```
#pragma comment(linker, “/STACK:102400000,102400000")

// IO 外挂
#define BUFSIZE 20000000
char buf[BUFSIZE], *pt = buf;
#define scan(t) do { \
  t = 0; \
  while (!((*pt) >= '0' && (*pt) <= '9')) pt ++; \
  while (((*pt) >= '0' && (*pt) <= '9')) t = t * 10 + (*(pt ++)) - '0'; \
} while (0)

int main() {
        fread(buf, 1, BUFSIZE, stdin);
        scan(N); scan(M); // ...
}

// 网络流、费用流
class NetFlow {
public:
    struct edge {
        int value, rev, y;
    };
    vector<edge> flow[maxn];
    bool flag[maxn];
    int dis[maxn];
    int n;
    queue<int> q;
    void init(int m = maxn - 1) {
        n = m;
        while(!q.empty())q.pop();
        for(int i = 1;i <= n; i ++) {
            flow[i].clear();
        }
    }
    void addedge(int x, int y, int v = 1) {
        edge t;
        t.value = v;
        t.rev = flow[y].size();
        t.y = y;
        flow[x].push_back(t);
        t.value = 0;
        t.rev = flow[x].size() - 1;
        t.y = x;
        flow[y].push_back(t);
    }
    void addedge_u(int x, int y, int v = 1) {
        edge t;
        t.value = v;
        t.rev = flow[y].size();
        t.y = y;
        flow[x].push_back(t);
        t.rev = flow[x].size() - 1;
        t.y = x;
        flow[y].push_back(t);
```

```
    }
    DATA_TYPE bfs(int s, int t) {
        for(int i = 1; i <= n; i ++) {
            flag[i] = 0;
        }
        dis[s] = 0;
        flag[s] = 1;
        q.push(s);
        while(!q.empty()) {
            int x = q.front();
            q.pop();
            for(int i = 0; i < flow[x].size(); i ++) if(!flag[flow[x][i].y] && flow[x][i].value) {
                int y = flow[x][i].y;
                flag[y] = 1;
                q.push(y);
                dis[y] = dis[x] + 1;
            }
        }
        return flag[t];
    }
    DATA_TYPE dfs(int x, int v, int t) {
        DATA_TYPE val = 0;
        if(x == t) {
            return v;
        }
        DATA_TYPE used = 0;
        for(int i = 0; i < flow[x].size(); i ++) if(!flag[flow[x][i].y] && flow[x][i].value && dis[flow[x][i].y] >
dis[x] && (val = dfs(flow[x][i].y, min(v - used, flow[x][i].value), t))) {
            flow[x][i].value -= val;
            flow[flow[x][i].y][flow[x][i].rev].value += val;
            used += val;
            if(used == v)return v;
        }
        if(used == 0)flag[x] = true;
        return used;
    }
    DATA_TYPE dinic(int s, int t) {
        DATA_TYPE ans = 0;
        while(bfs(s, t)) {
            for(int i = 1; i <= n; i ++) {
                flag[i] = 0;
            }
            while(int v = dfs(s, INF, t)) {
                ans += v;
                for(int i = 1; i <= n; i ++) {
                    flag[i] = 0;
                }
            }
        }
        return ans;
    }
    DATA_TYPE spfa(int s, int t) {
        for(int i = 1; i <= n; i ++) {
            flag[i] = 0;
            pi[i].x = pi[i].i = pi[i].minflow = 0;
            dis[i] = INF;
```

```
        }
        dis[s] = 0;
        flag[s] = 1;
        pi[s].minflow = INF;
        q.push(s);
        while(!q.empty()) {
            int x = q.front();
            q.pop();
            for(int i = 0; i < flow[x].size(); i ++) if(dis[flow[x][i].y] > dis[x] + flow[x][i].cost && flow[x]
[i].value) {
                int y = flow[x][i].y;
                dis[y] = dis[x] + flow[x][i].cost;
                pi[y].x = x;
                pi[y].i = i;
                pi[y].minflow = min(pi[x].minflow, flow[x][i].value);
                if(!flag[y]) {
                    flag[y] = 1;
                    q.push(y);
                }
            }
            flag[x] = 0;
        }
        return dis[t];
    }
    DATA_TYPE mincostflow(int s, int t) {
        DATA_TYPE ans = 0;
        while(spfa(s, t) != INF) {
            int p = t;
            DATA_TYPE v = pi[p].minflow;
            while(pi[p].x) {
                flow[pi[p].x][pi[p].i].value -= v;
                flow[flow[pi[p].x][pi[p].i].y][flow[pi[p].x][pi[p].i].rev].value += v;
                ans += flow[pi[p].x][pi[p].i].cost * v;
                p = pi[p].x;
            }
        }
        return ans;
    }
};
NetFlow net;
int main(int argc, const char * argv[]) {
    int i,j,m,n,N;
    while (scanf("%d%d",&m,&n)!=EOF) {
        net.init(n);
        for(i=1;i<=m;i++){
            int x,y,z;
            scanf("%d%d%d",&x,&y,&z);
            net.addedge(x,y,z);
        }
        printf("%d\n",net.dinic(1, n));
    }
    return 0;
}
```

```
//LUP分解、求逆
#define DATA_TYPE double
class Matrix{
public:
    DATA_TYPE L[maxn][maxn], U[maxn][maxn], PA[maxn][maxn], A[maxn][maxn], temp[maxn]
[maxn], INV[maxn][maxn], Y[maxn], B[maxn], X[maxn];
    int size, Pi[maxn], sign;

    void scan(int size){
        sign = 0;
        this->size = size;
        for(int i = 1; i <= size; i ++){
            for(int j = 1; j <= size; j ++){
                cin >> A[i][j];
            }
        }
    }

    void LUP(){
        for(int i = 1; i <= size; i ++){
            for(int j = 1; j <= size; j ++){
                PA[i][j] = A[i][j];
            }
        }
        for(int i = 1; i <= size; i ++)Pi[i] = i;
        for(int i = 1; i <= size; i ++){
            DATA_TYPE tem = 0;
            int j1 = -1;
            for(int j = i; j <= size; j ++){
                //if(PA[j][i].abs() > tem){
                  //  tem = PA[j][i].abs();
                if(fabs(PA[j][i]) > tem){
                    tem = fabs(PA[j][i]);
                    j1 = j;
                }
            }
            swap1(i, j1);
            L[i][i] = 1;
            U[i][i] = PA[i][i];
            for(int j = i + 1; j <= size; j ++){
                PA[j][i] = PA[j][i] / PA[i][i];
                for(int k = i + 1; k <= size; k ++){
                    PA[j][k] = PA[j][k] - PA[j][i] * PA[i][k];
                }
            }
            for(int j = 1; j <= size; j ++){
                for(int k = 1; k <= size; k ++){
                    if(k >= j){
                        U[j][k] = PA[j][k];
                    }
                    else {
                        L[j][k] = PA[j][k];
                    }
                }
            }
        }
```

```
    }

    void swap1(int i,int j){
        if(i == j)return;
        for(int k = 1; k <= size; k ++){
            DATA_TYPE t = PA[i][k];
            PA[i][k] = PA[j][k];
            PA[j][k] = t;
        }
        int t = Pi[i];
        Pi[i] = Pi[j];
        Pi[j] = t;
        sign = !sign;
    }

    void solve(){
        for(int i = 1; i <= size; i ++){
            Y[i] = B[Pi[i]];
            for(int j = 1; j < i; j ++){
                Y[i] = Y[i] - (L[i][j] * Y[j]);
            }
        }
        for(int i = size; i >= 1; i --){
            X[i] = Y[i];
            for(int j = i + 1; j <= size; j ++){
                X[i] = X[i] - (U[i][j] * X[j]);
            }
            X[i] = X[i] / U[i][i];
        }
    }

    void inverse(){
        LUP();
        for(int i = 1; i <= size; i ++){
            B[i] = 0;
        }
        for(int i = 1; i <= size; i ++){
            B[i] = 1;
            B[i - 1] = 0;
            solve();
            for(int j = 1; j <= size; j ++){
                INV[j][i] = X[j];
            }
        }
    }
};
Matrix a;
```