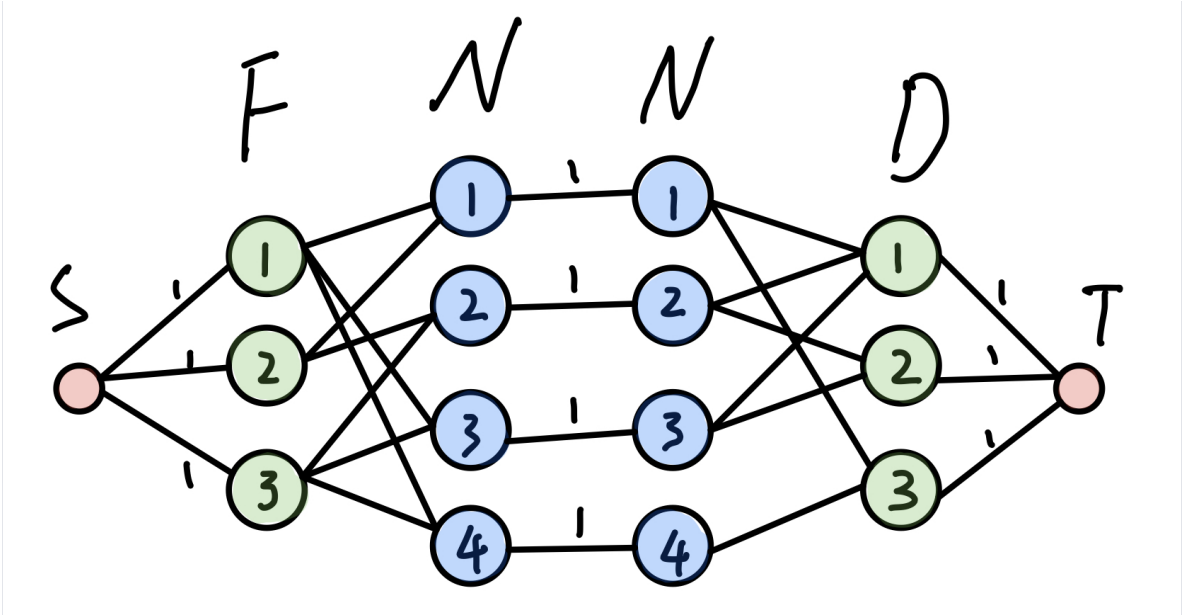
B - Dining(POJ-3281)

题意

每头牛都有各自喜欢的食物和饮料，而每种食物或饮料只能分配给一头牛。最多能有多少头牛可以同时得到喜欢的食物和饮料？

思路

最大流
将牛拆成点，限制牛的流量。保证增光路上对应得是牛喜欢的食物和水




```
#include <iostream>
#include <queue>
#include <cstring>
const int maxn = 4e2 + 5;
const int inf = 0x3f3f3f3f;
using namespace std;
struct ac{
    int v, c, nex;
}edge[maxn << 4];
int head[maxn], curedge[maxn], cnt, s, e;
int dis[maxn];
void init() {
    memset(head, -1, sizeof(head));
    cnt = 0;
}
void addedge(int u, int v, int c) {
    edge[cnt] = {v, c, head[u]};
    head[u] = cnt++;
    edge[cnt] = {u, 0, head[v]};
    head[v] = cnt++;
}
int bfs() {
    memset(dis, 0, sizeof(dis));
    dis[s] = 1;
    queue<int> que;
```

```
    que.push(s);
    while (!que.empty()) {
        int u = que.front();
        que.pop();
        for (int i = head[u]; i != -1; i = edge[i].nex) {
            int v = edge[i].v;
            int c = edge[i].c;
            if (dis[v] || c == 0) continue;
            dis[v] = dis[u] + 1;
            que.push(v);
        }
    }
    return dis[e] > 0;
}
int dfs(int u, int flow) {
    if (u == e || flow == 0) return flow;
    for (int &i = curedge[u]; i != -1; i = edge[i].nex) {
        int v = edge[i].v;
        int c = edge[i].c;
        if (dis[v] != dis[u] + 1) continue;
        int tmp = dfs(v, min(flow, c));
        if (tmp > 0) {
            edge[i].c -= tmp;
            edge[i^1].c += tmp;
            return tmp;
        }
    }
    dis[u] = -1;
    return 0;
}
int Dinic() {
    int ans = 0, tmp;
    while (bfs()) {
        for (int i = 0; i <= e; ++i) curedge[i] = head[i];
        while ((tmp = dfs(s, inf)) > 0) ans += tmp;
    }
    return ans;
}
int main () {
    ios::sync_with_stdio(false);
    cin.tie(0), cout.tie(0);
    int n, f, d;
    while (cin >> n >> f >> d) {
        init();
        s = 0, e = n*2 + f + d + 1;
        for (int i = 1; i <= n; ++i) {
            int a, b, t;
            cin >> a >> b;
            for (int j = 1; j <= a; ++j) {
                cin >> t;
                addedge(t, f+i, 1);
            }
            for (int j = 1; j <= b; ++j) {
                cin >> t;
                addedge(f+n+i, n*2+f+t, 1);
            }
        }
        for (int j = 1; j <= f; ++j) addedge(s, j, 1);
```

```
        for (int j = 1; j <= d; ++j) addedge(n*2+f+j, e, 1);
        for (int j = 1; j <= n; ++j) addedge(f+j, f+n+j, 1);
        cout << Dinic() << endl;
    }
    return 0;
}
```

D - Going Home(POJ-2195)

题意

n个人要进到房子里面，每个人的花费是他移动的距离。求n个人进房子的最小花费

思路

费用流

```
#include <iostream>
#include <queue>
#include <cmath>
#include <cstring>
const int maxn = 1e4 + 5;
const int inf = 0x3f3f3f3f;
using namespace std;
int path[maxn], dis[maxn], head[maxn], vis[maxn];
int cnt, s, e;
void init() {
    memset(head, -1, sizeof(head));
    cnt = 0;
}
struct ac{
    int v, c, cost, nex;
}edge[maxn << 11];
void addedge(int u, int v, int c, int cost) {
    edge[cnt] = {v, c, cost, head[u]};
    head[u] = cnt++;
    edge[cnt] = {u, 0, -cost, head[v]};
    head[v] = cnt++;
}
int spfa(int s, int e) {
    memset(vis, 0, sizeof(vis));
    memset(dis, inf, sizeof(dis));
    memset(path, -1, sizeof(path));
    queue<int> que;
    que.push(s);
    dis[s] = 0;
    vis[s] = 1;
    while (!que.empty()) {
        int u = que.front();
        que.pop();
        vis[u] = 0;
        for (int i = head[u]; i != -1; i = edge[i].nex) {
            int v = edge[i].v;
            int c = edge[i].c;
            int cost = edge[i].cost;
            if (dis[v] > dis[u] + cost && c > 0) {
                dis[v] = dis[u] + cost;
```

```
                path[v] = i;
                if (vis[v]) continue;
                vis[v] = 1;
                que.push(v);
            }
        }
    }
    return dis[e] != inf;
}
int MincostMaxflow(int s, int e, int &cost) {
    int maxflow = 0;
    while (spfa(s, e)) {
        int flow = inf;
        for (int i = path[e]; i != -1; i = path[edge[i^1].v]) {
            flow = min(flow, edge[i].c);
        }
        for (int i = path[e]; i != -1; i = path[edge[i^1].v]) {
            edge[i].c -= flow;
            edge[i^1].c += flow;
            cost += flow * edge[i].cost;
        }
        maxflow += flow;
    }
    return maxflow;
}
int main () {
    ios::sync_with_stdio(false);
    cin.tie(0), cout.tie(0);
    int n, m; char c;
    while (cin >> n >> m, n) {
        init();
        vector<pair<int,int> > H, M;
        for(int i = 0; i < n; ++i) {
            for (int j = 0; j < m; ++j) {
                cin >> c;
                if (c == 'm') M.push_back(make_pair(i, j));
                if (c == 'H') H.push_back(make_pair(i, j));
            }
        }
        int cntH = H.size();
        int cntM = M.size();
        s = 0, e = cntH + cntM + 1;
        for (int i = 0; i < cntM; ++i) {
            addedge(s, i+1, 1, 0);
        }
        for (int i = 0; i < cntH; ++i) {
            addedge(cntM+i+1, e, 1, 0);
        }
        for (int i = 0; i < cntH; ++i) {
            for (int j = 0; j < cntM; ++j) {
                int cost = fabs(H[i].first - M[j].first) + fabs(H[i].second -
M[j].second);
                addedge(j+1, cntM+i+1, 1, cost);
            }
        }
        int cost = 0;
        MincostMaxflow(s, e, cost);
        cout << cost << endl;
```

```
    }
    return 0;
}
```