Genetic Algorithm: Draw a picture according to the given picture

Yuhan Wang
Yumeng Wang NUID: 001828064
Group number: #317

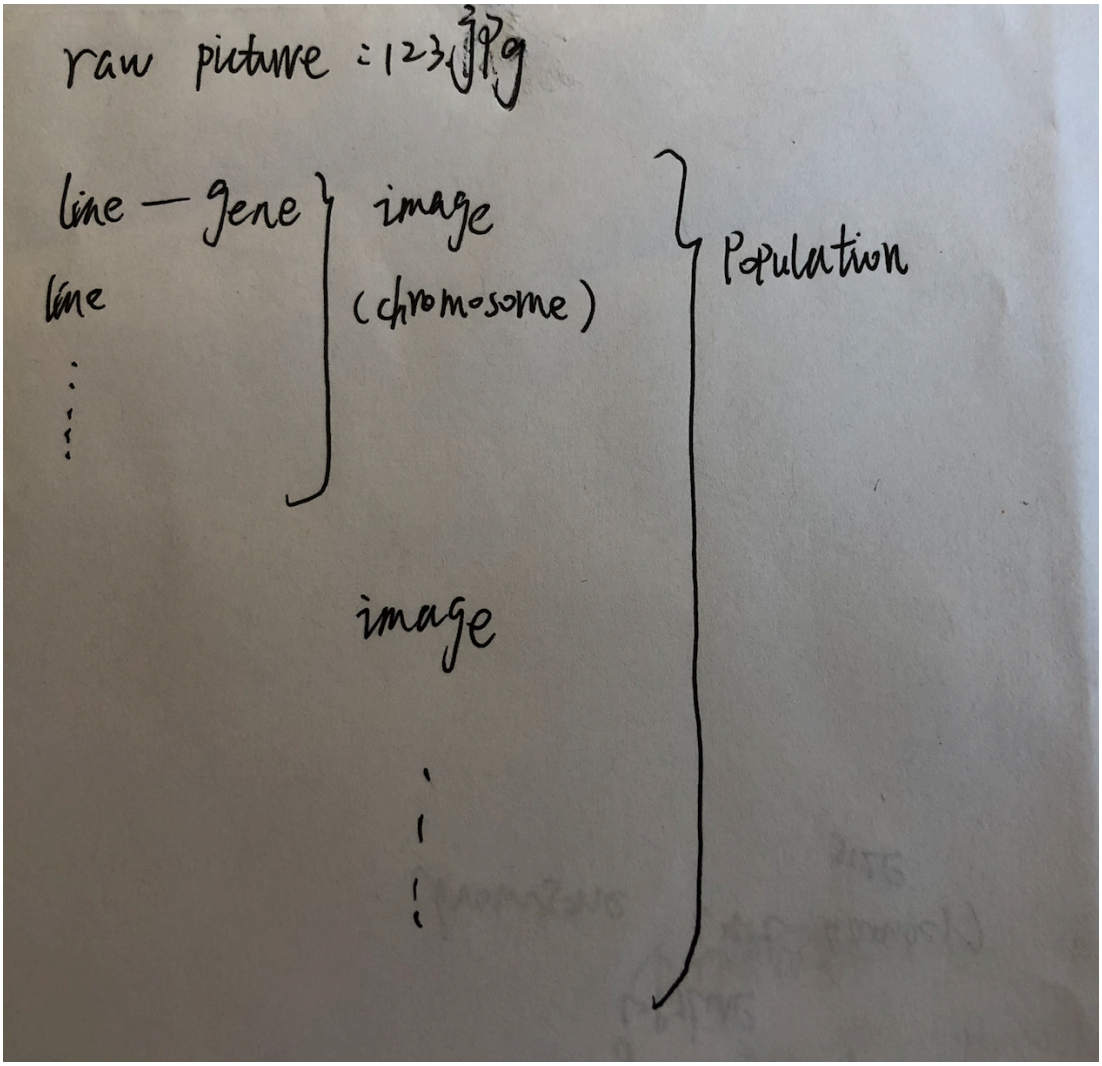
Problem

We use genetic algorithm to draw a picture according to the given picture. The problem proposes the situation: given a raw picture which may have an animal or figure on it, we need to draw a picture according to the information given by the raw picture. After our discussion, we find that this problem can be transformed into:

1. drawing a picture as a panel which has the same length and width of the raw picture
2. finding the RGB value of the pixel point on the raw picture
3. setting the same RGB value of the pixel point to the corresponding position on the new panel.

Implementation Design

1. Genetic code: we use the function Line (length, width) to represent a line in which (x1, y1) and (x2, y2) set as the start and end point of a line. In the function, x1, y1, x2, y2 are randomly produced according to the length and width of the panel.
2. Gene expression: the function Line (l, w) represents a line of an image on the panel.
3. Phenotype: in our design, a chromosome composed of 200 lines is an individual. That is to say, an image composed of 200 lines is a candidate solution to the problem.



4. Fitness function: we use a 2-D array to store the RGB VALUE of the pixel point on the panel. We divide the picture into length* width points and get each RGB value of these points. We try to compare points on panel(pixel[i][j]) with points(picture[i][j]) on raw picture. However, we may miss to compare the RGB value between two points. Thus, based on one point, such as pixel[i][j], to draw a square, which means we connect pixel[i-1] [j-1], pixel[i][j-1], pixel[i-1][j] and pixel[i][j] . Then we compare each of these four points' RGB value with the corresponding RGB VALUE of picture[i][j] on the raw picture, which can more accurately represent the accuracy of RGB value near this point. In the process of comparison, there are four possibilities:

Firstly, we paint black while the target point is black and this is the ideal choice so that we add 1 to the former fitness.

Secondly, we paint black while the target point is white so that we minus 1 to the former fitness.

Thirdly, we paint white while the target point is black so that we minus 0.5 to the former fitness. (Why minus 0.5 instead of 1? According to multiple tests，I found the result of comparison have more deviation in this situation. In order to reduce it, I minus less than previous situation. This is also the case in the following situation)

Fourthly, we paint white while the target point is black and this is the second choice so that we add 0.5 to the former fitness.

When we compare pixel[i][j] with picture[i][j], we judge based on the above criteria. However, when we compare pixel[i-1] [j], pixel[i][j-1], pixel[i-1] [j-1] with picture[i][j], we add or minus less in above 4 situations. Because the result of comparison has more deviation in this situation, we should reduce error.

We also find that the value being added or minus is a key essential to the result because if we do not set a reasonable value, the program will be stuck easily. Thus, above values are obtained from multiple experiments.

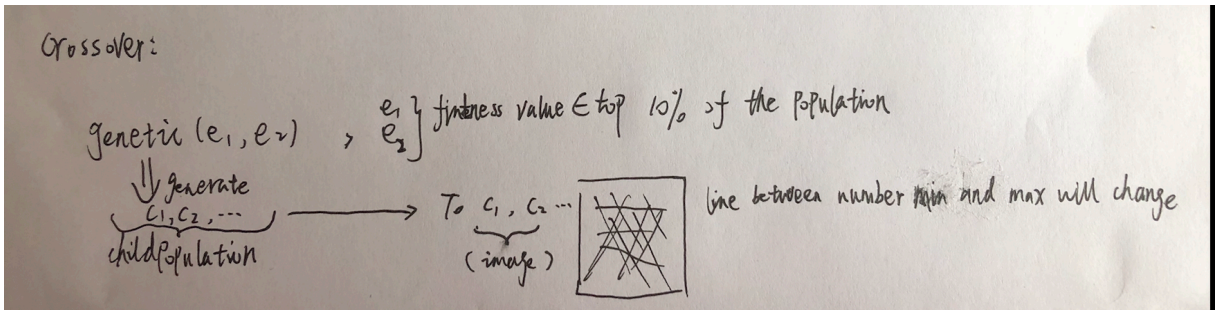
5. In our design, a population is composed of 250 individuals which means a population is a collection of 250 images.
6. Sort: we get the fitness value of each image in the population, and sort their fitness in a descending order.
7. Crossover: Choosing the top 10% fitness value and choose two of them randomly as parents to generate the next generation and we set the maximum iteration number to be 20000.
8. Mutation: we set a mutation rate to 0.8 to guarantee the variety of the next generation since we find that as the program processes, it is likely to be stuck at some point without a reasonable mutation rate. When the mutation rate is smaller than the maximum mutation rate then the line position on the panel

will mutate. After 2000 generation, we can basically get the outline of the graphics, so we can reduce the mutation rate to increase the possibility to get better population. (You can see near 2000 generations, the fitness score has little change. It means these population are relatively close to picture.)

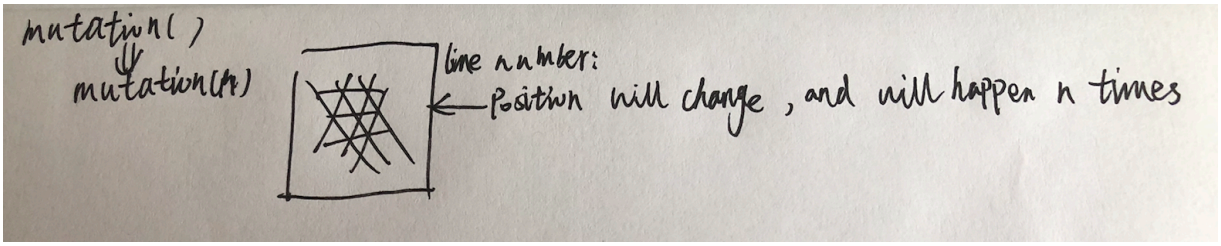
```
Generation 1998 Best score 28297.180000057026
--------------------------------
Generation 1999 Best score 28297.180000057026
--------------------------------
Generation 2000 Best score 28297.180000057026
--------------------------------
Generation 2001 Best score 28297.180000057026
--------------------------------
Generation 2002 Best score 28297.180000057026
--------------------------------
Generation 2003 Best score 28297.180000057026
--------------------------------
Generation 2004 Best score 28297.180000057026
--------------------------------
```

Implementation details:

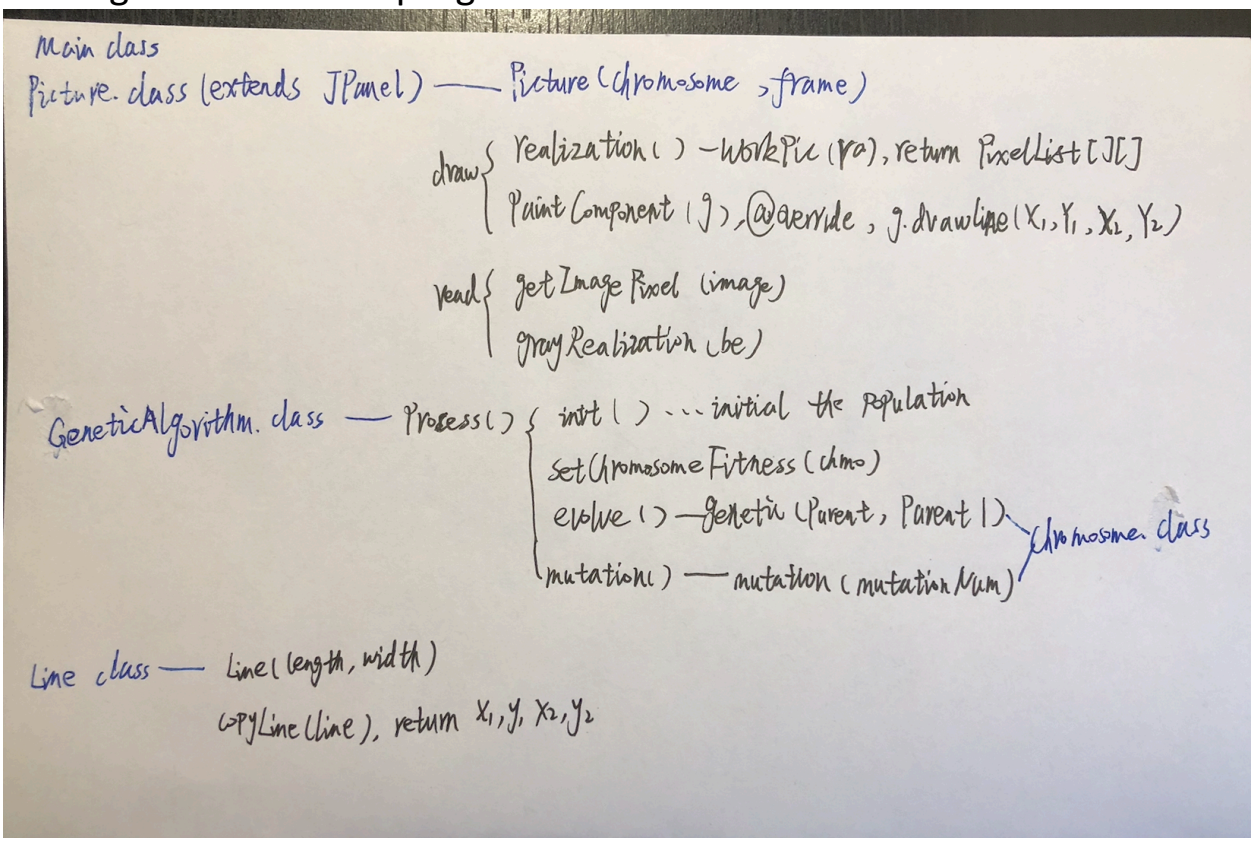
1. Crossover:



2. Mutation:

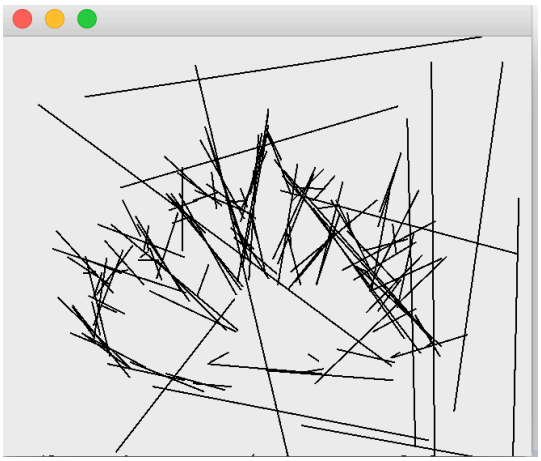


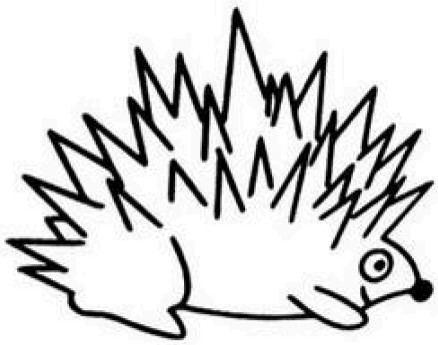
The logic of the whole program:



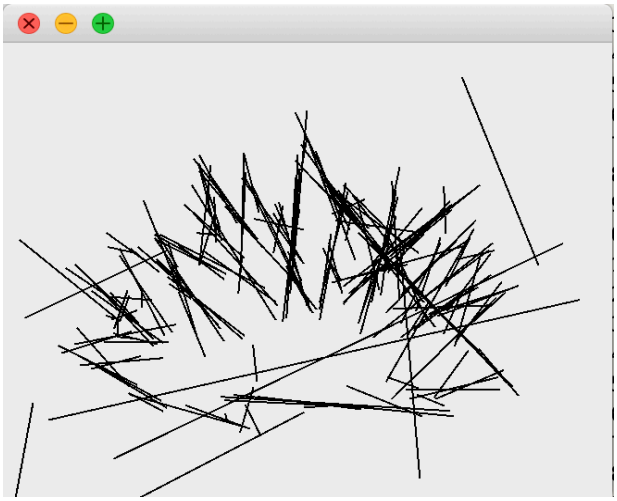
Results:

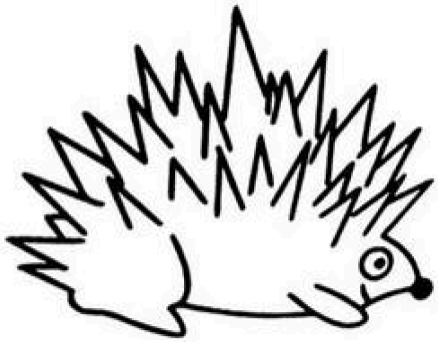
After running for 2 hours, we get the left picture:





After running for 10 hours (20000 generation), we get the left picture:





Output of the program

Result of test runs

All the Unit Test pass.