**例1、骑士的的放置**

1. **inline int ID(int x, int y)**
2. **{**
3. **int id = (x - 1) \* m + y;**
4. **if(x < 1 || x > n || y < 1 || y > m || flag[id])**
5. **return -1;**
6. **return id;**
7. **}**
9. **bool dfs(int u)**
10. **{**
11. **for(int v : G[u])**
12. **{**
13. **if(vis[v]) continue;**
14. **vis[v] = 1;**
15. **if(!match[v] || dfs(match[v]))**
16. **{**
17. **match[v] = u;**
18. **return true;**
19. **}**
20. **}**
21. **return false;**
22. **}**
24. **void init()**
25. **{**
26. **for(int i = 1; i <= k; i++)**
27. **{**
28. **int x, y;**
29. **read(x); read(y);**
30. **flag[ID(x, y)] = 1;**
31. **}**
33. **for(int x = 1; x <= n; x++)**
34. **{**
35. **int lx, ly, id1, id2;**
36. **for(int y = 1; y <= m; y++)**
37. **{**
38. **id1 = ID(x, y);**
39. **if(id1 == -1 || (x + y) & 1) continue;**
40. **for(int i = 0; i < 8; i++)**
41. **{**
42. **lx = x + dir[0][i];**
43. **ly = y + dir[1][i];**
44. **id2 = ID(lx, ly);**
45. **if(id2 != -1) G[id1].pb(id2);**
46. **}**
47. **}**
48. **}**
49. **}**
51. **signed main()**
52. **{**
53. **read(n); read(m); read(k);**
55. **init();**
57. **int ans = 0;**
58. **for(int i = 1; i <= n \* m; i++)**
59. **{**
60. **mem(vis);**
61. **if(dfs(i)) ans++;**
62. **}**
64. **printf("%d", n \* m - ans - k);**
65. **return 0;**
66. **}**

**例2、POJ 3041-Asteroids**

1. **bool dfs(int u)**
2. **{**
3. **for(int i = 0, v; i < G[u].size(); i++)**
4. **{**
5. **v = G[u][i];**
6. **if(vis[v] == dfn) continue;**
7. **vis[v] = dfn;**
8. **if(!match[v] || dfs(match[v]))**
9. **{**
10. **match[v] = u;**
11. **return true;**
12. **}**
13. **}**
14. **return false;**
15. **}**
17. **void work()**
18. **{**
19. **for(int i = 0; i <= n; i++) G[i].clear(), match[i] = 0;**
21. **while(k--)**
22. **{**
23. **int x, y;**
24. **read(x); read(y);**
25. **G[x].pb(y);**
26. **}**
28. **int ans = 0;**
29. **for(int i = 1; i <= n; i++)**
30. **{**
31. **dfn++;**
32. **if(dfs(i)) ans++;**
33. **}**
35. **printf("%d\n", ans);**
36. **}**
37. **signed main()**
38. **{**
39. **while(~scanf("%d%d", &n, &k))**
40. **work();**
41. **return 0;**
42. **}**

**例3、A-HDU 1045-Fire Net**

1. **char mp[N][N];**
2. **int flagx[N][N], flagy[N][N];**
3. **int vis[N], match[N];**
4. **int n, m, cntx, cnty;**
6. **vector<int> G[N];**
7. **bool dfs(int u)**
8. **{**
9. **for(int v : G[u])**
10. **{**
11. **if(vis[v]) continue;**
12. **vis[v] = 1;**
13. **if(!match[v] || dfs(match[v]))**
14. **{**
15. **match[v] = u;**
16. **return true;**
17. **}**
18. **}**
19. **return false;**
20. **}**
21. **void work()**
22. **{**
23. **mem(flagx); mem(flagy); mem(match);**
24. **for(int i = 0; i < N; i++)**
25. **G[i].clear();**
26. **cntx = cnty = 0;**
28. **for(int i = 1; i <= n; i++)**
29. **for(int j = 1; j <= n; j++)**
30. **cin >> mp[i][j];**
32. **for(int i = 1; i <= n; i++)**
33. **{**
34. **for(int j = 1; j <= n; j++)**
35. **{**
36. **//            if(mp[i][j] == 'X') continue;**
37. **if(mp[i][j] == '.')**
38. **{**
39. **if(j == 1) flagx[i][j] = ++cntx;**
40. **else       flagx[i][j] = mp[i][j - 1] == '.' ? flagx[i][j - 1] : ++cntx;**
41. **}**
43. **if(mp[j][i] == '.')**
44. **{**
45. **if(j == 1) flagy[j][i] = ++cnty;**
46. **else       flagy[j][i] = mp[j - 1][i] == '.' ? flagy[j - 1][i] : ++cnty;**
47. **}**
48. **}**
49. **}**
51. **for(int i = 1; i <= n; i++)**
52. **for(int j = 1; j <= n; j++)**
53. **G[flagx[i][j]].pb(flagy[i][j]);**
55. **int ans = 0;**
56. **for(int i = 1; i <= cntx; i++)**
57. **{**
58. **mem(vis);**
59. **ans += dfs(i);**
60. **}**
61. **cout << ans << endl;**
62. **}**
63. **signed main()**
64. **{**
65. **Buff;**
66. **while(cin >> n)**
67. **if(n) work();**
68. **else  break;**
69. **//        work();**
70. **return 0;**
71. **}**

**例4、E-HDU 2819-Swap**

1. **void** init()
2. {
3. dfn = 0;
4. **for**(**int** i = 0; i <= n; i++)
5. {
6. match[i] = vis[i] = 0;
7. G[i].clear();
8. }
9. **for**(**int** u = 1; u <= n; u++)
10. **for**(**int** v = 1; v <= n; v++)
11. {
12. **int** x; read(x);
13. **if**(x)  G[u].pb(v);
14. }
15. }
17. **bool** dfs(**int** u)
18. {
19. **for**(**int** i = 0, v; i < G[u].size(); i++)
20. {
21. v = G[u][i];
22. **if**(vis[v] == dfn) **continue**;
23. vis[v] = dfn;
24. **if**(!match[v] || dfs(match[v]))
25. {
26. match[v] = u;
27. \_match[u] = v;
28. **return** **true**;
29. }
30. }
31. **return** **false**;
32. }
34. **void** work()
35. {
36. init();
38. **int** ans = 0;
39. **for**(**int** i = 1; i <= n; i++)
40. {
41. dfn++;
42. **if**(dfs(i)) ans++;
43. }
45. **if**(ans < n)
46. {
47. printf("-1\n");
48. **return** ;
49. }
51. **int** cnt = 0;
53. **for**(**int** i = 1; i <= n; i++)
54. {
55. **int** j = 1;
56. **while**(j <= n && match[j] != i) j++;
57. **if**(i != j)
58. {
59. a[++cnt] = i; b[cnt] = j;
60. swap(match[i], match[j]);
61. }
62. }
64. printf("%d\n", cnt);
65. **for**(**int** i = 1; i <= cnt; i++)
66. printf("C %d %d\n", a[i], b[i]);
67. }
69. **signed** main()
70. {
71. **while**(~read(n))
72. work();
73. **return** 0;
74. }

** 例5、H-POJ 3020-Antenna Placement**

1. **int** dir[2][4] = {{0, 0, 1, -1}, {1, -1, 0, 0}};
2. **int** ID(**int** x, **int** y)
3. {
4. **if**(x < 1 || x > n || y < 1 || y > m || mp[x][y] == 'o')
5. **return** -1;
6. **return** m \* (x - 1) + y;
7. }
9. **void** init()
10. {
11. read(n); read(m);
13. ans = dfn = 0;
14. **for**(**int** i = 0; i <= n \* m; i++)
15. {
16. match[i] = vis[i] = 0;
17. G[i].clear();
18. }
20. **for**(**int** i = 1; i <= n; i++) reads(mp[i] + 1);
21. **for**(**int** x = 1; x <= n; x++)
22. {
23. **int** id1, id2, lx, ly;
24. **for**(**int** y = 1; y <= m; y++)
25. {
26. **if**(mp[x][y] == 'o') **continue**;
27. id1 = ID(x, y);
28. ans++;
29. **for**(**int** i = 0; i < 4; i++)
30. {
31. lx = x + dir[0][i];
32. ly = y + dir[1][i];
33. id2 = ID(lx, ly);
34. **if**(id2 != -1) G[id1].pb(id2);
35. }
36. }
37. }
38. }
39. **bool** dfs(**int** u)
40. {
41. **for**(**int** i = 0, v; i < G[u].size(); i++)
42. {
43. v = G[u][i];
44. **if**(vis[v] == dfn) **continue**;
45. vis[v] = dfn;
46. **if**(!match[v] || dfs(match[v]))
47. {
48. match[v] = u;
49. **return** **true**;
50. }
51. }
52. **return** **false**;
53. }
55. **void** work()
56. {
57. init();
59. **int** cnt = 0;
60. **for**(**int** i = 1; i <= n \* m; i++)
61. {
62. dfn++;
63. **if**(dfs(i)) cnt++;
64. }
65. printf("%d\n", ans - cnt / 2);
66. }
67. **signed** main()
68. {
69. **int** t; read(t);
70. **while**(t--)
71. work();
72. **return** 0;
73. }

**例6、CH 6803 导弹防御塔**

1. **const** **double** eps = 1e-8;
2. **int** n, m, tot;
3. **double** T1, T2, V;
4. **struct** Point
5. {
6. **double** x, y;
7. Point(**double** x = 0, **double** y = 0) : x(x), y(y){}
8. }p1[N], p2[N];
10. **struct** node
11. {
12. **int** id;
13. **double** t;
14. }a[N];
16. **double** getDis(**const** Point& A, **const** Point& B)
17. {
18. **double** x = A.x - B.x;
19. **double** y = A.y - B.y;
20. **return** sqrt(x \* x + y \* y);
21. }
23. **void** init()
24. {
25. mem(match);
26. **for**(**int** i = 0; i <= m; i++)
27. G[i].clear();
28. }
30. **bool** dfs(**int** u)
31. {
32. **for**(**int** v : G[u])
33. {
34. **if**(vis[v]) **continue**;
35. vis[v] = 1;
36. **if**(!match[v] || dfs(match[v]))
37. {
38. match[v] = u;
39. **return** **true**;
40. }
41. }
42. **return** **false**;
43. }
45. **bool** judge(**double** mid)
46. {
47. init();
48. **for**(**int** i = 1; i <= m; i++)
49. **for**(**int** j = 1; j <= tot; j++)
50. **if**(a[j].t + getDis(p1[a[j].id], p2[i]) / V <= mid)
51. G[i].pb(j);
52. **for**(**int** i = 1; i <= m; i++)
53. {
54. mem(vis);
55. **if**(!dfs(i)) **return** **false**;
56. }
57. **return** **true**;
58. }
60. **signed** main()
61. {
62. read(n); read(m); readf(T1); readf(T2);readf(V);
63. T1 /= 60.00;
64. **for**(**int** i = 1; i <= m; i++)
65. {
66. readf(p2[i].x); readf(p2[i].y);
67. }
68. **for**(**int** i = 1; i <= n; i++)
69. {
70. readf(p1[i].x); readf(p1[i].y);
71. }
73. tot = n \* m;
74. **for**(**int** i = 1; i <= n; i++)
75. **for**(**int** j = 1; j <= m; j++)
76. {
77. a[(i - 1) \* m + j].id = i;
78. a[(i - 1) \* m + j].t = (j - 1) \* (T1 + T2) + T1;
79. }
81. **double** l = T1, r = 1e5, mid, ans = 0.00;
82. **while**(r - l >= eps)
83. {
84. mid = (l + r) / 2;
85. **if**(judge(mid)) r = mid - eps, ans = mid;
86. **else**           l = mid + eps;
87. }
88. printf("%.6lf\n", ans);
89. **return** 0;
90. }

**例7、M-POJ 2289-Jamie's Contact Groups**

1. **const** **double** eps = 1e-8;
2. **struct** node
3. {
4. **int** cnt;
5. **int** kid[N];
6. }match[N];
7. **void** init()
8. {
9. dfn = 0;
10. **for**(**int** i = 0; i <= n; i++) G[i].clear();
11. **char** str[20], op;
12. **for**(**int** i = 1, x; i <= n; i++)
13. {
14. reads(str);
15. **while**(1)
16. {
17. read(x); readc(op);
18. G[i].pb(x + 1);
19. **if**(op == '\n') **break**;
20. }
21. }
22. }
24. **bool** dfs(**int** u, **int** mid)
25. {
26. **for**(**int** i = 0, v; i < G[u].size(); i++)
27. {
28. v = G[u][i];
29. **if**(vis[v] == dfn) **continue**;
30. vis[v] = dfn;
31. **if**(match[v].cnt < mid)
32. {
33. match[v].kid[++match[v].cnt] = u;
34. **return** **true**;
35. }
36. **for**(**int** j = 1; j <= match[v].cnt; j++)
37. {
38. **if**(dfs(match[v].kid[j], mid))
39. {
40. match[v].kid[j] = u;
41. **return** **true**;
42. }
43. }
44. }
45. **return** **false**;
46. }
48. **bool** judge(**int** mid)
49. {
50. mem(match);
51. mem(vis); dfn = 0;
52. **int** ans = 0;
53. **for**(**int** i = 1; i <= n; i++)
54. {
55. dfn++;
56. **if**(!dfs(i, mid)) **return** **false**;
57. }
58. **return** 1;
59. }
61. **void** work()
62. {
63. init();
65. **int** L = 1, R = n, mid;
66. **while**(L < R)
67. {
68. mid = (L + R) >> 1;
69. **if**(judge(mid)) R = mid;
70. **else**           L = mid + 1;
71. }
73. printf("%d\n", R);
74. }
76. **signed** main()
77. {
78. **while**(~scanf("%d%d", &n, &m), n + m)
79. work();
80. **return** 0;
81. }

**例8、Q-HDU 3488-Tour**

1. **const** **double** eps = 1e-8;
2. **int** vx[N], vy[N];
3. **int** lx[N], ly[N];
4. **bool** dfs(**int** i)
5. {
6. vx[i] = 1;
7. **for**(**int** j = 1; j <= n; j++)
8. **if**(!vy[j] && lx[i] + ly[j] == w[i][j])
9. {
10. vy[j] = 1;
11. **if**(!match[j] || dfs(match[j]))
12. {
13. match[j] = i;
14. **return** **true**;
15. }
16. }
17. **return** **false**;
18. }
20. **void** update()
21. {
22. **int** delta = INF;
23. **for**(**int** i = 1; i <= n; i++)
24. **if**(vx[i])
25. **for**(**int** j = 1; j <= n; j++)
26. **if**(!vy[j])
27. delta = min(delta, lx[i] + ly[j] - w[i][j]);
28. **for**(**int** i = 1; i <= n; i++)
29. {
30. **if**(vx[i]) lx[i] -= delta;
31. **if**(vy[i]) ly[i] += delta;
32. }
33. }
35. **int** KM()
36. {
37. **for**(**int** i = 1; i <= n; i++)
38. {
39. match[i] = lx[i] = ly[i] = 0;
40. **for**(**int** j = 1; j <= n; j++)
41. lx[i] = max(lx[i], w[i][j]);
42. }
44. **for**(**int** i = 1; i <= n; i++)
45. **while**(1)
46. {
47. mem(vx); mem(vy);
48. **if**(dfs(i)) **break**;
49. **else**       update();
50. }
52. **int** ans = 0;
53. **for**(**int** i = 1; i <= n; i++)
54. ans += -w[match[i]][i];
55. **return** ans;
56. }
58. **void** init()
59. {
60. read(n); read(m);
61. **for**(**int** i = 1; i <= n; i++)
62. **for**(**int** j = 1; j <= n; j++)
63. w[i][j] = -INF;
64. **for**(**int** i = 1; i <= m; i++)
65. {
66. **int** u, v, cost;
67. read(u); read(v); read(cost);
68. w[u][v] = max(w[u][v], -cost);
69. }
70. }
72. **signed** main()
73. {
74. **int** t; read(t);
75. **while**(t--)
76. {
77. init();
78. printf("%d\n", KM());
79. }
80. **return** 0;
81. }

**B-HDU 2444-The Accomodation of Students（二分图的判定 + 匈牙利算法）**

1. **bool** dfs(**int** u, **int** col)
2. {
3. color[u] = col;
4. **for**(**int** v : G[u])
5. {
6. **if**(!color[v] && !dfs(v, 3 - col)) **return** **false**;
7. **if**(color[v] == col)               **return** **false**;
8. }
9. **return** **true**;
10. }
11. **bool** dfs(**int** u)
12. {
13. **for**(**int** v : G[u])
14. {
15. **if**(vis[v]) **continue**;
16. vis[v] = 1;
17. **if**(!match[v] || dfs(match[v]))
18. {
19. match[v] = u;
20. **return** **true**;
21. }
22. }
23. **return** **false**;
24. }
25. **void** init()
26. {
27. **for**(**int** i = 0; i <= n; i++)
28. {
29. G[i].clear();
30. match[i] = 0; color[i] = 0;
31. }
32. }
33. **int** work()
34. {
35. init();
36. **for**(**int** i = 1; i <= m; i++)
37. {
38. **int** u, v;
39. read(u); read(v);
40. G[u].pb(v); G[v].pb(u);
41. }
43. **for**(**int** i = 1; i <= n; i++)
44. **if**(!color[i] && !dfs(i, 1)) **return** -1;
46. **int** ans = 0;
47. **for**(**int** i = 1; i <= n; i++)
48. {
49. **if**(color[i] != 1) **continue**;
50. mem(vis);
51. ans += dfs(i);
52. }
54. **return** ans;
55. }
56. **signed** main()
57. {
58. **while**(~scanf("%d%d", &n, &m))
59. {
60. **int** ans = work();
61. **if**(ans == -1) puts("No");
62. **else**          printf("%d\n", ans);
63. }
65. **return** 0;
66. }

**D-HDU 1281-棋盘游戏（拆行列法 + 增广路的理解）**

**大概题意 ：找“車”的最多能放几个，有几个是不可代替的（重要点）**

1. **struct** point
2. {
3. **int** x, y;
4. point(**int** x = 0, **int** y = 0) : x(x), y(y) {}
5. }p[N \* N];
6. **bool** dfs(**int** u)
7. {
8. **for**(**int** v : G[u])
9. {
10. **if**(!flag[u][v] || vis[v]) **continue**;
11. vis[v] = 1;
12. **if**(!match[v] || dfs(match[v]))
13. {
14. match[v] = u;
15. **return** **true**;
16. }
17. }
18. **return** **false**;
19. }
21. **int** Count()
22. {
23. **int** ans = 0;
24. mem(match);
25. **for**(**int** i = 1; i <= n; i++)
26. {
27. mem(vis);
28. **if**(dfs(i)) ans++;
29. }
30. **return** ans;
31. }
33. **int** T = 0;
34. **void** init()
35. {
36. mem(flag);
37. **for**(**int** i = 0; i < N; i++)
38. G[i].clear();
40. **for**(**int** i = 1; i <= k; i++)
41. {
42. **int** u, v;
43. read(u); read(v);
44. flag[u][v] = 1;
45. G[u].pb(v);
46. }
47. }
49. **void** work()
50. {
51. init();
52. **int** ans1 = 0, ans2 = 0;
53. ans1 = Count();
54. **int** cnt = 0;
55. **for**(**int** i = 1; i <= m; i++)
56. {
57. **if**(!match[i]) **continue**;
58. p[++cnt] = point(match[i], i);
59. }
61. **for**(**int** i = 1; i <= ans1; i++)
62. {
63. **int** u = p[i].x, v = p[i].y;
64. flag[u][v] = 0;
65. **if**(ans1 != Count()) ans2++;
66. flag[u][v] = 1;
67. }
68. printf("Board %d have %d important blanks for %d chessmen.\n", ++T, ans2, ans1);
69. }
70. **signed** main()
71. {
72. **while**(~scanf("%d%d%d", &n, &m, &k))
73. work();
74. **return** 0;
75. }

**F-HDU 2389-Rain on your Parade（卡 HK 算法）**

1. **struct** point
2. {
3. **double** x, y, v;
4. point(**double** x = 0, **double** y = 0, **double** v = 0) : x(x), y(y), v(v) {}
5. }p1[N], p2[N];
6. **double** getDis(**const** point &A, **const** point &B)
7. {
8. **double** x = A.x - B.x;
9. **double** y = A.y - B.y;
10. **return** sqrt(x \* x + y \* y);
11. }
12. **void** init()
13. {
14. readf(T);
15. read(n);
16. **for**(**int** i = 1; i <= n; i++)
17. {
18. readf(p1[i].x);
19. readf(p1[i].y);
20. readf(p1[i].v);
21. }
22. read(m);
23. **for**(**int** i = 1; i <= m; i++)
24. {
25. readf(p2[i].x);
26. readf(p2[i].y);
27. }
29. **for**(**int** i = 0; i < N; i++) G[i].clear();
30. nx = n;
32. **for**(**int** i = 1; i <= n; i++)
33. **for**(**int** j = 1; j <= m; j++)
34. {
35. **if**(getDis(p1[i], p2[j]) <= p1[i].v \* T)
36. G[i].pb(j);
37. }
38. }
40. **int** bfs()
41. {
42. queue<**int**> q;
43. **while**(q.size()) q.pop();
44. dis = INF;
45. mem(dx, -1); mem(dy, -1);
46. **for**(**int** i = 1; i <= nx; i++)
47. **if**(lx[i] == -1)  q.push(i), dx[i] = 0;
49. **while**(q.size())
50. {
51. **int** u = q.front(); q.pop();
52. **if**(dx[u] > dis) **break**;
53. **for**(**int** v : G[u])
54. {
55. **if**(dy[v] == -1)
56. {
57. dy[v] = dx[u] + 1;
58. **if**(ly[v] == -1)      dis = dy[v];
59. **else**
60. {
61. dx[ly[v]] = dy[v] + 1;
62. q.push(ly[v]);
63. }
64. }
65. }
66. }
67. **return** dis != INF;
68. }
70. **int** dfs(**int** u)
71. {
72. **for**(**int** v : G[u])
73. {
74. **if**(!vis[v] && dy[v] == dx[u] + 1)
75. {
76. vis[v] = 1;
77. **if**(ly[v] != -1 && dy[v] == dis) **continue**;
78. **if**(ly[v] == -1 || dfs(ly[v]))
79. {
80. lx[u] = v;
81. ly[v] = u;
82. **return** 1;
83. }
84. }
85. }
86. **return** 0;
87. }
89. **int** HK()
90. {
91. mem(lx, -1); mem(ly, -1);
92. **int** ans = 0;
93. **int** x;
94. **while**(bfs())
95. {
96. mem(vis, 0);
97. **for**(**int** i = 1; i <= nx; i++)
98. **if**(lx[i] == -1)
99. ans += dfs(i);
100. }
101. **return** ans;
102. }
104. **signed** main()
105. {
106. **int** t; read(t);
107. **for**(**int** i = 1; i <= t; i++)
108. {
109. init();
110. printf("Scenario #%d:\n%d\n\n",i, HK());
111. }

114. **return** 0;
115. }

**J-HDU 1151-Air Raid（有向图最小路径覆盖）**

1. **int** match[N], vis[N], dfn;
2. **int** dir[2][4] = {{0, 0, 1, -1}, {1, -1, 0, 0}};
3. vector<**int**> G[N];
4. **int** n, m;
6. **void** init()
7. {
8. read(n); read(m);
9. dfn = 0;
10. **for**(**int** i = 0; i <= n; i++)
11. {
12. match[i] = vis[i] = 0;
13. G[i].clear();
14. }
15. **for**(**int** i = 1; i <= m; i++)
16. {
17. **int** u, v;
18. read(u); read(v);
19. G[u].pb(v);
20. }
21. }
23. **bool** dfs(**int** u)
24. {
25. **for**(**int** i = 0, v; i < G[u].size(); i++)
26. {
27. v = G[u][i];
28. **if**(vis[v] == dfn) **continue**;
29. vis[v] = dfn;
30. **if**(!match[v] || dfs(match[v]))
31. {
32. match[v] = u;
33. **return** **true**;
34. }
35. }
36. **return** **false**;
37. }
39. **void** work()
40. {
41. init();
43. **int** ans = 0;
44. **for**(**int** i = 1; i <= n; i++)
45. {
46. dfn++;
47. **if**(dfs(i)) ans++;
48. }
50. printf("%d\n", n - ans);
51. }
53. **signed** main()
54. {
55. **int** t;
56. read(t);
57. **while**(t--)  work();
58. **return** 0;
59. }

**K-POJ 2594-Treasure Exploration（有向图最小路径覆盖）**

1. **void** init()
2. {
3. //    read(n); read(m);
4. dfn = 0;
5. **for**(**int** i = 0; i <= n; i++)
6. {
7. match[i] = vis[i] = 0;
8. G[i].clear(); mem(dp[i]);
9. }
10. **for**(**int** i = 1, u, v; i <= m; i++)
11. {
12. read(u); read(v);
13. dp[u][v] = 1;
14. }
15. **for**(**int** k = 1; k <= n; k++)
16. **for**(**int** i = 1; i <= n; i++)
17. **for**(**int** j = 1; j <= n; j++)
18. dp[i][j] |= dp[i][k] & dp[k][j];
19. **for**(**int** i = 1; i <= n; i++)
20. **for**(**int** j = 1; j <= n; j++)
21. **if**(dp[i][j]) G[i].pb(j);
22. }
23. **bool** dfs(**int** u)
24. {
25. **for**(**int** i = 0, v; i < G[u].size(); i++)
26. {
27. v = G[u][i];
28. **if**(vis[v] == dfn) **continue**;
29. vis[v] = dfn;
30. **if**(!match[v] || dfs(match[v]))
31. {
32. match[v] = u;
33. **return** **true**;
34. }
35. }
36. **return** **false**;
37. }
38. **void** work()
39. {
40. init();
42. **int** ans = 0;
43. **for**(**int** i = 1; i <= n; i++)
44. {
45. dfn++;
46. **if**(dfs(i)) ans++;
47. }
49. printf("%d\n", n - ans);
50. }
51. **signed** main()
52. {
53. **while**(~scanf("%d%d", &n, &m), n + m)
54. work();
55. **return** 0;
56. }

**L-HDU 3829-Cat VS Dog**

1. **struct** people
2. {
3. string like, dis;
4. }A[N], B[N];
6. **void** init()
7. {
8. cntA = cntB = dfn = 0;
9. **for**(**int** i = 0; i < N; i++)
10. {
11. match[i] = vis[i] = 0;
12. G[i].clear();
13. }
15. string str1, str2;
16. **for**(**int** i = 1; i <= k; i++)
17. {
18. cin >> str1 >> str2;
19. **if**(str1[0] == 'C')
20. {
21. A[++cntA].like = str1;
22. A[cntA].dis = str2;
23. }
24. **else**
25. {
26. B[++cntB].like = str1;
27. B[cntB].dis = str2;
28. }
29. }
30. **for**(**int** i = 1; i <= cntA; i++)
31. **for**(**int** j = 1; j <= cntB; j++)
32. **if**(A[i].like == B[j].dis || A[i].dis == B[j].like)
33. G[i].pb(j);
34. }
35. **bool** dfs(**int** u)
36. {
37. **for**(**int** i = 0, v; i < G[u].size(); i++)
38. {
39. v = G[u][i];
40. **if**(vis[v] == dfn) **continue**;
41. vis[v] = dfn;
42. **if**(!match[v] || dfs(match[v]))
43. {
44. match[v] = u;
45. **return** **true**;
46. }
47. }
48. **return** **false**;
49. }
51. **void** work()
52. {
53. init();
55. **int** ans = k;
56. **for**(**int** i = 1; i <= cntA; i++)
57. {
58. dfn++;
59. **if**(dfs(i)) ans--;
60. }
62. cout << ans << endl;
63. }
65. **signed** main()
66. {
67. Buff;
68. **while**(cin >> n >> m >> k)
69. work();
70. **return** 0;
71. }

74. // 二分图的问题 在于 找矛盾与不矛盾，即找 X 部 与 Y 部;

**8) V-POJ 2771-Guardian of Decency**

1. int Abs(**int** x) {  **return** x < 0 ? -x : x;  }
2. **struct** people
3. {
4. **int** h;
5. **char** music[110], sport[110];
6. **bool** judge(people A)
7. {
8. **if**(Abs(h - A.h) > 40) **return** 0;
9. **if**(strcmp(music, A.music)) **return** 0;
10. **if**(strcmp(sport, A.sport) == 0) **return** 0;
11. **return** 1;
12. }
13. }F[N], M[N];
14. **void** init()
15. {
16. read(n);
17. dfn = cntF = cntM = 0;
18. **for**(**int** i = 0; i <= n; i++)
19. {
20. G[i].clear();
21. match[i] = vis[i] = 0;
22. }
24. **int** h;
25. **char** str[101];
26. **for**(**int** i = 1; i <= n; i++)
27. {
28. read(h); reads(str);
29. **if**(str[0] == 'M')
30. {
31. M[++cntM].h = h;
32. reads(M[cntM].music);
33. reads(M[cntM].sport);
35. }
36. **else** **if**(str[0] == 'F')
37. {
38. F[++cntF].h = h;
39. reads(F[cntF].music);
40. reads(F[cntF].sport);
41. }
42. }
44. **for**(**int** i = 1; i <= cntM; i++)
45. **for**(**int** j = 1; j <= cntF; j++)
46. **if**(M[i].judge(F[j])) G[i].pb(j);
47. }
49. **bool** dfs(**int** u)
50. {
51. **for**(**int** i = 0, v; i < G[u].size(); i++)
52. {
53. v = G[u][i];
54. **if**(vis[v] == dfn) **continue**;
55. vis[v] = dfn;
56. **if**(!match[v] || dfs(match[v]))
57. {
58. match[v] = u;
59. **return** **true**;
60. }
61. }
62. **return** **false**;
63. }
64. **void** work()
65. {
66. init();
67. **int** ans = n;
68. **for**(**int** i = 1; i <= cntM; i++)
69. {
70. ++dfn;
71. **if**(dfs(i)) ans--;
72. }
73. printf("%d\n", ans);
74. }
75. **signed** main()
76. {
77. **int** t; read(t);
78. **while**(t--)
79. work();
80. **return** 0;
81. }