

1 简介

我们要寻找有最多完美打击的种子。当然，每一场战斗都掉落完美打击的概率实在太小，可能不适合直接搜索。不过，这种小的线性增长在指数面前可以忽略不计，所以我们退而求其次。剧本如下：

开局选择涅奥的骨头，得到飞鞋和变牌遗物，同时得到愧疚（因为前期更简单）。变牌遗物不管是树叶药膏还是新叶都行，总之只要至少变出一张完美打击就够了。虽然后续也有可能获得完美打击，但是在开局筛选比遍历路径筛选快得多。

第二幕自然是佩尔，开始克隆。我们需要尽可能多的火堆。道中最多只有三个火堆，因为火堆只能在第 6 行到倒数第三行之间生成，而且无法相邻。有飞鞋的话，最多有 8 个道中火堆，可以复制 10 次。

在第二层的第一个火堆之前必须出现一个商店，售卖多利之镜，这样我们可以额外复制一次。在这个商店前还需要冰梆，能够让复制的底数从 2 变为 3。在第三层我们必须遇到事件“镜中倒影”，可以复制整个牌组。这样一共可以复制 12 次，大概 53 万张完美打击。游戏本体完全没法处理过来这么多，不过这就是后话了。

1.1 概率

阶段	通过率	累计分母
涅奥的骨头	1 / 10	10
骨头给出飞鞋、变牌遗物和愧疚	1 / 1911	1.9 万
至少变出一张完美打击	1 / 55.7	1.1×10^6
遇见佩尔，而且给出克隆	1 / 26	2.8×10^7
第 2, 3 幕能进入 8 个火堆	6.7%	3.6×10^8
事件要求	1 / 5800	1.7×10^{12}

火堆、事件之类的数据不太好算，是实测出来的概率。有些事件是有条件的，可能打乱随机数，我们搜索的时候认为所有单人事件都会出现，注意一下满足条件就可以了。

其实前四个阶段是非常容易计算的，只有后面两个需要 CPU 上的复杂搜索。不过能通过前四个阶段的种子也没几个，所以主要的压力还是在前面，而由于简单，所以好优化。把前四个放进 GPU 就可以了。下面就写了我具体是怎么做的。

2 优化

2.1 算法优化

涅奥遗物的的第一步是，抽取一个有代价的遗物放在第三个选项，然后再根据这个抽取前两个选项的遗物。然而，我们并不关心前两个选项，所以只需要把第一步单独拿出来就可以了。

然后是变牌。如果变牌的结果不对，我们就完全不需要去抽骨头给出的遗物。变牌只需要三次随机数，而骨头的遗物需要 27 次，所以是很赚的。

而且，骨头实际上是随机打乱 28 个遗物，然后取前两个，但我们只关心其中三个遗物，所以不必维护整个 28 元素的数组。我们只需要交换下标就可以了。所以一切都在寄存器里，不需要内存访问。

2.2 GPU 部署

如果直接把这个写成

```
if (!bone()) return;
if (!transform()) return;
if (!shuffle()) return;
```

那会导致 warp 利用率极其低下。毕竟第一步就只有 10% 的概率，32 个线程里只有 3.2 个在干活。而且，只要有一个线程在干活，warp 就不能去干其他工作。所以，第一步是拆分 kernel。

拆分好说，但是 kernel 之间怎么沟通就不好说了。肯定不能每个种子都 `atomicAdd` 一次，毕竟第一阶段的通过率还很高，如果使用 2^{28} 大小分块，那么会产生数百万个 `atomicAdd` 指令。所以，我们让每个 warp 用 `__ballot_sync` 统计可行种子的个数，然后 warp leader 用一次 shared atomic 在 block 内预留位置，并把所有种子写入 shared memory 里的缓冲。然后，整个 block 只需一次全局 `atomicAdd` 在 HBM 缓冲区里预留区间即可。

还有一些关于 ALU 使用的优化。我的显卡 RTX 4060 的 FP64 ALU 非常少，考虑到这里没有 NaN 和 Inf 之类的玩意，大肥鱼帮我写了一个只需要位运算的等价形式。而且，它还把一些 64 位拆分成了 32 位，去掉了一些 XXHash64 中用不到的部分（毕竟我们只随机一两次），来提高 ALU 的利用率。这些我就没仔细看了，相信大肥鱼就好了。

3 种子

进阶 10:

```
00084RVR6DEC  index=14536416037180  rests=5+3
0008HCJJ01PS  index=15198627738252  rests=5+3
0008ZJ1PRD06  index=16047505442194  rests=4+4
0009SYSLPBNL  index=17488394954340  rests=4+4
```

第四个种子一层发育好一些。第一个种子连冲四个精英，属实是不要命了。第二个虽然一层火堆多，但没走到火堆就死掉了。第三个也没好到哪里去，小怪太多了。

五十万张完美打击的视频已经发布了，建议三连一下（）