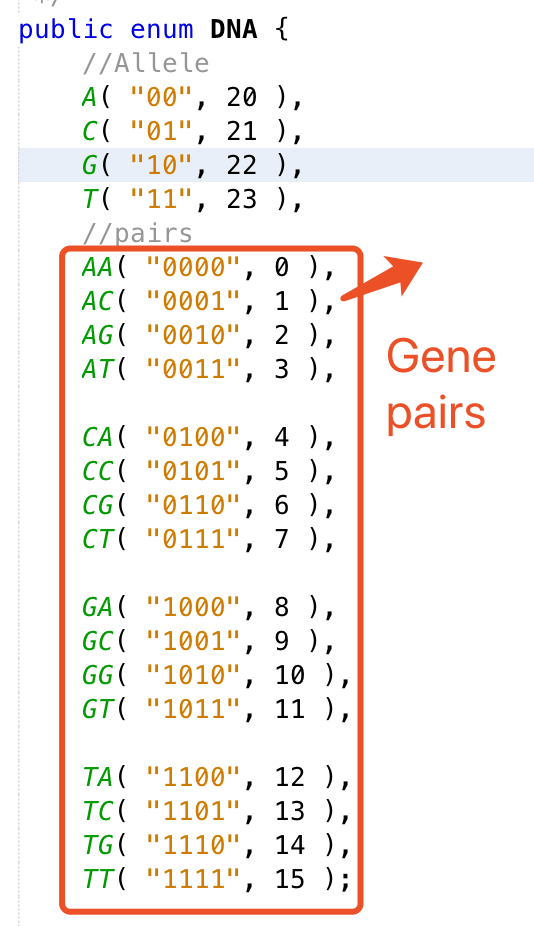
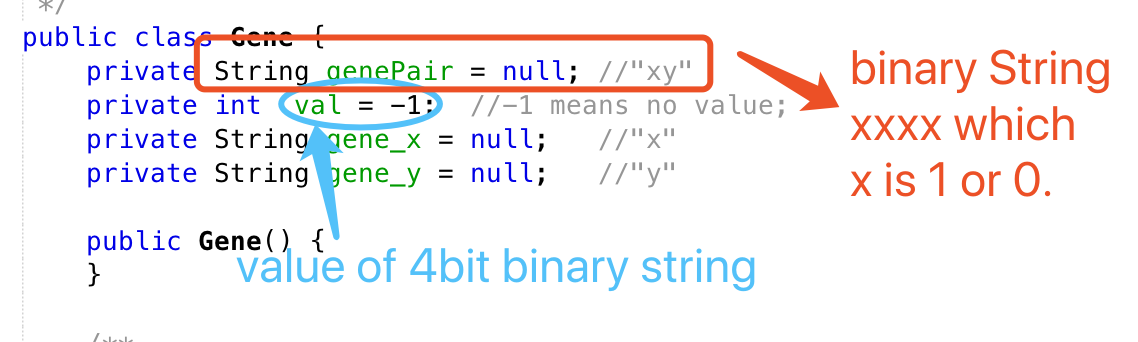
INFO6205 Final Project\_No.09 Bo Han (001815357) & Junyuan Gu (001825583)

Problem: Travelling Sales Problem, find a shortest cycle from a designated city node.

1. We use “ACGT” to implement a genetic code, where ‘A’,’C’,’G’,’T’ are alleles and there are 16 kinds of gene combination(thus use 4 bits binary String as key). The genotype is 16 bit width, which stands 4 gene pairs. Because we use 16 bit genotype to stand for city code( assuming 100 cities, the city code is from 0 to 99), the most 8 significant bits are always zero.



2. Gene expression: We use 4-bit binary String as well as integer value to stand for Gene, therefore 4 Genes ( 16bit binary String )stands for the code of a city node(from 0 to 99).



3. Fitness function: we use Position(x, y) thus compute the distance between two cities. Of course the fitness is the inverse of the distance. And “coor.txt” is the initialization data of city positions.

4. About sort function: we provide quicksort and mergesort. Actually we use mergesort() method.

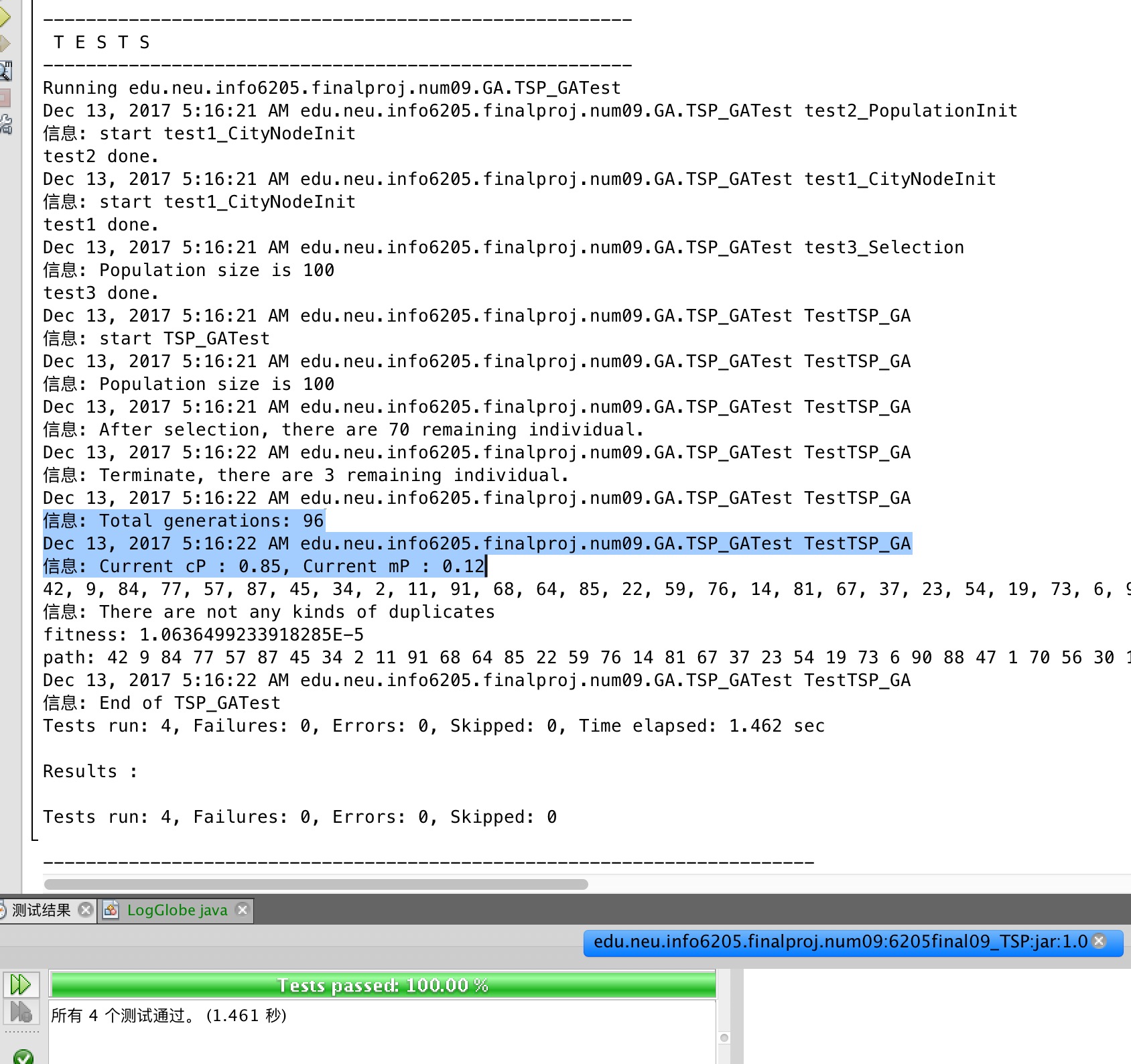
About log: we use java.util.logging.Logger to log the unit test.

About unit test: we test initialization of city and Population, selection of Population respectively and the whole evolution.

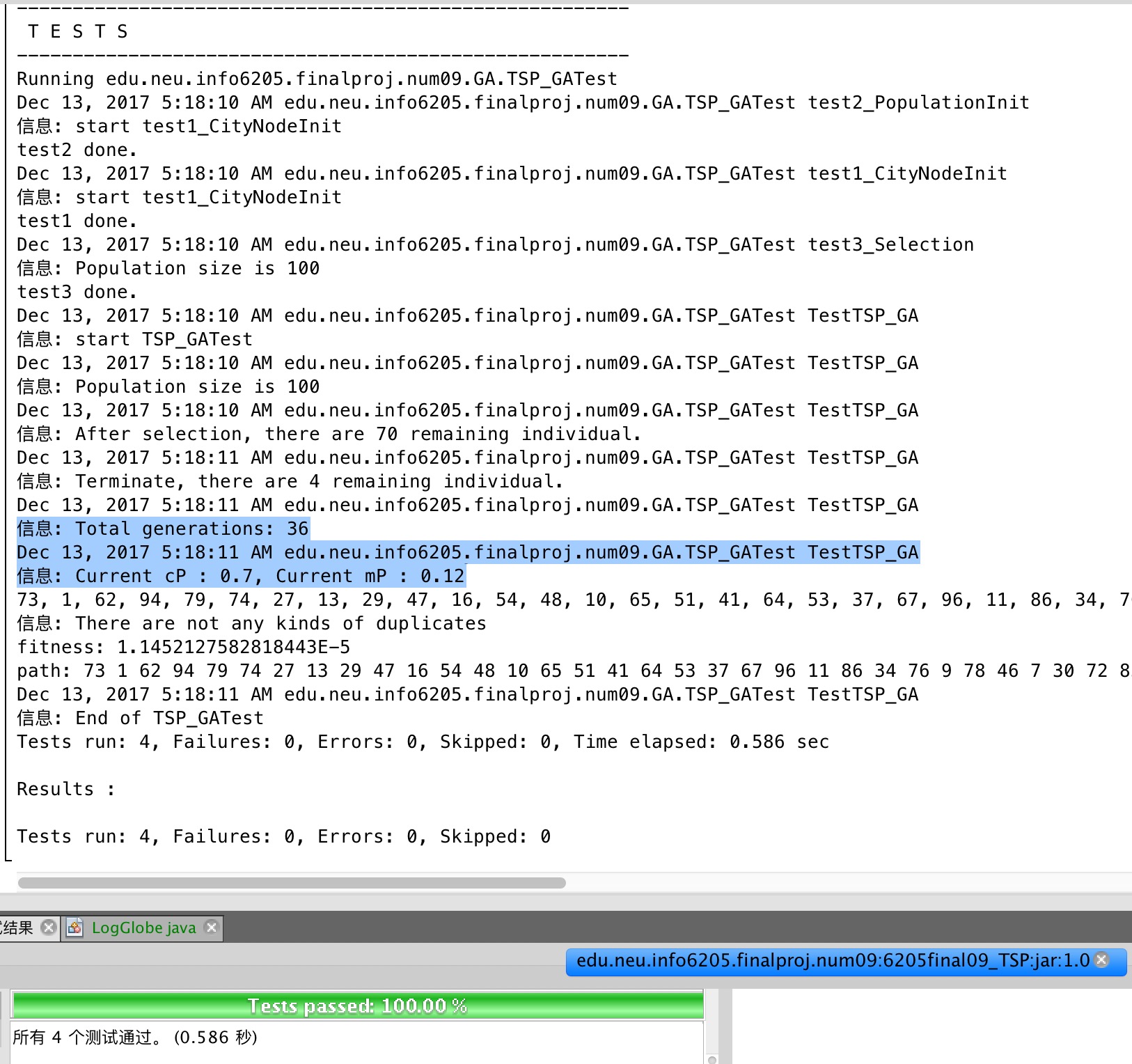
5. Findings: ( our city nodes are 100, and initial population size is 100, and we cull the 30% worst case using sort by fitness every generation.)

a. when the cP(crossover Probability) is less than 0.75, the whole population size will converge quickly(the iteration is about 30 to 38 times). In this case we only get quite a small number of possible good solutions. The solution space is huge since there are 100 city nodes.

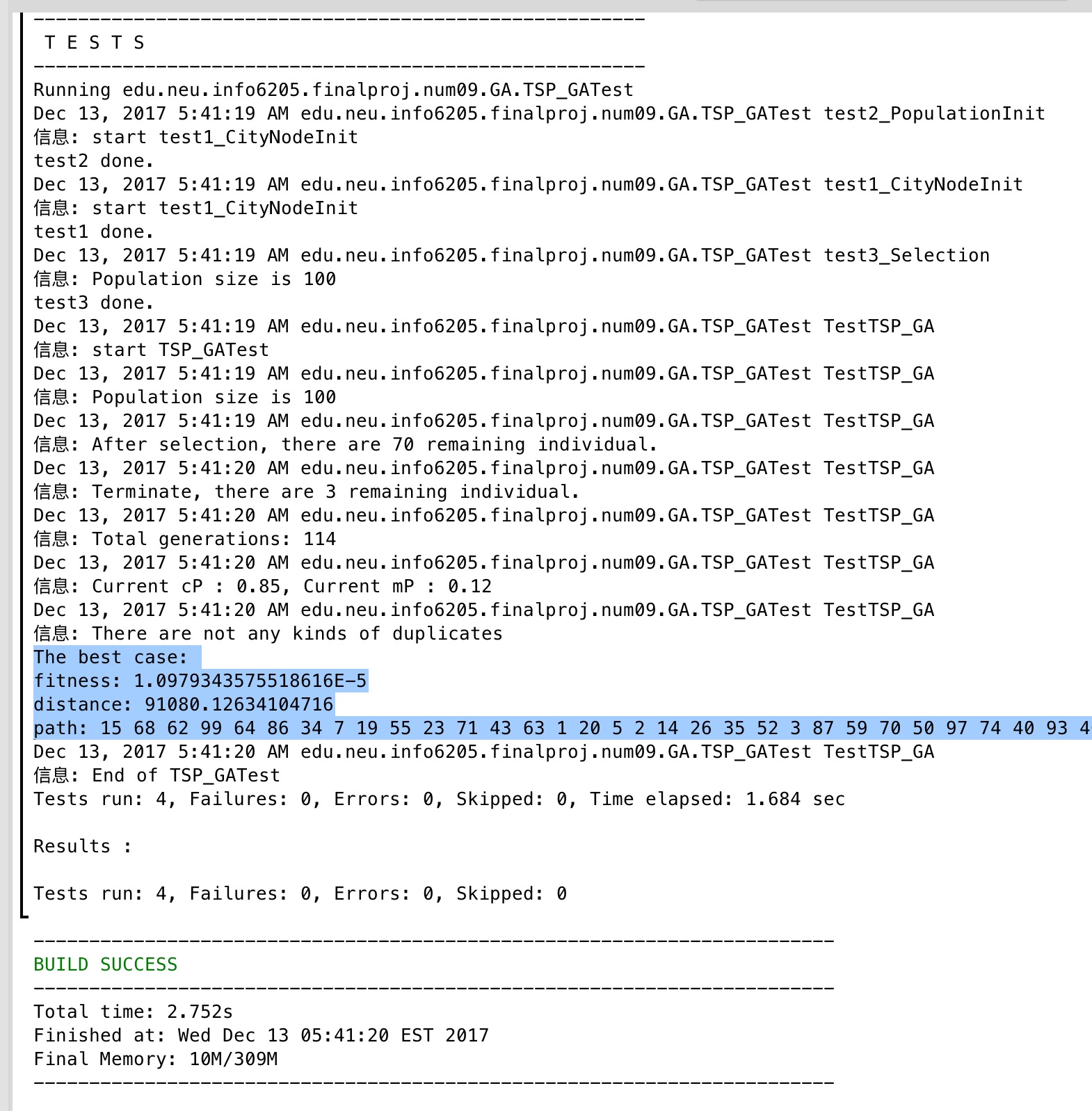
Current cP: 0.85, mP:0.12



Current cP: 0.7, mP:0.12



Best case:



b. When the cP is about 0.85, the convergency speed will be slow(the iteration is about 100 to 130).

c. When we reduce the city node to 20,

6. About Crossover solution: Firstly, we stuck at this method, we could not perform a precise way to identify the duplicate CityNodes. Finally, we find solution as follows:

A. Two individuals send to crossover(), method, use mother to initialize this child.

B. Generate two random indexes to indicate the start and end point

C. Copy the Chromosomes between start point and end point from father.

D. We implement additional operations, actually the extra ArrayList and contains() method to check the duplicate chromosomes.