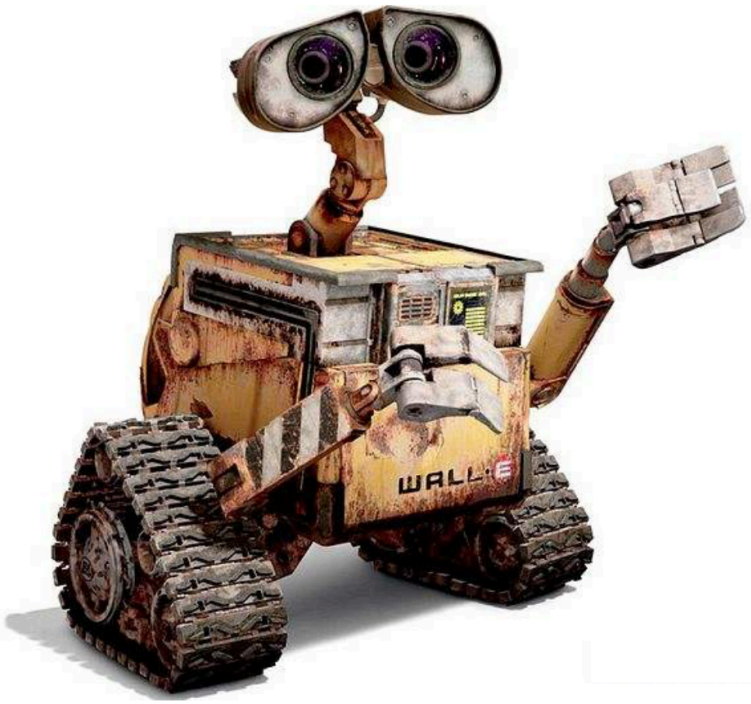
Trash Picker Robot Robby

Genetic Algorithm Problem in Melanie Mitchell's book Complexity



Ruohua Yin & Wanxin Chen

Group 513

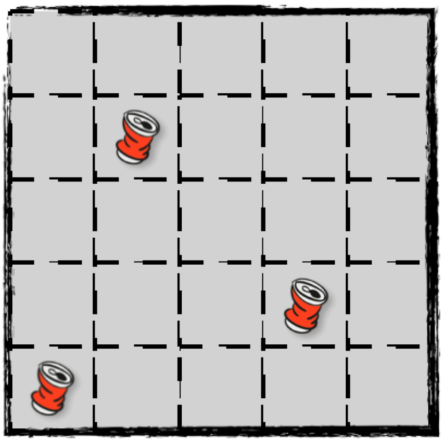
2018 Spring

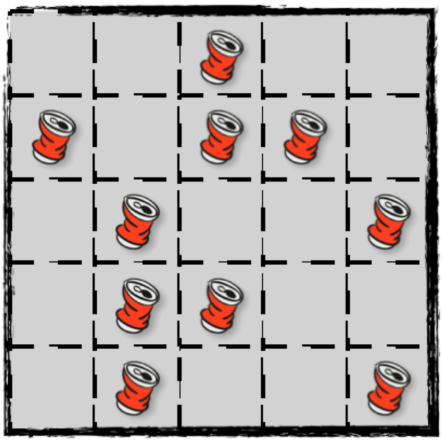
INFO6205 Final Project Report

Problem Description

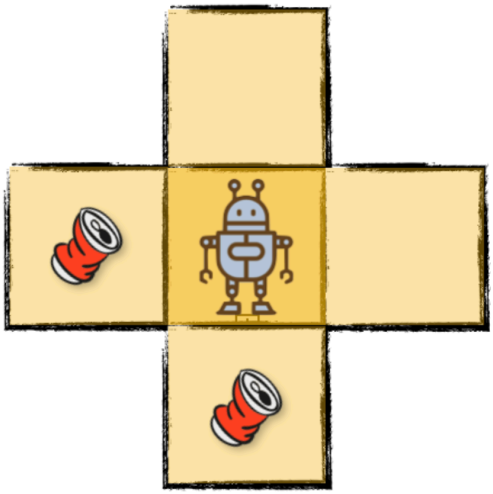
Robby is a self-directed robot tasked to pickup cans littered around a M×N board. Before the Robby can get into work, we definitely need to make rules for the work pattern he has, also the environment he is going to work on.

As mentioned before the environment we have here is a M × N board. The grids belong to it have a possibility having a trash awaited to be picked up.

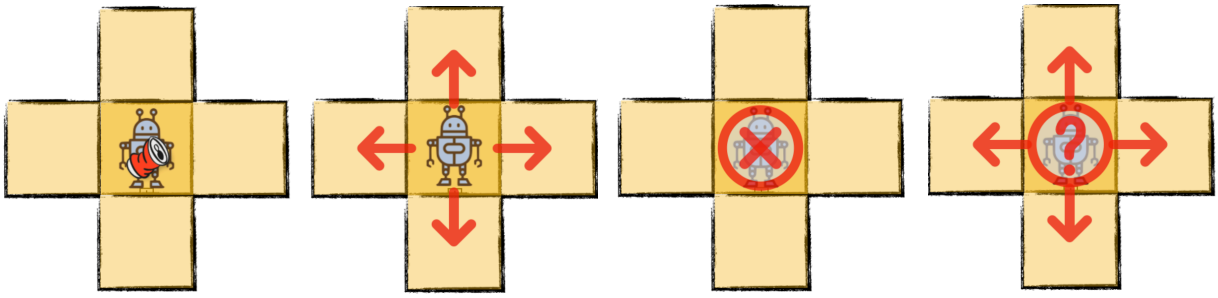




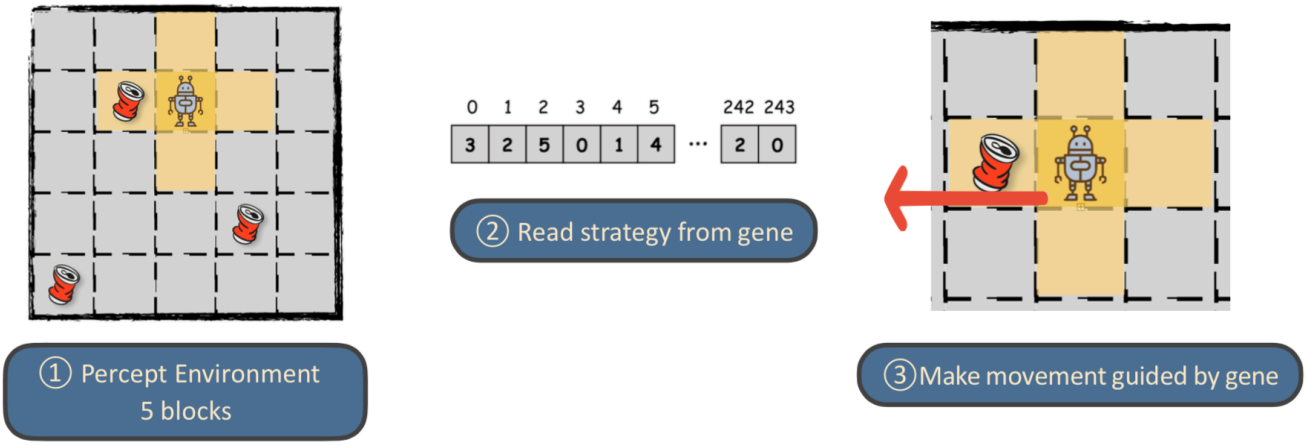
However, robot Robby is not clever and capable enough with his limited perception over 5 surrounding grid world - north, south, west, east and his original spot.



He has 7 different behaviors which are collect trash, move north, south, west, east, stay in the grid, and a random move respectively.



The working pattern is based on the different scenarios, which is written in each Robby's gene. Each time before he makes a single movement, he will percept the status of his surroundings, then read the gene inside him. Robby will follow the gene to make actions completely.



About how genotype (the order and content of genes) and phenotype (strategies which guide its movement) are designed will be introduced in details later.

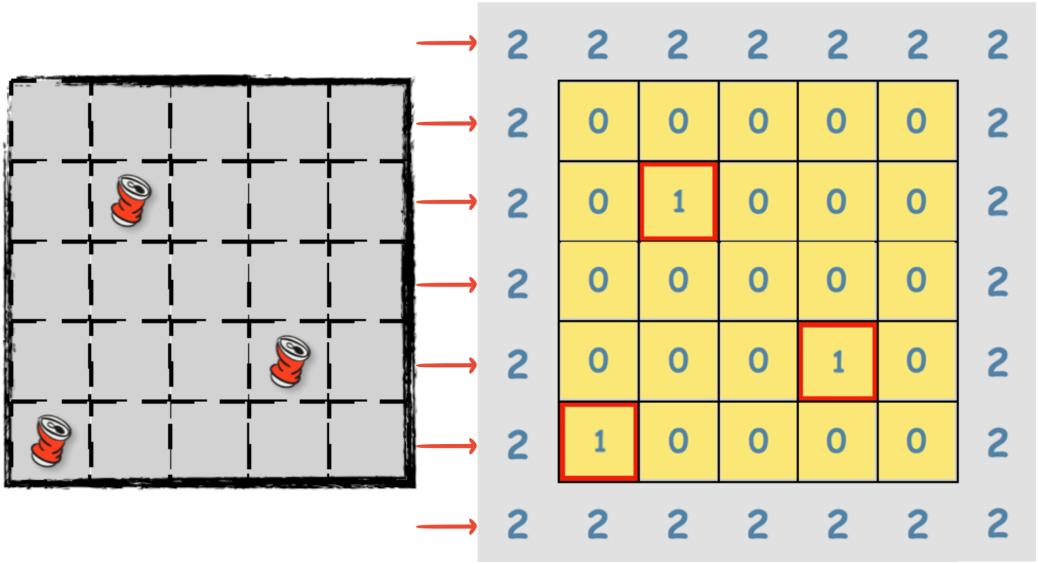
Model Detail Design

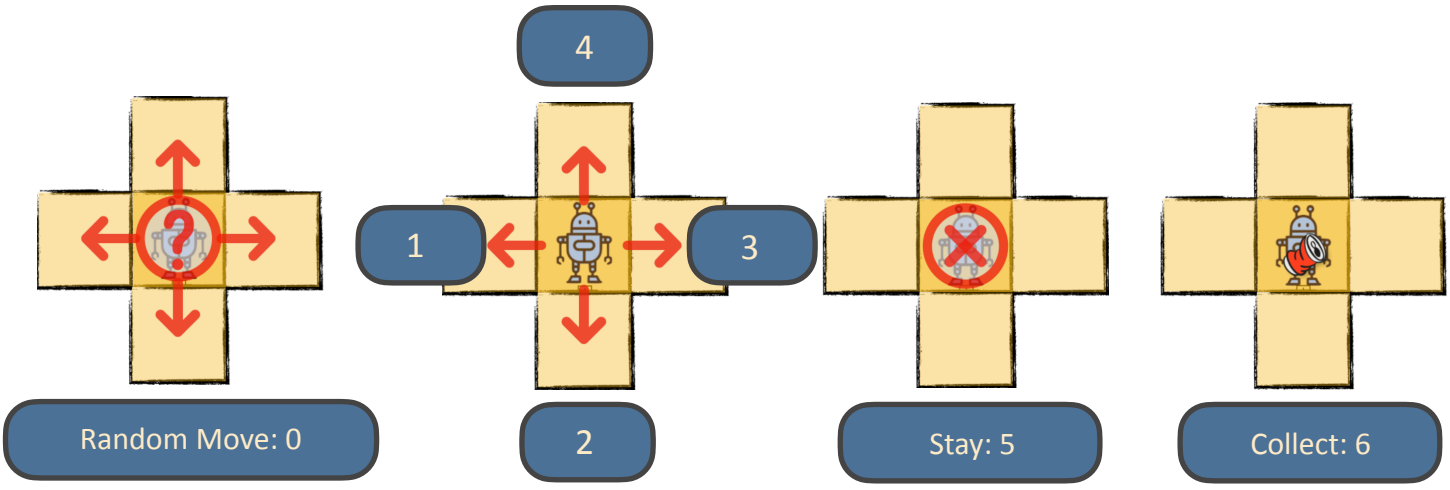
Environment Design:

There are 3 kinds of situations for a grid which are

①Grid with a trash ② Empty grid ③ Wall.

The third situation is going to be analyzed at first. The map is represented by a matrix with 3 different number inside(0 for empty, 1 for a trash, 2 for wall), but a map with M × N grids should be represented in a (M+2) * (N+2) matrix. The extra 2 for M and N is to represent wall on the 4 edges.





Robot Design:

A Robot has 7 different kinds of movement:

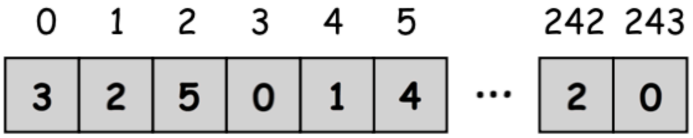
In the implement process in Java, we use the number from 0-6 to represent different movements.

```
***** 0-6 to represent Movement *****
                N:4
                 ↑
W:1    ←    CO:6,ST:5,RA:0    →    E:3
                 ↓
                S:2
** ***** ***** ***** ***** ***** **
```

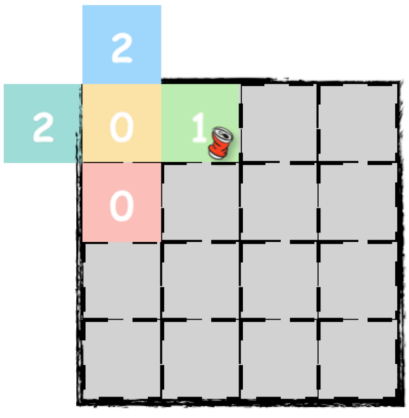
Gene Design:

1. Gene order:

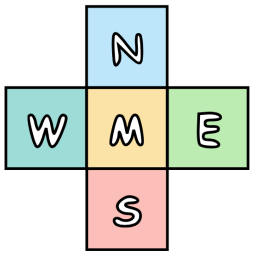
As the gene inside a robot should be able to store the strategies for 243($=3^5$) situations (5 blocks view with 3 different stuff insides). So 243 gene positions should be included for each individual.

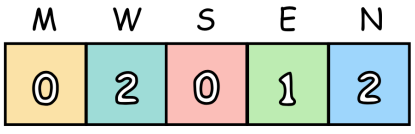


The way we decide the position of distinct strategies is a Ternary numeral system.



One situation Robby will face



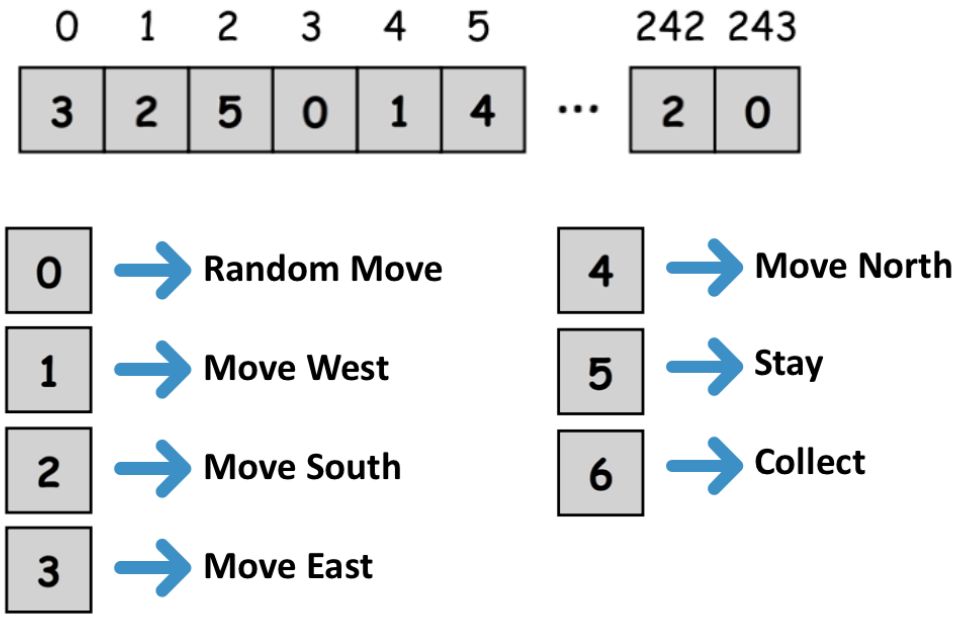


$3^4\times0+3^3\times2+3^2\times0+3^1\times1+3^0\times2 = 59$

The strategy should be stored in position 59

Note that there are some gene positions don't make sense. Like a robby cannot stay on the wall, hence the middle block represented by 2 won't come out for a robot, but it doesn't matter because even for human, there are some genes that have nothing to do with the environment. These kinds of genes will do mutation and crossover following the reproduction rules, while actually the environment won't select them at all.

2. Genotype and Phenotype Design:



Fitness Function Design:

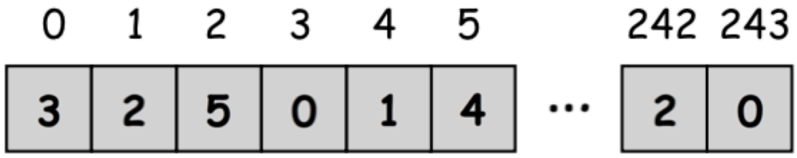
- Pick up trash successfully → +10 scores
- Pick up trash in an empty grid → -2 scores
- Crash on the wall and bounced back → -5 scores
- Move or Stay → Nothing changes

Evolution Process

Initializations:

1. Initialization of individual:

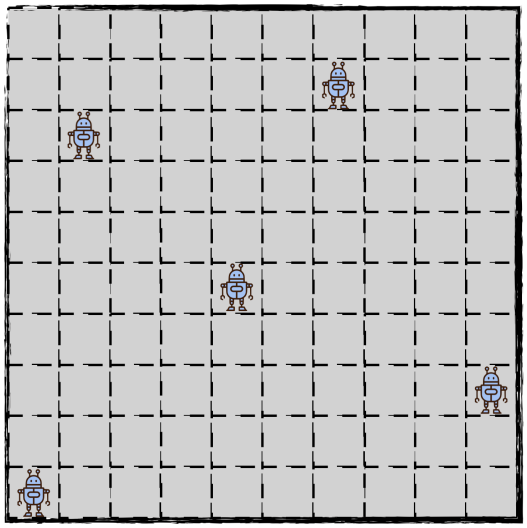
We set up the population of each generation, which is 200. The 243 genes for each individual are inputted randomly from 0 - 6.



2. Initialization of the map:

A map is the environment for a individual to gain scores. We initialize the environment to be a 10×10 board, each grid has the 50% possibility for having a trash. Before each individual gets into the game, the map will be initialized.

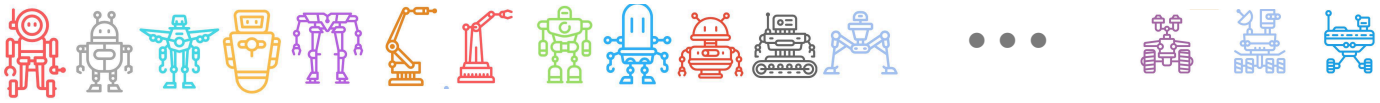
3. Initialization of the position:



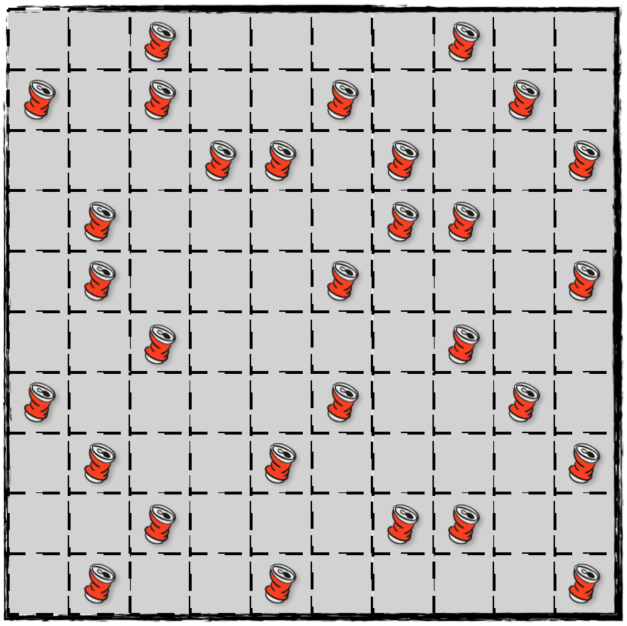
Robby will get a position in the board with random X and Y axes.

You may have questions here if we initialize the map and each individual's position, it may force the individuals to play different games.

But what we can tell it is that it doesn't matter for the individuals to have different starting point or play on different maps. For the reason that this is only an evolving process, not a real match. The grid in the board have the same possibility to have a trash. The distinct starting point and map may let them face a different situation at the beginning of each game, which means as different select direction. If each Robby start at (0, 0), the situation could be too similar. It may raise the selection speed on the gene on particular position, which is unwilling to be seen.



First Generation Initialization



Play a Game of Trash Picking

Each individual in a generation is put into the map then play the trash picking following their strategy guided by their gene. The scores they gain are calculated by the fitness function we designed.

Fewer they crushed on the wall, made pick action in an empty grid and more correct pick on the trash, they will definitely perform better in the game.

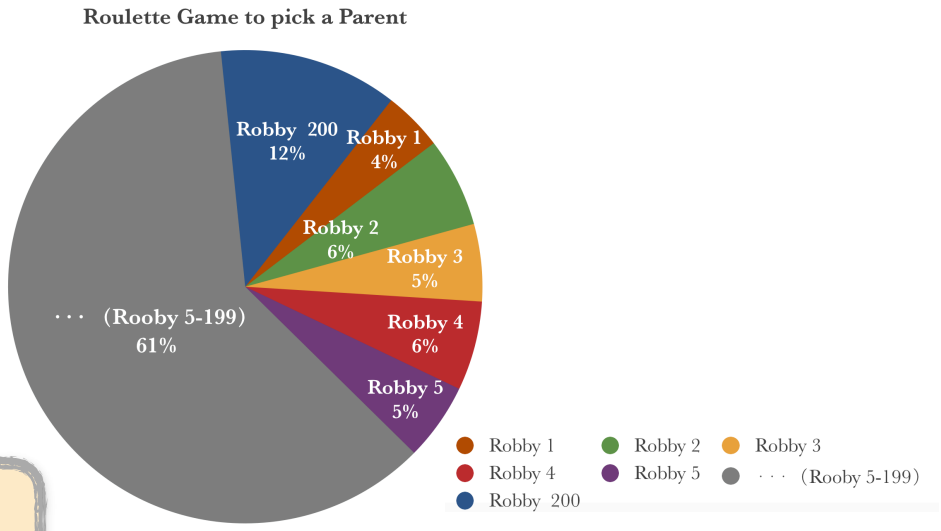
Rank the Scores

The individual with a higher score (which is calculated by the fitness function) has a larger possibility to be selected as a parent, which means a bigger chance to propagate its genes.

Selection → Roulette

Selection → Roulette

We choose the parents of the next generation by playing Roulette game. The ratio of area they have in the circle is decide by the score percentage among total points of the generation.

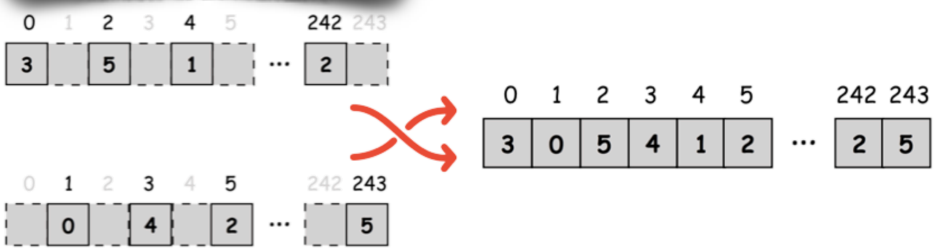


Reproduction

The reproduction pattern we designed for the robot is they have both asexual and sexual ability. Higher the sexual reproduction rate, faster the evolution process the generation has.

Asexual 10%

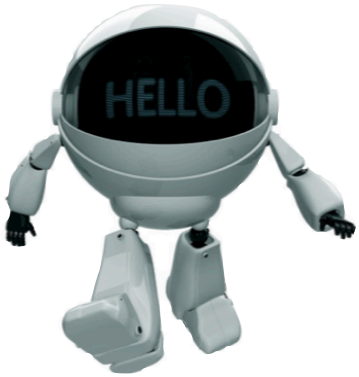
Sexual 90%



Mutation

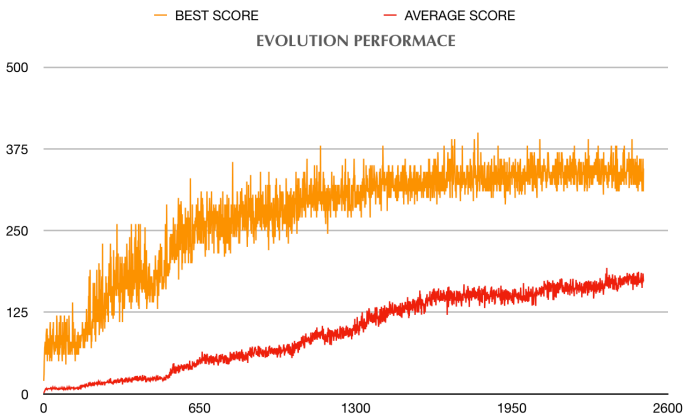
Before the new individual is being made, each gene out of 243 has a 0.05% possibility mutating to a new genetic code from 0-6.

The New Generation



Results & Conclusions

It can be predicted that the first generation could perform really bad because they have such a small amount of good genes generated by list of random number. Hence much of them couldn't make a right pick up action even if they have a trash in their current grid. And So much of them ended their game by make a crush action to the wall continuously because so long as the situation in the 5 block doesn't change, they will make the same action by reading the same gene. However, some of the individuals can make some scores although their genes are composed by random numbers. Higher the score they have, higher the chance they get to propagate their genes although their genes are still not good enough as initial generations. The trend is in the direction of a increasing performance on the game, their genes which hold a good strategy will transfer to their child increasingly.



The left diagram is the result came out from one of the evolving process. Both the best and average performance in every generations continuously made progress and converge to where theocratically the best point. But what is the reason why they didn't converge to the exact point which is the best performance.

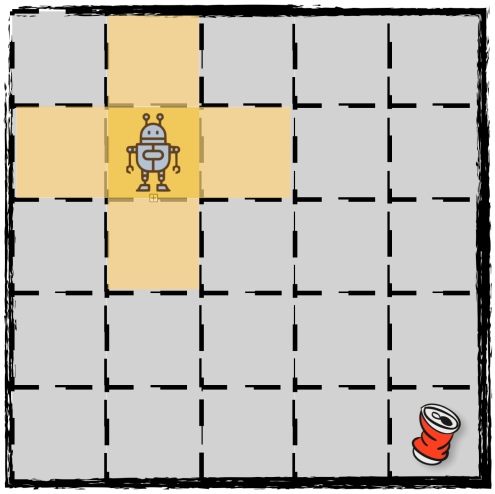
① Mutation Rate and Mortality rate

```
public static int EVOLUTIONGENERATION = 2500;
public static int ROW = 10;
public static int COL = 10;
public static double POSSIBILITY = 0.5;
public static int POPULATION = 200;
public static int MOVETIMES = 100;
public static double ASEXUALRATE = 0.1;
public static double MUTATIONRATE = 0.0005;
```

The former screenshot is the configuration of parameters for the evolution process. Although the mutation rate for each gene is only 0.05%, there are 243 genes for each of 200 individuals in 1 generation, so there will always have a group of genes change from the old to new one. The bad performance genes have chance to become a good one, in the mean time, a selected good gene have chance to become a bad one randomly.

And in the evolution patter we designed, even the individual have a low score calculated from the fitness function, they still have the chance to be selected as a parent. So eliminate all individuals with a bad gene could be a much longer process than 2500.

② It's limited by the nature of the robot view



Another reason for the gap between is that Robby only has a view for 5 blocks. If you take a look at the situation which is shown left, you got to understand. In such situations, a robot with a good gene would move to a random direction. But a random move is obviously not a good news for him to pick up a trash out of his sight. If the map is large enough, no matter how good the gene inside him is, nothing could guarantee Robby to find a trash which is distant to him.

③ The Performance here is trash picking score, not gene itself.

What theocratically the best is the robot made movement like, he can pick up a trash every time he makes a move to a direction. Every 2 actions made 10 scores. So 100 action can make 500 grades. But it's not a guarantee that the grids with a trash is linked with each other.

During the evolving process, the strategy stored in their gene will become better and better, but the map is different and generated randomly.

So, we try to extract some of the gene on typical position of the individuals with the best performance in the final generation. As is printed below, no more than 20 individual's first 60-70 genes there are. We can conclude that different individuals have most of their gene similar after 2500 generation evolving. We can collect some critical gene and use statistic method to see how good they are being trained.

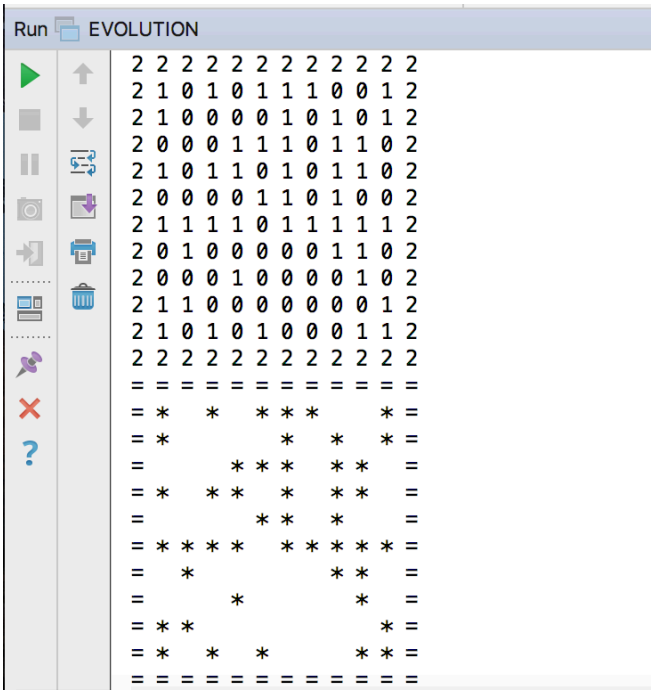
```
Best Gene:
042303401240301202452010150112140104131321222043304030002332006342333145330033126666636666666362046642062
042303401240301202452010150112140104131321222043304030002332006342333145330033126666636666666362046642062
042303401240301202452010150112140104131321222043304030002332006342333145330033126666636666666362046642062
042303401240301202452010150112140104131321222043304030002332006342333145330033126666636666666362046642062
042303401240301202452010150112140104131321222043304030002332006342333145330033126666636666666362046642062
042303401240301202452010150112140104131321222043304030002332006342333145330033126666636666666362046642062
042303401240301202452010150112140104131321222043304030002332006342333145330033126666636666666362046642062
042303401240301202452010150112140104131321222043304030002332006342333145330033126666636666666362046642062
042303401240301202452010150112140104131321222043304030002332006342333145330033126666636666666362046642062
042303401240301202452010150112140104131321222043304030002332006342333145330033126666636666666362046642062
042303401240301202452010150112140104131321222043304030002332006342333145330033126666636666666362046642062
042303401240301202452010150112140104131321222043304030002332006342333145330033126666636666666362046642062
042303401240301202452010150112140104131321222043304030002332006342333145330033126666636666666362046642062
042303401240301202452010150112140104131321222043304030002332006342333145330033126666636666666362046642062
042303401240301202452010150112140104131321222043304030002332006342333145330033126666636666666362046642062
```

Situation					
Gene Position	**0**	**81**	**3**	**59**	**30**
Description	5 blocks empty	A trash in the middle	A trash in the right grid	On the left and up most corner, a trash in the right grid	1 trash in the right grid, 1 trash in the left grid
0 Random Move	99.5%	0%	0%	99.5%	0%
1 ← Move Left	0.05%	0%	0%	0.5%	99.5%
2 ↓ Move Down	0%	0%	0%	0%	0%
3 → Move Right	0%	0%	100%	0%	0.5%
4 ↑ Move Up	0%	0%	0%	0%	0%
5 Stay	0%	0%	0%	0%	0%
6 Collect	0%	100%	0%	0%	0%

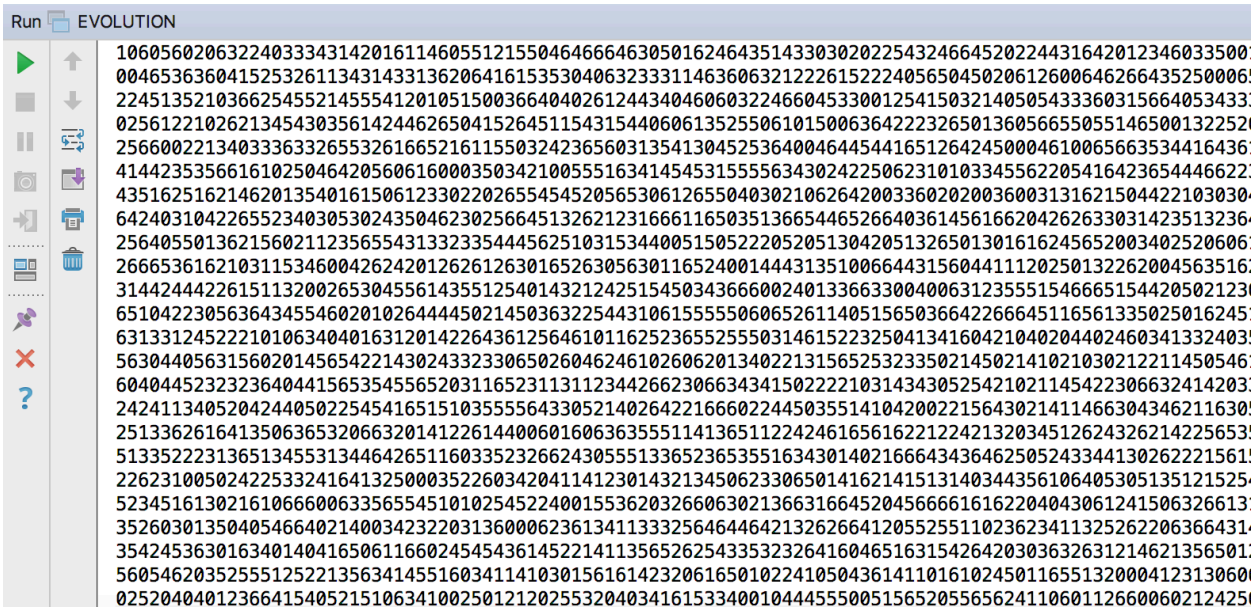
Evidence of Running

Program Outputs:

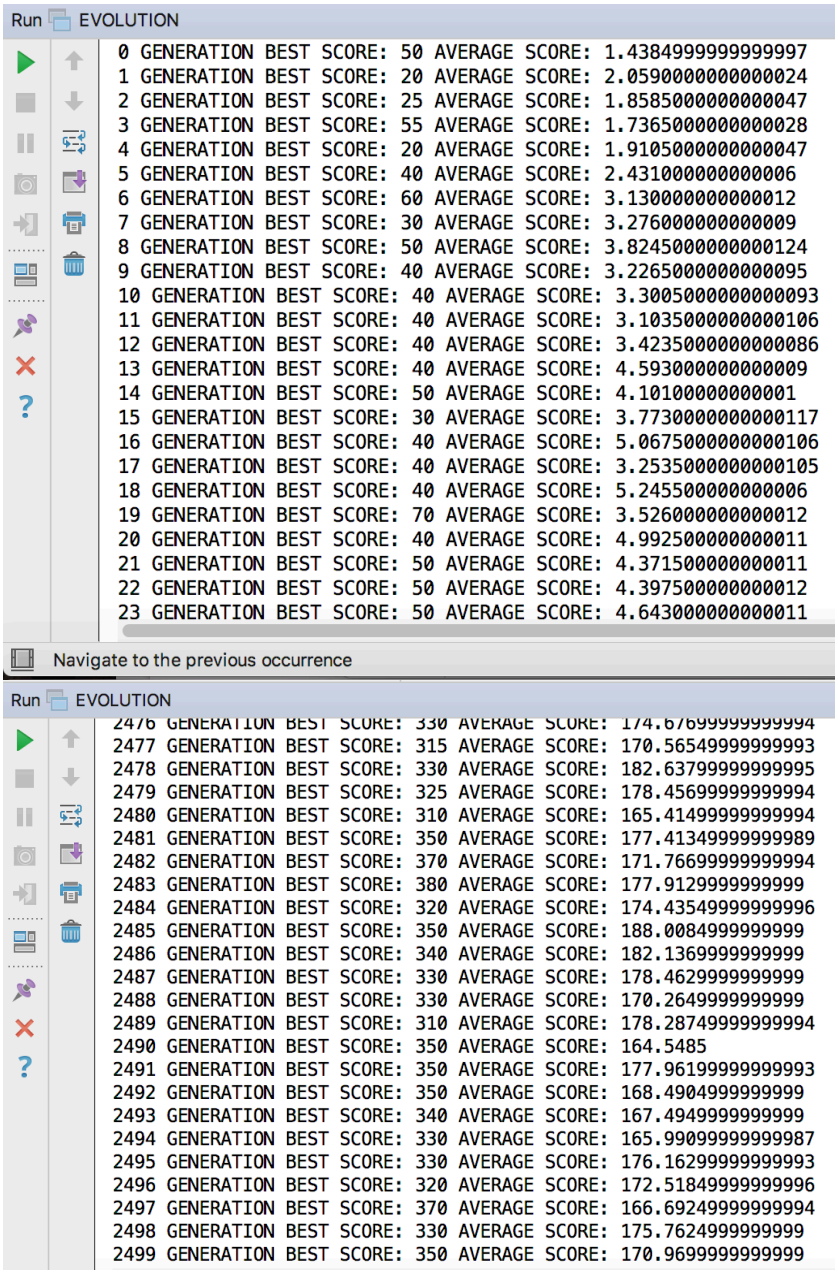
1. Map Initialization



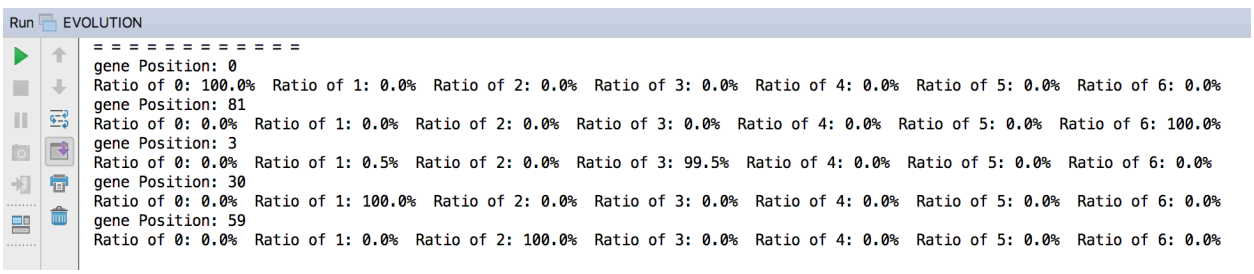
2. Genes Initialization for the first generation



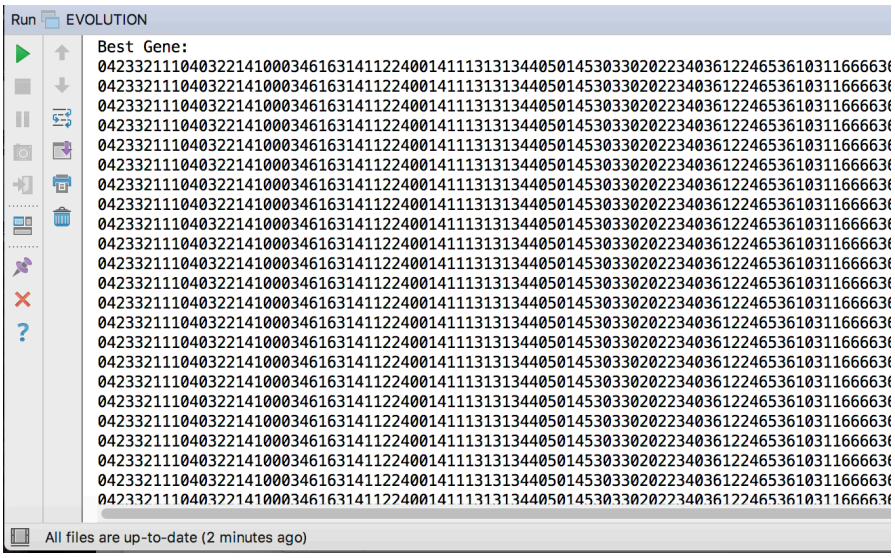
3. Evolution Process



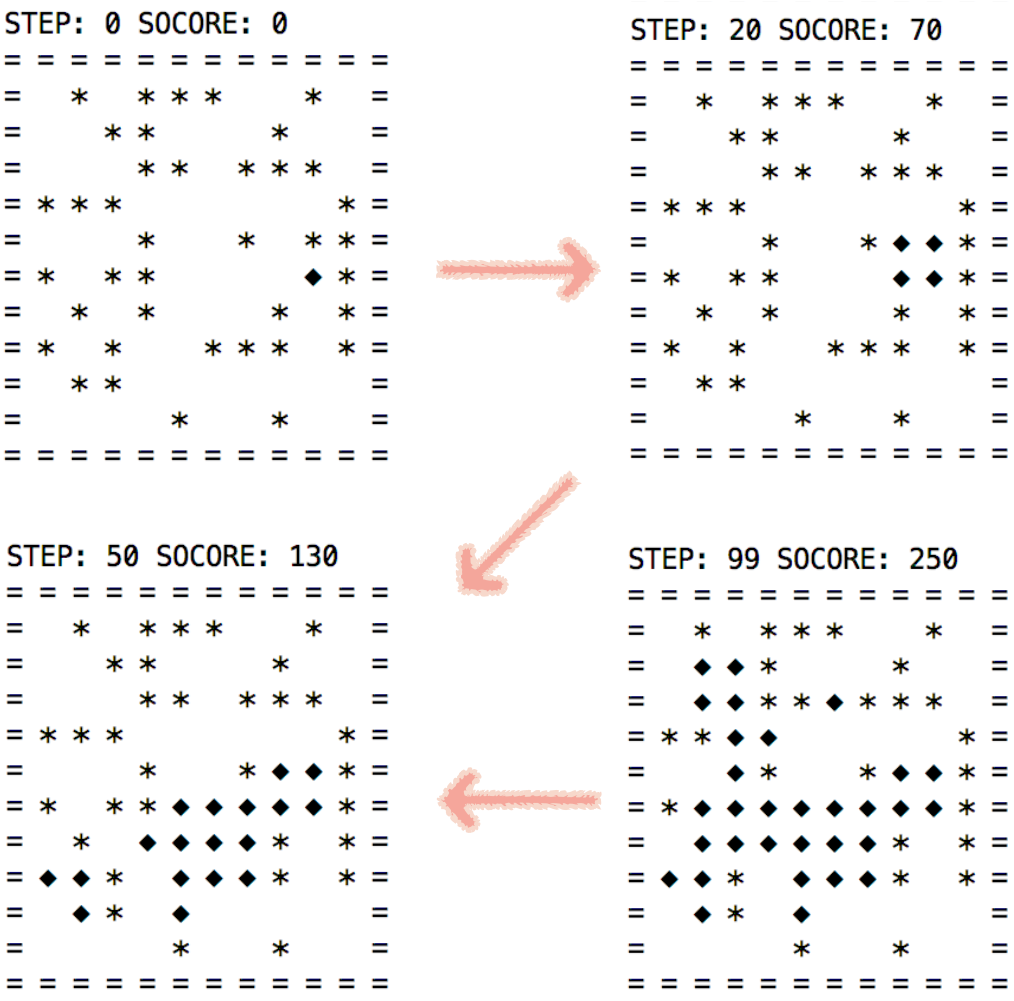
4. Statistic Results to Create Genes



5. Best Gene

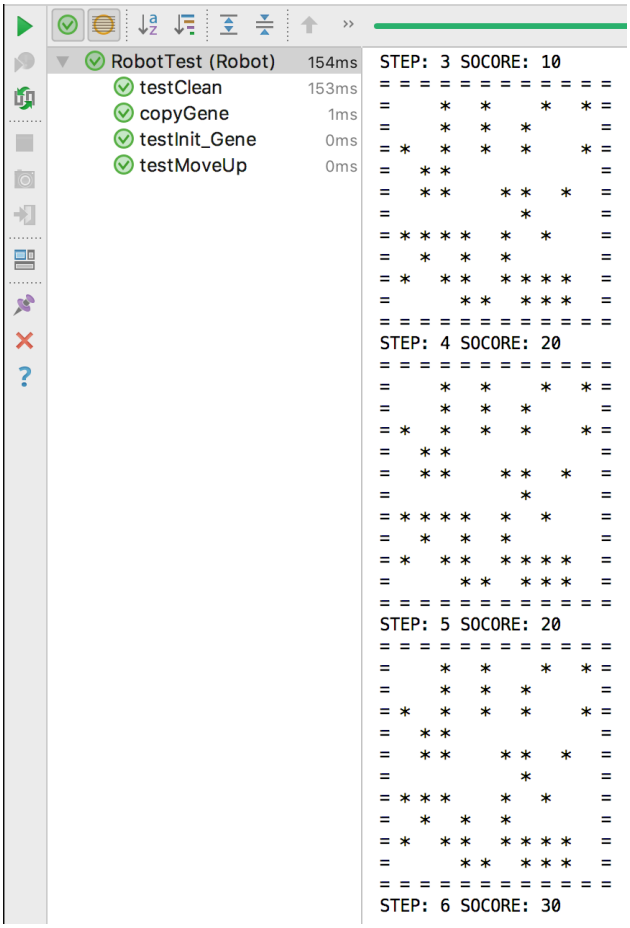


6. Route Records

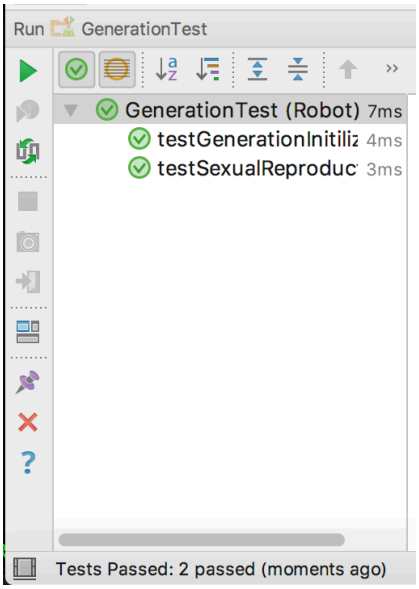


Unit Tests

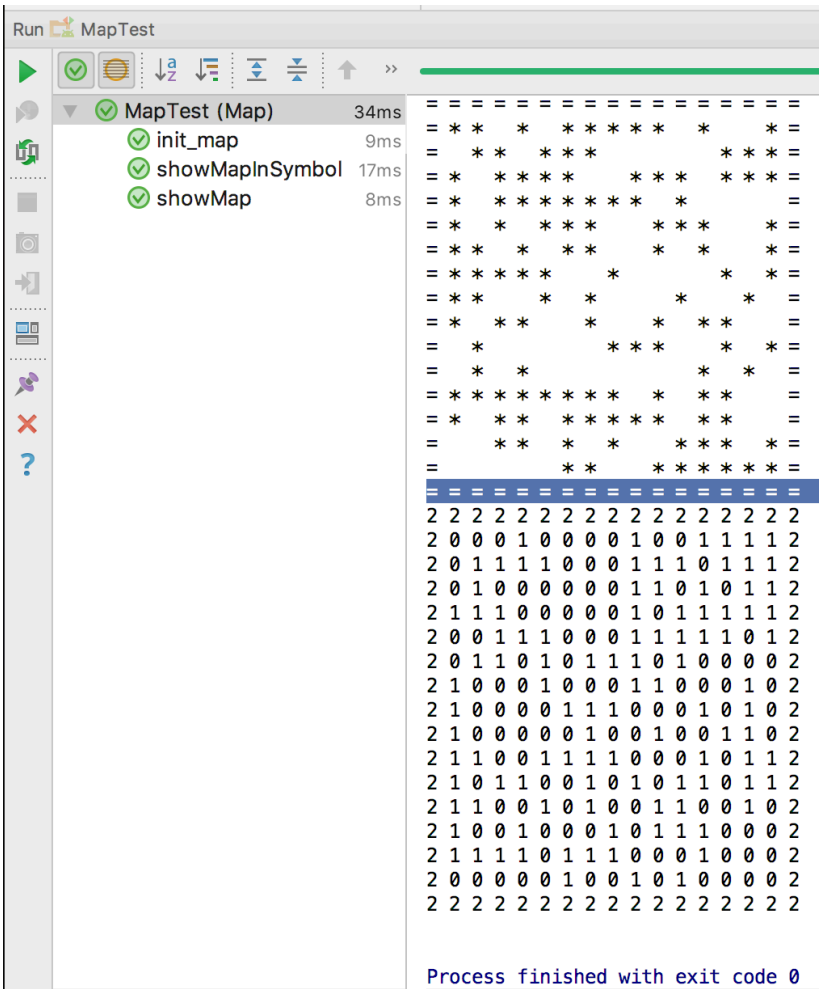
Robot Tests:



Generation Tests:



Map Tests:


Run MapTest
MapTest (Map) 34ms
init_map 9ms
showMapInSymbol 17ms
showMap 8ms
Process finished with exit code 0