ICFP Contest 2022

Specification Vo.o

Alperen Keles

1 Introduction

The mighty wizards of Lambda land has seen all your poses from last year and they were absolutely fascinated by them. They were so inspired by you that they have been discovering the secret arts of painting for the rest of the year, waiting for you to join them. Your task, if you are to accept it; will be to develop algorithms for robo-painters of the future. After all, there are so many paintings to do, and so little of us functional programmers to do them.

1.1 Updates

Note that there will be updates to this specification, and more problems will be released during the contest. This will happen at these specific times:

- 12 hours into the contest (new problems only, no changes to specification)
- 24 hours into the contest (after the lightning division ends)
- 36 hours into the contest
- 48 hours into the contest

1.2 Changelog

Any changes will be published here.

2 Problem Specification

The task is to **paint** a given **canvas** with the least **cost** and highest **similarity**.

As part of the task, you will be given;

- a set of **painting moves** applicable over the **canvas**,
- an **instruction language** to express your set of **moves**,
- a **cost function** for calculating the **cost** of each **move**,
- a **similarity function** for calculating the **similarity** of your **painted canvas** to the **target painting**.

As part of individual problems, you will be given;

- an initial **canvas**,
- a target **painting**.

2.1 Canvas

Canvas is an abstract 2-dimensional pixel space of **RGBA** channels.

Each **move** transforms the canvas in a different sense.

After all moves are applied to a **canvas**, it can be rendered to a **painting**.

Canvas is made out of **blocks**.

2.1.1 Blocks

Blocks are either a set of sub-blocks; or a structure with shape and color. In the eyes of the functional programmer, one might imagine such a definition.

```
data Block = Set<Block> | ConceteBlock Shape Color
```

The **initial canvas** only holds exactly one block, colored according to the individual problem.

Blocks are uniquely defined by their **block_id**.

The initial canvas has **block_id = 0**.

2.2 Painting

Painting is a concrete 2-dimensional pixel space of **RGBA** channels. For individual problems, you will be provided with **PNG** files for the paintings.

2.3 Moves

We present you with **6** different moves you can use to paint your canvases.

2.3.1 Cut Moves

Cut moves take a block, and some cut instruction over that block. Create new sub-blocks; preserving the colors.

2.3.1.1 Line Cut Move

A line cut move takes a block(defined by its block-id), an orientation(V for Vertical, H for Horizontal), an offset(the 1-d coordinate for the cut operation) and creates 2 sub-blocks of the given block.

Sub-blocks of line cuts are numbered from *bottom to top, or left to right*.

todo@Keles: put a picture here

2.3.1.2 Point Cut Move

A point cut move takes a block(defined by its block-id), an offset(the 2-d coordinate for the cut operation) and creates 4 sub-blocks of the given block.

Sub-blocks of point cuts are numbered from *bottom left using reverse clock-wise numbering*.

todo@Keles: put a picture here

2.3.2 Color Move

Color move takes a block, and some color on **RGBA** space. It changes the color content of the given block to the given color.

2.3.3 Swap Move

Swap move takes two blocks. It swaps the color contents of the given blocks.

Blocks must have the same shape to be swapped.

2.3.4 Merge Move

Merge move takes two blocks. It merges the blocks by creating a new block, adding these blocks to this new block as sub-blocks.

Blocks must be **compatible** to be merged. They must be adjoint, and their adjoint sides must have the same length. Informally; the merge of the blocks must create a new rectangle.

2.4 Instruction Language

Your task is to apply a set of moves to a canvas to similarize it to a given target painting. The way you will provide these set of moves is via submitting an **ISL(Instruction Set Language)** file for each problem. **ISL code** directly corresponds to the set of moves given above.

A BNF form of the ISL grammar is given in this file.

2.5 Cost Function

Each move has a `base + dynamic` cost.

Below is the table for **base cost for each move**

Move Type	Base Cost
Line Cut	2
Point Cut	3
Color	1
Swap	2
Merge	2

The function for dynamic cost is;

```
dynamic_cost(move, block) = base_cost x 1/size(block) x [alpha]
```

For each submission, these score are calculated for each move and aggregated for the total cost calculation.

2.6 Similarity Function

After processing all moves of a submission, the system calculates the similarity of the result to the target painting.

This is done via calculating **pixel difference** for each pixel and aggregating those results.

Pixel difference is calculated via **Euclidian Distance of HSV Values of Pixels**. Calculation is given below;

$$d_h = \frac{min(|h_1 - h_0|, 360 - |h_1 - h_0|)}{180}$$
$$d_v = \frac{|v_1 - v_0|}{255}$$
$$d_s = |s_1 - s_0|$$
$$distance = \sqrt{d_h \times d_h + d_s \times d_s + d_v \times d_v}$$

3 Submission

You will submit **ISL code** over panel inside the portal. For each problem, you will have 20 tokens for submission. Each submission will spend 1 token, and tokens will be refilled to 20 at the start of each hour.

4 Deadlines

As traditional, the contest will have a Lightning Division spanning the first 24 hours. To qualify for the Lightning Division prize, submit your ISL codes by September 3, 2021, 12:00pm (noon) UTC.

To qualify for the Full Division prize, submit your ISL cides by September 5, 2021, 12:00pm (noon) UTC.

In order to qualify for any prizes, your source code must be submitted by the end of the contest as well. You can do this through the web portal.

5 Determing the Winner

We will use the same procedure to determine the winner in both the lightning and full divisions, ranking the teams by cumulative score, computed as the sum of scores for each task.