B4 - Functional Programming

B-FUN-400

Bootstrap

image compressor

2.0

Part 1 - Getting used to Haskell

The core task of the project is implementing a kmeans algorithm. It involves recursively computing Voronoi clusters around moving centroids. But before to be able to run such an algorithm, let us practice a few intermediate steps with Haskell.

Distance

Your program should be able to compute the euclidian distance between two vectors.

```
~/B-FUN-400> distance (3,4,5) (3,8,2)
5.0
```

Before implementing a complicated parser for "(x,y,z)", look around: maybe there's a neat trick to do it...

Max by distance

Your program should be able to read a file (first argument) and find within this file the vector with smallest euclidian distance to a given other vector (second argument).

```
~/B-FUN-400> head list
(33,18,109)
(33,17,109)
(35,18,111)
(35,21,109)
(38,21,112)
```

```
~/B-FUN-400> closest list (34,18,112)
(35,18,111)
```

Expel outliers

Your program should be able to read a file, remove sequentially the 5 vectors whose sum of distance to others is maximum, and return the remaining list.

The distances must be recomputed each turn, since the list has changed as one vector has been removed

Part 2 - Image handling

At this point, you are probably experienced enough to handle the project without extra training. However, you might want to actually handle real images. We are going to discover the image processing library to be used for the project, JuicyPixels.

Image reading

Using the JuicyPixels library, read an image (PNG8 for instance, but it is not mandatory), and dump its pixels in a file.

I suggest you have a look at `readImage` , `convertRGB8` and `pixelMap`.

Image writing

Once all the pixels of your image dumped in a file, sort them regarding a criteria of your choice (their red component for instance).

Then re-encode all these pixels into a new image.

If you're still valiant, palettize this new image.