ZJOI 矩阵游戏

题意

语言表达能力有限，直接截图放链接：

小 Q 是一个非常聪明的孩子，除了国际象棋，他还很喜欢玩一个电脑益智游戏—矩阵游戏。矩阵游戏在一个 $n \times n$ 黑白方阵进行（如同国际象棋一般，只是颜色是随意的）。每次可以对该矩阵进行两种操作：

- 行交换操作：选择矩阵的任意两行，交换这两行（即交换对应格子的颜色）。
- 列交换操作：选择矩阵的任意两列，交换这两列（即交换对应格子的颜色）。

游戏的目标，即通过若干次操作，使得方阵的主对角线(左上角到右下角的连线)上的格子均为黑色。

对于某些关卡，小 Q 百思不得其解，以致他开始怀疑这些关卡是不是根本就是无解的！于是小 Q 决定写一个程序来判断这些关卡是否有解。

输入格式

本题单测试点内有多组数据。

第一行包含一个整数 T，表示数据的组数，对于每组数据，输入格式如下：

第一行为一个整数，代表方阵的大小 n。 接下来 n 行，每行 n 个非零即一的整数，代表该方阵。其中 0 表示白色，1 表示黑色。

输出格式

对于每组数据，输出一行一个字符串，若关卡有解则输出 `Yes`，否则输出 `No`。

链接：zjoi矩阵游戏

放这题有什么意义呢

~~当然是因为实在没啥好放的子~~，这是非常简单的二分图和网络流建模的问题，适合刚学二分图批配或者网络流的新手，~~比如我~~初步体会一下建模的艺术。

题解

(建立在已经掌握一些模板算法的基础上。) 这题想让我们干啥呢？给定一个方阵，我们要通过若干次交换行和列使其对角元上都有1个1(即(i,i)位置)，我们不妨从结果考虑，如果成立,则不看对角元素以外的元素，必有对角元素上都有1，我们任意做交换操作，都满足，任意一个行都唯一的与一个列对应，即对于每一行都存在一列有1，且1的位置互不相同。而交换是可逆的，所以只要满足上述条件就一定会成立。这种模型很容易想到二分图批配问题，即将行看作左列，列看作右列，作二分图批配，若能完全匹配，则成立。

二分匹配的两种思路

1

匈牙利算法：本质上是一个递归找对应得方法，基于一些比较复杂得图论得定理，这里不做详细描述(其实我不会)，在本题中，如果一次找批配失败找不到则宣告失败，应为注定无法完全匹配。

2

网络流：建立虚点，源点和汇点，汇点链接右端点，源点链接左端点，跑ek或者$dinic$最大流。由于$dinic$最大流得特点，这里分层最多也只能分4层，而且还有一些玄学原因，所以这里跑$dinic$算法复杂度是相当优秀的,似乎是$O(\sqrt n m)$远远高于普通最大流和匈牙利算法。**dinic果然赛高！！！**

代码

写这题时还不会网络流gg……所以这里放一个匈牙利算法的板子。

```
#include <bits/stdc++.h>
using namespace std;
const int maxn=2000;
bool vis[maxn];
vector<int> p[maxn];
int mth[maxn];
int n,m,e;
bool dfs(int now)
{
    int len=p[now].size();
    for(int i=0;i<len;i++)
    {
        int v=p[now][i];
        if(vis[v]) continue;//如果是被走过的点就直接找下一个
        vis[v]=1;
        if(!mth[v]||dfs(mth[v]))//先判断是否已经被批配，然后判断若换匹配能否成立
        {
            mth[v]=now;
            return 1;
         }
    }
    return 0;
}
void ini()
{
    for(int i=1;i<=2*n;i++) vis[i]=0;
}
int main()
```

```
{
    int t,ans;
    scanf("%d",&t);
    while(t--)
    {
        int x,yes=1;
    scanf("%d",&n);
    for(int i=1;i<=2*n;i++) mth[i]=0,p[i].clear();
    for(int i=1;i<=n;i++)
    {
        for(int j=1;j<=n;j++)
        {
            scanf("%d",&x);
            if(x) p[i].push_back(j+n);
        }
    }
    for(int i=1;i<=n;i++)
    {
        ini();
        if(!dfs(i))
        {
          yes=0;
         break;
        }
    }
    if(yes) printf("Yes\n");
    else printf("No\n");
    }
}
```