

## Contents

1	Basic	1
1.1	/.bashrc	1
1.2	/.vimrc	1
1.3	Faster cin	1
1.4	BigInt	1
1.5	Random	2
2	Mathmatics	2
2.1	Theorem	2
2.2	Miller Rabin	2
2.3	ax+by=gcd(a,b)	3
2.4	FFT	3
2.5	Hash	3
2.6	GaussElimination	3
2.7	Inverse	3
2.8	IterSet	4
2.9	LinearPrime	4
2.10	SG	4
3	Geometry	4
3.1	2D Point Template	4
3.2	Intersection of two circle	4
3.3	Convex Hull	4
3.4	外心 Circumcentre	5
3.5	Smallest Covering Circle	5
4	Flow	5
4.1	Dinic	5
4.2	KM	6
4.3	KM	6
4.4	min cost max flow	7
5	Graph	7
5.1	Strongly Connected Component(SCC)	7
5.2	Euler Circuit	7
5.3	Hungarian	8
5.4	Maximum Clique	8
5.5	Tarjan	8
5.6	一般圖匹配	9
5.7	LCA	9
6	Data Structure	10
6.1	Disjoint Set	10
6.2	Sparse Table	10
6.3	Treap	10
7	String	10
7.1	KMP	10
7.2	AC	11
7.3	Z-value	11
7.4	Suffix Array	11
8	Dark Code	11
8.1	輸入優化	11
9	Search	
10	Others	
11	Persistence	

## Basic

```
1 /.bashrc
```

```
1 tabs -4
```

```
2 /.vimrc
```

```
2 set nu "行號"
2 set tabstop=4 "tab寬度"
2 set autoindent
2 set smartindent
2 set softtabstop=4
2 set shiftwidth=4
2 set cindent
4
4 se ai ar sm nu rnu is
4 se mouse=a bs=2 so=6 ts=4 ttm=100
4
4 nmap <F2> :! gedit %<.in <CR>
4 nmap <F4> :! date > %<.pt; cat -n % > %<.pt; lpr %<.pt
4 <CR>
4 nmap <F9> :! clear ; g++ -std=c++11 -Wall -O2 % -o %<.
4 out; ./%<.out < %<.in <CR>
4 nmap <F10> :! clear ; g++ -std=c++11 -Wall -O2 % -o
4 %<.out; ./%<.out <CR>
```

## Faster cin

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

int main(){
    ios_base::sync_with_stdio(false);
    cin.tie(0);
}
```

## BigInt

```
struct BigInt{
    static const int LEN = 60;
    static const int BIGMOD = 10000;
    int s;
    int vl, v[LEN];
    // vector<int> v;
    BigInt() : s(1) { vl = 0; }
    BigInt(long long a) {
        s = 1; vl = 0;
        if (a < 0) { s = -1; a = -a; }
        while (a) {
            push_back(a % BIGMOD);
            a /= BIGMOD;
        }
    }
    BigInt(string str) {
        s = 1; vl = 0;
        int stPos = 0, num = 0;
        if (!str.empty() && str[0] == '-') {
            stPos = 1;
            s = -1;
        }
        for (int i=SZ(str)-1, q=1; i>=stPos; i--) {
            num += (str[i] - '0') * q;
            if ((q *= 10) >= BIGMOD) {
                push_back(num);
                num = 0; q = 1;
            }
        }
        if (num) push_back(num);
    }
    int len() const { return vl; /* return SZ(v); */ }
    bool empty() const { return len() == 0; }
    void push_back(int x) { v[vl++] = x; /* v.PB(x); */ }
    void pop_back() { vl--; /* v.pop_back(); */ }
```

```

int back() const { return v[vl-1]; /* return v.back()
; */ }
void n() { while (!empty() && !back()) pop_back(); }
void resize(int nl) {
    vl = nl; fill(v, v+vl, 0);
    // v.resize(nl); // fill(ALL(v), 0);
}
void print() const {
    if (empty()) { putchar('0'); return; }
    if (s == -1) putchar('-');
    printf("%d", back());
    for (int i=len()-2; i>=0; i--) printf("%.4d",v[i]);
}
friend std::ostream& operator << (std::ostream& out,
    const Bigint &a) {
    if (a.empty()) { out << "0"; return out; }
    if (a.s == -1) out << "-";
    out << a.back();
    for (int i=a.len()-2; i>=0; i--) {
        char str[10];
        snprintf(str, 5, "%.4d", a.v[i]);
        out << str;
    }
    return out;
}
int cp3(const Bigint &b) const {
    if (s != b.s) return s > b.s ? 1 : -1;
    if (s == -1) return -(*this).cp3(-b);
    if (len() != b.len()) return len()>b.len()?1:-1;
    for (int i=len()-1; i>=0; i--)
        if (v[i]!=b.v[i]) return v[i]>b.v[i]?1:-1;
    return 0;
}
bool operator < (const Bigint &b) const { return cp3(b)
== -1; }
bool operator <= (const Bigint &b) const { return cp3(b)
>= 0; }
bool operator >= (const Bigint &b) const { return cp3(b)
>= 0; }
bool operator == (const Bigint &b) const { return cp3(b)
== 0; }
bool operator != (const Bigint &b) const { return cp3(b)
!= 0; }
bool operator > (const Bigint &b) const { return cp3(b)
== 1; }
Bigint operator - () const {
    Bigint r = (*this);
    r.s = -r.s;
    return r;
}
Bigint operator + (const Bigint &b) const {
    if (s == -1) return -(*this)+(-b);
    if (b.s == -1) return (*this)-(-b);
    Bigint r;
    int nl = max(len(), b.len());
    r.resize(nl + 1);
    for (int i=0; i<nl; i++) {
        if (i < len()) r.v[i] += v[i];
        if (i < b.len()) r.v[i] += b.v[i];
        if (r.v[i] >= BIGMOD) {
            r.v[i+1] += r.v[i] / BIGMOD;
            r.v[i] %= BIGMOD;
        }
    }
    r.n();
    return r;
}
Bigint operator - (const Bigint &b) const {
    if (s == -1) return -(*this)-(-b);
    if (b.s == -1) return (*this)+(-b);
    if ((*this) < b) return -(b-(*this));
    Bigint r;
    r.resize(len());
    for (int i=0; i<len(); i++) {
        r.v[i] += v[i];
        if (i < b.len()) r.v[i] -= b.v[i];
        if (r.v[i] < 0) {
            r.v[i] += BIGMOD;
            r.v[i+1]--;
        }
    }
    r.n();
}

```

```

return r;
}
Bigint operator * (const Bigint &b) {
    Bigint r;
    r.resize(len() + b.len() + 1);
    r.s = s * b.s;
    for (int i=0; i<len(); i++) {
        for (int j=0; j<b.len(); j++) {
            r.v[i+j] += v[i] * b.v[j];
            if (r.v[i+j] >= BIGMOD) {
                r.v[i+j+1] += r.v[i+j] / BIGMOD;
                r.v[i+j] %= BIGMOD;
            }
        }
    }
    r.n();
    return r;
}
Bigint operator / (const Bigint &b) {
    Bigint r;
    r.resize(max(1, len()-b.len()+1));
    int oriS = s;
    Bigint b2 = b; // b2 = abs(b)
    s = b2.s = r.s = 1;
    for (int i=r.len()-1; i>=0; i--) {
        int d=0, u=BIGMOD-1;
        while(d<u) {
            int m = (d+u+1)>>1;
            r.v[i] = m;
            if ((r*b2) > (*this)) u = m-1;
            else d = m;
        }
        r.v[i] = d;
    }
    s = oriS;
    r.s = s * b.s;
    r.n();
    return r;
}
Bigint operator % (const Bigint &b) {
    return (*this)-(*this)/b*b;
}
};

```

## Random

```

inline int ran(){
    static int x = 20167122;
    return x = (x * 0xdefaced + 1) & INT_MAX;
}

```

## Mathmatics

### Theorem

```

/*
Lucas's Theorem:
For non-negative integer n,m and prime P,
C(m,n) mod P = C(m/M,n/M) * C(m%M,n%M) mod P
= mult_i ( C(m_i,n_i) )
where m_i is the i-th digit of m in base P.
-----

```

```

Pick's Theorem
A = i + b/2 - 1
-----

```

```

Kirchhoff's theorem
A_{ii} = deg(i), A_{ij} = (i,j) \in E ? -1 : 0
Deleting any one row, one column, and cal the det(A)
*/

```

## Miller Rabin

```

typedef long long LL;

```

```

LL bin_pow(LL a, LL n, LL MOD){
    LL re=1;
    while (n>0){
        if (n&1)re = re*a %MOD;
        a = a*a %MOD;
        n>>=1;
    }
    return re;
}
bool is_prime(LL n){
    //static LL sprp[3] = { 2LL, 7LL, 61LL};
    static LL sprp[7] = { 2LL, 325LL, 9375LL,
        28178LL, 450775LL, 9780504LL,
        1795265022LL };
    if (n==1 || (n&1)==0 ) return n==2;
    int u=n-1, t=0;
    while ( (u&1)==0 ) u>>=1, t++;
    for (int i=0; i<7; i++){
        LL x = bin_pow( sprp[i]%n, u, n);
        if (x==0 || x==1 || x==n-1)continue;

        for (int j=1; j<t; j++){
            x=x*x%n;
            if (x==1 || x==n-1)break;
        }
        if (x==n-1)continue;
        return 0;
    }
    return 1;
}

```

### ax+by=gcd(a,b)

```

typedef pair<int, int> pii;
pii extgcd(int a, int b){
    if(b == 0) return make_pair(1, 0);
    else{
        int p = a / b;
        pii q = extgcd(b, a % b);
        return make_pair(q.second, q.first - q.second * p);
    }
}

```

### FFT

```

const double pi = atan(1.0)*4;
struct Complex {
    double x,y;
    Complex(double _x=0,double _y=0)
        :x(_x),y(_y) {}
    Complex operator + (Complex &tt) { return Complex(x
        +tt.x,y+tt.y); }
    Complex operator - (Complex &tt) { return Complex(x
        -tt.x,y-tt.y); }
    Complex operator * (Complex &tt) { return Complex(x
        *tt.x-y*tt.y,x*tt.y+y*tt.x); }
};
void fft(Complex *a, int n, int rev) {
    // n是大于等于相乘的两个数组长度的2的幂次
    // 从0开始表示长度，对a进行操作
    // rev==1进行DFT，== -1进行IDFT
    for (int i = 1, j = 0; i < n; ++ i) {
        for (int k = n>>1; k > (j^=k); k >>= 1);
        if (i<j) std::swap(a[i],a[j]);
    }
    for (int m = 2; m <= n; m <= 1) {
        Complex wm(cos(2*pi*rev/m),sin(2*pi*rev/m));
        for (int i = 0; i < n; i += m) {
            Complex w(1.0,0.0);
            for (int j = i; j < i+m/2; ++ j) {
                Complex t = w*a[j+m/2];
                a[j+m/2] = a[j] - t;
                a[j] = a[j] + t;
                w = w * wm;
            }
        }
    }
}

```

```

    if (rev== -1) {
        for (int i = 0; i < n; ++ i) a[i].x /= n,a[i].y
            /= n;
    }
}

```

### Hash

```

typedef long long LL;
LL X=7122;
LL P1=712271227;
LL P2=179433857;
LL P3=179434999;

struct HASH{
    LL a, b, c;
    HASH(LL a=0, LL b=0, LL c=0):a(a),b(b),c(c){ }
    HASH operator + (HASH B){
        return HASH((a+B.a)%P1,(b+B.b)%P2,(c+B.c)%P3);
    }
    HASH operator + (LL B){
        return (*this)+HASH(B,B,B);
    }
    HASH operator * (LL B){
        return HASH(a*B%P1,a*B%P2,a*B%P3);
    }
    bool operator < (const HASH &B)const{
        if (a!=B.a)return a<B.a;
        if (b!=B.b)return b<B.b;
        return c<B.c;
    }
    void up(){ (*this) = (*this)*X; }
};

int main(){
}

```

### GaussElimination

```

// by bcw_codebook

const int MAXN = 300;
const double EPS = 1e-8;

int n;
double A[MAXN][MAXN];

void Gauss() {
    for(int i = 0; i < n; i++) {
        bool ok = 0;
        for(int j = i; j < n; j++) {
            if(fabs(A[j][i]) > EPS) {
                swap(A[j], A[i]);
                ok = 1;
                break;
            }
        }
        if(!ok) continue;

        double fs = A[i][i];
        for(int j = i+1; j < n; j++) {
            double r = A[j][i] / fs;
            for(int k = i; k < n; k++) {
                A[j][k] -= A[i][k] * r;
            }
        }
    }
}

```

### Inverse

```

int inverse[100000];
void invTable(int b, int p) {
    inverse[1] = 1;
    for( int i = 2; i <= b; i++ ) {
        inverse[i] = (long long)inverse[p%i] * (p-p/i) % p;
    }
}

```

```

}
}

int inv(int b, int p) {
    return b == 1 ? 1 : ((long long)inv(p % b, p) * (p-p/
        b) % p);
}

```

## IterSet

```

// get all subset in set S

for (int i = S; i ; i = (i-1) & S ) {

}

```

## LinearPrime

```

const int MAXP = 100; //max prime
vector<int> P; // primes
void build_prime(){
    static bitset<MAXP> ok;
    int np=0;
    for (int i=2; i<MAXP; i++){
        if (ok[i]==0)P.push_back(i), np++;
        for (int j=0; j<np && i*P[j]<MAXP; j++){
            ok[ i*P[j] ] = 1;
            if ( i%P[j]==0 )break;
        }
    }
}

```

## SG

### Sprague-Grundy

1. 雙人、回合制
2. 資訊完全公開
3. 無隨機因素
4. 可在有限步內結束
5. 沒有和局
6. 雙方可採取的行動相同

SG(S) 的值為 0：後手(P)必勝  
不為 0：先手(N)必勝

```

int mex(set S) {
    // find the min number >= 0 that not in the S
    // e.g. S = {0, 1, 3, 4} mex(S) = 2
}

state = []
int SG(A) {
    if (A not in state) {
        S = sub_states(A)
        if( len(S) > 1 ) state[A] = reduce(operator.xor, [
            SG(B) for B in S])
        else state[A] = mex(set(SG(B) for B in next_states(
            A)))
    }
    return state[A]
}

```

## Geometry

### 2D Point Template

```

typedef double T;
struct Point {
    T x,y;

```

```

Point (T _x=0, T _y=0):x(_x),y(_y){}

bool operator < (const Point &b)const{
    return atan2(y,x) < atan2(b.y,b.x);
}
bool operator == (const Point &b)const{
    return atan2(y,x) == atan2(b.y,b.x);
}
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);
}
T operator * (const Point &b)const{
    return x*b.x + y*b.y;
}
T operator % (const Point &b)const{
    return x*b.y - y*b.x;
}
Point operator * (const T &d)const{
    return Point(d*x,d*y);
}
T abs2() { return x*x+y*y; }
T abs() { return sqrt( abs2() ); }
};
typedef Point pdd;
inline double abs2(pdd a){
    return a.abs2();
}

```

### Intersection of two circle

```

typedef Point pdd;
typedef ld double;
vector<pdd> interCircle(pdd o1, double r1, pdd o2,
    double r2) {
    ld d2 = (o1 - o2).abs2();
    ld d = sqrt(d2);
    if (d < fabs(r1-r2)) return {};
    if (d > r1+r2) return {};
    pdd u = 0.5*(o1+o2) + ((r2*r2-r1*r1)/(2.0*d2))*(o1-o2);
    double A = sqrt((r1+r2+d) * (r1-r2+d) * (r1+r2-d) *
        (-r1+r2+d));
    pdd v = A / (2.0*d2) * pdd(o1.S-o2.S, -o1.F+o2.F);
    return {u+v, u-v};
}

```

### Convex Hull

```

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

typedef long long LL;
const int MAXN = 100005;
const LL INF = (1LL)<<62;

struct Point{
    LL x, y;
    Point (LL x=0, LL y=0):x(x),y(y){}
    bool operator < (const Point &B)const {
        if (x!=B.x)return x<B.x;
        return y<B.y;
    }
    Point operator - (Point B){
        return Point(x-B.x,y-B.y);
    }
};

LL cross(Point A, Point B){
    return A.x*B.y-A.y*B.x;
}

LL Abs(LL x){
    return x>0?x:-x;
}

LL AreaU[MAXN], AreaD[MAXN];
void find_CH(int N, Point P[], LL Area[]){
    static vector<Point> U, D;

```

```

static vector<int> Ui, Di;
U.clear(), Ui.clear();
D.clear(), Di.clear();
int uz=0, dz=0;

for (int i=0; i<N; i++){
    while (uz>=2 && cross(P[i]-U[uz-2],U[uz-1]-U[uz-2])<=0)U.pop_back(), Ui.pop_back(), uz--;
    if (uz<=1)AreaU[i]=0;
    else AreaU[i] = AreaU[ Ui[uz-1] ] + Abs(cross(P[i]-U[0],U[uz-1]-U[0]));
    U.push_back(P[i]),Ui.push_back(i),uz++;

    while (dz>=2 && cross(P[i]-D[dz-2],D[dz-1]-D[dz-2])>=0)D.pop_back(), Di.pop_back(), dz--;
    if (dz<=1)AreaD[i]=0;
    else AreaD[i] = AreaD[ Di[dz-1] ] + Abs(cross(P[i]-D[0],D[dz-1]-D[0]));
    D.push_back(P[i]),Di.push_back(i),dz++;

    Area[i] = AreaU[i] + AreaD[i];
    //printf("Area[%d]=%lld\n",i ,Area[i]);
}
//puts("");
}

int N;
Point P[MAXN];
LL AreaL[MAXN], AreaR[MAXN];

int main(){
    input();

    find_CH(N,P,AreaL);
    for (int i=0; i<N; i++)P[i].x*=-1;
    reverse(P,P+N);
    find_CH(N,P,AreaR);
    reverse(AreaR,AreaR+N);
    reverse(P,P+N);

    LL Ans = min(AreaL[N-1],AreaR[0]);
    for (int i=0; i<N-1; i++){
        if (P[i].x!=P[i+1].x){
            Ans = min (Ans,AreaL[i]+AreaR[i+1]);
        }
    }
    if (P[0].x==P[N-1].x)Ans=0;
    printf("%lld\n", (Ans+1)/2LL);
}

```

## 外心 Circumcentre

```

#include "2Dpoint.cpp"

pdd circumcentre(pdd &p0, pdd &p1, pdd &p2){
    pdd a = p1-p0;
    pdd b = p2-p0;
    double c1 = a.abs2()*0.5;
    double c2 = b.abs2()*0.5;
    double d = a % b;
    double x = p0.x + ( c1*b.y - c2*a.y ) / d;
    double y = p0.y + ( c2*a.x - c1*b.x ) / d;
    return pdd(x,y);
}

```

## Smallest Covering Circle

```

#include "circumcentre.cpp"
pair<pdd,double> SmallestCircle(int n, pdd _p[]){
    static const int MAXN = 1000006;
    static pdd p[MAXN];
    memcpy(p,_p,sizeof(pdd)*n);
    random_shuffle(p,p+n);

    double r2=0;
    pdd cen;
    for (int i=0; i<n; i++){
        if ( (cen-p[i]).abs2() <=r2)continue;

```

```

        cen = p[i], r2=0;
        for (int j=0; j<i; j++){
            if ( (cen-p[j]).abs2()<=r2 )continue;
            cen = (p[i]+p[j])*0.5;
            r2 = (cen-p[i]).abs2();
            for (int k=0; k<j; k++){
                if ( (cen-p[k]).abs2()<=r2 )continue;
                cen = circumcentre(p[i],p[j],p[k]);
                r2 = (cen-p[k]).abs2();
            }
        }
    }
    return {cen,r2};
}
// auto res = SmallestCircle();

```

## Flow

### Dinic

```

const int INF = 1<<29;
struct Dinic{ //0(VVE)
    static const int MAXV = 5003;
    struct Edge{
        int from, to, cap, flow;
    };

    int n, m, s, t, d[MAXV], cur[MAXV];
    vector<Edge> edges;
    vector<int> G[MAXV];

    void init(int _n=MAXV){
        edges.clear();
        for (int i=0; i<_n; i++)G[i].clear();
    }

    void AddEdge(int from, int to, int cap){
        edges.push_back( {from,to,cap,0} );
        edges.push_back( {to,from,0,0} );
        m = edges.size();
        G[from].push_back(m-2);
        G[to].push_back(m-1);
    }

    bool dinicBFS(){
        memset(d,-1,sizeof(d));
        queue<int> que;
        que.push(s); d[s]=0;
        while (!que.empty()){
            int u = que.front(); que.pop();
            for (int ei:G[u]){
                Edge &e = edges[ei];
                if (d[e.to]<0 && e.cap>e.flow){
                    d[e.to]=d[u]+1;
                    que.push(e.to);
                }
            }
        }
        return d[t]>=0;
    }

    int dinicDFS(int u, int a){
        if (u==t || a==0)return a;
        int flow=0, f;
        for (int &i=cur[u]; i<(int)G[u].size(); i++){
            Edge &e = edges[ G[u][i] ];
            if (d[u]+1!=d[e.to])continue;
            f = dinicDFS(e.to, min(a, e.cap-e.flow) );
            if (f>0){
                e.flow += f;
                edges[ G[u][i]^1 ].flow -=f;
                flow += f;
                a -= f;
                if (a==0)break;
            }
        }
        return flow;
    }
}

```

```

}

int maxflow(int s, int t){
    this->s = s, this->t = t;
    int flow=0, mf;
    while ( dinicBFS() ){
        memset(cur,0,sizeof(cur));
        while ( (mf=dinicDFS(s,INF)) )flow+=mf;
    }
    return flow;
}
}dinic;

// s=0, t=1;
int fnd(int id ,int out=0){
    // out=0 入點 out=1 出點
    static int spr=1;
    //spr=2 時每個點分成入點,出點
    return id*spr+out+2;
}

```

## KM

```

struct KM{
    // Maximum Bipartite Weighted Matching (Perfect Match)
    static const int MXN = 650;
    static const int INF = 2147483647; // long long
    int n,match[MXN],vx[MXN],vy[MXN];
    int edge[MXN][MXN],lx[MXN],ly[MXN],slack[MXN];
    // ^^^^ long long
    void init(int _n){
        n = _n;
        for (int i=0; i<n; i++)
            for (int j=0; j<n; j++)
                edge[i][j] = 0;
    }
    void add_edge(int x, int y, int w){ // long long
        edge[x][y] = w;
    }
    bool DFS(int x){
        vx[x] = 1;
        for (int y=0; y<n; y++){
            if (vy[y]) continue;
            if (lx[x]+ly[y] > edge[x][y]){
                slack[y] = min(slack[y], lx[x]+ly[y]-edge[x][y]);
            }
            else {
                vy[y] = 1;
                if (match[y] == -1 || DFS(match[y])){
                    match[y] = x;
                    return true;
                }
            }
        }
        return false;
    }
    int solve(){
        fill(match,match+n,-1);
        fill(lx,lx+n,-INF);
        fill(ly,ly+n,0);
        for (int i=0; i<n; i++)
            for (int j=0; j<n; j++)
                lx[i] = max(lx[i], edge[i][j]);
        for (int i=0; i<n; i++){
            fill(slack,slack+n,INF);
            while (true){
                fill(vx,vx+n,0);
                fill(vy,vy+n,0);
                if ( DFS(i) ) break;
                int d = INF; // long long
                for (int j=0; j<n; j++)
                    if (!vy[j]) d = min(d, slack[j]);
                for (int j=0; j<n; j++){
                    if (vx[j]) lx[j] -= d;
                    if (vy[j]) ly[j] += d;
                    else slack[j] -= d;
                }
            }
        }
        int res=0;
    }
}

```

```

    for (int i=0; i<n; i++){
        res += edge[match[i]][i];
    }
    return res;
}
}graph;

```

## KM

```

const int MAX_N = 400 + 10;
const ll INF64 = 0x3f3f3f3f3f3f3fLL;
int nl , nr;
int pre[MAX_N];
ll slack[MAX_N];
ll W[MAX_N][MAX_N];
ll lx[MAX_N] , ly[MAX_N];
int mx[MAX_N] , my[MAX_N];
bool vx[MAX_N] , vy[MAX_N];
void augment(int u) {
    if(!u) return;
    augment(mx[pre[u]]);
    mx[pre[u]] = u;
    my[u] = pre[u];
}
inline void match(int x) {
    queue<int> que;
    que.push(x);
    while(1) {
        while(!que.empty()) {
            x = que.front();
            que.pop();
            vx[x] = 1;
            REP1(y , 1 , nr) {
                if(vy[y]) continue;
                ll t = lx[x] + ly[y] - W[x][y];
                if(t > 0) {
                    if(slack[y] >= t) slack[y] = t ,
                        pre[y] = x;
                    continue;
                }
                pre[y] = x;
                if(!my[y]) {
                    augment(y);
                    return;
                }
                vy[y] = 1;
                que.push(my[y]);
            }
        }
        ll t = INF64;
        REP1(y , 1 , nr) if(!vy[y]) t = min(t , slack[y]);
        REP1(x , 1 , nl) if(vx[x]) lx[x] -= t;
        REP1(y , 1 , nr) {
            if(vy[y]) ly[y] += t;
            else slack[y] -= t;
        }
        REP1(y , 1 , nr) {
            if(vy[y] || slack[y]) continue;
            if(!my[y]) {
                augment(y);
                return;
            }
            vy[y] = 1;
            que.push(my[y]);
        }
    }
}
int main() {
    int m;
    RI(nl , nr , m);
    nr = max(nl , nr);
    while(m-->0) {
        int x , y;
        ll w;
        RI(x , y , w);
        W[x][y] = w;
        lx[x] = max(lx[x] , w);
    }
    REP1(i , 1 , nl) {
        REP1(x , 1 , nl) vx[x] = 0;
    }
}

```

```

    REP1(y, 1, nr) vy[y] = 0, slack[y] = INF64;
    match(i);
}
ll ans = 0LL;
REP1(x, 1, nl) ans += W[x][mx[x]];
PL(ans);
REP1(x, 1, nl) printf("%d%c", W[x][mx[x]] ? mx[x]
: 0, "\n"[x == nl]);
return 0;
}

```

## min cost max flow

```

// from: https://github.com/bobogei81123/bcw_codebook/
// blob/master/codes/Graph/Flow/CostFlow.cpp
typedef pair<long long, long long> pll;
struct CostFlow {
    static const int MXN = 205;
    static const long long INF = 102938475610293847LL;
    struct Edge {
        int v, r;
        long long f, c;
    };
    int n, s, t, prv[MXN], prvl[MXN], inq[MXN];
    long long dis[MXN], fl, cost;
    vector<Edge> E[MXN];
    void init(int _n, int _s, int _t) {
        n = _n; s = _s; t = _t;
        for (int i=0; i<n; i++) E[i].clear();
        fl = cost = 0;
    }
    void add_edge(int u, int v, long long f, long long c) {
        E[u].PB({v, SZ(E[v]), f, c});
        E[v].PB({u, SZ(E[u])-1, 0, -c});
    }
    pll flow() {
        while (true) {
            for (int i=0; i<n; i++) {
                dis[i] = INF;
                inq[i] = 0;
            }
            dis[s] = 0;
            queue<int> que;
            que.push(s);
            while (!que.empty()) {
                int u = que.front(); que.pop();
                inq[u] = 0;
                for (int i=0; i<SZ(E[u]); i++) {
                    int v = E[u][i].v;
                    long long w = E[u][i].c;
                    if (E[u][i].f > 0 && dis[v] > dis[u] + w) {
                        prv[v] = u; prvl[v] = i;
                        dis[v] = dis[u] + w;
                        if (!inq[v]) {
                            inq[v] = 1;
                            que.push(v);
                        }
                    }
                }
            }
            if (dis[t] == INF) break;
            long long tf = INF;
            for (int v=t, u, l; v!=s; v=u) {
                u=prv[v]; l=prvl[v];
                tf = min(tf, E[u][l].f);
            }
            for (int v=t, u, l; v!=s; v=u) {
                u=prv[v]; l=prvl[v];
                E[u][l].f -= tf;
                E[v][E[u][l].r].f += tf;
            }
            cost += tf * dis[t];
            fl += tf;
        }
        return {fl, cost};
    }
} fflow;

```

## Graph

### Strongly Connected Component(SCC)

```

#define MXN 100005
#define PB push_back
#define FZ(s) memset(s,0,sizeof(s))

struct Scc{
    int n, nScc, vst[MXN], bln[MXN];
    vector<int> E[MXN], rE[MXN], vec;
    void init(int _n){
        n = _n;
        for (int i=0; i<MXN; i++){
            E[i].clear();
            rE[i].clear();
        }
    }
    void add_edge(int u, int v){
        E[u].PB(v);
        rE[v].PB(u);
    }
    void DFS(int u){
        vst[u]=1;
        for (auto v : E[u])
            if (!vst[v]) DFS(v);
        vec.PB(u);
    }
    void rDFS(int u){
        vst[u] = 1;
        bln[u] = nScc;
        for (auto v : rE[u])
            if (!vst[v]) rDFS(v);
    }
    void solve(){
        nScc = 0;
        vec.clear();
        FZ(vst);
        for (int i=0; i<n; i++)
            if (!vst[i]) DFS(i);
        reverse(vec.begin(),vec.end());
        FZ(vst);
        for (auto v : vec){
            if (!vst[v]){
                rDFS(v);
                nScc++;
            }
        }
    }
};

```

### Euler Circuit

```

//CF 723E
#include <bits/stdc++.h>
using namespace std;

const int MAXN = 300;

struct EDGE{
    int u, v;
    int type;
};

int n, m, deg[MAXN];
vector<EDGE> edges;
vector<int> G[MAXN];
bool vis[MAXN*MAXN];
bool alive[MAXN][MAXN];
bool visN[MAXN];
vector<int> ans;

void add_edge(int u, int v, int type=0){
    edges.push_back( EDGE{u,v,type} );
    edges.push_back( EDGE{v,u,type} );
    G[u].push_back( edges.size()-2 );
    G[v].push_back( edges.size()-1 );
    deg[u]++, deg[v]++;
}

```



```

    alive[u][v]=alive[v][u]!=type^1;
}

void input(){
    memset(visN,0,sizeof(visN));
    memset(vis,0,sizeof(vis));
    memset(alive,0,sizeof(alive));
    memset(deg,0,sizeof(deg));
    edges.clear();
    ans.clear();
    for (int i=0; i<MAXN; i++)G[i].clear();

    scanf("%d%d",&n,&m);
    for (int i=0; u, v; i<m; i++){
        scanf("%d%d",&u,&v);
        add_edge(u,v);
    }
}

void add_Graph(){
    vector<int> tmp;
    for (int i=1; i<=n; i++)if (deg[i]%2==1){
        tmp.push_back(i);
    }
    printf("%d\n",n-tmp.size());
    for (int i=0; i<tmp.size(); i+=2){
        add_edge(tmp[i],tmp[i+1],1);
    }
}

void dfs(int u){
    visN[u]=1;
    for (int i=0; i<G[u].size(); i++)if (!vis[ G[u][i]
        ]>>1 ]){
        EDGE &e = edges[ G[u][i] ];
        int v = e.v;
        vis[ G[u][i]>>1 ]=1;
        dfs(v);
    }
    ans.push_back(u);
}

int main(){
    int T; scanf("%d",&T);
    while (T--){
        input();
        add_Graph();
        for (int i=1; i<=n; i++)if (!visN[i]){
            dfs(i);
            for (int j=0 ;j<ans.size()-1; j++){
                int u = ans[j], v=ans[j+1];
                if (alive[u][v]){
                    alive[u][v]=alive[v][u]=0;
                    printf("%d %d\n",u ,v);
                }
            }
            ans.clear();
        }
    }
}

```

## Hungarian

```

vector<int> G[MAXN];
int n;
int match[MAXN]; // Matching Result
int visit[MAXN];

bool dfs(int u) {
    for (auto v:G[u]) {
        if (!visit[v]) {
            visit[v] = true;
            if (match[v] == -1 || dfs(match[v])) {
                match[v] = u;
                match[u] = v;
                return true;
            }
        }
    }
}

```

```

return false;
}

int hungarian() {
    int res = 0;
    memset(match, -1, sizeof(match));
    for (int i = 0; i < n; i++) {
        if (match[i] == -1) {
            memset(visit, 0, sizeof(visit));
            if (dfs(i)) res += 1;
        }
    }
    return res;
}

```

## Maximum Clique

```

const int MAXN = 105;
int best;
int m ,n;
int num[MAXN];
// int x[MAXN];
int path[MAXN];
int g[MAXN][MAXN];

bool dfs( int *adj, int total, int cnt ){
    int i, j, k;
    int t[MAXN];
    if( total == 0 ){
        if( best < cnt ){
            // for( i = 0; i < cnt; i++) path[i] = x[i];
            best = cnt; return true;
        }
        return false;
    }
    for( i = 0; i < total; i++){
        if( cnt+(total-i) <= best ) return false;
        if( cnt+num[adj[i]] <= best ) return false;
        // x[cnt] = adj[i];
        for( k = 0, j = i+1; j < total; j++ )
            if( g[ adj[i] ][ adj[j] ] )
                t[ k++ ] = adj[j];
        if( dfs( t, k, cnt+1 ) ) return true;
    }
    return false;
}

int MaximumClique(){
    int i, j, k;
    int adj[MAXN];
    if( n <= 0 ) return 0;
    best = 0;
    for( i = n-1; i >= 0; i-- ){
        // x[0] = i;
        for( k = 0, j = i+1; j < n; j++ )
            if( g[i][j] ) adj[k++] = j;
        dfs( adj, k, 1 );
        num[i] = best;
    }
    return best;
}

```

## Tarjan

```

int n;
vector<int> G[MAXN];
stack<int> stk;
int dfn[MAXN], low[MAXN];
bool ins[MAXN];
int scc[MAXN], scn, count;

void tarjan(int u){
    dfn[u] = low[u] = ++count;
    stk.push(u);
    ins[u] = true;
    for(auto v:G[u]){
        if(!dfn[v]){
            tarjan(v);

```



```

        low[u] = min(low[u], low[v]);
    }else if(ins[v]){
        low[u] = min(low[u], dfn[v]);
    }
}

if(dfn[u] == low[u]){
    int v;
    do {
        v = stk.top();
        stk.pop();
        scc[v] = scn;
        ins[v] = false;
    } while(v != u);
    scn++;
}
}

void GetSCC(){
    count = scn = 0;
    for(int i = 0 ; i < n ; i++){
        if(!dfn[i]) tarjan(i);
    }
}

```

## 一般圖匹配

```

/// {{{ general graph matching template by jacky860226
#define MAXN 505
vector<int> g[MAXN]; //用vector存圖
int pa[MAXN], match[MAXN], st[MAXN], S[MAXN], vis[
    MAXN];
//pa表示交錯樹每個節點的父節點
//match[u]=v表示u和v匹配，同時match[v]=u
//st[u]=B表示節點u屬於B這朵花
//S[u]={-1:沒走過 0:偶點 1:奇點}
//vis只用在找lca的時候檢查是不是走過了
int n; //n個點，編號為1 ~ n
inline int lca(int u, int v){
    //找花的花托，也就是交錯樹的lca
    //這種方法可以不用清空vis陣列就可以判斷有沒有經過
    static int t=0;
    for(++t; swap(u, v)){
        if(u==0) continue;
        if(vis[u]==t) return u;
        vis[u]=t;
        u=st[pa[match[u]]];
    }
}

#define qpush(u) q.push(u), S[u]=0
//因為丟進queue裡的節點必為偶點，故把兩個操作寫在一起
inline void flower(int u, int v, int l, queue<int> &q){
    //這個函數用來設定花裡面所有點的pa
    while(st[u]!=l){
        pa[u]=v; //所有未匹配邊的pa都是雙向的
        v=match[u];
        if(S[v]==1) qpush(v); //所有奇點變偶點
        st[u]=st[v]=l;
        //注意這邊以花的花托代表這個花
        //所以 st[u]=st[v]=l 就是設定 u 和 v 屬於 l 這
        //朵花
        u=pa[v];
    }
}

inline bool agument(int u, int v){
    //擴充增廣路
    for(int lst=u; v=lst, u=pa[v]){
        lst=match[u];
        match[u]=v;
        match[v]=u;
    }
}

inline bool bfs(int u){
    for(int i=1; i<=n; ++i) st[i]=i; //自己一個點也是奇環
    memset(S+1, -1, sizeof(int)*n);
    queue<int> q;
    qpush(u);
    while(q.size()){

```

```

        u=q.front(), q.pop();
        for(size_t i=0; i<q[u].size(); ++i){
            int v=g[u][i];
            if(S[v]==-1){
                pa[v]=u;
                S[v]=1;
                if(!match[v]){ //有增廣路直接擴充
                    agument(u, v);
                    return true;
                }
                qpush(match[v]);
            }else if(!S[v] && st[v]!=st[u]){
                int l=lca(v, u); //遇到花，做花的處理
                flower(v, u, l, q);
                flower(u, v, l, q);
            }
        }
    }
    return false;
}

inline int blossom(){
    //ans表示最大匹配數量
    memset(pa+1, 0, sizeof(int)*n);
    memset(match+1, 0, sizeof(int)*n);
    int ans=0;
    for(int i=1; i<=n; ++i)
        if(!match[i] && bfs(i)) ++ans;
    return ans;
}

/// }}}
int main() {
    int t;
    RI(t);
    while(t--){
        int m;
        RI(n, m);
        REP1(i, 1, n) g[i].clear();
        REP(i, m){
            int x, y;
            RI(x, y);
            x++, y++;
            g[x].PB(y);
            g[y].PB(x);
        }
        PL(blossom());
    }
    return 0;
}

```

## LCA

```

//lv紀錄深度
//father[多少層次][誰]
//已經建好每個人的父親是誰 (father[0][i]已經建好)
//已經建好深度 (lv[i]已經建好)
void makePP(){
    for(int i = 1; i < 20; i++){
        for(int j = 2; j <= n; j++){
            father[i][j]=father[i-1][ father[i-1][j] ];
        }
    }
}

int find(int a, int b){
    if(lv[a] < lv[b]) swap(a, b);
    int need = lv[a] - lv[b];
    for(int i = 0; need!=0; i++){
        if(need&1) a=father[i][a];
        need >>= 1;
    }
    for(int i = 19; i >= 0; i--){
        if(father[i][a] != father[i][b]){
            a=father[i][a];
            b=father[i][b];
        }
    }
    return a!=b?father[0][a] : a;
}

```

## Data Structure

### Disjoint Set

```
struct DisjointSet{
    int n, fa[MAXN];

    void init(int size) {
        for (int i = 0; i <= size; i++) {
            fa[i] = i;
        }
    }

    void find(int x) {
        return fa[x] == x ? x : find(fa[x]);
    }

    void unite(int x, int y) {
        p[find(x)] = find(y);
    }
} djs;
```

### Sparse Table

```
const int MAXN = 200005;
const int lgN = 20;

struct SP{ //sparse table
    int Sp[MAXN][lgN];
    function<int(int,int)> opt;
    void build(int n, int *a){ // 0 base
        for (int i=0; i<n; i++) Sp[i][0]=a[i];

        for (int h=1; h<lgN; h++){
            int len = 1<<(h-1), i=0;
            for (; i+len<n; i++){
                Sp[i][h] = opt( Sp[i][h-1] , Sp[i+len][h-1] );
                for (; i<n; i++)
                    Sp[i][h] = Sp[i][h-1];
            }
        }
        int query(int l, int r){
            int h = __lg(r-l+1);
            int len = 1<<h;
            return opt( Sp[l][h] , Sp[r-len+1][h] );
        }
    };
};
```

### Treap

```
#include<bits/stdc++.h>
using namespace std;
template<class T,unsigned seed>class treap{
public:
    struct node{
        T data;
        int size;
        node *l,*r;
        node(T d){
            size=1;
            data=d;
            l=r=NULL;
        }
        inline void up(){
            size=1;
            if(l)size+=l->size;
            if(r)size+=r->size;
        }
        inline void down(){
        }
    }*root;
    inline int size(node *p){return p?p->size:0;}
    inline bool ran(node *a,node *b){
        static unsigned x=seed;
        x=0xdefaced*x+1;
```

```
        unsigned all=size(a)+size(b);
        return (x%all+all)%all<size(a);
    }
    void clear(node *&p){
        if(p)clear(p->l),clear(p->r),delete p,p=NULL;
    }
    ~treap(){clear(root);}
    void split(node *o,node *&a,node *&b,int k){
        if(!k)a=NULL,b=o;
        else if(size(o)==k)a=o,b=NULL;
        else{
            o->down();
            if(k<=size(o->l)){
                b=o;
                split(o->l,a,b->l,k);
                b->up();
            }else{
                a=o;
                split(o->r,a->r,b,k-size(o->l)-1);
                a->up();
            }
        }
    }
    void merge(node *&o,node *a,node *b){
        if(!a||!b)o=a?a:b;
        else{
            if(ran(a,b)){
                a->down();
                o=a;
                merge(o->r,a->r,b);
            }else{
                b->down();
                o=b;
                merge(o->l,a,b->l);
            }
            o->up();
        }
    }
    void build(node *&p,int l,int r,T *s){
        if(l>r)return;
        int mid=(l+r)>>1;
        p=new node(s[mid]);
        build(p->l,l,mid-1,s);
        build(p->r,mid+1,r,s);
        p->up();
    }
    inline int rank(T data){
        node *p=root;
        int cnt=0;
        while(p){
            if(data<=p->data)p=p->l;
            else cnt+=size(p->l)+1,p=p->r;
        }
        return cnt;
    }
    inline void insert(node *&p,T data,int k){
        node *a,*b,*now;
        split(p,a,b,k);
        now=new node(data);
        merge(a,a,now);
        merge(p,a,b);
    }
};
treap<int ,20141223>bst;
int n,m,a,b;
int main(){
    //當成二分查找樹用
    while(~scanf("%d",&a))bst.insert(bst.root,a,bst.rank(a));
    while(~scanf("%d",&a))printf("%d\n",bst.rank(a));
    bst.clear(bst.root);
    return 0;
}
```

## String

### KMP

```

template<typename T>
void build_KMP(int n, T *s, int *f){ // 1 base
    f[0]=-1, f[1]=0;
    for (int i=2; i<=n; i++){
        int w = f[i-1];
        while (w>=0 && s[w+1]!=s[i])w = f[w];
        f[i]=w+1;
    }
}

template<typename T>
int KMP(int n, T *a, int m, T *b){
    build_KMP(n,b,f);
    int ans=0;

    for (int i=1, w=0; i<=n; i++){
        while ( w>=0 && b[w+1]!=a[i] )w = f[w];
        w++;
        if (w==m){
            ans++;
            w=f[w];
        }
    }
    return ans;
}

```

## AC

```

// by bcw_codebook
struct ACautomata{
    struct Node{
        int cnt,dp;
        Node *go[26], *fail;
        Node (){
            cnt = 0;
            dp = -1;
            memset(go,0,sizeof(go));
            fail = 0;
        }
    };

    Node *root, pool[1048576];
    int nMem;

    Node* new_Node(){
        pool[nMem] = Node();
        return &pool[nMem++];
    }

    void init(){
        nMem = 0;
        root = new_Node();
    }

    void add(const string &str){
        insert(root,str,0);
    }

    void insert(Node *cur, const string &str, int pos){
        if (pos >= (int)str.size()){
            cur->cnt++;
            return;
        }
        int c = str[pos]-'a';
        if (cur->go[c] == 0){
            cur->go[c] = new_Node();
        }
        insert(cur->go[c],str,pos+1);
    }

    void make_fail(){
        queue<Node*> que;
        que.push(root);
        while (!que.empty()){
            Node* fr=que.front();
            que.pop();
            for (int i=0; i<26; i++){
                if (fr->go[i]){
                    Node *ptr = fr->fail;
                    while (ptr && !ptr->go[i]) ptr = ptr->fail;
                    if (!ptr) fr->go[i]->fail = root;
                    else fr->go[i]->fail = ptr->go[i];
                    que.push(fr->go[i]);
                }
            }
        }
    }
}

```

```

    }
}
};

```

## Z-value

```

z[0] = 0;
for ( int bst = 0, i = 1; i < len ; i++ ) {
    if ( z[bst] + bst <= i ) z[i] = 0;
    else z[i] = min(z[i - bst], z[bst] + bst - i);
    while ( str[i + z[i]] == str[z[i]] ) z[i]++;
    if ( i + z[i] > bst + z[bst] ) bst = i;
}

```

## Suffix Array

```

const int MAX_N = 100000 + 10;
char str[MAX_N];
int sa[MAX_N], cnt[MAX_N], rk[2][MAX_N], ht[MAX_N];
inline void build_sa(int n, int m) {
    int *x = rk[0], *y = rk[1];
    REP(i, m) cnt[i] = 0;
    REP(i, n + 1) cnt[x[i]] = str[i]++;
    REP1(i, 1, m - 1) cnt[i] += cnt[i - 1];
    for(int i = n; i >= 0; i--) sa[--cnt[x[i]]] = i;
    for(int k = 1; k <= 1) {
        int p = 0;
        REP1(i, n - k + 1, n) y[p++] = i;
        REP(i, n + 1) if(sa[i] >= k) y[p++] = sa[i] - k;
        REP(i, m) cnt[i] = 0;
        REP(i, n + 1) cnt[x[y[i]]]++;
        REP1(i, 1, m - 1) cnt[i] += cnt[i - 1];
        for(int i = n; i >= 0; i--) sa[--cnt[x[y[i]]]]
            = y[i];
        swap(x, y);
        p = 1;
        x[sa[0]] = 0;
        REP1(i, 1, n) x[sa[i]] = (y[sa[i]] == y[sa[i]
            - 1]) && y[sa[i] + k] == y[sa[i - 1] + k] ?
            p - 1 : p++;
        if(p >= n + 1) break;
        else m = p;
    }
}

void build_ht(int n) {
    int *x = rk[0];
    REP1(i, 1, n) x[sa[i]] = i;
    for(int i = 0, h = 0; i < n; i++) {
        int j = sa[x[i]-1];
        if(h) h--;
        while(str[i + h] == str[j + h]) h++;
        ht[x[i]] = h;
    }
}

int main() {
    scanf("%s",str);
    int n = strlen(str);
    build_sa(n, 128);
    build_ht(n);
    REP1(i, 1, n) printf("%d%c",sa[i] + 1," \n"[i == n
        ]);
    REP1(i, 2, n) printf("%d%c",ht[i]," \n"[i == n]);
    return 0;
}

```

## Dark Code

## 輸入優化

```

#include <stdio.h>

char getc(){

```

```

static const int bufsize = 1<<16;
static char B[bufsize], *S=B, *T=B;
return (S==T&&(T=(S=B)+fread(B,1,bufsize,stdin),S==T)
?0:*S++);
}

template <class T>
bool input(T& a){
    a=(T)0;
    register char p;
    while ((p = getc()) < '-')
        if (p==0 || p==EOF) return false;
    if (p == '-')
        while ((p = getc()) >= '0') a = a*10 - (p^'0');
    else {
        a = p ^ '0';
        while ((p = getc()) >= '0') a = a*10 + (p^'0');
    }
    return true;
}

template <class T, class... U>
bool input(T& a, U&... b){
    if (!input(a)) return false;
    return input(b...);
}

```

Search

Others

Persistence