Chomp 游戏，D. Gale, *A Curious Nim-type game*, Amer. Math. Monthly **81** (1974)，不再赘述。

Chomp 游戏研究

二行 Chomp 游戏

令 (a,b)（$a\geq b\geq 0$ 且 $a>0$）表示一个第一行有 a 列，第二行有 b 列的 Chomp 游戏。

有定理显示：$(a,a-1)$ 是 $\mathcal{P}$（必败态）而任意的 (a,b)（$b\neq a-1$）都是 $\mathcal{N}$（必胜态）。

应用数学归纳法不难证明。

三行 Chomp 游戏

令 (a,b,c) 表示一个第一行有 a 列，第二行有 b 列，第三行有 c 列的 Chomp 游戏。

一些 c 比较小的情况，手玩

当 $c=0$ 时，我们通过二行 Chomp 游戏，已知这个新游戏的一些情况。

于是我们考虑研究 $c=1$ 的情况。

我们可以发现 $(b+1,b,1)$ 均是 $\mathcal{N}$ 态，这是因为它们都能转移到 $(b+1,b,0)$ 这个 $\mathcal{P}$ 态。

并且 $(1,1,1)$ 当然也是 $\mathcal{N}$ 态，那么 $c=1$ 时的 $\mathcal{P}$ 态到底在哪里呢?

我们注意到 $(2,2,1)$ 和 $(3,1,1)$ 实际上是 $\mathcal{P}$ 态，读者可以自行验证。

而这实际上宣判了所有的 a,b「比它们大」的状态都是 $\mathcal{N}$ 态:

- 对于 $(a,b,1)$，只要 $a\geq 3$ 并且 $b\geq 2$ 就能转移到 $(2,2,1)$。
- 而如果 $a\geq 4$ 并且 $b=1$ 就能转移到 $(3,1,1)$。
- 这些便覆盖了所有的 $a\geq b\geq 1$ 的状态。

结论：当 $c=1$ 时，游戏 (a,b,c) 唯二的 $\mathcal{P}$ 态就是 $(2,2,1)$ 和 $(3,1,1)$。

接下来研究 $c=2$ 的情况。

同样地，有 $(b+1,b,2)$ 均是 $\mathcal{N}$ 态，因为都能转移到 $(b+1,b,0)$ 这个 $\mathcal{P}$ 态。

而 $(2,2,2)$ 也是 $\mathcal{N}$ 态。并且 $(3,3,2)$ 因为可以转移到 $(3,1,1)$ 这个 $\mathcal{P}$ 态，它也是 $\mathcal{N}$ 态。

考虑 $(4,2,2)$，不难验证它无法转移到任何 $\mathcal{P}$ 态，所以它也是 $\mathcal{P}$ 态。

这便意味着 $(4,3,2),(4,4,2)$ 都是 $\mathcal{N}$ 态，以及所有的 $(4+\alpha,2,2)$（$\alpha\geq 1$）也都是 $\mathcal{N}$ 态。

注意 $(a,b,2)$ 当 $a\geq b\geq 3$ 时转移到 $(*,*,1)$ 时均无法得到 $\mathcal{P}$ 态，所以仅需考虑转移到 $(*,*,2)$ 的 $\mathcal{P}$ 态的情况。

那么考虑目前唯一的 $\mathcal{P}$ 态 $(4,2,2)$，不能转移到它的「最小」的状态是 $(5,3,2)$。

所以 $(5,3,2)$ 当然是 $\mathcal{P}$ 态，而不能转移到它的「最小」的状态是 $(6,4,2)$。

由 $(6,4,2)$ 又能推出下一个 $\mathcal{P}$ 态是 $(7,5,2)$，以此类推。

我们可以猜想：$(a,b,2)$ 是 $\mathcal{P}$ 态当且仅当 $a-b=2$。

这个猜想也可以由数学归纳法证明，不过比 $c=0$ 时的情况复杂一些。

结论：当 $c=2$ 时，游戏 $(a,b,2)$ 是 $\mathcal{P}$ 态当且仅当 $a-b=2$。

对于 $c=3$ 的情况，我们有唯三的三个 $\mathcal{P}$ 态 $(5,5,3),(6,3,3),(7,4,3)$，具体推导过程在此不详述。

结论：当 $c=3$ 时，游戏 (a,b,c) 唯三的 $\mathcal{P}$ 态是 $(5,5,3),(6,3,3),(7,4,3)$。

手玩太麻烦了，不是吗？

使用计算机帮助寻找

对于 $c=0\sim 408$ 的情况，我们可以打表获得 c 值恒定时的所有 $\mathcal{P}$ 态的规律。

当然，c 还可以更大，但是这里我只打到了 $c\le 408$ 的情况。

可惜的是，对于不同的 c 值的变化，呈现出的规律似乎是无序的，暂时并无法从中总结出任何模式。

所以我们仅能对 c 在一定范围内的情况给出这些规律，详见 `table1.txt`。源程序详见 `code3.cpp`。

需要特别注意的是，$c=120$ 时它的规律与其他情况不同，有着长为 2 的周期，而不像其他情况仅仅是一直重复，这是很有趣的。

这个 $c=120$ 是**第一个反例**，接下来还有两个反例是 $c=400$ 和 $c=402$。

更加特别的是，$c=400$ 的周期和 $c=120$ 是一样的，都为 2，但是 $c=402$ 的周期为 4。

这两个新的反例靠得那么近是有原因的，实际上是 $c=400$ 直接影响了 $c=402$ 的情况，并且把它的循环周期变成了 4。

上述打表程序的时间复杂度是 $\mathcal{O}(k^4)$ 的，其中 k 指的是 c 的最大值，即求出所有 $c\in[0,k]$ 之间的答案的复杂度。

接下来需要研究的是能否在 $\mathcal{O}(k^3)$ 甚至 $\mathcal{O}(k^2)$ 的时间内求出这些信息（应该做不到低于 $\mathcal{O}(k^2)$ 的复杂度，因为信息量有那么多）。

我们先展示打出的表中对于 c 在 $0\sim 10$ 中的情况，其中 `c: (a,b)` 表示 (a,b,c) 是一个 $\mathcal{P}$ 态：

```
0: pat(1+k,0+k)...
1: (3,1),(2,2)
2: pat(4+k,2+k)...
3: (6,3),(7,4),(5,5)
4: (8,4),(9,5),(10,6),(7,7)
5: (10,5),(9,6),pat(11+k,7+k)...
6: (11,6),(12,7),(13,8),(9,9)
7: (13,7),(14,8),(12,9),pat(15+k,10+k)...
8: (15,8),(14,9),(16,10),(17,11),(12,12)
9: (16,9),(17,10),(14,11),pat(18+k,12+k)...
10: (18,10),(19,11),(20,12),(21,13),(14,14)
```

其中 `pat(a+k,b+k)...` 表示的是形如 $(a+k,b+k,c)$（$k\in\mathbb{N}$）的情形，包含无限种情况。

我们考虑对于一个 (a,b,c) 如何根据已知信息去推断它是否为 $\mathcal{P}$ 态。

换言之，也就是寻找一个已知的 $\mathcal{P}$ 态 $\mathbf{P}$ 使得 $\mathbf{X}=(a,b,c)$ 能转移到 $\mathbf{P}$。

如果找不到这样的 $\mathbf{P}$，则 $\mathbf{X}$ 即为 $\mathcal{P}$ 态，否则为 $\mathcal{N}$ 态。

先考虑 $\mathbf{P}$ 的 c 不为当前的 c 的情况，简记为 $c(\mathbf{P})<c(\mathbf{X})$：

- 此时想要 $\mathbf{X}$ 能转移到 $\mathbf{P}$，必须有选取的那个位置 (i,j) 满足 $j=c(\mathbf{P})+1$，而 i 可以等于 $1,2,3$ 中的任意一个。
- 如果 $i=3$，则即是 $\mathbf{X}=(a,b,c)$ 而 $\mathbf{P}=(a,b,c_2)$，仅有第三行的长度不同。
  - 换言之，也就是如果 $\mathbf{P}=(a_1,b_1,c_1)$ 则能转移到的 $\mathbf{X}=(a_1,b_1,q)$（满足 $c_1<q\le b_1$）。
- 如果 $i=2$，则即是 $\mathbf{X}=(a,b,c)$ 而 $\mathbf{P}=(a,c_2,c_2)$。
  - 换言之，也就是如果 $\mathbf{P}=(a_1,c_1,c_1)$ 则能转移到的 $\mathbf{X}=(a_1,q_1,q_2)$（满足 $c_1<q_2\le q_1\le a_1$）。
- 如果 $i=3$，则即是 $\mathbf{X}=(a,b,c)$ 而 $\mathbf{P}=(c_2,c_2,c_2)$。
  - 这是不可能的，因为没有任何 (q,q,q) 是 $\mathcal{P}$ 态。

我们需要注意，上述分析中，$i=2$ 时的情况对应的 $\mathbf{P}$ 必须满足第二行和第三行长度相等。

再考虑 $\mathbf{P}$ 的 c 等于当前的 c 的情况，即 $c(\mathbf{P})=c(\mathbf{X})$：

- 第一种情况是 $\mathbf{X}=(a,b,c)$ 而 $\mathbf{P}=(a,b_2,c)$。
  - 换言之，也就是如果 $\mathbf{P}=(a_1,b_1,c_1)$ 则能转移到的 $\mathbf{X}=(a_1,q,c_1)$（满足 $b_1<q\le a_1$）。
- 第二种情况是 $\mathbf{X}=(a,b,c)$ 而 $\mathbf{P}=(a_2,b,c)$。
  - 换言之，也就是如果 $\mathbf{P}=(a_1,b_1,c_1)$ 则能转移到的 $\mathbf{X}=(q,b_1,c_1)$（满足 $q>a_1$）。
- 第三种情况是 $\mathbf{X}=(a,b,c)$ 而 $\mathbf{P}=(b_2,b_2,c)$。
  - 换言之，也就是如果 $\mathbf{P}=(b_1,b_1,c_1)$ 则能转移到的 $\mathbf{X}=(q_1,q_2,c_1)$（满足 $q_1>b_1$ 和 $q_2\ge b_1$）。

总结一下

从 $\mathbf{P}$ 到 $\mathbf{X}$ 有**五种**转移（或者应该叫做「反向转移」）方式！

它们如下所示（假设 $\mathbf{P}=(a,b,c)$）：

1. 对 $\mathbf{P}$ 的限制：无限制；
   转移：$\mathbf{X}=(a,b,q)$，其中 $c<q\le b$。
2. 对 $\mathbf{P}$ 的限制：$b=c$；
   转移：$\mathbf{X}=(a,q_1,q_2)$，其中 $c<q_2\le q_1\le a$。
3. 对 $\mathbf{P}$ 的限制：无限制；
   转移：$\mathbf{X}=(a,q,c)$，其中 $b<q\le a$。
4. 对 $\mathbf{P}$ 的限制：无限制；
   转移：$\mathbf{X}=(q,b,c)$，其中 $q>a$。
5. 对 $\mathbf{P}$ 的限制：$a=b$；
   转移：$\mathbf{X}=(q_1,q_2,c)$，其中 $q_1>b$ 且 $q_2\ge b$。

其中前两种转移是将 c 增大了的，后三种转移的 c 值并不会改变。

我们将这些转移分别编号为 $1\sim 5$，以便后文中提到。

上述这些转移的使用将在下文中给出具体的例子，看不懂或觉得太复杂不想看的话不要着急。

除此之外，我们还可以证明这样一个定理：

- **定理 1：** 当 (a,b,c) 三维中有任意两维确定时，不论剩余的那一维如何变化，最多只能产生一个 $\mathcal{P}$ 态。
- **证明 1：** 假设有两个 $\mathcal{P}$ 态产生了，考虑剩余那一维较大的，它必然能转移到较小的那个，导出矛盾。

这条定理在一定程度上限制了 $\mathcal{P}$ 态的个数。

实际上，根据打出的表，我们可以做出如下猜测：

- 游戏 (a,b,c) 中 $\mathcal{P}$ 态的个数（分两类：零散无规律的，以及 a,b 之差恒定且以等差数列形式递增的）是 $\mathcal{O}(c)$ 级别的。
- 游戏 (a,b,c) 中 a 值最大的，零散无规律的 $\mathcal{P}$ 态（或有规律的 $\mathcal{P}$ 态中 a 值最小的），它的 a 值为 $2c+\mathcal{O}(1)$。

在打出的表（$c=0\sim 408$）的范围内，这两个结论均大致符合实际情况。

也就是说，我们可以猜测：在 $c\le k$ 的情况下，游戏 (a,b,c) 的 $\mathcal{P}$ 态的数目（有规律的序列仅算一个）是 $\mathcal{O}(k^2)$ 的。

且第二个猜测能帮助我们控制计算的范围，控制在 $[2c+\mathcal{O}(1)]^2$ 内，我们有望能找到一个 $\mathcal{O}(k^3)$ 的算法。

图解计算过程

回到当 $c=0\sim 10$ 时打出的表：

```
0: pat(1+k,0+k)...
1: (3,1),(2,2)
2: pat(4+k,2+k)...
3: (6,3),(7,4),(5,5)
4: (8,4),(9,5),(10,6),(7,7)
5: (10,5),(9,6),pat(11+k,7+k)...
6: (11,6),(12,7),(13,8),(9,9)
7: (13,7),(14,8),(12,9),pat(15+k,10+k)...
8: (15,8),(14,9),(16,10),(17,11),(12,12)
9: (16,9),(17,10),(14,11),pat(18+k,12+k)...
10: (18,10),(19,11),(20,12),(21,13),(14,14)
```

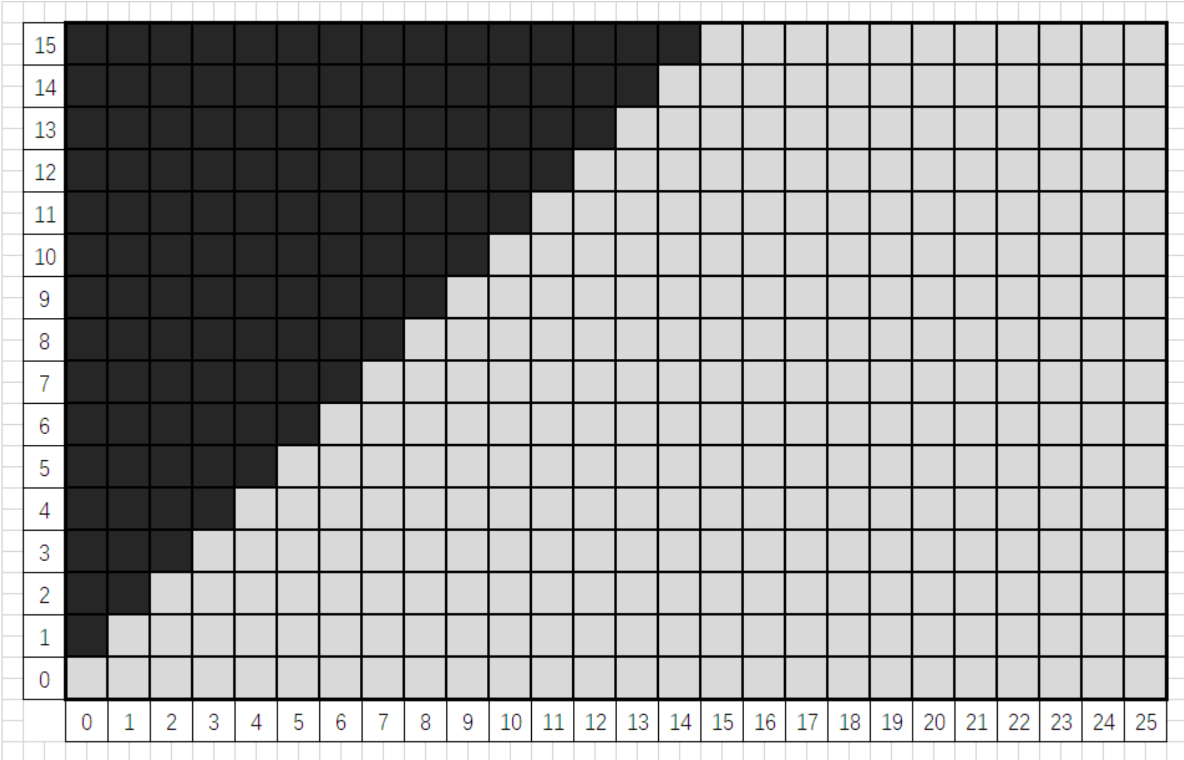
以及五种转移方式（假设 $\mathbf{P}=(a,b,c)$）：

1. 对 $\mathbf{P}$ 的限制：无限制；
   转移：$\mathbf{X}=(a,b,q)$，其中 $c<q\le b$。
2. 对 $\mathbf{P}$ 的限制：$b=c$；
   转移：$\mathbf{X}=(a,q_1,q_2)$，其中 $c<q_2\le q_1\le a$。
3. 对 $\mathbf{P}$ 的限制：无限制；
   转移：$\mathbf{X}=(a,q,c)$，其中 $b<q\le a$。
4. 对 $\mathbf{P}$ 的限制：无限制；
   转移：$\mathbf{X}=(q,b,c)$，其中 $q>a$。
5. 对 $\mathbf{P}$ 的限制：$a=b$；
   转移：$\mathbf{X}=(q_1,q_2,c)$，其中 $q_1>b$ 且 $q_2\ge b$。

接下来将配合图片显示如何通过这五种方式，用手玩快速地验证打出的表是否无误。

我们考虑按照 c 从小到大的顺序验证，对于固定的 c，我们再将 (a,b,c) 是 $\mathcal{P}$ 态的位置记录下来，并在二维平面上 (a,b) 处标记。

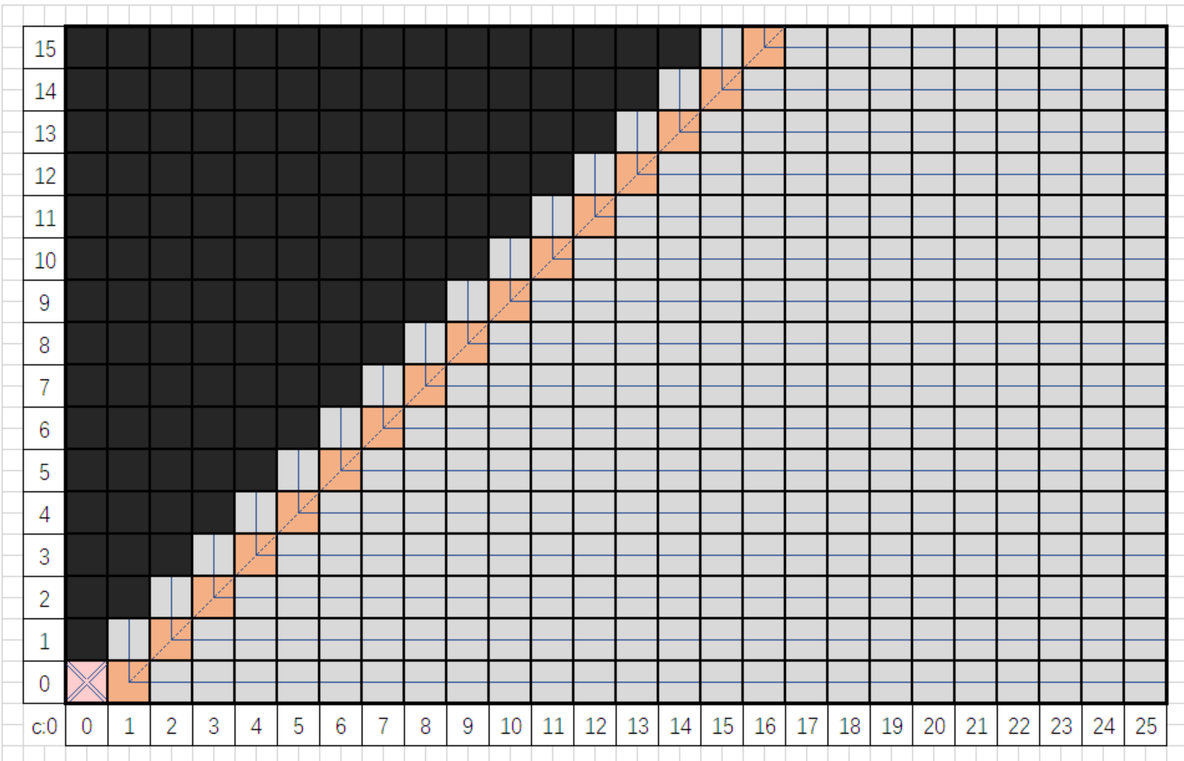
首先我们画出这样的一个「二维平面」：



其中将会展示当 c 固定，(a,b) 任意时的 $\mathcal{P}$ 态的情况，横坐标表示 a 的值，纵坐标表示 b 的值。

因为一定有 $b \le a$，所以对角线以上的部分被涂黑了。

考虑 $c=0$ 的情况：



其中 $(a,b)=(0,0)$ 的位置被特殊标记了，因为那里严格来说并不是一个合法的状态。

接下来我们找到「最小」的未被标记的位置，这里的「最小」指的是左下部分已经全部被标记了（偏序关系）。

我们找到了 $(1,0)$，对应着游戏 $(1,0,0)$，我们把它标记为一个 $\mathcal{P}$ 态，用橙色 ■ 标记。

对于每个 $\mathcal{P}$ 态，对其应用上文中的 3 号和 4 号转移，也即 (a,b,c) 转移到 (q,b,c) 和 (a,q,c)。

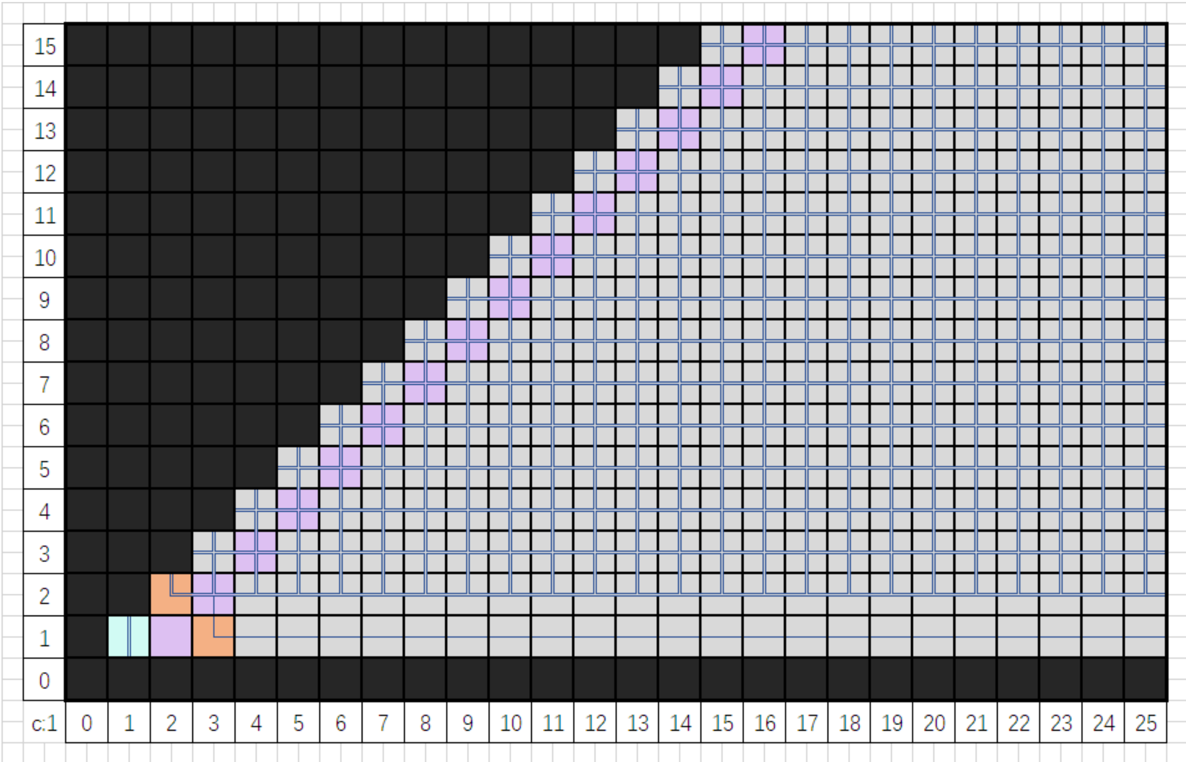
对应到图上也就是向右侧和上侧延伸出两条线，途经的位置均被该 $\mathcal{P}$ 态转移到，所以一定是 $\mathcal{N}$ 态。

接下来再找到最小的未被标记的位置，这里是 $(2,1)$，再对其应用 3 号和 4 号转移，就又消去了一行一列的 $\mathcal{P}$ 态的可能性。

以此类推，找到 $(3,2),(4,3),(5,4),\ldots$，已经可以看出循环的规律了，所以用虚线把最终循环的部分连接起来。

这就是当 $c=0$ 时的情况，仅利用了上文中的两种转移方式，确定了所有的 $\mathcal{P}$ 态的规律。

接下来再考虑 $c=1$：



一个最明显的变化就是，$b=0$ 的那一行也像 $b>a$ 的部分一样被涂黑了，因为此时必须要有 $b\geq c=1$。

除了 $b=0$ 消失了之外，还出现了「浅紫色 ■」和「浅蓝色 ■ 加格子中间的双竖线」以及覆盖整个 $b\geq 2$ 区域的网格型标记。

这就对应着 $c=0$ 时没用到的另外三种转移方式：

1. 浅紫色 ■：对应了第 1 种转移：$(a,b,c)\to(a,b,q)$。
   可以看出它的 (a,b) 是不变的，也就是会一直留在平面的相同位置，导致对更大的 c 来说，这些位置一定是 $\mathcal{N}$ 态。
2. 浅蓝色 ■ 加格子中间的双竖线：对应了第 2 种转移：$(a,c,c)\to(a,q_1,q_2)$。
   可以看出它的 a 值是不变的，在图中体现的就是一个恒定的 $a=a_0$ 的竖条区域被覆盖，而且也会影响到之后所有 c 值的平面。
3. 覆盖整个区域的网格型标记：对应了第 5 种转移：$(b,b,c)\to(q_1,q_2,c)$。
   这是 c 值不发生改变的转移类型，所以不用颜色进行标记。
   这个转移能转移到所有 (q_1,q_2)，只要满足 $q_1,q_2\geq b$ 就行（不包括 (b,b) 本身）。
   所以在图中体现的是一个无限延伸的，线条类型为双线的网格型覆盖标记。

实际上是先用第 $1,2$ 种转移，也就是格子颜色的转移，因为它们是只能从较小的 c 转移到较大的。

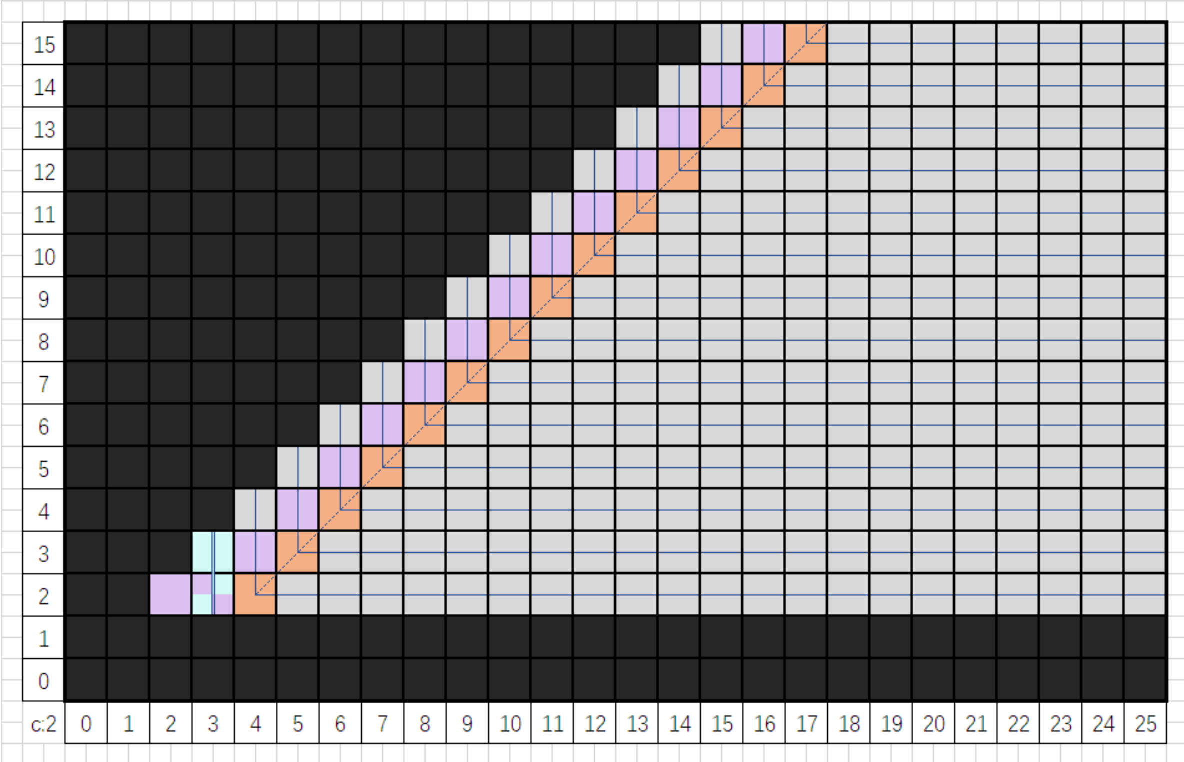
然后我们再使用橙色 ■ 去标记可以被确认的 $\mathcal{P}$ 态。

这里标记了 $(3,1)$，然后用第 $3,4$ 种转移，向右向上延伸出两条线型标记。

然后发现接下来的 $\mathcal{P}$ 态是 $(2,2)$，它落在了对角线上，所以触发了第 5 种转移。

它就把它右上角的所有位置都覆盖掉了，它们都变成了 $\mathcal{N}$ 态。

接下来我们观察 $c=2$ 的情况：



这里黑色的部分又上涨了一行。

然后是颜色转移（即第 $1,2$ 种转移，对应浅紫色 和浅蓝色 的背景颜色）。

颜色转移可以继承上一个 c 的颜色，但是还要加上的是上一个 c 的 $\mathcal{P}$ 态带来的影响。

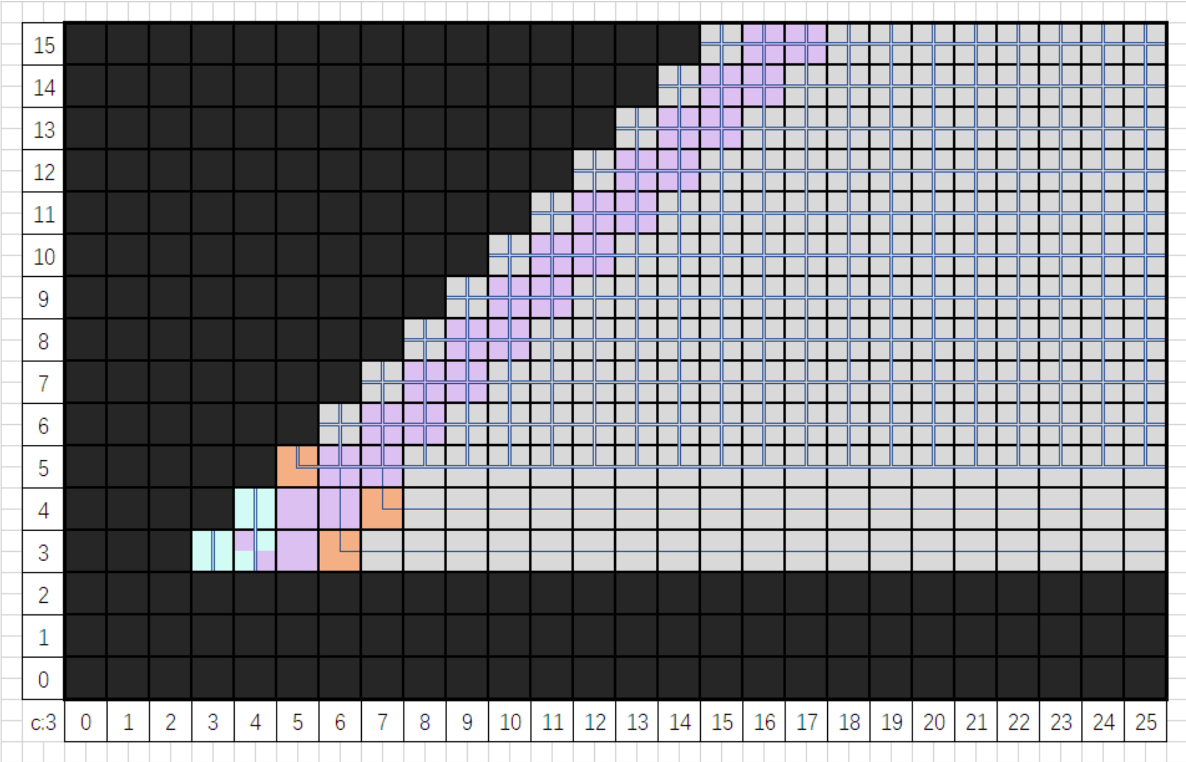
在这张图中的体现就是 $(2,2)$ 变紫了，而 $(3,2)$ 和 $(3,3)$ 也都添上了一抹蓝色（换句话说是 $a=3$ 这一整列都会这样）。

注意这里 $(3,2)$ 既有紫色又有蓝色，这是可能存在的，这意味着它会被两个 c 值更小的 $\mathcal{P}$ 态转移到。

颜色转移结束后，就开始进行使用第 $3,4$ 种转移的，c 值不改变的转移方式（也就是自己转移给自己）。

这样又延伸出一条形如 $(4,2),(5,3),...$ 的橙色格子序列。这就是 $c=2$ 时的所有 $\mathcal{P}$ 态。

接下来是 $c=3$：



黑色部分又上涨了一行。

然后继续搞颜色转移，注意到紫色和蓝色都又被染进了新的格子里。

蓝色的 $a = 4$ 列是由于 $(4, 2, 2)$ 是 $\mathcal{P}$ 态，应用第 2 种转移。

新加入的紫色的斜对角线，显然就是由于 $c = 2$ 时的无限延伸的 $\mathcal{P}$ 态序列，应用第 1 种转移。

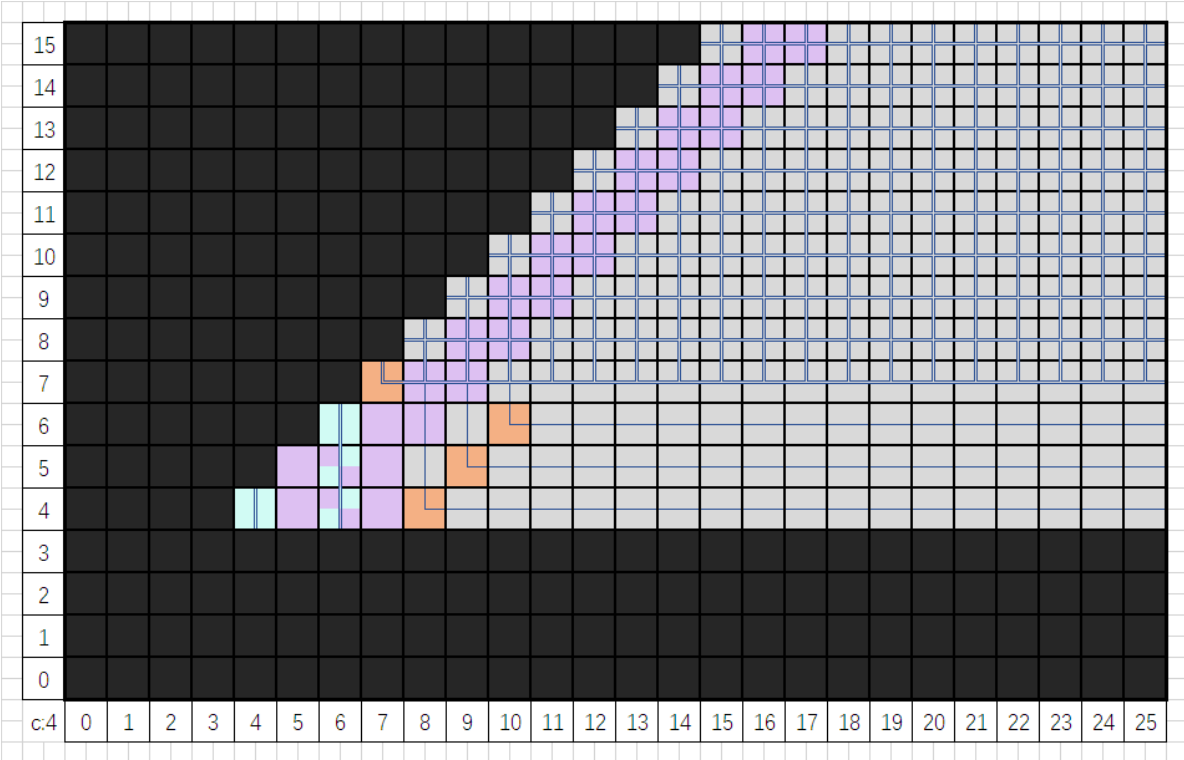
做完了颜色转移，再用橙色标记新的 $c = 3$ 时的 $\mathcal{P}$ 态。

标记了 $(6, 3)$ 和 $(7, 4)$ 后，别忘了对它们使用第 $3, 4$ 种转移，也就是朝着右方和上方发射两条线，标记沿途的格子变成 $\mathcal{N}$ 态。

最后标记到了 $(5, 5)$，因为它的 $a = b$，所以触发了第 5 种转移，把右上方的所有格子都标记成了 $\mathcal{N}$ 态。

这三个就是 $c = 3$ 时唯三的 $\mathcal{P}$ 态了。

我们还可以继续考虑 $c = 4$：



黑色行上涨和颜色转移这里就不展开了。如何确定新的 $\mathcal{P}$ 态也由读者自行完成吧。

每次转移不要忘记：

1. 黑色行上涨一行。
2. 紫色转移：继承上一个 c 值的平面，但是把上一个 c 值的平面的橙色（$\mathcal{P}$ 态）改成紫色（第 1 种转移）。
3. 蓝色转移：上一个 c 值中的橙色格子，一定有一个是紧贴下方黑色行的，请把它所在列在当前状态下染成蓝色（第 2 种转移）。

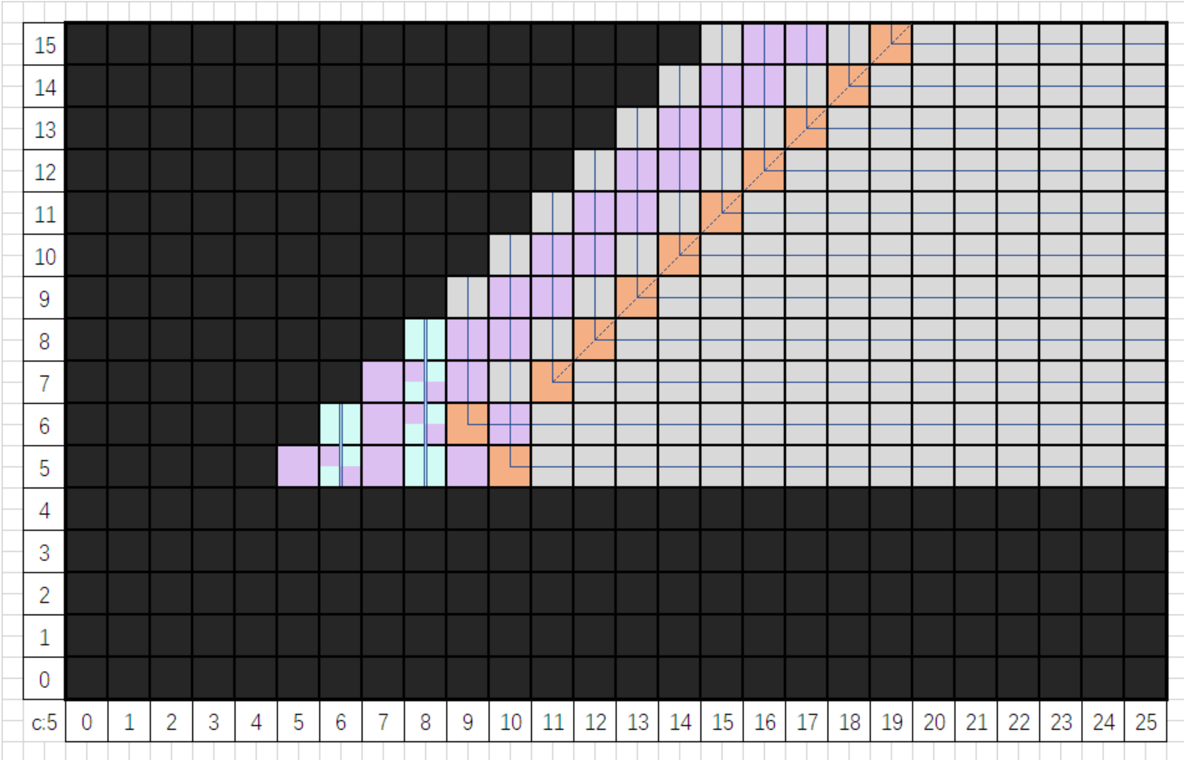
然后按照行从低到高去确定每个橙色格子，然后应用第 $3,4$ 种转移。

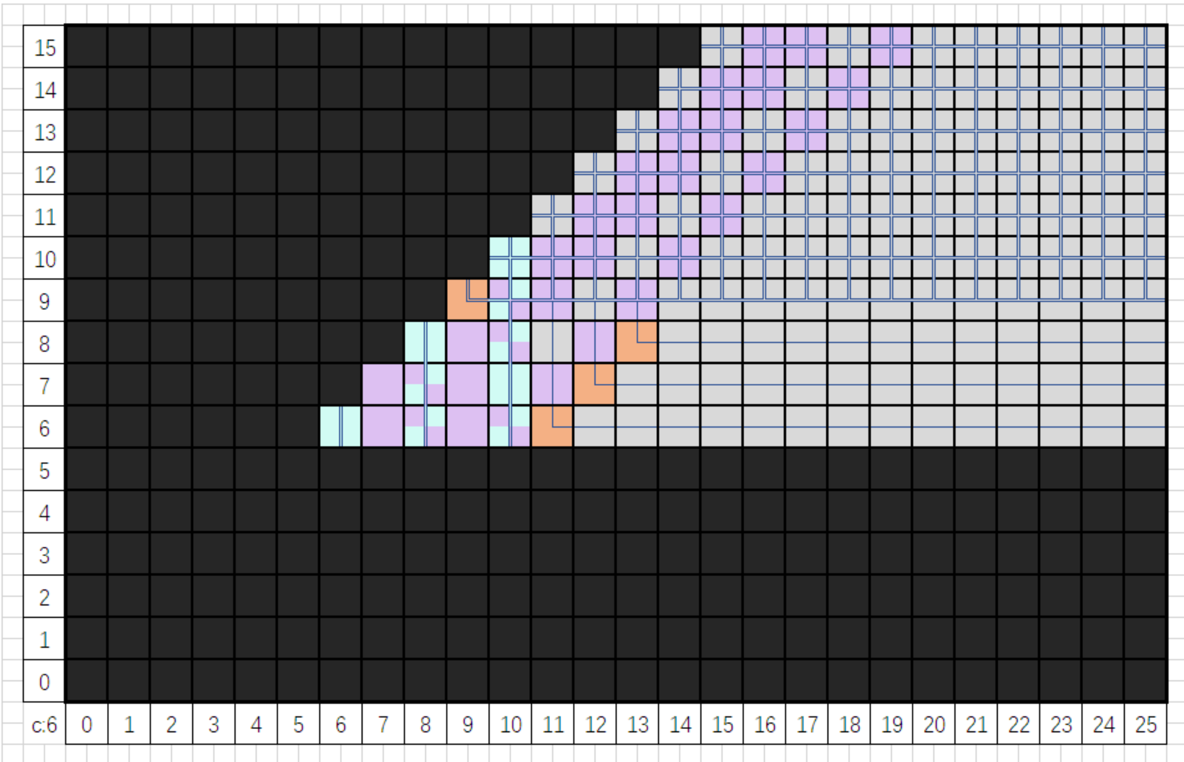
最后如果标记到了对角线上，就应用第 5 种转移，然后退出。

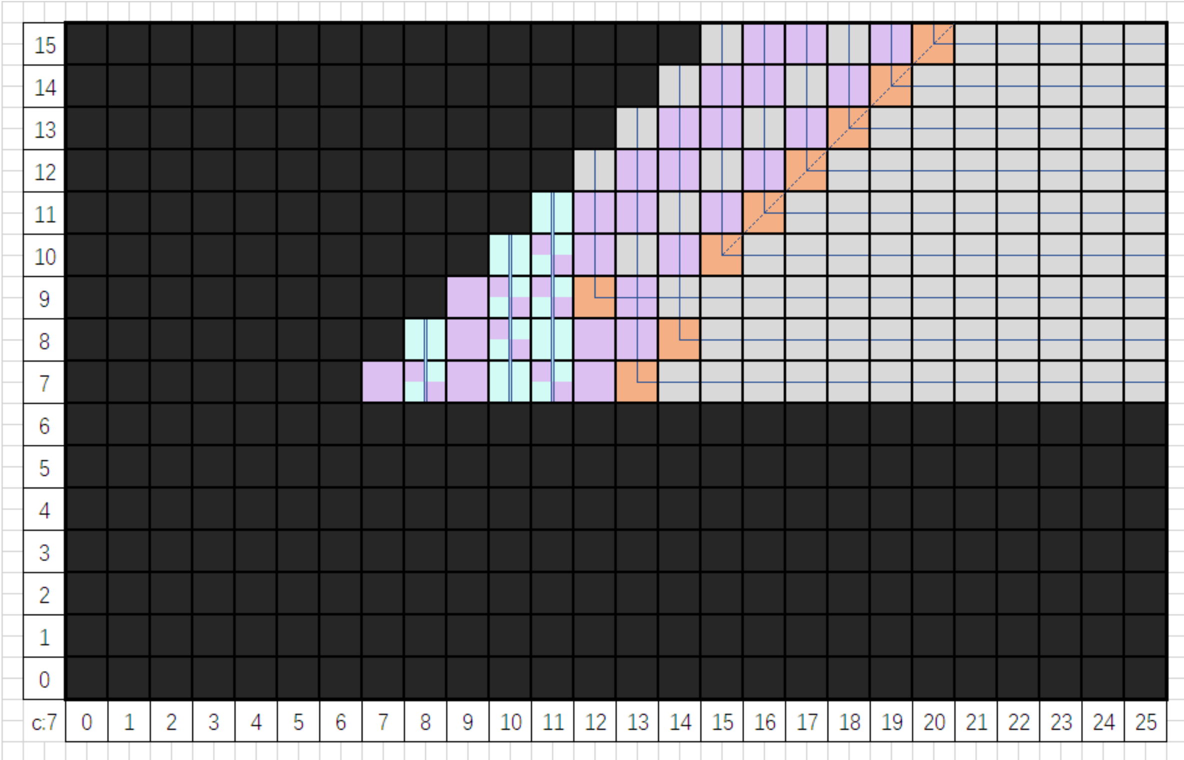
否则，永远不会标记到对角线上，那一定是出现了循环，标记其，然后退出，把 c 加上 1，进入下一个平面。

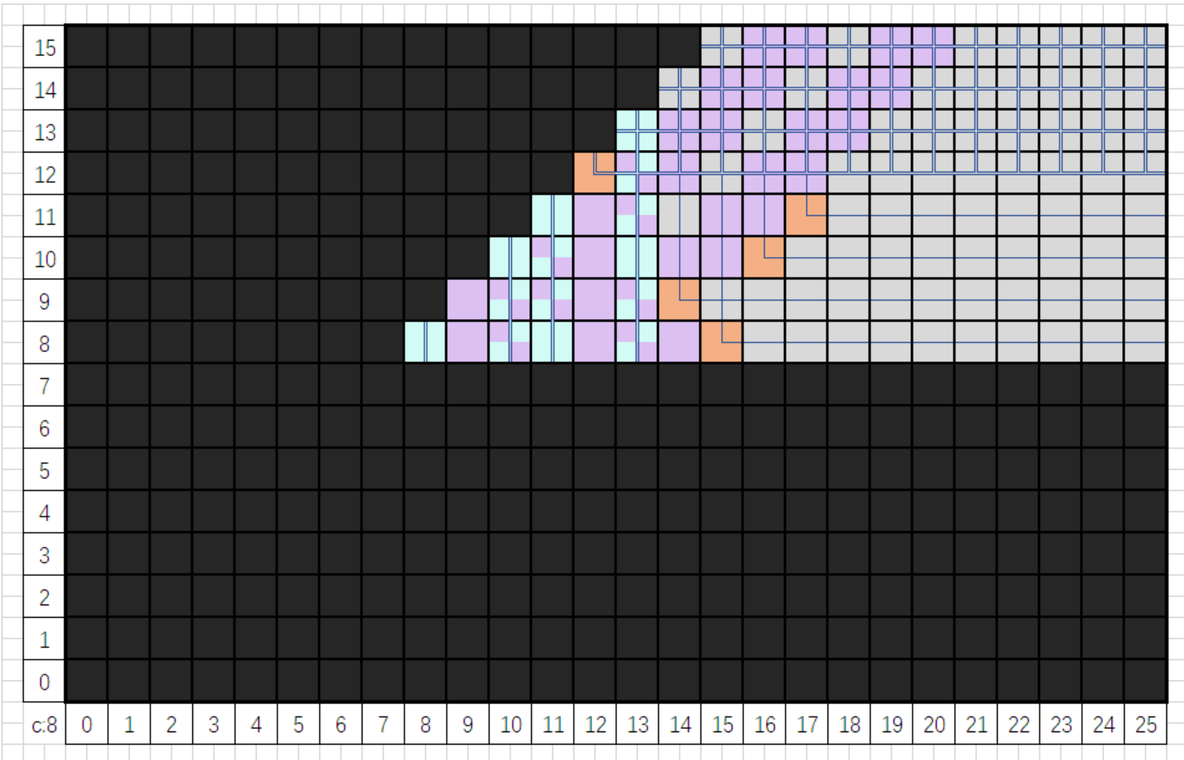
按照这样的顺序去依次处理每个 c 值，就最终能够得到所有的 $\mathcal{P}$ 态。

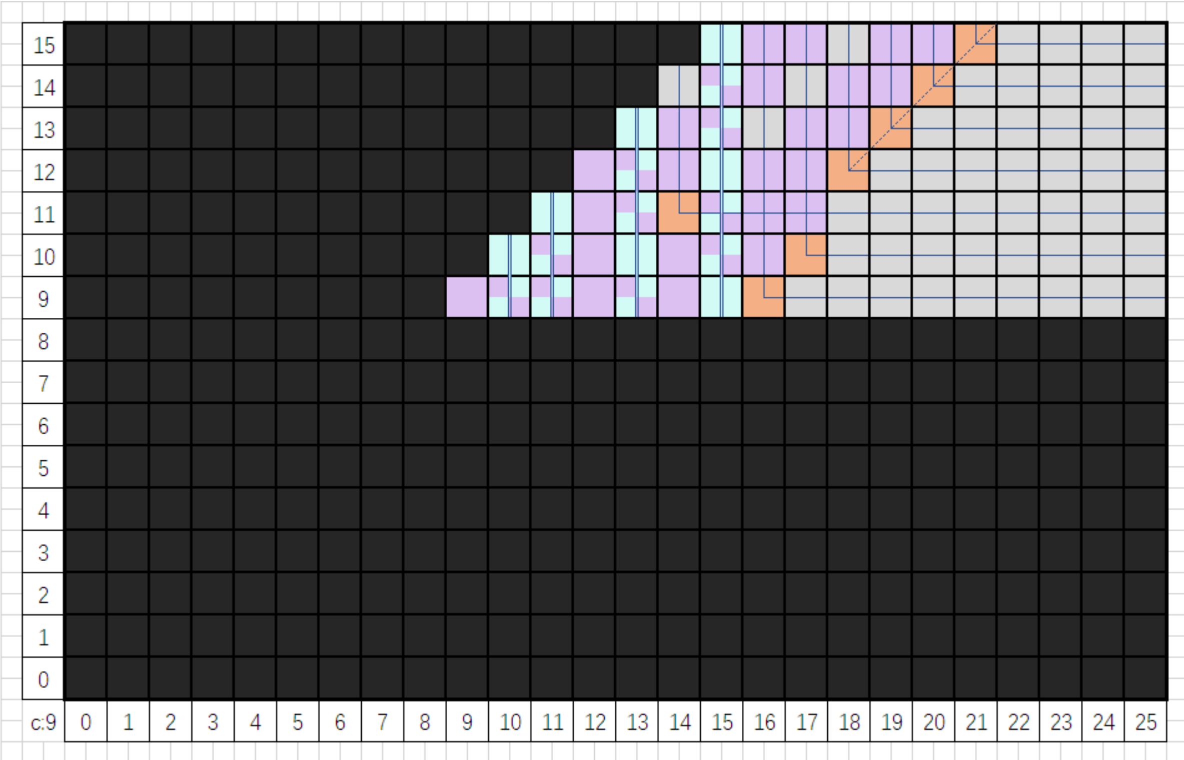
在这里再贴出 $c=5\sim 10$ 时的平面状态（看左下角以确认 c 的值），以供读者验证：

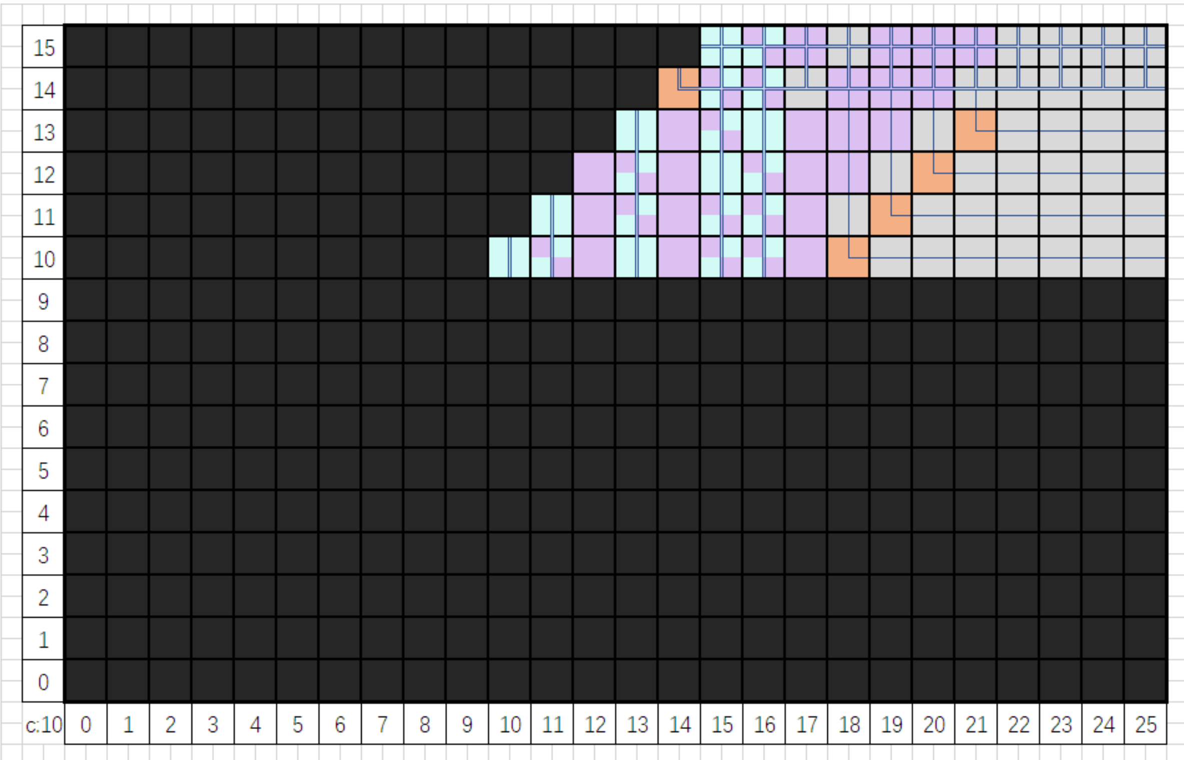

c:5


c:6


15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
c:7
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25


15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
c:8
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25





可以看出，确实是有点混沌的。难以找到明显的，通用的规律。

但是我们此时就可以根据这种做法写出新的计算 $\mathcal{P}$ 态的程序了。

打表程序更新

很可惜我并没有找到比较合适的判断是否进入循环的方法。

总感觉想到的各种方法都比较别扭，正确性也不太敢保证。

所以不如采取更激进的策略：和先前的程序相同，根据 $\mathcal{P}$ 态的 a,b 范围的猜想，直接算到 $a,b\le 2k+\mathcal{O}(1)$ 不就好了。

这个猜想似乎在一定范围内都是成立的。所以我们的打表程序的时间复杂度可以做到 $\mathcal{O}(k^3)$，空间复杂度 $\mathcal{O}(k^2)$。

但在实测的时候，它的常数是非常小的，可以轻松在 20 秒内跑过 $k=4300$ 的数据范围，并把整理得的规律输出到文件中。

对于 $c \le 4300$ 的表，参见 `table2.txt`。其中 `c: pat(a,b,q)` 表示 $(a+kq, b+kq, c)$ $(k \in \mathbb{N})$ 均为 $\mathcal{P}$ 态。

如果你不想看前面的一大堆垃圾信息，参见 `table3.txt`，这张表仅包含了后面的周期信息。

注意 `table3.txt` 的 k 值到达了 6600，所以运行时间也到达了 40 秒。

而且还开了个 13280×13280 的 `int` 数组，使用的内存超过了 512 MiB。

其中如果该 c 值对应的 $\mathcal{P}$ 态个数有限，会写 `finite(x)` 表示总个数为 x。
否则会写 `inf(x)` 表示不在周期中的 $\mathcal{P}$ 态总个数为 x，然后会给出 `pat(a,b),cyc=q` 的形式，表示的意义类似。

注意，在这个范围内，周期长度为 3 的和为 9 的都出现了：

- 长度为 3 的有 $c = 2027, 2751, 6539$。
- 长度为 9 的有 $c = 6541$。

我将把提到的程序的代码文件和提到的表格放在同文件夹下。

同在文件夹下的还有引用到的 12 张图片和作图工具：一个 Microsoft Excel 工作表。

感谢你的阅读。

参考文献

[1] Doron Zeilberger, *Three-Rowed CHOMP*, Adv. Applied Math. **26** (2001) 168-179.

[2] Andries E. Brouwer, *The game of Chomp*, 2018 or later.