INFO6205 Final Project Report

Conway's Game Of life

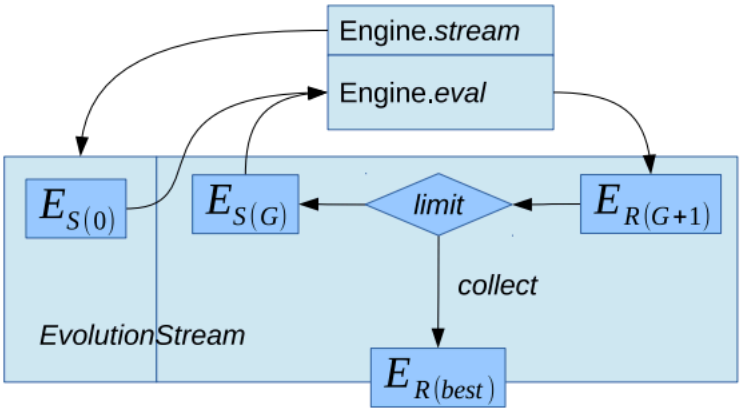
Team: Manlun Song(001834966); Yupeng Zhang(001051130); Zhongxia Huang(001054050)

Problems: Conway's Game Of Life.
We are trying to find out the best start pattern which can go through Conway's game of life. The more generation, the better start pattern.

Implamentation: Jenetics.
The best way to solve this problem is Gene Algorithm. And .Jenetics is an advanced Genetic Algorithm, respectively an Evolutionary Algorithm, library written in modern day Java. After few days reading and learning, we found Jenetics can provide powerful and efficient tools for solving game of life problem.

Base concept implamentation:

- The architecture of Jenetic GA:
  The basic metaphor of the Jenetics library is the Evolution Stream, implemented via the Java 8 Stream API. Therefore it is no longer necessary (and advised) to perform the evolution steps in an imperative way. An evolution stream is powered by—and bound to—an Evolution Engine, which performs the needed evolution steps for each generation; the steps are described as below.



- Genotype:
  Matrixes that generated randomly by jenetics.engine.Codecs;
  At the beginning, we tried to use binary digit as the genotype and transferred them into the data structure which can be passed into GAME class. It worked well. But as we learn deeper of Jenetic, we found there is a better choice: Codecs. Codecs contains factory methods for creating common problem encodings. And the start pattern can be regarded as a matrix which only have 0 and 1. 1 means there is a cell and 0 means

nothing there. Therefore, we decided use Codecs to randomly generate Genotype that we need.

- Phenotype:
  Set of Points. We transfer matrix to Points. The two-dimension indexes of the 1 elements in matrix are used as Points' coordinate.

- Engine.eval(Fitness):
  We use the number of generations that Phenotype can last in Conway's game of life. The more the better. In our eval method, we pass a matrix into it, and transfer the matrix to a set of points. And call Game.run() to get each matrix's fitness.

- Expression:
  Transform Matrixes to Points. We call toPoints() method.

Jenetics Engine Implamentation:

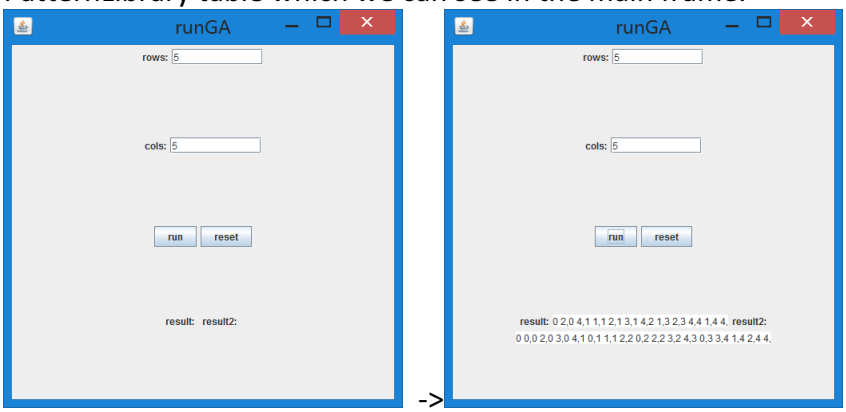
- Engine:
  We set Optimize to make GA maximize the fitness function. And instantiate a Mutator so that our GA can mutate the pattern(mutation probability: 0.5). Finally set the population size as 1000. The population size represents the number of start patterns in each GA generation.

- Engine.stream
  We set the limitation, selector and collector for Engine.stream. The number of GA generations are limited in 100. In order to make it more efficient, we put an execution time(500ms) limitation on every stream. And collector to collect the best Phenotype.

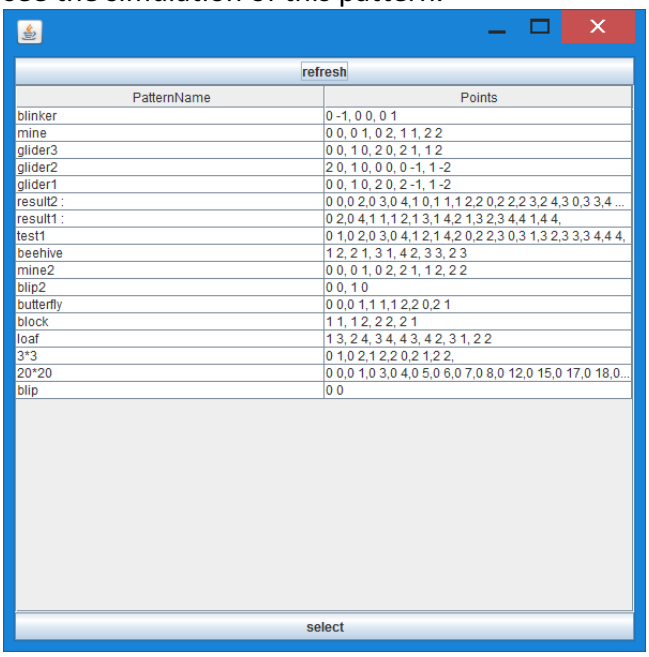
UI Implamentation:

- UI:
- 

Step1:
click “GA” button in the main frame, then we come to the runGA frame.
set the extent of start pattern click “run” to run GA and we can generate start patterns randomly and return a best start pattern.
The best start pattern will be stored in both the result.txt file in the source file and the PatternLibrary table which we can see in the main frame.

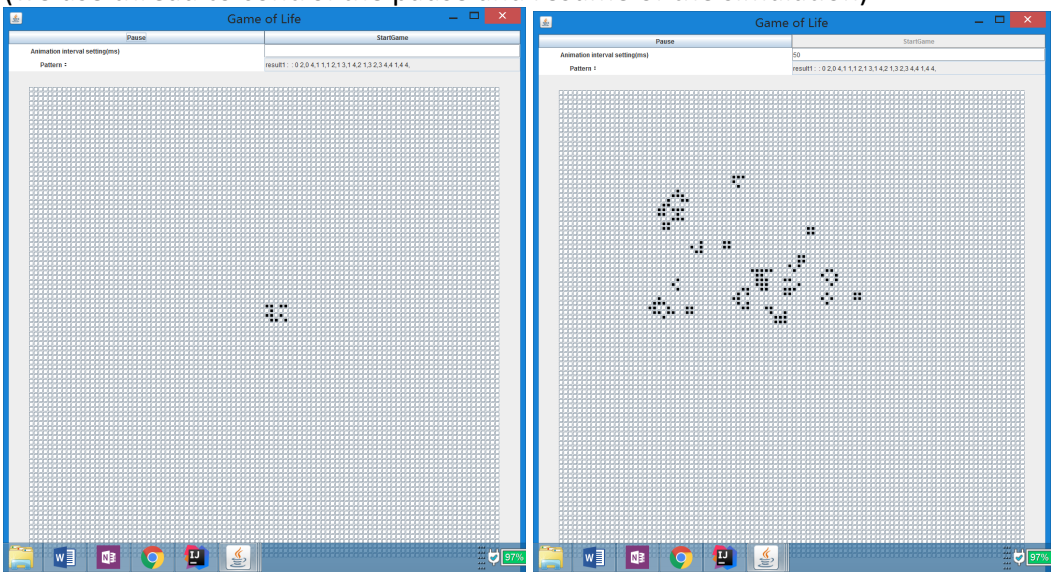


Step2:
Click “patternLibrary” button and we can see a pattern table. Select the pattern you want to run and then click “select” button, then we will come to the playgame frame to see the simulation of this pattern.



Step3:
In the playgame frame, we can set the animation interval(ms) of simulation in the textfield, then we click strat to watch the simulation.
During the simulation, we can click “pause” to pause the game and “resume” to resume the game.

(we use thread to control the pause and resume of the simulation)



Multithreading Implamentation:

We use Callable and Future classes to implement multithreading, which can run two genetic algorithms at the same time.Whenever you click the start button, a thread pool will be created and two threads will be assigned to the two genetic algorithms. When the program finishes running, it will output two different starting parts and close the thread pool.

Results:

Each time we found a start pattern, we write it into a txt file, so that we can track it.
The start patterns that can last over 1000 generation in Conway's game of life:
seed:111010101100 Points:0 0,0 1,0 2,1 1,2 0,2 2,
seed:011001111100 Points:0 1,0 2,1 2,2 0,2 1,2 2,
seed:00100111111000011000 Points:0 2,1 1,1 2,1 3,2 0,2 1,2 2,3 3,
seed:011110010110100111111000011000 Points:0 1,0 2,0 3,0 4,1 2,1 4,2 0,2 2,3 0,3 1,3 2,3 3,3 4,4 4,
seed:00110010101000001011011011010001000 Points:0 2,0 3,1 1,1 3,2 0,3 0,3 2,3 3,4 0,4 1,4 3,4 4,5 1,