Genetic Algorithm: Maze Problem

Team No. 201

Ruixing Li 001886987

Chen Qiu 001877500

Problem Description

Generally, Given a maze, indicating the starting point and the ending point, finding a valid and feasible path from the starting point to the ending point is a maze problem.

In our project, we make a little change from maze problem, the player is supposed to find a gem in the maze first, use the gem to skip the trap and find a way to the exit.

GA Components

GA code typically has the following functions:

1.Genetic code

Initially, we set 100 players in maze and the maximum number of generation is 30000.A play has 200 genes which means that he/she can move 200 steps at most, a gene is generator randomly from parent.

2.Fitness Function.

Probably the most important bit of a GA next to figuring out how to encode the population's genes. Well, in our project, gene represent the move direction down, left, up, right encoded as 1,2,3,4 in the maze

We applied BFS to get the shortest distance as well as the path among gem, trap, and exit position, if a player is able to collect more gems, correspondingly he/she will get more score in the process.

Additionally, if any of the moves go over 200 steps, we impose a penalty on the score. Which mean that the player should end the game and lose all score right away.

3.Survival Function.

We applied priorityqueue and rewrite the compare function to get the top 5 highest score which we calculate from fitness function for the generation, these candidates will play the role as parents to get offspring

4.Crossover.

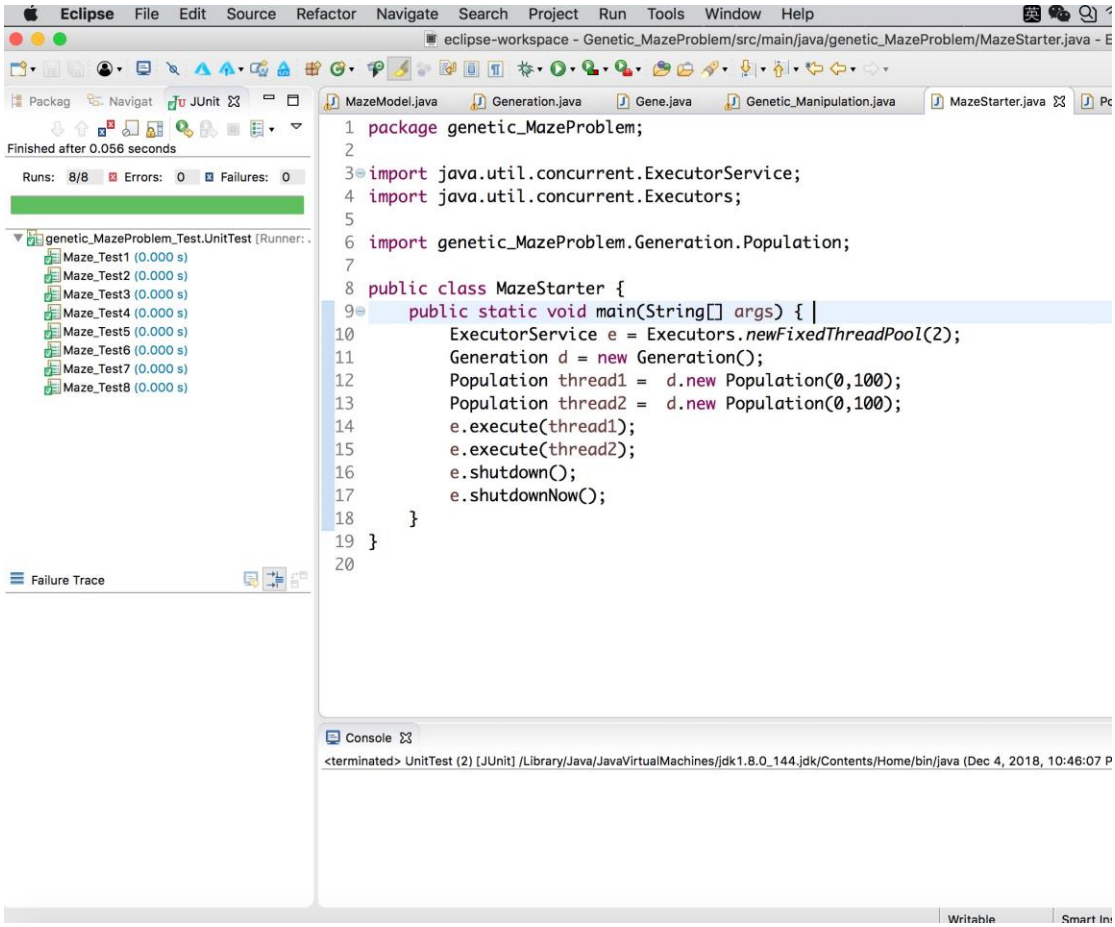
For each generation, we select the top 5 players who get the highest scores as parent to generate offspring, to get enough offspring, we do crossover multiple times

5.Mutation.

Some of the genes have a random chance of mutating when producing offspring. Mutation plays a pretty important part of the GA. If there is no mutation then the solution will kind of all end up being the same with no chance of breaking free for other potential solutions. In our project, we set

mutation rate to 0.005, which the probability to mutation for each gene is 0.5%. So roughly 91% of the time we should have a mutation occurring for a player.

4.Unit Tests



5.Parallel processing

We conducted two threads simulating two population of same creature with base amount of 100 each. And merge the result together in priorityqueue to get parents.

6.Conclusion

From the screen shot we can see that finally the play's status turns to 3

which means that he/she get to the exit successfully, you can see the thread state and generation state from the console.