Optional Contest: Hog Strategy

hog_contest.zip (hog_contest.zip)

A strategy is
A single function that makes
Ten thousand choices.

Instructions

> This contest is completely optional!

Download a blank `final_strategy.py` file and simple tests for correct formatting as a zip archive (hog_contest.zip). Type `python3 ok` to run the provided tests.

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

```
python3 ok --submit
```

> If you see an SSL error when trying to submit, that's an issue with your local installation of Python 3 on Mac OS X. Navigate to `Applications/Python 3.[Version]` and run `Install Certificates.command` by double-clicking it.

The contest ends on **Monday, September 20 at 11:59 PM**. We will use your latest submission before this date to determine the final results of the contest.

You can use the `compare_strategies.py` file to compute the exact win rate between any pair of two strategies using our server. Depending on server load, we may rate-limit or disable this functionality at any point, so consider re-implementing this file yourself locally!

Your strategy should not take more than about 5 minutes to run on a single-threaded machine with about 8GB 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.

Leaderboard

The leaderboard can be found here (https://hog-contest.cs61a.org). It will continuously update as the contest progresses. Go to the server log (https://hog-contest.cs61a.org/log) to see the status of your submission.

Contest rules

Teams can have one or two people. Each person can only be part of one team.

Each submitted strategy will play against all other submissions. The player to go first will be determined by a flip of a fair coin. We will exactly compute the expected win rate for each player, so that 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 has an expected win rate strictly above 50.0001%. We will rank submissions based on the number of matches they won. Ties count as losses.

The top three submissions will earn the following:

1. First place gets 3 points of extra credit.
2. Second place gets 2 points of extra credit.
3. Third place gets 1 point of extra credit.

Winners will also be publicly recognized in future iterations of 61A.

Game rules

To make things more interesting, we've introduced a new rule (Time Trot) in addition to the original set of special rules. Here are the full set of rules:

- **Sow Sad**. 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 1/7 (0.14285714...), where `n` is the opponent's score. As a special case, if `n` is 0, the player scores 7 points.

- **Hog Pile**. After points for the turn are added to the current player's score, if the players' scores are the same, the current player's score doubles.
- **Time Trot**. A turn involves a player rolling dice, and each turn is numbered, starting from 0. If a player chooses to roll a number of dice `k` on turn `n`, and `n % 8 == k`, then that player gets an extra turn immediately after the current turn. However, a player cannot get an extra turn immediately after another extra turn.

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 and target score will be printed when you submit, and will also 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 (such as the standard library, NumPy, or a C compiler) in your submission. Make sure all your logic is in the `final_strategy.py` file that you submit. If you are unsure whether you can use a particular library, ask us!
- 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 on Piazza.

Happy coding!