联合训练图论场题解报告

A.Euler

题意：略

分析：

这题主要是先掌握欧拉通路的概念，然后是如何判断图是否存在欧拉通路。
欧拉通路：通过图中每条边且只通过一次，并且经过每一顶点的通路。
欧拉回路：通过图中每条边且只通过一次，并且经过每一顶点的回路。
无向图：
    欧拉通路：连通图＋只存在0个或者两个度数为奇数的点。
    欧拉回路：连通图＋所有节点的度数均为偶数。
有向图：
    欧拉通路：连通图＋（所有点的入度＝出度 || 出两个点之外其他点的入度＝出度，一个点的入度－出度＝1，一个点的出度－入度＝1）。

```
ii e[500*500];
int in[510], out[510];//indegree,outdegree
int f[510];//判断是否连通
int find(int x) {
    return f[x] == -1 ? x : f[x] = find(f[x]);
}
int main(int argc, const char * argv[])
{
    // freopen("in.txt","r",stdin);
    // freopen("out.txt","w",stdout);
    // clock_t _ = clock();

    int t, n, m;
    scanf("%d", &t);
    while(t--) {
        scanf("%d%d", &n, &m);
        for (int i = 0;i < m;++i)
            scanf("%d%d", &e[i].first, &e[i].second);
        memset(in, 0,sizeof in);
        memset(out, 0,sizeof out);
        memset(f, -1,sizeof f);
        int cnt = 0;
        for (int i = 0;i < m;++i) {
            in[e[i].first]++;
            in[e[i].second]++;
            int t1 = find(e[i].first);
            int t2 = find(e[i].second);
            if (t1 != t2) f[t1] = t2;
        }
        int o=0;
        for (int i = 1;i <= n;++i) {
            find(i);
            if (f[i] == -1) o++;
        }//o == 说明图连通
        for (int i = 1;i <= n;++i)
            if (in[i] & 1) cnt++;
        if (cnt == 0 || cnt == 2) {
            if (o == 1) printf("Yes");
            else printf("No");
        }else printf("No");
        printf(" ");


        memset(in, 0,sizeof in);
        memset(f, -1,sizeof f);
        cnt = 0;
        for (int i = 0;i < m;++i) {
            out[e[i].first]++;
            in[e[i].second]++;
        }
        if (o > 1) puts("No");
        else {
            vector<int> vec;
            for (int i = 1;i <= n;++i) {
                if (in[i] != out[i]) vec.push_back(i);
            }
            if (vec.size() != 2 && vec.size() != 0) puts("No");
            else {
                if (vec.size() == 0) {
                    puts("Yes");
                    continue;
                }
                int u = vec[0], v = vec[1];
                if (in[u] - out[u] == 1 && in[v] - out[v] == -1) puts("Yes");
                else if (in[u] - out[u] == -1 && in[v] - out[v] == 1) puts("Yes");
                else puts("No");
            }
        }
    }
    // printf("\nTime cost: %.2fs\n", 1.0 * (clock() - _) / CLOCKS_PER_SEC);
```

```
71.     return 0;
72. }
```

B.-0你的电脑炸了

题意

判断给出的图是否合法。

分析

4*4的格子中，每个位置会出现指定的某些数字，但是由于覆盖的作用，只会看见最上面的一个，其他的被压在了下面。A覆盖了B，B覆盖了A，这样是显然不成立的，这题就是判断是否会出现相
ps:初始化的表要仔细打。。。

```
1.  vector<int> have[5][5];
2.  int a[5][5];
3.  int in[10];
4.  vector<int> G[10];
5.  void init() {
6.      have[1][1].push_back(1);
7.      have[1][2].push_back(1), have[1][2].push_back(2);
8.      have[1][3].push_back(2), have[1][3].push_back(3);
9.      have[1][4].push_back(3);
10.
11.     have[2][1].push_back(1), have[2][1].push_back(4);
12.     have[2][2].push_back(1), have[2][2].push_back(2), have[2][2].push_back(4), have[2][2].push_back(5);
13.     have[2][3].push_back(2), have[2][3].push_back(3), have[2][3].push_back(5), have[2][3].push_back(6);
14.     have[2][4].push_back(3), have[2][4].push_back(6);
15.
16.     have[3][1].push_back(4), have[3][1].push_back(7);
17.     have[3][2].push_back(4), have[3][2].push_back(5), have[3][2].push_back(7), have[3][2].push_back(8);
18.     have[3][3].push_back(5), have[3][3].push_back(6), have[3][3].push_back(8), have[3][3].push_back(9);
19.     have[3][4].push_back(6), have[3][4].push_back(9);
20.
21.     have[4][1].push_back(7);
22.     have[4][2].push_back(7), have[4][2].push_back(8);
23.     have[4][3].push_back(8), have[4][3].push_back(9);
24.     have[4][4].push_back(9);
25. }
26. void done(int u,int i, int j) {
27.     for (int k = 0;k < have[i][j].size();++k) {
28.         int v = have[i][j][k];
29.         if (u == v) continue;
30.         G[u].push_back(v);
31.         in[v]++;
32.     }
33. }
34. void getmap() {
35.     for (int i = 1;i <= 9;++i) {
36.         G[i].clear();
37.         in[i] = 0;
38.     }
39.     for (int i = 1;i <= 4;++i) {
40.         for (int j = 1;j <= 4;++j)
41.             done(a[i][j], i, j);
42.     }
43. }
44. void solve() {
45.     getmap();
46.     queue<int> que;
47.     for (int i = 1;i <= 9;++i) {
48.         if (in[i] == 0) {
49.             que.push(i);
50.         }
51.     }
52.     while(!que.empty()) {
53.         int u = que.front();
54.         que.pop();
55.         for (int i = 0;i < G[u].size();++i) {
56.             int v = G[u][i];
57.             if (--in[v] == 0) {
58.                 que.push(v);
59.             }
60.         }
61.     }
62.     bool ok = true;
63.     for (int i = 1;i <= 9;++i)
64.         if (in[i] > 0) ok = false;
65.     if (ok) puts("Lucky dog!");
66.     else puts("BOOM!");
67. }
68. int main(int argc, const char * argv[])
69. {
70.     // freopen("in.txt","r",stdin);
71.     // freopen("out.txt","w",stdout);
72.     // clock_t _ = clock();
73.
74.     init();
75.     int T;
76.     cin >> T;
77.     while(T--) {
78.         for (int i = 1;i <= 4;++i) {
79.             for (int j = 1;j <= 4;++j)
```

```
                scanf("%d", &a[i][j]);
        }
        solve();
    }

    // printf("\nTime cost: %.2fs\n", 1.0 * (clock() - _) / CLOCKS_PER_SEC);
    return 0;
}
```

C.寻找fly真迹

题意

略

分析

首先用补图还是很容易想到的（毕竟正着搞相对复杂了）。建立补图之后很容易就想到了二分图染色，最后判断是否成立。

```
const int maxn = 5e2 + 10;
int n, m;
int col[maxn];
vector<int> G[maxn];
int g[maxn][maxn];
bool dfs(int u,int color) {
    col[u] = color;
    for (int i = 0;i < G[u].size();++i) {
        int v = G[u][i];
        if (col[u] == col[v]) return false;
        if (!col[v] && !dfs(v, -color)) return false;
    }
    return true;
}
int main(int argc, const char * argv[])
{
    // freopen("in.txt","r",stdin);
    // freopen("out.txt","w",stdout);
    // clock_t _ = clock();

    int t;
    cin >> t;
    while(t--) {
        scanf("%d%d", &n, &m);
        memset(g, 0,sizeof g);
        int u, v;
        for (int i = 1;i <= m;++i) {
            scanf("%d%d", &u, &v);
            g[u][v] = g[v][u] = 1;
        }
        for (int i = 1;i <= n;++i) {
            G[i].clear();
            for (int j = 1;j <= n;++j)
                if (i != j && !g[i][j]) G[i].push_back(j);
        }
        bool yes = true;
        memset(col, 0,sizeof col);
        for (int i = 1;i <= n;++i) {
            if (!col[i] && G[i].size() && !dfs(i, 1)) {
                yes=false;
            }
        }
        for (int i = 1;i <= n;++i) {
            for (int j = 1;j <= n;++j) {
                if (i == j) continue;
                if (g[i][j] && col[i]*col[j]<0)yes=false;
                if (!g[i][j] && col[i]*col[j]>=0)yes=false;
            }
        }
        if (yes) puts("Yes");
        else puts("No");
    }

    // printf("\nTime cost: %.2fs\n", 1.0 * (clock() - _) / CLOCKS_PER_SEC);
    return 0;
}
```

D.一食堂or二食堂，it's a question

题意

好懂，可能就“使得任意两人走过的距离加上二人所在食堂的距离的最大值最小”这句话需要解释下。任意两个人A，B 如果在同一个食堂的话就是他两个走过的距离和，如果不在同一个食堂就再

分析

求最大最小，显然二分结果值。然后根据二分值建图判断可行性。
我们可以用bool值表示每个人的选择。对于第i个人而言，i表示其选择第一食堂，i+N表示其选择二食堂。
<1> 相互憎恨的两个人x,y
必然建立四条边(x,y+N)，(y,x+N),(x+N,y),(y+N,x)。表示两个选择不同的食堂。

<2> 相互喜欢的两个人x,y
必然建立四条边(x,y),(y,x),(x+N,y+N),(y+N,x+N)表示两个人选择同一个食堂。
<3> 对于人意的两个人x,y
这里就看代码了。。。

```
/*****************************************
Author      :Crazy_AC(JamesQi)
Time        :2016
File Name   :
                     _ooOoo_
                    o8888888o
                    88" . "88
                    (| -_- |)
                    O\  =  /O
                 ____/`---'\____
               .'  \|     |//  `.
              /  \|||  :  |||//  \
             /  _|||||  -:-  |||||-  \
             |   | \\  -  /// |   |
             | \_|  ''\---/''  |   |
             \  .-\__  `-`  ___/-. /
           ___`. .'  /--.--\  `. . __
        ."" '<  `.___\_<|>_/___.'  >'"".
       | | :  `- \`.;`\ _ /`;.`/ - ` : | |
       \  \ `-.   \_ __\ /__ _/   .-` /  /
======`-.____`-.___\_____/___.-`____.-'======
                   `======'
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
          佛祖保佑       永无BUG
*****************************************/
// #pragma comment(linker, "/STACK:1024000000,1024000000")
#include <iostream>
#include <algorithm>
#include <iomanip>
#include <sstream>
#include <string>
#include <stack>
#include <queue>
#include <deque>
#include <vector>
#include <map>
#include <set>
#include <cstdio>
#include <cstring>
#include <cmath>
#include <cstdlib>
#include <climits>
using namespace std;
#define MEM(x,y) memset(x, y,sizeof x)
#define pk push_back
#define lson rt << 1
#define rson rt << 1 | 1
#define bug cout << "BUG HERE\n"
typedef long long LL;
typedef unsigned long long ULL;
typedef pair<int,int> ii;
typedef pair<ii,int> iii;
const double eps = 1e-8;
const double pi = 4 * atan(1);
const int inf = 1 << 30;
const int INF = 0x3f3f3f3f;
const int MOD = 1e9 + 7;
int nCase = 0;
int dcmp(double x){//精度正负、0的判断
    if (fabs(x) < eps) return 0;
    return x < 0?-1:1;
}
inline int read(){
    char c = getchar();
    while (!isdigit(c)) c = getchar();
    int x = 0;
    while (isdigit(c)) {
        x = x * 10 + c - '0';
        c = getchar();
    }
    return x;
}
const int maxn = 2010;
struct Edge{
    int to, nxt;
    Edge() {}
    Edge(int to,int nxt) {
        this->to = to;
        this->nxt = nxt;
    }
}edges[maxn*maxn];
int N, A, B;
int head[maxn], ecnt;
void add(int u,int v) {
    edges[ecnt] = Edge(v, head[u]), head[u] = ecnt++;
}
int dfn[maxn], low[maxn], depth;
bool in[maxn];
stack<int> st;
int belong[maxn];
int block;
```

```
92. void tarjan(int u) {
93.     dfn[u] = low[u] = ++depth;
94.     in[u] = true;
95.     st.push(u);
96.
97.     for (int i = head[u]; ~i;i = edges[i].nxt) {
98.         int v = edges[i].to;
99.         if (dfn[v] == -1) {
100.            tarjan(v);
101.            low[u] = min(low[u], low[v]);
102.        }else if (in[v]) low[u] = min(low[u], dfn[v]);
103.    }
104.
105.    if (dfn[u] == low[u]) {
106.        block++;
107.        while(true) {
108.            int x = st.top();
109.            st.pop();
110.            in[x] = false;
111.            belong[x] = block;
112.            if (x == u) break;
113.        }
114.    }
115. }
116. struct point {
117.    int x, y;
118.    void read() {
119.        scanf("%d%d", &x, &y);
120.    }
121. }p[maxn], hate[maxn], like[maxn], s1, s2;
122. int dis[maxn][maxn];
123. inline int dist(point& a,point& b) {//曼哈顿距离
124.    return abs(a.x - b.x) + abs(a.y - b.y);
125. }
126. bool ok() {
127.    depth = block = 0;
128.    memset(dfn, -1,sizeof dfn);
129.    for (int i = 1;i <= 2*N;++i)
130.        if (dfn[i] == -1) tarjan(i);
131.    for (int i = 1;i <= N;++i)
132.        if (belong[i] == belong[i+N]) return false;
133.    return true;
134. }
135.
136. void getmap(int limit) {
137.    ecnt = 0;
138.    memset(head, -1,sizeof head);
139.    for (int i = 1;i <= A;++i) {
140.        add(hate[i].x, hate[i].y + N);
141.        add(hate[i].y, hate[i].x + N);
142.
143.        add(hate[i].x + N, hate[i].y);
144.        add(hate[i].y + N, hate[i].x);
145.    }
146.
147.    for (int i = 1;i <= B;++i) {
148.        add(like[i].x, like[i].y);
149.        add(like[i].y, like[i].x);
150.
151.        add(like[i].x + N, like[i].y + N);
152.        add(like[i].y + N, like[i].x + N);
153.    }
154.
155.    for (int i = 1;i <= N;++i) {
156.        for (int j = i + 1;j <= N;++j) {
157.            if (dis[i][N+1] + dis[j][N+1] > limit) {//不能一同食堂
158.                add(i, j + N);
159.                add(j, i + N);
160.            }
161.            if (dis[i][N+2] + dis[j][N+2] > limit) {
162.                add(i + N, j);
163.                add(j + N, i);
164.            }
165.            if (dis[i][N+1] + dis[j][N+2] + dis[N+1][N+2] > limit) {
166.                add(i, j);
167.                add(j+N, i+N);
168.            }
169.            if (dis[i][N+2] + dis[j][N+1] + dis[N+1][N+2] > limit) {
170.                add(i+N,j+N);
171.                add(j, i);
172.            }
173.        }
174.    }
175. }
176. int main(int argc, const char * argv[])
177. {
178.    // freopen("in.txt","r",stdin);
179.    // freopen("out.txt","w",stdout);
180.    // clock_t _ = clock();
181.
182.    int T_T;
183.    scanf("%d", &T_T);
184.    while(T_T--) {
185.        scanf("%d%d%d", &N, &A, &B);
186.        s1.read(), s2.read();
187.        dis[N+1][N+2] = dist(s1, s2);
188.        for (int i = 1;i <= N;++i)
```

```
189.         {
190.             p[i].read();
191.             dis[i][N+1] = dist(p[i], s1);
192.             dis[i][N+2] = dist(p[i], s2);
193.         }
194.         for (int i = 1;i <= A;++i)
195.             hate[i].read();
196.         for (int i = 1;i <= B;++i)
197.             like[i].read();
198.         int low = 0, high = 50000000;
199.         int ans = -1;
200.         while(low <= high) {
201.             int mid = (low + high) / 2;
202.             getmap(mid);
203.             if (ok()) {
204.                 ans = mid;
205.                 high = mid - 1;
206.             }else low = mid + 1;
207.         }
208.         printf("%d\n", ans);
209.     }
210.
211.     // printf("\nTime cost: %.2fs\n", 1.0 * (clock() - _) / CLOCKS_PER_SEC);
212.     return 0;
213. }
```

E.Division

题意

略

分析

套路，缩点+匹配

```
 1. const int N = 5000 + 10;
 2. const int M = 100000 + 10;
 3. int n, m;
 4. int head[N], pnt[M], nxt[M], cnt;
 5. void init() {
 6.     memset(head, -1,sizeof head);
 7.     cnt = 0;
 8. }
 9. void addedge(int u,int v) {
10.     pnt[cnt] = v, nxt[cnt] = head[u],head[u] = cnt++;
11. }
12. int dfn[N], low[N], belong[N];
13. int Times;
14. stack<int> st;
15. int scc;
16. void Tarjan(int u) {
17.     dfn[u] = low[u] = ++Times;
18.     st.push(u);
19.     for (int i = head[u];~i;i = nxt[i]) {
20.         int v = pnt[i];
21.         if (!dfn[v]) {
22.             Tarjan(v);
23.             low[u] = min(low[u], low[v]);
24.         }else if (!belong[v]) low[u] = min(low[u], dfn[v]);
25.     }
26.     if (dfn[u] == low[u]) {
27.         ++scc;
28.         while(true) {
29.             int x = st.top();
30.             st.pop();
31.             belong[x] = scc;
32.             if (x == u) break;
33.         }
34.     }
35. }
36. vector<int> G[N];
37.
38.
39. int linker[N];
40. bool vis[N];
41.
42. bool dfs(int u) {
43.     for (int i = 0;i < G[u].size();++i) {
44.         int v = G[u][i];
45.         if (vis[v]) continue;
46.         vis[v] = true;
47.         if (linker[v] == -1 || dfs(linker[v])) {
48.             linker[v] = u;
49.             return true;
50.         }
51.     }
52.     return false;
53. }
54.
55. int Hungary(){
56.     int ans = 0;
57.     memset(linker, -1,sizeof linker);
```

```
    for (int i = 1;i <= scc;++i) {
        memset(vis, false,sizeof vis);
        if (dfs(i)) ans++;
    }
    return ans;
}

int main()
{
    // freopen("in.txt","r",stdin);
    // freopen("out.txt","w",stdout);
    int t;
    scanf("%d",&t);
    while(t--) {
        scanf("%d%d",&n,&m);
        init();
        int u, v;
        for (int i = 1;i <= m;++i) {
            scanf("%d%d",&u,&v);
            addedge(u, v);
        }
        memset(dfn, 0,sizeof dfn);
        memset(belong, 0,sizeof belong);
        scc = Times = 0;
        for (int i = 1;i <= n;++i) G[i].clear();
        for (int i = 1;i <= n;++i) if (!dfn[i]) Tarjan(i);
        for (int i = 1;i <= n;++i) {
            for (int j = head[i];~j;j = nxt[j]) {
                int v = pnt[j];
                if (belong[i] != belong[v]) {
                    G[belong[i]].push_back(belong[v]);
                }
            }
        }
        printf("%d\n", scc - Hungary());
    }
    return 0;
}
```

F.meixiuxiu学图论

题意

求所有环中的最大边权的最小值。

分析

可以两种做法。
1.二分结果然后scc.
2.最小生成树的应用.
说第二种吧。。。把边排序，然后合并第一个形成的环的最后一天边就是答案。

```
const int maxn = 5e5 + 10;
const int maxm = 2e6 + 10;
struct Edge {
    int u, v, c;
    Edge() {}
    Edge(int u,int v,int c) {
        this->u = u;
        this->v = v;
        this->c = c;
    }
    bool operator < (const Edge& rhs) const {
        return c < rhs.c;
    }
    void read() {
        scanf("%d%d%d", &u, &v, &c);
    }
}e[maxm];
int f[maxn];
int find(int x) {
    return f[x] == -1?x : f[x] = find(f[x]);
}
int main(int argc, const char * argv[])
{
    // freopen("in.txt","r",stdin);
    // freopen("out.txt","w",stdout);
    // clock_t _ = clock();

    int n , m;
    int t;
    scanf("%d", &t);
    while(t--) {
        scanf("%d%d", &n, &m);
        for (int i = 0;i < m;++i)
            e[i].read();
        sort(e,e+m);
        memset(f, -1,sizeof f);
        int ans = -1;
        for (int i = 0;i < m;++i) {
            int t1 = find(e[i].u);
            int t2 = find(e[i].v);
```

```
            if (t1 != t2) {
                f[t1] = t2;
            }else if (ans == -1) ans = e[i].c;
        }
        if (ans == -1) puts("No solution!");
        else printf("%d\n", ans);
    }

    // printf("\nTime cost: %.2fs\n", 1.0 * (clock() - _) / CLOCKS_PER_SEC);
    return 0;
}
```

G.最短路

题意

略

分析

问的是最短路的条数，且任意两条路不重复。首先每条路径都是最短的，所以就建立最短路树。然后就是最大流算法了。并附上最大流模版。。。

```
const int maxn = 1010;
vector<ii> G[2][maxn];//v, cost
int n, m;
int dis1[maxn], dis2[maxn], in[maxn];
void spfa(int s,int t,int* d, int o) {
    for (int i = 1;i <= n;++i)
        d[i] = INF;
    memset(in, 0,sizeof in);
    d[s] = 0;
    queue<int> que;
    que.push(s);
    while(!que.empty()) {
        int u = que.front();
        que.pop();
        in[u] = 0;
        for (int i = 0;i < G[o][u].size();++i) {
            int v = G[o][u][i].first;
            int cost = G[o][u][i].second;
            if (d[v] > d[u] + cost) {
                d[v] = d[u] + cost;
                if (in[v] == 0) {
                    in[v] = 1;
                    que.push(v);
                }
            }
        }
    }
}
struct Edge{
    int from, to, cap, flow;
    Edge(){}
    Edge(int from,int to,int cap,int flow):from(from),to(to),cap(cap),flow(flow){}
};
struct ISAP{
    int p[maxn], num[maxn], cur[maxn], d[maxn];
    int s, t, n, m;
    bool vis[maxn];

    vector<int> G[maxn];
    vector<Edge> edges;

    void init(int n) {
        this->n = n;
        for (int i = 0;i <= n;++i) {
            G[i].clear();
            d[i] = INF;
        }
        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 = (int)edges.size();
        G[from].push_back(m - 2);
        G[to].push_back(m - 1);
    }

    bool bfs() {
        memset(vis, false,sizeof vis);

        queue<int> que;
        d[t] = 0;
        vis[t] = true;
        que.push(t);

        while(!que.empty()) {
            int u = que.front();
            que.pop();

            for (int i = 0;i < G[u].size();++i) {
```

```
                Edge& e = edges[G[u][i]^1];
                if (e.cap > e.flow && !vis[e.from]) {
                    vis[e.from] = true;
                    d[e.from] = d[u] + 1;
                    que.push(e.from);
                }
            }
        }
        return vis[s];
    }

    int Augment() {
        int u = t, flow = INF;
        while(u != s) {
            Edge& e = edges[p[u]];
            flow = min(flow, e.cap - e.flow);
            u = edges[p[u]].from;
        }

        u = t;
        while(u != s) {
            edges[p[u]].flow += flow;
            edges[p[u]^1].flow -= flow;
            u = edges[p[u]].from;
        }
        return flow;
    }

    int MaxFlow(int s,int t) {
        this->s = s,this->t = t;
        int ret = 0;
        bfs();
        if (d[s] >= n) return 0;

        memset(num, 0,sizeof num);
        memset(cur, 0,sizeof cur);
        for (int i = 0;i < n;++i) {
            if (d[i] < INF) num[d[i]]++;
        }
        int u = s;

        while(d[s] < n) {

            if (u == t) {
                ret += Augment();
                u = s;
            }

            bool ok = false;
            for (int i = cur[u];i < G[u].size();++i) {
                Edge& e = edges[G[u][i]];
                if (e.cap > e.flow && d[u] == d[e.to] + 1) {
                    ok = true;
                    p[e.to] = G[u][i];
                    cur[u] = i;
                    u = e.to;
                    break;
                }
            }

            if (!ok) {
                int Min = n - 1;
                for (int i = 0;i < G[u].size();++i) {
                    Edge& e = edges[G[u][i]];
                    if (e.cap > e.flow) Min = min(Min, d[e.to]);
                }
                if (--num[d[u]] == 0) break;
                num[d[u] = Min + 1]++;
                cur[u] = 0;
                if (u != s) u = edges[p[u]].from;
            }
        }
        return ret;
    }
}solve;
int main(int argc, const char * argv[])
{
    // freopen("in.txt","r",stdin);
    // freopen("out.txt","w",stdout);
    // clock_t _ = clock();

    int t;
    cin >> t;
    while(t--) {
        scanf("%d%d", &n, &m);
        for (int i = 1;i <= n;++i)
            G[0][i].clear(), G[1][i].clear();
        int u, v, c;
        for (int i = 1;i <= m;++i) {
            scanf("%d%d%d", &u, &v, &c);
            G[0][u].push_back(ii(v, c));
            G[1][v].push_back(ii(u, c));
        }
        int st, ed;
        scanf("%d%d", &st, &ed);
        spfa(st, ed, dis1, 0);
        spfa(ed, st, dis2, 1);
```

```
        int limit = dis1[ed];
        if (limit == INF) {
            printf("%d\n", 0);
            continue;
        }

        solve.init(n);

        for (int i = 1;i <= n;++i) {
            for (int j = 0;j < G[0][i].size();++j) {
                int k = G[0][i][j].first;
                if (dis1[i] != INF && dis2[k] != INF && dis1[i] + dis2[k] + G[0][i][j].second == limit) {
                    solve.addedge(i, k, 1);
                }
            }
        }
        int ans = solve.MaxFlow(st, ed);
        printf("%d\n", ans);
    }

    // printf("\nTime cost: %.2fs\n", 1.0 * (clock() - _) / CLOCKS_PER_SEC);
    return 0;
}
```

H.NightMare2

题意

从1号点到n号点在k的时间内逃出去的前提下，能带走的最大价值珠宝。

分析

又是一种套路。二分答案，然后跑最短路，看能不能逃出去。

```
/*****************************************
Author      :Crazy_AC(JamesQi)
Time        :2016
File Name   :
                     _ooOoo_
                    o8888888o
                    88" . "88
                    (| -_- |)
                    O\  =  /O
                 ____/`---'\____
               .'  \|     |//  `.
              /  \|||  :  |||//  \
             /  _||||| -:- |||||-  \
             |   | \\  -  /// |   |
             | \_|  ''\---/''  |   |
             \  .-\__  `-`  ___/-. /
           ___`. .'  /--.--\  `. . __
        ."" '<  `.___\_<|>_/___.'  >'"".
       | | :  `- \`.;`\ _ /`;.`/ - ` : | |
       \  \ `-.   \_ __\ /__ _/   .-` /  /
======`-.____`-.___\_____/___.-`____.-'======
                   `=---='
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
           佛祖保佑       永无BUG
*****************************************/
// #pragma comment(linker, "/STACK:1024000000,1024000000")
#include <iostream>
#include <algorithm>
#include <iomanip>
#include <sstream>
#include <string>
#include <stack>
#include <queue>
#include <deque>
#include <vector>
#include <map>
#include <set>
#include <cstdio>
#include <cstring>
#include <cmath>
#include <cstdlib>
#include <climits>
using namespace std;
#define MEM(x,y) memset(x, y,sizeof x)
#define pk push_back
#define lson rt << 1
#define rson rt << 1 | 1
#define bug cout << "BUG HERE\n"
typedef long long LL;
typedef unsigned long long ULL;
typedef pair<int,int> ii;
typedef pair<ii,int> iii;
const double eps = 1e-8;
const double pi = 4 * atan(1);
const long long inf = 1e20 + 10;
const int INF = 0x3f3f3f3f;
const int MOD = 1e9 + 7;
int nCase = 0;
int dcmp(double x){//精度正负、0的判断
```

```
    if (fabs(x) < eps) return 0;
    return x < 0?-1:1;
}
inline int read(){
    char c = getchar();
    while (!isdigit(c)) c = getchar();
    int x = 0;
    while (isdigit(c)) {
        x = x * 10 + c - '0';
        c = getchar();
    }
    return x;
}
const int maxn = 1e4 + 10;
struct Edge {
    int v, limit, cost;
    Edge() {}
    Edge(int v,int limit, int cost) {
        this->v = v;
        this->limit = limit;
        this->cost = cost;
    }
};
vector<Edge> G[maxn];
int n, m, k;
struct node {
    int p;
    long long cost;
    node() {}
    node(int p,long long cost) {
        this->p = p;
        this->cost = cost;
    }
    bool operator < (const node& rhs) const {//小的优先
        return cost > rhs.cost;
    }
};
long long dis[maxn];
int vis[maxn];
bool solve(int limit) {
    // memset(dis, INF,sizeof dis);
    for (int i = 1;i <= n;++i)
        dis[i] = inf;
    memset(vis, 0,sizeof vis);
    dis[1] = 0;
    priority_queue<node> que;
    que.push(node(1, 0));
    while(!que.empty()) {
        node temp = que.top();
        que.pop();
        int u = temp.p;
        if (vis[u]) continue;
        vis[u] = 1;
        for (int i = 0;i < G[u].size();++i) {
            if (G[u][i].limit < limit) continue;
            int v = G[u][i].v;
            long long cost = G[u][i].cost;
            if (dis[v] > dis[u] + cost) {
                dis[v] = dis[u] + cost;
                que.push(node(v, dis[v]));
            }
        }
    }
    return dis[n] <= k;
}


int main(int argc, const char * argv[])
{
    freopen("in.txt","r",stdin);
    // freopen("out.txt","w",stdout);
    // clock_t _ = clock();

    int t;
    scanf("%d",&t);
    while(t--) {
        scanf("%d%d%d", &n, &m, &k);
        for (int i = 1;i <= n;++i)
            G[i].clear();
        int u, v, l, c;
        int high = 0, low = INF;
        for (int i = 1;i <= m;++i) {
            scanf("%d%d%d%d", &u, &v, &l, &c);
            G[u].push_back(Edge(v, l, c));
            G[v].push_back(Edge(u, l, c));
            high = max(high, l);
            low = min(low, l);
        }
        int ans = -1;
        while(low <= high) {
            int mid = (high + low) >> 1;
            if (solve(mid)) {
                low = mid + 1;
                ans = mid;
            }else high = mid - 1;
        }
        if (solve(low)) ans = max(ans, low);
```

```
        if (solve(high)) ans = max(ans, high);
        if (ans == -1) puts("Poor RunningPhoton!");
        else printf("%d\n", ans);
    }

    // printf("\nTime cost: %.2fs\n", 1.0 * (clock() - _) / CLOCKS_PER_SEC);
    return 0;
}
```

I.玛雅，好简单

题意

略

分析

求无向图桥边的模版题。。。。

```
/*****************************************
Author      :Crazy_AC(JamesQi)
Time        :2016
File Name   :
                      _ooOoo_
                     o8888888o
                     88" . "88
                     (| -_- |)
                     O\  =  /O
                  ____/`---'\____
                .'  \|     |//  `.
               /  \|||  :  |||//  \
              /  _||||| -:- |||||-  \
              |   | \\  -  /// |   |
              | \_|  ''\---/''  |   |
              \  .-\__  `-`  ___/-. /
            ___`. .'  /--.--\  `. . __
         ."" '<  `.___\_<|>_/___.'  >'"".
        | | :  `- \`.;`\ _ /`;.`/ - ` : | |
        \  \ `-.   \_ __\ /__ _/   .-` /  /
======`-.____`-.___\_____/___.-`____.-'======
                   `=---='
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
          佛祖保佑       永无BUG
*****************************************/
// #pragma comment(linker, "/STACK:1024000000,1024000000")
#include <iostream>
#include <algorithm>
#include <iomanip>
#include <sstream>
#include <string>
#include <stack>
#include <queue>
#include <deque>
#include <vector>
#include <map>
#include <set>
#include <cstdio>
#include <cstring>
#include <cmath>
#include <cstdlib>
#include <climits>
using namespace std;
#define MEM(x,y) memset(x, y,sizeof x)
#define pk push_back
#define lson rt << 1
#define rson rt << 1 | 1
#define bug cout << "BUG HERE\n"
typedef long long LL;
typedef unsigned long long ULL;
typedef pair<int,int> ii;
typedef pair<ii,int> iii;
const double eps = 1e-8;
const double pi = 4 * atan(1);
const int inf = 1 << 30;
const int INF = 0x3f3f3f3f;
const int MOD = 1e9 + 7;
int nCase = 0;
int dcmp(double x){//精度正负、0的判断
    if (fabs(x) < eps) return 0;
    return x < 0?-1:1;
}
inline int read(){
    char c = getchar();
    while (!isdigit(c)) c = getchar();
    int x = 0;
    while (isdigit(c)) {
        x = x * 10 + c - '0';
        c = getchar();
    }
    return x;
}
const int maxn = 10010;
vector<int> G[maxn];
```

```
int dfn[maxn], low[maxn], depth;
bool in[maxn];
int cnt;
stack<int> st;
void dfs(int u,int fa) {
    dfn[u] = low[u] = ++depth;
    st.push(u);
    in[u] = true;
    int first = 1;
    for (int i = 0;i < G[u].size();++i) {
        int v = G[u][i];
        if (first && v == fa) {
            first = 0;
            continue;
        }
        if (dfn[v] == -1) {
            dfs(v, u);
            if (low[v] > dfn[u]) cnt++;
            low[u] = min(low[u], low[v]);
        }else if (in[v]) low[u] = min(low[u], dfn[v]);
    }
    if (dfn[u] == low[u]) {
        while(true) {
            int x = st.top();
            st.pop();
            in[x] = false;
            if (x == u) break;
        }
    }
}
int n, m;
void solve() {
    memset(dfn, -1,sizeof dfn);
    cnt = depth = 0;
    for (int i  =1;i <= n;++i)
        if (dfn[i] == -1) dfs(i, -1);
    printf("Case %d: %d\n", ++nCase, cnt);
}
int main(int argc, const char * argv[])
{
    // freopen("in.txt","r",stdin);
    // freopen("out.txt","w",stdout);
    // clock_t _ = clock();

    int t;
    scanf("%d", &t);
    while(t--) {
        scanf("%d%d", &n, &m);
        for (int i = 1;i <= n;++i)
            G[i].clear();
        int u, v;
        for (int i = 1;i <= m;++i) {
            scanf("%d%d", &u, &v);
            G[u].push_back(v);
            G[v].push_back(u);
        }
        solve();
    }

    // printf("\nTime cost: %.2fs\n", 1.0 * (clock() - _) / CLOCKS_PER_SEC);
    return 0;
}
```

J.An Easy Problem

题意

就是选最少的人进行路径覆盖

分析

先闭包传递，然后二分图匹配最小路径覆盖。

```
const int N = 1010;
vector<int> G1[N], G2[N], G3[N];//原图，扩张图，缩点后的图。
bool vis[N];
int pre[N], low[N], Belong[N], scc_cnt, Times;
int n, m;
stack<int> st;
void Tarjan(int u) {
    pre[u] = low[u] = ++Times;
    st.push(u);
    for (int i = 0;i < (int)G2[u].size();++i) {
        int v = G2[u][i];
        if (!pre[v]) {
            Tarjan(v);
            low[u] = min(low[u], low[v]);
        }else if (!Belong[v]) low[u] = min(low[u], pre[v]);
    }
    if (pre[u] == low[u]) {
        scc_cnt++;
        while(true) {
            int x = st.top();
```

```
            st.pop();
            Belong[x] = scc_cnt;
            if (x == u) break;
        }
    }
}
void FindSCC() {
    memset(pre, 0,sizeof pre);
    memset(Belong, 0,sizeof Belong);
    Times = scc_cnt = 0;
    for (int i = 1;i <= n;++i) {
        if (!pre[i]) Tarjan(i);
    }
}
void BFS(int st) {
    queue<int> que;
    que.push(st);
    memset(vis, false,sizeof vis);
    vis[st] = true;
    while(!que.empty()) {
        int u = que.front();
        que.pop();
        for (int i = 0;i < G1[u].size();++i) {
            int v = G1[u][i];
            if (vis[v]) continue;
            vis[v] = true;
            G2[st].push_back(v);//对原图进行这种扩展是不会形成可行环的。
            que.push(v);
        }
    }
}
void Initation() {
    for (int i = 1;i <= n;++i)
        BFS(i);
}
void Input() {
    scanf("%d%d",&n,&m);
    for (int i = 1;i <= n;++i)
        G1[i].clear(),G2[i].clear(),G3[i].clear();
    int u, v;
    for (int i = 1;i <= m;++i) {
        scanf("%d%d",&u,&v);
        G1[u].push_back(v);
    }
}
void Trans() {
    for (int u = 1;u <= n;++u) {
        for (int i = 0;i < G2[u].size();++i) {
            int v = G2[u][i];
            if (Belong[u] != Belong[v])
                G3[Belong[u]].push_back(Belong[v]);
        }
    }
}
int linker[N];
bool Search(int u) {
    for (int i = 0;i < G3[u].size();++i) {
        int v = G3[u][i];
        if (vis[v]) continue;
        vis[v] = true;
        if (linker[v] == -1 || Search(linker[v])) {
            linker[v] = u;
            return true;
        }
    }
    return false;
}
int Hungary() {
    int ret = 0;
    memset(linker, -1,sizeof linker);
    for (int i = 1;i <= scc_cnt;++i) {
        memset(vis, false,sizeof vis);
        if (Search(i)) ret++;
    }
    return ret;
}
int main()
{
    // freopen("in.txt","r",stdin);
    // freopen("out.txt","w",stdout);
    int t, icase = 0;
    scanf("%d",&t);
    while(t--) {
        Input();
        Initation();
        FindSCC();
        // printf("SCC = %d\n", scc_cnt);
        // for (int i = 1;i <= n;++i)
            // printf("%d ", Belong[i]);
        // puts("");
        Trans();

        // for (int i = 1;i <= n;++i) {
            // printf("<%d>::", i);
            // for (int j = 0;j < G2[i].size();++j)
                // printf("%d ", G2[i][j]);
            // puts("");
```

```
118.         // }
119.
120.         // for (int i = 1;i <= scc_cnt;++i) {
121.         //  printf("i = %d:::", i);
122.         //  for (int j = 0;j < G3[i].size();++j)
123.         //      printf("%d ",G3[i][j]);
124.         //  puts("");
125.         // }
126.         printf("Case %d: %d\n", ++icase, scc_cnt - Hungary());
127.     }
128.     return 0;
129. }
```

K.投票

题意

投票是单向的且具有传递性，求获得票数最多的人。。。

分析

首先一个连通分量里面的人获得的票数肯定是一样的，然后缩点成DAG题，反向建边，再从入读为0的点开始搜索。

```
  1. /*****************************************
  2. Author      :Crazy_AC(JamesQi)
  3. Time        :2016
  4. File Name   :
  5.                       _ooOoo_
  6.                      o8888888o
  7.                      88" . "88
  8.                      (| -_- |)
  9.                      O\  =  /O
 10.                   ____/`---'\____
 11.                 .'  \|     |//  `.
 12.                /  \|||  :  |||//  \
 13.               /  _|||||| -:- ||||||-  \
 14.               |   | \\  -  /// |   |
 15.               | \_|  ''\---/''  |   |
 16.               \  .-\__  `-`  ___/-. /
 17.             ___`. .'  /--.--\  `. . __
 18.          ."" '<  `.___\_<|>_/___.'  >'"".
 19.         | | :  `- \`.;`\ _ /`;.`/ - ` : | |
 20.         \  \ `-.   \_ __\ /__ _/   .-` /  /
 21. ======`-.____`-.___\_____/___.-`____.-'======
 22.                    `=----='
 23. ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
 24.           佛祖保佑       永无BUG
 25. *****************************************/
 26. // #pragma comment(linker, "/STACK:1024000000,1024000000")
 27. #include <iostream>
 28. #include <algorithm>
 29. #include <iomanip>
 30. #include <sstream>
 31. #include <string>
 32. #include <stack>
 33. #include <queue>
 34. #include <deque>
 35. #include <vector>
 36. #include <map>
 37. #include <set>
 38. #include <cstdio>
 39. #include <cstring>
 40. #include <cmath>
 41. #include <cstdlib>
 42. #include <climits>
 43. using namespace std;
 44. #define MEM(x,y) memset(x, y,sizeof x)
 45. #define pk push_back
 46. #define lson rt << 1
 47. #define rson rt << 1 | 1
 48. #define bug cout << "BUG HERE\n"
 49. typedef long long LL;
 50. typedef unsigned long long ULL;
 51. typedef pair<int,int> ii;
 52. typedef pair<ii,int> iii;
 53. const double eps = 1e-8;
 54. const double pi = 4 * atan(1);
 55. const int inf = 1 << 30;
 56. const int INF = 0x3f3f3f3f;
 57. const int MOD = 1e9 + 7;
 58. int nCase = 0;
 59. int dcmp(double x){//精度正负、0的判断
 60.     if (fabs(x) < eps) return 0;
 61.     return x < 0?-1:1;
 62. }
 63. inline int read(){
 64.     char c = getchar();
 65.     while (!isdigit(c)) c = getchar();
 66.     int x = 0;
 67.     while (isdigit(c)) {
 68.         x = x * 10 + c - '0';
 69.         c = getchar();
 70.     }
```

```
    return x;
}
const int maxn = 5010;
const int maxm = 30010;
int head[maxn], pnt[maxm], nxt[maxm], ecnt;
int n, m;
int dfn[maxn], low[maxn], in[maxn], depth;
int belong[maxn], block;
int cnt1[maxn],cnt2[maxn], cnt3[maxn];
stack<int> st;
void dfs(int u) {
    dfn[u] = low[u] = ++depth;
    in[u] = 1;
    st.push(u);
    for (int i = head[u]; ~i; i = nxt[i]) {
        int v = pnt[i];
        if (dfn[v] == -1) {
            dfs(v);
            low[u] = min(low[u], low[v]);
        }else if (in[v]) low[u] = min(low[u], dfn[v]);
    }
    if (dfn[u] == low[u]) {
        block++;
        while(true) {
            int x = st.top();
            st.pop();
            in[x] = 0;
            belong[x] = block;
            cnt1[block]++;
            if (x == u) break;
            // cnt1[block]++;
        }
    }
}
void find_scc() {
    memset(dfn, -1,sizeof dfn);
    memset(in, 0,sizeof in);
    memset(cnt1, 0,sizeof cnt1);//scc
    depth = block = 0;
    for (int i = 0;i < n;++i)
        if (dfn[i] == -1) dfs(i);
}
vector<int> G[maxn];
int mark[maxn];
int search(int u) {
    int sum = 0;
    for (int i = 0;i < G[u].size();++i) {
        int v = G[u][i];
        if (mark[v]) continue;
        mark[v] = 1;
        sum += cnt1[v];
        sum += search(v);
    }
    return sum;
}
void solve() {
    for (int i = 1;i <= block;++i)
        G[i].clear();
//rebuild new graph
    for (int u = 0;u < n;++u) {
        for (int i = head[u]; ~i;i = nxt[i]) {
            int v = pnt[i];
            if (belong[u] != belong[v]) {
                G[belong[v]].push_back(belong[u]);
                in[belong[u]]++;
            }
        }
    }
    memset(cnt2, 0,sizeof cnt2);
    for (int i = 1;i <= block;++i) {
        if (in[i] == 0) {
            memset(mark, 0,sizeof mark);
            cnt2[i] = search(i);
        }
    }
    int Max = 0;
    for (int i = 0;i < n;++i) {
        cnt3[i] = cnt1[belong[i]] + cnt2[belong[i]] - 1;
        // cout << "cnt3 = " << cnt3[i] << endl;
        Max = max(Max, cnt3[i]);
    }
    printf("Case %d: %d\n", ++nCase, Max);
    int first = 1;
    for (int i = 0;i < n;++i) {
        if (cnt3[i] == Max) {
            if (first) first = 0;
            else printf(" ");
            printf("%d", i);
        }
    }
    printf("\n");
}
int main(int argc, const char * argv[])
{
    // freopen("in.txt","r",stdin);
    // freopen("out.txt","w",stdout);
    // clock_t _ = clock();
```

```
168.
169.      int t;
170.      cin >> t;
171.      while(t--) {
172.          scanf("%d %d", &n, &m);
173.          memset(head, -1,sizeof head), ecnt = 0;
174.          int u, v;
175.          for (int i = 1;i <= m;++i) {
176.              scanf("%d %d", &u, &v);
177.              pnt[ecnt] = v, nxt[ecnt] = head[u], head[u] = ecnt++;
178.          }
179.          find_scc();
180.          solve();
181.      }
182.
183.      // printf("\nTime cost: %.2fs\n", 1.0 * (clock() - _) / CLOCKS_PER_SEC);
184.      return 0;
185.  }
```