知识精炼（三）

主讲人：邓哲也

NOIP 2016 Day2 愤怒的小鸟

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$a<0$。

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)。

NOIP 2016 Day2 愤怒的小鸟

如果某只小鸟的飞行轨迹经过了(xi,yi),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(xi,yi)，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。

小猪的总数 ≤ 18

NOIP 2016 Day2 愤怒的小鸟

数据范围非常小，可以考虑状态压缩DP。

设 f[s] 表示清除掉 s 集合中的猪花费的最小步数。

思考转移。

在已有的 s 集合基础上，再选择一条抛物线使得它经过 t 集合的点。

那么就可以用 f[s] + 1 去更新 f[s | t]

NOIP 2016 Day2 愤怒的小鸟

三点确定一条抛物线。

而三点之中必须有一个原点，因此只要两个点就能确定一条抛物线。

因此我们可以枚举 s 集合以外的任意两个点，算出经过这两个点的抛物线，枚举所有的点看是否落在抛物线上，得到抛物线经过的点集 t。

```
f[s | t] = min(f[s | t], f[s] + 1);
```

NOIP 2016 Day2 愤怒的小鸟

经过点 i 和 j 的抛物线经过的点集 t[i][j] 可以预处理。

时间复杂度 $O(n^3)$

之后 DP 枚举每个集合，对每个集合都要枚举两个点。

时间复杂度 $O(2^n n^2)$

NOIP 2016 Day2 愤怒的小鸟

预处理：

$x[i]^2a + x[i]b = y[i]$

$x[j]^2a + x[j]b = y[j]$

解二元一次方程组，得到 a 和 b。

如果 a >= 0，不符合题意，t[i][j] = 0

否则对每个点判断一下是否落在这条抛物线上，如果第 k 个点落在抛物线上，t[i][j] |= (1 << (k - 1));

NOIP 2016 Day2 愤怒的小鸟

```
memset(f, 63, sizeof(f));
f[0] = 0;
for (int s = 0; s < (1 << n); s ++) {
    if (f[s] == 0x3f3f3f3f) continue;
    for (int i = 0;i < n;j ++)
        if (!(s >> i & 1))
            for(int j = 0;j < n;j ++)
                if (!(s >> j & 1))
                    f[s | t[i][j]] = min(f[s | t[i][j]], f[s]
                    + 1);
}
```

f[(1 << n) - 1] 就是答案。

下节课再见