二分图的判定

Author	炎炎龙虾
Created	@Jul 26, 2020 3:13 PM

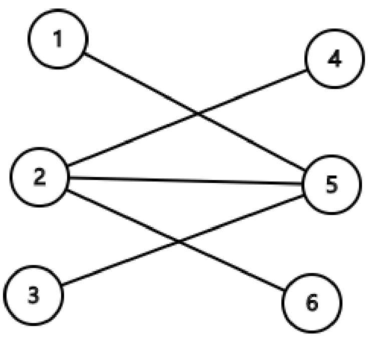
定义：

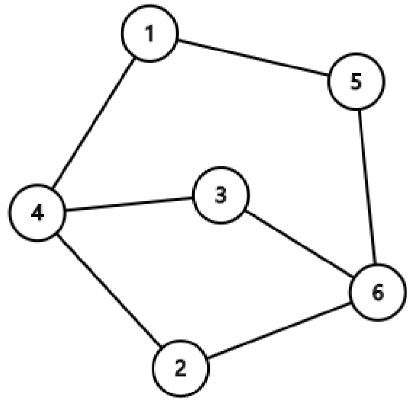
首先，什么是二分图？

百度百科上的定义是：“设$G=(V,E)$是一个无向图，如果顶点V可分割为两个互不相交的子集(A,B)，并且图中的每条边$(i，j)$所关联的两个顶点i和j分别属于这两个不同的顶点集$(i\in A,j\in B)$，则称图G为一个二分图。”

通俗点说，如果一个无向图所有的点可以分成两个部分，每个部分内部都没有边相连。

什么意思呢？举个例子吧。





上图就是一个典型的二分图，图可以分成两个部分，一个是$\{1,2,3\}$，另一个是$\{4,5,6\}$。这两个部分内部都没有边相连，$1,2$不连通，$2,3$不连通，$3,4$不连通，右半部分同理。

再举一个例子：

这就不是一个二分图，大家可以自行枚举，至于它为什么不是二分图，马上就讲。

定理：

一个无向图是二分图的**充分必要条件**就是该图内的每一个环的长度都是偶数。

①先证明必要性。假设有一个环C长度为s，包含s个节点$\{v_1,v_2,v_3,...,v_s\}$。可以将这个环单独剖离出来，看做一个待判定的图（若一个图子图不是二分图，则该图也一定不是二分图）。可以采用摇摆法，不妨设该环可以分为V_1和V_2（注意大小写）两个内部互不相交的点集，$v_1 \in V_1$。则必然$v_2 \in V_2$。归纳一下，得到：

$$v_1,v_3,v_5,v_7...v_{s-1} \in V_1$$
$$v_2,v_4,v_6,v_8...v_s \in V_2$$

由此可以发现，若该环是二分图，则$s-1$一定是奇数，s一定是偶数，因此定理中的必要性得证。

②可以采用摇摆法加反证法。为了简化原问题并不影响结果，我们可以假设图G是一个连通图。假设该图内每一个环的长度都是偶数，因此我们可以定义该图的两个子集分别为V_1和V_2，对于任意的$v \in V_1$，$V_1\{v|v_i \text{与} v \text{的距离为偶数}\}$，$V_2$同理。为了证明$V_1$内部没有边相连接，就可以采用反证法。假设$v_i,v_j \in V_1$，且存在一条边

(v_i, v_j)。由前面的假设可知，v与v_i之间的距离为偶数，v与v_j之间的距离也为偶数，而v_i与v_j之间的距离为1，因此，有v，v_i和v_j构成的环的长度就是奇数，与所给条件不符，因此假设不成立，因此必要性得证。

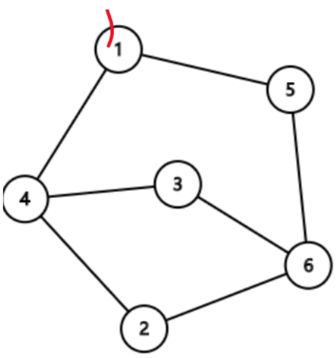
有了这个定理，再看看上面的那个图，会发现由$\{1,4,3,6,5\}$构成的环的长度为奇数，因此，那不是一个二分图。

好了，上面讲了这个定理，证明了这么久，有什么用吗？很抱歉地告诉你，并没有，但是你可以跟朋友秀一下“不打草稿肉眼判定二分图”，相信只会**染色法**的你的朋友一定会大吃一惊的。

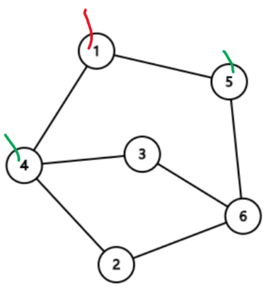
实现：

刚刚提到了染色法，染色法是什么？

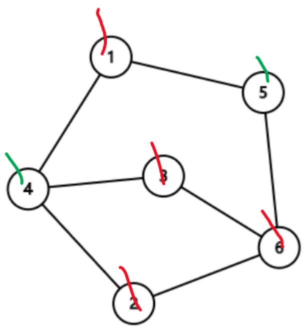
举个例子吧，可以自己体会一下。还是刚刚那个图：



从一号节点开始，染上红色



遍历所有与1相邻的节点，染上另一种颜色。



在遍历$4,5$，染色时发现出现了相邻的边染上了同种颜色的情况，因此判定它不是二分图。

由此，可知一个无向图是二分图的充要条件就是可二着色（可以用两种颜色染出来）。

还可以推广一下：一个无向图是n分图的充要条件就是可n着色（可以用n种颜色染出来）。

~~仿佛就是小学奥数里的染色问题。~~

UVA上有一道模板题：

Online Judge

Online Judge

https://onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=12&page=show_problem&problem=945

Bicoloring

problemUrl]: https://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=12&page=show_problem&problem=945 [PDF](https://uva.onlinejudge.org/external/100/p10004.pdf) ![]

https://www.luogu.com.cn/problem/UVA10004

下面直接附上代码吧，就是深搜染色：

```
//
// Created by admin on 2020/7/21.
//
#include <bits/stdc++.h>
using namespace std;
int n,m,color[300];
vector<int> no[300];
bool h;
bool dfs(int x,int c)
{
    color[x]=c;
    for(int i=0;i<no[x].size();i++)
    {
        int y=no[x][i];
        if(color[y]==c)//相邻边染上了同种颜色
            return false;
        if(color[y]==0&&!(dfs(y,-c)))//只要有一个点无法染色，全部返回false
            return false;
    }
    return true;
}
int main()
{
    while (true)
    {
        h=false;
        memset(no,0,sizeof(no));
        memset(color,0,sizeof(color));
        cin>>n;
```

```
        if(n==0)
            break;
        cin>>m;
        int x,y;
        for(int i=1;i<=m;i++)
        {
            cin>>x>>y;
            no[x].push_back(y);
            no[y].push_back(x);
        }
        for(int i=1;i<=n;i++)
            if(color[i]==0)//没有染色（原图不一定连通，一次染色不一定能染上全部）
            {
                if(!dfs(i,1))
                {
                    h=true;
                    break;
                }
            }
        if(h)
            cout<<"NOT BICOLORABLE."<<endl;
        else
            cout<<"BICOLORABLE."<<endl;
    }
    return 0;
}
```

UVA上还有一题，近似于模板：

Online Judge

Online Judge

https://onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=17&page=show_problem&problem=1446

Montesco vs Capuleto

problemUrl]: https://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=17&page=show_problem&problem=1446 [PDF](https://uva.onlinejudge.org/external/105/p10505.pdf) ![]

https://www.luogu.com.cn/problem/UVA10505

再附上代码：

```
//
// Created by admin on 2020/7/21.
//
#include <bits/stdc++.h>
using namespace std;
int n,m,color[4000],tot,colored;
vector<int> no[4000];
bool b;
void dfs(int x,int c)
{
    color[x]=c;
    tot++;
    if (c==1)
        colored++;
    for(int i=0;i<no[x].size();i++)
    {
        int y=no[x][i];
        if (color[y]==0)
            dfs(y,-c);
        else
            if(color[y]==c)
                b=false;
    }
}
int main()
{
    int t;
    cin>>t;
    while (t--)
    {
        int ans=0;

        memset(no,0,sizeof(no));
        memset(color,0,sizeof(color));
        cin>>n;
        int x;
        for (int j = 1; j <=n; ++j)
        {
            cin>>m;
            for(int i=1;i<=m;i++)
            {
                cin>>x;
                if(x>n)
                    continue;
                no[j].push_back(x);
                no[x].push_back(j);
            }
        }
        for(int i=1;i<=n;i++)
            if(color[i]==0)
            {
                tot=colored=0;
                b=true;
                dfs(i,1);
                if(b)
                    ans+=max(colored,tot-colored);
```

```
            }
        cout<<ans<<endl;
    }
    return 0;
}
```