位运算与状压DP

BY GY

位运算

BY GY

补码

- 正数的补码与原码相同
- 负数的补码为源码除符号位外取反然后加一
- 如-5的源码（八位二进制）为1000 0101
- 除符号位外取反：1111 1010
- 加一：1111 1011

按位与（&）

- 参加运算的两个数，换算为二进制后，只有当相应位上的数都是1时，该位才取1，否则该为为0
- 例：10&6=2
- 10：　　1010
- 6：　　 0110
- 2：　　 0010

按位或（|）

- 参加运算的两个数，换算为二进制后，只要相应位上存在1，那么该位就取1，均不为1，即为0。
- 例：10|6=14
- 10： 1010
- 6：   0110
- 14： 1110

按位异或（^）

- 参加运算的两个数，换算为二进制后，只有当相应位上的数字不相同时，该为才取1，若相同，即为0。
- 例：10^6=12
- 10：　1010
- 6：　 0110
- 12：　1100

左移（<<）

- a<<b
- 将a化为二进制后各二进制位向左移动b位,空位补0
- 例：9<<2=36
- 9:   1001
- 36：100100

右移（>>）

- a>>b
- 将a化为二进制后各二进制位向右移动b位,最低位继续向右移时直接删去
- 例：9>>2=2
- 9: 001001
- 2: 0010

按位取反（~）

- 参加运算的一个数，换算为二进制后，每个位上都取相反值，1变成0，0变成1。
- 例：~10=5
- 10：1010
- 5： 0101
- 注：取反运算结果与二进制位数有关，上边是在四位二进制的情况下的结果，如果是八位二进制结果应为：
- 1111 0101

位运算应用

- 1. <<1相当于X2 >>1相当于/2
- 2.判断奇数还是偶数：&1=1为奇数，&1=0为偶数
- 3.只保留最低位的1：x&(-x)
- 4.去除最右侧的1：x&（x-1）
- 可以利用3或4判断有多少个1
- 5.两个数字交换a=a^b;
- b=b^a;
- a=b^a;

位运算应用

功能	示例	位运算
去掉最后一位	(101101->10110)	x >> 1
在最后加一个0	(101101->1011010)	x << 1
在最后加一个1	(101101->1011011)	x << 1+1
把最后一位变成1	(101100->101101)	x | 1
把最后一位变成0	(101101->101100)	x | 1-1
最后一位取反	(101101->101100)	x ^ 1
把右数第k位变成1	(101001->101101,k=3)	x | (1 << (k-1))
把右数第k位变成0	(101101->101001,k=3)	x & ~ (1 << (k-1))
右数第k位取反	(101001->101101,k=3)	x ^ (1 << (k-1))
取末三位	(1101101->101)	x & 7
取末k位	(1101101->1101,k=5)	x & (1 << k-1)
取右数第k位	(1101101->1,k=4)	x >> (k-1) & 1
把末k位变成1	(101001->101111,k=4)	x | (1 << k-1)
末k位取反	(101001->100110,k=4)	x ^ (1 << k-1)
把右边连续的1变成0	(100101111->100100000)	x & (x+1)
把右起第一个0变成1	(100101111->100111111)	x | (x+1)
把右边连续的0变成1	(11011000->11011111)	x | (x-1)
取右边连续的1	(100101111->1111)	(x ^ (x+1)) >> 1
去掉右起第一个1的左边	(100101000->1000)	x & (x ^ (x-1))

状压DP

BY GY

状压DP

- 状态压缩动态规划
- 将一个状态压缩为一个二进制数
- 其余与正常DP相同
- 复杂度一般为O（2^n）
- 适用于数据范围较小的题目（一般为个位数或十几，20几乎就是极限）
- 也可以通过预先筛去不符合条件的情况降低复杂度

例1：洛谷p1879玉米田

农场主John新买了一块长方形的新牧场，这块牧场被划分成M行N列(1 ≤ M ≤ 12; 1 ≤ N ≤ 12)，每一格都是一块正方形的土地。John打算在牧场上的某几格里种上美味的草，供他的奶牛们享用。

遗憾的是，有些土地相当贫瘠，不能用来种草。并且，奶牛们喜欢独占一块草地的感觉，于是John不会选择两块相邻的土地，也就是说，没有哪两块草地有公共边。

John想知道，如果不考虑草地的总块数，那么，一共有多少种种植方案可供他选择？（当然，把新牧场完全荒废也是一种方案）

输入输出格式

输入格式：

第一行：两个整数M和N，用空格隔开。

第2到第M+1行：每行包含N个用空格隔开的整数，描述了每块土地的状态。第i+1行描述了第i行的土地，所有整数均为0或1，是1的话，表示这块土地足够肥沃，0则表示这块土地不适合种草。

输出格式：

一个整数，即牧场分配总方案数除以100,000,000的余数。

玉米田

- 数据范围小，考虑状压DP
- 每一块田只有是否能种草两种情况，可以用二进制01来表示，可以用二进制储存每一排的状态
- 下一排如何选择只与当前排和下一排的土地情况有关
- 满足无后效性
- 可以使用状压DP

玉米田

- 先记录草地情况
- 因为判断当前状
- 态是否和土地情
- 况匹配与判断是
- 否和上一排冲突
- 使用的是同一个
- 函数（均为1说明冲突），所以1代表贫瘠，0代表肥沃

```
for(int i=1;i<=n;i++){
    for(int j=1;j<=m;j++){
        int x;
        scanf("%d",&x);
        a[i]=(a[i]<<1)|(x^1);
    }
}
```

玉米田

- 草地不能有相邻的（上下左右），所以可以先将左右相邻的情况提前筛去

```
for(int i=0;i<(1<<m);i++){
    if(!judge1(i))
    can[++cnt]=i;
}
```

```
bool judge1(int x){//是否有相邻的
    return x&(x<<1);
}
```

玉米田

- DP部分我采用记忆化搜索实现，可以尝试改为递推
- 最终结果就是f[1][0]

```
int dfs(int now,int last){
    if(now==n+1) return 1;
    if(f[now][last]>-1) return f[now][last];
    int ans=0;
    for(int i=1;i<=cnt;i++){
        if(!judge2(can[i],last)&&!judge2(can[i],a[now]))
        ans=(ans+dfs(now+1,can[i]))%p;
    }
    return f[now][last]=ans;
}
```

例2：洛谷p1171售货员的难题

题目描述

某乡有n个村庄($1 < n \le 20$)，有一个售货员，他要到各个村庄去售货，各村庄之间的路程$s(0 < s < 1000)$是已知的，且A村到B村与B村到A村的路大多不同。为了提高效率，他从商店出发到每个村庄一次，然后返回商店所在的村，假设商店所在的村庄为1，他不知道选择什么样的路线才能使所走的路程最短。请你帮他选择一条最短的路。

输入输出格式

输入格式：

村庄数n和各村之间的路程(均是整数)。

输出格式：

最短的路程。

售货员的难题

- n<=20 状压的极限情况，可以考虑
- 需要记录的状态有已经经过了那些村庄和当前在哪个村庄，村庄只有经过和没经过两种情况，可以用二进制表示
- 村庄数量较少，之间的距离使用邻接矩阵储存即可

售货员的难题

- 因为求最小值
- 先将DP数组初
- 始化为极大值
- 数组第一维表
- 示经过村庄的
- 情况，第二维
- 表示当前在哪个村庄，转移类似于最短路的松弛操作

```
memset(f,0x3f,sizeof(f));
f[1][1]=0;
for(int s=1;s<=(1<<n)-1;s++){
    for(int v=1;v<=n;v++){
        if(s&(1<<(v-1))) continue;
        for(int k=1;k<=n;k++){
            if(s&(1<<(k-1)))
            f[s|(1<<(v-1))][v]=min(f[s|(1<<(v-1))][v],f[s][k]+a[k][v]);
        }
    }
}
```

售货员的难题

- 因为还要返回，所以有如下操作

```
for(int i=1;i<=n;i++){
    ans=min(ans,f[(1<<n)-1][i]+a[i][1]);
}
```

- 需要注意的地方：数据很极限，如果代码写的丑，常数大跑不过去吸口氧即可

售货员的难题

- 还有一种递归实现的方法，更好理解，但常数更大，吸氧也跑不过去,放在这里仅供参考

```
void dfs1(int s,int u){
    if(s==(1<<n)-1) return;
    for(int v=1;v<=n;v++){
        if((1<<(v-1))&s) continue;
        if(f[s|(1<<(v-1))][v]>f[s][u]+a[u][v]){
            f[s|(1<<(v-1))][v]=f[s][u]+a[u][v];
            dfs1(s|(1<<(v-1)),v);
        }
    }
}
```

```
for(int i=1;i<=n;i++){
    ans=min(ans,f[(1<<n)-1][i]+a[i][1]);
}
```

P2831愤怒的小鸟

题目描述

`Kiana` 最近沉迷于一款神奇的游戏无法自拔。

简单来说，这款游戏是在一个平面上进行的。

有一架弹弓位于 $(0,0)$ 处，每次 `Kiana` 可以用它向第一象限发射一只红色的小鸟，小鸟们的飞行轨迹均为形如 $y=ax^2+bx$ 的曲线，其中 a,b 是 `Kiana` 指定的参数，且必须满足 $a<0$，a,b 都是实数。

当小鸟落回地面（即 x 轴）时，它就会瞬间消失。

在游戏的某个关卡里，平面的第一象限中有 n 只绿色的小猪，其中第 i 只小猪所在的坐标为 (x_i,y_i)。

如果某只小鸟的飞行轨迹经过了 (x_i,y_i)，那么第 i 只小猪就会被消灭掉，同时小鸟将会沿着原先的轨迹继续飞行；

如果一只小鸟的飞行轨迹没有经过 (x_i,y_i)，那么这只小鸟飞行的全过程就不会对第 i 只小猪产生任何影响。

例如，若两只小猪分别位于 $(1,3)$ 和 $(3,3)$，`Kiana` 可以选择发射一只飞行轨迹为 $y=-x^2+4x$ 的小鸟，这样两只小猪就会被这只小鸟一起消灭。

而这个游戏的目的，就是通过发射小鸟消灭所有的小猪。

这款神奇游戏的每个关卡对 `Kiana` 来说都很难，所以 `Kiana` 还输入了一些神秘的指令，使得自己能更轻松地完成这个游戏。这些指令将在【输入格式】中详述。

假设这款游戏一共有 T 个关卡，现在 `Kiana` 想知道，对于每一个关卡，至少需要发射多少只小鸟才能消灭所有的小猪。由于她不会算，所以希望由你告诉她。

愤怒的小鸟

输入输出格式

输入格式:

第一行包含一个正整数 T，表示游戏的关卡总数。

下面依次输入这 T 个关卡的信息。每个关卡第一行包含两个非负整数 n, m，分别表示该关卡中的小猪数量和 Kiana 输入的神秘指令类型。接下来的 n 行中，第 i 行包含两个正实数 x_i, y_i，表示第 i 只小猪坐标为 (x_i, y_i)。数据保证同一个关卡中不存在两只坐标完全相同的小猪。

如果 $m = 0$，表示 `Kiana` 输入了一个没有任何作用的指令。

如果 $m = 1$，则这个关卡将会满足：至多用 $\lceil n/3 + 1 \rceil$ 只小鸟即可消灭所有小猪。

如果 $m = 2$，则这个关卡将会满足：一定存在一种最优解，其中有一只小鸟消灭了至少 $\lfloor n/3 \rfloor$ 只小猪。

保证 $1 \leq n \leq 18$，$0 \leq m \leq 2$，$0 < x_i, y_i < 10$，输入中的实数均保留到小数点后两位。

上文中，符号 $\lceil c \rceil$ 和 $\lfloor c \rfloor$ 分别表示对 c 向上取整和向下取整，例如：$\lceil 2.1 \rceil = \lceil 2.9 \rceil = \lceil 3.0 \rceil = \lfloor 3.0 \rfloor = \lfloor 3.1 \rfloor = \lfloor 3.9 \rfloor = 3$。

输出格式:

对每个关卡依次输出一行答案。

输出的每一行包含一个正整数，表示相应的关卡中，消灭所有小猪最少需要的小鸟数量。

愤怒的小鸟

- 题干极长，堪比阅读题
- 一眼看上去没什么思路
- 给的几个特殊约束到现在我也不知道该怎么用
- 个人感觉是藏得比较深的一道状压DP
- 但是题目是给了提示的
- 就是数据范围

愤怒的小鸟

<table>
<tr><th>测试点编号</th><th>n</th><th>m</th><th>T</th></tr>
<tr><td>1</td><td rowspan="2">≤ 2</td><td rowspan="12">$= 0$</td><td>≤ 10</td></tr>
<tr><td>2</td><td>≤ 30</td></tr>
<tr><td>3</td><td rowspan="2">≤ 3</td><td>≤ 10</td></tr>
<tr><td>4</td><td>≤ 30</td></tr>
<tr><td>5</td><td rowspan="2">≤ 4</td><td>≤ 10</td></tr>
<tr><td>6</td><td>≤ 30</td></tr>
<tr><td>7</td><td>≤ 5</td><td rowspan="4">≤ 10</td></tr>
<tr><td>8</td><td>≤ 6</td></tr>
<tr><td>9</td><td>≤ 7</td></tr>
<tr><td>10</td><td>≤ 8</td></tr>
<tr><td>11</td><td>≤ 9</td><td rowspan="4">≤ 30</td></tr>
<tr><td>12</td><td>≤ 10</td></tr>
<tr><td>13</td><td rowspan="2">≤ 12</td><td>$= 1$</td></tr>
<tr><td>14</td><td>$= 2$</td></tr>
<tr><td>15</td><td rowspan="3">≤ 15</td><td>$= 0$</td><td rowspan="3">≤ 15</td></tr>
<tr><td>16</td><td>$= 1$</td></tr>
<tr><td>17</td><td>$= 2$</td></tr>
<tr><td>18</td><td rowspan="3">≤ 18</td><td>$= 0$</td><td rowspan="3">≤ 5</td></tr>
<tr><td>19</td><td>$= 1$</td></tr>
<tr><td>20</td><td>$= 2$</td></tr>
</table>

愤怒的小鸟

- 能够发现n最大为18，数据较小，可以考虑状压DP
- 利用二进制数记录猪的死活
- 小鸟飞的抛物线，而三个点可以唯一确定一条抛物线
- 所以我们可以用出发点（原点）和任意两头猪确定一条抛物线，同时判断有多少猪在这条抛物线上
- 利用二进制记录下这条抛物线可以打死哪些猪
- 再使用DP求出最优解

愤怒的小鸟

- 但是抛物线必须
- 是开口向下的，
- 所以会出现一次
- 发射只能杀死一
- 头猪的情况，这
- 种情况也需要记
- 录下来。

```
for(int i=1;i<=n;i++)
    can[++cnt]=1<<(i-1);
for(int i=1;i<=n;i++){
    for(int j=i+1;j<=n;j++){
        if(equal(x[i],x[j])) continue;
        double a=(x[j]*y[i]-x[i]*y[j])/(x[i]*x[j]*(x[i]-x[j]));
        if(a>=0) continue;
        double b=(x[i]*x[i]*y[j]-x[j]*x[j]*y[i])/(x[i]*x[j]*(x[i]-x[j]));
        int s=0;
        for(int k=1;k<=n;k++){
            if(check(x[k],y[k],a,b))
                s|=1<<(k-1);
        }
        if(!vis[s]){
            vis[s]=1;
            can[++cnt]=s;
        }
    }
}
```

愤怒的小鸟

```
#define eps 1e-6
```

```
bool equal(double a,double b){
    double abs=a-b;
    if(abs<0) abs=-abs;
    return abs<eps;
}
bool check(double x,double y,double a,double b){
    double yy=a*x*x+b*x;
    return equal(y,yy);
}
```

愤怒的小鸟

- DP部分很简单
- 从下一状态已
- 发现的最有情
- 况和当前状态
- 发射小鸟转移
- 过去中选一个
- 更优的。

```
memset(f,0x3f,sizeof(f));
f[0]=0;
for(int i=0;i<(1<<n);i++){
    for(int j=1;j<=cnt;j++){
        f[i|can[j]]=min(f[i|can[j]],f[i]+1);
    }
}
printf("%d\n",f[(1<<n)-1]);
```

- 需要注意的是这道题是多组数据，不要忘记初始化

课后练习

- 简单题：p1896 p2704
- 稍微难一点的：p2915
- 难题：p3959