Overcooked

Genetic Algorithm for learning robot behavior

Qian Liao & Min Hu

Group 208

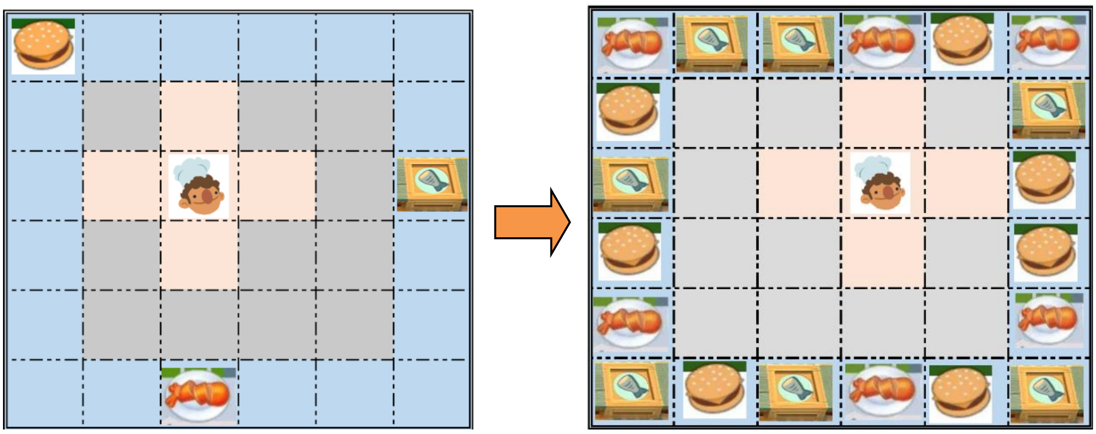
2018 Fall

Genetic Algorithm Project Report

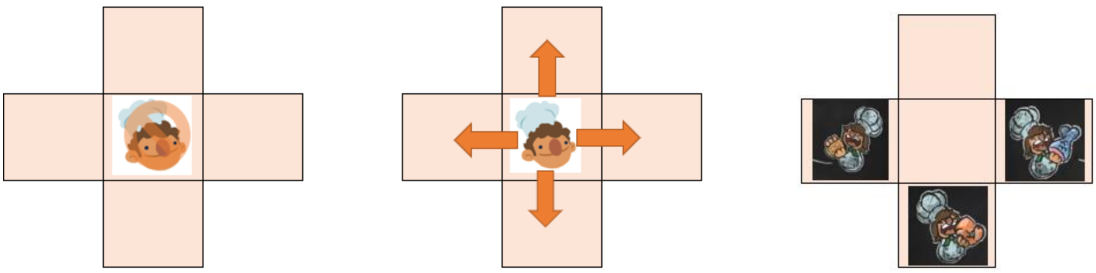
Problem Description

Overcooked is a cooking simulation game. Players in Overcooked take on the role of chefs in a kitchen, preparing meals via preparation of ingredients, cooking, serving, and cleaning up all while under a time limit to complete as many dishes as possible.

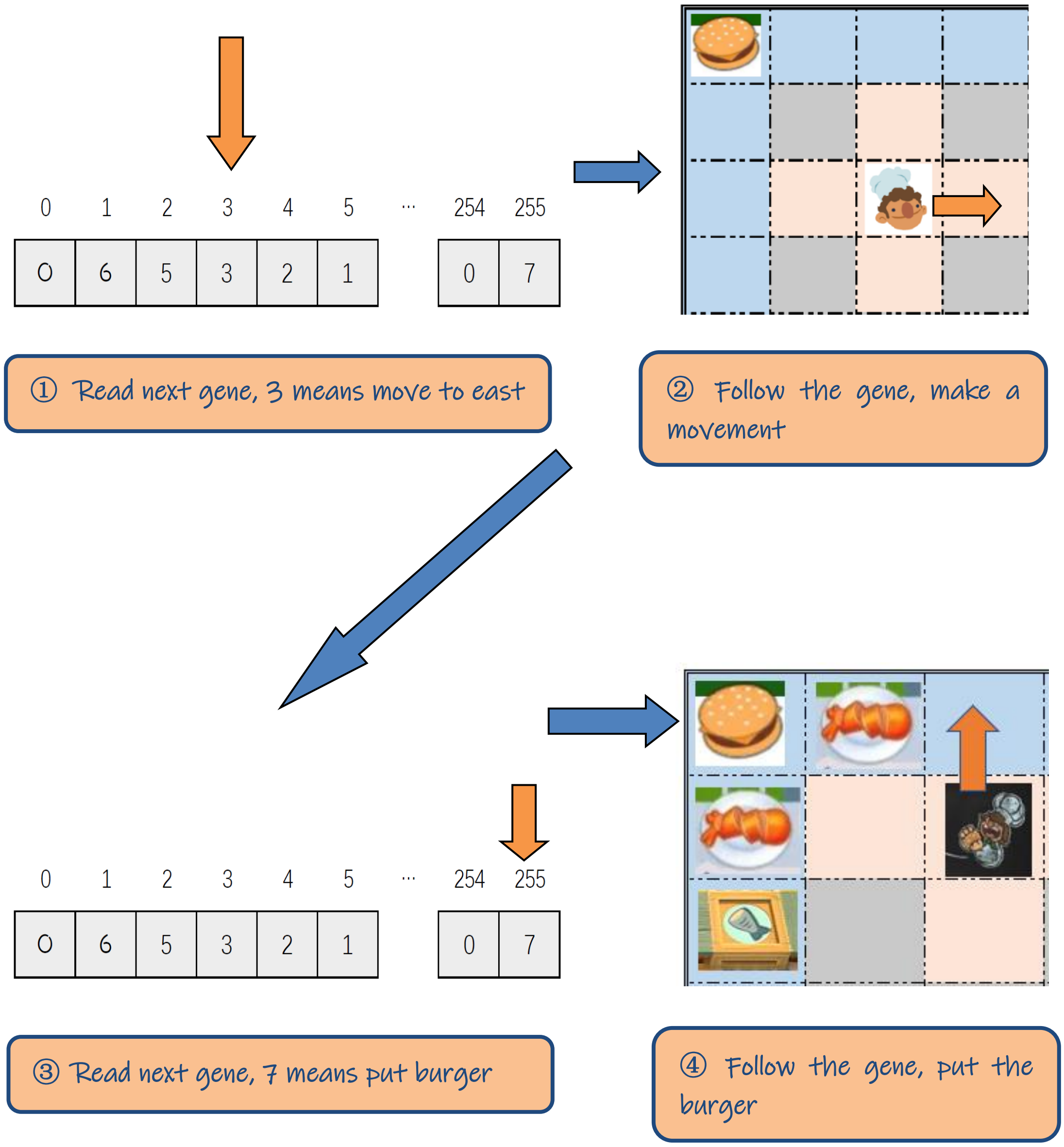
To use genetic algorithm to learn robot, we choose to analyze behaviors of chefs. We use a $N \times N$ map to mock the restaurant of Overcooked where cooks stand at the central areas and tables were put around cooks. In this $N \times N$ map, we use external cells mocking tables, and chefs should put three kinds of food at correct place.



One chef has eight behaviors which are stay in the cell, move one step to north, south, west, east, and put salmon, shrimp and burger on the table.



For Genetic Algorithm, these behaviors can consititute the gene of each chef. Each time before the chef makes a movement, he will read his gene, follow the number he get and do that corresponding behavior written in his gene.

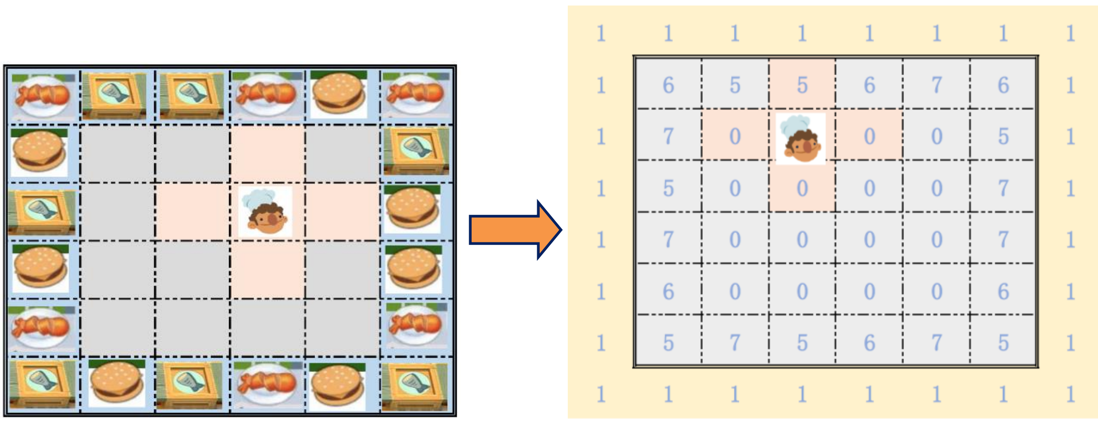


Implementation Details

A. Restaurant Design

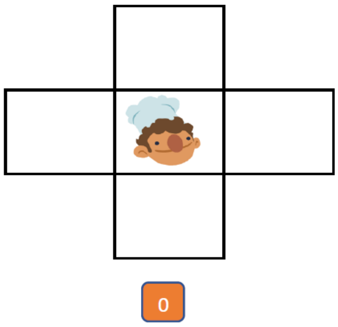
We mock restaurant as a $N \times N$ map, each cell of map can represent a part of a restaurant. We use 1 represents the wall of restaurant, 0 represents the empty areas for chefs to move, 5 represents tables prepare for salmon, 6 represents tables prepare for shrimp, 7 represents tables prepare for burger.

The reason we use numbers to replace each cell is to help chefs remember the edge of a restaurant, the areas them can move and the correct place to put corresponding foods.

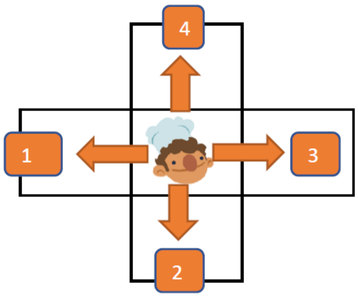


B. Chef's Behaviors Design

We use number 0 to 7 stand for chefs' eight behaviors. These numbers which represent behaviors of chef is corresponding to the numbers of map. For example, number 5 represent chef putting salmon, it also represent tables waitting for salmon on the map.



0 : chef stay in the cell

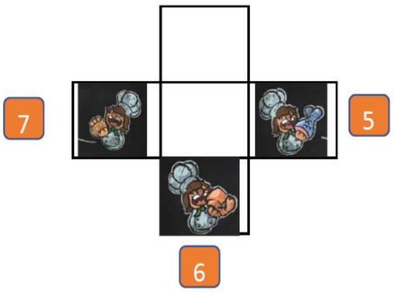


1 : chef move to west

2 : chef move to south

3 : chef move to east

4: chef move to north



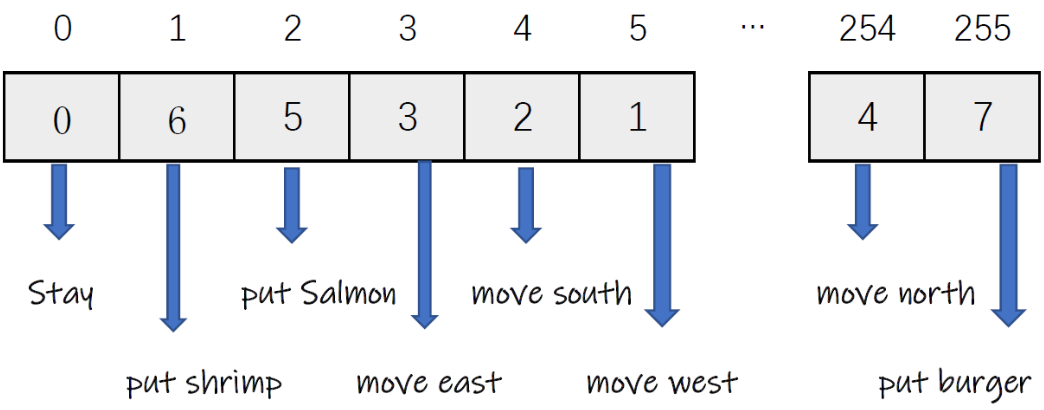
5 : chef put Salmon on the table

6 : chef put Shrimp on the table

7 : chef put Burger on the table

C. Gene Design

We design genes inside a chef as a list of genens to decide their behaviors. For a gene of N bases, there are 4^N different possible 'alleles', so we decide each gene sequence has 256(= 4^4) genes. Each gene has 8 possibilities which are number from 0 to 7 and the position of each gene in a gene sequence is randomly sort from 0 to 255.



D. Fitness Function

To check whether a chef is a good candidate, we decide to calculate scores for chef's behavior. Chefs can get scores only when they put foods at the correct place which is corresponding to tables of restaurant.

a) Put salmon correctly → + 30 scores

b) Put shrimp correctly → + 30 scores

c) Put burger correctly → + 30 scores

d) Put wrong food → - 4 scores

e) Move and Stay → + 0 score

To choose greate candidates of each generation, we set a threshold value to screen out excellent descendants. For each generation, each chef finished all behaviors written in their gene and we will calculate the best score and average score. For those chefs whose scores are best scores will be chosen as the best genes in each generation.

Evolution

A. Initialization

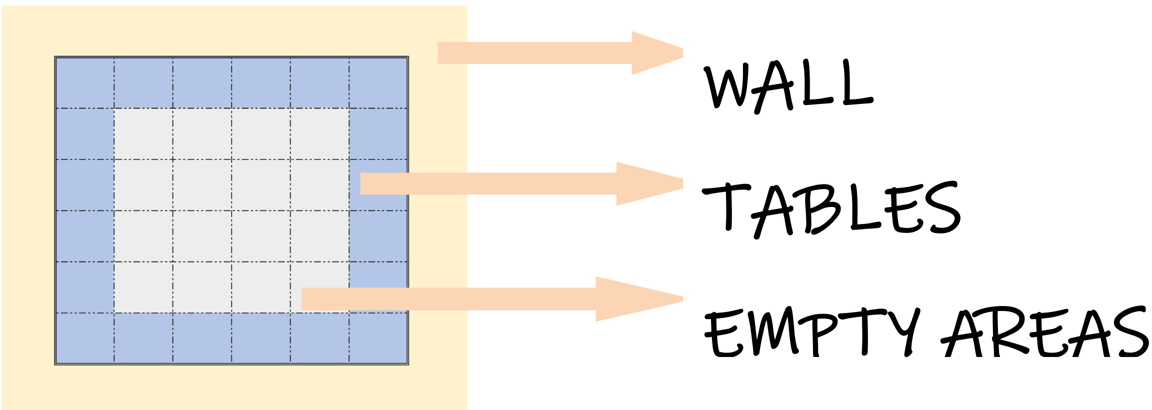
a) Generation Initialization

We set up the number of generation is 2500 and the population of each generation is 200. Each individual has a gene sequence with 256(=4^4) genes and each gene is random number from 0 to 7.

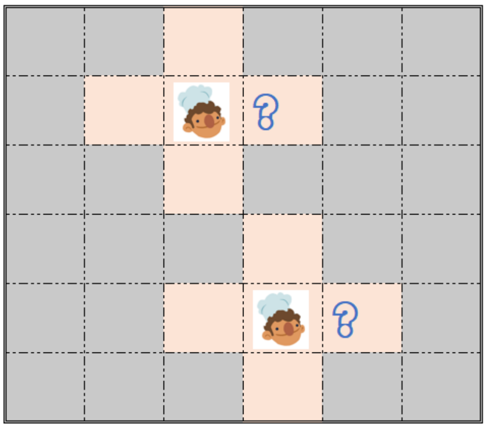
0	1	2	3	4	5	···	254	255
0	6	5	3	2	1		0	7

b) Environment Initialization

We generate a 10×10 map mocking the restaurant and set up the external cells with number from 5 to 7 randomly mocking tables with orders. Each chef has his own restaurant which means we initialize a new 10×10 map for each new indivitual before his movement.

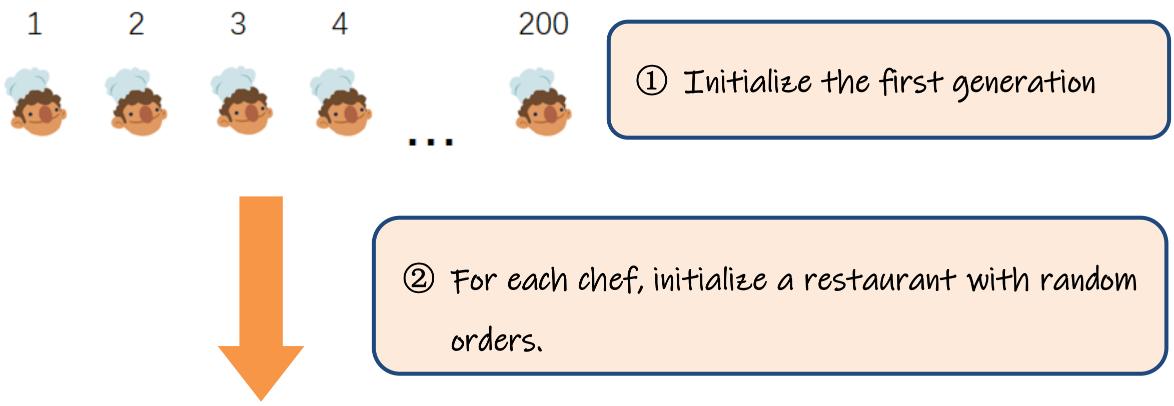


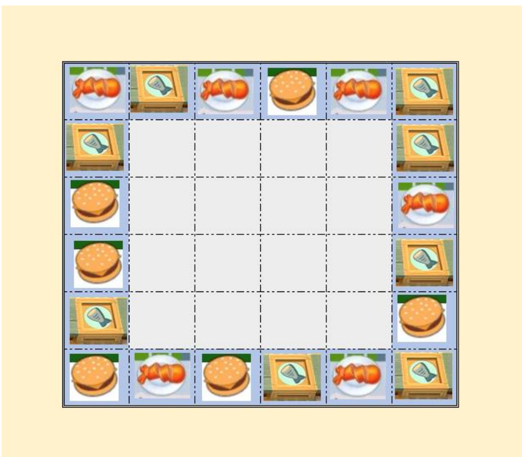
c) Position Initialization



To analyze chefs' behavior, we need to provide an initialization position to chef on the map. To avoid influence of environment, this position should be random. So when the chef initialize, he will get random coordinates. To insure fairness, each chef generates at random position and belongs to his own map which has same size.

B. Evolution Process





Each chef in a generation is put into a map then deliver correct food to corresponding orders.
Chefs' behavior guided by their gene.
According to their behavior they will get some scores calculated by the fitness function.
More correct foods they delivered, more scores they got.

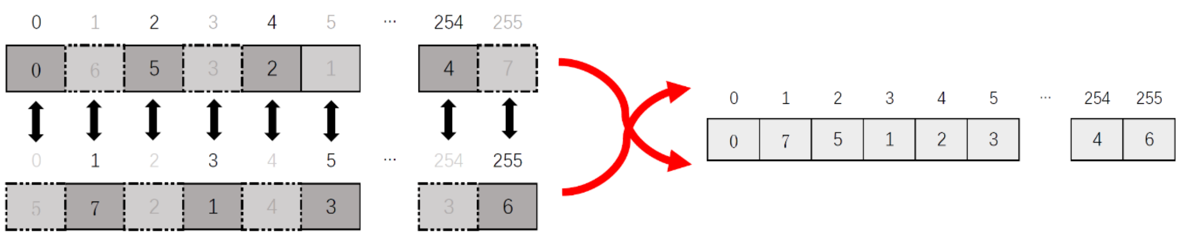
③ Compare chefs' scores within one generations

We use the best score to measure the performance of chefs of each generation, after some generations, chefs perform better. For choosing great candidates, higher scores mean higher possibility.

④ Candidates Selection for reproduction

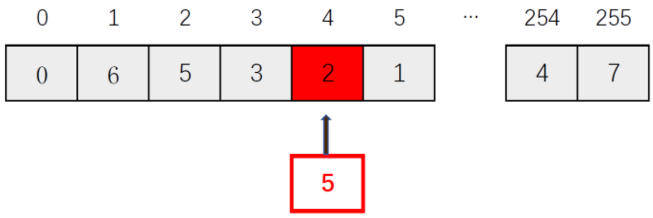
We choose the parents of the next generation by selection Function. For selecting great candidates, we will do first to screen out chefs whose scores are greater than 0 and provide them larger proportion to calculate average scores as next filter. Second, we will calculate their score percentage among total points to determine final candidates.

⑤ Crossover & Reproduction



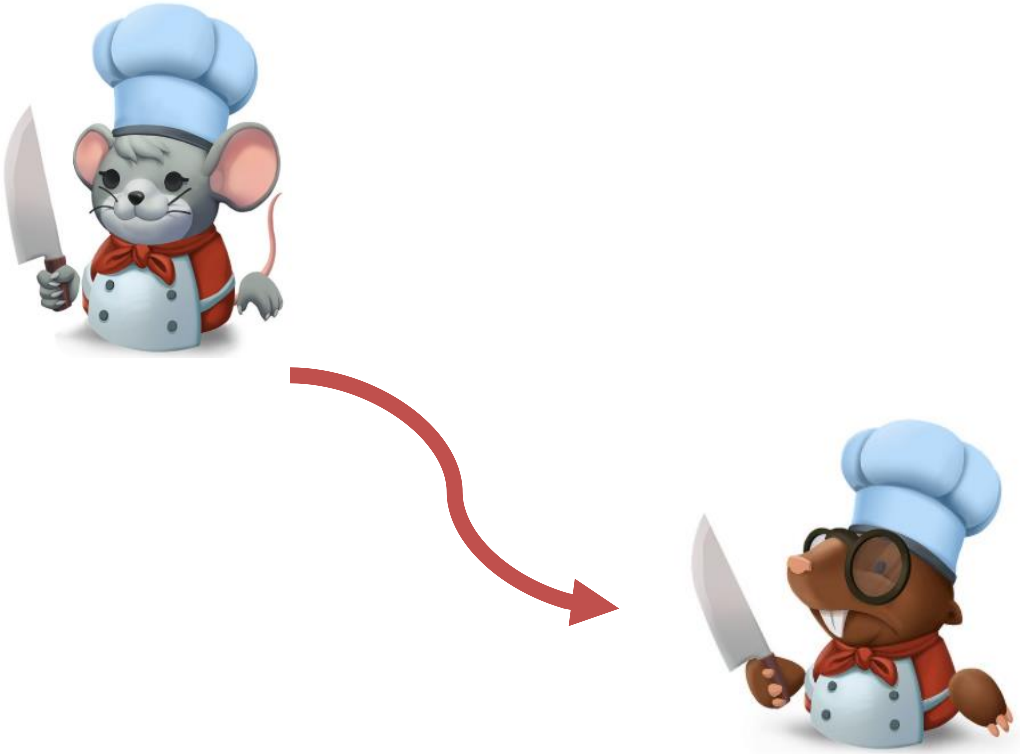
To reproduce descendants, we design to crossover genes from selection candidates. To insure the diversity of genes, the genes of these two mating individuals are different, which means we will guarantee no selfing.

⑥ Mutation



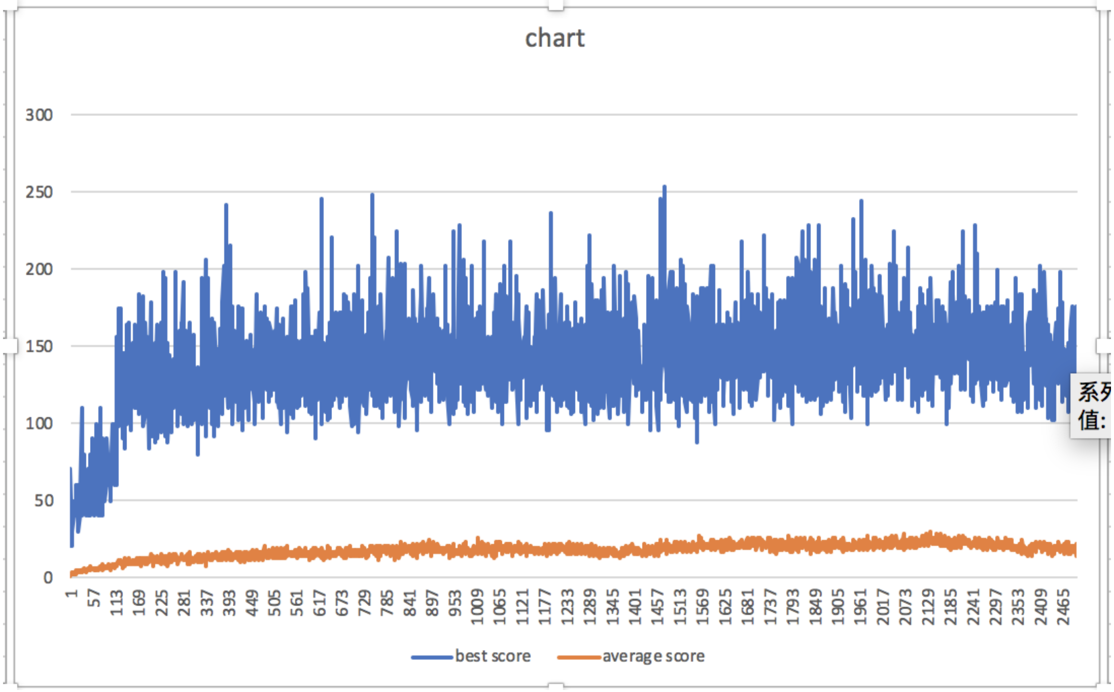
Gene mutation should be extremely rare, so we design a number named as Mutation Rate to control mutation of gene. Only if rate is greater than mutation rate, one genetic code of one individual of one generation is changed.

⑦ New Generation



Result & Conclusion

We can know the first generation had bad performance, because their genes are totally random and have small amount of good gene. Hence most of these cooks can't do a right put action, they may put fish onto shrimp position or even put foods onto empty position. However, some of the individuals can make some scores although their genes are composed by random numbers. The higher scores the cook get, the more possible the cook can survival and generate new generations, although their genes are not good enough.



From upper chart, we can notice the best score increased noticeable and average score also increased a little. But how the genes of cooks become better?

① Mutation Rate

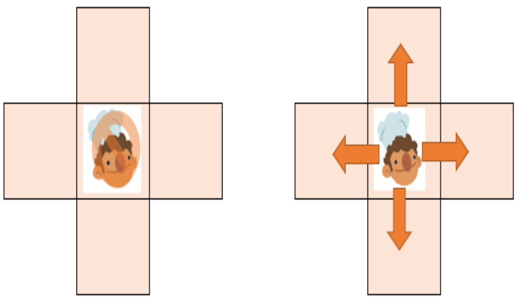
```
public class Evolution {
    public static final int GENERATION = 2500;
    public static final int ROW = 10;
    public static final int COL = 10;
    public static final int COOK_NUM = 200;
    public static final int MOVE_TIMES = 100;
    public static final double MUTATION_RATE = 0.001;
```

We define the mutation rate as 0.1%. Although it is very small, but every cook have 256 gene and every generation have 200 cooks. It is possible for every cook that bad gene mutate to a good gene, and vice versa.

② Selection parents of next generation

In our evolution class, even the cook have a bad score calculated from our fitness method, they still have chance to remain and become parent of next generation.

③ Limited direction of cook



Another reason is that cook only have 5 direction(stay, go left, go right, go upper, go down) . In this situation ,if a cook with good gene, he/she should move random direction.

But a random direction move is not a good action for cook to right place and put right food. So if the map is big enough, no matter how good gene the cook have, nothing could guarantee cook put right food on right place.

④ The performance is delivery food, not gene itself

The high score is get from deliver right food to customers. If cook move to customer give fish to customer who want fish, cook will get 30 scores. Two action could get 30 scores, so 100 actions could get 1500 scores in theory. But we can’t guarantee every action is valid. During the evolving process, the strategy in their gene is becoming better, but the map is different and even random.

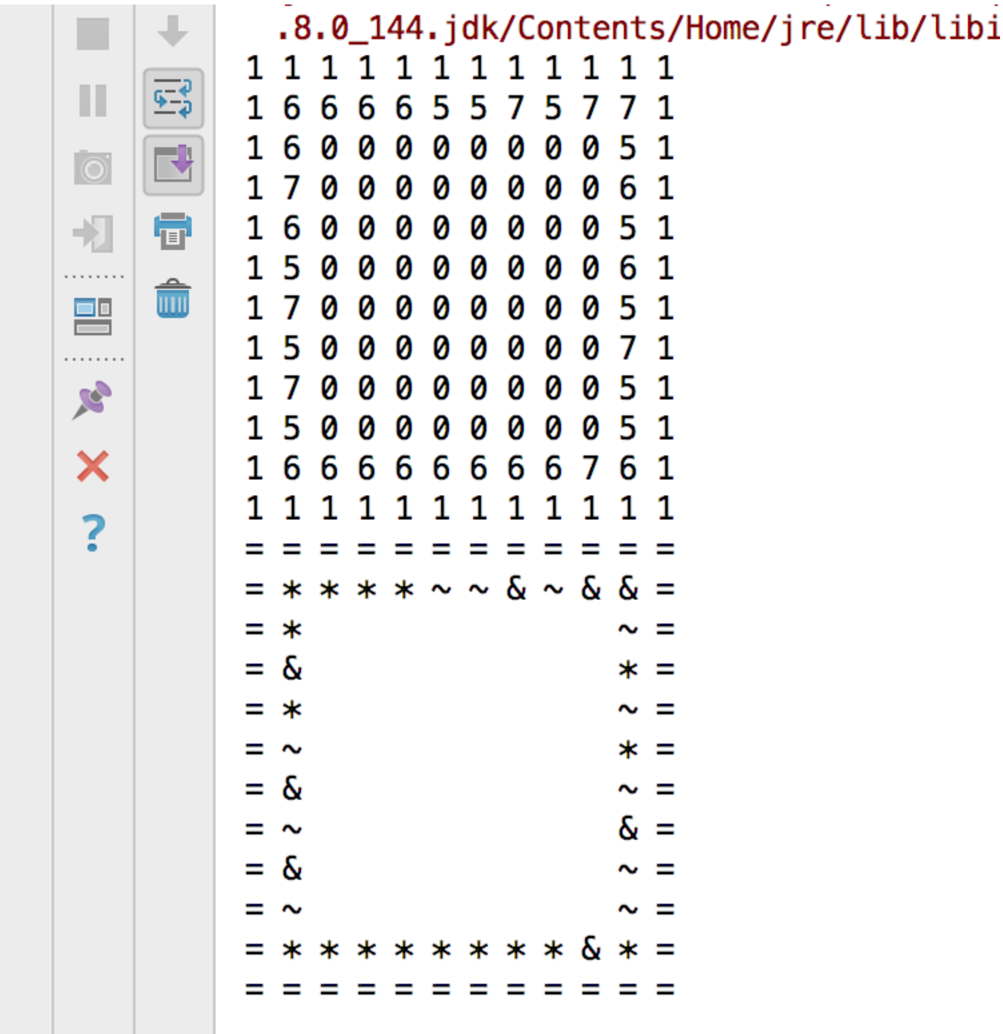
So we choose the best gene from the last generation:

```
Best Gene:
4432435410243301014243151343143355131424213542114452154220400254221131520115014251543025552510555050151432150341135344
4432435410243301014253131303143355131424213542014452154220401254221131520115014451543425552514555050151432150341135344
4130445010253504014143131043143153121224213542114452154124401254221132500314044150543520525345545310143014004213534
4430445010253504014153131003143155121223213542014452154120401254221132500114044150543520525345545310143014004213534
4432435410243301014244131343143355141324243542114452154220401254221131520115014251543425552554555050101432150041135344
4430445010253504014143131043143155121223213542114452154120401254221132500113044150543520525345545310143014044213534
4432435410243301014243131343143355131424213542114452154220401254221131520115014251543425552514555050151432150341135344
4430445010253504014143131043143155121223213542014452154120401254221132500114044150543520525345545310143014004213534
4432435410243301014243131343143355141424213542114452154220401254221131520115014251543425552514555050101432150341135344
4432435410243301014254131303143355141424243542014452154220401254221131520115014251543425552534555050101432150041135344
1433435410243301014253131303143355141420213542014452154221401254221131500115014251543420525345550501012321003411353344
4432435410243301014244131343143355141424243542014452154220401254221131520115014251543425552534555050101432150041135344
4430445010253504014143131043143155121223213542014452154120401254221132500114044150543520525345545310143014004213534
4235435410243302054255131303143355141122213240014452154220401254231431500115014251543420525345545010143214034413544
4433435410243301014243131143143355141420213542014452154221401254221131500115014251543420525345550501014321503411353344
4430445010253504014143131043143155121220213542014452154120401254221132500114044150543520525545545310143014004013534
```

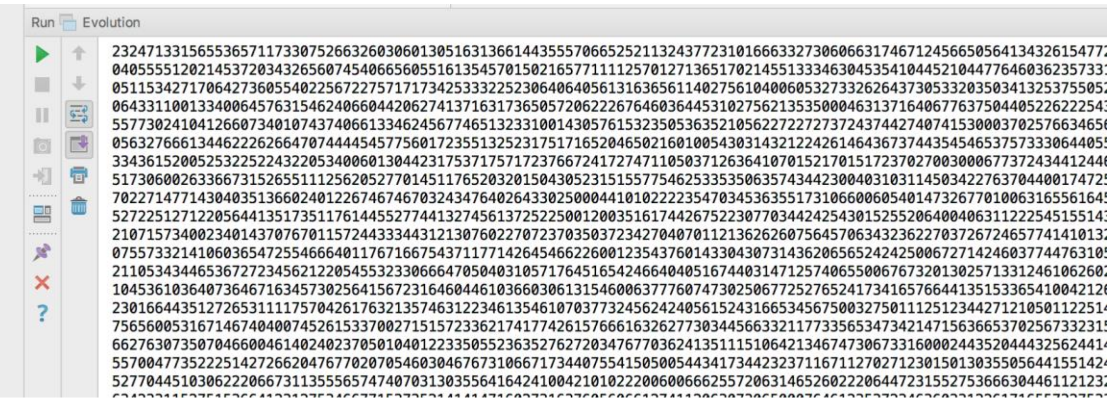
We can find most of these best gene is very similar after 2500 generation evolving.

Evidence of Running

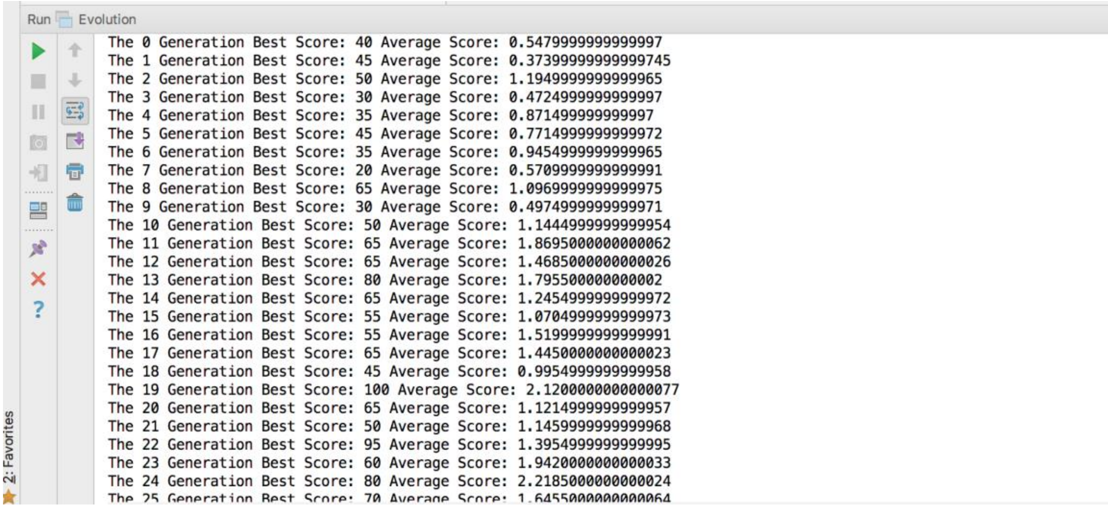
1. Restaurant initialization



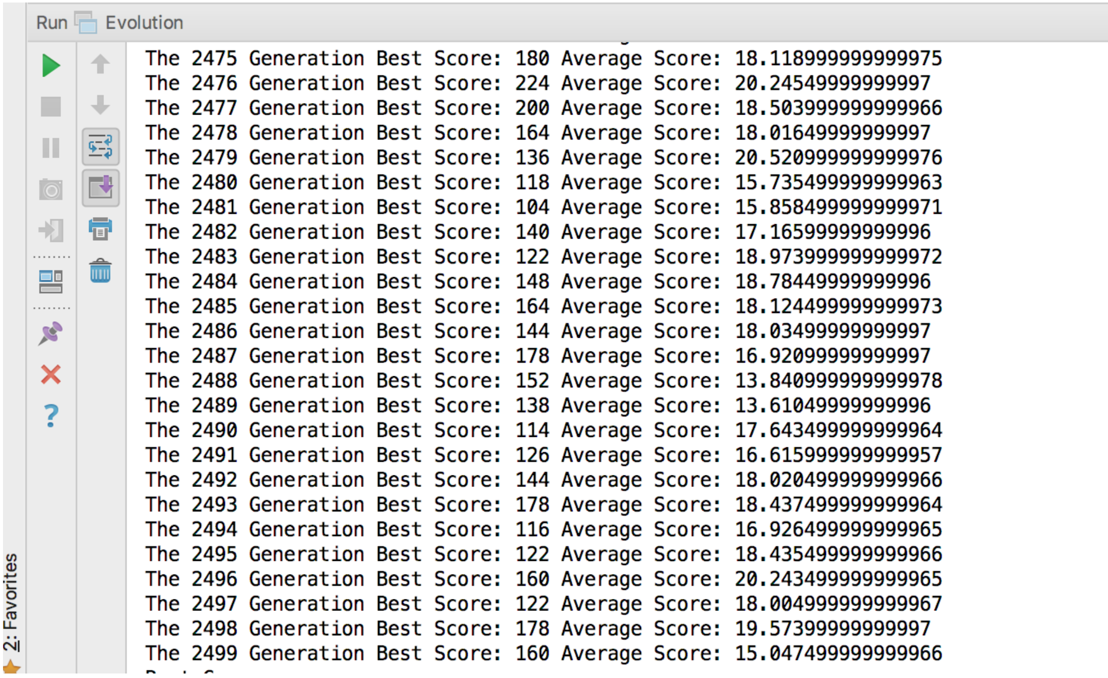
2. First generation gene initialization



3. Evolution Process



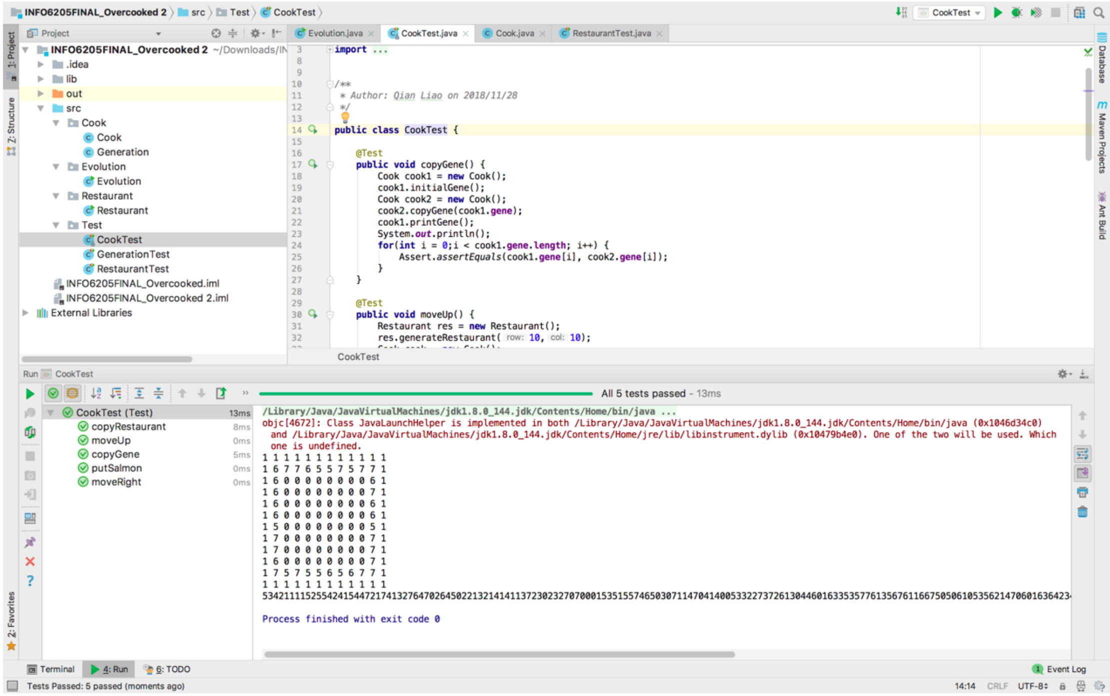
```
Run  Evolution
The 0 Generation Best Score: 40 Average Score: 0.5479999999999997
The 1 Generation Best Score: 45 Average Score: 0.37399999999999745
The 2 Generation Best Score: 50 Average Score: 1.1949999999999965
The 3 Generation Best Score: 30 Average Score: 0.4724999999999997
The 4 Generation Best Score: 35 Average Score: 0.871499999999997
The 5 Generation Best Score: 45 Average Score: 0.7714999999999972
The 6 Generation Best Score: 35 Average Score: 0.9454999999999965
The 7 Generation Best Score: 20 Average Score: 0.5709999999999991
The 8 Generation Best Score: 65 Average Score: 1.0969999999999975
The 9 Generation Best Score: 30 Average Score: 0.4974999999999971
The 10 Generation Best Score: 50 Average Score: 1.1444999999999954
The 11 Generation Best Score: 65 Average Score: 1.8695000000000062
The 12 Generation Best Score: 65 Average Score: 1.4685000000000026
The 13 Generation Best Score: 80 Average Score: 1.795500000000002
The 14 Generation Best Score: 65 Average Score: 1.2454999999999972
The 15 Generation Best Score: 55 Average Score: 1.0704999999999973
The 16 Generation Best Score: 55 Average Score: 1.5199999999999991
The 17 Generation Best Score: 65 Average Score: 1.4450000000000023
The 18 Generation Best Score: 45 Average Score: 0.9954999999999958
The 19 Generation Best Score: 100 Average Score: 2.1200000000000077
The 20 Generation Best Score: 65 Average Score: 1.1214999999999957
The 21 Generation Best Score: 50 Average Score: 1.1459999999999968
The 22 Generation Best Score: 95 Average Score: 1.3954999999999995
The 23 Generation Best Score: 60 Average Score: 1.9420000000000033
The 24 Generation Best Score: 80 Average Score: 2.2185000000000024
The 25 Generation Best Score: 70 Average Score: 1.6455000000000064
```



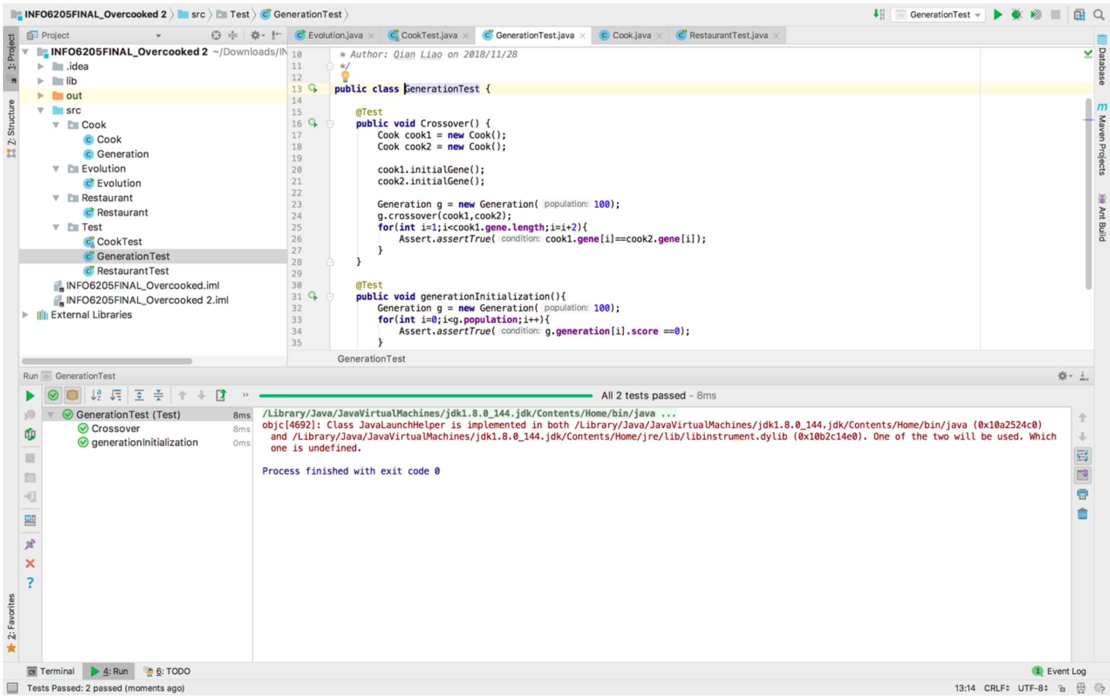
```
Run  Evolution
The 2475 Generation Best Score: 180 Average Score: 18.118999999999975
The 2476 Generation Best Score: 224 Average Score: 20.24549999999997
The 2477 Generation Best Score: 200 Average Score: 18.503999999999966
The 2478 Generation Best Score: 164 Average Score: 18.01649999999997
The 2479 Generation Best Score: 136 Average Score: 20.520999999999976
The 2480 Generation Best Score: 118 Average Score: 15.735499999999963
The 2481 Generation Best Score: 104 Average Score: 15.858499999999971
The 2482 Generation Best Score: 140 Average Score: 17.16599999999996
The 2483 Generation Best Score: 122 Average Score: 18.973999999999972
The 2484 Generation Best Score: 148 Average Score: 18.78449999999996
The 2485 Generation Best Score: 164 Average Score: 18.124499999999973
The 2486 Generation Best Score: 144 Average Score: 18.03499999999997
The 2487 Generation Best Score: 178 Average Score: 16.92099999999997
The 2488 Generation Best Score: 152 Average Score: 13.840999999999978
The 2489 Generation Best Score: 138 Average Score: 13.61049999999996
The 2490 Generation Best Score: 114 Average Score: 17.643499999999964
The 2491 Generation Best Score: 126 Average Score: 16.615999999999957
The 2492 Generation Best Score: 144 Average Score: 18.020499999999966
The 2493 Generation Best Score: 178 Average Score: 18.437499999999964
The 2494 Generation Best Score: 116 Average Score: 16.926499999999965
The 2495 Generation Best Score: 122 Average Score: 18.435499999999966
The 2496 Generation Best Score: 160 Average Score: 20.243499999999965
The 2497 Generation Best Score: 122 Average Score: 18.004999999999967
The 2498 Generation Best Score: 178 Average Score: 19.57399999999997
The 2499 Generation Best Score: 160 Average Score: 15.047499999999966
```

Unit test

1. Cook test



2. Generation test



3. Restaurant test