



导航

- [首页](#)
- [社区主页](#)
- [当前事件](#)
- [最近更改](#)
- [随机页面](#)
- [使用帮助](#)
- [NOCOW地图](#)
- [新手试练场](#)

搜索

工具箱

- [链入页面](#)
- [链出更改](#)
- [特殊页面](#)
- [可打印版](#)
- [永久链接](#)

为防止广告，目前nocow只有登录用户能够创建新页面。如要创建页面请先[登录/注册](#)（新用户需要等待1个小时才能正常使用该功能）。

匈牙利算法

求最大匹配的一种显而易见的算法是：先找出全部匹配，然后保留匹配数最多的。但是这个算法的复杂度为边数的指数级函数。因此，需要寻求一种更加高效的算法。

增广路的定义(也称增广轨或交错轨)：若P是图G中一条连通两个未匹配顶点的路径，并且属M的边和不属M的边(即已匹配和待匹配的边)在P上交替出现，则称P为相对于M的一条增广路径。（M为一个匹配）

由增广路的定义可以推出下述三个结论：

1. P的路径长度必定为奇数，第一条边和最后一条边都不属于M。
2. P经过取反操作可以得到一个更大的匹配M'。
3. M为G的最大匹配当且仅当不存在相对于M的增广路径。

用增广路求最大匹配(称作匈牙利算法，匈牙利数学家Edmonds于1965年提出)

算法轮廓：

1. 置M为空
2. 找出一条增广路径P，通过取反操作获得更大的匹配M'代替M
3. 重复(2)操作直到找不出增广路径为止

程序清单：

```
const maxm=200; maxn=200;
var i,j,k,m,n,ans:longint;
    g:array[1..maxm,1..maxn]of boolean;
    y:array[1..maxn]of boolean;
    lk:array[1..maxn]of longint;

function find(x:longint):boolean;
var i:longint;
begin
    for i:=1 to n do
        if g[x,i] and (not y[i])
            then begin
                y[i]:=true;
                if (lk[i]=0)or find(lk[i])
                    then begin
                        lk[i]:=x;
                        exit(true);
                    end;
            end;
    exit(false);
end;

begin//main
    init//初始化
    for i:=1 to n do
        begin
            fillchar(y,sizeof(y),0);
            if find(i) then inc(ans);
        end;
    writeln(ans);
end.
```

```
#include<stdio.h>
#include<string.h>

bool g[201][201];
int n,m,ans;
bool b[201];
int link[201];

bool init()
{
    int _x,_y;
```

```

memset(g,0,sizeof(g));
memset(link,0,sizeof(link));
ans=0;
if (scanf ("%d%d",&n,&m)==EOF) return false;
for (int i=1;i<=n;i++)
{
    scanf ("%d",&_x);
    for (int j=0;j<_x;j++)
    {
        scanf ("%d",&_y);
        g[ i ][_y]=true;
    }
}
return true;

bool find(int a)
{
    for (int i=1;i<=m;i++)
    {
        if (g[a][ i ]==1&&!b[ i ])
        {
            b[ i ]=true;
            if (link[ i ]==0||find(link[ i ]))
            {
                link[ i ]=a;
                return true;
            }
        }
    }
    return false;
}

int main()
{
    while (init())
    {
        for (int i=1;i<=n;i++)
        {
            memset(b,0,sizeof(b));
            if (find(i)) ans++;
        }
        printf ("%d\n",ans);
    }
}

```

每次增广时间为 $O(E)$,最多进行 $O(V)$ 次迭代,时间复杂度为 $O(VE)$.



此页面已被浏览过10,540次。 本页面由NOCOW匿名用户58.49.51.35于2012年3月28日 (星期三) 21:49做出最后修改。 在cosechy@gmail.com、NOCOW匿名用户110.178.211.234、122.227.190.196和121.204.48.134和其他的工人基础上。 本站全部文字内容使用GNU Free Documentation License 1.2授权。



[隐私权政策](#)

[关于NOCOW](#)

[免责声明](#)

陕ICP备09005692号