二分图匹配常用模板

一、匈牙利算法 + 二分图判定

1. // 二分图判定
2. **bool dfs(int u, int col)**
3. **{**
4. **color[u] = col;**
5. **for(int v : G[u])**
6. **{**
7. **if(!color[v] && !dfs(v, 3 - col)) return false;**
8. **if(color[v] == col)               return false;**
9. **}**
10. **return true;**
11. **}**
12. **// main()**
13. **{**
14. **for(int i = 1; i <= n; i++)**
15. **if(!color[i] && !dfs(i, 1)) return -1;**
16. **}**
17. **//匈牙利算法**
18. **int dfn, ans;**
19. **bool dfs(int u)**
20. **{**
21. **for(int v : G[u])**
22. **{**
23. **if(vis[v]) continue;**
24. **vis[v] = 1;**
25. **if(!match[v] || dfs(match[v]))**
26. **{**
27. **match[v] = u;**
28. **return true;**
29. **}**
30. **}**
31. **return false;**
32. **}**
33. **//main()**
34. **ans = dfn = 0;  //ans : 为最大匹配数，dfn: 为时间戳**
35. **mem(match); mem(vis);**
36. **for(int i = 1; i <= n; i++)**
37. **{**
38. **dfn++;**
39. **ans += dfs(i);**
40. **}**

二、HK算法

1. **vector<int> G[N];**
2. **int dx[N], dy[N], lx[N], ly[N], vis[N];**
3. **int nx, ny, n, m, dis;**
4. **void builGraph()**
5. **{**
6. **for(int i = 0; i < N; i++)**
7. **G[i].clear();**
8. **cin >> n >> m;**
9. **nx = ny = n;**
10. **for(int i = 1; i <= m; i++)**
11. **{**
12. **int u, v;**
13. **cin >> u >> v;**
14. **G[u].pb(v);**
15. **}**
16. **}**
17. **int bfs()**
18. **{**
19. **queue<int> q;**
20. **while(q.size()) q.pop();**
21. **dis = INF;**
22. **mem(dx, -1); mem(dy, -1);**
23. **for(int i = 1; i <= nx; i++)**
24. **if(lx[i] == -1)  q.push(i), dx[i] = 0;**
26. **while(q.size())**
27. **{**
28. **int u = q.front(); q.pop();**
29. **if(dx[u] > dis) break;**
30. **for(int v : G[u])**
31. **{**
32. **if(dy[v] == -1)**
33. **{**
34. **dy[v] = dx[u] + 1;**
35. **if(ly[v] == -1)      dis = dy[v];**
36. **else**
37. **{**
38. **dx[ly[v]] = dy[v] + 1;**
39. **q.push(ly[v]);**
40. **}**
41. **}**
42. **}**
43. **}**
44. **return dis != INF;**
45. **}**
47. **int dfs(int u)**
48. **{**
49. **for(int v : G[u])**
50. **{**
51. **if(!vis[v] && dy[v] == dx[u] + 1)**
52. **{**
53. **vis[v] = 1;**
54. **if(ly[v] != -1 && dy[v] == dis) continue;**
55. **if(ly[v] == -1 || dfs(ly[v]))**
56. **{**
57. **lx[u] = v;**
58. **ly[v] = u;**
59. **return 1;**
60. **}**
61. **}**
62. **}**
63. **return 0;**
64. **}**
66. **int HK()**
67. **{**
68. **int ans = 0;**
69. **mem(lx, -1); mem(ly, -1);**
70. **while(bfs())**
71. **{**
72. **mem(vis, 0);**
73. **for(int i = 1; i <= nx; i++)**
74. **if(lx[i] == -1)**
75. **ans += dfs(i);**
76. **}**
77. **return ans;**
78. **}**
80. **signed main()**
81. **{**
82. **builGraph();**
83. **cout << HK() << endl;**
84. **return 0;**
85. **}**

三、KM算法

1. /\*
2. **二分图最佳匹配：Kuhn-Munkers算法的运行的必要条件是必须存在一个完备匹配**
3. **如果求一个最大权匹配(不一定完备)可以把不存在的边权值赋为0。**
4. **pro1:**
5. **求二分图的最佳匹配（即最大权完备匹配）**
6. **pro2:**
7. **求二分图的最小权完备匹配**
8. **思路：只需将所有的边权值取其相反数，求最大权完备匹配，匹配的值再取相反数即可**
9. **pro3:**
10. **求二分图的乘积最大完备匹配**
11. **思路：则每条边权取自然对数，然后求最大和权匹配，求得的结果a再算出e^a就是最大积匹配**
12. \*/
14. **const double eps = 1e-6;**
15. **int vx[N], vy[N], lx[N], ly[N], match[N], w[N][N];**
16. **int n, m;**
18. **bool dfs(int i)**
19. **{**
20. **vx[i] = 1;**
21. **for(int j = 1; j <= n; j++)**
22. **// 求pro1, pro2**
23. **if(!vy[j] && lx[i] + ly[j] - w == 0)**
24. **//求pro3**
25. **//        if(!vy[j] && lx[i] + ly[j] - w[i][j] < eps)**
26. **{**
27. **vy[j] = 1;**
28. **if(!match[j] || dfs(match[j]))**
29. **{**
30. **match[j] = i;**
31. **return true;**
32. **}**
33. **}**
35. **return false;**
36. **}**
38. **void update()**
39. **{**
40. **// 求 pro3 注意要用double**
41. **int delta = INF;**
42. **for(int i = 1; i <= n; i++)  if(vx[i])**
43. **for(int j = 1; j <= n; j++)  if(!vy[j])**
44. **delta = min(delta, lx[i] + ly[j] - w[i][j]);**
45. **for(int i = 1; i <= n; i++)**
46. **{**
47. **if(vx[i]) lx[i] -= delta;**
48. **if(vy[i]) ly[i] += delta;**
49. **}**
50. **}**
52. **double KM()**
53. **{**
54. **for(int i = 1; i <= n; i++)**
55. **{**
56. **match[i] = lx[i] = ly[i] = 0;**
57. **for(int j = 1; j <= n; j++)**
58. **lx[i] = max(lx[i], w[i][j]);**
59. **}**
61. **for(int i = 1; i <= n; i++)**
62. **while(1)**
63. **{**
64. **mem(vx); mem(vy);**
65. **if(dfs(i)) break;**
66. **else       update();**
67. **}**
69. **int ans = 0;**
70. **//求pro1, pro3(注意用double)**
71. **for(int i = 1; i <= n; i++)**
72. **ans += w[match[i]][i];**
73. **// 求pro2,**
74. **//    for(int i = 1; i <= n; i++)**
75. **//        ans += -w[match[i]][i];**
77. **// 求pro1, pro2;**
78. **return ans;**
79. **//求pro3**
80. **//    return exp(ans);**
81. **}**
83. **void init()**
84. **{**
85. **cin >> n >> m;**
87. **// 初始化, 不论pro1, pro2, pro3都是这样初始化**
88. **for(int i = 1; i <= n; i++)**
89. **for(int i = 1; i <= n; i++)**
90. **w[i][j] = -INF;**
92. **for(int i = 1; i <= m; i++)**
93. **{**
94. **int u, v, cost;**
95. **cin >> u >> v >> cost;**
97. **// 求pro1**
98. **cost = cost;**
99. **// 求pro2**
100. **cost = -cost;**
101. **// 求pro3**
102. **cost = log(cost);**
103. **// 去重边**
104. **w[u][v] = max(m[u][v], cost);**
105. **}**
106. **}**
108. **signed main()**
109. **{**
110. **while(1)**
111. **{**
112. **init();**
113. **cout << KM() << endl;**
114. **}**
115. **return 0;**
116. **}**