据结构

4.1 树状数组

```
1 // O(log n) 查询和修改数组的前缀和
2 // 注意下标应从 1 开始 n 是全局变量
3 int bit[N], n;
4 int sum(int i){
5     int s = 0;
6     while(i){
7         s += bit[i];
8         i -= i&-i;
9     }
10    return s;
11 }
12 void add(int i, int x){
13     while(i<=n){
14         bit[i] += x;
15         i += i&-i;
16     }
17 }
```

4.2 线段树

4.2.1 声明

```
1 #define lson rt<<1           // 左儿子
2 #define rson rt<<1|1       // 右儿子
3 #define Lson l,m,lson      // 左子树
4 #define Rson m+1,r,rson    // 右子树
5 void PushUp(int rt);         // 用 lson 和 rson 更新 rt
6 void PushDown(int rt[, int m]); // rt 的标记下移, m 为区间长度 (若与标记有
   ↳ 关)
7 void build(int l, int r, int rt); // 以 rt 为根节点, 对区间 [l, r] 建立线
   ↳ 段树
8 void update([...,] int l, int r, int rt) // rt[l, r] 内寻找目标并更新
9 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

```
1 const int maxn = 50010;
2 int sum[maxn << 2];
3 void PushUp(int rt) {
4     sum[rt] = sum[lson] + sum[rson];
5 }
6 void build(int l, int r, int rt) {
7     if (l == r) {scanf("%d", &sum[rt]); return;} // 建立的时候直接输
   ↳ 入叶节点
8     int m = (l + r) >> 1;
9     build(Lson); build(Rson);
10    PushUp(rt);
11 }
12 void update(int p, int add, int l, int r, int rt) {
13     if (l == r) {sum[rt] += add; return;}
14     int m = (l + r) >> 1;
15     if (p <= m) update(p, add, Lson);
```

```

16         else update(p, add, Rson);
17         PushUp(rt);
18     }
19     int query(int L, int R, int l, int r, int rt) {
20         if (L <= l && r <= R) {return sum[rt];}
21         int m = (l + r) >> 1, s = 0;
22         if (L <= m) s += query(L, R, Lson);
23         if (m < R) s += query(L, R, Rson);
24         return s;
25     }

```

4.2.3 区间更新-区间查询

```

1  // seg[rt] 用于存放懒惰标记, 注意 PushDown 时标记的传递
2  const int maxn = 101010;
3  int seg[maxn << 2], sum[maxn << 2];
4  void PushUp(int rt) {
5      sum[rt] = sum[lson] + sum[rson];
6  }
7  void PushDown(int rt, int m) {
8      if (seg[rt] == 0) return;
9      seg[lson] += seg[rt];
10     seg[rson] += seg[rt];
11     sum[lson] += seg[rt] * (m - (m >> 1));
12     sum[rson] += seg[rt] * (m >> 1);
13     seg[rt] = 0;
14 }
15 void build(int l, int r, int rt) {
16     seg[rt] = 0;
17     if (l == r) {scanf("%lld", &sum[rt]); return;}
18     int m = (l + r) >> 1;
19     build(Lson); build(Rson);
20     PushUp(rt);
21 }
22 void update(int L, int R, int add, int l, int r, int rt) {
23     if (L <= l && r <= R) {
24         seg[rt] += add;
25         sum[rt] += add * (r - l + 1);
26         return;
27     }
28     PushDown(rt, r - l + 1);
29     int m = (l + r) >> 1;
30     if (L <= m) update(L, R, add, Lson);
31     if (m < R) update(L, R, add, Rson);
32     PushUp(rt);
33 }
34 int query(int L, int R, int l, int r, int rt) {
35     if (L <= l && r <= R) return sum[rt];
36     PushDown(rt, r - l + 1);
37     int m = (l + r) >> 1, ret = 0;
38     if (L <= m) ret += query(L, R, Lson);
39     if (m < R) ret += query(L, R, Rson);
40     return ret;
41 }

```

4.3 字典树

```
1  struct Node {
2      char c;
3      Node* next[26];
4      Node(char cc) {
5          c = cc;
6          REP(i, 26)next[i] = NULL;
7      }
8      ~Node() {
9          REP(i, 26) if (next[i] != NULL) {
10             next[i]->~Node();
11             delete next[i];
12             next[i] = NULL;
13         }
14     }
15     bool empty() {
16         REP(i, 26)if (next[i])return 0;
17         return 1;
18     }
19 };
20
21 class Trie {
22 public:
23     Node *rt;
24     Trie() {
25         rt = new Node('*');
26     }
27     ~Trie() {
28         rt->~Node();
29     }
30     void insert(char s[]) {
31         Node *p = rt;
32         for (int i = 0; s[i]; i++) {
33             int d = s[i] - 'A';
34             if (!p->next[d])
35                 p->next[d] = new Node(s[i]);
36             p = p->next[d];
37         }
38     }
39     int find(char s[]) {
40         Node *p = rt;
41         for (int i = 0; s[i]; i++) {
42             int d = s[i] - 'A';
43             if (!p->next[d]) return 0;
44             p = p->next[d];
45         }
46         return 1;
47     }
48     void remove(char s[]) {
49         stack<Node*> st;
50         Node *pp = rt;
51         for (int i = 0; s[i]; i++) {
52             int d = s[i] - 'A';
53             if (!pp->next[d]) return;
```

```

54         st.push(pp);
55         pp = pp->next[d];
56     }
57     pp->~Node();
58     while (!st.empty()) {
59         Node *p = st.top(); st.pop();
60         p->next[pp->c - 'A'] = NULL;
61         pp = p;
62         bool f = 1;
63         REP(i, 26) if (p->next[i]) f = 0;
64         if (f) {
65             p->~Node();
66             if (!st.empty()) st.top()->next[p->c - 'A'] = NULL;
67         }
68         else break;
69     }
70     if (rt == NULL) rt = new Node('*');
71 }
72 };

```

4.4 RMQ

```

1  const int MAXN = 200000 + 100;
2  int mmax[MAXN][30], mmin[MAXN][30];
3  int a[MAXN], n, k;
4
5  void init() {
6      for (int i = 1; i <= n; i++) {
7          mmax[i][0] = mmin[i][0] = a[i];
8      }
9      for (int j = 1; (1 << j) <= n; j++)
10         for (int i = 1; i + (1 << j) - 1 <= n; i++) {
11             mmax[i][j] = max(mmax[i][j - 1], mmax[i + (1 << (j - 1))][j - 1]);
12             mmin[i][j] = min(mmin[i][j - 1], mmin[i + (1 << (j - 1))][j - 1]);
13         }
14 }
15
16 // op=0/1 返回 [l,r] 最大/小值
17 int rmq(int l, int r, int op) {
18     int k = 0;
19     while ((1 << (k + 1)) <= r - l + 1) k++;
20     if (op == 0) return max(mmax[l][k], mmax[r - (1 << k) + 1][k]);
21     return min(mmin[l][k], mmin[r - (1 << k) + 1][k]);
22 }

```

5 图论

5.1 并查集

```

1  const int MAXN = 128;
2  int n, fa[MAXN], ra[MAXN];
3  void init() {
4      for (int i = 0; i <= n; i++) {
5          fa[i] = i; ra[i] = 0;
6      }

```



```

7 }
8 int find(int x) {
9     if (fa[x] != x) fa[x] = find(fa[x]);
10    return fa[x];
11 }
12 void unite(int x, int y) {
13     x = find(x); y = find(y); if (x == y) return;
14     if (ra[x] < ra[y]) fa[x] = y;
15     else {
16         fa[y] = x; if (ra[x] == ra[y]) ra[x]++;
17     }
18 }
19 bool same(int x, int y) {
20     return find(x) == find(y);
21 }

```

5.2 最小生成树

5.2.1 Kruskal

```

1 vector<pair<int, PII>> G;
2 void add_edge(int u, int v, int d) {
3     G.pb(mp(d, mp(u, v)));
4 }
5 int Kruskal(int n) {
6     init(n);
7     sort(G.begin(), G.end());
8     int m = G.size();
9     int num = 0, ret = 0;
10    for (int i = 0; i < m; i++) {
11        pair<int, PII> p = G[i];
12        int x = p.Y.X;
13        int y = p.Y.Y;
14        int d = p.X;
15        if (!same(x, y)) {
16            unite(x, y);
17            num++;
18            ret += d;
19        }
20        if (num == n - 1) break;
21    }
22    return ret;
23 }

```

5.2.2 Prim

```

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

```

```

11     for (int i = 1; i < n; i++)
12         lowc[i] = cost[0][i];
13     for (int i = 1; i < n; i++) {
14         int minc = INF;
15         int p = -1;
16         for (int j = 0; j < n; j++)
17             if (!vis[j] && minc > lowc[j]) {
18                 minc = lowc[j];
19                 p = j;
20             }
21         if (minc == INF) return -1;
22         vis[p] = 1;
23         for (int j = 0; j < n; j++)
24             if (!vis[j] && lowc[j] > cost[p][j]) lowc[j] =
↪ cost[p][j];
25     }
26     return ans;
27 }

```

5.3 最短路

5.3.1 Dijkstra-邻接矩阵

```

1 // N 为点数最大值 求 s 到所有点的最短路
2 // 要求边权值为非负数 模板为有向边
3 // dis[x] 为起点到点 x 的最短路 inf 表示无法走到
4 // 记得初始化
5 const int N = 100;           // 点数最大值
6 const int INF = 0x3f3f3f3f;
7 int G[N][N], dis[N];
8 bool vis[N];
9 void init(int n) {
10     set(G, 0x3f);
11 }
12 void add_edge(int u, int v, int w) {
13     G[u][v] = min(G[u][v], w);
14 }
15 void Dijkstra(int s, int n) {
16     set(vis, 0);
17     set(dis, 0x3f);
18     dis[s] = 0;
19     for (int i = 0; i < n; i++) {
20         int x, minDis = INF;
21         for (int j = 0; j < n; j++) {
22             if (!vis[j] && dis[j] <= minDis) {
23                 x = j;
24                 minDis = dis[j];
25             }
26         }
27         vis[x] = 1;
28         for (int j = 0; j < n; j++)
29             dis[j] = min(dis[j], dis[x] + G[x][j]);
30     }
31 }

```

5.3.2 Dijkstra-邻接表数组

```
1 // 点最大值: MAX_N 边最大值: MAX_E
2 // 求起点 s 到每个点 x 的最短路 dis[x]
3 const int MAX_N = "Edit"; // 点数最大值
4 const int MAX_E = "Edit";
5 const int INF = 0x3F3F3F3F;
6 int tot;
7 int Head[MAX_N], vis[MAX_N], dis[MAX_N];
8 int Next[MAX_E], To[MAX_E], W[MAX_E];
9 void init() {
10     tot = 0;
11     memset(Head, -1, sizeof(Head));
12 }
13 void add_edge(int u, int v, int d) {
14     W[tot] = d;
15     To[tot] = v;
16     Next[tot] = Head[u];
17     Head[u] = tot++;
18 }
19 void Dijkstra(int s, int n) {
20     memset(vis, 0, sizeof(vis));
21     memset(dis, 0x3F, sizeof(dis));
22     dis[s] = 0;
23     for (int i = 0; i < n; i++) {
24         int x, min_dis = INF;
25         for (int j = 0; j < n; j++) {
26             if (!vis[j] && dis[j] <= min_dis) {
27                 x = j;
28                 min_dis = dis[j];
29             }
30         }
31         vis[x] = 1;
32         for (int j = Head[x]; j != -1; j = Next[j]) {
33             int y = To[j];
34             dis[y] = min(dis[y], dis[x] + W[j]);
35         }
36     }
37 }
```

5.3.3 Dijkstra-邻接表向量

```
1 // MAXN: 点数最大值
2 // 求起点 s 到所有点 x 的最短路 dis[x]
3 // 记得初始化
4 const int MAXN = "Edit";
5 const int INF = 0x3F3F3F3F;
6 vector<int> G[MAXN];
7 vector<int> GW[MAXN];
8 bool vis[MAXN];
9 int dis[MAXN];
10 void init(int n) {
11     for (int i = 0; i < n; i++) {
12         G[i].clear();
13         GW[i].clear();
14     }
```

```

15 }
16 void add_edge(int u, int v, int w) {
17     G[u].push_back(v);
18     GW[u].push_back(w);
19 }
20 void Dijkstra(int s, int n) {
21     memset(vis, false, sizeof(vis));
22     memset(dis, 0x3F, sizeof(dis));
23     dis[s] = 0;
24     for (int i = 0; i < n; i++) {
25         int x;
26         int min_dis = INF;
27         for (int j = 0; j < n; j++) {
28             if (!vis[j] && dis[j] <= min_dis) {
29                 x = j;
30                 min_dis = dis[j];
31             }
32         }
33         vis[x] = true;
34         for (int j = 0; j < (int)G[x].size(); j++) {
35             int y = G[x][j];
36             int w = GW[x][j];
37             dis[y] = min(dis[y], dis[x] + w);
38         }
39     }
40 }

```

5.3.4 Dijkstra-优先队列

```

1 // pair< 权值, 点 >
2 // 记得初始化
3 const int MAXN = "Edit";
4 const int INF = 0x3F3F3F3F;
5 typedef pair<int, int> PII;
6 typedef vector<PII> VII;
7 VII G[MAXN];
8 int vis[MAXN], dis[MAXN];
9 void init(int n) {
10     for (int i = 0; i < n; i++)
11         G[i].clear();
12 }
13 void add_edge(int u, int v, int w) {
14     G[u].push_back(make_pair(w, v));
15 }
16 void Dijkstra(int s, int n) {
17     memset(vis, 0, sizeof(vis));
18     memset(dis, 0x3F, sizeof(dis));
19     dis[s] = 0;
20     priority_queue<PII, VII, greater<PII> > q;
21     q.push(make_pair(dis[s], s));
22     while (!q.empty()) {
23         PII t = q.top();
24         int x = t.second;
25         q.pop();
26         if (vis[x]) continue;

```

```

27         vis[x] = 1;
28         for (int i = 0; i < (int)G[x].size(); i++) {
29             int y = G[x][i].second;
30             int w = G[x][i].first;
31             if (!vis[y] && dis[y] > dis[x] + w) {
32                 dis[y] = dis[x] + w;
33                 q.push(make_pair(dis[y], y));
34             }
35         }
36     }
37 }

```

5.3.5 Bellman-Ford(可判负环)

```

1 // 求出起点 s 到每个点 x 的最短路 dis[x]
2 // 存在负环返回 1 否则返回 0
3 // 记得初始化
4 const int MAX_N = "Edit";          // 点数最大值
5 const int MAX_E = "Edit";          // 边数最大值
6 const int INF = 0x3F3F3F3F;
7 int From[MAX_E], To[MAX_E], W[MAX_E];
8 int dis[MAX_N], tot;
9 void init() {tot = 0;}
10 void add_edge(int u, int v, int d) {
11     From[tot] = u;
12     To[tot] = v;
13     W[tot++] = d;
14 }
15 bool Bellman_Ford(int s, int n) {
16     memset(dis, 0x3F, sizeof(dis));
17     dis[s] = 0;
18     for (int k = 0; k < n - 1; k++) {
19         bool relaxed = 0;
20         for (int i = 0; i < tot; i++) {
21             int x = From[i], y = To[i];
22             if (dis[y] > dis[x] + W[i]) {
23                 dis[y] = dis[x] + W[i];
24                 relaxed = 1;
25             }
26         }
27         if (!relaxed) break;
28     }
29     for (int i = 0; i < tot; i++)
30         if (dis[To[i]] > dis[From[i]] + W[i])
31             return 1;
32     return 0;
33 }

```

5.3.6 SPFA

```

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

```

```

6  bool vis[MAXN];
7  int dis[MAXN];
8  int inqueue[MAXN];
9  void init(int n) {
10     for (int i = 0; i < n; i++)
11         G[i].clear();
12 }
13 void add_edge(int u, int v, int w) {
14     G[u].push_back(make_pair(v, w));
15 }
16 bool SPFA(int s, int n) {
17     memset(vis, 0, sizeof(vis));
18     memset(dis, 0x3F, sizeof(dis));
19     memset(inqueue, 0, sizeof(inqueue));
20     dis[s] = 0;
21     queue<int> q;          // 待优化的节点入队
22     q.push(s);
23     while (!q.empty()) {
24         int x = q.front();
25         q.pop();
26         vis[x] = false;
27         for (int i = 0; i < G[x].size(); i++) {
28             int y = G[x][i].first;
29             int w = G[x][i].second;
30             if (dis[y] > dis[x] + w) {
31                 dis[y] = dis[x] + w;
32                 if (!vis[y]) {
33                     q.push(y);
34                     vis[y] = true;
35                     if (++inqueue[y] >= n) return 0;
36                 }
37             }
38         }
39     }
40     return 1;
41 }

```

5.3.7 Floyd 算法

```

1  //  $O(n^3)$  求出任意两点间最短路
2  const int MAXN = "Edit";
3  const int INF = 0x3F3F3F3F;
4  int G[MAXN][MAXN];
5  void init(int n) {
6     memset(G, 0x3F, sizeof(G));
7     for (int i = 0; i < n; i++)
8         G[i][i] = 0;
9 }
10 void add_edge(int u, int v, int w) {
11     G[u][v] = min(G[u][v], w);
12 }
13 void Floyd(int n) {
14     for (int k = 0; k < n; k++)
15         for (int i = 0; i < n; i++)
16             for (int j = 0; j < n; j++)

```

```

17         G[i][j] = min(G[i][j], G[i][k] + G[k][j]);
18     }

```

5.4 拓扑排序

5.4.1 邻接矩阵

```

1  // 存图前记得初始化
2  // Ans 存放拓排结果, G 为邻接矩阵, deg 为入度信息
3  // 排序成功返回 1, 存在环返回 0
4  const int MAXN = "Edit";
5  int Ans[MAXN];           // 存放拓扑排序结果
6  int G[MAXN][MAXN];       // 存放图信息
7  int deg[MAXN];           // 存放点入度信息
8  void init() {
9      memset(G, 0, sizeof(G));
10     memset(deg, 0, sizeof(deg));
11     memset(Ans, 0, sizeof(Ans));
12 }
13 void add_edge(int u, int v) {
14     if (G[u][v]) return;
15     G[u][v] = 1;
16     deg[v]++;
17 }
18 bool Toposort(int n) {
19     int tot = 0;
20     queue<int> que;
21     for (int i = 0; i < n; ++i)
22         if (deg[i] == 0) que.push(i);
23     while (!que.empty()) {
24         int v = que.front(); que.pop();
25         Ans[tot++] = v;
26         for (int i = 0; i < n; ++i)
27             if (G[v][i] == 1)
28                 if (--deg[i] == 0) que.push(i);
29     }
30     if (tot < n) return false;
31     return true;
32 }

```

5.4.2 邻接表

```

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

```

```

15 }
16 void add_edge(int u, int v) {
17     if (S[make_pair(u, v)]) return;
18     G[u].push_back(v);
19     S[make_pair(u, v)] = 1;
20     deg[v]++;
21 }
22 bool Toposort(int n) {
23     int tot = 0; queue<int> que;
24     for (int i = 0; i < n; ++i)
25         if (deg[i] == 0) que.push(i);
26     while (!que.empty()) {
27         int v = que.front(); que.pop();
28         Ans[tot++] = v;
29         for (int i = 0; i < G[v].size(); ++i) {
30             int t = G[v][i];
31             if (--deg[t] == 0) que.push(t);
32         }
33     }
34     if (tot < n) return false;
35     return true;
36 }

```

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

```

1  #define MAXN 200
2  struct stack {
3      int top, node[MAXN];
4  } s;
5
6  int G[MAXN][MAXN]; // 邻接矩阵
7  int n; // 顶点个数
8
9  void dfs(int x) {
10     int i;

```



```

11     s.node[++s.top] = x;
12     for (int i = 0; i < n; i++)
13         if (G[i][x] > 0) {
14             G[i][x] = G[x][i] = 0;
15             dfs(i);
16             break;
17         }
18 }
19 void Fleury(int x) {
20     int i, b;
21     s.node[s.top = 0] = x;
22     while (s.top >= 0) {
23         b = 0;
24         for (int i = 0; i < n; i++)
25             if (G[s.node[s.top]][i] > 0) {
26                 b = 1;
27                 break;
28             }
29         if (b == 0) {
30             printf("%d ", s.node[s.top] + 1);
31             s.top--;
32         }
33         else {
34             s.top--;
35             dfs(s.node[s.top] + 1);
36         }
37     }
38     printf("\n");
39 }
40
41 int main() {
42     int i, j;
43     int m, s, t; // 边数, 读入的边的起点和终点
44     int degree, num, start; // 每个顶点的度、奇度顶点个数、欧拉回路的起点
45     scanf("%d%d", &n, &m);
46     set(G, 0);
47     for (i = 0; i < m; i++) {
48         scanf("%d%d", &s, &t)
49         G[s - 1][t - 1] = G[t - 1][s - 1] = 1;
50     }
51     num = 0; start = 0;
52     for (i = 0; i < n; i++) {
53         degree = 0;
54         for (j = 0; j < n; j++)
55             degree += G[i][j];
56         if (degree & 1) {
57             start = i;
58             num++;
59         }
60     }
61     if (num == 0 || num == 2) Fleury(start);
62     else puts("No Euler path");
63     return 0;
64 }

```

6 计算几何

6.1 定义

```
1  #define eps 1e-8
2  #define pi M_PI
3  #define zero(x) ((fabs(x)<eps?1:0))
4  #define sgn(x) (fabs(x)<eps?0:((x)<0?-1:1))
5  #define mp make_pair
6  #define X first
7  #define Y second
8
9  struct point {
10     double x, y;
11     point(double a = 0, double b = 0) {x = a; y = b;}
12     point operator - (const point& b) const {
13         return point(x - b.x, y - b.y);
14     }
15     point operator + (const point &b) const {
16         return point(x + b.x, y + b.y);
17     }
18     // 两点是否重合
19     bool operator == (point& b) {
20         return zero(x - b.x) && zero(y - b.y);
21     }
22     // 点积 (以原点为基准)
23     double operator * (const point &b) const {
24         return x * b.x + y * b.y;
25     }
26     // 叉积 (以原点为基准)
27     double operator ^ (const point &b) const {
28         return x * b.y - y * b.x;
29     }
30     // 绕 P 点逆时针旋转 a 弧度后的点
31     point rotate(point b, double a) {
32         double dx, dy; (*this - b).split(dx, dy);
33         double tx = dx * cos(a) - dy * sin(a);
34         double ty = dx * sin(a) + dy * cos(a);
35         return point(tx, ty) + b;
36     }
37     // 点坐标分别赋值到 a 和 b
38     void split(double &a, double &b) {
39         a = x; b = y;
40     }
41 };
42
43 struct line {
44     point s, e;
45     line() {}
46     line(point ss, point ee) {s = ss; e = ee;}
47 };
```

6.2 位置关系

6.2.1 两点间距离

```
1 double dist(point a, point b) {
2     return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y));
3 }
```

6.2.2 直线与直线的交点

```
1 // <0, *> 表示重合; <1, *> 表示平行; <2, P> 表示交点是 P;
2 pair<int, point> spoint(line l1, line l2) {
3     point res = l1.s;
4     if (sgn((l1.s.x - l1.e.x) * (l2.s.y - l2.e.y) - (l1.s.y - l1.e.y) * (l2.s.x - l2.e.x)) == 0)
5         return mp(1, res);
6     double t = ((l1.s.x - l2.s.x) * (l2.s.y - l2.e.y) - (l1.s.y - l1.e.y) * (l2.s.x - l2.e.x)) / ((l1.s.x - l1.e.x) * (l2.s.y - l2.e.y) - (l1.s.y - l1.e.y) * (l2.s.x - l2.e.x));
7     res.x += (l1.e.x - l1.s.x) * t;
8     res.y += (l1.e.y - l1.s.y) * t;
9     return mp(2, res);
10 }
```

6.2.3 判断线段与线段相交

```
1 bool segxseg(line l1, line l2) {
2     return
3         max(l1.s.x, l1.e.x) <= min(l2.s.x, l2.e.x) &&
4         max(l2.s.x, l2.e.x) <= min(l1.s.x, l1.e.x) &&
5         max(l1.s.y, l1.e.y) <= min(l2.s.y, l2.e.y) &&
6         max(l2.s.y, l2.e.y) <= min(l1.s.y, l1.e.y) &&
7         sgn((l2.s.x - l1.s.x) * (l1.e.y - l1.s.y) - (l1.s.x - l1.e.x) * (l2.s.y - l2.e.y)) <= 0 &&
8         sgn((l1.s.x - l2.s.x) * (l2.e.y - l2.s.y) - (l2.s.x - l2.e.x) * (l1.e.y - l1.s.y)) <= 0;
9 }
```

6.2.4 判断线段与直线相交

```
1 bool segxline(line l1, line l2) {
2     return sgn((l2.s.x - l1.s.x) * (l1.e.y - l1.s.y) - (l1.s.x - l1.e.x) * (l2.s.y - l2.e.y)) <= 0;
3 }
```

6.2.5 点到直线距离

```
1 point pointtoline(point P, line L) {
2     point res;
3     double t = ((P.x - L.s.x) * (L.e.y - L.s.y) - (P.y - L.s.y) * (L.e.x - L.s.x)) / ((L.e.x - L.s.x) * (L.e.y - L.s.y) - (L.e.y - L.s.y) * (L.e.x - L.s.x));
4     res.x = L.s.x + (L.e.x - L.s.x) * t;
5     res.y = L.s.y + (L.e.y - L.s.y) * t;
6     return res;
7 }
```

6.2.6 点到线段距离

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

```

3         double t = ((p - l.s) * (l.e-l.s)) / ((l.e-l.s) * (l.e-l.s));
4         if (t >= 0 && t <= 1) {
5             res.x = l.s.x + (l.e.x - l.s.x) * t;
6             res.y = l.s.y + (l.e.y - l.s.y) * t;
7         }
8         else res = dist(p, l.s) < dist(p, l.e) ? l.s : l.e;
9         return res;
10    }

```

6.2.7 点在线段上

```

1 bool PointOnSeg(point p, line l) {
2     return
3         sgn((l.s - p) ^ (l.e-p)) == 0 &&
4         sgn((p.x - l.s.x) * (p.x - l.e.x)) <= 0 &&
5         sgn((p.y - l.s.y) * (p.y - l.e.y)) <= 0;
6 }

```

6.3 多边形

6.3.1 多边形面积

```

1 double area(point p[], int n) {
2     double res = 0;
3     for (int i = 0; i < n; i++)
4         res += (p[i] ^ p[(i + 1) % n]) / 2;
5     return fabs(res);
6 }

```

6.3.2 点在凸多边形内

```

1 // 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）
2 // 点的编号：[0,n)
3 // -1：点在凸多边形外
4 // 0：点在凸多边形边界上
5 // 1：点在凸多边形内
6 int PointInConvex(point a, point p[], int n) {
7     for (int i = 0; i < n; i++) {
8         if (sgn((p[i] - a) ^ (p[(i + 1) % n] - a)) < 0)
9             return -1;
10        else if (PointOnSeg(a, line(p[i], p[(i + 1) % n])))
11            return 0;
12    }
13    return 1;
14 }

```

6.3.3 点在任意多边形内

```

1 // 射线法, poly[] 的顶点数要大于等于 3，点的编号 0~n-1
2 // -1：点在凸多边形外
3 // 0：点在凸多边形边界上
4 // 1：点在凸多边形内
5 int PointInPoly(point p, point poly[], int n) {
6     int cnt;
7     line ray, side;
8     cnt = 0;

```

```

9      ray.s = p;
10     ray.e.y = p.y;
11     ray.e.x = -1000000000000.0; // -INF, 注意取值防止越界
12     for (int i = 0; i < n; i++) {
13         side.s = poly[i];
14         side.e = poly[(i + 1) % n];
15         if (PointOnSeg(p, side)) return 0;
16         //如果平行轴则不考虑
17         if (sgn(side.s.y - side.e.y) == 0)
18             continue;
19         if (PointOnSeg(side.s, ray)) {
20             if (sgn(side.s.y - side.e.y) > 0) cnt++;
21         }
22         else if (PointOnSeg(side.e, ray)) {
23             if (sgn(side.e.y - side.s.y) > 0) cnt++;
24         }
25         else if (segxseg(ray, side)) cnt++;
26     }
27     return cnt % 2 == 1 ? 1 : -1;
28 }

```

6.3.4 判断凸多边形

```

1 // 点可以是顺时针给出也可以是逆时针给出
2 // 点的编号 1~n-1
3 bool isconvex(point poly[], int n) {
4     bool s[3];
5     set(s, 0);
6     for (int i = 0; i < n; i++) {
7         s[sgn((poly[(i + 1) % n] - poly[i]) ^ (poly[(i + 2) % n] -
8         ↪ poly[i])) + 1] = 1;
9         if (s[0] && s[2]) return 0;
10    }
11    return 1;
12 }

```

6.3.5 小结

```

1 #include <stdlib.h>
2 #include <math.h>
3 #define MAXN 1000
4 #define offset 10000
5 #define eps 1e-8
6 #define zero(x) (((x)>0?(x):-x)<eps)
7 #define _sign(x) ((x)>eps?1:((x)<-eps?-1:0))
8 struct point{double x,y;};
9 struct line{point a,b;};
10
11 double xmult(point p1,point p2,point p0){
12     return (p1.x-p0.x)*(p2.y-p0.y)-(p2.x-p0.x)*(p1.y-p0.y);
13 }
14
15 // 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
16 int is_convex(int n,point* p){
17     int i,s[3]={1,1,1};

```

```

18     for (i=0;i<n&&s[1]|s[2];i++)
19         s[_sign(xmult(p[(i+1)%n],p[(i+2)%n],p[i]))]=0;
20     return s[1]|s[2];
21 }
22
23 // 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
24 int is_convex_v2(int n,point* p){
25     int i,s[3]={1,1,1};
26     for (i=0;i<n&&s[0]&&s[1]|s[2];i++)
27         s[_sign(xmult(p[(i+1)%n],p[(i+2)%n],p[i]))]=0;
28     return s[0]&&s[1]|s[2];
29 }
30
31 // 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
32 int inside_convex(point q,int n,point* p){
33     int i,s[3]={1,1,1};
34     for (i=0;i<n&&s[1]|s[2];i++)
35         s[_sign(xmult(p[(i+1)%n],q,p[i]))]=0;
36     return s[1]|s[2];
37 }
38
39 // 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
40 int inside_convex_v2(point q,int n,point* p){
41     int i,s[3]={1,1,1};
42     for (i=0;i<n&&s[0]&&s[1]|s[2];i++)
43         s[_sign(xmult(p[(i+1)%n],q,p[i]))]=0;
44     return s[0]&&s[1]|s[2];
45 }
46
47 // 判点在任意多边形内，顶点按顺时针或逆时针给出
48 // 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
49 int inside_polygon(point q,int n,point* p,int on_edge=1){
50     point q2;
51     int i=0,count;
52     while (i<n)
53         for (count=i=0,q2.x=rand()+offset,q2.y=rand()+offset;i<n;i++)
54             if
55                 ↪ (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].y-q.y)<eps))
56                     return on_edge;
57                 else if (zero(xmult(q,q2,p[i])))
58                     break;
59                 else if
60                 ↪ (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)%n])<-eps)
61                     count++;
62     return count&1;
63 }
64
65 inline int opposite_side(point p1,point p2,point l1,point l2){
66     return xmult(l1,p1,l2)*xmult(l1,p2,l2)<-eps;
67 }
68
69 inline int dot_online_in(point p,point l1,point l2){
70     return
71     ↪ zero(xmult(p,l1,l2))&&(l1.x-p.x)*(l2.x-p.x)<eps&&(l1.y-p.y)*(l2.y-p.y)<eps;
72 }

```

```

70
71 // 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
72 int inside_polygon(point l1,point l2,int n,point* p){
73     point t[MAXN],tt;
74     int i,j,k=0;
75     if (!inside_polygon(l1,n,p)||!inside_polygon(l2,n,p))
76         return 0;
77     for (i=0;i<n;i++)
78         if
↪ (opposite_side(l1,l2,p[i],p[(i+1)%n])&&opposite_side(p[i],p[(i+1)%n],l1,l2))
79             return 0;
80     else if (dot_online_in(l1,p[i],p[(i+1)%n]))
81         t[k++]=l1;
82     else if (dot_online_in(l2,p[i],p[(i+1)%n]))
83         t[k++]=l2;
84     else if (dot_online_in(p[i],l1,l2))
85         t[k++]=p[i];
86     for (i=0;i<k;i++)
87         for (j=i+1;j<k;j++){
88             tt.x=(t[i].x+t[j].x)/2;
89             tt.y=(t[i].y+t[j].y)/2;
90             if (!inside_polygon(tt,n,p))
91                 return 0;
92         }
93     return 1;
94 }
95
96 point intersection(line u,line v){
97     point ret=u.a;
98     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))
99             /((u.a.x-u.b.x)*(v.a.y-v.b.y)-(u.a.y-u.b.y)*(v.a.x-v.b.x));
100     ret.x+=(u.b.x-u.a.x)*t;
101     ret.y+=(u.b.y-u.a.y)*t;
102     return ret;
103 }
104
105 point barycenter(point a,point b,point c){
106     line u,v;
107     u.a.x=(a.x+b.x)/2;
108     u.a.y=(a.y+b.y)/2;
109     u.b=c;
110     v.a.x=(a.x+c.x)/2;
111     v.a.y=(a.y+c.y)/2;
112     v.b=b;
113     return intersection(u,v);
114 }
115
116 // 多边形重心
117 point barycenter(int n,point* p){
118     point ret,t;
119     double t1=0,t2;
120     int i;
121     ret.x=ret.y=0;
122     for (i=1;i<n-1;i++)
123         if (fabs(t2=xmult(p[0],p[i],p[i+1]))>eps){

```

```

124         t=barycenter(p[0],p[i],p[i+1]);
125         ret.x+=t.x*t2;
126         ret.y+=t.y*t2;
127         t1+=t2;
128     }
129     if (fabs(t1)>eps)
130         ret.x/=t1,ret.y/=t1;
131     return ret;
132 }

```

6.4 整数点问题

6.4.1 线段上整点个数

```

1 int OnSegment(line l) {
2     return __gcd(fabs(l.s.x - l.e.x), fabs(l.s.y - l.e.y)) + 1;
3 }

```

6.4.2 多边形边上整点个数

```

1 int OnEdge(point p[], int n) {
2     int i, ret = 0;
3     for (i = 0; i < n; i++)
4         ret += __gcd(fabs(p[i].x - p[(i + 1) % n].x), fabs(p[i].y -
↪ p[(i + 1) % n].y));
5     return ret;
6 }

```

6.4.3 多边形内整点个数

```

1 int InSide(point p[], int n) {
2     int i, area = 0;
3     for (i = 0; i < n; i++)
4         area += p[(i + 1) % n].y * (p[i].x - p[(i + 2) % n].x);
5     return (fabs(area) - OnEdge(n, p)) / 2 + 1;
6 }

```

6.5 圆

```

1 point waixin(point a, point b, point c) {
2     double a1 = b.x - a.x, b1 = b.y - a.y, c1 = (a1 * a1 + b1 * b1) / 2;
3     double a2 = c.x - a.x, b2 = c.y - a.y, c2 = (a2 * a2 + b2 * b2) / 2;
4     double d = a1 * b2 - a2 * b1;
5     return point(a.x + (c1 * b2 - c2 * b1) / d, a.y + (a1 * c2 - a2 * c1) /
↪ d);
6 }

```

6.6 经典题

```

1 #include <cstdio>
2 #include <cmath>
3 #include <algorithm>
4 using namespace std;
5 const int N = 100100;
6 struct Point {
7     double x, y;

```



```

8  };
9  int n;
10 Point p[N], tmp[N];
11
12 bool cmp(Point a, Point b) {return a.x == b.x ? a.y < b.y : a.x < b.x;}
13 bool cmpy(Point a, Point b) {return a.y < b.y;}
14 double dis(Point a, Point b) {
15     double dx = a.x - b.x;
16     double dy = a.y - b.y;
17     return sqrt(dx * dx + dy * dy);
18 }
19 double solve(int l, int r) {
20     double d = 1e20;
21     if (l == r) return d;
22     if (l + 1 == r) return dis(p[l], p[r]);
23     int mid = l + r >> 1;
24     double d1 = solve(l, mid);
25     double d2 = solve(mid + 1, r);
26     d = min(d1, d2);
27     int k = 0;
28     for (int i = l; i <= r; i++)
29         if (fabs(p[i].x - p[mid].x) <= d)
30             tmp[k++] = p[i];
31     sort(tmp, tmp + k, cmpy);
32     for (int i = 0; i < k; i++)
33         for (int j = i + 1; j < k; j++) {
34             if (tmp[j].y - tmp[i].y > d) break;
35             d = min(d, dis(tmp[i], tmp[j]));
36         }
37     return d;
38 }
39 int main() {
40     while (scanf("%d", &n) && n != 0) {
41         for (int i = 0; i < n; i++)
42             scanf("%lf %lf", &p[i].x, &p[i].y);
43         sort(p, p + n, cmp);
44         printf("%.2lf\n", solve(0, n - 1) / 2);
45     }
46     return 0;
47 }

```

7 字符串

7.1 KMP

```

1  // 返回 y 中 x 的个数
2  int ne[N];
3  void initkmp(char x[], int m) {
4      int i, j; j = ne[0] = -1; i = 0;
5      while (i < m) {
6          while (j != -1 && x[i] != x[j])
7              j = ne[j];
8          ne[++i] = ++j;
9      }
10 }

```

```

11 int kmp(char x[], int m, char y[], int n) {
12     int i, j, ans; i = j = ans = 0;
13     initkmp(x, m);
14     while (i < n) {
15         while (j != -1 && y[i] != x[j]) j = ne[j];
16         i++; j++;
17         if (j >= m) {
18             ans++; j = ne[j];
19         }
20     }
21     return ans;
22 }

```

7.2 Manacher 最长回文子串

```

1 // O(n) 求解最长回文子串
2 const int N = 1000100;
3 char s[N], str[N << 1];
4 int p[N << 1];
5 void Manacher(char s[], int &n) {
6     str[0] = '$';
7     str[1] = '#';
8     for (int i = 0; i < n; i++) {
9         str[(i << 1) + 2] = s[i];
10        str[(i << 1) + 3] = '#';
11    }
12    n = 2 * n + 2;
13    str[n] = 0;
14    int mx = 0, id;
15    for (int i = 1; i < n; i++) {
16        p[i] = mx > i ? min(p[2 * id - i], mx - i) : 1;
17        while (str[i - p[i]] == str[i + p[i]]) p[i]++;
18        if (p[i] + i > mx) {
19            mx = p[i] + i;
20            id = i;
21        }
22    }
23 }
24 int solve(char s[]) {
25     int n = strlen(s);
26     Manacher(s, n);
27     int res = 0;
28     for (int i = 0; i < n; i++)
29         res = max(res, p[i]);
30     return res - 1;
31 }

```

7.3 AC 自动机

```

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

```

```

7     struct node {
8         node *go[AC_V], *fail, *fa;
9         int fg, id;
10    } pool[AC_N], *cur, *root, *q[AC_N];
11    node* newnode() {
12        node *p = cur++;
13        memset(p->go, 0, sizeof(p->go));
14        p->fail = p->fa = NULL; p->fg = 0;
15        return p;
16    }
17    void init() { cur = pool; root = newnode();}
18    node* append(node *p, int w) {
19        if (!p->go[w]) p->go[w] = newnode(), p->go[w]->fa = p;
20        return p = p->go[w];
21    }
22    void build() {
23        int t = 1;
24        q[0] = root;
25        rep(i, 0, t) rep(j, 0, AC_SIGMA) if (q[i]->go[j]) {
26            node *v = q[i]->go[j], *p = v->fa->fail;
27            while (p && !p->go[j]) p = p->fail;
28            if (p) v->fail = p->go[j]; else v->fail = root;
29            q[t++] = q[i]->go[j];
30        } else {
31            node *p = q[i]->fail;
32            while (p && !p->go[j]) p = p->fail;
33            if (p) q[i]->go[j] = p->go[j]; else q[i]->go[j] = root;
34        }
35    }
36    int query(char s[]) {
37        node *p = root;
38        int res = 0;
39        for (int i = 0; s[i]; i++) {
40            p = p->go[s[i] - 'a'];
41            node *v = p;
42            while (v != root) {
43                res += v->fg;
44                v->fg = 0;
45                v = v->fail;
46            }
47        }
48        return res;
49    }
50 } T;
51 typedef AC_automaton::node ACnode;
52
53
54 const int MAXN = 1000000 + 1000;
55 char txt[MAXN];
56
57 int main() {
58     #ifdef MANGOGAO
59         freopen("data.in", "r", stdin);
60     #endif
61

```

```

62     int t;
63     scanf("%d", &t);
64     while (t--) {
65         int n;
66         scanf("%d", &n);
67         T.init();
68         char s[55];
69         rep(i, 0, n) {
70             ACnode *p = T.root;
71             scanf("%s", s);
72             for (int j = 0; s[j]; j++)
73                 p = T.append(p, s[j] - 'a');
74             p->fg++;
75         }
76         T.build();
77         scanf("%s", txt);
78         printf("%d\n", T.query(txt));
79     }
80     return 0;
81 }

```

8 动态规划

8.1 最大子序列和

```

1 // 传入序列 a 和长度 n, 返回最大子序列和
2 // 限制最短长度: 用 cnt 记录长度, rt 更新时判断
3 int MaxSeqSum(int a[], int n) {
4     int rt = 0, cur = 0;
5     for (int i = 0; i < n; i++) {
6         cur += a[i];
7         rt = rt < cur ? cur : rt;
8         cur = cur < 0 ? 0 : cur;
9     }
10    return rt;
11 }

```

8.2 最长上升子序列 LIS

```

1 // 序列下标从 1 开始, LIS() 返回长度, 序列存在 lis[] 中
2 #define N 100100
3 int n, len, a[N], b[N], f[N];
4 int Find(int p, int l, int r) {
5     int mid;
6     while (l <= r) {
7         mid = (l + r) >> 1;
8         if (a[p] > b[mid]) l = mid + 1;
9         else r = mid - 1;
10    }
11    return f[p] = l;
12 }
13 int LIS(int lis[]) {
14     int len = 1;
15     f[1] = 1;
16     b[1] = a[1];

```

```

17     for (int i = 2; i <= n; i++) {
18         if (a[i] > b[len]) b[++len] = a[i], f[i] = len;
19         else b[Find(i, 1, len)] = a[i];
20     }
21     for (int i = n, t = len; i >= 1 && t >= 1; i--)
22         if (f[i] == t)
23             lis[--t] = a[i];
24     return len;
25 }

```

8.3 最长公共上升子序列 LCIS

```

1 // 序列下标从 1 开始
2 int LCIS(int a[], int b[], int n, int m) {
3     set(dp, 0);
4     for (int i = 1; i <= n; i++) {
5         int ma = 0;
6         for (int j = 1; j <= m; j++) {
7             dp[i][j] = dp[i - 1][j];
8             if (a[i] > b[j]) ma = max(ma, dp[i - 1][j]);
9             if (a[i] == b[j]) dp[i][j] = ma + 1;
10        }
11    }
12    return *max_element(dp[n] + 1, dp[n] + 1 + m);
13 }

```

9 附录

```

1 typedef long long ll;
2 namespace fastIO { //包含所有类型
3     #define BUF_SIZE 100000
4     #define OUT_SIZE 100000
5     #define ll long long
6     // fread->read
7     bool IOerror = 0;
8     inline char nc() {
9         static char buf[BUF_SIZE], *p1 = buf + BUF_SIZE, *pend = buf + BUF_SIZE;
10        if (p1 == pend) {
11            p1 = buf;
12            pend = buf + fread(buf, 1, BUF_SIZE, stdin);
13            if (pend == p1) {
14                IOerror = 1;
15                return -1;
16            }
17            //{printf("IO error!\n");system("pause");for (;;);exit(0);}
18        }
19        return *p1++;
20    }
21    inline bool blank(char ch) {
22        return ch == ' ' || ch == '\n' || ch == '\r' || ch == '\t';
23    }
24    inline void read(int &x) {
25        bool sign = 0;
26        char ch = nc();

```

```

27     x = 0;
28     for (; blank(ch); ch = nc())
29         ;
30     if (IOerror) return;
31     if (ch == '-') sign = 1, ch = nc();
32     for (; ch >= '0' && ch <= '9'; ch = nc()) x = x * 10 + ch - '0';
33     if (sign) x = -x;
34 }
35 inline void read(ll &x) {
36     bool sign = 0;
37     char ch = nc();
38     x = 0;
39     for (; blank(ch); ch = nc())
40         ;
41     if (IOerror) return;
42     if (ch == '-') sign = 1, ch = nc();
43     for (; ch >= '0' && ch <= '9'; ch = nc()) x = x * 10 + ch - '0';
44     if (sign) x = -x;
45 }
46 inline void read(double &x) {
47     bool sign = 0;
48     char ch = nc();
49     x = 0;
50     for (; blank(ch); ch = nc())
51         ;
52     if (IOerror) return;
53     if (ch == '-') sign = 1, ch = nc();
54     for (; ch >= '0' && ch <= '9'; ch = nc()) x = x * 10 + ch - '0';
55     if (ch == '.') {
56         double tmp = 1;
57         ch = nc();
58         for (; ch >= '0' && ch <= '9'; ch = nc()) tmp /= 10.0, x += tmp * (ch -
↪ '0');
59     }
60     if (sign) x = -x;
61 }
62 inline void read(char *s) {
63     char ch = nc();
64     for (; blank(ch); ch = nc())
65         ;
66     if (IOerror) return;
67     for (; !blank(ch) && !IOerror; ch = nc()) *s++ = ch;
68     *s = 0;
69 }
70 inline void read(char &c) {
71     for (c = nc(); blank(c); c = nc())
72         ;
73     if (IOerror) {
74         c = -1;
75         return;
76     }
77 }
78 // getchar->read
79 inline void read1(int &x) {
80     char ch;

```

```

81     int bo = 0;
82     x = 0;
83     for (ch = getchar(); ch < '0' || ch > '9'; ch = getchar())
84         if (ch == '-') bo = 1;
85     for (; ch >= '0' && ch <= '9'; x = x * 10 + ch - '0', ch = getchar())
86         ;
87     if (bo) x = -x;
88 }
89 inline void read1(ll &x) {
90     char ch;
91     int bo = 0;
92     x = 0;
93     for (ch = getchar(); ch < '0' || ch > '9'; ch = getchar())
94         if (ch == '-') bo = 1;
95     for (; ch >= '0' && ch <= '9'; x = x * 10 + ch - '0', ch = getchar())
96         ;
97     if (bo) x = -x;
98 }
99 inline void read1(double &x) {
100     char ch;
101     int bo = 0;
102     x = 0;
103     for (ch = getchar(); ch < '0' || ch > '9'; ch = getchar())
104         if (ch == '-') bo = 1;
105     for (; ch >= '0' && ch <= '9'; x = x * 10 + ch - '0', ch = getchar())
106         ;
107     if (ch == '.') {
108         double tmp = 1;
109         for (ch = getchar(); ch >= '0' && ch <= '9'; tmp /= 10.0, x += tmp *
↪ (ch - '0'), ch = getchar())
110             ;
111     }
112     if (bo) x = -x;
113 }
114 inline void read1(char *s) {
115     char ch = getchar();
116     for (; blank(ch); ch = getchar())
117         ;
118     for (; !blank(ch); ch = getchar()) *s++ = ch;
119     *s = 0;
120 }
121 inline void read1(char &c) {
122     for (c = getchar(); blank(c); c = getchar())
123         ;
124 }
125 // scanf->read
126 inline void read2(int &x) {
127     scanf("%d", &x);
128 }
129 inline void read2(ll &x) {
130     #ifdef _WIN32
131         scanf("%I64d", &x);
132     #else
133     #ifdef __linux
134         scanf("%lld", &x);

```

```

135  else
136      puts("error:can't recognize the system!");
137  endif
138  endif
139  }
140  inline void read2(double &x) {
141      scanf("%lf", &x);
142  }
143  inline void read2(char *s) {
144      scanf("%s", s);
145  }
146  inline void read2(char &c) {
147      scanf(" %c", &c);
148  }
149  inline void readln2(char *s) {
150      gets(s);
151  }
152  // fwrite->write
153  struct Ostream_fwrite {
154      char *buf, *p1, *pend;
155      Ostream_fwrite() {
156          buf = new char[BUF_SIZE];
157          p1 = buf;
158          pend = buf + BUF_SIZE;
159      }
160      void out(char ch) {
161          if (p1 == pend) {
162              fwrite(buf, 1, BUF_SIZE, stdout);
163              p1 = buf;
164          }
165          *p1++ = ch;
166      }
167      void print(int x) {
168          static char s[15], *s1;
169          s1 = s;
170          if (!x) *s1++ = '0';
171          if (x < 0) out('-'), x = -x;
172          while (x) *s1++ = x % 10 + '0', x /= 10;
173          while (s1-- != s) out(*s1);
174      }
175      void println(int x) {
176          static char s[15], *s1;
177          s1 = s;
178          if (!x) *s1++ = '0';
179          if (x < 0) out('-'), x = -x;
180          while (x) *s1++ = x % 10 + '0', x /= 10;
181          while (s1-- != s) out(*s1);
182          out('\n');
183      }
184      void print(ll x) {
185          static char s[25], *s1;
186          s1 = s;
187          if (!x) *s1++ = '0';
188          if (x < 0) out('-'), x = -x;
189          while (x) *s1++ = x % 10 + '0', x /= 10;

```



```

190     while (s1-- != s) out(*s1);
191 }
192 void println(ll x) {
193     static char s[25], *s1;
194     s1 = s;
195     if (!x) *s1++ = '0';
196     if (x < 0) out('-'), x = -x;
197     while (x) *s1++ = x % 10 + '0', x /= 10;
198     while (s1-- != s) out(*s1);
199     out('\n');
200 }
201 void print(double x, int y) {
202     static ll mul[] = {1, 10, 100, 1000, 10000, 100000, 1000000, 10000000,
↪ 100000000, 1000000000, 10000000000LL, 100000000000LL, 1000000000000LL,
↪ 10000000000000LL, 100000000000000LL, 1000000000000000LL,
↪ 10000000000000000LL, 100000000000000000LL};
203     if (x < -1e-12) out('-'), x = -x;
204     x *= mul[y];
205     ll x1 = (ll)floor(x);
206     if (x - floor(x) >= 0.5) ++x1;
207     ll x2 = x1 / mul[y], x3 = x1 - x2 * mul[y];
208     print(x2);
209     if (y > 0) {
210         out('.');
211         for (size_t i = 1; i < y && x3 * mul[i] < mul[y]; out('0'), ++i)
212             ;
213         print(x3);
214     }
215 }
216 void println(double x, int y) {
217     print(x, y);
218     out('\n');
219 }
220 void print(char *s) {
221     while (*s) out(*s++);
222 }
223 void println(char *s) {
224     while (*s) out(*s++);
225     out('\n');
226 }
227 void flush() {
228     if (p1 != buf) {
229         fwrite(buf, 1, p1 - buf, stdout);
230         p1 = buf;
231     }
232 }
233 ~Ostream_fwrite() {
234     flush();
235 }
236 } Ostream;
237 inline void print(int x) {
238     Ostream.print(x);
239 }
240 inline void println(int x) {
241     Ostream.println(x);

```

```

242 }
243 inline void print(char x) {
244     Ostream.out(x);
245 }
246 inline void println(char x) {
247     Ostream.out(x);
248     Ostream.out('\n');
249 }
250 inline void print(ll x) {
251     Ostream.print(x);
252 }
253 inline void println(ll x) {
254     Ostream.println(x);
255 }
256 inline void print(double x, int y) {
257     Ostream.print(x, y);
258 }
259 inline void println(double x, int y) {
260     Ostream.println(x, y);
261 }
262 inline void print(char *s) {
263     Ostream.print(s);
264 }
265 inline void println(char *s) {
266     Ostream.println(s);
267 }
268 inline void println() {
269     Ostream.out('\n');
270 }
271 inline void flush() {
272     Ostream.flush();
273 }
274 // puts->write
275 char Out[OUT_SIZE], *o = Out;
276 inline void print1(int x) {
277     static char buf[15];
278     char *p1 = buf;
279     if (!x) *p1++ = '0';
280     if (x < 0) *o++ = '-', x = -x;
281     while (x) *p1++ = x % 10 + '0', x /= 10;
282     while (p1-- != buf) *o++ = *p1;
283 }
284 inline void println1(int x) {
285     print1(x);
286     *o++ = '\n';
287 }
288 inline void print1(ll x) {
289     static char buf[25];
290     char *p1 = buf;
291     if (!x) *p1++ = '0';
292     if (x < 0) *o++ = '-', x = -x;
293     while (x) *p1++ = x % 10 + '0', x /= 10;
294     while (p1-- != buf) *o++ = *p1;
295 }
296 inline void println1(ll x) {

```

```

297     print1(x);
298     *o++ = '\n';
299 }
300 inline void print1(char c) {
301     *o++ = c;
302 }
303 inline void println1(char c) {
304     *o++ = c;
305     *o++ = '\n';
306 }
307 inline void print1(char *s) {
308     while (*s) *o++ = *s++;
309 }
310 inline void println1(char *s) {
311     print1(s);
312     *o++ = '\n';
313 }
314 inline void println1() {
315     *o++ = '\n';
316 }
317 inline void flush1() {
318     if (o != Out) {
319         if (*(o - 1) == '\n') *--o = 0;
320         puts(Out);
321     }
322 }
323 struct puts_write {
324     ~puts_write() {
325         flush1();
326     }
327 } _puts;
328 inline void print2(int x) {
329     printf("%d", x);
330 }
331 inline void println2(int x) {
332     printf("%d\n", x);
333 }
334 inline void print2(char x) {
335     printf("%c", x);
336 }
337 inline void println2(char x) {
338     printf("%c\n", x);
339 }
340 inline void print2(ll x) {
341     #ifdef _WIN32
342         printf("%I64d", x);
343     #else
344     #ifdef __linux
345         printf("%lld", x);
346     #else
347         puts("error:can't recognize the system!");
348     #endif
349 #endif
350 }
351 inline void println2(ll x) {

```

```
352     print2(x);
353     printf("\n");
354 }
355 inline void println2() {
356     printf("\n");
357 }
358 #undef ll
359 #undef OUT_SIZE
360 #undef BUF_SIZE
361 }; // namespace fastIO
362 using namespace fastIO;
```