Hog Strategy Contest 2021

Instructions

> This contest is completely optional!

Download a blank `final_strategy.py` file and simple tests for correct formatting as a zip archive from QQ group or course website. Type `python3 ok` to run the provided tests.

Submit a `final_strategy.py` file containing a function called `final_strategy` and a `PLAYER_NAME`.

```
$ python ok --submit
```

The contest ends on **Sunday, November 7 at 11:59 PM**. We will use your **latest submission** before this date to determine the final results of the contest.

Your strategy should not take more than about **5 minutes** to run on a single-threaded machine with about 1GB of RAM. Specifically, the following loop should take no more than about 5 minutes:

```
for i in range(300):
    for j in range(300):
        final_strategy(i, j)
```

We won't enforce this strictly, but strategies that take significantly longer to run (hours, days) or require special hardware (e.g. GPUs) to run fast will be disqualified.

> 注意（详见下方Submission Rules）：
>
> 1. 可以使用的Python库包括标准库、NumPy和SciPy，不可以使用硬件加速。
> 2. 你的策略应该是确定性的，是一个纯函数，使用非纯函数可能会导致未知结果。
> 3. 如果你想使用Project01中的代码，需要将他们拷贝到提交的文件中。

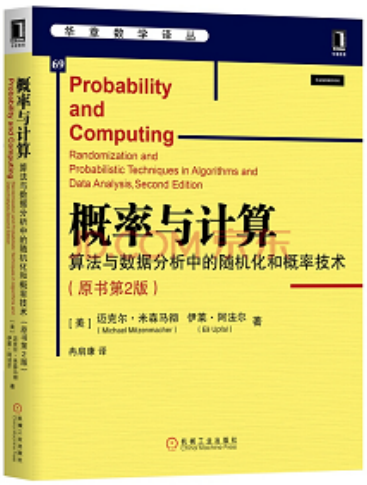
Leaderboard

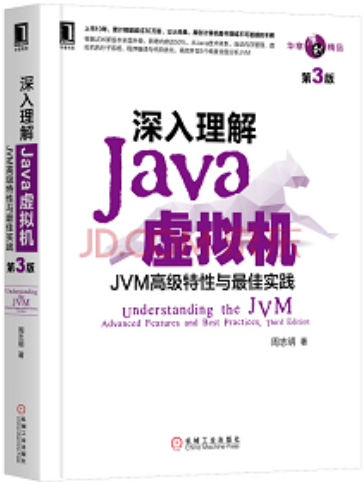
The leaderboard can be found here. It will continuously update as the contest progresses.

Contest Rules

Each submitted strategy will play against all other submissions in a set of 10^6 hog game scenarios. The outcome of this tournament will be determined by strategy alone and not the roll of the dice or flip of a coin. A submission scores a match point each time it wins exactly more than half of all scenarios. Ties count as losses.

We will rank submissions based on the number of matches they won. If two submissions won same number of matches, the one with small code size has a higher rank.





The top three submissions can choose a prize from the above books:

1. Modern Operating Systems: Principle and Implementation （现代操作系统：原理与实现，陈海波著）
2. Understanding the JVM: Advanced Features and Best Practices （深入理解Java虚拟机：JVM高级特性与最佳实践，周志明著）
3. Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis （概率与计算：算法与数据分析中的随机化和概率技术，Michael Mitzenmacher著，冉启康译）

Besides that, two lucky participants will receive two mysterious gifts.

Game Rules

After testing, we **removed Feral Hog rule** since rolling more dice will receive more points in average. We also **patched Picky Piggy rule** to avoid two players making no progress.

Here are the full set of rules:

- **Pig Out.** If any of the dice outcomes is a 1, the current player's score for the turn is 1.
- **Picky Piggy.** A player who chooses to roll zero dice scores the n-th digit of the decimal expansion of $1/21 (0.04761904\ldots)$ where n is the opponent's score. As a special

case, if n is 0, the player scores only 1 point.
    - If player A roll zero dice and scores 0 points, and then player B also roll zero dice and scores 0 points, player A loses.
- **Swine Swap.** Define *the excitement of the game* to be three to the power of the sum of both players' scores. After points of the turn are added to the current player's score, if the excitement's first digit and last digit are the same, the scores of both players should be swapped.

Size Penalty

To discourage you from just hard-coding your program output in your submission, we have introduced a size penalty for large submissions. Specifically, your `GOAL_SCORE` will not simply be 100, but will instead depend on the size of the `final_strategy.py` file that you submit, according to the following function:

```
def target_score(program_size_in_bytes):
    scaled_score = program_size_in_bytes / 10000 * 300
    return int(min(max(scaled_score, 50), 300))
```

Your program size will be displayed on the scoreboard.

Submission Rules

All strategies must be deterministic, pure functions of the current player scores. Non-deterministic strategies or strategies based on the history of the game will not behave as expected when submitted and will likely not do well.

- You may use most general-purpose libraries (standard library, NumPy, and SciPy) in your submission. Make sure all your logic is in the `final_strategy.py` file that you submit.
- You may import functions from the files provided in the `hog_contest` folder, but may not modify them. If you want to use any other logic from your `hog.py` project file, you must copy it into `final_strategy.py`.

If you have any questions about the rules, don't hesitate to post in QQ group.

Some Tips by TA

(小故事)

在2019年的时候，某位助教参与了中国计算机学会举办的CCF大学生计算机系统与程序设计竞赛。 有一题是设计一个针对不同优先级任务的调度器（也就是操作系统中的scheduler），评测程序会计算你的调度器在不同负载下的综合完成率作为性能指标。 这道题共有16个测试点，每个测试点正确得1分，性能指标最高的选手+5.25分，其余选手按照性能指标与最高的比值进行给分。

如果你们对操作系统有过了解，其实调度器是有很多实现方式的，Linux v2.6引入的完全公平调度器（Completely Fair Scheduler，CFS）大约有2100+行代码。 但是比赛哪有那么多时间呢？ 助教直接交了一个随机（其实随机算法是有名字的，叫做Lottery Scheduling），获得了90分，也就是说大约有最高分的90%性能。

回到我们的Hog Strategy Contest，你辛辛苦苦想出来的代码未必能取得高分，可能一个另辟蹊径或者投机取巧的方法反而能大获全胜。