# DLX（Dancing Links X 舞蹈链）

## 简介

求解精准覆盖问题：给定一个集合和若干个子集，选择若干个子集使得原集合中每一个元素出现且仅出现一次。

## 思路

用矩阵的方式存储子集：每一行代表一个子集，每一列代表原集合中的一个元素，某行某列用0和1代表这一个子集是否含有这一个元素。

1. 对于一个存在1的列找一个行
2. 删除该行、行上点的所在列、包含行上点所在列的行$\to$1.
3. 余下列但无可用行$\to$回溯
4. 不余——输出答案/储存答案

### 实现方法

利用十字循环链表。

#### 初始化

```
void init()
{
    cnt=m;
    for(int i=0;i<=m;i++)
        node[i]={i,i,i-1,i+1,0,i};
    node[m].d=0;
    node[0].a=m;
}
```

#### 添加点

```
for(int i=1;i<=n;i++)
    for(int j=1,a;j<=m;j++)
    {
        scanf("%d",&a);
        if(a)
        {
            tot[j]++;
            node[++cnt]={node[j].w,j,0,0,i,j};
            node[node[j].w].s=cnt;
            node[j].w=cnt;
```

```
            if(!left[i])
                left[i]=node[cnt].a=node[cnt].d=cnt;
            else
            {
                node[cnt].d=left[i];
                node[cnt].a=node[left[i]].a;
                node[node[left[i]].a].d=cnt;
                node[left[i]].a=cnt;
            }
        }
    }
```

### 删除列

```
void remove(int x)
{
    node[node[x].a].d=node[x].d;
    node[node[x].d].a=node[x].a;
    for(int i=node[x].s;i!=x;i=node[i].s)
        for(int j=node[i].d;j!=i;j=node[j].d)
        {
            node[node[j].w].s=node[j].s;
            node[node[j].s].w=node[j].w;
            tot[node[j].c]--;
        }
}
```

### 恢复列

```
void recover(int x)
{
    for(int i=node[x].w;i!=x;i=node[i].w)
        for(int j=node[i].d;j!=i;j=node[j].d)
        {
            node[node[j].w].s=node[node[j].s].w=j;
            tot[node[j].c]++;
        }
    node[node[x].a].d=node[node[x].d].a=x;
}
```

### X主体

```
bool dlx(int dep)
{
    if(!node[0].d)
```

```
    {
        for(int i=1;i<dep;i++)
            printf("%d ",ans[i]);
        return true;
    }
    int C=node[0].d;
    for(int i=node[C].d;i;i=node[i].d)
        if(tot[i]<tot[C])
            C=i;
    if(node[C].s==C)
        return false;
    remove(C);
    for(int i=node[C].s;i!=C;i=node[i].s)
    {
        ans[dep]=node[i].r;
        for(int j=node[i].a;j!=i;j=node[j].a)
            remove(node[j].c);
        if(dlx(dep+1))
            return true;
        for(int j=node[i].d;j!=i;j=node[j].d)
            recover(node[j].c);
    }
    recover(C);
    return false;
}
```

注：恢复列时要从删除列的最后一个开始恢复。

# 例题

### 【模板】舞蹈链（DLX）

**来源**：洛谷p4929

**概述**：模板

**答案**：模板

```
#include<cstdio>
using namespace std;
int n,m,cnt,left[505],ans[505],tot[505];
struct unit{int w,s,a,d,r,c;}node[6000];
void init()
{
    cnt=m;
    for(int i=0;i<=m;i++)
        node[i]={i,i,i-1,i+1,0,i};
    node[m].d=0;
```

```
    node[0].a=m;
}
void remove(int x)
{
    node[node[x].a].d=node[x].d;
    node[node[x].d].a=node[x].a;
    for(int i=node[x].s;i!=x;i=node[i].s)
        for(int j=node[i].d;j!=i;j=node[j].d)
        {
            node[node[j].w].s=node[j].s;
            node[node[j].s].w=node[j].w;
            tot[node[j].c]--;
        }
}
void recover(int x)
{
    for(int i=node[x].w;i!=x;i=node[i].w)
        for(int j=node[i].d;j!=i;j=node[j].d)
        {
            node[node[j].w].s=node[node[j].s].w=j;
            tot[node[j].c]++;
        }
    node[node[x].a].d=node[node[x].d].a=x;
}
bool dlx(int dep)
{
    if(!node[0].d)
    {
        for(int i=1;i<dep;i++)
            printf("%d ",ans[i]);
        return true;
    }
    int C=node[0].d;
    for(int i=node[C].d;i;i=node[i].d)
        if(tot[i]<tot[C])
            C=i;
    if(node[C].s==C)
        return false;
    remove(C);
    for(int i=node[C].s;i!=C;i=node[i].s)
    {
        ans[dep]=node[i].r;
        for(int j=node[i].a;j!=i;j=node[j].a)
            remove(node[j].c);
        if(dlx(dep+1))
            return true;
        for(int j=node[i].d;j!=i;j=node[j].d)
            recover(node[j].c);
    }
    recover(C);
```

```
    return false;
}
int main()
{
    freopen("input.in","r",stdin);
    //freopen("output.out","w",stdout);
    scanf("%d%d",&n,&m);
    init();
    for(int i=1;i<=n;i++)
        for(int j=1,a;j<=m;j++)
        {
            scanf("%d",&a);
            if(a)
            {
                tot[j]++;
                node[++cnt]={node[j].w,j,0,0,i,j};
                node[node[j].w].s=cnt;
                node[j].w=cnt;
                if(!left[i])
                    left[i]=node[cnt].a=node[cnt].d=cnt;
                else
                {
                    node[cnt].d=left[i];
                    node[cnt].a=node[left[i]].a;
                    node[node[left[i]].a].d=cnt;
                    node[left[i]].a=cnt;
                }
            }
        }
    if(!dlx(1))
        printf("No Solution!");
    return 0;
}
```

# 总结