匹配与流
- 增广路
- Hungrian
- Hopcroft-Karp
- Kuhn-Munkres
- Dinic
- ISAP
- MCMF-SPFA
- MCMF-Dijkstra

带小数的费用流
- Edmonds(Blossom)

其他
- HLPP
- 带上下界的网络流
  - 带上下界的最大流
- km(N^3)
- SW算法求全局最小割
- LGV引理

匹配与流

增广路

注：在每个testcase前需将nc置零，用到前几个点就`addv`几个点。

注2：费用流均为多路增广，返回的`pair<ll,ll>`中`first`为最大流，`second`为最小费用。

Hungrian

二分图最大匹配

时间复杂度：$O(nm)$

```
typedef pair<int, int> pii;

namespace matching {

vector<int> g[N];
int lnk[N]; bool vis[N];

bool dfs(int u) {
    vis[u] = 1;
    for (int v : g[u]) {
        if (lnk[v] == u || vis[lnk[v]]) continue;
        if (lnk[v] && !dfs(lnk[v])) continue;
        lnk[v] = u;
        return true;
    }
    return false;
}

vector<pii> match(int nl, int nr, const vector<pii>& es) {
    //  initialize
    for (int i = 1; i <= nl; ++i)
        g[i].clear();
    for (pii p : es)
        g[p.first].push_back(p.second);
    fill_n(lnk + 1, nr, 0);
    //  do matching
    for (int i = 1; i <= nl; ++i) {
        fill_n(vis + 1, nl, false);
        dfs(i);
    }
    //  get result
    vector<pii> res;
    for (int i = 1; i <= nr; ++i)
        if (lnk[i]) res.emplace_back(lnk[i], i);
    return res;
}

}
```

Hopcroft-Karp

二分图最大匹配

```cpp
vector<int> g[N]; int n1, n2;
int lnk[N], dis[N], dm; bool vis[N];
void clr_hk(int n, int m) {
    n1 = n; n2 = m;
    for (int i = 1; i <= n1 + n2; ++i)
        g[i].clear();
}

bool bfs_hk() {
    queue<int> q; dm = INT_MAX;
    fill(dis + 1, dis + n1 + n2 + 1, -1);
    for (int i = 1; i <= n1; ++i)
        if (!lnk[i]) q.push(i), dis[i] = 0;
    while (!q.empty()) {
        int u = q.front(); q.pop();
        if (dis[u] > dm) break;
        for (int v : g[u]) {
            if (dis[v] != -1) continue;
            dis[v] = dis[u] + 1;
            if (!lnk[v]) dm = dis[v];
            else dis[lnk[v]] = dis[v] + 1, q.push(lnk[v]);
        }
    }
    return dm != INT_MAX;
}

int dfs_hk(int u) {
    for (int v : g[u]) {
        if (vis[v] || dis[v] != dis[u] + 1) continue;
        vis[v] = 1;
        if (lnk[v] && dis[v] == dm) continue;
        if (lnk[v] && !dfs_hk(lnk[v])) continue;
        lnk[v] = u; lnk[u] = v; return 1;
    }
    return 0;
}

int hk() {
    fill (lnk + 1, lnk + n1 + n2 + 1, 0);
    int res = 0;
    while (bfs_hk()) {
        fill (vis + 1, vis + n1 + n2 + 1, 0);
        for (int i = 1; i <= n1; ++i)
            if (!lnk[i] && dfs_hk(i)) res++;
    }
    return res;
}
```

Kuhn-Munkres

二分图最大权匹配

```
namespace matching {

vector<int> g[N];
int lnk[N], pre[N];
bool vis[N];

ll a[N], b[N], w[N][N], sl[N];

vector<pair<pii, ll>> match(int nl, int nr, const vector<pair<pii, ll>>& es) {
    nr = max(nl, nr);
    fill_n(lnk + 1, nr, 0);
    for (int i = 1; i <= nl; ++i)
        fill_n(w[i] + 1, nr, 0);
    for (pair<pii, ll> e : es)
        w[e.first.first][e.first.second] = max(w[e.first.first][e.first.second],
e.second);
    for (int i = 1; i <= nl; ++i)
        a[i] = *max_element(w[i] + 1, w[i] + nr + 1);
    fill_n(b + 1, nr, 0);
    for (int i = 1, j, u, vt = 0; i <= nl; ++i) {
        fill_n(vis + 1, nr, 0);
        fill_n(sl + 1, nr, inf);
        fill_n(pre + 1, nr, 0);
        lnk[0] = i;
        for (j = 0; u = lnk[j]; j = vt) {
            ll d = inf; vis[j] = 1;
            for (int v = 1; v <= nr; ++v) {
                ll t = a[u] + b[v] - w[u][v];
                if (vis[v]) continue;
                if (sl[v] > t) sl[v] = t, pre[v] = j;
                if (sl[v] < d) d = sl[v], vt = v;
            }
            for (int v = 0; v <= nr; ++v) {
                if (vis[v]) a[lnk[v]] -= d, b[v] += d;
                else sl[v] -= d;
            }
        }
        for (; j; j = pre[j]) lnk[j] = lnk[pre[j]];
    }
    vector<pair<pii, ll>> res;
    for (int i = 1; i <= nr; ++i)
        res.emplace_back(pii(lnk[i], i), a[lnk[i]] + b[i]);
    return res;
}

}
```

Dinic

最大流

```cpp
const ll inf = LLONG_MAX;
struct edge { int v, p; ll c, f; };
vector<edge> g[N];
int cur[N], dis[N], nc;

int addv(int cnt) {
    while (cnt--) {
        int p = ++nc;
        g[p].resize(0);
    }
    return nc;
}

void adde(int u, int v, ll c) {
    g[u].push_back({ v, g[v].size(), c, 0 });
    g[v].push_back({ u, g[u].size() - 1, c, c });
}

bool bfs(int s, int t) {
    fill_n(dis + 1, nc, INT_MAX); queue<int> q;
    q.push(s); dis[s] = 0;
    while (!q.empty()) {
        int u = q.front(); q.pop();
        for (edge e : g[u]) {
            if (dis[e.v] != INT_MAX || e.c == e.f) continue;
            dis[e.v] = dis[u] + 1; q.push(e.v);
        }
    }
    return dis[t] != INT_MAX;
}

ll dfs(int u, int t, ll df) {
    if (u == t) return df; ll sf = 0;
    for (int& i = cur[u]; i != g[u].size(); ++i) {
        edge& e = g[u][i]; int v = e.v;
        if (dis[v] != dis[u] + 1 || e.c == e.f) continue;
        ll f = dfs(v, t, min(df, e.c - e.f));
        sf += f; e.f += f; g[v][e.p].f -= f; df -= f;
        if (!df) break;
    }
    return sf;
}

ll dinic(int s, int t, ll f = inf) {
    ll sf = 0;
    while(bfs(s, t)) {
        fill_n(cur + 1, nc, 0);
        ll df = dfs(s, t, f);
        sf += df; f -= df;
    }
    return sf;
}
```

ISAP

最大流

```
typedef long long ll;
const ll inf = LLONG_MAX;

struct edge { int p, v; ll c, f; };
vector<edge> g[N];
int cur[N], dis[N], gap[N], nc;

void fclr() { nc = 0; }

int addv(int cnt = 1) {
    while(cnt--) g[++nc].clear();
    return nc;
}

void adde(int u, int v, ll c) {
    g[u].push_back({ g[v].size(), v, c, 0 });
    g[v].push_back({ g[u].size() - 1, u, c, c });
}

ll dfs_isap(int s, int t, int u, ll f) {
    if (u == t) return f;
    ll sf = 0;
    for (int& i = cur[u]; i != g[u].size(); ++i) {
        edge& e = g[u][i]; int v = e.v;
        if (e.c == e.f || dis[u] != dis[v] + 1) continue;
        ll df = dfs_isap(s, t, v, min(f, e.c - e.f));
        e.f += df; sf += df; g[v][e.p].f -= df; f -= df;
        if (!f) return sf;
    }
    if (!--gap[dis[u]]) dis[s] = nc + 1;
    gap[++dis[u]]++; cur[u] = 0;
    return sf;
}

ll isap(int s, int t, ll f = inf) {
    ll sf = 0; queue<int> q;
    fill_n(dis + 1, nc, -1);
    fill_n(cur + 1, nc, 0);
    fill_n(gap + 1, nc, 0);
    gap[0] = 1; dis[t] = 0; q.push(t);
    while(!q.empty()) {
        int u = q.front(); q.pop();
        for (edge e : g[u]) {
            int v = e.v;
            if (g[v][e.p].c == g[v][e.p].f || dis[v] != -1) continue;
            dis[v] = dis[u] + 1; gap[dis[v]]++; q.push(v);
        }
    }
    while(dis[s] < nc) sf += dfs_isap(s, t, s, f);
    return sf;
}
```

MCMF-SPFA

最小费用最大流

```cpp
typedef long long ll;

const ll inf = LLONG_MAX;
struct edge { int v, p; ll c, f, w; };
vector<edge> g[N];
int cur[N], nc;

int addv(int cnt) {
    while (cnt--) {
        int p = ++nc;
        g[p].resize(0);
    }
    return nc;
}

void adde(int u, int v, ll c, ll w) {
    g[u].push_back({ v, g[v].size(), c, 0, w });
    g[v].push_back({ u, g[u].size() - 1, c, c, -w });
}


bool inq[N]; ll dis[N];
bool spfa_mcmf(int s, int t) {
    fill_n(dis + 1, nc, LLONG_MAX);
    queue<int> q; q.push(s); inq[s] = 1; dis[s] = 0;
    while (!q.empty()) {
        int u = q.front(); q.pop(); inq[u] = 0;
        for (edge e : g[u]) {
            int v = e.v; ll dv = e.w + dis[u];
            if (dis[v] <= dv || e.c == e.f) continue;
            dis[v] = dv; if (!inq[v]) q.push(v), inq[v] = 1;
        }
    }
    return dis[t] != LLONG_MAX;
}

ll dfs_mcmf(int u, int t, ll f) {
    if (u == t) return f; ll sf = 0; inq[u] = 1;
    for (int& i = cur[u]; i != g[u].size(); ++i) {
        edge& e = g[u][i]; int v = e.v;
        if (e.c == e.f || dis[u] + e.w != dis[v] || inq[v]) continue;
        ll df = dfs_mcmf(v, t, min(f, e.c - e.f));
        e.f += df; sf += df; f -= df; g[v][e.p].f -= df;
        if (!f) break;
    }
    inq[u] = 0; return sf;
}

pair<ll, ll> mcmf(int s, int t, ll mf = inf) {
    ll sf = 0, sc = 0;
    while (spfa_mcmf(s, t)) {
        fill_n(cur + 1, nc, 0);
        ll df = dfs_mcmf(s, t, mf);
        mf -= df; sf += df;
        sc += dis[t] * df;
    }
    return { sf, sc };
}
```

MCMF-Dijkstra

最小费用最大流（原始对偶）

```
typedef long long ll;

const ll inf = LLONG_MAX;
struct edge { int v, p; ll c, f, w; };
vector<edge> g[N];
int cur[N], nc;

int addv(int cnt) {
    while (cnt--) {
        int p = ++nc;
        g[p].resize(0);
    }
    return nc;
}

void adde(int u, int v, ll c, ll w) {
    g[u].push_back({ v, g[v].size(), c, 0, w });
    g[v].push_back({ u, g[u].size() - 1, c, c, -w });
}

bool inq[N]; ll dis[N], pot[N];
bool dijkstra(int s, int t) {
    fill_n(dis + 1, nc, LLONG_MAX);
    typedef pair<ll, int> pli;
    priority_queue<pli, vector<pli>, greater<pli>> pq;
    pq.push({ dis[s] = 0, s });
    while (!pq.empty()) {
        pli p = pq.top(); pq.pop();
        ll du = p.first; int u = p.second;
        if (dis[u] < du) continue;
        for (edge e : g[u]) {
            int v = e.v; if (e.c == e.f) continue;
            ll dv = du + e.w + pot[u] - pot[v];
            if (dis[v] > dv) pq.push({ dis[v] = dv, v });
        }
    }
    return dis[t] != inf;
}

ll dfs_mcmf(int u, int t, ll f) {
    if (u == t) return f; ll sf = 0; inq[u] = 1;
    for (int& i = cur[u]; i != g[u].size(); ++i) {
        edge& e = g[u][i]; int v = e.v;
        if (e.c==e.f||dis[u]+e.w+pot[u]-pot[v]!=dis[v]||inq[v])continue;
        ll df = dfs_mcmf(v, t, min(f, e.c - e.f));
        e.f += df; sf += df; f -= df; g[v][e.p].f -= df;
        if (!f) break;
    }
    inq[u] = 0; return sf;
}
```

```
pair<ll, ll> mcmf(int s, int t, ll mf = inf) {
    ll sf = 0, sc = 0;
    fill_n(pot + 1, nc, 0);
    while (dijkstra(s, t)) {
        fill_n(cur + 1, nc, 0);
        ll df = dfs_mcmf(s, t, mf);
        mf -= df; sf += df;
        sc += (dis[t] + pot[t]) * df;
        for (int i = 1; i <= nc; ++i) pot[i] += dis[i];
    }
    return { sf, sc };
}
```

带小数的费用流

```
#include <algorithm>
#include <cctype>
#include <climits>
#include <cmath>
#include <cstdio>
#include <cstring>
#include <iostream>
#include <queue>
#include <stack>
#include <vector>
#define LL long long
#define P pair<int, int>
using namespace std;
template <typename T>
inline void read(T& t)
{
    int f = 0, c = getchar();
    t = 0;
    while (!isdigit(c))
        f |= c == '-', c = getchar();
    while (isdigit(c))
        t = t * 10 + c - 48, c = getchar();
    if (f)
        t = -t;
}
template <typename T, typename... Args>
inline void read(T& t, Args&... args)
{
    read(t);
    read(args...);
}
const int maxn = 5005;
const double inf = 1e9;
struct Edge {
    int from, to, cap, flow;
    double cost;
    Edge(int from, int to, int cap, double cost)
    {
        this->from = from;
        this->to = to;
        this->cap = cap;
        this->flow = 0;
        this->cost = cost;
    }
};

struct MMMF {
    vector<Edge> edges;
    vector<int> G[maxn];
    double d[maxn];
    int p[maxn];
    bool inq[maxn];
    int n, s, t;
    void addEdge(int from, int to, int cap, double cost)
    {
        Edge e1(from, to, cap, cost), e2(to, from, 0, -cost);
        edges.push_back(e1);
        edges.push_back(e2);
        int m = edges.size();
        G[from].push_back(m - 2);
        G[to].push_back(m - 1);
    }
    int Fini()
    {
        int x = t;
        int a = inf;
        while (x != s) {
            Edge& e = edges[p[x]];
            a = min(a, e.cap - e.flow);
            x = e.from;
        }
        x = t;
        while (x != s) {
            Edge& e = edges[p[x]];
            e.flow += a;
            edges[p[x] ^ 1].flow -= a;
            x = e.from;
        }
        return a;
    }
    int maxflow = 0;
    double mincost = 0.0;
    bool SPFA()
    {
        queue<int> q;
        for (int i = 1; i <= n; i++)
            d[i] = inf, inq[i] = 0;
        ;
        d[s] = 0;
        q.push(s);
        inq[s] = 1;
        while (!q.empty()) {
            int x = q.front();
           // cout << d[x] << endl;
            q.pop();
            inq[x] = 0;
            for (int i = 0; i < G[x].size(); i++) {
                Edge& e = edges[G[x][i]];
                if (e.cap > e.flow) {
                    if (d[e.to] > d[x] + e.cost + 1e-5) {
                        d[e.to] = d[x] + e.cost;
                        p[e.to] = G[x][i];
                        if (!inq[e.to]) {
                            q.push(e.to);
                            inq[e.to] = 1;
                        }
                    }
                }
            }
        }
        int a = Fini();
        if (d[t] > 1e8)
            return false;
        maxflow += a;
        mincost += a * d[t];
        return true;
    }
    void maxFlow(int s, int t, int n)
    {
        this->s = s;
        this->t = t;
        this->n = n;
        while (SPFA())
            ;
        //      cerr << mincost << endl;
    }
};
int S[1002], D[1005];
int main()
{
    //  freopen("g.in", "r", stdin);
    int T;
    cin >> T;
    while (T--) {
        MMMF mmmf;
        int n, m;
        cin >> n >> m;
        int s = 2 * n + 1, t = s + 1;
        for (int i = 1; i   <= n; ++i) {
            scanf("%d%d", &S[i], &D[i]);
            int minn = min(S[i], D[i]);
            S[i] -= minn;
            D[i] -= minn;
            if (S[i]) {
                mmmf.addEdge(s, i, S[i], 0);
            }
            if (D[i]) {
                mmmf.addEdge(i, t, D[i], 0);
            }
        }
        for (int i = 1; i <= m; ++i) {
            int x, y, w;
            double cost;
            cin >> x >> y >> w >> cost;
            cost = log(1 - cost);
            if (w > 0) {
                mmmf.addEdge(x, y, 1, 0);
            }
            if (w > 1) {
                mmmf.addEdge(x, y, w - 1, -cost);
            }
        }
        mmmf.maxFlow(s, t, t);
        double ans = 1 - exp(-mmmf.mincost);
        printf("%.2lf\n", ans);
    }
}
```

Edmonds(Blossom)

其他

HLPP

```
typedef long long ll;

const ll inf = LLONG_MAX;
struct edge { int v, p; ll c, f; };
vector<edge> g[N];
int h[N], gap[N << 1], nc; ll ef[N]; bool inq[N];
typedef priority_queue<int, vector<int>, function<bool(int,int)>> pq;

int addv(int cnt) {
    while (cnt--) {
        int p = ++nc; g[p].resize(0);
        ef[p] = h[p] = gap[p] = inq[p] = 0;
    }
    return nc;
}

void adde(int u, int v, ll c) {
    g[u].push_back({ v, g[v].size(), c, 0 });
    g[v].push_back({ u, g[u].size() - 1, c, c });
}

bool bfs(int s, int t) {
    fill_n(h + 1, nc, INT_MAX);
    queue<int> q; q.push(t); h[t] = 0;
    while (!q.empty()) {
        int u = q.front(); q.pop();
        for (edge e : g[u])
            if (e.f && h[e.v] == INT_MAX)
                h[e.v] = h[u] + 1, q.push(e.v);
    }
    return h[s] != INT_MAX;
}

void push(pq& q, int u, bool flg = 0) {
    for (edge& e : g[u]) {
        if (e.c == e.f || h[e.v] == INT_MAX || (!flg && h[e.v] + 1 != h[u]))
continue;
        ll df = min(ef[u], e.c - e.f);
        e.f += df; g[e.v][e.p].f -= df;
        ef[u] -= df; ef[e.v] += df;
        if (!inq[e.v]) q.push(e.v), inq[e.v] = 1;
        if (!ef[u]) break;
    }
}

ll hlpp(int s, int t, ll mf = inf) {
    if (!bfs(s, t)) return 0; h[s] = nc; inq[s] = inq[t] = 1;
    pq q([&](int i1, int i2) { return h[i1] < h[i2]; });
    ef[s] = mf; for (int i = 1; i <= nc; ++i)
        if (h[i] != INT_MAX) gap[h[i]]++;
    push(q, s, 1); while (!q.empty()) {
        int u = q.top(); q.pop(); inq[u] = 0;
        push(q, u); if (!ef[u]) continue;
        if (!--gap[h[u]]) for (int i = 1; i <= nc; ++i)
            if (i!=s&&i!=t&&h[u]<h[i])h[i]=max(h[i],nc+1);
        h[u] = 2 * nc;
        for (edge e : g[u]) if (e.c != e.f && h[e.v] != INT_MAX)
            h[u] = min(h[u], h[e.v] + 1);
        ++gap[h[u]]; q.push(u); inq[u] = 1;
    }
    return mf - ef[s];
}
```

带上下界的网络流

带上下界的最大流

```
ll df[N];
void adde_c(int u, int v, ll cl, ll cr) {
    adde(u, v, cr - cl);
    df[u] -= cl;
    df[v] += cl;
}

bool feasible_flow() {
    int s = addv(1), t = addv(1);
    ll se = 0;
    for (int i = 1; i <= nc - 2; ++i) {
        if (!df[i]) continue;
        if (df[i] > 0) adde(s, i, df[i]), se += df[i];
        else adde(i, t, -df[i]);
    }
    se -= isap(s, t, se); nc -= 2;
    for (int i = 1; i <= nc; ++i)
        while (g[i].back().v > nc) g[i].pop_back();
    return !se;
}

ll max_flow(int s, int t) {
    adde(t, s, inf);
    if (!feasible_flow()) return -1;
    return isap(s, t);
}

ll min_flow(int s, int t) {
    adde(t, s, inf);
    int i1 = g[t].size() - 1, i2 = g[s].size() - 1;
    if (!feasible_flow()) return -1;
    ll sf = g[t][i1].f;
    g[t][i1].c = g[t][i1].f = 0;
    return sf - isap(t, s);
}
```

km(N^3)

```
const ll Maxn=505;
const ll inf=1e18;
ll n,m,map[Maxn][Maxn],matched[Maxn];
ll slack[Maxn],pre[Maxn],ex[Maxn],ey[Maxn];//ex,ey顶标
bool visx[Maxn],visy[Maxn];
void match(ll u)
{
    ll x,y=0,yy=0,delta;
    memset(pre,0,sizeof(pre));
    for(ll i=1;i<=n;i++)slack[i]=inf;
    matched[y]=u;
    while(1)
    {
        x=matched[y];delta=inf;visy[y]=1;
        for(ll i=1;i<=n;i++)
        {
            if(visy[i])continue;
            if(slack[i]>ex[x]+ey[i]-map[x][i])
            {
                slack[i]=ex[x]+ey[i]-map[x][i];
                pre[i]=y;
            }
            if(slack[i]<delta){delta=slack[i];yy=i;}
        }
        for(ll i=0;i<=n;i++)
        {
            if(visy[i])ex[matched[i]]-=delta,ey[i]+=delta;
            else slack[i]-=delta;
        }
        y=yy;
        if(matched[y]==-1)break;
    }
    while(y){matched[y]=matched[pre[y]];y=pre[y];}
}
ll KM()
{
    memset(matched,-1,sizeof(matched));
    memset(ex,0,sizeof(ex));
    memset(ey,0,sizeof(ey));
    for(ll i=1;i<=n;i++)
    {
        memset(visy,0,sizeof(visy));
        match(i);
    }
    ll res=0;
    for(ll i=1;i<=n;i++)
        if(matched[i]!=-1)res+=map[matched[i]][i];
    return res;
}
```

SW算法求全局最小割

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 601;
int fa[N], siz[N], edge[N][N];
int find(int x) { return fa[x] == x ? x : fa[x] = find(fa[x]); }
int dist[N], vis[N], bin[N];
int n, m;
int contract(int &s, int &t) {  // Find s,t
  memset(dist, 0, sizeof(dist));
  memset(vis, false, sizeof(vis));
  int i, j, k, mincut, maxc;
  for (i = 1; i <= n; i++) {
    k = -1;
    maxc = -1;
    for (j = 1; j <= n; j++)
      if (!bin[j] && !vis[j] && dist[j] > maxc) {
        k = j;
        maxc = dist[j];
      }
    if (k == -1) return mincut;
    s = t;
    t = k;
    mincut = maxc;
    vis[k] = true;
    for (j = 1; j <= n; j++)
      if (!bin[j] && !vis[j]) dist[j] += edge[k][j];
  }
  return mincut;
}
const int inf = 0x3f3f3f3f;
int Stoer_Wagner() {
  int mincut, i, j, s, t, ans;
  for (mincut = inf, i = 1; i < n; i++) {
    ans = contract(s, t);
    bin[t] = true;
    if (mincut > ans) mincut = ans;
    if (mincut == 0) return 0;
    for (j = 1; j <= n; j++)
      if (!bin[j]) edge[s][j] = (edge[j][s] += edge[j][t]);
  }
  return mincut;
}
int main() {
  ios::sync_with_stdio(0), cin.tie(0);
  cin >> n >> m;
  if (m < n - 1) {
    cout << 0;
    return 0;
  }
  for (int i = 1; i <= n; ++i) fa[i] = i, siz[i] = 1;
  for (int i = 1, u, v, w; i <= m; ++i) {
    cin >> u >> v >> w;
    int fu = find(u), fv = find(v);
    if (fu != fv) {
      if (siz[fu] > siz[fv]) swap(fu, fv);
      fa[fu] = fv, siz[fv] += siz[fu];
    }
    edge[u][v] += w, edge[v][u] += w;
  }
  int fr = find(1);
  if (siz[fr] != n) {
    cout << 0;
    return 0;
  }
  cout << Stoer_Wagner();
  return 0;
}
```

LGV引理

```
#include <bits/stdc++.h>
#define F(i, l, r) for(int i = (l), _end_ = (int)(r); i <= _end_; ++i)
#define f(i, r, l) for(int i = (r), _end_ = (int)(l); i >= _end_; --i)
#define Set(a, v) memset(a, v, sizeof(a))
#define file(a) freopen(a".in","r",stdin),freopen(a".out","w",stdout)
using namespace std;
bool chkmin(long long &a, long long b) {return b < a ? a = b, 1 : 0;}
bool chkmax(long long &a, long long b) {return b > a ? a = b, 1 : 0;}
inline long long read() {
 long long x = 0, fh = 1; char ch = getchar();
    for (; !isdigit(ch); ch = getchar() ) if (ch == '-') fh = -1;
    for (; isdigit(ch); ch = getchar() ) x = (x<<1) + (x<<3) + (ch ^ '0');
    return x * fh;
}
int k;
int ls[1005];
int cnt[1005];
const int mod=998244353;
struct node{
    int l,r;
    int c[205][205];
    node operator *(const node &rhs)const{
        node res;
        res.l=l,res.r=rhs.r;
        F(i,1,l)F(j,1,rhs.r)res.c[i][j]=0;
        F(i,1,l)F(j,1,res.r)F(k,1,rhs.l)res.c[i][j]=(res.c[i][j]+1ll*c[i]
[k]*rhs.c[k][j]%mod)%mod;
        return res;
    }
}p[305];
int getl(node now){
    int res=1,w=1;
    F(i,1,now.l){
        F(j,i+1,now.r){
            while(now.c[i][i]){
                int div=now.c[j][i]/now.c[i][i];
                F(k,i,now.l){
                    now.c[j][k]=(now.c[j][k]-1ll*div*now.c[i][k]%mod+mod)%mod;
                }
                swap(now.c[i],now.c[j]);
                w=-w;
            }
            swap(now.c[i],now.c[j]);
            w=-w;
        }
    }
    F(i,1,now.l)res=1ll*now.c[i][i]*res%mod;
    res=1ll*w*res;
    return (res+mod)%mod;
}
int main(){
#ifndef ONLINE_JUDGE
    file("a");
#endif
```

```
    int t=read();
    while(t--){
        k=read();
        F(i,1,k)cnt[i]=read();
        F(i,1,k-1){
            p[i].l=cnt[i];
            p[i].r=cnt[i+1];
            F(x,1,p[i].l){
                F(y,1,p[i].r){
                    p[i].c[x][y]=0;
                }
            }
        }
        F(i,1,k-1){
            ls[i]=read();
        }
        F(i,1,k-1){
            F(j,1,ls[i]){
                int x=read(),y=read();
                p[i].c[x][y]++;
            }
        }
        node now=p[1];
        F(i,2,k-1){
            now=now*p[i];
        }
        int ans=getl(now);
        printf("%d\n",ans);
    }
    return 0;
}
```