Dinic

复杂度

$$O = \begin{cases} V^2 * E & \{V, E\} \in \text{一般图} \\ \sqrt{V} * E & \{V, E\} \in \text{二分图} \end{cases}$$

```cpp
#include <bits/stdc++.h>

using namespace std;
#define maxn 250
#define INF 0x3f3f3f3f

struct Edge {
    int from, to, cap, flow;
    Edge() {}
    Edge(int u, int v, int c, int f) : from(u), to(v), cap(c), flow(f) {}
};

struct Dinic {
    int n, m, s, t;
    vector<Edge> edges;
    vector<int> G[maxn];
    int d[maxn], cur[maxn];
    bool vis[maxn];

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

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

    bool BFS() {
        memset(vis, 0, sizeof(vis));
        queue<int> Q;
        Q.push(s);
        d[s] = 0;
        vis[s] = 1;
        while(!Q.empty()) {
            int x = Q.front();
            Q.pop();
            for (int i = 0; i < G[x].size(); i++) {
```

```
                Edge& e = edges[G[x][i]];
                if(!vis[e.to] && e.cap > e.flow) {
                    vis[e.to] = 1;
                    d[e.to] = d[x] + 1;
                    Q.push(e.to);
                }
            }
        }
        return vis[t];
    }

    int DFS(int x, int a) {
        if(x == t || a == 0) return a;
        int flow = 0, f;
        for (int& i = cur[x]; i < G[x].size(); i++) {
            Edge& e = edges[G[x][i]];
            if(d[x]+1 == d[e.to] && (f = DFS(e.to, min(a, e.cap-e.flow)))> 0) {
                e.flow += f;
                edges[G[x][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;
        while(BFS()) {
            memset(cur, 0, sizeof(cur));
            flow += DFS(s, INF);
        }
        return flow;
    }
};

set<int> a, b;
int main()
{
    Dinic dinic;
    int n, m;
    scanf("%d%d", &n, &m);
    int l, r;
    while(~scanf("%d%d", &l, &r)) {
        dinic.AddEdge(l, r, 1);
        a.insert(l);
```

```
        b.insert(r);
    }
    int s = 0, t = n+1;
    for(auto it : a) dinic.AddEdge(s, it, 1);
    for(auto it : b) dinic.AddEdge(it, t, 1);
    int ans = dinic.Maxflow(s, t);
    printf("%d\n", ans);
    return 0;
}
```