arXiv:2303.07266v1 [cs.DS] 9 Mar 2023

# 2048 游戏的计算界限

Alexey Slizkov*

**摘要**

2048 是一款单人电子游戏，数百万人都在玩，其中大多数玩家使用移动设备。我们首次严格证明，存在一种算法，其获胜概率至少为 0.99969；还证明存在一种策略，能够保证获得 256 方块（概率为 1）。

## 1 引言

2048 是由意大利网页开发者 Gabriele Cirulli 开发的游戏，并于 2014年3月托管在 GitHub 上。这是一款供单人游戏的滑块拼图，目标是在网格上合并编号方块，直到获得数字为 2048 的方块。达成这一目标后游戏不会结束，玩家可以继续游戏，尝试生成更大的编号方块 [1]。人们长期以来一直在探索这款游戏的理论和实践层面。例如， [2] 和 [4] 证明了一些困难性结果（NP 困难性和 PSPACE困难性）。另一方面，也有大量研究关注该游戏的启发式方法以及机器学习策略的适用性；例如， [5] 采用了蒙特卡洛树搜索，而 [6], [3]则讨论了其他机器学习方法。

请记住，方块会以相同的概率出现在随机空闲单元格中，其中出现 2 的概率为 90%，出现 4 的概率为 10%；概率的来源就是这些。

最自然的问题之一是，这个游戏的最优获胜概率是多少。基于人工智能的算法表明，这个概率可能非常高，但此前尚无证明。我们首次证明，只需使用整数进行计算，并选取和遍历一个相对较小的位置集合（玩家可以留在其中并以很高的概率获胜），就能得出获胜概率至少为 0.99969486。

自然的博弈论问题是：是否存在一种策略，能够始终达到2048（也就是说，前述概率恰好为1）；或者是否存在一种策略，能够始终阻止玩家达到2048。我们研究了较小场地大小上的游戏，并通过在一台拥有超过500GB随机存取存储器和80至96个核心的机器上进行为期三个月的计算，证明了最大可达方块的256下界。我们还展示了关于在不同场地大小上保证可达不同方块的若干其他类似结果；这次每次计算最长持续数天，使用的是一台配备RTX 4090 GPU、64GB随机存取存储器和两块速度为3GB/s的固态硬盘的机器。

*elexunix@gmail.com；俄罗斯国立高等经济大学，俄罗斯联邦莫斯科

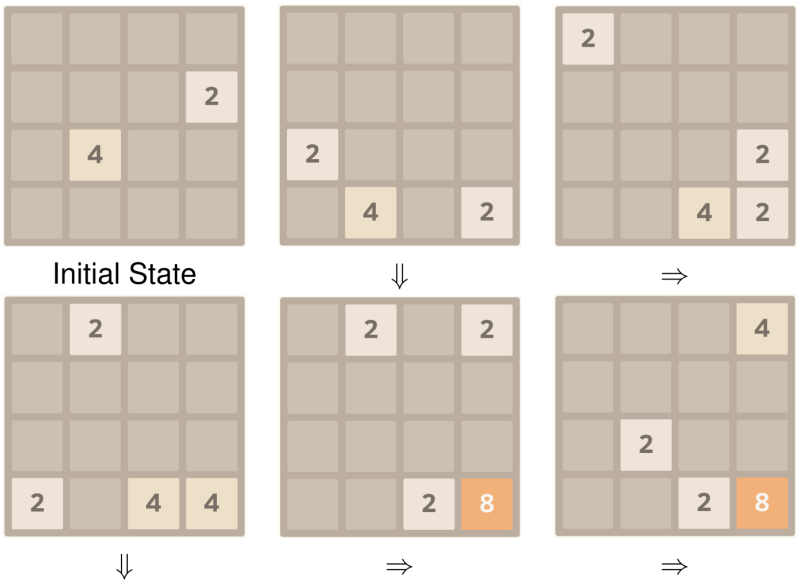



图 1：2048 游戏的前六次操作。来源： [4]

## 2 保证可达性

让我们固定场地大小 $m \times n$ 和目标方块 $T$。考虑由所有方块均小于 $T$的 $m \times n$ 位置构成的图$G_{m,\ n,\ T}$。对于保证可达性和获胜概率的问题，只需考虑这些位置。注意，这样的位置共有$2(\log_2 T)^{m \times n}$ 个（每个单元格可以为空，或拥有方块 $2^1,\ 2^2, \ldots\ldots$，或 $2^{\log_2 T-1}$，并且可能轮到玩家或计算机行动）。对于 $m = n = 4$ 和 $T = 256$，这会得到 $2 \cdot 8^{16} \approx 5.63 \cdot 10^{14}$个位置，数量尚可处理，但显然无法装入随机存取存储器或存储驱动器。基本思路是按某种顺序遍历该图，并将每个顶点标记为胜利或失败（目标是实现 $T$）；终止局面是指要么可以合并两个 $T/2$ 方块，要么无法进行任何游戏。谁能赢得游戏这一答案，就蕴含在初始棋盘[1]的数据中。

为了在所需内存限制下解决这个问题，我们按和将游戏图拆分为层：一个位置的和是其中所有方块牌面数字的总和，空单元格按 0 计算。请注意，即使有些方块合并，玩家滑动操作也不会改变这个和；而电脑的操作（添加新方块）会使和恰好增加 2 或 4。如果我们按以下和与回合将图拆分为各层，并按此顺序遍历，就能一次只存储其中三层： [和 $mnT/2$，玩家回合], [和 $mnT/2-2$，电脑回合], [和 $mnT/2-2$，玩家回合], [和 $mnT/2-4$，电脑回合], $\ldots\ldots$，[和 4，电脑回合], [和 4，玩家回合], [和 2，电脑回合], [和 2，玩家回合], [和 0，电脑回合]。不过，这些层有多大呢?

[1]原版游戏开始时，棋盘上有两个随机方块。为了明确起见并简化分析（排除这两个方块带来的依赖），我们考虑从空棋盘开始游戏，并由电脑先操作。不过，对于保证达到 256 的结果，我们证明了，只要棋盘上有两个方块且轮到玩家操作，该结果就不依赖于初始配置；在所有 $4 \cdot \binom{16}{2} = 240$ 情形下，游戏最终都可获胜

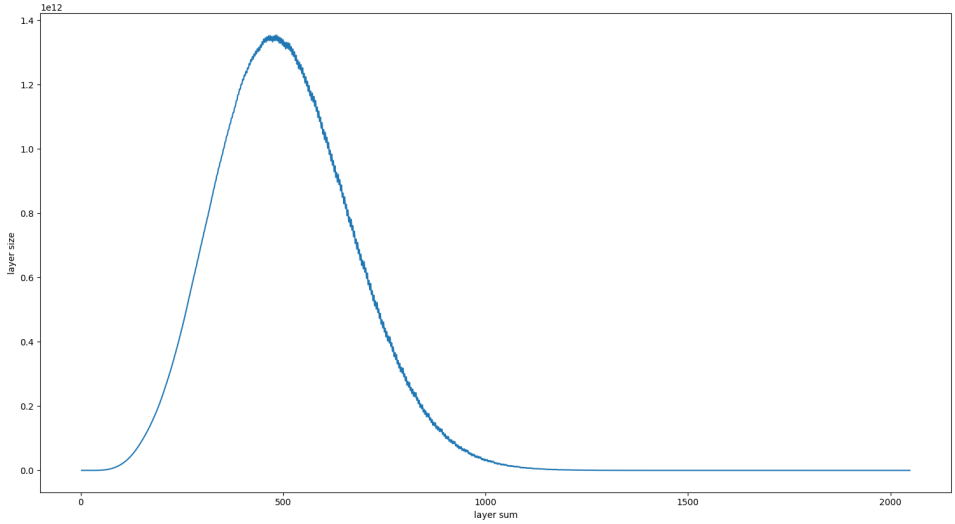



图 2：游戏图各层的大小$G_{m,n,T}$

我们编写了一个额外的程序来计算这些层中的位置数量，结果如图 2 所示，最大的层包含 $1353817378016 \approx 1.35 \cdot 10^{12}$ 个位置。我们决定为每个位置存储一位，因此需要 508GB 随机存取存储器。我们通过将层拆分为大小为 $2^{14}$ 个位置的块，并使用块索引的哈希将每个块伪随机地分配给一个线程，从而实现最多 96 个线程的并行处理。结果发现，不使用哈希进行直接划分会导致各线程的工作量不均，负载不平衡。但此时又出现了另一个问题：我们需要一种有效的方法，对以大型位数组形式存储的这些层进行索引。我们需要能够解决两类问题：快速索引，即根据位置在层内的编号获取该位置；以及获取给定位置的索引。

这两个问题均通过动态规划求解，使用预先计算的、大小为 $(mn+1) \times (mnT/2+1)$ 的数组。该数组存储了对于每种单元格数量及其方块牌面数字之和，用方块和空单元格填满这些单元格且使总和符合要求的可能方式数。例如，数组中第 (5,216) 个单元格的值，就是总和为 216 的五个方块或空单元格的五元组数量。该数组在程序启动时预先计算一次，按单元格数量递增的顺序进行，从 0 个单元格开始；对于总和为 0 的情况，可能方式数为 1，对于所有正总和则为 0。具体过程详见算法 1。计算完这张辅助表（称为 $I$，因为它用于子问题的索引）后，我们可以按如下方式求出给定位置在某一层内的索引：设 $a_1$、……、 $a_{mn}$ 依次为该位置各单元格中的方块牌面数字，空单元格记为 0。那么该索引为

$$\sum_{i=1}^{mn} \begin{cases} 0, \text{ if } a_i = 0, \\ I\left[mn-i\right]\left[\sum_{j=i+1}^{mn} a_j\right] + \sum_{k=1}^{\log_2 a_i - 1} I\left[mn-i\right]\left[\sum_{j=i+1}^{mn} a_j - 2^k\right], \text{ otherwise.} \end{cases}$$

通过算法 2 解决了逆问题：根据索引查找位置。

**算法 1** 填充 $I$ 辅助表

**对于** $s \leftarrow 0, 2, \ldots,$ $mnT/2$ **执行**
  **如果** $s = 0$ **则**
    $I[0][s] \leftarrow 1$
  **else**
    $I[0][s] \leftarrow 0$
  **结束如果**
**结束循环**
**对于** $c \leftarrow 1, \ldots,$ $mn$ **执行**
  **对于** $s \leftarrow 0, 2, \ldots,$ $mnT/2$ **执行**
    $I[c][s] \leftarrow I[c-1][s]$
    $t \leftarrow 2$
    **当** $t \leq s$ **执行时**
      $I[c][s] \leftarrow I[c][s] + I[c-1][s-t]$
      $t \leftarrow 2 \cdot t$
    **结束当**
  **结束对于**
**结束循环**

**算法 2** 根据其和以及该和所在层中的索引查找位置

$r \leftarrow$索引
$s \leftarrow$和
**对于** $i \leftarrow 1, \ldots,$ $mn$ **执行**
  **如果** $r < I[mn-i][s]$ **，则**
    $a_i \leftarrow 0$
  **else**
    $r \leftarrow r - I[mn-i][s]$
    $t \leftarrow 2$
    **当** $r < T[mn-i][s-t]$ **时执行**
      $r \leftarrow r - I[mn-i][s-t]$
      $t \leftarrow 2 \cdot t$
    **结束循环**
    $a_i \leftarrow t$
    $s \leftarrow s - t$
  **结束条件判断**
**结束 for 循环**
▷ 此时， $r = 0$ 和 $s = 0$

计算证明了 256 是可达的。作者保存了所有可达性二进制数组，并将其压缩至 9 TB，目前存储在 NRU HSE计算机科学学院的分布式存储中。

我们已经进行了类似的计算，其中一些使用 CUDA 技术在图形处理器上编程，以加快计算速度。我们的主要保证可达性结果可总结于表 3 中。更多结果见附录。

然而，这张表表明，在原始场地大小下，可达到的最大方块可能远小于 2048，因此存在一种生成方块的策略，能让玩家永远无法达到 2048。我们

| $m\backslash n$ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 2 | 4 | 4 | 4 | 4 | 4 |
| 2 | 4 | 8 | 16 | 32 | 32 | 64 |
| 3 | 4 | 16 | 32 | 64 | 128 | |
| 4 | 4 | 32 | 64 | ? | | |
| 5 | 4 | 32 | 128 | | | |
| 6 | 4 | 64 | | | | |

图3：不同场地大小下保证可达到的最大方块数

对如何组织 512 方块的计算有一些想法，无需数 TB 的随机存取存储器，也无需耗费数年计算时间，尽管该图包含超过 3.7 千万亿个节点。

## 3 概率下界

概率下界的计算依赖于关于常数和层及其内部有效索引的相同思路。但这一次，我们必须为每个位置存储比一位更多的信息：我们需要在每个节点中存储概率的下界，也就是一个实数。我们用浮点数近似计算了一些结果（见附录），但也让程序仅使用整数算术来证明下界，并始终向下取整。结果表明，uint16_t 不足以存储概率（乘以 $2^{16}$），它给出的下界很低；而 uint32_t 满足了我们的需求（uint64_t 带来的精度提升很小，因此我们没有使用它）。

对于我们的主要结果（在原始棋盘上获得原始 2048 方块的概率为 0.99969486）来说，为了使计算可行，我们大幅限制了允许的位置集合。事实上，计算仅用了一天半就完成了，主要瓶颈在磁盘带宽（3GB/s）。我们使用了固态硬盘，所有计算的总写入字节数（TBW）约为 400TB，读取量约为 1.2PB。执行这项计算的图形处理器所在机器只有 64GB 随机存取存储器，而最大层的大小超过了这一容量。实际上，计算使用了以下可用资源的分层结构：24GB 的高速 GDDR6 图形处理器内存，带宽为 1TB/s，用于存储访问最频繁的数据；接着是 64GB 主机随机存取存储器，我们的配置带宽为 41GB/s，主机随机存取存储器与图形处理器内存之间的数据传输完全由 Nvidia CUDA 编译器和驱动程序通过 CUDA API 中的 cudaMallocManaged 等调用进行管理；再接下来是读写速度约为 3GB/s 的高速固态硬盘，用于存储我们处理的全部三个层。我们将每层分成 32GB 的批次，这样两个批次就能装入内存。算法 3 描述了这一过程

更详细地说。

**算法 3** 无法装入随机存取存储器的处理层详情

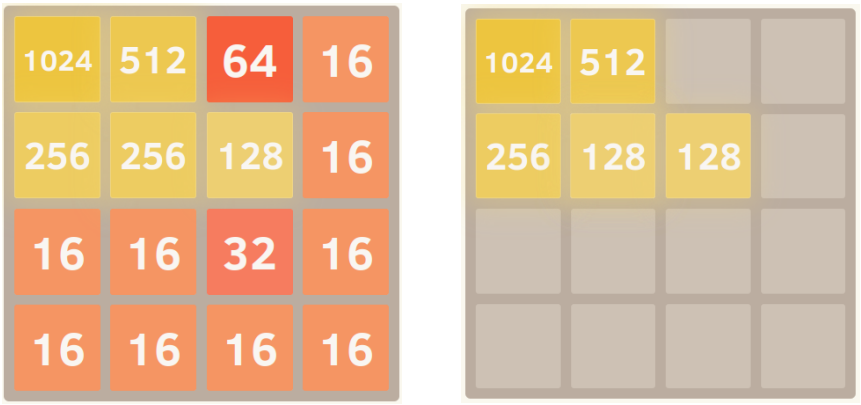
**对于** $s \leftarrow$ 最大和, 最大和 $-2$ , . . ., 0 **do**
  **对于** 计算机层中和为 $s$ 的每个批次 $B_0$ **执行**
    用零概率初始化批次 $B_0$
    **对**玩家层中和为 $s+2$ 的每个批次 $B_1$ **执行**
      从磁盘读取批次 $B_1$
      将批次 $B_1$ 数据所隐含的概率加到批次 $B_0$
    **结束循环**
    **对**玩家层中和为 $s+4$ 的每个批次 $B_1$ **执行**
      从磁盘读取批次 $B_1$
      将批次 $B_1$ 数据所隐含的概率加到批次 $B_0$
    **结束循环**
    在批次 $B_0$中，将累积的概率和除以单元格数——
空闲单元格的数量
    将批次 $B_0$ 写入磁盘
  **结束循环**
  如果存在，则从磁盘中移除和为 $s+4$ 的玩家层
  **对于** 玩家层中和为 $s$ 的每个批次 $B_0$ **执行**
    将批次 $B_0$ 初始化为零概率
    将批次 $B_0$ 中终端胜利节点的概率替换为 1
    **对于** 计算机层中和为 $s$ 的每个批次 $B_1$ **执行**
      在批次 $B_0$ 中，使用批次 $B_1$ 更新概率
    **结束循环**
  **结束循环**
**结束循环**

通过按照图4a中的数值限制每个单元格上的方块，实现了对待考察位置数量的大幅削减。这使待考察状态数量从 $2{\cdot}11^{16} \approx 9.19{\cdot}10^{16}$ 减少到 $11{\cdot}10{\cdot}9^2{\cdot}8{\cdot}7{\cdot}6{\cdot}5^9 \approx 1.17 \cdot 10^{13}$，缩减了近 8000 倍。不过可以注意到，在这些限制下，不可能在彼此相邻的位置形成两个 1024方块。事实上，我们设定的目标是让较大的方块出现在图4b所示的位置。可以证明，这足以获得 2048 方块（见引理 1），而事实证明，这些关于如何获得 2048 方块的限制仍使我们的胜率下界达到 0.99969486。

**引理 1** 无论其他方块如何，五个大方块如图*4b*所示的配置都是必胜的。

# 4 未来工作

要遍历更大的图，例如包含约 $3.71 \cdot 10^{15}$ 个状态的图 $G_{4,\ 4,\ 512}$，我们需要进一步拆分层，因为批处理方法会因二次复杂度而变得不可行：在计算的某个阶段，三层的总大小将达到 1.7TB。虽然无法通过采用另一种遍历图的方式减少这部分数据量，但我们可以更细致地研究边与边之间的图的边。以下想法




(a) 每个单元格的最大允许方块数　　(b) 获胜配置所需的方块

图4：计算中的最大允许方块数和所需方块数

似乎很有希望：将层进一步拆分为包含固定数量方块的组。在我们的例子中，将有 735471 组，每组最多仅包含 1630 亿个位置（20GB）。

而且，我们用来证明原始获胜概率下界的方法，经过上述修改后，也可用于证明更大方块数的下界，例如 32768 和 65536；这些数字受到该问题的一些研究人员关注：例如，[7] 声称约有 3% 的概率能达到 65536，大幅打破了此前约 0.02% 的纪录。

## 参考文献

[1] *2048*（电子游戏）。URL：https://en.wikipedia.org/wiki/2048%5C_(video%5C_game)（访问于 2023-03-08）。[2] Stefan Langerman。《Threes!、Fives、1024! 和 2048 都很难》。载于：理论计算机科学（2018）。[3] Kiminori Matsuzaki。《使用反向时序一致性学习和重新开始来开发 2048 玩家》。载于：计算机游戏进展（2017）。[4] Rahul Mehta。《2048 是（PSPACE）难题，但有时很容易》。载于：*arXiv* 预印本，*arXiv:1408.6315*，版本 *1*（2014）。[5] Philip Rodgers 和 John Levine。《对 2048 人工智能策略的调查》。载于：*IEEE*计算智能与游戏会议（2014）。[6] Sukovic Goran Tuponja Boris。《演化神经网络来玩 2048 游戏》。载于：电信论坛（2016）。[7] Hanhong Xue。URL：https://github.com/macroxue/2048-ai（访问于 2023-03-08）。

# A 补充计算结果

## A.1 保证可达性

下表总结了保证可达性的结果：

| 场地大小 | 游戏过程中允许的最大方块数 | Goal | 结果 |
|---|---|---|---|
| $2 \times 4$ | 16 16 16 8<br>16 16 16 8 | 32 | 可获胜，最小 |
| $2 \times 4$ | 32 32 32 32<br>32 32 32 32 | 64 | 不可获胜 |
| $3 \times 3$ | 16 16 16<br>16 16 16<br>8 8 8 | 32 | 可获胜，最小 |
| $3 \times 3$ | 16 16 8<br>16 16 8<br>8 16 16 | 32 | 可获胜，最小 |
| $3 \times 3$ | 16 8 16<br>16 16 16<br>16 8 16 | 32 | 不可获胜，极大 |
| $3 \times 3$ | 16 16 16<br>16 8 16<br>16 16 16 | 32 | 不可获胜，极大 |
| $3 \times 3$ | 32 32 32<br>32 32 32<br>32 32 32 | 64 | 不可获胜 |
| $2 \times 5$ | 16 16 16 16 8<br>16 16 16 8 8 | 32 | 可获胜，最小 |
| $2 \times 5$ | 32 32 32 32 32<br>32 32 32 32 32 | 64 | 不可获胜 |
| $2 \times 6$ | 16 32 32 32 32 16<br>16 32 32 32 32 16 | 64 | 可获胜，最小 |
| $2 \times 6$ | 64 64 64 64 64 64<br>64 64 64 64 64 64 | 128 | 不可获胜 |
| $3 \times 4$ | 32 32 32 16<br>16 32 32 16<br>16 16 16 16 | 64 | 可获胜，最小 |
| $3 \times 4$ | 16 32 32 16<br>32 32 32 16<br>16 16 16 8 | 64 | 可获胜，最小 |
| $3 \times 4$ | 32 32 16 32<br>32 32 32 16<br>32 16 16 32 | 64 | 不可获胜，极大 |
| $3 \times 4$ | 32 32 32 32<br>32 32 16 32<br>32 32 16 32 | 64 | 不可获胜，极大 |
| $3 \times 4$ | 16 32 32 16<br>16 32 32 16<br>32 16 16 32 | 64 | 不可获胜，极大 |

| 场地大小 | 游戏过程中允许的最大方块数 | Goal | 结果 |
|---|---|---|---|
| $3 \times 4$ | 64 64 64 64<br>64 64 64 64<br>64 64 64 64 | 128 | 不可获胜 |
| $3 \times 5$ | 64 64 64 32 32<br>64 64 64 32 32<br>32 32 32 32 32 | 128 | 可获胜 |
| $3 \times 5$ | 64 64 32 32 32<br>64 64 64 32 32<br>32 32 32 32 32 | 128 | 不可获胜 |
| $3 \times 5$ | 64 64 64 32 32<br>64 64 32 32 32<br>32 32 32 32 32 | 128 | 不可获胜 |
| $3 \times 5$ | 32 64 64 64 16<br>32 64 64 64 32<br>32 32 32 32 32 | 128 | 可获胜 |
| $3 \times 5$ | 64 32 32 32 32<br>32 64 64 64 32<br>32 32 32 32 32 | 128 | 不可获胜 |
| $3 \times 5$ | 64 64 64 64 32<br>64 64 64 32 32<br>32 32 32 32 32 | 128 | 可获胜 |
| $3 \times 5$ | 64 64 64 32 32<br>64 64 64 32 32<br>32 32 32 32 32 | 128 | 不可获胜 |
| $3 \times 5$ | 64 64 64 32 32<br>64 64 64 32 32<br>64 64 64 32 32 | 128 | 可获胜 |
| $3 \times 5$ | 64 64 64 32 32<br>64 64 64 32 32<br>64 64 32 32 32 | 128 | 不可获胜 |
| $3 \times 5$ | 128 128 128 128 128<br>128 128 128 128 128<br>128 128 128 128 128 | 256 | 不可获胜 |
| $4 \times 4$ | 32 64 64 32<br>32 64 64 32<br>32 64 32 32<br>32 32 32 32 | 128 | 可获胜 |
| $4 \times 4$ | 32 64 32 32<br>32 64 64 32<br>32 64 32 32<br>32 32 32 32 | 128 | 不可获胜 |
| $4 \times 4$ | 64 64 64 32<br>64 64 64 32<br>64 32 32 32<br>32 32 32 32 | 128 | 不可获胜 |

| 场地大小 | 游戏过程中允许的最大方块数 | Goal | 结果 |
|---|---|---|---|
| $4 \times 4$ | 32 64 64 32<br>32 64 32 32<br>32 64 32 32<br>32 32 32 32 | 128 | 不可获胜 |
| $4 \times 4$ | 32 32 32 32<br>32 64 64 32<br>32 64 64 32<br>32 32 32 32 | 128 | 可获胜 |
| $4 \times 4$ | 64 64 64 64<br>64 128 128 64<br>64 128 128 64<br>64 64 64 64 | 256 | 不可获胜 |
| $2 \times 2 \times 2$ | 16 16 16 16<br>16 16 16 16 | 32 | 可获胜 |
| $2 \times 2 \times 2$ | 32 32 32 32<br>32 32 32 32 | 64 | 不可获胜 |
| $2 \times 2 \times 3$ | 128 128 128 128 128 128<br>128 128 128 128 128 128 | 256 | 可获胜 |
| $2 \times 2 \times 3$ | 256 256 256 256 256 256<br>256 256 256 256 256 256 | 512 | 不可获胜 |

## A.2 概率的近似下界

这些结果是使用浮点数计算得出的，因此由于可能存在舍入误差，不能视为已证明。在这些计算中，目标改为引理 1 所给出的目标。

| 场地大小 | 期间允许的最大方块数 play | Goal | 计算数据类型 | 近似下界 下界 on 概率-ity |
|---|---|---|---|---|
| $3 \times 4$ | 256 128 64 64<br>64 64 64 64<br>64 64 64 64 | 512 | 32位浮点数 | 0.999998 |
| $3 \times 4$ | 256 128 16 16<br>64 64 32 16<br>16 16 16 16 | 512 | 32位浮点数 | 0.999966 |
| $3 \times 4$ | 512 256 128 128<br>128 128 128 128<br>128 128 128 128 | 1024 | 32位浮点数 | 0.999544 |
| $3 \times 4$ | 512 256 32 16<br>128 128 64 16<br>16 16 16 16 | 1024 | 32位浮点数 | 0.998857 |
| $3 \times 4$ | 1024 512 128 128<br>256 256 128 128<br>128 128 128 128 | 2048 | 32位浮点数 | 0.957837 |

| 场地大小 | 期间允许的最大方块数 play | Goal | 计算 数据类型 | 近似 下限 界 on 概率- ity |
|---|---|---|---|---|
| $3 \times 4$ | 1024 512 64 64<br>256 256 128 64<br>64 64 64 64 | 2048 | 32位浮点数 | 0.955559 |
| $3 \times 4$ | 1024 512 64 16<br>256 256 128 16<br>16 16 32 16 | 2048 | 32位浮点数 | 0.910929 |
| $3 \times 4$ | 2048 1024 128 128<br>512 512 256 128<br>128 128 128 128 | 4096 | 32位浮点数 | 0.447708 |
| $3 \times 4$ | 2048 1024 128 128<br>512 512 256 128<br>128 128 128 128 | 4096 | 16位浮点数 | 0.432 |
| $3 \times 4$ | 4096 2048 256 256<br>1024 1024 512 256<br>256 256 256 256 | 8192 | 32位浮点数 | 0.000140 |
| $3 \times 5$ | 256 128 32 32 32<br>64 64 32 32 32<br>32 32 32 32 32 | 512 | 16位浮点数 | 0.997 |
| $3 \times 5$ | 256 128 64 32 32<br>64 64 32 32 32<br>64 32 32 32 32 | 512 | 16位浮点数 | 0.997 |
| $4 \times 4$ | 1024 512 64 16<br>256 256 128 16<br>16 16 32 16<br>16 16 16 16 | 2048 | 16位浮点数 | 0.997 |

## A.3 概率的精确下界

用于存储概率精确下界的整数类型是 uint32_t。

| 场地大小 | 期间允许的最大方块数 play | Goal | 严格的（以整数 算术证明的）概率下界 概率 |
|---|---|---|---|
| $3 \times 4$ | 256 128 64 64<br>64 64 64 64<br>64 64 64 64 | 512 | 0.99999617 |
| $3 \times 4$ | 512 256 128 128<br>128 128 128 128<br>128 128 128 128 | 1024 | 0.99954025 |
| $3 \times 4$ | 1024 512 256 256<br>256 256 256 256<br>256 256 256 256 | 2048 | 0.95793830 |

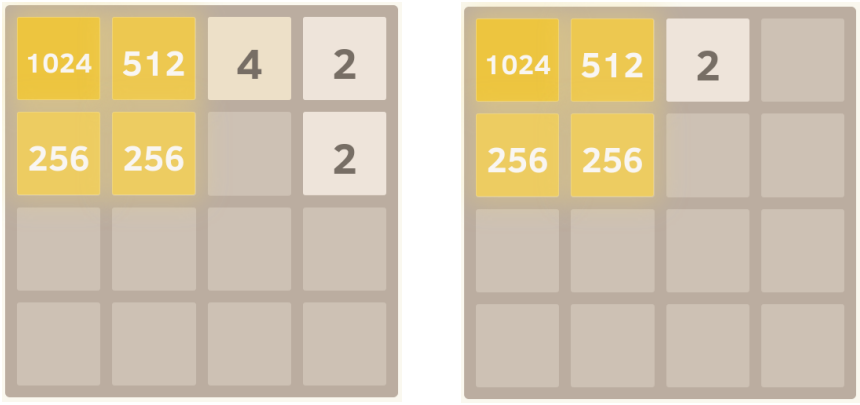
| 场地大小 | 期间允许的最大方块数<br>play | Goal | 严格（以整数<br>算术证明）的概率下界<br>概率 |
|---|---|---|---|
| $3 \times 4$ | 2048 1024 512 512<br>512 512 512 512<br>512 512 512 512 | 4096 | 0.45631551 |
| $3 \times 4$ | 4096 2048 256 256<br>1024 1024 512 256<br>256 256 256 256 | 8192 | 0.00012353 |
| $4 \times 4$ | 1024 512 64 16<br>256 256 128 16<br>16 16 32 16 | 2048 | 0.99969486 |

# B 引理 1 的证明

我们先向左滑动；然后，如果前两行中除了我们的四个方块外没有其他方块，就按右、上、右的顺序滑动即可获胜。否则，我们向上滑动，再向上滑动，再向上滑动，并尽可能持续向上滑动。此时，第一行右半部分的方块数量至少与第二行右半部分的方块数量相同。共有三种情况。

如果此时第一行已填满，就会出现类似图 5a 的情况；我们按右、左、右、左.....的顺序滑动，最终再向上滑动，即可合并两个 512 方块。

如果第一行恰好有三个方块，且第二行也恰好有三个方块，我们按右、上、右的顺序滑动即可获胜。




(a) 第一种情况的位置示例　　(b) 第三种情况的位置示例

图5

最后，如果第一行恰好有三个方块，第二行恰好有两个方块，那么我们就处于类似图5b的情况。在这种情况下，我们先向右滑动，然后尽可能多次向左滑动；如果此时第一行已经填满，我们就通过右-左-右-左-.....-上的操作获胜；否则，我们处于相同的情况（只是两个256变成了512，但这并不重要），不过第三行和第四行中所有方块的和增加了，于是我们重复这一过程。注意，这个和不可能无限增加。

■