

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）。

```
1.  int e[500*500];
2.  int in[510], out[510]; // indegree, outdegree
3.  int f[510]; // 判断是否连通
4.  int find(int x) {
5.      return f[x] == -1 ? x : f[x] = find(f[x]);
6.  }
7.  int main(int argc, const char * argv[])
8.  {
9.      // freopen("in.txt", "r", stdin);
10.     // freopen("out.txt", "w", stdout);
11.     // clock_t _ = clock();
12.
13.     int t, n, m;
14.     scanf("%d", &t);
15.     while(t--) {
16.         scanf("%d%d", &n, &m);
17.         for (int i = 0; i < m; ++i)
18.             scanf("%d%d", &e[i].first, &e[i].second);
19.         memset(in, 0, sizeof in);
20.         memset(out, 0, sizeof out);
21.         memset(f, -1, sizeof f);
22.         int cnt = 0;
23.         for (int i = 0; i < m; ++i) {
24.             in[e[i].first]++;
25.             in[e[i].second]++;
26.             int t1 = find(e[i].first);
27.             int t2 = find(e[i].second);
28.             if (t1 != t2) f[t1] = t2;
29.         }
30.         int o = 0;
31.         for (int i = 1; i <= n; ++i) {
32.             find(i);
33.             if (f[i] == -1) o++;
34.         } // o == 说明图连通
35.         for (int i = 1; i <= n; ++i)
36.             if (in[i] & 1) cnt++;
37.         if (cnt == 0 || cnt == 2) {
38.             if (o == 1) printf("Yes");
39.             else printf("No");
40.         } else printf("No");
41.         printf(" ");
42.
43.         memset(in, 0, sizeof in);
44.         memset(f, -1, sizeof f);
45.         cnt = 0;
46.         for (int i = 0; i < m; ++i) {
47.             out[e[i].first]++;
48.             in[e[i].second]++;
49.         }
50.         if (o > 1) puts("No");
51.         else {
52.             vector<int> vec;
53.             for (int i = 1; i <= n; ++i) {
54.                 if (in[i] != out[i]) vec.push_back(i);
55.             }
56.             if (vec.size() != 2 && vec.size() != 0) puts("No");
57.             else {
58.                 if (vec.size() == 0) {
59.                     puts("Yes");
60.                     continue;
61.                 }
62.                 int u = vec[0], v = vec[1];
63.                 if (in[u] - out[u] == 1 && in[v] - out[v] == -1) puts("Yes");
64.                 else if (in[u] - out[u] == -1 && in[v] - out[v] == 1) puts("Yes");
65.                 else puts("No");
66.             }
67.         }
68.     }
69. }
70. // 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)
```

```
80.         scanf("%d", &a[i][j]);
81.     }
82.     solve();
83. }
84.
85. // printf("\nTime cost: %.2fs\n", 1.0 * (clock() - _) / CLOCKS_PER_SEC);
86. return 0;
87. }
```

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。

```
1. const int maxn = 5e2 + 10;
2. int n, m;
3. int col[maxn];
4. vector<int> G[maxn];
5. int g[maxn][maxn];
6. bool dfs(int u,int color) {
7.     col[u] = color;
8.     for (int i = 0;i < G[u].size();++i) {
9.         int v = G[u][i];
10.        if (col[u] == col[v]) return false;
11.        if (!col[v] && !dfs(v, -color)) return false;
12.    }
13.    return true;
14. }
15. int main(int argc, const char * argv[])
16. {
17.     // freopen("in.txt","r",stdin);
18.     // freopen("out.txt","w",stdout);
19.     // clock_t _ = clock();
20.
21.     int t;
22.     cin >> t;
23.     while(t--){
24.         scanf("%d%d", &n, &m);
25.         memset(g, 0,sizeof g);
26.         int u, v;
27.         for (int i = 1;i <= m;++i) {
28.             scanf("%d%d", &u, &v);
29.             g[u][v] = g[v][u] = 1;
30.         }
31.         for (int i = 1;i <= n;++i) {
32.             G[i].clear();
33.             for (int j = 1;j <= n;++j)
34.                 if (i != j && !g[i][j]) G[i].push_back(j);
35.         }
36.         bool yes = true;
37.         memset(col, 0,sizeof col);
38.         for (int i = 1;i <= n;++i) {
39.             if (!col[i] && G[i].size() && !dfs(i, 1)) {
40.                 yes=false;
41.             }
42.         }
43.         for (int i = 1;i <= n;++i) {
44.             for (int j = 1;j <= n;++j) {
45.                 if (i == j) continue;
46.                 if (g[i][j] && col[i]*col[j]<0)yes=false;
47.                 if (!g[i][j] && col[i]*col[j]>=0)yes=false;
48.             }
49.         }
50.         if (yes) puts("Yes");
51.         else puts("No");
52.     }
53.
54.     // printf("\nTime cost: %.2fs\n", 1.0 * (clock() - _) / CLOCKS_PER_SEC);
55.     return 0;
56. }
```

D.一食堂or二食堂，it's a question

题意

看懂，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。。。

```

1.  /*****
2.  Author      :Crazy_AC(JamesQi)
3.  Time       :2016
4.  File Name  :
5.
6.              _oo0oo_
7.              o8888888o
8.              88" . "88
9.              (| -_- |)
10.             O\ = /O
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.     }
71.     return x;
72. }
73. const int maxn = 2010;
74. struct Edge{
75.     int to, nxt;
76.     Edge() {}
77.     Edge(int to, int nxt) {
78.         this->to = to;
79.         this->nxt = nxt;
80.     }
81. }edges[maxn*maxn];
82. int N, A, B;
83. int head[maxn], ecnt;
84. void add(int u, int v) {
85.     edges[ecnt] = Edge(v, head[u]), head[u] = ecnt++;
86. }
87. int dfn[maxn], low[maxn], depth;
88. bool in[maxn];
89. stack<int> st;
90. int belong[maxn];
91. 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;
183.     scanf("%d", &T);
184.     while(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. int linker[N];
39. bool vis[N];
40.
41. bool dfs(int u) {
42.    for (int i = 0; i < G[u].size(); ++i) {
43.        int v = G[u][i];
44.        if (vis[v]) continue;
45.        vis[v] = true;
46.        if (linker[v] == -1 || dfs(linker[v])) {
47.            linker[v] = u;
48.            return true;
49.        }
50.    }
51.    return false;
52. }
53.
54. int Hungary() {
55.    int ans = 0;
56.    memset(linker, -1, sizeof linker);
57. }
```

```
58.     for (int i = 1;i <= scc;++i) {
59.         memset(vis, false,sizeof vis);
60.         if (dfs(i)) ans++;
61.     }
62.     return ans;
63. }
64.
65. int main()
66. {
67.     // freopen("in.txt","r",stdin);
68.     // freopen("out.txt","w",stdout);
69.     int t;
70.     scanf("%d",&t);
71.     while(t--) {
72.         scanf("%d%d",&n,&m);
73.         init();
74.         int u, v;
75.         for (int i = 1;i <= m;++i) {
76.             scanf("%d%d",&u,&v);
77.             addedge(u, v);
78.         }
79.         memset(dfn, 0,sizeof dfn);
80.         memset(belong, 0,sizeof belong);
81.         scc = Times = 0;
82.         for (int i = 1;i <= n;++i) G[i].clear();
83.         for (int i = 1;i <= n;++i) if (!dfn[i]) Tarjan(i);
84.         for (int i = 1;i <= n;++i) {
85.             for (int j = head[i];~j;j = nxt[j]) {
86.                 int v = pnt[j];
87.                 if (belong[i] != belong[v]) {
88.                     G[belong[i]].push_back(belong[v]);
89.                 }
90.             }
91.         }
92.         printf("%d\n", scc - Hungary());
93.     }
94.     return 0;
95. }
```

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。

```
1. const int maxn = 5e5 + 10;
2. const int maxm = 2e6 + 10;
3. struct Edge {
4.     int u, v, c;
5.     Edge() {}
6.     Edge(int u,int v,int c) {
7.         this->u = u;
8.         this->v = v;
9.         this->c = c;
10.    }
11.    bool operator < (const Edge& rhs) const {
12.        return c < rhs.c;
13.    }
14.    void read() {
15.        scanf("%d%d%d", &u, &v, &c);
16.    }
17. }e[maxm];
18. int f[maxn];
19. int find(int x) {
20.     return f[x] == -1?x : f[x] = find(f[x]);
21. }
22. int main(int argc, const char * argv[])
23. {
24.     // freopen("in.txt","r",stdin);
25.     // freopen("out.txt","w",stdout);
26.     // clock_t _ = clock();
27.
28.     int n , m;
29.     int t;
30.     scanf("%d", &t);
31.     while(t--) {
32.         scanf("%d%d", &n, &m);
33.         for (int i = 0;i < m;++i)
34.             e[i].read();
35.         sort(e,e+m);
36.         memset(f, -1,sizeof f);
37.         int ans = -1;
38.         for (int i = 0;i < m;++i) {
39.             int t1 = find(e[i].u);
40.             int t2 = find(e[i].v);
```

```
41.         if (t1 != t2) {
42.             f[t1] = t2;
43.         }else if (ans == -1) ans = e[i].c;
44.     }
45.     if (ans == -1) puts("No solution!");
46.     else printf("%d\n", ans);
47. }
48.
49. // printf("\nTime cost: %.2fs\n", 1.0 * (clock() - _) / CLOCKS_PER_SEC);
50. return 0;
51. }
```

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。。。

```
1. const int maxn = 1010;
2. vector<ii> G[2][maxn]; //v, cost
3. int n, m;
4. int dis1[maxn], dis2[maxn], in[maxn];
5. void spfa(int s, int t, int* d, int o) {
6.     for (int i = 1; i <= n; ++i)
7.         d[i] = INF;
8.     memset(in, 0, sizeof in);
9.     d[s] = 0;
10.    queue<int> que;
11.    que.push(s);
12.    while(!que.empty()) {
13.        int u = que.front();
14.        que.pop();
15.        in[u] = 0;
16.        for (int i = 0; i < G[o][u].size(); ++i) {
17.            int v = G[o][u][i].first;
18.            int cost = G[o][u][i].second;
19.            if (d[v] > d[u] + cost) {
20.                d[v] = d[u] + cost;
21.                if (in[v] == 0) {
22.                    in[v] = 1;
23.                    que.push(v);
24.                }
25.            }
26.        }
27.    }
28. }
29. struct Edge{
30.     int from, to, cap, flow;
31.     Edge(){}
32.     Edge(int from, int to, int cap, int flow):from(from), to(to), cap(cap), flow(flow){}
33. };
34. struct ISAP{
35.     int p[maxn], num[maxn], cur[maxn], d[maxn];
36.     int s, t, n, m;
37.     bool vis[maxn];
38.
39.     vector<int> G[maxn];
40.     vector<Edge> edges;
41.
42.     void init(int n) {
43.         this->n = n;
44.         for (int i = 0; i <= n; ++i) {
45.             G[i].clear();
46.             d[i] = INF;
47.         }
48.         edges.clear();
49.     }
50.
51.     void addedge(int from, int to, int cap) {
52.         edges.push_back(Edge(from, to, cap, 0));
53.         edges.push_back(Edge(to, from, 0, 0));
54.         m = (int)edges.size();
55.         G[from].push_back(m - 2);
56.         G[to].push_back(m - 1);
57.     }
58.
59.     bool bfs() {
60.         memset(vis, false, sizeof vis);
61.
62.         queue<int> que;
63.         d[t] = 0;
64.         vis[t] = true;
65.         que.push(t);
66.
67.         while(!que.empty()) {
68.             int u = que.front();
69.             que.pop();
70.
71.             for (int i = 0; i < G[u].size(); ++i) {
```



```

72.         Edge& e = edges[G[u][i]^1];
73.         if (e.cap > e.flow && !vis[e.from]) {
74.             vis[e.from] = true;
75.             d[e.from] = d[u] + 1;
76.             que.push(e.from);
77.         }
78.     }
79. }
80. return vis[s];
81. }
82.
83. int Augment() {
84.     int u = t, flow = INF;
85.     while(u != s) {
86.         Edge& e = edges[p[u]];
87.         flow = min(flow, e.cap - e.flow);
88.         u = edges[p[u]].from;
89.     }
90.
91.     u = t;
92.     while(u != s) {
93.         edges[p[u]].flow += flow;
94.         edges[p[u]^1].flow -= flow;
95.         u = edges[p[u]].from;
96.     }
97.     return flow;
98. }
99.
100. int MaxFlow(int s,int t) {
101.     this->s = s,this->t = t;
102.     int ret = 0;
103.     bfs();
104.     if (d[s] >= n) return 0;
105.
106.     memset(num, 0,sizeof num);
107.     memset(cur, 0,sizeof cur);
108.     for (int i = 0;i < n;++i) {
109.         if (d[i] < INF) num[d[i]]++;
110.     }
111.     int u = s;
112.
113.     while(d[s] < n) {
114.
115.         if (u == t) {
116.             ret += Augment();
117.             u = s;
118.         }
119.
120.         bool ok = false;
121.         for (int i = cur[u];i < G[u].size();++i) {
122.             Edge& e = edges[G[u][i]];
123.             if (e.cap > e.flow && d[u] == d[e.to] + 1) {
124.                 ok = true;
125.                 p[e.to] = G[u][i];
126.                 cur[u] = i;
127.                 u = e.to;
128.                 break;
129.             }
130.         }
131.
132.         if (!ok) {
133.             int Min = n - 1;
134.             for (int i = 0;i < G[u].size();++i) {
135.                 Edge& e = edges[G[u][i]];
136.                 if (e.cap > e.flow) Min = min(Min, d[e.to]);
137.             }
138.             if (--num[d[u]] == 0) break;
139.             num[d[u] = Min + 1]++;
140.             cur[u] = 0;
141.             if (u != s) u = edges[p[u]].from;
142.         }
143.     }
144.     return ret;
145. }
146. }solve;
147. int main(int argc, const char * argv[])
148. {
149.     // freopen("in.txt","r",stdin);
150.     // freopen("out.txt","w",stdout);
151.     // clock_t _ = clock();
152.
153.     int t;
154.     cin >> t;
155.     while(t--) {
156.         scanf("%d%d", &n, &m);
157.         for (int i = 1;i <= n;++i)
158.             G[0][i].clear(), G[1][i].clear();
159.         int u, v, c;
160.         for (int i = 1;i <= m;++i) {
161.             scanf("%d%d%d", &u, &v, &c);
162.             G[0][u].push_back(ii(v, c));
163.             G[1][v].push_back(ii(u, c));
164.         }
165.         int st, ed;
166.         scanf("%d%d", &st, &ed);
167.         spfa(st, ed, dis1, 0);
168.         spfa(ed, st, dis2, 1);

```

```

169.     int limit = dis1[ed];
170.     if (limit == INF) {
171.         printf("%d\n", 0);
172.         continue;
173.     }
174.
175.     solve.init(n);
176.
177.     for (int i = 1; i <= n; ++i) {
178.         for (int j = 0; j < G[0][i].size(); ++j) {
179.             int k = G[0][i][j].first;
180.             if (dis1[i] != INF && dis2[k] != INF && dis1[i] + dis2[k] + G[0][i][j].second == limit) {
181.                 solve.addedge(i, k, 1);
182.             }
183.         }
184.     }
185.     int ans = solve.MaxFlow(st, ed);
186.     printf("%d\n", ans);
187. }
188.
189. // printf("\nTime cost: %.2fs\n", 1.0 * (clock() - _) / CLOCKS_PER_SEC);
190. return 0;
191. }

```

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。

```

1.  /*****
2.  Author      :Crazy_AC(JamesQi)
3.  Time       :2016
4.  File Name  :
5.
6.              _ooOoo_
7.              o8888888o
8.              88" . "88
9.              (| -_- |)
10.             O\ = /O
11.
12.             /---'\---/
13.            /'---'  \---'\
14.           /___|___|\___\
15.          /   |   |   \
16.         /___|___|\___\
17.        /___|___|\___\
18.       /___|___|\___\
19.      /___|___|\___\
20.     /___|___|\___\
21.    /___|___|\___\
22.   /___|___|\___\
23.  /___|___|\___\
24.  =====
25.  ~~~~~
26.
27.          佛祖保佑      永无BUG
28.
29.  *****/
30.  // #pragma comment(linker, "/STACK:102400000,102400000")
31.  #include <iostream>
32.  #include <algorithm>
33.  #include <iomanip>
34.  #include <iostream>
35.  #include <string>
36.  #include <stack>
37.  #include <queue>
38.  #include <deque>
39.  #include <vector>
40.  #include <map>
41.  #include <set>
42.  #include <cstdio>
43.  #include <cstring>
44.  #include <cmath>
45.  #include <cstdlib>
46.  #include <climits>
47.  using namespace std;
48.  #define MEM(x,y) memset(x, y,sizeof x)
49.  #define pk push_back
50.  #define lson rt << 1
51.  #define rson rt << 1 | 1
52.  #define bug cout << "BUG HERE\n"
53.  typedef long long LL;
54.  typedef unsigned long long ULL;
55.  typedef pair<int,int> ii;
56.  typedef pair<ii,int> iii;
57.  const double eps = 1e-8;
58.  const double pi = 4 * atan(1);
59.  const long long inf = 1e20 + 10;
60.  const int INF = 0x3f3f3f3f;
61.  const int MOD = 1e9 + 7;
62.  int nCase = 0;
63.  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.     }
71.     return x;
72. }
73. const int maxn = 1e4 + 10;
74. struct Edge {
75.     int v, limit, cost;
76.     Edge() {}
77.     Edge(int v,int limit, int cost) {
78.         this->v = v;
79.         this->limit = limit;
80.         this->cost = cost;
81.     }
82. };
83. vector<Edge> G[maxn];
84. int n, m, k;
85. struct node {
86.     int p;
87.     long long cost;
88.     node() {}
89.     node(int p,long long cost) {
90.         this->p = p;
91.         this->cost = cost;
92.     }
93.     bool operator < (const node& rhs) const { //小的优先
94.         return cost > rhs.cost;
95.     }
96. };
97. long long dis[maxn];
98. int vis[maxn];
99. bool solve(int limit) {
100.     // memset(dis, INF,sizeof dis);
101.     for (int i = 1;i <= n;++i)
102.         dis[i] = inf;
103.     memset(vis, 0,sizeof vis);
104.     dis[1] = 0;
105.     priority_queue<node> que;
106.     que.push(node(1, 0));
107.     while(!que.empty()) {
108.         node temp = que.top();
109.         que.pop();
110.         int u = temp.p;
111.         if (vis[u]) continue;
112.         vis[u] = 1;
113.         for (int i = 0;i < G[u].size();++i) {
114.             if (G[u][i].limit < limit) continue;
115.             int v = G[u][i].v;
116.             long long cost = G[u][i].cost;
117.             if (dis[v] > dis[u] + cost) {
118.                 dis[v] = dis[u] + cost;
119.                 que.push(node(v, dis[v]));
120.             }
121.         }
122.     }
123.     return dis[n] <= k;
124. }
125.
126.
127. int main(int argc, const char * argv[])
128. {
129.     freopen("in.txt","r",stdin);
130.     // freopen("out.txt","w",stdout);
131.     // clock_t _ = clock();
132.
133.     int t;
134.     scanf("%d",&t);
135.     while(t--) {
136.         scanf("%d%d%d", &n, &m, &k);
137.         for (int i = 1;i <= n;++i)
138.             G[i].clear();
139.         int u, v, l, c;
140.         int high = 0, low = INF;
141.         for (int i = 1;i <= m;++i) {
142.             scanf("%d%d%d%d", &u, &v, &l, &c);
143.             G[u].push_back(Edge(v, l, c));
144.             G[v].push_back(Edge(u, l, c));
145.             high = max(high, l);
146.             low = min(low, l);
147.         }
148.         int ans = -1;
149.         while(low <= high) {
150.             int mid = (high + low) >> 1;
151.             if (solve(mid)) {
152.                 low = mid + 1;
153.                 ans = mid;
154.             }else high = mid - 1;
155.         }
156.         if (solve(low)) ans = max(ans, low);

```

```
157.         if (solve(high)) ans = max(ans, high);
158.         if (ans == -1) puts("Poor RunningPhoton!");
159.         else printf("%d\n", ans);
160.     }
161.
162.     // printf("\nTime cost: %.2fs\n", 1.0 * (clock() - _) / CLOCKS_PER_SEC);
163.     return 0;
164. }
```

I.玛雅，好简单

题意

略

分析

求无向图桥边的模版题。。。

```
1.  /*****
2.  Author       :Crazy_AC(JamesQi)
3.  Time        :2016
4.  File Name   :
5.
6.              _oo0oo_
7.              o8888888o
8.              88" . "88
9.              (| -_- |)
10.             O\ = /O
11.
12.          /--'\
13.         /'--' \
14.        /'''\
15.       /'''\
16.      /'''\
17.     /'''\
18.    /'''\
19.   /'''\
20.  /'''\
21. /'''\
22. /'''\
23. /'''\
24. =====
25. 佛祖保佑    永无BUG
26. *****/
27. // #pragma comment(linker, "/STACK:1024000000,1024000000")
28. #include <iostream>
29. #include <algorithm>
30. #include <iomanip>
31. #include <sstream>
32. #include <string>
33. #include <stack>
34. #include <queue>
35. #include <deque>
36. #include <vector>
37. #include <map>
38. #include <set>
39. #include <cstdio>
40. #include <cstring>
41. #include <cmath>
42. #include <cstdlib>
43. #include <limits>
44. using namespace std;
45. #define MEM(x,y) memset(x, y, sizeof x)
46. #define pk push_back
47. #define lson rt << 1
48. #define rson rt << 1 | 1
49. #define bug cout << "BUG HERE\n"
50. typedef long long LL;
51. typedef unsigned long long ULL;
52. typedef pair<int,int> ii;
53. typedef pair<ii,int> iii;
54. const double eps = 1e-8;
55. const double pi = 4 * atan(1);
56. const int inf = 1 << 30;
57. const int INF = 0x3f3f3f3f;
58. const int MOD = 1e9 + 7;
59. int nCase = 0;
60. int dcmp(double x){//精度正负、0的判断
61.     if (fabs(x) < eps) return 0;
62.     return x < 0?-1:1;
63. }
64. inline int read(){
65.     char c = getchar();
66.     while (!isdigit(c)) c = getchar();
67.     int x = 0;
68.     while (isdigit(c)) {
69.         x = x * 10 + c - '0';
70.         c = getchar();
71.     }
72.     return x;
73. }
74. const int maxn = 10010;
75. vector<int> G[maxn];
```

```
75. int dfn[maxn], low[maxn], depth;
76. bool in[maxn];
77. int cnt;
78. stack<int> st;
79. void dfs(int u,int fa) {
80.     dfn[u] = low[u] = ++depth;
81.     st.push(u);
82.     in[u] = true;
83.     int first = 1;
84.     for (int i = 0;i < G[u].size();++i) {
85.         int v = G[u][i];
86.         if (first && v == fa) {
87.             first = 0;
88.             continue;
89.         }
90.         if (dfn[v] == -1) {
91.             dfs(v, u);
92.             if (low[v] > dfn[u]) cnt++;
93.             low[u] = min(low[u], low[v]);
94.         }else if (in[v]) low[u] = min(low[u], dfn[v]);
95.     }
96.     if (dfn[u] == low[u]) {
97.         while(true) {
98.             int x = st.top();
99.             st.pop();
100.            in[x] = false;
101.            if (x == u) break;
102.        }
103.    }
104. }
105. int n, m;
106. void solve() {
107.     memset(dfn, -1,sizeof dfn);
108.     cnt = depth = 0;
109.     for (int i = 1;i <= n;++i)
110.         if (dfn[i] == -1) dfs(i, -1);
111.     printf("Case %d: %d\n", ++nCase, cnt);
112. }
113. int main(int argc, const char * argv[])
114. {
115.     // freopen("in.txt","r",stdin);
116.     // freopen("out.txt","w",stdout);
117.     // clock_t _ = clock();
118.
119.     int t;
120.     scanf("%d", &t);
121.     while(t--) {
122.         scanf("%d%d", &n, &m);
123.         for (int i = 1;i <= n;++i)
124.             G[i].clear();
125.         int u, v;
126.         for (int i = 1;i <= m;++i) {
127.             scanf("%d%d", &u, &v);
128.             G[u].push_back(v);
129.             G[v].push_back(u);
130.         }
131.         solve();
132.     }
133.
134.     // printf("\nTime cost: %.2fs\n", 1.0 * (clock() - _) / CLOCKS_PER_SEC);
135.     return 0;
136. }
```

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。

```
1. const int N = 1010;
2. vector<int> G1[N], G2[N], G3[N]; //原图, 扩张图, 缩点后的图。
3. bool vis[N];
4. int pre[N], low[N], Belong[N], scc_cnt, Times;
5. int n, m;
6. stack<int> st;
7. void Tarjan(int u) {
8.     pre[u] = low[u] = ++Times;
9.     st.push(u);
10.    for (int i = 0;i < (int)G2[u].size();++i) {
11.        int v = G2[u][i];
12.        if (!pre[v]) {
13.            Tarjan(v);
14.            low[u] = min(low[u], low[v]);
15.        }else if (!Belong[v]) low[u] = min(low[u], pre[v]);
16.    }
17.    if (pre[u] == low[u]) {
18.        scc_cnt++;
19.        while(true) {
20.            int x = st.top();
```

```
21.         st.pop();
22.         Belong[x] = scc_cnt;
23.         if (x == u) break;
24.     }
25. }
26. }
27. void FindSCC() {
28.     memset(pre, 0, sizeof pre);
29.     memset(Belong, 0, sizeof Belong);
30.     Times = scc_cnt = 0;
31.     for (int i = 1; i <= n; ++i) {
32.         if (!pre[i]) Tarjan(i);
33.     }
34. }
35. void BFS(int st) {
36.     queue<int> que;
37.     que.push(st);
38.     memset(vis, false, sizeof vis);
39.     vis[st] = true;
40.     while(!que.empty()) {
41.         int u = que.front();
42.         que.pop();
43.         for (int i = 0; i < G1[u].size(); ++i) {
44.             int v = G1[u][i];
45.             if (vis[v]) continue;
46.             vis[v] = true;
47.             G2[st].push_back(v); //对原图进行这种扩展是不会形成可行环的。
48.             que.push(v);
49.         }
50.     }
51. }
52. void Initiation() {
53.     for (int i = 1; i <= n; ++i)
54.         BFS(i);
55. }
56. void Input() {
57.     scanf("%d%d", &n, &m);
58.     for (int i = 1; i <= n; ++i)
59.         G1[i].clear(), G2[i].clear(), G3[i].clear();
60.     int u, v;
61.     for (int i = 1; i <= m; ++i) {
62.         scanf("%d%d", &u, &v);
63.         G1[u].push_back(v);
64.     }
65. }
66. void Trans() {
67.     for (int u = 1; u <= n; ++u) {
68.         for (int i = 0; i < G2[u].size(); ++i) {
69.             int v = G2[u][i];
70.             if (Belong[u] != Belong[v])
71.                 G3[Belong[u]].push_back(Belong[v]);
72.         }
73.     }
74. }
75. int linker[N];
76. bool Search(int u) {
77.     for (int i = 0; i < G3[u].size(); ++i) {
78.         int v = G3[u][i];
79.         if (vis[v]) continue;
80.         vis[v] = true;
81.         if (linker[v] == -1 || Search(linker[v])) {
82.             linker[v] = u;
83.             return true;
84.         }
85.     }
86.     return false;
87. }
88. int Hungary() {
89.     int ret = 0;
90.     memset(linker, -1, sizeof linker);
91.     for (int i = 1; i <= scc_cnt; ++i) {
92.         memset(vis, false, sizeof vis);
93.         if (Search(i)) ret++;
94.     }
95.     return ret;
96. }
97. int main()
98. {
99.     // freopen("in.txt", "r", stdin);
100.    // freopen("out.txt", "w", stdout);
101.    int t, icase = 0;
102.    scanf("%d", &t);
103.    while(t--) {
104.        Input();
105.        Initiation();
106.        FindSCC();
107.        // printf("SCC = %d\n", scc_cnt);
108.        // for (int i = 1; i <= n; ++i)
109.        //     printf("%d ", Belong[i]);
110.        // puts("");
111.        Trans();
112.
113.        // for (int i = 1; i <= n; ++i) {
114.        //     printf("<td>:", i);
115.        //     for (int j = 0; j < G2[i].size(); ++j)
116.        //         printf("%d ", G2[i][j]);
117.        //     puts("");
118.    }
```

```

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.
6.              _ooOoo_
7.              o8888888o
8.              88" . "88
9.              (| -_- |)
10.             O\ = /O
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.     }

```

```
71.     return x;
72. }
73. const int maxn = 5010;
74. const int maxm = 30010;
75. int head[maxn], pnt[maxm], nxt[maxm], ecnt;
76. int n, m;
77. int dfn[maxn], low[maxn], in[maxn], depth;
78. int belong[maxn], block;
79. int cnt1[maxn], cnt2[maxn], cnt3[maxn];
80. stack<int> st;
81. void dfs(int u) {
82.     dfn[u] = low[u] = ++depth;
83.     in[u] = 1;
84.     st.push(u);
85.     for (int i = head[u]; ~i; i = nxt[i]) {
86.         int v = pnt[i];
87.         if (dfn[v] == -1) {
88.             dfs(v);
89.             low[u] = min(low[u], low[v]);
90.         } else if (in[v]) low[u] = min(low[u], dfn[v]);
91.     }
92.     if (dfn[u] == low[u]) {
93.         block++;
94.         while(true) {
95.             int x = st.top();
96.             st.pop();
97.             in[x] = 0;
98.             belong[x] = block;
99.             cnt1[block]++;
100.            if (x == u) break;
101.            // cnt1[block]++;
102.        }
103.    }
104. }
105. void find_scc() {
106.     memset(dfn, -1, sizeof dfn);
107.     memset(in, 0, sizeof in);
108.     memset(cnt1, 0, sizeof cnt1); //scc
109.     depth = block = 0;
110.     for (int i = 0; i < n; ++i)
111.         if (dfn[i] == -1) dfs(i);
112. }
113. vector<int> G[maxn];
114. int mark[maxn];
115. int search(int u) {
116.     int sum = 0;
117.     for (int i = 0; i < G[u].size(); ++i) {
118.         int v = G[u][i];
119.         if (mark[v]) continue;
120.         mark[v] = 1;
121.         sum += cnt1[v];
122.         sum += search(v);
123.     }
124.     return sum;
125. }
126. void solve() {
127.     for (int i = 1; i <= block; ++i)
128.         G[i].clear();
129.     //rebuild new graph
130.     for (int u = 0; u < n; ++u) {
131.         for (int i = head[u]; ~i; i = nxt[i]) {
132.             int v = pnt[i];
133.             if (belong[u] != belong[v]) {
134.                 G[belong[v]].push_back(belong[u]);
135.                 in[belong[u]]++;
136.             }
137.         }
138.     }
139.     memset(cnt2, 0, sizeof cnt2);
140.     for (int i = 1; i <= block; ++i) {
141.         if (in[i] == 0) {
142.             memset(mark, 0, sizeof mark);
143.             cnt2[i] = search(i);
144.         }
145.     }
146.     int Max = 0;
147.     for (int i = 0; i < n; ++i) {
148.         cnt3[i] = cnt1[belong[i]] + cnt2[belong[i]] - 1;
149.         // cout << "cnt3 = " << cnt3[i] << endl;
150.         Max = max(Max, cnt3[i]);
151.     }
152.     printf("Case %d: %d\n", ++nCase, Max);
153.     int first = 1;
154.     for (int i = 0; i < n; ++i) {
155.         if (cnt3[i] == Max) {
156.             if (first) first = 0;
157.             else printf(" ");
158.             printf("%d", i);
159.         }
160.     }
161.     printf("\n");
162. }
163. int main(int argc, const char * argv[])
164. {
165.     // freopen("in.txt", "r", stdin);
166.     // freopen("out.txt", "w", stdout);
167.     // 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. }
```