

Contents

1	Basic	1
1.1	makefile	1
1.2	/.vimrc	1
1.3	default code	1
1.4	debug list	1
2	Flow	1
2.1	Dinic	1
2.2	KM	2
2.3	KM	2
2.4	min cost max flow	3
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	4
3.5	Smallest Covering Circle	4
4	Mathmatics	4
4.1	LinearPrime	4
4.2	BigInt	5
4.3	Random	6
4.4	Theorem	6
4.5	Miller Rabin	6
4.6	ax+by=gcd(a,b)	6
4.7	FFT	6
4.8	FWHT	6
4.9	Hash	7
4.10	GaussElimination	7
4.11	Inverse	7
4.12	IterSet	7
4.13	SG	7
5	Graph	7
5.1	Dijkstra	7
5.2	Dijkstra - python2	8
5.3	Euler Circuit	8
5.4	一般圖匹配	8
5.5	Hungarian	9
5.6	Strongly Connected Component(SCC)	9
5.7	LCA	10
5.8	Maximum Clique	10
5.9	Tarjan	10
5.10	2-SAT	10
6	Data Structure	11
6.1	Disjoint Set	11
6.2	Sparse Table	11
6.3	Treap	11
7	String	12
7.1	KMP	12
7.2	AC 自動機	12
7.3	Z-value	12
7.4	Suffix Array	12
7.5	Suffix Automaton	13
7.6	迴文字動機	13
7.7	smallest rotation	13
8	Dark Code	14
8.1	輸入優化	14
9	Search	14
10	Others	14
10.1	數位統計	14
10.2	Stable Marriage	14
10.3	STL	15
11	Persistence	15

Basic

makefile

```

1  CPP = g++ -std=c++11 -O2
2
3  name = $(basename $(file))
4  type = $(suffix $(file))
5  exe = $(name).out
6
7  $(exe): $(file)
8  ifeq ($(type),.cpp)
9      $(CPP) -D AC -o $(exe) $(name).cpp
10 else ifeq ($(type),.py)
11     cp $(file) $(exe)
12     chmod +x $(exe)
13 endif
14
15 clean:
16     rm *.out

```

/.vimrc

```

1  set nu ai si cin ts=4 sw=4 sts=4
2
3  nmap <F2> :! gedit %<.in %<*.in &<CR>
4  nmap <F4> :! date > %<.pt; cat -n % > %<.pt; lpr %<.pt
5      <CR>
6  nmap <F9> :! clear ; g++ -std=c++11 -O2 -D AC -o %<.
7      out % ; for i in %<*.in; do echo $i; ./%<.out < $i;
8      echo "" ; done <CR>
9  nmap <F10> :! clear ; g++ -std=c++11 -O2 -D AC -o %<.
10     out % ; ./%<.out <CR>
11 nmap <C-I> :! read -p "CASE:" CASE; gedit %<_CASE.in <
12     CR>

```

default code

```

1  #include <bits/stdc++.h>
2  using namespace std;
3
4  int main(){
5  #ifndef AC
6      freopen("", "r", stdin);
7  #endif
8      ios_base::sync_with_stdio(0);
9      cin.tie(0);
10 }

```

debug list

模板要記得 init
 把邊界條件都加入測資
 邊界條件 (過程溢位, 題目數據範圍), 會不會爆 long long
 是否讀錯題目
 環狀 or 凸包問題一定要每種都算 n 次
 比較容易有問題的地方換人寫
 注意公式有沒有推錯或抄錯
 精度誤差 sqrt(大大的東西) + EPS
 測試 %lld or %I64d
 喇分 random_shuffle 隨機演算法

Flow

Dinic

```
// 只能拿一次的點都要在點上加上 cap 1
const int INF = 1<<29;
struct Dinic{ //O(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 = 0x3f3f3f3f3f3f3f3fLL;
int n1, nr;
int pre[MAX_N];
```

```

11 slack[MAX_N];
11 w[MAX_N][MAX_N];
11 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;
                11 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]);
            }
        }
        11 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(n1 , nr , m);
    nr = max(n1 , nr);
    while(m--) {
        int x , y;
        11 w;
        RI(x , y , w);
        W[x][y] = w;
        lx[x] = max(lx[x] , w);
    }
    REP1(i , 1 , n1) {
        REP1(x , 1 , nl) vx[x] = 0;
        REP1(y , 1 , nr) vy[y] = 0 , slack[y] = INF64;
        match(i);
    }
    11 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};
    }
} flow;

```

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 "2Dpoint.cpp"

// return H, 第一個點會在 H 出現兩次
void ConvexHull(vector<Point> &P, vector<Point> &H){
    int n = P.size(), m=0;
    sort(P.begin(),P.end());
    H.clear();

    for (int i=0; i<n; i++){
        while (m>=2 && (P[i]-H[m-2]) % (H[m-1]-H[m-2])
            <0)H.pop_back(), m--;
        H.push_back(P[i]), m++;
    }
}
```

```
for (int i=n-2; i>=0; i--){
    while (m>=2 && (P[i]-H[m-2]) % (H[m-1]-H[m-2])
        <0)H.pop_back(), m--;
    H.push_back(P[i]), m++;
}
}
```

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(,);
```

Mathematics

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;
        }
    }
}
```

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;
}

```

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;

inline LL bin_mul(LL a, LL n,const LL& MOD){
    LL re=0;
    while (n>0){
        if (n&1) re += a;
        a += a; if (a>MOD) a-=MOD;
        n>>=1;
    }
    return re%MOD;
}

inline LL bin_pow(LL a, LL n,const LL& MOD){
    LL re=1;
    while (n>0){
        if (n&1) re = bin_mul(re,a,MOD);
        a = bin_mul(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<3; 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;
    }
}

```

FWHT

```

// FWHT template

const int MAXN = 1<<20;

void FWHT(int a[], int l=0, int r=MAXN-1){
    if (l==r)return;

    int mid = (l+r)>>1+1, n = r-l+1;

```

```
FWHT(a,l,mid-1);
FWHT(a,mid,r);

for (int i=0; i<(n>>1); i++){
    int a1=a[l+i], a2=a[mid+i];
    a[l+i] = a1+a2;
    a[mid+i] = a1-a2;
}
}
```

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 ) {
}
}
```

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]
}
```

Graph

Dijkstra

```
typedef struct Edge{
    int v; long long len;
    bool operator > (const Edge &b)const { return len>b
        .len; }
} State;

const long long INF = 1LL<<60;

void Dijkstra(int n, vector<Edge> G[], long long d[],
    int s, int t=-1){
```

```

static priority_queue<State, vector<State>, greater
<State> > pq;
while ( pq.size() )pq.pop();
for (int i=1; i<=n; i++)d[i]=INF;
d[s]=0; pq.push( (State){s,d[s]} );
while ( pq.size() ){
    auto x = pq.top(); pq.pop();
    int u = x.v;
    if (d[u]<x.len)continue;
    if (u==t)return;
    for (auto &e:G[u]){
        if (d[e.v] > d[u]+e.len){
            d[e.v] = d[u]+e.len;
            pq.push( (State) {e.v,d[e.v]} );
        }
    }
}
}
}

```

Dijkstra - python2

```

from heapq import *
INF = 2*10**10000
t = input()
for pp in range(t):
    n, m = map(int, raw_input().split())
    g, d, q = [[] for _ in range(n+1)], [0] + [INF] * n,
    [(0, 0)]
    #for i in range(1, m):
    # a[i], b[i], c[i], l[i], o[i] = map(int, input().
    split())
    for _ in range(m):
        u, v, c, l, o = map(int, raw_input().split())
        g[u] += [(o, v, c, l)]
    while q:
        u = heappop(q)[1]
        for e in g[u]:
            k = d[u] / e[2]
            if k < 0:
                k = 0
            else:
                k = k * e[3]
            t, v = d[u] + e[0] + k, e[1]
            if t < d[v]:
                d[v] = t
                heappush(q, (d[v], v))
    print(d[n])

```

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",&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();
        }
    }
}

```

一般圖匹配

```

#define MAXN 505
vector<int>g[MAXN]; //用vector存圖
int pa[MAXN],match[MAXN],st[MAXN],S[MAXN],vis[MAXN];
int t,n;
inline int lca(int u,int v){ //找花的花托
    for(++t;swap(u,v)){
        if(u==0)continue;
        if(vis[u]==t)return u;
        vis[u]=t; //這種方法可以不用清空vis陣列
        u=st[pa[match[u]]];
    }
}

```



```

    }
}
#define qpush(u) q.push(u), S[u]=0
inline void flower(int u, int v, int l, queue<int> &q){
    while(st[u]!=1){
        pa[u]=v; //所有未匹配邊的pa都是雙向的
        if(S[v]==match[u]==1) qpush(v); //所有奇點變偶點
        st[u]=st[v]=1, u=pa[v];
    }
}
inline bool bfs(int u){
    for(int i=1; i<=n; ++i) st[i]=i; //st[i]表示第i個點的集合
    memset(S+1, -1, sizeof(int)*n); // -1: 沒走過 0: 偶點 1: 奇點
    queue<int> q; qpush(u);
    while(q.size()){
        u=q.front(), q.pop();
        for(size_t i=0; i<g[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
                    for(int l=1; l==1; u=pa[v])
                        l=match[u], match[u]=v, match[v]=u;
                    return 1;
                }
                qpush(match[v]);
            } else if(!S[v] && st[v]!=st[u]){
                int l=lca(st[v], st[u]); //遇到花，做花的處理
                flower(v, u, l, q), flower(u, v, l, q);
            }
        }
    }
    return 0;
}
inline int blossom(){
    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, m; cin >> T;

    while (cin >> n >> m){
        for(int i=1; i<=n; i++) g[i].clear();
        for(int i=1, u, v; i<=m; i++){
            cin >> u >> v;
            g[u].push_back(v);
            g[v].push_back(u);
        }
        cout << blossom() << endl;
    }
}

```

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;
}

```

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++;
            }
        }
    }
};

```

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;
}

```

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

```

ccd ..

// 0 base
struct TarjanSCC{
    static const int MAXN = 1000006;

```

```

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

    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(){
        memset(dfn,0,sizeof(dfn));
        memset(low,0,sizeof(low));
        memset(ins,0,sizeof(ins));
        memset(scc,0,sizeof(scc));
        count = scn = 0;
        for(int i = 0 ; i < n ; i++ ){
            if(!dfn[i]) tarjan(i);
        }
    }
}SCC;

```

2-SAT

```

const int MAXN = 2020;

struct TwoSAT{
    static const int MAXv = 2*MAXN;
    vector<int> GO[MAXv],BK[MAXv],stk;
    bool vis[MAXv];
    int SC[MAXv];

    void imply(int u,int v){ // u imply v
        GO[u].push_back(v);
        BK[v].push_back(u);
    }

    int dfs(int u,vector<int>*G,int sc){
        vis[u]=1, SC[u]=sc;
        for (int v:G[u])if (!vis[v])
            dfs(v,G,sc);
        if (G==GO)stk.push_back(u);
    }

    int scc(int n=MAXv){
        memset(vis,0,sizeof(vis));
        for (int i=0; i<n; i++)if (!vis[i])
            dfs(i,GO,-1);
        memset(vis,0,sizeof(vis));
        int sc=0;
        while (!stk.empty()){
            if (!vis[stk.back()])
                dfs(stk.back(),BK,sc++);
            stk.pop_back();
        }
    }
}

```

```

}SAT;

int main(){
    SAT.scc(2*n);
    bool ok=1;
    for (int i=0; i<n; i++){
        if (SAT.SC[2*i]==SAT.SC[2*i+1])ok=0;
    }
    if (ok){
        for (int i=0; i<n; i++){
            if (SAT.SC[2*i]>SAT.SC[2*i+1]){
                cout << i << endl;
            }
        }
    }
    else puts("NO");
}

```

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(m,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 自動機

```

// remember make_fail() !!!
// notice MLE

const int sigma = 62;
const int MAXC = 200005;

inline int idx(char c){
    if ('A'<= c && c <= 'Z')return c-'A';
    if ('a'<= c && c <= 'z')return c-'a' + 26;
    if ('0'<= c && c <= '9')return c-'0' + 52;
}

struct ACautomaton{
    struct Node{
        Node *next[sigma], *fail;
        int cnt; // dp
        Node(){
            memset(next,0,sizeof(next));
        }
    };
};

```

```

fail=0;
cnt=0;
}
} buf[MAXC], *bufp, *ori, *root;

void init(){
    bufp = buf;
    ori = new (bufp++) Node();
    root = new (bufp++) Node();
}

void insert(int n, char *s){
    Node *ptr = root;
    for (int i=0; s[i]; i++){
        int c = idx(s[i]);
        if (ptr->next[c]==NULL)
            ptr->next[c] = new (bufp++) Node();
        ptr = ptr->next[c];
    }
    ptr->cnt=1;
}

Node* trans(Node *o, int c){
    while (o->next[c]==NULL) o = o->fail;
    return o->next[c];
}

void make_fail(){
    static queue<Node*> que;

    for (int i=0; i<sigma; i++){
        ori->next[i] = root;
        root->fail = ori;

        que.push(root);
        while ( que.size() ){
            Node *u = que.front(); que.pop();
            for (int i=0; i<sigma; i++){
                if (u->next[i]==NULL)continue;
                u->next[i]->fail = trans(u->fail,i);
                que.push(u->next[i]);
            }
            u->cnt += u->fail->cnt;
        }
    }
}
} ac;

```

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 = 1020304;
int ct[MAX], he[MAX], rk[MAX];
int sa[MAX], tsa[MAX], tp[MAX][2];
void suffix_array(char *ip){
    int len = strlen(ip);
    int alp = 256;
    memset(ct, 0, sizeof(ct));
    for(int i=0;i<len;i++) ct[ip[i]+1]++;
    for(int i=1;i<alp;i++) ct[i]+=ct[i-1];
    for(int i=0;i<len;i++) rk[i]=ct[ip[i]];
    for(int i=1;i<len;i*=2){
        for(int j=0;j<len;j++){
            if(j+i>len) tp[j][1]=0;
            else tp[j][1]=rk[j+i]+1;
        }
    }
}

```

```

    tp[j][0]=rk[j];
}
memset(ct, 0, sizeof(ct));
for(int j=0;j<len;j++) ct[tp[j][1]+1]++;
for(int j=1;j<len+2;j++) ct[j]+=ct[j-1];
for(int j=0;j<len;j++) tsa[ct[tp[j][1]]++]=j;
memset(ct, 0, sizeof(ct));
for(int j=0;j<len;j++) ct[tp[j][0]+1]++;
for(int j=1;j<len+1;j++) ct[j]+=ct[j-1];
for(int j=0;j<len;j++)
    sa[ct[tp[tsa[j]][0]]++]=tsa[j];
rk[sa[0]]=0;
for(int j=1;j<len;j++){
    if( tp[sa[j]][0] == tp[sa[j-1]][0] &&
        tp[sa[j]][1] == tp[sa[j-1]][1] )
        rk[sa[j]] = rk[sa[j-1]];
    else
        rk[sa[j]] = j;
}
}
for(int i=0,h=0;i<len;i++){
    if(rk[i]==0) h=0;
    else{
        int j=sa[rk[i]-1];
        h=max(0,h-1);
        for(;ip[i+h]==ip[j+h];h++);
    }
    he[rk[i]]=h;
}
}
}

```

Suffix Automaton

```

// par : fail link
// val : a topological order ( useful for DP )
// go[x] : automata edge ( x is integer in [0,26) )

struct SAM{
    struct State{
        int par, go[26], val;
        State () : par(0), val(0){ FZ(go); }
        State (int _val) : par(0), val(_val){ FZ(go); }
    };
    vector<State> vec;
    int root, tail;

    void init(int arr[], int len){
        vec.resize(2);
        vec[0] = vec[1] = State(0);
        root = tail = 1;
        for (int i=0; i<len; i++)
            extend(arr[i]);
    }
    void extend(int w){
        int p = tail, np = vec.size();
        vec.PB(State(vec[p].val+1));
        for ( ; p && vec[p].go[w]==0; p=vec[p].par)
            vec[p].go[w] = np;
        if (p == 0){
            vec[np].par = root;
        } else {
            if (vec[vec[p].go[w]].val == vec[p].val+1){
                vec[np].par = vec[p].go[w];
            } else {
                int q = vec[p].go[w], r = vec.size();
                vec.PB(vec[q]);
                vec[r].val = vec[p].val+1;
                vec[q].par = vec[np].par = r;
                for ( ; p && vec[p].go[w] == q; p=vec[p].par)
                    vec[p].go[w] = r;
            }
        }
        tail = np;
    }
};

```

迴文字動機

```

// remember init()      !!!
// remember make_fail() !!!
// insert s need 1 base !!!
// notice MLE
const int sigma = 62;
const int MAXC = 1000006;
inline int idx(char c){
    if ('a'<= c && c <= 'z')return c-'a';
    if ('A'<= c && c <= 'Z')return c-'A'+26;
    if ('0'<= c && c <= '9')return c-'0'+52;
}
struct PalindromicTree{
    struct Node{
        Node *next[sigma], *fail;
        int len, cnt; // for dp
        Node(){
            memset(next,0,sizeof(next));
            fail=0;
            len = cnt = 0;
        }
    } buf[MAXC], *bufp, *even, *odd;

    void init(){
        bufp = buf;
        even = new (bufp++) Node();
        odd = new (bufp++) Node();
        even->fail = odd;
        odd->len = -1;
    }

    void insert(char *s){
        Node* ptr = even;
        for (int i=1; s[i]; i++){
            ptr = extend(ptr,s+i);
        }
    }

    Node* extend(Node *o, char *ptr){
        int c = idx(*ptr);
        while ( *ptr != *(ptr-1-o->len) )o=o->fail;
        Node *&np = o->next[c];
        if (!np){
            np = new (bufp++) Node();
            np->len = o->len+2;
            Node *f = o->fail;
            if (f){
                while ( *ptr != *(ptr-1-f->len) )f=f->fail;
                np->fail = f->next[c];
            }
            else {
                np->fail = even;
            }
            np->cnt = np->fail->cnt;
        }
        np->cnt++;
        return np;
    }
} PAM;

```

smallest rotation

```

string mcp(string s){
    int n = s.length();
    s += s;
    int i=0, j=1;
    while (i<n && j<n){
        int k = 0;
        while (k < n && s[i+k] == s[j+k]) k++;
        if (s[i+k] <= s[j+k]) j += k+1;
        else i += k+1;
        if (i == j) j++;
    }
}

```

```
int ans = i < n ? i : j;
return s.substr(ans, n);
}
```

Contact GitHub API Training Shop Blog About

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

數位統計

```
int dfs(int pos, int state1, int state2 ....., bool
    limit, bool zero) {
    if ( pos == -1 ) return 是否符合條件;
    int &ret = dp[pos][state1][state2][...];
    if ( ret != -1 && !limit ) return ret;
    int ans = 0;
    int upper = limit ? digit[pos] : 9;
    for ( int i = 0 ; i <= upper ; i++ ) {
        ans += dfs(pos - 1, new_state1, new_state2,
            limit & ( i == upper ), ( i == 0 ) && zero);
    }
    if ( !limit ) ret = ans;
    return ans;
}

int solve(int n) {
    int it = 0;
    for ( ; n ; n /= 10 ) digit[it++] = n % 10;
    return dfs(it - 1, 0, 0, 1, 1);
}
```

Stable Marriage

```
// normal stable marriage problem
// input:
//3
//Albert Laura Nancy Marcy
//Brad Marcy Nancy Laura
//Chuck Laura Marcy Nancy
//Laura Chuck Albert Brad
//Marcy Albert Chuck Brad
//Nancy Brad Albert Chuck

#include <bits/stdc++.h>
using namespace std;
const int MAXN = 505;

int n;
int favor[MAXN][MAXN]; // favor[boy_id][rank] = girl_id
;
int order[MAXN][MAXN]; // order[girl_id][boy_id] = rank
;
int current[MAXN]; // current[boy_id] = rank; boy_id
will pursue current[boy_id] girl.
int girl_current[MAXN]; // girl[girl_id] = boy_id;

void initialize() {
    for ( int i = 0 ; i < n ; i++ ) {
        current[i] = 0;
        girl_current[i] = n;
        order[i][n] = n;
    }
}

map<string, int> male, female;
string bname[MAXN], gname[MAXN];
int fit = 0;

void stable_marriage() {
    queue<int> que;
    for ( int i = 0 ; i < n ; i++ ) que.push(i);
    while ( !que.empty() ) {
        int boy_id = que.front();
        que.pop();

        int girl_id = favor[boy_id][current[boy_id]];
        current[boy_id] ++;

        if ( order[girl_id][boy_id] < order[girl_id][
            girl_current[girl_id]] ) {
            if ( girl_current[girl_id] < n ) que.push(
                girl_current[girl_id]); // if not the first
                time
            girl_current[girl_id] = boy_id;
        } else {
            que.push(boy_id);
        }
    }
}

int main() {
    cin >> n;

    for ( int i = 0 ; i < n ; i++ ) {
        string p, t;
        cin >> p;
        male[p] = i;
        bname[i] = p;
        for ( int j = 0 ; j < n ; j++ ) {
            cin >> t;
            if ( !female.count(t) ) {
                gname[fit] = t;
                female[t] = fit++;
            }
            favor[i][j] = female[t];
        }
    }
}
```

```
}

for ( int i = 0 ; i < n ; i++ ) {
    string p, t;
    cin >> p;
    for ( int j = 0 ; j < n ; j++ ) {
        cin >> t;
        order[female[p]][male[t]] = j;
    }
}

initialize();
stable_marriage();

for ( int i = 0 ; i < n ; i++ ) {
    cout << bname[i] << " " << gname[favor[i][current[i]
        ] - 1]] << endl;
}
}
```

STL

```
// algorithm
random_shuffle(a,a+n);
next_permutation(a,a+n); // need sort
nth_element (a, a+k, a+n); // kth
*min_element(a,a+n);
*unique(a,a+n); // need sort
stable_sort(a,a+n); // merge sort

// bitset (s[0] is right most)
operator[] //
count() // count number of 1
set() // all to 1
set(k) // s[k] to 1
set(k,0) // s[k] to 0
flip() // all flip
flip(k) // s[k] flip
to_ulong()
to_string()
```

Persistence