ICFP Programming Contest 2019

Lambda-coin and Lambda-chain

Version 1.0

ICFP Programming Contest Organisers

June 22, 2019, 10:01am UTC

In the past 24 hours, the ICFP contestants were very active on the Internet, discussing wrapping, drilling, teleporting, and other contest-related activities. This has not gone unnoticed by e-commerce providers, and as a result the contest organisers' mailbox got overwhelmed with emails entitled **"Best Deals on Your Wrapping Gear"** and similarly. Out of curiosity, we have checked some of these emails and indeed found quite interesting deals, which we decided to share with the contest participants:

Item	Price (LAM)
Manipulator	1000
Fast Wheels	300
Drill	700
Teleport	1200
Cloning	2000

In the table above, the prices for the boosters are given in *lambda-coins* (LAM), the world's soon-to-be most popular and stable digital currency. The characteristic feature of lambda-coin is that it is powered by Lambda-chain — a novel distributed consensus protocol that has been designed to be environmentally-friendly. In the past, digital currencies were predominantly obtained by the energy-intense mining process known as *Proof-of-Work*, which had no additional benefits besides minting coins, and was harming Earth's atmosphere, accelerating global warming. Since then, the scientific community managed to devise better protocols that award digital currency for solving timely problems without harming the environment.

The Lambda-chain mining mechanism, being extremely versatile, can serve multiple purposes, but at the moment, luckily for us, it has been set up (using a decentralised consensus mechanism), to award lambda-coins for solving worker-wrapper problems via the patented scheme known as *Proof-of-Wrap*®. While this activity is not mandatory for winning the contest, we encourage the teams to take part in Lambda-chain mining. This is a great way to earn some lambda-coins, which can be then used to buy boosters to improve the teams' solutions for the main contest.

1 Proof-of-Wrap® in a Nutshell

In the Proof-of-Wrap® scheme for mining lambda-coins, the participants (*a.k.a.* miners) contribute solutions to worker-wrapper tasks, while also proposing some tasks to solve. The mining protocol is semi-synchronised and works in rounds. In each round, a new *block problem* is announced. A block problem consists of the two components:

1. a standard *worker-wrapper task*, similar to those from this year contest,
2. a *block puzzle*, whose solution provides a candidate for the next block's task (*cf.* Section 1.1).

To win some lambda-coins, one has to participate in a round by submitting valid solutions to both components of the corresponding block problem.

1.1 Block puzzles

A *solution* for a *block puzzle* is a *worker-wrapper task*. Each puzzle is determined by a series of constraints, encoded as a string with the following format:

$$\begin{aligned} \mathit{puzzle} &::= \mathit{bNum}, \mathit{eNum}, \mathit{tSize}, \mathit{vMin}, \mathit{vMax}, \mathit{mNum}, \mathit{fNum}, \mathit{dNum}, \mathit{rNum}, \mathit{cNum}, \mathit{xNum} \ \#\ \mathit{iSqs} \ \#\ \mathit{oSqs} \\ \mathit{iSqs}, \mathit{oSqs} &::= \texttt{repSep}\ (\mathit{point}, ",") \end{aligned}$$

The first component (*bNum*) is a number of the current block (positive integer) that provides this puzzle. The second (*eNum*) is an *epoch* number, which determines the complexity of the puzzle (the epoch number increases every four blocks). The remaining components define the constraints to be satisfied. *tSize*–*xNum* are non-negative integers, while *iSqs* and *oSqs* are comma-separated lists of square coordinates. A task, aiming to solve a block puzzle, should satisfy the following constraints:

1. It should define a map M, which is a simple rectilinear polygon with non-negative integer coordinates, without self-intersections, containing no zero-length edges, and with no adjacent collinear triples of vertices. It should also follow the same convention *wrt.* the "polygon-on-the-left" encoding as the maps from the main contest.
2. The puzzle-solving task may have *no obstacles* (unlike the tasks from the main contest).
3. The initial position of the worker-wrapper should be within the map M.
4. M should be contained within the non-negative square with x/y-coordinates less or equal *tSize* (*the task is not too large*).
5. At least one of the maximal x/y-dimensions of M should be larger or equal than $\mathit{tSize} - \lfloor 0.1 \times \mathit{tSize} \rfloor$ (*the task is not too small*).
6. The area of M must be at least $\lceil 0.2 \times \mathit{tSize}^2 \rceil$ (*the task is not too sparse*).
7. The polygon M should have no less than *vMin* and no more than *vMax* vertices (*the task is not too boring and not too hairy*).
8. The task should contain precisely the numbers of boosters specified by the puzzle:
   - *mNum* manipulator extensions
   - *fNum* fast wheels,
   - *dNum* drills,
   - *rNum* teleports,
   - *cNum* cloning boosters,
   - *xNum* spawn points.
9. M should contain all squares with coordinates from *iSqs*.
10. M must contain no squares with coordinates from *oSqs*.

An example of a block puzzle (provided in a *.cond*-file) with a possible solution (a task described by a *.desc*-file) is available at

`https://icfpcontest2019.github.io/download/chain-puzzle-examples.zip`

A JavaScript checker of block puzzle solutions is available at

`https://icfpcontest2019.github.io/puzzle_checker/`

1.2 Lambda-chain submission rules

A submission from a team for a block problem is considered valid if it

- provides a valid solution to the block's task, and
- provides a task satisfying the constraints of the block's puzzle.

The initial block task is provided by contest organisers. After that, a task for a block N will be randomly selected from the puzzle solutions provided by the contestants in their submissions for the block $N-1$. According to the specification above, all block tasks, except for the very first one, will have no obstacles.

2 The Lambda-chain Protocol

2.1 Lambda-chain client

The source code for the Lambda-chain client is publically available and can be found at:

https://icfpcontest2019.github.io/download/lambda-client.zip

The archive contains a `README.md` file that explains how to install and use the client. Please refer to that document for instructions. If you have forgotten your team public ID (to set up your `PublicKey`), please consult your profile page. To see the help message, run

```
./lambda-cli.py --help
```

or

```
./lambda-cli.py <subcommand> --help
```

2.2 Testing phase

The Lambda-chain protocol is now online, but is in a special "testing" mode for the time being, to allow teams to become familiar with the protocol. For now, the only available block is block number 0, which is a test block for which no submissions can be sent. All Lambda-chain client commands are available, with the exception of the `submit` command.

The `submit` command will become available when the genesis block (block number 1) is released and mining commences.

2.3 Block format

The `getblockinfo` command returns all the public information describing the current block being mined, in the form of a JSON object with the following fields:

- `block` — block number (integer)
- `block_subs` — number of submissions received for this block thus far (integer)
- `block_ts` — block creation time as UNIX timestamp (float)
- `excluded` — list of team public IDs (as strings) which cannot submit for this block
- `balances` — dictionary from team public IDs (as strings) to LAM amount (integer)
- `puzzle` — block puzzle (string), in the format described in Section 1.1
- `task` — block worker-wrapper task (string), in the usual format

2.4 Mining

Mining will commence **June 22, 2019, 14:00pm UTC** (4 hours after the end of the Lightning Division) and will run for 41 hours, ending **June 24, 2019, 07:00am UTC** (3 hours before the end of the contest). After the mining period ends, the protocol will revert to "read-only" mode for the remainder of the contest. Previously earned lambda-coins can still be spent, but no more blocks are created and no more lambda-coins can be earned.

2.4.1 Mining rounds

The protocol proceeds in semi-synchronised rounds, with each round corresponding to one block. While round N is ongoing, teams can submit solutions to block N's task and puzzle via the Lambda-chain client. A round ends when either:

- (*normal deadline*) at least 25 submissions have been received and 15 minutes have elapsed since the block was created, **OR**
- (*late deadline*) 30 minutes have elapsed since the block was created

A team can make only one submission per round and **submissions are final** — if an incorrect submission is made, it cannot be corrected. A submission is valid only if *both* the solution to the block task and the solution to the block puzzle are valid.

Solutions are submitted as per the instructions provided in the client `README.md` of the client.

Final block. If at the end of the mining period, the last block in the chain has received at least one submission, new submissions for this final block *will still be accepted* (despite the mining period having ended) until either the normal deadline or the late deadline for the block is reached. This is to ensure a fair distribution of rewards.

2.4.2 Mining rewards

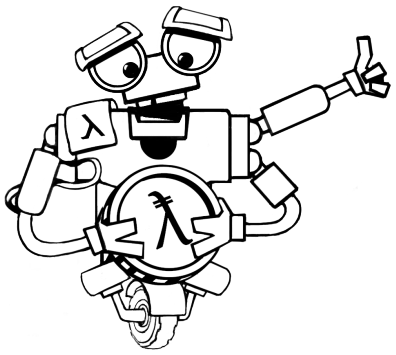
Each block has a bounty of 100,000 LAM which will be split among the teams with the 25 best solutions for the corresponding block task (or the teams who made valid submissions by the *late deadline*) proportionally to the quality of the solution. The quality of a solution is computed as a fraction $\frac{t_{best}}{t}$, where t is a number of time units taken by the solution, and t_{best} is the minimum time among all teams' submitted solutions for this block task.

A task for the next block is chosen randomly from the block puzzle solutions provided by the teams with **top 10 ranked block task solutions**. A team whose task is selected for the next block is awarded an additional 2,000 LAM and is *banned* from participating in the next immediate round of lambda-coin mining.

3 Lambda-Coins and the Contest Task Solutions

Teams can use lambda-coins earned by participating in the Lambda-chain protocol and accummulated on their wallets to purchase boosters, which can be used in solutions for the main contest tasks. The prices for boosters are defined in the table at the beginning of this document.

3.1 Enhanced submission format



When submitting solutions for contest tasks, a team has an option to include into their *.zip*-archive a number of files named *prob-NNN.buy* for selected problems *NNN*. These files act as *booster purchase descriptors* and specify boosters the team is willing to purchase to enhance their solutions for the corresponding tasks. A booster purchase descriptor is simply a series of (possibly repeating) booster codes, *e.g.*, `BBFFCCCLR`.[1] The purchased boosters are assumed to be available from the very beginning of the worker-wrapper execution and, thus, can be used immediately.

The checker and visualiser for the contest solutions have been now updated to take optional *.buy*-files. An example of a task solution relying on purchased boosters is also available:

`https://icfpcontest2019.github.io/download/purchasing-examples.zip`

[1] The booster purchase descriptor cannot include the spawn point symbol `X`.

The contest server will first check that the wallet balance of a team is sufficient to purchase boosters in *all* the descriptors of the submitted solution. If the team's wallet funds are insufficient, the entire submission will be rejected.

With each new successful submission, the team is *reimbursed for the previously purchased boosters*, before the new amount of LAM is charged. That is, a team can spend funds on its wallet multiple times, experimenting with different booster purchasing strategies, without the fear of running out of funds, as the wallet balance *will only grow* as more Lambda-chain blocks are being solved.

3.2 Unspent lambda-coins

The final balance of a team's wallet is computed after processing its last successful submission and subtracting the amount necessary to make the last purchase of the boosters. Any unspent lambda-coins on the team's wallet will be *converted into score points* using the ratio **1 to 1** and added to the final team's score. The Live Standings, when resumed, will be updated to reflect this.