# 二分图匹配匈牙利算法(DFS, BFS两种实现模板)

3个重要结论：

最大匹配数：最大匹配的匹配边的数目

最小点覆盖数：选取最少的点，使任意一条边至少有一个端点被选择

最大独立集：选取最多的点，使任意所选两点均不相连

最小路径覆盖数：对于一个 DAG（有向无环图），选取最少条路径，使得每个顶点属于且仅属于一条路径。路径长可以为 0（即单个点）。

最小点覆盖数＝最大匹配数

最小路径覆盖 ＝顶点数－最大匹配数  
二分图最大独立集 = 顶点数 - 最大匹配数

*//匈牙利算法,二分图匹配,DFS实现*

#include <stdio.h>

#include <string.h>

#include <algorithm>

#include <vector>

using namespace std;

#define maxn 100005

int vis[maxn];

int link[maxn];

vector<int>G[maxn]; *//存边*

int dfs(int s){//找增广路

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

int t = G[s][i];

if(!vis[t]){

vis[t] = 1;

if(link[t] == -1 || dfs(link[t])){//找到增广路

link[t] = s;

return 1;

}

}

}

return 0;//未找到

}

int max\_pipei(int n){

int ans = 0;

for(int i=1; i<=n; i++){

memset(vis, 0, sizeof(vis));

ans+=dfs(i);

}

return ans;//匹配数

}

int main(){

int ld, rd; *//左半边的点数, 右半边的点数*

while(~scanf("%d%d", &ld, &rd)){

for(int i=0; i<maxn; i++)

G[i].clear();

memset(link, -1, sizeof(link));//匹配初始化

int m; *//边数*

scanf("%d", &m);

for(int i=0; i<m; i++){

int u, v;

scanf("%d%d", &u, &v);

*//无向边*

G[u].push\_back(v+ld);

G[v+ld].push\_back(u);

}

printf("%d\n", max\_pipei(ld));

}

return 0;

}

*//二分图匹配,匈牙利算法(BFS实现)*

#include <stdio.h>

#include <string.h>

#include <queue>

#include <algorithm>

#define maxn 10005

using namespace std;

vector<int>G[maxn]; *//存边*

int vis[maxn]; *//判断一个点是否在交替路中*

int link[maxn]; *//存连接点*

int pre[maxn]; *//存前一点*

queue<int>q;

int Hungarian(int n){

int ans = 0;

memset(vis, -1, sizeof(vis));

memset(link, -1, sizeof(link));

memset(pre, -1, sizeof(pre));

for(int i=1; i<=n; i++){

if(link[i]==-1){

while (!q.empty()) q.pop();

pre[i] = -1;

bool flag = false;

q.push(i);

while(!q.empty() && !flag){

int u = q.front();

for(int j=0; j<G[u].size() && !flag; j++){*//注意如果falg为真,(找到一个未匹配的点,就不必继续下去了)*

int v = G[u][j];

if(vis[v] != i){

vis[v] = i;

q.push(link[v]);

if(link[v]>=0){ *//在已匹配点中*

pre[link[v]] = u;

}

else{ *//在未匹配点中*

flag = true;

int d = u;

int e = v;

while(d != -1){ *//找到一个未匹配点, 不断的往回更新, 让他们重选下一个*

int t = link[d];

link[d] = e;

link[e] = d;

d = pre[d];

e = t;

}

}

}

}

q.pop();

}

if(link[i] != -1)

ans ++;

}

}

return ans;

}

int main(){

int ld, rd; *//左半边点数, 右半边点数*

while(~scanf("%d%d", &ld, &rd)){

for(int i=0; i<maxn; i++)

G[i].clear();

int m; *//边数*

scanf("%d", &m);

for(int i=0; i<m; i++){

*//编号1-ld,1-rd*

int u, v;

scanf("%d%d", &u, &v);

*//无向图,u和v可以是同一个数*

G[u].push\_back(v+ld);

G[v+ld].push\_back(u);

}

printf("最大匹配数是:%d\n",Hungarian(ld));

}

return 0;

}